STATA TECHNICAL BULLETIN

September 1998

STB-45

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.

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

dm60 Digamma and trigamma functions

Joseph Hilbe, Arizona State University, hilbe@asu.edu

The `digamma` and `trigamma` functions are the first and second derivatives respectively of the log-gamma function. Several numeric approximations exist, but the ones used in `digamma` and `trigamma` seem particularly accurate (see Abramowitz and Stegun 1972).

I have prepared two versions of each program: one where the user specifies a new variable name to contain the calculated values from those listed in an existing variable; the other, an immediate command for which the user simply types a number after the command.

Syntax

`digamma` *varname* [`if` *exp*] [`in` *range*] `,` `generate(`*newvar*`)`

`trigamma` *varname* [`if` *exp*] [`in` *range*] `,` `generate(`*newvar*`)`

`digami` *#*

`trigami` *#*

Examples

We find the values of the digamma and trigamma functions for $x = 1, 2, \ldots, 10$:

```
. set obs 10
obs was 0, now 10
. gen x=_n
. digamma x, g(dx)
. trigamma x, g(tx)
. list
               x          dx          tx
  1.           1   -.577216    1.645019
  2.           2   .4227843    .6449341
  3.           3   .9227843    .3949341
  4.           4   1.256118     .283823
  5.           5   1.506118     .221323
  6.           6   1.706118     .181323
  7.           7   1.872784    .1535452
  8.           8   2.015641     .133137
  9.           9   2.140641     .117512
 10.          10   2.251753    .1051663
```

Reference

Abramowitz, M. and I. Stegun. 1972. *Handbook of Mathematical Functions*. New York: Dover.

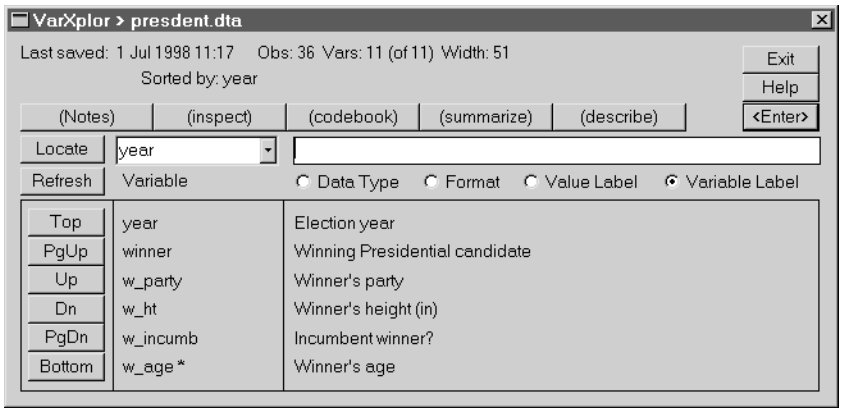
dm61 A tool for exploring Stata datasets (Windows and Macintosh only)

John R. Gleason, Syracuse University, loesljrg@ican.net

Surely, `describe` is one of the first commands issued by new Stata users, and one of the first invoked by Stata veterans when they begin to examine a new dataset. Simple forms of the `list`, `summarize`, `tabulate`, and `graph` commands enjoy a similar status, and for good reason. These commands are central to the processes of learning about Stata (for new users) and of understanding an unfamiliar dataset (for all users). This insert presents `varxplor`, a tool that might be viewed as an interactive form of the `describe` command, in much the same sense that Stata's Browse window is an interactive form of the `list` command. `varxplor` resembles Stata's Variables window, except that more information is available, a given variable can be more easily located, and more can be done once a variable is located. (Note that `varxplor` uses the dialog programming features new in Version 5.0, for Windows and Macintosh only, and so is restricted to those platforms.)

We begin by demonstrating some features of `varxplor` and delay presentation of its formal syntax. First, `varxplor` provides much of the same information as `describe`. Most importantly, there is a Variables window that shows variable names, variable labels, value label assignments, data types, and output formats. Unlike the `describe` command, only one of those categories of information is visible at a time; but in return, one gains the ability to scroll about in the list of variables in either dataset order or alphabetical order, to rapidly locate any variable, and, with a single mouse click, to execute commands such as `summarize` or `list` on that variable. The set of commands that can be so executed is configurable; better still, `varxplor` has its own rendition of Stata's command line, from which most Stata commands can be issued.

