STATA TECHNICAL BULLETIN

January 1997

STB-35

A publication to promote communication among Stata users

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

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

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

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

Contents of this issue

crc45	New options for survival-time data

Two new options have been added to `stset`: `first` and `force`.

Syntax

The syntax of the `stset` command is

`stset` *timevar* [*failvar*] [*weight*] [`, id(`*idvar*`) t0(`*entrytimevar*`) first force gent0(`*newvar*`) noshow `]

The first option

`first` specifies that in time-varying data (data for which `id()` is also specified), observations after the first failure event are to be ignored and, in fact, deleted from the data.

`first` is useful when you have multiple outcome variables. For instance, consider the following bit of data:

```
. list patid time r1 r2 if patid==123
        patid       time         r1         r2
  1.      123          1          0          0
  2.      123          3          1          0
  3.      123          7          1          0
  4.      123         20          1          1
```

`r1` and `r2` are two different outcome variables and assume you intend to analyze them one at a time. Say you wish to analyze outcome `r1` and that the corresponding event can occur at most once. In that case, the value of `r1` is irrelevant in the third and fourth observations; here those values are coded as 1, but they could have been coded as missing or any other value. Whatever the value of `r1` for the third and fourth observations, those observations must be removed from the data lest it appear that patient 123 has reentered the data and is eligible for the failure event to recur. Specifying `first` removes the observations.

If you type

```
. stset time r1, id(patid) first
```

the dataset will be changed to contain only two observation on patient 123:

```
. list patid time r1 r2 if patid==123
        patid       time         r1         r2
  1.      123          1          0          0
  2.      123          3          1          0
```

Starting with the *original* dataset, if you type

```
. stset time r2, id(patid) first
```

the dataset will contain all four observations for patient 123 (but fewer on some other patient if event `r2` ever preceded `r1`).

`first` is also useful when analyzing multiple-failure data and you wish to create and set a dataset of the first failure events.

Since `first` can change the data, be sure you have saved your original data on disk. `stset, first` will change the data only if the `stset` operation is successful and then only if any observations need to be deleted.

The force option

The other new option is `force`. `force` drops observations with missing values of the time, outcome, `id()`, `t0()`, or weighting variables, so `force`, like `first`, can change the data and the same cautions (and features) apply.

`force` is useful in cases where data contains missing values and dropping the observations is the appropriate thing to do. We recommend you do not specify `force` at the outset. Instead, try the command without `force` and find out if there is a problem. If there is, specify `force` only after determining that dropping the observations is the appropriate correction.

crc46	Better numerical derivatives and integrals

The new `dydx` and `integ` commands are improved versions of Stata's utilities for calculating numerical first derivatives and integrals (see [R] **range** in the Stata Reference Manual).

These new versions first fit a cubic spline to the data and then compute the derivative or integral of the spline. This method gives more accurate estimates of the derivative and integral, especially for smooth functions, than did the old versions which used simple approximation formulas.

The old `dydx` and `integ` commands also did not deal properly with datasets in which there were tied x values. If there are tied x values, the new versions use the mean of y at the tied x values. The new versions also allow `if` and `in` restrictions and have a `by(`*varlist*`)` option.

Syntax

The syntax of the new `dydx` and `integ` is

`dydx` *yvar xvar* [`if` *exp*] [`in` *range*], `generate(`*newvar*`)` [`replace` `by(`*varlist*`)`]

`integ` *yvar xvar* [`if` *exp*] [`in` *range*] [, `generate(`*newvar*`)` `replace` `by(`*varlist*`)` `trapezoid` `initial(#)`]

Options

`generate(`*newvar*`)` specifies the name of the new variable to be created. It is not optional for `dydx`.

`replace` requests that if an existing variable is specified for `generate()`, it should be overwritten.

`by(`*varlist*`)` requests that the derivative or integral be computed separately for each set of values of *varlist*.

`trapezoid` specifies that the integral be computed using the trapezoidal rule (i.e., the sum of $(x_i - x_{i-1})(y_i + y_{i-1})/2$) rather than using cubic splines (the default). Cubic splines will give superior results for most smooth functions; for irregular functions, `trapezoid` may give better results.

`initial(#)` specifies the initial value of the integral; i.e., the value of the integral at $\min(x)$. If not specified, the initial value is taken as 0.

Examples

As an example, we will generate a dataset containing 100 points from the function $\exp(-x/6)\sin(x)$ over the interval $[0, 12.56]$.

```
. range x 0 12.56 100
obs was 0, now 100
. gen y = exp(-x/6)*sin(x)
```

A graph of these data is shown in Figure 1.

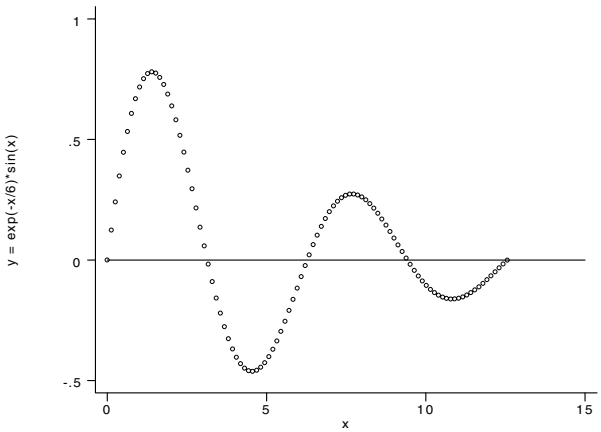

Figure 1.

We estimate the derivative using the new `dydx`, and compute the relative difference between this estimate and the true derivative.

```
. dydx y x, gen(dy)
. gen dytrue = exp(-x/6)*(cos(x) - sin(x)/6)
. gen error = abs(dy - dytrue)/dytrue
```

Figure 2 shows a graph of the error (the variable `error`). The error is greatest at the endpoints as one would expect. The error is approximately 0.5% at each endpoint, but the error quickly falls to less than 0.01%.

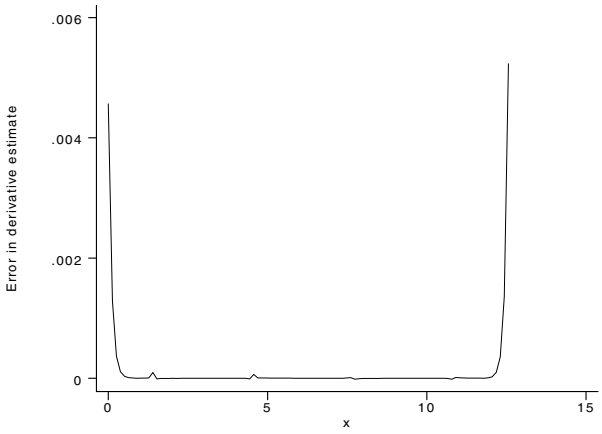

Figure 2. Relative error in derivative estimate.

We now estimate the integral using `integ`:

```
. integ y x, gen(iy)
number of points = 100
integral         = .85316396
. gen iytrue = (36/37)*(1 - exp(-x/6)*(cos(x) + sin(x)/6))
. display iytrue[_N]
.85315901
. display abs($S_2 - iytrue[_N])/iytrue[_N]
5.799e-06
. gen diff = iy - iytrue
```

The relative difference between the estimate (stored in `$S_2`) and the true value of the integral is about 6×10^{-6}. A graph of the absolute difference (`diff`) is shown in Figure 3. Note that here, error is cumulative. Again, most of the error is due to a relatively poorer fit near the endpoints.

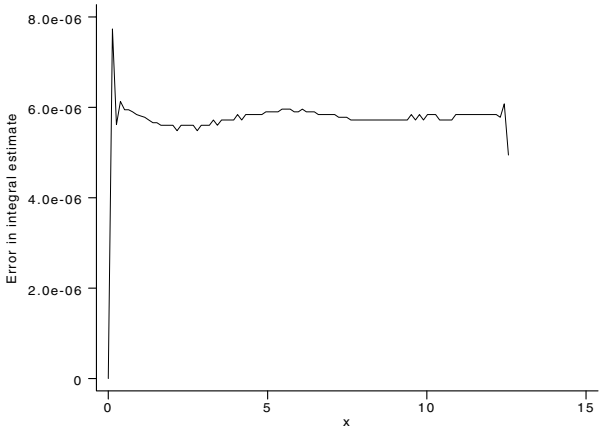

Figure 3. Absolute error in integral estimate.

Saved Results

`integ` saves the number of unique x points in the global macro `S_1` and the estimate of the integral in `S_2`.

Methods and Formulas

Consider a set of data points $(x_1, y_1), \ldots, (x_n, y_n)$ generated by a function $y = f(x)$. `dydx` and `integ` first fit these points with a cubic spline. The cubic spline is then analytically differentiated (integrated) to give an approximation for the derivative (integral) of f.

The cubic spline (see, for example, Press et al. (1992)) consists of $n-1$ cubic polynomials $P_i(x)$, with the ith one defined on the interval $[x_i, x_{i+1}]$:

$$P_i(x) = y_i a_i(x) + y_{i+1} b_i(x) + y''_i c_i(x) + y''_{i+1} d_i(x)$$

where

$$a_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} \qquad b_i(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

$$c_i(x) = \frac{1}{6}(x_{i+1} - x_i)^2 a_i(x)\{[a_i(x)]^2 - 1\} \qquad d_i(x) = \frac{1}{6}(x_{i+1} - x_i)^2 b_i(x)\{[b_i(x)]^2 - 1\}$$

and y''_i and y''_{i+1} are constants whose values will be determined as described below. The notation for these constants is justified by the fact that $P''_i(x_i) = y''_i$ and $P''_i(x_{i+1}) = y''_{i+1}$.

Since $a_i(x_i) = 1$, $a_i(x_{i+1}) = 0$, $b_i(x_i) = 0$, and $b_i(x_{i+1}) = 1$, therefore $P_i(x_i) = y_i$ and $P_i(x_{i+1}) = y_{i+1}$. Thus, the P_i jointly define a function that is continuous at the interval boundaries. It is also desirable that the first derivative be continuous at the interval boundaries; that is,

$$P'_i(x_{i+1}) = P'_{i+1}(x_{i+1})$$

The above $n-2$ equations (one equation for each point except the two endpoints) and the values of the first derivative at the endpoints, $P'_1(x_1)$ and $P'_{n-1}(x_n)$, determine the n constants y''_i.

The value of the first derivative at an endpoint is set to the value of the derivative obtained by fitting a quadratic to the endpoint and the two adjacent points; namely, we use

$$P'_1(x_1) = \frac{y_1 - y_2}{x_1 - x_2} + \frac{y_1 - y_3}{x_1 - x_3} - \frac{y_2 - y_3}{x_2 - x_3}$$

and a similar formula for the upper endpoint.

`dydx` approximates $f'(x_i)$ using $P'_i(x_i)$.

`integ` approximates $F(x_i) = F(x_1) + \int_{x_1}^{x_i} f(x)\,dx$ using

$$I_0 + \sum_{k=1}^{i-1} \int_{x_k}^{x_{k+1}} P_k(x)\,dx$$

where I_0 (an estimate of $F(x_1)$) is the value specified by the `initial(#)` option. If the `trapezoid` option is specified, `integ` approximates the integral using the trapezoidal rule:

$$I_0 + \sum_{k=1}^{i-1} \frac{1}{2}\,(x_{k+1} - x_k)(y_{k+1} + y_k)$$

If there are ties among the x_i, the mean of y_i is computed at each set of ties, the cubic spline is fit to these values.

Acknowledgment

The new versions of `dydx` and `integ` were inspired by the `dydx2` command written by Patrick Royston of the Royal Postgraduate Medical School in London.

Reference

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical Recipes in C, The Art of Scientific Computing*. 2d ed. Cambridge: Cambridge University Press.

dm43	Automatic recording of definitions

Jeroen Weesie, Utrecht University, Netherlands, weesie@weesie.fsw.ruu.nl

In this insert I describe programs `genl` and `repl` that enhance Stata's `generate` and `replace` commands with the following functionality. First, automatic variable labels are generated to record the way in which variables were defined or changed. Second, we employ characteristics, a language element introduced in version 4.0 of Stata, to save the complete definition of variables. These definitions can be displayed (program `defntion`) or reapplied (program `redo`).

Syntax

`genl` [*type*] *newvar* `=` *exp* [`if` *exp*] [`in` *range*] [`, by(`*varlist*`) label(`*string*`) nodef`]

`repl` *oldvar* `=` *exp* [`if` *exp*] [`in` *range*] [`, by(`*varlist*`) label(`*string*`) oldlabel nodef`]

`redo` *varlist*

`defntion` [*varlist*] [`, other`]

Description

`genl` and `repl` are substitutes for Stata internal commands `generate` and `replace`, but they also automatically generate variable labels and create a characteristic `Defntion` that contains the defining expression.