To illustrate, `use` the dataset `presdent.dta` (included with this insert), issue the command `varxplor`, and a dialog box resembling Figure 1 will appear. The top portion of the dialog displays general information about `presdent.dta`: a count of variables and observations, date last saved, how sorted, and so on. The Variables window is at the bottom. It is divided into three sections: the middle one shows a list of current variable names; a multi-purpose display panel is located to the right of the names; and to the left is a stack of six scroll buttons used to navigate the variable list. The contents of the multi-purpose display are controlled by the radio buttons above it. To show the data types of variables rather than variable labels, for example, click the **Data Type** button and then click any of the six scroll buttons, or the **Refresh** button (to avoid scrolling the variable list).


VarXplor > presdent.dta

Last saved: 1 Jul 1998 11:17 Obs: 36 Vars: 11 (of 11) Width: 51
Sorted by: year

Locate: year

Variable: Data Type / Format / Value Label / [✓ Variable Label]

Variable	Variable Label
year	Election year
winner	Winning Presidential candidate
w_party	Winner's party
w_ht	Winner's height (in)
w_incumb	Incumbent winner?
w_age *	Winner's age

Figure 1.

If more than six variables are available, the scroll buttons traverse the variable list in an obvious manner. Clicking **Dn**, for example, shifts downward one position in the variable list, that is, moves the variables one position upward in the Variables window. Clicking **PgDn** shifts six positions downward in the variable list, and clicking **Bottom** shifts so that the last variable appears in the bottom position of the Variables window. The topmost variable name also appears in the Locator window (actually, edit box) beside the **Locate** button. Any variable can be located by typing its name into that window or by selecting its name from the dropdown list triggered by the downarrow at the right edge of the window. Clicking **Locate** then shifts so that the variable named in the Locator window appears in the Variables window below, in the top position if possible.

Immediately above the Locator window is a row of five launch buttons: By default, clicking a launch button issues the command named on the button, for the variable whose name appears in the Locator window. In Figure 1, for example, clicking the button marked **(inspect)** would issue the command `inspect year` so that

```
. inspect year
year:  Election year                           Number of Observations
--------------------                                               Non-
                                               Total   Integers    Integers
|                  #           Negative            -          -           -
|  #   #   #   #   #           Zero                -          -           -
|  #   #   #   #   #           Positive           36         36           -
|  #   #   #   #   #                           -----      -----       -----
|  #   #   #   #   #           Total              36         36           -
|  #   #   #   #   #           Missing             -
+----------------------                        -----
1856              1996                            36
  (36 unique values)
```

appears in the Results window. Clicking the **(Notes)** button would in this case have no effect, because the variable `year` has no notes. But variable `w_age` does have notes; that is why its name is suffixed by '`*`' in the Variables window. So, placing the name `w_age` in the Locator window and clicking the **(Notes)** button prints the following in the Results window:

```
w_age:
  1.  Winner´s ages taken from 1995 Universal Almanac
```

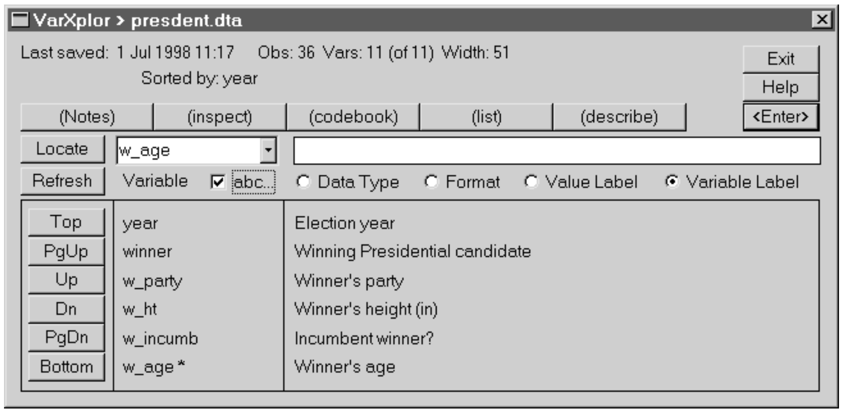
The parentheses on the launch buttons are meant to remind you that the command name between them is merely the current assignment. Each launch button can be reconfigured as desired (see below). If the launch buttons prove to be inadequate, `varxplor` has its own command line, the wide edit box positioned to the right of the Locator window. Most Stata commands can be issued from `varxplor` by typing them into that edit box and pressing the *Enter* key on the keyboard, or clicking the *Enter* button at the right end of the edit box.