`redo` uses the characteristic `Defntion` to redefine the variables in the order implied by *varlist*. This allows a primitive spreadsheet to be constructed.

`defntion` displays the definitions of the variables.

Remarks

Stata's variable labels may not exceed 31 characteristics, thus truncation may be necessary; to avoid confusion, truncated labels end in "...".

The definition comprises, optionally, the `by` *varlist* clause, the defining expression, and the case restrictions `if` and `in`.

Options

`label(`*string*`)` specifies the label of the variable instead of the defining expression. Specify `label(_)` to force an empty label. Specifying a label not does affect the definition of the characteristic `Defntion`.

`nodef` suppresses the generation (`genl`) or modification (`repl`) of the characteristic `Defntion`.

`oldlabel` specifies that the label of the variable changed not be changed.

`by(`*varlist*`)` specifies that the expression should be evaluated "`by` *varlist*`:`". The sort order of the variables is changed only if needed.

`other` specifies that `defntion` include all variables, even those without a definition.

Examples

The command

```
. genl age = Tnow - Tbirth
```

creates `age` as the difference `Tnow - TBirth` and generates a variable label "`Tnow - TBirth`", and a characteristic `Defntion` equal to "`Tnow - TBirth`". Thus, this command is equivalent to the three commands:

```
. gen age = Tnow - Tbirth
. label var age "Tnow - Tbirth"
. char age[Defntion] = "Tnow - Tbirth"
```

Further examples of the use of genl and repl are

```
. genl N = invnorm(uniform())          normal variate
. genl M = N + 1, label(own label)     specify own label
. repl M = M + uniform(), nodef        replace with new label, do not change definition
```

The by() option allows the specification of subgroups relative to which expressions should be evaluated. Thus, the command

```
. genl int Z = M + _n, by(age)
```

is, apart from the generation of variable labels and the characteristic Defntion, equivalent to the two commands

```
. sort age
. quietly by age: gen int Z = M + _n
```

The command defntion lists all defined variables, i.e., all variables with associated characteristic Defntion.

```
. defntion
Variable | definition (char Defntion)
---------+--------------------------------------------------------------------
age      | Tnow - Tbirth
N        | invnorm(uniform())
M        | N + 1
Z        | by age: M + _n
```

Since N is defined using pseudo-random numbers, re-evaluating the expression generates new values for N:

```
. redo N
```

Note that the variable M was defined in terms of N. Since genl stored the definition of M, it is possible to update M:

```
. redo M
```

Since variables are updated in the specified order, it would have been easier to have issued the command

```
. redo N M
```

gr22 | Binomial smoothing plot

Nicholas J. Cox, University of Durham, UK, FAX (011) 44-91-374-2456, n.j.cox@durham.ac.uk

The Stata command smooth is described in the manual (see [R] **smooth**) as providing robust nonlinear smoothers. It offers a set of ingredients, including running medians of varying lengths, from which users can assemble their own customized smoothers. The menu also includes Hanning, a moving average (running mean) with weights in the ratio 1:2:1, so that in fact users have the further possibility of assembling various linear smoothers. In a now classical work on spectrum estimation in the late 1940s, J. W. Tukey named this smoother after Julius von Hann, an Austrian climatologist who liked using it. The smoothers concerned are collectively called binomial smoothers, with weights in proportion to the binomial coefficients. They remain widely used in several fields, including climatology. It is easy to produce these smoothers using smooth by repeating Hanning. Thus, the binomial smoother of length 5, with weights in the ratio 1:4:6:4:1, is obtained by Hanning twice. The possibility of combining smoothers follows from the fact that convolution of binomial coefficients yields other binomial coefficients. One reference on the associated mathematics is Graham, Knuth, and Patashnik (1989).

What bsmplot does is automate this so that the user gets immediately a graph of smoothing results by specifying the window length, that is, the number of values in the binomial smoother. The most obvious application is to regularly spaced time series, such as temperatures for a series of years.

Syntax

`bsmplot` *yvar xvar* [`if` *exp*] [`in` *range*] [`,` `window(#)` *graph_options*]

Description

`bsmplot` produces a plot of both *yvar* and the result of smoothing *yvar* by a binomial filter against *xvar*. Normally, but not necessarily, *xvar* will contain equally spaced times and *yvar* some fluctuating response.

Binomial filters have weights given by the coefficients in the expansion of $(a+b)^n$. Thus,

$$(a+b)^2 = 1a + 2ab + 1b \qquad (a+b)^4 = 1a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + 1b^4$$

and more compactly, listing here only examples with odd numbers of weights, which are the practical choices in smoothing:

1 2 1	length 3
1 4 6 4 1	length 5
1 6 15 20 15 6 1	length 7
1 8 28 56 70 56 28 8 1	length 9

and so on. Each weight is divided by the sum of the weights to produce scaled weights that have sum 1. Thus, smoothing with weights in the ratio 1:2:1 means compute the smoothed value at index i by the sum of one-fourth the values at indices $i-1$ and $i+1$ and one-half the value at index i, as each weight is divided by their sum 4.

Options

`window(#)` specifies the length of the binomial filter. The default is 3, giving the filter with weights in the ratio 1:2:1, often known as Hanning. The length must be an odd number and at least 3.

graph_options are options allowed with `graph, twoway`. The default values are `xlabel ylabel c(||s) s(iii) gap(6)`. `t1title` shows the window length. `c(..s)` would show just the smooth values, suppressing the plot of the data.

Example

We consider smoothing the global average temperature data for 1856 to 1990 given in Figure 1.

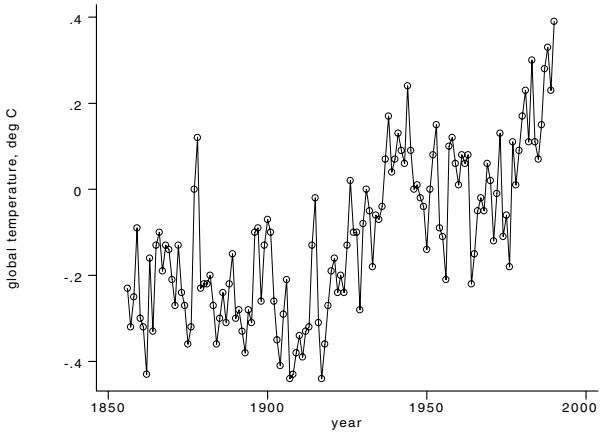

Figure 1. Unsmoothed data.

In Figure 2, we show the result of using `bsmplot` for the smoother of length 11.

```
. bsmplot gltemp year, window(11)
```

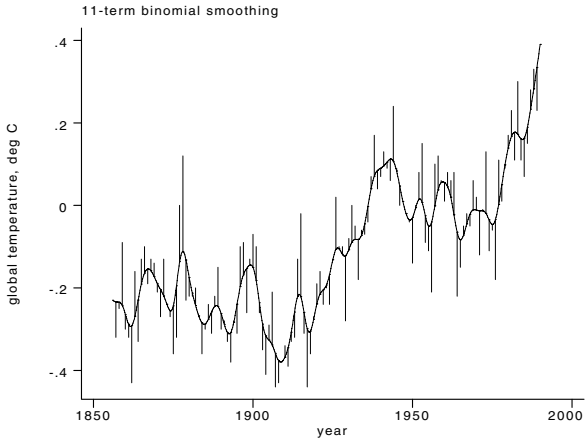

Figure 2. Smoother of length 11.

In Figure 3, we increase the window width to 21 and use the `c(..s)` option so that we only see the smoothed data.

```
. bsmplot gltemp year, window(11) c(..s)
```

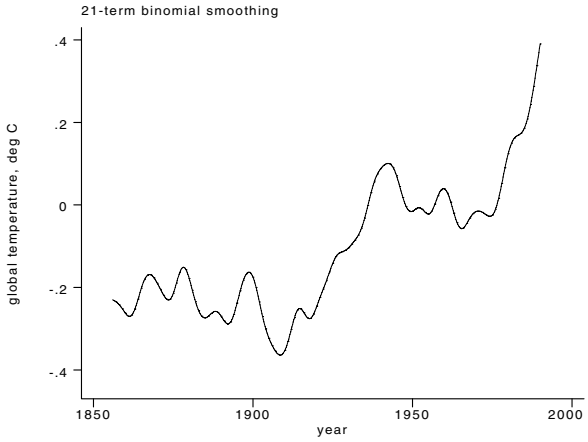

Figure 3. Smoother of length 21.

Reference

Graham, R. L., D. E. Knuth, and O. Patashnik. 1989. *Concrete mathematics: A foundation for computer science.* Reading, MA: Addison–Wesley.

gr23	Graphical assessment of the Cox model proportional hazards assumption

Joanne M. Garrett, University of North Carolina at Chapel Hill, garrettj@med.unc.edu

One of the main assumptions of the Cox proportional hazards model is that the hazard ratio is proportional over time. For example, suppose a group of cancer patients on an experimental treatment are followed for 10 years. If the hazard of dying is twice the rate for the non-treated group as for treated group (HR = 2.0), the proportional hazards assumption presumes this ratio is the same at one year, two years, or at any point on the time scale. Because the Cox model, by definition, is constrained to follow this assumption, it is important to evaluate its validity before modeling. If the assumption fails, alternative modeling choices would be more appropriate (e.g., a stratified Cox model).

This insert offers two programs, `stphplot` and `stcoxkm`, which can be used to graphically evaluate the proportional hazards assumption. Although using a graph to assess the validity of the assumption is somewhat subjective, it can be a helpful tool. Both programs are additions to the new survival-time data and commands created in version 5.0 of Stata and follow that format. Data must be `stset` before these commands will work (see [R] **st stset**).

`stphplot` plots −ln(−ln(survival)) curves for each category of a nominal or ordinal independent variable versus ln(time). These are often referred to as "loglog" plots. Optionally, these estimates can be adjusted for covariates. If the plotted lines are reasonably parallel, the proportional hazards assumption has not been violated, and it would be appropriate to base the estimate for that variable on a single baseline survivor function. Another graphical, though less common method of evaluating the proportional hazards assumption is to plot the Kaplan–Meier observed survival curves and compare them to the Cox predicted curves for the same variable. This plot can be graphed using `stcoxkm`. When the predicted and observed values are close together, the proportional hazards assumption has not been violated.

Syntax of the stphplot command

`stphplot` [`if` *exp*] [`, by(`*xvar*`) strata(`*xvar*`) adjust(`*varlist*`) noshow` *graph_options*]

Options

`by(`*xvar*`)` produces separate "loglog" lines for each value of *xvar*, where *xvar* is a nominal or ordinal independent variable.

`strata(`*xvar*`)` is an alternative to `by()`; the `adjust()` option must also be included (see [R] **st sts graph**).

`adjust(`*varlist*`)` adjusts the estimates to that for 0 values of the *varlist* specified. Any adjusted variables are centered automatically by the program before estimation, and therefore do not have to be centered beforehand; `adjust()` can be specified for either `by()` or `strata()`.

`noshow` prevents `stphplot` from showing the key `st` variables.

graph_options are most of the options available in `graph`.

Syntax of the stcoxkm command

`stcoxkm` [`if` *exp*] [`, by(`*xvar*`) noshow` *graph_options*]

Options

`by(`*xvar*`)` produces separate predicted and observed lines for each value of *xvar*, a nominal or ordinal independent variable.

graph_options are most of the options available in `graph`; if `ylabel()` is selected, values for the y-axis must be included.

Note: `strata()` and `adjust()` are not available for `stcoxkm`.

Example 1

All the examples come from a leukemia remission study. The data consist of 42 patients who are followed over time to see how long (`weeks`) before they go out of remission (`relapse`: 1 = yes, 0 = no). Half the patients receive a new experimental drug and the other half receive a standard drug (`trtment1`: 1 = drug A, 0 = standard). White blood cell count, a strong indicator of the presence of leukemia, is divided into three categories (`wbc3cat`: 1 = normal, 2 = moderate, 3 = high).

```
. describe
Contains data from Leukemia.dta
  obs:            42                          Leukemia Remission Study
 vars:             8                          27 Dec 1996 12:47
 size:           504 (98.9% of memory free)
------------------------------------------------------------------------------
   1. weeks       byte   %8.0g                 Weeks in Remission
   2. relapse     byte   %8.0g       yesno     Relapse
   3. trtment1    byte   %8.0g       trtlbl1   Treatment I
   4. trtment2    byte   %8.0g       trtlbl2   Treatment II
   5. wbc3cat     byte   %9.0g       wbclbl    White Blood Cell Count
   6. wbc1        byte   %8.0g                 wbc3cat==Normal
   7. wbc2        byte   %8.0g                 wbc3cat==Moderate
   8. wbc3        byte   %8.0g                 wbc3cat==High
------------------------------------------------------------------------------
Sorted by:  weeks
```

```
. stset weeks relapse
data set name:  Leukemia.dta
             id:  --              (meaning each record a unique subject)
     entry time:  --              (meaning all entered at time 0)
      exit time:  weeks
  failure/censor:  relapse
```

In this example, we examine whether or not the proportional hazards assumption holds for drug A versus the standard drug (`trtment1`). First, we will use `stphplot`, followed by `stcoxkm`.

```
. stphplot, by(trtment1) c(ll) xlabel ylabel
```

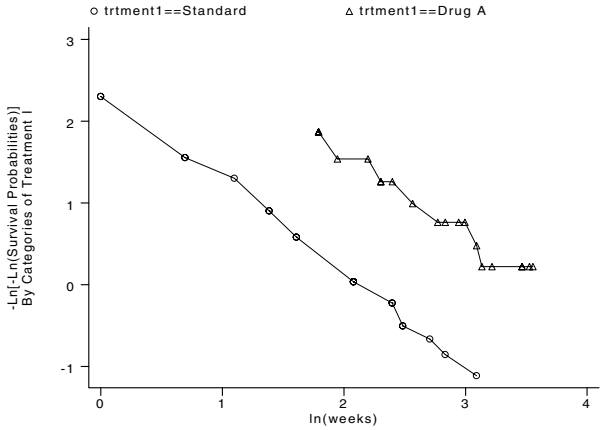

Figure 1.

```
. stcoxkm, by(trtment1) c(..ll) s(OOST) xlabel
```

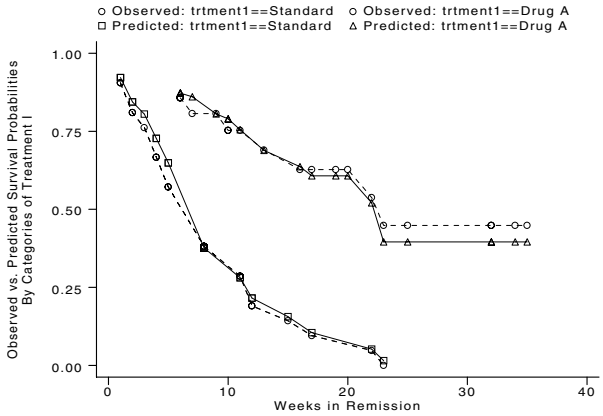

Figure 2.

Figure 1 (`stphplot`) displays lines that are parallel, implying the proportional hazards assumption for `trtment1` has not been violated. This is confirmed in Figure 2 (`stcoxkm`) where the observed values (connected by STAGE with dotted lines) and predicted values (connected by solid lines) are close together.

The graph in Figure 3 is the same as Figure 1, adjusted for white blood cell count (using two dummy variables). The adjustment variables were centered temporarily by `stphplot` before the adjustment was made.

```
. stphplot, strata(trtment1) c(ll) xlabel ylabel adj(wbc2 wbc3)
```

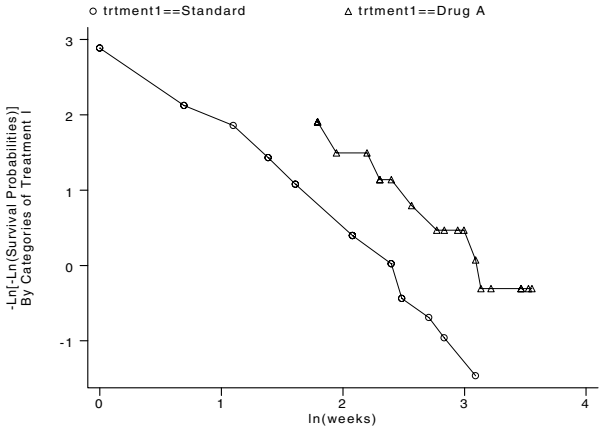

Figure 3.

Note that the lines in Figure 3 are still parallel, although somewhat closer together. Examining the proportional hazards assumption on a variable without adjustment for covariates is usually adequate as a diagnostic tool before using the Cox model. However, if it is known that adjustment for covariates in a final model is necessary, one may wish to re-examine whether or not the proportional hazards assumption still holds.

Example 2

We use the same data to examine the proportional hazards assumption for white blood cell count (`wbc3cat`). Separate lines are created for each of the three categories. Predicted values for the Cox model are based on estimates using dummy variables.

```
. stphplot, by(wbc3cat) c(lll) xlabel ylabel
```

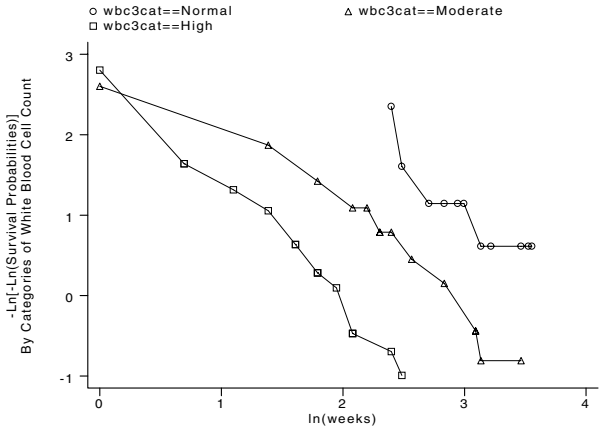

Figure 4.

```
. stcoxkm, by(wbc3cat) c(...lll) s(OOOTTT) xlabel
```

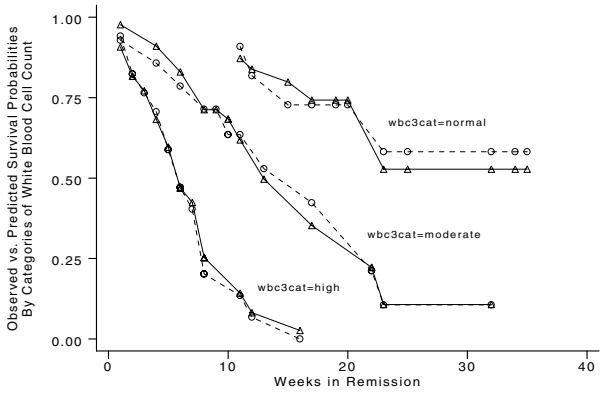

Figure 5.

Once again the lines are reasonably parallel (Figure 4). Although this is not true for the first time points for each of the categories of white blood cell count, the majority of the data do not violate the proportional hazards assumption. The observed and predicted estimates are close (Figure 5), particularly for the "high" white blood cell count group; less so for the "moderate" and "normal" groups. However, the differences are rather small, and modeling this variable either way should not change the interpretation of the results.

Example 3

There is another variable on this dataset which measures a different drug (`trtment2`: 1 = drug B, 0 = standard). We wish to examine the proportional hazards assumption for this variable.

```
. stphplot, by(trtment2) c(ll) xlabel ylabel
```

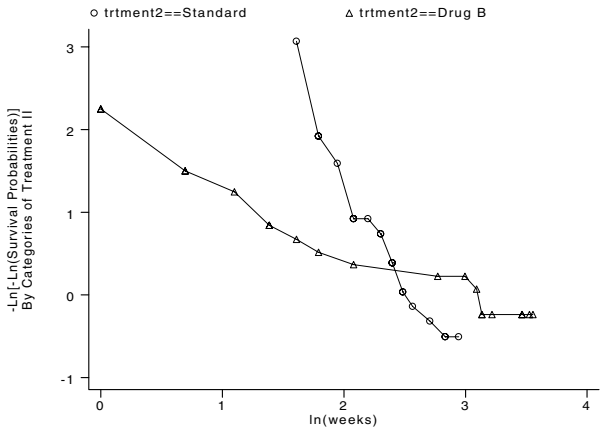

Figure 6.

```
. stcoxkm, by(trtment2) c(..ll) s(OOST) xlabel
```

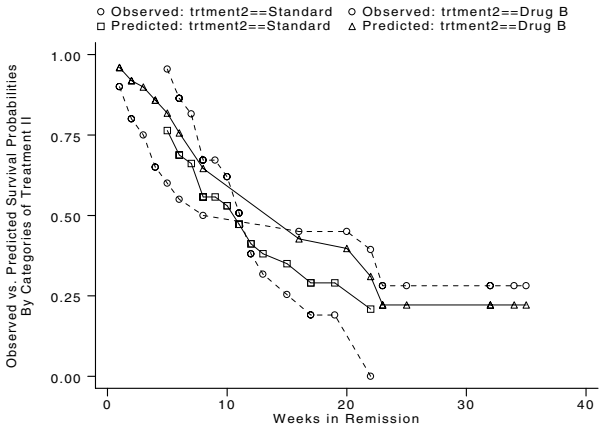

Figure 7.

Clearly this variable violates the proportion hazards assumption. Not only are the lines non-parallel, they cross (Figure 6). The observed and predicted values deviate from each other badly (Figure 7). We have overestimated the positive effect of drug B for the first half of the study, and have underestimated it in the later weeks. A single hazard ratio describing the effect of this drug would be inappropriate. We definitely would want to stratify this variable in our Cox model.

ip14 Programming utility: Numeric lists

Jeroen Weesie, Utrecht University, Netherlands, weesie@weesie.fsw.ruu.nl

This insert describes a programming utility that is useful in programs with options that should accept a list of numbers. With a `numlist` style option, a user can be allowed to specify `1-5` rather than typing `1 2 3 4 5` in full, and `0-1/0.25` rather than typing `0 0.25 0.50 0.75 1.0`. The command `numlist` verifies that the user has specified a legal numeric list, and—much like `unabbrev`—returns an expanded numeric list to the calling program.

`numlist` has a `sort` option that specifies that the expanded numeric list should be in increasing order.

Syntax

The syntax of `numlist` is

`numlist` *numeric_list* [`,` `min`(#) `max`(#) `real` `sort` `format`(*fmt*)]

where *numeric_list* is described below and *fmt* is any legal display format.

Description

`numlist` expands a *numeric_list* which should satisfy the syntax

numeric_list := *numeric_term* [*numeric_list*]
numeric_term := [# | #-# | #-#/#]

with semantics

expression	expands to
n	n
n_1 - n_2	$n_1 \quad n_1+1 \quad n_1+2 \quad \ldots \quad n_2$
n_1 - n_2 / n_3	$n_1 \quad n_1+n_3 \quad n_1+2n_3 \quad \ldots \quad n_2$

A *numeric_term* may not contain embedded blanks, while subsequent *numeric_terms* should be separated by white space.

A number may be real if the option `real` is specified, otherwise number should be an integer. Note that if `n2` < 0 (in the second or third form), you have to enter two subsequent minus signs. Real numbers written in scientific notation may produce a syntax error.

The expanded list of numbers, separated by blanks, is returned in the global macro `S_1`. The number of numbers is returned in the global macro `S_2`.

Options

`min(#)` specifies the minimum number of terms allowed. The default is `min(1)` if not specified.

`max(#)` specifies the maximum number of terms allowed. The default is `max(infinity)`. Note though that Stata imposes a (version dependent) limit on the length of macros.

`real` specifies that the numbers in a *numeric_list* may be reals.

`sort` sorts the values in increasing order.

`format(`*fmt*`)` specifies a display format, such as `%5.3f` or `%9.2g`, that is used to format the numbers in the expanded numeric list.

Examples

```
. numlist 1-5                 expands to    1 2 3 4 5
. numlist 1-5/2               expands to    1 3 5
. numlist 1-6/2               expands to    1 3 5
. numlist 1 5-7 11-17/2       expands to    1 5 6 7 11 13 15 17
```

Negative numbers may be included as well. Note, in particular, that negative upper-limits to ranges and negative increments are permitted, but lead to funny looking expressions.

```
. numlist -2-1                expands to    -2 -1 0 1
. numlist -6--2               expands to    -6 -5 -4 -3 -2
. numlist -6--2/2             expands to    -6 -4 -2
. numlist -2--6/-2            expands to    -2 -4 -6
```

Lists may be sorted in increasing order, for example,

```
. numlist 1-10/2 2-10/2, sort         expands to    1 2 3 4 5 6 7 8 9 10
```

Finally, lists in real numbers may lead to unexpected and unwanted results due to Stata's default formatting:

```
. numlist 0-10/2.5, real              expands to    0 2.5 5 7.5 10
. numlist 0-1/0.3, real               expands to    0 0.3 0.6 0.88888888889
. numlist 0-1/0.1, real f(%3.1f)      expands to    0 0.3 0.6 0.9
```

`numlist` serves a similar function for numeric_lists as Stata's `unabbrev` for varlists. Both will be of interest mainly to programmers. Below we give the outline of how a program might use `numlist`. In the outline, `also()` is an option that should allow a `numlist`.

```
program define ...
        ...
        local options "ALso(string) ..."
        ...
        parse "`*'"
        ...
        if "`also'" ~= "" {
                numlist `also'
                local also "$S_1"
        }
        ...
end
```

Acknowledgment

`numlist` is a modified version of the `lexp` program (which is a subroutine of the `for2` command) written by Patrick Royston. `numlist` allows for *numeric_lists* with negative and/or real-valued numbers, and it optionally sorts lists.

ip15	A dialog box layout manager for Stata

Timothy J. Schmidt, Federal Reserve Bank of Kansas City, tschmidt@frbkc.org

In this insert, we present a program called `diablo` that constructs a Stata dialog box containing user-specified window controls. The program thinks of a dialog box as a series of rows of controls and a user of `diablo` can very easily specify the nature and contents of these rows without having to specify their exact location. This greatly simplifies the construction of dialog boxes.