Customizing varxplor

The syntax of the `varxplor` command is

`varxplor` [*varlist*] [, `alpha b1(`*cmdstr*`) b2(`*cmdstr*`) b3(`*cmdstr*`) b4(`*cmdstr*`) b5(`*cmdstr*`)`]

If a *varlist* is present, only those variables can be visited by `varxplor`; otherwise, all variables are available.

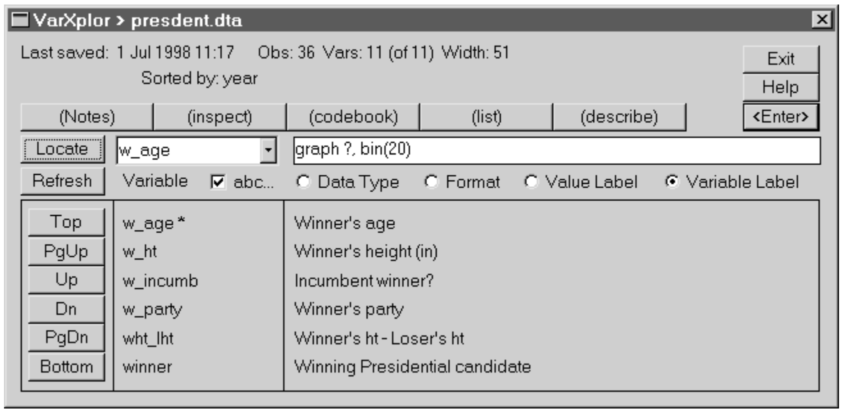


```
Last saved: 1 Jul 1998 11:17   Obs: 36  Vars: 11 (of 11)  Width: 51
Sorted by: year
Locator: w_age
Variable  [✓ abc...]   ( ) Data Type  ( ) Format  ( ) Value Label  (•) Variable Label

year        Election year
winner      Winning Presidential candidate
w_party     Winner's party
w_ht        Winner's height (in)
w_incumb    Incumbent winner?
w_age*      Winner's age
```

Figure 2.

If the `alpha` option is present, `varxplor` creates an alphabetized list of variable names when it is launched, and shows a check box labeled `abc...` just below the Locator window (see Figure 2). Checking that box causes `varxplor` to begin traversing the variable list in alphabetic order, rather than in dataset order. The change takes place at the next screen update, caused by clicking a scroll button or the **Refresh** or **Locate** buttons. In Figure 2, for example, the variables are shown in dataset order, but the `abc...` box has just been checked. Clicking the **Locate** button would switch to alphabetic order, with the variable `w_age` in the topmost position; Figure 3 shows the result. The `alpha` option also has a more profound, but less visible effect: Since an alphabetic variable list is at hand, the **Locate** operation finds variables using binary search, rather than linear search starting from the top of the list. As a result, locating a variable is much faster (whether or not the `abc...` box is checked) when the `alpha` option is given. Alphabetizing the variable list makes startup slower but pays off in faster locate operations, especially when there are many variables.



```
Last saved: 1 Jul 1998 11:17   Obs: 36  Vars: 11 (of 11)  Width: 51
Sorted by: year
Locator: w_age     graph ?, bin(20)
Variable  [✓ abc...]   ( ) Data Type  ( ) Format  ( ) Value Label  (•) Variable Label

w_age*      Winner's age
w_ht        Winner's height (in)
w_incumb    Incumbent winner?
w_party     Winner's party
wht_lht     Winner's ht - Loser's ht
winner      Winning Presidential candidate
```

Figure 3.