Syntax

The `diablo` command assumes two forms, both of which must be used in constructing a dialog box. To specify the window controls for each row of the dialog box, the syntax is

`diablo #,` *wincontrol* [`|` *wincontrol*] ...

where # is a positive integer indicating the row number in the dialog box and *wincontrol* is one of the following:

wincontrol	window control
`bu["`*label*`",` *macroname* [`,` *options*] `]`	command button
`ch["`*text*`",` *macroname* [`,` *options*] `]`	check box
`ed[`*macroname* [`,` *options*]`]`	edit box
`st[`*text_macroname* [`,` *options*]`]`	static text
`ra["`*text*`", "`*text*`"` [`, "`*text*`"` ...]`,` *macroname*`]`	radio button
`ls[`*list_macroname*`,` *macroname* [`,` *options*]`]`	single choice list box
`lm[`*list_macroname*`,` *macroname* [`,` *options*]`]`	multiple choice list box
`cs[`*list_macroname*`,` *macroname* [`,` *options*]`]`	single choice combo box
`cm[`*list_macroname*`,` *macroname* [`,` *options*]`]`	multiple choice combo box
`i`	invisible place holder

Note that all Stata window controls are available. In addition, `diablo` accepts an invisible place holder to allow the user to control spacing in the dialog box. Several window controls in a row can be specified by separating them with a "pipe" ("`|`"). All window controls (except `i`) require a list of parameters contained in square brackets and separated by commas. The window controls, denoted by two-letter abbreviations, accept all valid Stata options listed in [R] **window control**. Options must be specified as the final parameter in a window control parameter list.

After defining each row of a dialog box using the above syntax, one creates and displays the dialog box with the command

`diablo "`*box_title*`"`

where *box_title* is the text that will appear in the dialog box title bar.

Discussion

One of the most exciting new features in Stata version 5.0 is the ability to construct dialog boxes. Stata users can now use dialog boxes to construct a custom graphical user interface (GUI) for their programs.

While Stata provides a rich set of window controls, it can be tedious to construct dialog boxes using Stata's `window control` ([R] **window control**) and `window dialog` ([R] **window dialog**) commands. Each control requires a separate Stata command. Moreover, each window control requires explicit coordinates for placement in the dialog box. These requirements may be burdensome if the process of constructing a dialog box involves a lot of trial and error. For instance, adding an additional window control to a dialog box would usually require the programmer to recalculate the placement coordinates for all of the other controls in the box.

Some computer languages, such as Java, provide tools to reduce the effort required to construct dialog boxes and other window controls. These tools, often called "layout managers," allow the user to specify the elements of a dialog box while abstracting from most of the details of their construction. The purpose of `diablo` ("DIAlog Box LayOut manager") is to provide such a tool for Stata programmers.

`diablo` simplifies and automates several features of window control specification and dialog box construction. Instead of issuing a separate Stata command for each window control in a dialog box, `diablo` users give one command for each row. For example, the following command creates the first row of a dialog box with a static text field and an edit box:

```
. diablo 1, st[Text] | ed[EDvar]
```

In this example, the text in the static text field is contained in the global variable `Text`. The global variable `EDvar` will contain whatever the user types into the edit box. The integer `1` indicates that the controls in the command will be placed in the first row of the dialog box. Programmers can specify dialog box rows in any order since `diablo` will not actually construct the dialog box until the user issues a final `diablo` command with the dialog box title in quotes. For instance,

```
. diablo "Linear regression"
```

creates a dialog box with the title "Linear regression".

`diablo` positions the window controls in a rectangular $r \times c$ matrix where r is the number of rows specified by the programmer (in the prior `diablo` commands) and c is the maximum number of controls specified in any of those rows. If the user specifies fewer controls in one row than another, `diablo` will pad the right side of the shorter row with empty space. For example,

```
. diablo 1, st[Text1] | st[Text2]
. diablo 2, ls[LBvars, LBpick]
. diablo 3, bu["OK", OKchek] | bu["Cancel", DBcancel]
. diablo "Test dialog box"
```

is equivalent to

```
. diablo 1, st[Text1] | st[Text2]
. diablo 2, ls[LBvars, LBpick] | i
. diablo 3, bu["OK", OKchek] | bu["Cancel", DBcancel]
. diablo "Test dialog box"
```

`diablo`'s `i` (read "invisible") control serves as a place holder in the layout matrix. In this example, both sets of commands instruct `diablo` to create a 3×2 dialog box with two static text controls in row 1 and a single choice list box in the first column of the second row. The second column of the second row would be empty.

Some Stata window controls have text labels associated with them. For instance, command buttons require a text parameter to label the button. `diablo` assists the programmer with these controls by automatically scaling command buttons, radio buttons, check boxes, and static text according to the length of their associated labels. `diablo` also aligns radio buttons in a row at equal horizontal intervals determined by the longest button label.

Examples

The following artificial example will illustrate how to use `diablo` to construct a dialog box with each of Stata's window controls. The result is shown in Figure 1.

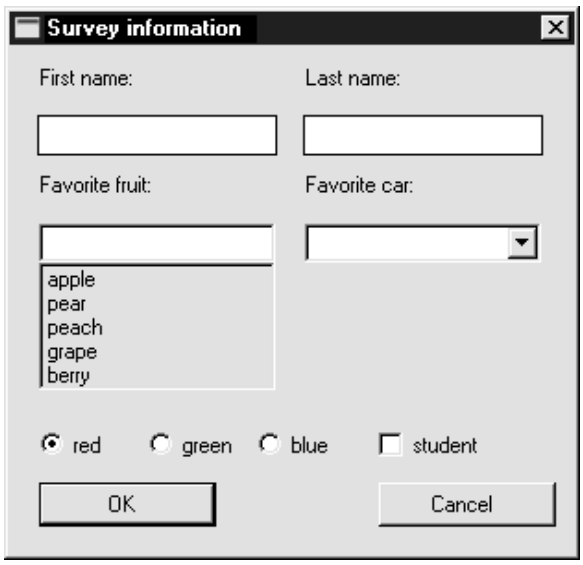

Figure 1.

To construct this dialog box, we first define the global variables containing the static text and list box options.

```
. global Name1 "First name:"
. global Name2 "Last name:"
. global Fruit "Favorite fruit:"
. global Car "Favorite car:"
. global LBfruit "apple pear peach grape orange"
. global LBcars "Acura;BMW;Chevy;Dodge;Ferrari"
```

Stata's `static` control requires a global macro that contains the text to be displayed. In this example, global macros `Name1`, `Name2`, `Fruit`, and `Car` contain the text for the static controls in our sample dialog box. Stata's list box and combo box controls require a global macro that contains the choices presented in the list. In this example, global macros `LBfruit` and `LBcars` contain the lists of choices for the list box and combo box that will be included in our sample dialog box. Note that spaces separate the choices in `LBfruit`, while semicolons separate the choices in `LBcars`. (I chose different separators in the two lists to demonstrate the use of Stata's `parse` option in the `combo box` control.)

The next step is to use `diablo` to specify the rows of the dialog box. This sample dialog box will have six rows. Each row requires a separate `diablo` command.

```
. diablo 1, st[Name1] | st[Name2]
. diablo 2, ed[EDname1] | ed[EDname2, password maxlen 8]
. diablo 3, st[Fruit] | st[Car]
. diablo 4, ls[LBfruit, LBpick] | cs[LBcars, CBpick, parse(;)]
. diablo 5, ra["red", "green", "blue", RAcolor] | ch["student", CHpick]
. diablo 6, bu["OK", OKchek, default] | bu["Cancel", DBcancel]
```

The first row contains two static text controls (`st`). `diablo`'s static text controls require only one parameter—the global macro that contains the text to be displayed.

The second row contains two edit controls (`ed`). `diablo`'s edit controls require only one parameter—the name of a global macro to hold the text entered by the user. In this example, `EDname1` will contain any text entered by the user in the first edit control in the second row. The second edit control in the row demonstrates how to specify any valid options for that control. The Stata manual ([R] **window control**) enumerates the valid options for each control. Two options, `password` and `maxlen 8`, are specified after the required global variable name. `diablo` syntax requires any options to be specified only after all required parameters.

After a third row with two static text controls, the fourth row of this sample dialog box contains a list box (`ls`) and a combo box (`cs`). Both types of boxes require two parameters. The first parameter is the global variable with the string of possible choices. The second parameter is the global variable which will contain the user's selection. Finally, the list and combo boxes will also accept a third parameter—the optional `parse` command. In the example above, the `parse` option is used in the combo box control to distinguish the choices, separated by semicolons, in the `LBcars` global variable. Because `diablo` parses

on commas to distinguish window control parameters, users should avoid using commas as field separators in their choice global variables (like LBcars).

The fifth diablo command creates a set of radio buttons (ra) and a check box (ch) in the fifth row of the dialog box. The radio button control requires button labels in quotes; Stata requires at least two buttons in each set of radio buttons. The example above creates a set of three radio buttons, respectively labeled "red", "green", and "blue." The global variable RAcolor will contain the user's selection. The status of the check box, here labeled "student," will be recorded in the global variable CHpick.

The sixth and final row contains two command buttons (bu), one labeled "OK" and the other labeled "Cancel". In the second parameter in a command button, specify a global variable that codifies an action when the user presses the button. In the example above, the first button also contains an option that marks it as the default button in the dialog box.

Now that we have specified the contents of the dialog box, we create it using the second form of the diablo command:

```
. diablo "Survey information"
```

The command in this example will create a dialog box with "Survey information" in the title bar. diablo will automatically handle all of the details of calculating the placement of each control in the dialog box.

To see how diablo simplifies the task of creating dialog boxes in Stata, we can create the same sample dialog box using standard Stata commands. We begin by defining the same global variables:

```
. global Name1 "First name:"
. global Name2 "Last name:"
. global Fruit "Favorite fruit:"
. global Car "Favorite car:"
. global LBfruit "apple pear peach grape orange"
. global LBcars "Acura;BMW;Chevy;Dodge;Ferrari"
```

Then we have to program each window control separately.

```
. window control static Name1 10 5 80 9
. window control static Name2 100 5 80 9
. window control edit 10 19 80 10 EDname1
. window control edit 100 19 80 10 EDname2 password maxlen 8
. window control static Fruit 10 34 80 9
. window control static Car 100 34 80 9
. window control ssimple LBfruit 10 48 80 50 LBpick
. window control scombo LBcars 100 48 80 30 CBpick parse(;)
. window control radbegin "red" 10 103 30 10 RAcolor
. window control radio "green" 45 103 30 10 RAcolor
. window control radend "blue" 80 103 30 10 RAcolor
. window control check "student" 120 103 70 10 CHpick
. window control button "OK" 10 118 60 12 OKchek default
. window control button "Cancel" 120 118 60 12 DBcancel
```

Finally, we create the dialog box with the window dialog ([R] **window dialog**) command.

```
. window dialog "Survey information" . . 195 150
```

Note the differences between the diablo method and the standard Stata method of creating the same dialog box. Using diablo almost always requires fewer (and never more) commands than the standard method. In this example, the diablo method requires eight fewer commands than the standard method. In addition, diablo eliminates the need to specify explicit coordinates for the placement of dialog box controls. This feature permits more general and flexible code and saves the programmer time in the development process. By requiring explicit placement coordinates, however, the standard Stata method offers finer control of dialog box layout.