The options `b1(`*cmdstr*`)` ... `b5(`*cmdstr*`)` configure the five launch buttons, in left to right order. The argument *cmdstr* is effectively just a string to be passed to Stata's command processor; the first word of that string is used to decorate the button. More precisely, when a launch button is clicked it is as though *cmdstr* has been entered on Stata's command line, with the variable name in the Locator window substituted wherever a certain placeholder appears in *cmdstr*; the default placeholder is the character '`?`'. For example, to make Button 4 launch the `list` command with the `nolabel` option, start `varxplor` with

```
. varxplor, b4(list ?, nolabel)
```

(The fourth button in Figures 2 and 3 reflects this startup option.) The character '`?`' will be replaced with the variable name in the Locator window when Button 4 is clicked. Note that the default setup for Button 4 is equivalent to either `b4(summarize)` or `b4(summarize ?)`—they are the same because `varxplor` automatically appends '`?`' when '*cmdstr*' consists of a single word. To force a null variable list, use (for example) `b4(summarize ,)`; note the blank space preceding the comma.

Also note that certain commands cannot be assigned to a launch button. For example, an option such as `b5(graph ?, bin(20))` is unacceptable to Stata's parser because of the embedded parentheses. However, the desired effect can be obtained by typing '`graph ?, bin(20)`' on `varxplor`'s command line and then clicking the *Enter* button. Still other commands are unacceptable both on the command line and as launch button assignments. In particular, because of limitations in Stata's macros, the left-quote ('`` ` ``') and double-quote ('`"`') characters must be avoided. The presence of either character can trigger a variety of errors. See also Remark 2 below.

Remarks

1. `varxplor` creates its list of variable names at startup, and that list cannot be updated while `varxplor` is active. So, while it is possible to use `varxplor`'s command line to create new variables, they will not appear on the variable list and cannot be visited without exiting `varxplor`. The variables counter at the top of the dialog will however be updated appropriately.
2. For the same reason, deleting or moving a variable would render `varxplor`'s variable list invalid, without warning. Hence, the Stata commands `drop`, `keep`, `move`, and `order` are forbidden in a button *cmdstr* and from `varxplor`'s command line. There are ways to defeat this prohibition, but they cannot be recommended.
3. As mentioned above, the variables counter at the top of the dialog will be updated if a new variable is created. Similar remarks apply to the other pieces of information shown there. For example, the `Sorted by:` item will respond to uses of the `sort` command from a launch button or the command line.
4. The **Locate** command accepts wildcards. For example, placing '`w_*`' in the Locator window and clicking **Locate** will search for variable names that begin with the characters '`w_`'.
5. The default launch button assignments are defined by a local macro near the top of the `varxplor.ado` file. Any other set of five one word Stata commands can be used as the defaults. The default variable placeholder character is defined by a nearby global macro; that too can be changed, if necessary.
6. One default button assignment is `Notes`, which displays notes associated with the variable in the Locator window. Observe that `Notes` is not Stata's `notes` command, but `varxplor`'s own internal routine for displaying notes. Also, it might seem curious that another launch button defaults to the `describe` command. This is because the contents of `varxplor`'s Variables window cannot be directly copied to the `Results` screen, and hence to a log file; invoking `describe` solves that problem.
7. Rather than typing `varxplor` from the command line, some users may prefer to click a menu item or press a keystroke combination. The included file `vxmenu.do` arranges for that. Typing `do vxmenu` on Stata's command line adds an item named **Xplor** to Stata's top-level menu. Afterward, one mouse click launches `varxplor`; Windows users can also use the `Alt-X` keystroke shortcut. To make this arrangement permanent, add the line `do vxmenu` to the file `profile.do`; alternatively, copy the contents of `vxmenu.do` into `profile.do`. (See [U] **5.7, 6.7,** or **7.6 Executing commands every time Stata is started**.)