As another example, we use diablo to produce a dialog box that has the same components as the example in the Stata manual (see [R] **window control**). The following program produces the dialog box in Figure 2, which only differs slightly from the Stata example.

```
program define stdb
        version 5.0

        local varlist opt
        parse "`*´"
        global DB_var "`varlist´"
```

```
        global CHpick1 1
        global CHpick2 0
        global CHpick3 0
        diablo 1, ch["Check 1 ", CHpick1] | ch["Check 2 ",CHpick2] | ch["Check 3 ",CHpick3]

        global RApick 1
        diablo 2, ra["radio 1","radio 2","radio 3",RApick]

        global Name1 "Please enter name:"
        global Name2 "Please enter password:"
        diablo 3, st[Name1] | st[Name2]
        diablo 4, ed[EDname1] | ed[EDname2, password maxlen 8]

        global Name3 "Select many:"
        global Name4 "Select one:"
        diablo 5, st[Name3] | st[Name4]
        diablo 6, lm[DB_var,DB_ms] | ls[DB_var,DB_ss]

        diablo 7, st[Name3] | st[Name4]
        diablo 8, cm[DB_var,DB_ms1] | cs[DB_var,DB_ss1]

        diablo 9, bu["OK", DBok, default] | bu["Cancel",DBcancel]

        local ok 1
        global DBok      "di `ok´"
        global DBcancel  "exit 3000"

        diablo "Test Dialog"
end
```

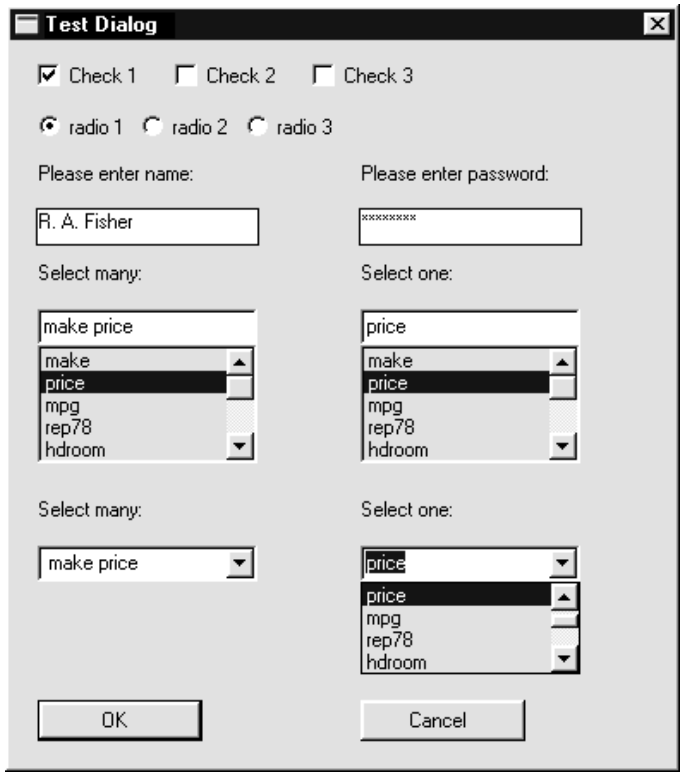

Figure 2.

Conclusion

In summary, the standard method of dialog box construction may be appropriate for boxes requiring unconventional configurations of window controls. For most applications, however, `diablo` offers significant advantages in writing compact, simple, and flexible code to construct dialog boxes in Stata.

sbe13.1 | Correction to age-specific reference intervals ("normal ranges")

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

In our article in the previous edition of STB (Wright and Royston 1996), there is an error in the description of the data from the American HANES I Survey of Health and Nutrition (page 29). The text states that the data are the heights of 2274 girls, while the axis labels and legends of the figures refer to weight. We would like to confirm that the results, graphs, and output from `xriml` are all obtained from the data on weight and that the word "height" should be read as "weight". We apologize for any confusion this may have caused.

Reference

Wright, E. and P. Royston. 1996. sbe13: Age-specific reference intervals ("normal ranges"). *Stata Technical Bulletin* 34: 24–34.

sg63 | Logistic regression: standardized coefficients and partial correlations

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

Many medical research journals have articles which provide standardized coefficients for logistic regressions models on clinical data. Although these can be produced quite easily in SAS, Stata does not provide for such output. Since standardized coefficients are unitless, they can provide information as to the relative worth a predictor has to the model.

I have provided a command called `lstand` to be used following the `logistic` command which displays the predictor, its coefficient, odds ratio, standardized coefficient, partial correlation (Atkinson's R), and the p-value. I have found it useful to directly import the resultant table into a word processor when preparing the results section of a manuscript.

Syntax

```
lstand
```

Remarks

Standardized coefficients are calculated in the same manner as in SAS:

$$\frac{Bs}{\sqrt{\pi^2/3}}$$

where the denominator is the standard deviation of the underlying logistic distribution, B is the predictor estimate, and s is the standard deviation of the predictor.

Partial correlations, otherwise known as Atkinson's R, are calculated as

$$R = \sqrt{\frac{W-2}{2|L_0|}}$$

where W is the Wald statistic and $|L_0|$ is the absolute value of the intercept log-likelihood. I had written a program to display partial correlations and other statistics for STB-7 (May 1992), but it is now outdated.

Example

We use the low birth weight data described in Hosmer and Lemeshow (1989, Appendix 1) and in the Stata manual (see [R] **logistic**). First, we use `logistic`:

```
. logistic low age lwt smoke ptl ht ui
Logit Estimates                                     Number of obs =    189
                                                    chi2(6)       =  25.88
                                                    Prob > chi2   = 0.0002
Log Likelihood = -104.39591                         Pseudo R2     = 0.1103
------------------------------------------------------------------------------
     low | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .9586258   .0331527     -1.222   0.222        .895801    1.025857
     lwt |   .9858131   .0065579     -2.148   0.032       .9730433    .9987505
   smoke |   1.734347   .5959725      1.602   0.109       .8843789    3.401213
     ptl |    1.80987    .630593      1.703   0.089       .9142657    3.582796
      ht |   6.439757   4.419149      2.714   0.007       1.677839     24.7166
      ui |   2.089219   .9537039      1.614   0.107       .8539267    5.111489
------------------------------------------------------------------------------
```

Now we see the results of `lstand`:

```
. lstand
Table of Predictor Estimates:
Standardized Coefficients and Partial Correlations
No.  Var          Coef          OR      St.Coef      PartCorr    Prob(z)
================================================================================
0   Constant   1.3790
1   age       -0.0423      0.9586      -0.1234        0.0000      0.222
2   lwt       -0.0143      0.9858      -0.2409       -0.1055      0.032
3   smoke      0.5506      1.7343       0.1486        0.0492      0.109
4   ptl        0.5933      1.8099       0.1614        0.0619      0.089
5   ht         1.8625      6.4398       0.2511        0.1512      0.007
6   ui         0.7368      2.0892       0.1447        0.0508      0.107
================================================================================
```

Reference

Hosmer, D. W., Jr., and S. Lemeshow. 1989. *Applied Logistic Regression.* New York: John Wiley & Sons.

sg64 | pwcorrs: An enhanced correlation display

Fred Wolfe, Arthritis Research Center, Wichita, KS, fwolfe@southwind.net

The Stata commands `correlate` and `pwcorr` display the Pearson correlation matrix for a group of variables. `correlate` performs the analysis using casewise deletion, while `pwcorr` is a pairwise procedure. `spearman` displays Spearman's rank correlation for a single pair of variables at a time. Although these commands provide the desired results, the default formatting of the output is often unsatisfactory.

Suppose one has a dataset of 200 variables and is interested in the correlation of only a few of these variables with many or all of the rest of the variables. Using `pwcorr` or `correlate` one must display the entire matrix. Since the procedure displays only seven columns at a time, a complete display of such a matrix would fill more than 75 computer screens. One could speed up the output by moving the variables of major interest to the front of the variable list using `aorder` and then "breaking" after a sufficient number of column variables and their row pairs have been displayed. To do this is awkward at best; to do it with the `spearman` procedure is impossible.

The `pwcorrs` command is an improved version of `pwcorr` and `spearman` that combines the features of both commands into single command that has enhanced formatting.

First, `pwcorrs` allows the specification of from 1 to 6 column variables. It then displays the correlations of these *n* variables with the selected row variables. If no column variables are selected, then `pwcorrs` defaults to the standard Stata method of displaying the entire correlation matrix.

`pwcorrs` also allows control over the row variables. Using the `select` option, the analyst can display an alphabetic list of row variables, e.g., `select(a b g,s-t)`.

`pwcorrs` will display the output in the dataset order or, optionally, in sorted order. If required, casewise deletion can be specified. To make the output more appropriate and more useful for exporting, variables that are specified more than once will be deleted, and string variables will be skipped and not displayed. The `above` option incorporates a suggestion of Nick Cox (University of Durham, UK) that specifies the minimum level of correlation coefficient required for printing.

The original code is modified from the standard Stata (version 5.0) ado files, and the help files were similarly derived from Stata's. Helpful criticisms, comments and ideas were obtained from Cox.

Syntax

```
pwcorrs [varlist] [weight] [if exp] [in range] [, obs sig print(#) star(#) bonferroni
        sidak above(#) vars(#) sort cwdeletion spearman select(list) ]
```

Options

`obs` adds a line to each row of the matrix reporting the number of observations used in calculating the correlation coefficient.

`sig` adds a line to each row of the matrix reporting the significance level of each correlation coefficient.

`print(#)` specifies the significance level of correlation coefficients to be printed. Coefficients with larger significance levels are left blank. `print(10)` or `print(.1)` would list only coefficients significant at the 10% level or better.

`star(#)` specifies the significance level of coefficients to be starred. `star(5)` or `star(.05)` would star all coefficients significant at the 5% level or better.

`bonferroni` makes the Bonferroni adjustment to calculated significance levels. This affects printed significance levels and the `print()` and `star()` options. `pwcorrs, print(.05) bonferroni` prints coefficients with Bonferroni-adjusted significance levels of .05 or less.

`sidak` makes the Sidak adjustment to calculated significance levels. This affects printed significance levels and the `print()` and `star()` options. `pwcorrs, print(.05) sidak` prints coefficients with Sidak adjusted significance levels of .05 or less.

`above(#)` specifies the minimum level of correlation coefficients to be printed. Coefficients with smaller coefficients are left blank. `above(.5)` would list only coefficients of 0.5 or greater.

`vars(#)` specifies that on the first # variables on the *varlist* are to be correlated with the variables on the *varlist*. This produces # columns of correlations rather than a correlation matrix. From 1 to 6 variables may be specified.

`sort` requests the *varlist* be reported in sorted order. If `vars(#)` is specified, the first `vars(#)` will not be sorted.

`cwdeletion` removes observations (cases) with missing values for the *varlist* from the calculations.

`spearman` specifies Spearman correlations. The default is to perform Pearson correlations.

`select(list)` selects the letters of the alphabet to be included in the analyses. A range can be selected as well, e.g., `(a f s-w)`.

Examples

We begin by looking at Stata's automobile data. Using the command `pwcorrs` with no arguments gives the same output as `pwcorr` except the `make` variable is not displayed in the list of correlations. The command

```
. pwcorrs price weight mpg displ, sig var(2) sort
```

results in

```
                          Pearson Correlations
             |    price   weight
-------------+------------------
       price |   1.0000
             |
             |
      weight |   0.5386   1.0000
             |   0.0000
             |
       displ |   0.4949   0.8949
             |   0.0000   0.0000
             |
         mpg |  -0.4686  -0.8072
             |   0.0000   0.0000
             |
```

since the `var(2)` option says to display correlations of the first two variables in the list with all the variables in the list. The `sig` option produces the rows of significance levels of the correlations, all of which are 0 to four decimal places in this example, and the `sort` option specifies that after the first two variables, the "row variables" are to be displayed in sorted order.

If we enter the command

```
. pwcorrs price weight mpg displ, sig obs var(2) above(0.5) sp cwd sort
```

we get

```
                          Spearman Correlations
(n=74)
          |    price   weight
----------+------------------
    price |   1.0000
          |
          |       74
          |
   weight |            1.0000
          |   0.0000
          |       74       74
          |
    displ |            0.9054
          |   0.0010   0.0000
          |       74       74
          |
      mpg |  -0.5419  -0.8576
          |   0.0000   0.0000
          |       74       74
          |
```

which are the Spearman correlations because of the `sp` option, the number of nonmissing observations for each pair of variables is displayed because of the `obs` option, only those correlations above 0.5 in absolute value are displayed because of the `above(0.5)` option, and all cases are removed for all variables if that observation is missing for any variable (none are removed in this example).

We next consider the `hacomp3.dta` dataset which is composed of demographic, clinical, radiographic, and laboratory variables obtained on 96 persons with rheumatoid arthritis and 96 persons with osteoarthritis. In this group, we are studying two laboratory markers of joint damage, hyaluronic acid (`ha`) and cartilage oligomeric matrix protein (`comp`). There are a total of 80 variables so `pwcorrs` makes it easier to study the correlations.