Acknowledgment

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

dm62 Joining episodes in multi-record survival time data

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

More complicated survival time data will usually contain multiple records (observations) per subject. This may be due to changes in covariates, to gaps in observations, and to recurrent events. Sometimes, multiple records are used while a single record could be used. Consider the following two cases:

```
          entry    survival      failure/
id        time     time          censor          x1      x2

112        0        10             0              3       0
112       10        14             1              3       0

117        5        12             0              0       1
117       12        17             0              0       1
117       17        22             0              0       1
```

The subject with `id==112` is described with 2 observations. In her first episode, the subject was at risk from time 0 to time 10, and was then censored. She re-entered at time 10, i.e., immediately after the end of the first episode, and failed at time 14. Note that her covariates x1 and x2 did not vary between the two episodes. This data representation is actually less economical than one in which subject 117 is described by a single observation as shown below. Case 117 consisted of three records that could also be joined into a single record.

```
        entry    survival     failure/
id      time     time         censor        x1      x2

112       0        14           1            3       0
117       5        22           1            0       1
```

So, when is it possible to join two episodes E1 and E2? The following 4 conditions should be met:

1. The episodes E1 and E2 belong to the same subject; are to be counted as sequential.
2. they are subsequent, i.e., the ending time of E1 coincides with the entry time of E2;
3. E1 was censored, not ended by a failure; and
4. meaningful covariates are identical on E1 and E2.

These conditions are easily generalized for more than 2 episodes. Why should one want to join such episodes? The reasons are practical, not really statistical. Using multiple records when only one suffices is a waste of computer memory. Using an uneconomical data representation may cause you to use approximate analytical methods when better methods would have been feasible with a more compact data representation. Second, most analytical commands require computing time that is proportional to the number of records (observations), not to the number of subjects. Thus, a more economical data representation reduces waiting time.

Syntax

The command `stjoin` implements joining of episodes on these 4 conditions. It should be invoked as

`stjoin` *varlist* [, `keep eps(`*#*`)`]

varlist specifies covariates that are meaningful for some analysis. Unless `keep` is specified, all other variables in the data (apart from the `st key` variables) are dropped. If `keep` is specified, all variables are kept in the data. Variables not included in *varlist* may of course be different on episodes that meet the 4 conditions. It is safe to assume that the values of such variables are chosen at random among the values of these variables on the joined episodes. `eps(`*#*`)` specifies the minimum elapsed time between exit and re-entry so that subsequent episodes are to be counted as sequential. `eps()` defaults to 0.

gr29 labgraph: placing text labels on two-way graphs

Jon Faust, Federal Reserve Board, faustj@frb.gov

`labgraph` is an extension of the `graph` (two-way version) command that allows one to place a list of text labels at specified (x, y) coordinates on the graph. Thus, one can label points of special interest or label lines on the graph. Labeling lines may be especially useful in rough drafts; differing line styles and a legend are probably preferred for more polished graphs.

All `graph` options are supported, two new options are supplied, and one `graph` option is modified. The two new options are for specifying labels and their locations:

`labtext(`*string* [`;`*string*]`...)` specifies a semicolon-delimited list of text labels.

`labxy(`*real*`,` *real* [`;` *real*`,` *real*]`...)` specifies a semicolon-delimited list of (x, y) coordinates for the labels.

You must state an (x, y) pair for each `labtext` string. The label will appear centered at $(\tilde{x}, y)$, where $\tilde{x}$ is the value of the x-axis variable in the `labgraph` command that is closest to the requested x coordinate.

The `trim` option of `graph` is modified. `trim` controls the size of text labels and defaults to 8 in `graph`. In `labgraph`, `trim` defaults to 32, the maximum allowed by `graph`.

The `psize` option of `graph` controls the size of the text for all the labels. The default `psize` is 100, larger numbers give larger text.

Example

The following commands produce the graph in Figure 1.

```
. set obs 20
. gen x=_n +0.1*invnorm(uniform())
. gen y1=_n    + 0.5*invnorm(uniform())
. gen y2=_n -5 + 0.5*invnorm(uniform())
. labgraph y1 y2 x, c(ll) s(..) t1ti(labgraph example) xlab ylab
> xline(7,10) yline(0,12) labtext(Line y1; Line y2) labxy(10,12;7,0)
```