First, a description of the dataset:

```
. describe
Contains data from hacomp3.dta
  obs:           192                          Hyaluron Collagen Matrix Protei
 vars:            80                          18 Dec 1996 12:08
 size:        37,248 (99.8% of memory free)
------------------------------------------------------------------------------
   1. ha          double %10.0g                 HA
   2. comp        double %10.0g                 COMP
   3. cibloc      str3   %9s                    Location of ELISA plate
   4. mosort      int    %8.0g                  Original sort Order
   5. plate       byte   %9.0g                  Plate #
   6. patkey      int    %10.0g                 Patient identifier
   7. encountr    byte   %8.0g                  Visit #
   8. specdate    int    %d                     Specimen date
   9. lastname    str14  %14s
  10. firstnam    str10  %10s
  11. age         float  %9.0g                  Age
  12. duration    float  %9.0g                  Disease duration
  13. followup    float  %9.0g                  Followup (years)
  (output omitted)
  80. hnar        byte   %9.0g                  Narrowing both hands additive
```

Now we get Spearman correlations for the variables `ha` and `comp` with all of the variables whose names start with the letters "a", "f", and any letter in the range "s" through "w":

```
. pwcorrs ha com _all, v(2) sort sel(a f s-w) sp
                          Spearman Correlations
          |       ha     comp
----------+------------------
       ha |   1.0000
     comp |   0.3848   1.0000
      age |   0.4940   0.1602
 am_stiff |   0.1675   0.1491
 anxindex |   0.0539  -0.0733
 fatigsca |   0.0673   0.0345
 followup |   0.0622   0.1599
 severity |   0.1855  -0.0135
      sex |   0.0978   0.1652
 sleepsca |   0.0999   0.0032
 smokenow |  -0.0079  -0.0391
 smokepst |   0.0117  -0.0154
  smokeve |   0.0204   0.0112
 specdate |  -0.0687   0.0183
  tjrhipl |   0.1161  -0.0021
  tjrhipr |  -0.0913  -0.0294
   tjrknl |   0.1900   0.0424
   tjrknr |   0.1902   0.1813
      wdi |   0.2440   0.1255
   weight |  -0.0919  -0.0369
 wpainsca |   0.0965  -0.0438
```

sg65 | Computing intraclass correlations and large ANOVAs

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

Stata has long offered the `loneway` command as a means of computing the intraclass correlation coefficient, and of performing a single-factor ANOVA with a large number of levels. However, `loneway` has some flaws: (1) its estimate of the intraclass correlation can be distorted in highly unbalanced analyses; (2) its estimate of the intraclass correlation is incorrect for weighted analyses; (3) its estimate of the reliability of a group-averaged score is incorrect; (4) for weighted analyses with missing data, it produces the wrong ANOVA table [*Editor's note: this bug has been fixed in the official updates on this STB disk*]; and (5) it is extremely slow. The last point may be critical because simulation might be the best way to learn about the sampling behavior of an intraclass correlation.

This insert offers three commands for improved calculation of intraclass correlations, including a replacement for `loneway`. The new commands also offer several variations on the basic ANOVA approach to estimating intraclass correlations. We begin with a review of the main features of that approach.

The random-effects ANOVA model

The intraclass correlation ρ has a long history of application, including studies of familial resemblance in biology, and of reliability of measurements in psychology and related disciplines. The oldest estimator of ρ is based on computing the ordinary Pearson r over all possible pairs of observations within a subgroup. Karlin et al. (1981) present a class of such pairwise estimators, a class that includes the maximum likelihood (under normality) estimator as a special case. Here, we concentrate on the ANOVA approach, where estimation of ρ is viewed as a variance components problem. The notation extends that of Donner (1986), who provides a very useful review of the subject.

Suppose we have a response variable y measured on each of n_i elements within each of k groups or classes. We adopt the model

$$y_{ij} = \mu + a_i + \varepsilon_{ij} \qquad i = 1, 2, \ldots, k; \qquad j = 1, 2, \ldots, n_i \tag{1}$$

where the a_i and the ε_{ij} are independent random variables with $E(a_i) = E(\varepsilon_{ij}) = 0$, $\mathrm{Var}(a_i) = \sigma_a^2$, and $\mathrm{Var}(\varepsilon_{ij}) = \sigma_\varepsilon^2$. Hence, $E(y_{ij}) = \mu$ and $\mathrm{Var}(y_{ij}) = \sigma_a^2 + \sigma_\varepsilon^2$. This is, of course, a random-effects analysis of variance model, where often it is also assumed that all random terms have Gaussian distributions. In studies of familial resemblance, the classes might be k families selected at random, and the i-th family has n_i members; the n_i perhaps vary widely across families. In studying the reliability of ratings or other assessments, one might have a random sample of k individuals or stimuli (the classes), each assessed by a panel of raters or judges. In this case, one would typically have the same number of ratings per stimulus: $n_1 = \cdots = n_k = n$. In any event, the total number of observations is $N = \sum_i n_i$.

Consider now a more general version of model (1), in which observation y_{ij} is in fact the mean of w_{ij} independent values of $y = \mu + a_i + \varepsilon_{ijk}$, with the a_i and ε_{ijk} as just described. Then model (1) is unchanged except that we now have

$\text{Var}(\varepsilon_{ij}) = \sigma_\varepsilon^2/w_{ij}$, so that $E(y_{ij}) = \mu$ and $\text{Var}(y_{ij}) = \sigma_a^2 + \sigma_\varepsilon^2/w_{ij}$. This is the weighting scheme presumed by Stata's `aweight` option; the unweighted version of model (1) is just the case where $w_{ij} = 1$ for all i and j.

Now define

$$\theta = \sigma_a^2/\sigma_\varepsilon^2 \quad \text{and} \quad \rho = \frac{\sigma_a^2}{\sigma_a^2 + \sigma_\varepsilon^2} = \frac{\theta}{1+\theta}$$

Under model (1), ρ equals the ordinary correlation between any two observations in the same class or group, i.e., the intraclass correlation. From this perspective, the central problem is to estimate a ratio of variance components (θ). The analysis of variance for this model would yield an among-groups variance (MS_a) and an error variance (MS_e):

$$\text{MS}_a = \sum_i w_{i\cdot}(\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2/(k-1) \quad \text{and} \quad \text{MS}_e = \sum_i \sum_j w_{ij}(y_{ij} - \bar{y}_{i\cdot})^2/(N-k)$$

where

$$w_{i\cdot} = \sum_j w_{ij}, \quad \bar{y}_{i\cdot} = \frac{\sum_j w_{ij} y_{ij}}{w_{i\cdot}}, \quad \text{and} \quad \bar{y}_{\cdot\cdot} = \frac{\sum_i w_{i\cdot}\bar{y}_{i\cdot}}{\sum_i w_{i\cdot}}$$

Straightforward calculations show that

$$E(\text{MS}_e) = \sigma_\varepsilon^2 \quad \text{and} \quad E(\text{MS}_a) = \sigma_\varepsilon^2 + g\sigma_a^2$$

where

$$g = \frac{\sum_i w_{i\cdot} - \sum_i w_{i\cdot}^2/\sum_i w_{i\cdot}}{k-1} \tag{2}$$

In the unweighted case, we have

$$w_{i\cdot} = n_i, \quad \sum_i w_{i\cdot} = N, \quad \text{and} \quad g = \frac{N - \sum_i n_i^2/N}{k-1} \tag{3}$$

Estimating rho

The core of the ANOVA method of estimating ρ is to equate MS_a and MS_e with their expected values, and solve for estimates of σ_ε^2 and σ_a^2, and hence of θ. This gives

$$\hat{\theta} = \frac{\text{MS}_a - \text{MS}_e}{g\text{MS}_e} = \frac{F^* - 1}{g} \tag{4}$$

where $F^* = \text{MS}_a/\text{MS}_e$ is the usual F-statistic. Then

$$\hat{\rho} = \frac{\hat{\theta}}{1+\hat{\theta}} = \frac{F^* - 1}{F^* - 1 + g} \tag{5}$$

Note that $F^* < 1$ is possible, in which case it is customary to set $\hat{\rho} = 0$. This is often referred to as *the* ANOVA estimate of ρ. Still, other possibilities must be considered, so define the function

$$r(x,a,b) = \max\left(\frac{x-a}{x-a+ab}, 0\right) \tag{6}$$

where all arguments are positive real numbers. Then the estimator given in (5) is just $\hat{\rho} = r(F^*, 1, g)$.

The unweighted, balanced case

To motivate some other possibilities, consider the unweighted, balanced version of model (1), where $w_{ij} = 1$ for all i and j, $n_1 = \cdots = n_k = n$, and $g = n$. If we add the usual normality assumption so that $a_i \sim N(0, \sigma_a^2)$ and $\varepsilon_{ij} \sim N(0, \sigma_\varepsilon^2)$, then F^* is distributed as a multiple of the central F distribution: $F^* = \text{MS}_a/\text{MS}_e \sim (1+n\theta)F(k-1, N-k)$. It is easy to show that (4) is a biased estimator of θ, and that an unbiased estimator would result if we instead used

$$\hat{\theta} = \frac{F^* - E(F)}{nE(F)} \qquad \text{where} \qquad E(F) = \frac{N-k}{N-k-2}$$

Starting with this estimate of θ, we would obtain

$$\hat{\rho} = \frac{\hat{\theta}}{1+\hat{\theta}} = r(F^*, E(F), n) \tag{7}$$

Also in this special case, exact confidence intervals for ρ are easily derived from the central F distribution. For confidence $1-\alpha$, the endpoints of the interval are $r(F^*, F_{1-\alpha/2}, n)$ and $r(F^*, F_{\alpha/2}, n)$ where F_p is the p-th quantile of $F(k-1, N-k)$. As $\alpha \to 1$, the endpoints converge to $r(F^*, F_{.5}, n)$, which suggests

$$\hat{\theta} = \frac{F^* - F_{.5}}{n} \quad \text{and} \quad \hat{\rho} = \frac{\hat{\theta}}{1+\hat{\theta}} = r(F^*, F_{.5}, n) \tag{8}$$

where $F_{.5} = F_{.5}(k-1, N-k)$. So, $\hat{\rho}$ in (8) is the center, in a sense, of a confidence interval for ρ at any α.

Thus, in the unweighted, balanced case, the estimators $r(F^*, 1, n)$, $r(F^*, F_{.5}, n)$, and $r(F^*, E(F), n)$ correspond to choosing 1, the median, or the mean as reference points in the $F(k-1, N-k)$ distribution. When k and n are large, those three points will be quite similar, but in small samples there may be useful distinctions among the resulting estimators of ρ. Also, the distribution of F^* is not generally proportional to central F outside this special case, so the nominal properties of the reference points $F_{.5}$ and $E(F)$ will then be at best approximate.

Finally, consider an application in which this special case often arises, estimating the reliability of assessment and measurement devices. In this situation, one has a set of n raters, judges, or measuring devices and, barring the occasional failure, the value of n is constant across the k objects assessed or measured. In this situation, ρ is the reliability of an individual rating or measurement, and one might well be interested in the reliability that would be attained if one were to use the mean of t measurements. (Typically one would pick $t = n$, but equations (9) and (10) below are valid for any $t > 0$.) If individual measures arise from model (1), then the mean $\bar{y}_{i\cdot}$ of t such measurements of a given object has variance $\mathrm{Var}(\bar{y}_{i\cdot}) = \sigma_a^2 + \sigma_\varepsilon^2/t$, and so the reliability is

$$\rho_t = \frac{\sigma_a^2}{\sigma_a^2 + \sigma_\varepsilon^2/t} = \frac{t\theta}{1+t\theta} \tag{9}$$

This is the so-called "Spearman–Brown prophecy formula," which may also be written in the form

$$\rho_t = \frac{t\rho}{1+(t-1)\rho} \tag{10}$$

Equation (10) can be found in Winer (1962, 127) and Nunnally (1978, 243), among many other sources.

To estimate ρ_t, one merely substitutes $\hat{\theta}$ for θ in (9), or $\hat{\rho}$ for ρ in (10). Hence, if one estimates ρ as $\hat{\rho} = r(x, a, b)$, then the corresponding estimate of ρ_t must be

$$\hat{\rho}_t = \frac{t\, r(x, a, b)}{1+(t-1)r(x, a, b)} = r(x, a, b/t)$$

For the common choice $\hat{\rho} = r(F^*, 1, n)$ and $t = n$, we obtain $\hat{\rho}_n = r(F^*, 1, 1) = (F^* - 1)/F^*$. $\hat{\rho}_n$ can be viewed as the reliability of the group means $\bar{y}_{i\cdot}$, an interpretation not generally available in unbalanced designs.

New commands for intraclass correlations

The flaws in `loneway` can now be stated succinctly: (1) The reported $\hat{\rho}$ is not the usual choice $r(F^*, 1, g)$ but is instead $r(F^*, 1, N/k)$, and N/k can be far from g in weighted or unbalanced analyses; (2) when the user requests the `aweight` option, the reported $\hat{\rho}$ is incorrect; (3) the reported $\hat{\rho}_t$ is inconsistent with both the Spearman–Brown formula and the incorrect version of that formula given in the Reference Manual; (4) when there are `aweights` and missing values of y or the group variable, the reported values of MS_a and MS_e are incorrect [*Editor's note: this bug has been fixed in the official updates on this STB disk*]; and (5) execution time is much too long, especially when $k \leq 375$.

Before demonstrating these flaws, we introduce a command to replace `loneway`. The syntax of `l1way` is

`l1way` *response_var group_var* [*weight*] [`if` *exp*] [`in` *range*] [`, center(`*Fpos*`) ems`]

As with `loneway`, `aweights` are allowed. For compatibility with `loneway`, by default `l1way` returns $\hat{\rho} = r(F^*, 1, N/k)$, where $r()$ is defined by (6), above. The option `ems` causes `l1way` to replace N/k with g in calculating $\hat{\rho}$, where g is given by (2) above. The option `center()` selects a value for the second argument of the function $r()$, i.e., a reference point in the $F(k-1, N-k)$ distribution. The argument *Fpos* may be `1` (the default), or `med` or `mean` which select $F_{.5}$ or $E(F)$ for the relevant F distribution as the reference point.

Consider an example generated from model (1) with $\sigma_\varepsilon^2 = 1$ and $\rho = 0.6$:

```
. use ex1

. list
             y     group          w
  1.  1.398982         1          1
  2.         .         1          1
  3.         .         2          1
  4.  -.853521         6          1
     (output omitted)
 21.         .         6          1
 22.  1.277835         1          1
 23.         .         5          1
 24.  .5526405         5        100

. tab group, summ(y)
Group       |     Summary of Response variable
variable    |        Mean   Std. Dev.       Freq.
------------+------------------------------------
          1 |   1.6333718   .51446953           3
          2 |   .67936756   1.3004536           3
          3 |  -1.2514399   .83630143           3
          4 |  -1.0245926   .81025697           4
          5 |  -.46231361   .87985187           3
          6 |  -.57802446   .24229907           3
------------+------------------------------------
      Total |  -.21239402   1.2476196          19
```

Note that some y values are missing so that $N = 19$ and $k = 6$. First, fit an unweighted version of model (1):

```
. loneway y group
             One Way Analysis of Variance for y: Response variable
    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between group        19.072222       5    3.8144444      5.54     0.0060
Within group         8.9457627      13    .68813559
------------------------------------------------------------------------
Total                28.017985      18    1.5565547

                                        R-squared = 0.6807
                      Estimated SD of group effect = .93189008
                         Estimated SD within group = .82953938
                           Intra-group correlation = 0.5893 (SD = 0.1387)

   Estimated reliability of a group-averaged score = 0.8179
                             Conservative t-deflator = 1.5174

. l1way y group, ems
```

(*The same ANOVA table reported by* `loneway`)

```
Intra-group r = 0.5899
Estimated reliability of a group mean (n=3.16) = 0.8196
```

`loneway` reports $\hat{\rho} = r(5.54, 1, N/k) = 0.5893$, whereas `l1way` (with the `ems` option) reports $\hat{\rho} = r(5.54, 1, g) = 0.5899$. The two values differ little here because the design is nearly balanced so that $N/k = 3.167$ is near $g = 3.158$.

Now examine the value $\hat{\rho}_t = 0.8179$ reported by `loneway`. Recall that $\hat{\rho}_t$ is the estimated reliability of the mean of t values of y: For what value of t is $\hat{\rho}_t = 0.8179$? In a balanced design, one might suppose that $t = n$, but here there is no single n. The Reference Manual claims that (in the present notation) $t = N/k$ and that $\hat{\rho}_{N/k} = (N/k)\hat{\rho}/(1 + (N/k)\hat{\rho}) = 3.167 \times 0.5893/(1 + 3.167 \times 0.5893)$. This proves to be 0.6511 which differs from the reported 0.8179. And if $\hat{\rho}_{N/k} = 0.8179$, $\hat{\rho} = 0.5893$ and this formula were correct, then working backward would give $N/k = 7.622$ as the "average group size," which makes little sense given that $\max\{n_i\} = 4$. In fact, the Manual's formula for $\hat{\rho}_t$ is incorrect; see (10) above.

`l1way` has used (10) to obtain $\hat{\rho}_g = r(5.54, 1, 1) = 0.8196$; the value of g is shown after `n=` in the output. A sensible interpretation in this case is that an average of 3 values of y from each group would have reliability near 0.82. Since 3 is near all of the n_i, 0.82 can be viewed as an estimate of the reliability of the cell means $\bar{y}_{i\cdot}$; in weighted or more unbalanced situations, the analogous interpretation may not hold. Finally, note that without the `ems` option, `l1way` would calculate $\hat{\rho} = r(F^*, Fpos, N/k)$ and $\hat{\rho}_{N/k} = r(F^*, Fpos, 1)$.

Next, consider fitting model (1) using `w` as the weight variable. `w` consists entirely of the value 1 except for observation 24, where `w` equals 100. That is, we suppose that the 24th value of y is actually the mean of 100 y values (an extreme example, admittedly). `l1way` produces

```
. l1way y group [aw=w], ems

            One Way Analysis of Variance for y: Response variable

    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between group        4.0902243      5    .81804486      5.53     0.0060
Within group           1.92339     13    .14795308
------------------------------------------------------------------------
Total                6.0136143     18    .33408968

Intra-group r = 0.4352
Estimated reliability of a group mean (n=5.88) = 0.8191
```

and the built-in command `oneway` gives the same ANOVA table, but `loneway` yields

```
. loneway y group [aw=w]

            One Way Analysis of Variance for y: Response variable

    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between group         3.682831      5    .73656621      4.11     0.0185
Within group         2.3307833     13    .17929102
------------------------------------------------------------------------
Total                6.0136143     18    .33408968

                     (output omitted)
```

The problem, it seems, is that the dataset contains observations where y is missing but w is not [*Editor's note: this bug has been fixed in the official updates on this STB disk*]:

```
. replace w = . if y == .
(5 real changes made, 5 to missing)

. loneway y group [aw=w]

            One Way Analysis of Variance for y: Response variable

    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between group        4.0902243      5    .81804486      5.53     0.0060
Within group           1.92339     13    .14795308
------------------------------------------------------------------------
Total                6.0136143     18    .33408968

                                    R-squared = 0.6802
                     Estimated SD of group effect = .43143552
                        Estimated SD within group = .38464669
                          Intra-group correlation = 0.5885 (SD = 0.1389)

   Estimated reliability of a group-averaged score = 0.8174
                          Conservative t-deflator = 1.5169
```

Note that because the weighted and unweighted analyses yield nearly the same F^*, `loneway` must produce nearly the value of $\hat{\rho} = r(F^*, 1, N/k)$ in the two cases. This is tantamount to claiming that it doesn't matter whether the 24th observation provides one y or the mean of 100 y values, which seems peculiar. To paint the problem more boldly, suppose that in fact every observation provides a y that is the mean of 100 individual y values from model (1):

```
. gen W = 100

. l1way y group [aw=W], ems

            One Way Analysis of Variance for y: Response variable

    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between group        19.072222       5    3.8144444      5.54     0.0060
Within group         8.9457627      13    .68813559
------------------------------------------------------------------------
Total                28.017985      18    1.5565547

Intra-group r = 0.0142
Estimated reliability of a group mean (n=315.79) = 0.8196
```

But, the command `loneway y group` produces *exactly* the same output as `loneway y group [aw=W]`. In particular, `loneway` claims that $\hat{\rho} = 0.5893$ whether each y is the mean of 1 or of 100 values; plainly, this is incorrect.

Note that because Stata scales the weights so that $\sum_i w_{i\cdot} = N$, the ANOVA table is the same in the unweighted and weighted analyses. In the unweighted case, working from model (1) gives $g = 3.158$, $\widehat{\sigma_\varepsilon^2} = \text{MS}_e = 0.6881$, and $\widehat{\sigma_a^2} = (3.8144 - 0.6881)/3.158 = 0.9900$. Then, $\hat{\theta} = 0.9900/0.6881 = 1.4387$, and $\hat{\rho} = 1.4387/(1 + 1.4387) = 0.5899$, as reported by `l1way`. But if each y is the mean of 100 values from model (1) as the weight variable `W` asserts, then $\text{MS}_e = 0.6881$ is actually an estimate of $\sigma_\varepsilon^2/100$, and so $\widehat{\sigma_\varepsilon^2} = 100\text{MS}_e = 68.81$. Then, $\hat{\theta} = 0.9900/68.81 = 0.014387$, and $\hat{\rho} = 0.014387/(1+0.014387) = 0.0142$, also as reported by `l1way`, just above. Note that $g = 315.79$ in this weighted analysis, so that `l1way`'s $\hat{\rho}_t = 0.8196$ is just an estimate that group means based on 316 individual y values would have reliability near 0.82.

Consider one more extreme example, this time to demonstrate the effects of imbalance in unweighted analyses. For a fixed value of N, a worst case occurs when one group is of size $N - k + 1$ and the remaining groups are of size 1, for then $g = 2 - k/N$. Data set `ex2.dta` creates this situation with $k = 6$, $n_1 = \cdots = n_5 = 1$, $n_6 = 100$, and y generated from model (1) with $\sigma_\varepsilon^2 = 1$ and $\rho = 0.6$. Given 99 df for estimating σ_ε^2, we might expect a sensible estimate of ρ, even though 5 of the 6 groups have $n = 1$:

```
. use ex2, replace

. l1way y group, ems

        One Way Analysis of Variance for y: Unweighted Response variable

    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between group        25.639151       5    5.1278303      5.15     0.0003
Within group         98.602474      99    .99598458
------------------------------------------------------------------------
Total                124.24162     104     1.194631

Intra-group r = 0.6810
Estimated reliability of a group mean (n=1.94) = 0.8058

. loneway y group
```

(*The same ANOVA table reported by* `l1way`)

```
                          Intra-group correlation = 0.1916 (SD = 0.0648)

   Estimated reliability of a group-averaged score = 0.2211
                          Conservative t-deflator = 4.3667
```

We have here $N/k = 17.5$ and $g = 1.94$, and `loneway` computes $\hat{\rho} = r(5.15, 1, 17.5) = 0.1916$ whereas `l1way` returns $\hat{\rho} = r(5.15, 1, 1.94) = 0.6810$. That is, $E(\text{MS}_a) = \sigma_\varepsilon^2 + 1.94\sigma_a^2$, but `loneway` acts as though $E(\text{MS}_a) = \sigma_\varepsilon^2 + 17.5\sigma_a^2$, and so gives an estimate far from the true ρ. `l1way` estimates that group means based on 2 values of y would have reliability near 0.81, whereas `loneway` appears to claim that the means based on 17.5 values of y would have reliability near 0.22. This latter value is wildly incorrect, because plugging $t = 17.5$ and $\hat{\rho} = 0.1916$ into equation (10) gives $\hat{\rho}_t = 0.8058$, the same value reported by `l1way`. Note also that the observed means $\bar{y}_{i\cdot}$ will not behave much like the means of 17–18 y values.

On computational efficiency

`loneway` requires more memory for temporary variables than `l1way`, and under Stata 5.0 for Windows `l1way` is about 2–2.5 times faster than `loneway`. Still, it is possible to do much better, at least in some situations. First, if the number of groups

k is less than 376 then the built-in command `oneway` can be used to do most of the work. The command `iclassr` takes just that approach; the syntax is

`iclassr` *response_var group_var* [*weight*] [`if` *exp*] [`in` *range*] [`, center(`*Fpos*`) ems noisily`]

The `noisily` option causes display of the ANOVA table which is otherwise suppressed; the other options are the same as those of `l1way`. For $k < 376$, `iclassr` does exactly the same job as `l1way`, but faster. How much faster? With the `ems` option, `iclassr` is perhaps only 20–25% faster than `l1way`. But when N/k is near g, it is reasonable to omit the `ems` option, and then `iclassr` is 2–3 times faster than `l1way`, or 4–6 times faster than `loneway`.

Second, suppose that $n_1 = \cdots = n_k = n = 2$. This is a common situation in reliability problems, where one has assessments of k objects and the goal is to estimate the inter-rater or inter-device reliability ρ for pairs of raters or measuring devices. Typically, the data would be unweighted, and then $g = n = 2$ which permits very rapid calculation of $\hat{\rho}$. The command `iclassr2` caters to this situation; the syntax is

`iclassr2` *response_var1 response_var2* [`if` *exp*] [`in` *range*] [`, center(`*Fpos*`)`]

`loneway`, `l1way`, and `iclassr` expect a single response variable y and a group or class variable ("long format"), but `iclassr2` expects paired values of y stored in two response variables ("wide format"). Weights are not permitted, and `iclassr2` enforces $n_i = 2$ by performing casewise deletion (observations are ignored if either response variable is missing). The `center` option is the same as in `l1way` and `iclassr`, and there is no need for an `ems` option.