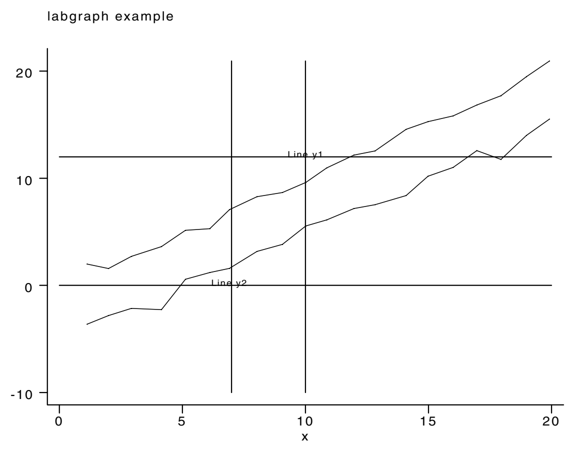


Figure 1. An example of using `labgraph`

The labels `Line y1` and `Line y2` will appear at approximately the requested coordinates, and the type size will be set to 100. The `xline` and `yline` options put cross hairs at the requested point for the labels, illustrating placement.

gr30	A set of 3D-programs

Guy D. van Melle, University of Lausanne, Switzerland, guy.van-melle@inst.hospvd.ch

Aside from William Gould's needleplot representations, `gr3`, Stata has very little 3D-capabilities. `gr3` appeared in STB-2 for Stata version 2.1 (Gould 1991) and was revisited and updated to version 3.0 in STB-12 (Gould 1993). It still works nicely if you set the version number to 3.0 first.

This insert presents three programs, `hidlin`, `altitude`, and `makfun`. The former two produce two different representations of a surface when you provide the equation of that surface, while the third one summarizes real data in a way suitable for use by either of the 2 graphing routines (it "makes" the function they require).

`hidlin` is a genuine 3D-program with hidden line removal (hence the name), whereas `altitude` uses colors and changing symbol-size to convey some sort of contour-plot appearance. The summary produced by `makfun` on real data is obtained from Stata's `pctile` and `collapse` commands. Counts are shown by default but you may request any of the statistics available for `collapse` on any other numerical variable in the dataset.

The three programs use macros and matrices whose names are prefixed by *hl*. The two graphing routines share most of their information and store in the same macros and matrices. These are kept after the graph completes, thus permitting a replay any time later in the same Stata session.

An auxiliary program `hltex` is used to display texts on these graphs.

Syntax for hidlin

```
hidlin func xinfo(# [,] # [,] # ) yinfo(# [,] # [,] # )
       eye(# [,] # [,] #) [, lines([x][y]) box coo neg
       xmargin(#) ymargin(#) text tmag(#) saving(filename) ]
```

where *func* stands for any name of your choosing (with a maximum of 6 characters). The function *func* is of the form $z = f(x, y)$ and must be declared in a separate ado-file (see the example below) whose name is *hlfunc* (the *hl* prefix will be stripped away).

User data are cleared and the function is calculated on the grid defined by `xinfo(`*xlow xhigh xstep*`)`, `yinfo(`*ylow yhigh ystep*`)` and viewed from a position declared by `eye(`*xpos ypos zpos*`)`; this is not a perspective view.

The grid and the eye position are required. The object is presented such that the lowest 2D-corner (projection) is “in front.”

`hidlin` can only represent a function, not data. However, the program `makfun` may be used to generate a function summarizing real data, with which `hidlin` may then produce a 3D representation.

Options

`xinfo`, `yinfo`, and `eye` must all be known. They define x (range and step) and y (range and step) for the grid as well as eye (x- y- z- directions), i.e., the position from where the surface is viewed. All specified values may be separated by commas and they may be expressions, e.g. `x( -_pi _pi, _pi/90 )` going from $-\pi$ to π in 180 steps.

`lines` requests that only x-curves be drawn when x is specified, or only y-curves when y is specified. More precisely, `lines(x)` asks to see the curves $z = f(x, y_0)$, where x varies at each successive level y_0 of y. Similarly, `lines(y)` requests the curves $z = f(x_0, y)$ where y varies at successive levels of x. When `lines` is omitted or when both x and y are specified, the default is to draw both sets of curves.

`box` asks that the surface be represented as a chunk of a solid 3D object.

`coo` asks that the coordinates of the “corners” of the surface be displayed.

`neg` requests that $-z$ be plotted rather than z (upside-down view).