In addition to allowing wide format data, `iclassr2` provides speed. At $k = 375$, `iclassr` and `iclassr2` are about equally fast, being about 3 times faster than `l1way` and at least 5–6 times faster than `loneway`. When k is large (say, $k \geq 1000$), `iclassr2` can be about 12 times faster than `loneway`.

Remarks

1. Because N/k can be far from g in unbalanced or weighted analyses, the intraclass correlation $\hat{\rho} = r(F^*, 1, N/k)$ reported by `loneway` can be a poor estimator, and one should instead compute $\hat{\rho} = r(F^*, 1, g)$.

2. But, it may be that $g \approx N/k$, e.g., in an unweighted analysis of a nearly balanced design. $\hat{\rho} = r(F^*, 1, N/k)$ and $\hat{\rho} = r(F^*, 1, g)$ will then be numerically similar, but there can be a large speed advantage in computing the former; omitting the `ems` option of `l1way` or `iclassr` gains that advantage. Most of the cost of g is for $\sum_i n_i^2/N$, which `loneway` stores in the global macro `S_3` but, interestingly, does not use to compute $\hat{\rho} = r(F^*, 1, g)$.

3. `l1way`, `iclassr`, and `iclassr2` omit some items reported by `loneway`. This includes the SD of $\hat{\rho}$ which `loneway` obtains from a formula that assumes a large k, and normally distributed y_{ij} in model (1). The SD is of little interest when ρ is far from 0, because of skew in the distribution. More importantly, one will often be better served by simulating the sampling properties of $\hat{\rho}$, and hence the emphasis on speed in this insert.

Saved Results

`l1way`, `iclassr`, and `iclassr2` each save $\hat{\rho}$ and $\hat{\rho}_t$ in global macros `S_1` and `S_2`, respectively. Following `iclassr`, `_result()` will hold the same contents as following the command `oneway`. `l1way` creates a similar situation by storing in scalars `S_1`–`S_6` the values that would be found in `_result(1)`–`_result(6)` following the command `oneway`; see [R] **oneway** for details.

Acknowledgment

This research was supported by a grant R01-MH54929 from the National Institute on Mental Health to Michael P. Carey.

References

Donner, A. 1986. A review of inference procedures for the intraclass correlation coefficient in the one-way random effects model. *International Statistical Review* 54: 67–82.

Karlin, S., E. C. Cameron, and P. T. Williams. 1981. Sibling and parent–offspring correlation estimation with a variable family size. *Proceedings of the National Academy of Sciences of the USA* 78: 2664–2668.

Nunnally, J. C. 1978. *Psychometric Theory*. 2d ed. New York: McGraw–Hill.

Winer, B. J. 1962. *Statistical Principles in Experimental Design*. New York: McGraw–Hill.

sg66 | Enhancements to the alpha command

Jeroen Weesie, Utrecht University, Netherlands, weesie@weesie.fsw.ruu.nl

This insert describes a rather extensive modification of Stata's `alpha` command, called `alpha2`. The main extensions that were made are

1. An option to display the effects of individual items. For each of the items, `alpha2` displays the item-test correlation, the item-rest correlation, the average inter-item covariance/correlation for the test scale excluding the item, and Cronbach's alpha.
2. Variable labels can be displayed in a compact format to ease the interpretation of the long item batteries.
3. An option to specify which items are to be reversed.
4. Options to characterize the observations to be used in terms of the number of nonmissing values among the items.

As far as I know, `alpha2` is backward compatible with `alpha`. For example, it returns the same results in the `S_#` macros. If the individual item effects are not requested, `alpha2` is (much) faster than `alpha` if listwise deletion is used, and somewhat slower otherwise.

We illustrate `alpha2` with the automobile data, `auto.dta`. However, we introduced some missing values in most of the variables and perturbated the `price` variable for illustration of several features of `alpha2`. First, I want to compare Cronbach's α for a standardized additive scale as computed by the standard `alpha` command, and by `alpha2`.

```
. alpha price hdroom rep78 trunk weight length turn displ gratio, std gen(X)

Scale = sum(standardized variables)
        rep78:  reversed
        gratio:  reversed

        Average interitem correlation:      0.5050
        Number of items in the scale:            9
        Scale Reliability Coefficient:      0.9018

. alpha2 price hdroom rep78 trunk weight length turn displ gratio, std

Test scale = mean(standardized items)
Reversed items: rep78 gratio

Average interitem correlation:      0.5050
Number of items in the scale:            9
Scale Reliability Coefficient:      0.9018
```

While the scale is actually rather satisfactory at first sight, we like to see whether all items fit the scale. This can be assessed informally with the new option `item`.

```
. alpha2 price hdroom rep78 trunk weight length turn displ gratio, std item

Test scale = mean(standardized items)

                        item-test     item-rest     inter-item
Item     |  Obs  Sign   correlation   correlation    correlation      alpha
---------+--------------------------------------------------------------------
price    |   61    +       0.3365        0.1748         0.6094       0.9258
hdroom   |   61    +       0.6959        0.5921         0.5214       0.8971
rep78    |   69    -       0.5760        0.4447         0.5555       0.9091
trunk    |   62    +       0.7796        0.6878         0.4990       0.8885
weight   |   61    +       0.9183        0.8835         0.4654       0.8744
length   |   56    +       0.9146        0.8791         0.4652       0.8744
turn     |   60    +       0.8796        0.8300         0.4756       0.8788
displ    |   64    +       0.9113        0.8676         0.4642       0.8739
gratio   |   68    -       0.8031        0.7253         0.4914       0.8855
---------+--------------------------------------------------------------------
Test                                                    0.5050       0.9018
---------+--------------------------------------------------------------------
```

Note that `alpha2` has a table-like output format that is similar to the format used by, for example, SPSS. We use the standard term `test` to denote the additive scale, and `item` instead of variable. The number of nonmissing values of the items ranges from 56 to 69 (See column 2). Like `alpha`, `alpha2` estimates correlations (covariances) using all available data by default; see below for options to modify this behavior.

The third column specifies the direction in which an item variable entered the scale. Here, a - sign means that the item was reversed. The bottom output line in the table gives the (average) inter-item correlation and the alpha coefficient for a test scale based on all items. These numbers are, as they should be, identical to the compact output of `alpha2` if the option `item` was not specified. The other lines of output describe the effect of single item on the scale.

The fourth column gives the item-test correlations. Apart from the sign of the correlation for items that entered the scale in reversed order, these correlations are, or should be, the same numbers that would be obtained with the commands

```
. alpha price ... gratio, std gen(X)
. corr  price ... gratio, gen(X)
```

Typically, we like the item-test correlations to be roughly the same for all items. Item-test correlations may actually not be very adequate to detect items that fit poorly, because the poorly fitting items may distort the scale. Thus, it may be more useful to consider item-rest correlations (Nunnally 1978), i.e., the correlation between an item and the scale that is formed by all other items. These are shown in column 5. The average inter-item correlations (covariances) of all items, excluding one, are shown in column 6. Finally, column 7 shows Cronbach's α coefficient for the test scale that consist of all but one item.

In this case, the `price` item does not seem to fit well in the scale on all respects. The item-test and item-rest correlations of `price` are much lower than for the other items. The average inter-item correlation increases substantially by removing the item `price`; apparently, `price` does not correlate so strongly with the other items. Finally, we see that Cronbach's α coefficient would increase from .9018 to .9258 if the item `price` is dropped—for well-fitting items we would of course expect that α decreases by "shortening" by test.

Note that .9258 would also be the obtained as the α statistic if we issue the `alpha2` command on the item-list that excludes `price`.

```
. alpha2 hdroom rep78 trunk weight length turn displ gratio, std item

Test scale = mean(standardized items)

                        item-test     item-rest     inter-item
Item     |  Obs  Sign   correlation   correlation    correlation      alpha
---------+--------------------------------------------------------------------
hdroom   |   61    +       0.7013        0.5951         0.6474       0.9278
rep78    |   69    -       0.5767        0.4386         0.6966       0.9414
trunk    |   62    +       0.7940        0.7024         0.6161       0.9183
weight   |   61    +       0.9298        0.8980         0.5717       0.9033
length   |   56    +       0.9251        0.8905         0.5772       0.9053
turn     |   60    +       0.8862        0.8355         0.5899       0.9097
displ    |   64    +       0.9244        0.8823         0.5722       0.9035
gratio   |   68    -       0.8229        0.7488         0.6078       0.9156
---------+--------------------------------------------------------------------
Test                                                    0.6094       0.9258
---------+--------------------------------------------------------------------
```

The variable names for the automobile data are actually reasonably informative. However, the items in long item batteries that are commonly used to measure personality traits, attitudes or values, etc., are usually named with indexed names such as `item12_1`, `item12_2`, etc. The content of the items used are often too closely related to make meaningful summaries in the 8 characters possible. Thus, I included an output option `label` that forces `alpha2` to produce the same statistical information in a more compact format that leaves room to includes variable (item) labels. In this compact format `alpha2` has to exclude the number of nonmissing values of the items, it displays the statistics in fewer digits, and uses the somewhat cryptic headers:

`it-cor`	=	item-test correlation
`ir-cor`	=	item-rest correlation (rest = test-item)
`ii-cor`	=	average inter-item correlation/covariance

```
. alpha2 hdroom rep78 trunk weight length turn displ gratio, std item label

Test scale = mean(standardized items)

Items    | S  it-cor  ir-cor   ii-cor   alpha   label
---------+-------------------------------------------------------------------
hdroom   | +   0.701   0.595    0.647   0.928   Headroom (in.)
rep78    | -   0.577   0.439    0.697   0.941   Repair Record 1978
trunk    | +   0.794   0.702    0.616   0.918   Trunk space (cu. ft.)
weight   | +   0.930   0.898    0.572   0.903   Weight (lbs.)
```

```
length   | +   0.925   0.890    0.577   0.905   Length (in.)
turn     | +   0.886   0.836    0.590   0.910   Turn Circle (ft.)
displ    | +   0.924   0.882    0.572   0.903   Displacement (cu. in.)
gratio   | -   0.823   0.749    0.608   0.916   Gear Ratio
---------+--------------------------------------------------------------------
Test                           0.609   0.926   mean(standardized items)
---------+--------------------------------------------------------------------
```

Finally, similar to `alpha`'s factor-analytic method to determine "automatically" the sign in which items are entered in the test scale, `alpha2` has an option `reverse` to set the signs manually.

Syntax

`alpha2` *varlist* [`if` *exp*] [`in` *range*] [`,` `asis` `coo` `detail`
`gen(`*newvar*`)` `item` `label` `min(`#`)` `reverse(`*varlist*`)` `std`]

Options

`asis` indicates that the sense (sign) of each item should be taken as presented in the data. The default is to empirically determine the sense (using principal components factor analysis) and reverse the scorings for any that enter negatively. Note that subscales use the same directions of the items. Use `reverse` (see below) to manually specify the signs (directions) of the items without transforming the data.

`coo` specifies that cases with missing values should be deleted "listwise". The default is pairwise computation of covariances/correlations.

`detail` specifies that the matrix of inter-item covariance or correlation should be displayed if `std` is specified. If different numbers of observations have been used to compute correlations or covariance, these are displayed as well.

`gen(`*newvar*`)` specifies that the scale constructed from *varlist* is to be stored in *newvar*. Unless `asis` is specified, the sense of items entering negatively is automatically reversed. If `std` is also specified, the scale is constructed using standardized (mean 0, variance 1) values of the individual items. Unlike most Stata commands, `gen()` does not employ casewise deletion. A score is created for every observation for which there is a response to at least one item (one variable in *varlist* is not missing). The additive score is divided by the number of items over which the sum is calculated.

`item` specifies that item-test and item-rest correlations and the effects of removing an item from the scale are displayed.

`label` requests that the detailed output table is displayed in a compact format that enables the inclusion of variable labels.

`min(`#`)` specifies that only cases with at least # observations are included in the computations. `coo` is a shorthand to `min(`k`)`, with k the number of variables in *varlist*.

`reverse(`*varlist*`)` specifies that the signs (directions) of the variables (items) in *varlist* should be reversed.

`std` specifies that the items in the scale are to be standardized (mean 0, variance 1) before summing.

Saved Results

Like `alpha`, `alpha2` returns the average inter-item covariance/correlation in `S_4`, the number of items in `S_5`, and the alpha statistic in `S_6`.

Reference

Nunnally, J. C. 1978. *Psychometric Theory*. 2d ed. New York: McGraw Hill.

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 sampling
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified
srd	robust methods & statistical diagnostics		

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

International Stata Distributors

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

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

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

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

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

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

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

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