`xmargin` and `ymargin` specify the sizes of the margins around the plot. They are expressed as a percentage of the graphing area. Default margins are 10 for both x and y (i.e., 10%).

`text` requests that stored texts be displayed. This option invokes the auxiliary program `hltex`.

`tmag` asks for a text magnification, the default being 100. The magnification applies to all texts shown by `hltex`.

`saving` stores the graph in *filename* which must be a new file.

Example of a function declaration

Below is a program defining the bivariate normal density surface. It generates the data, not the graph. It is invoked by

```
hlbivnor x y z [in #/#]
```

where `x` and `y` are any two existing variables and `z` is created or replaced as necessary.

Note that the code in this program preceding the actual declaration should appear in any *hlfunc* and that any parameter you wish to easily access should be declared in a global macro. In the present function, the correlation ρ between `x` and `y` is needed and the program assumes independence ($\rho = 0$) if `$rho` is empty.

```
*----------------
prog def hlbivnor
*----------------
      loc x `1´
      loc y `2´
      loc z `3´
      cap confirm var `z´
      if _rc qui gen `z´=.
      mac shift 3
      loc in "opt"
      parse "`*´"
*--- actual start of function declaration ---
if "$rho"==""  glo rho 0
loc r = 1 - ($rho)^2               /* need parentheses !!! */
loc c = 2 * _pi * sqrt(`r´)
#delimit ;
qui replace
`z´= exp( -(`x´^2 -2*$rho*`x´*`y´+`y´^2) / (2*`r´) )/ `c´ `in´
; #delimit cr
end
*----------------
```

Once the function is declared, the 3D representation may be obtained, for example by

```
. global rho=.9
. hidlin bivnor, x(-3 3 .2) y(-3 3 .3) e(2 -3 .5) b c
```

This is a coarse graph but gives a quick first look which is shown in Figure 1.

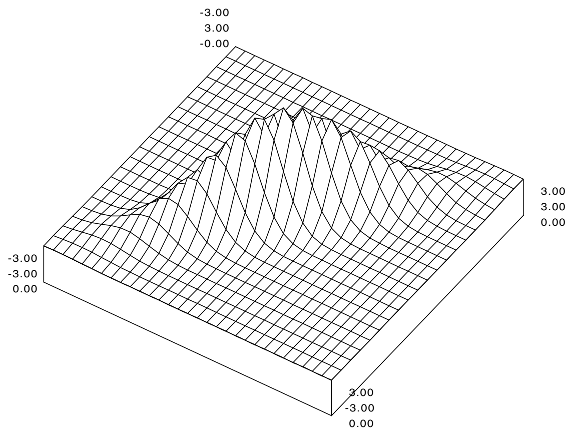


Figure 1. The bivariate normal density.

Inspection of this graph can give you an idea of a useful eye location and good values of the step sizes. For example,

```
. hidlin bivnor, x(-3 3 .1) y(-3 3 .15) e(-2 -4 .3) b c l(y)
```

gives the graph in Figure 2, which is the same as that in Figure 1 except that the location of the viewing eye has been moved, the grid is twice as fine, and only the y lines have been drawn.

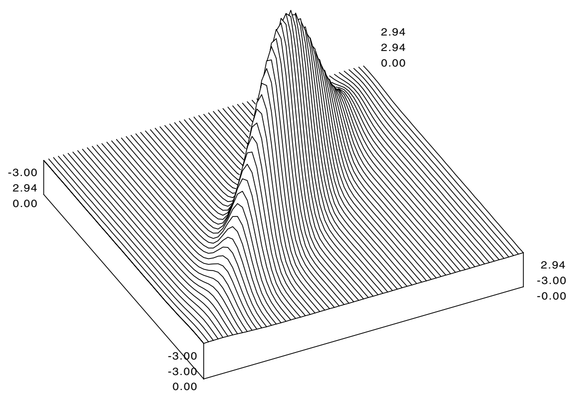


Figure 2. The bivariate normal density from a different view and using a finer grid.

Note that when the stepsizes are very small, that is, when the number of grid points becomes huge, graphing will become annoyingly slow if your function is very wavy or bumpy. The maximum allowed is 500 by 500 but that is already visually much too dense.

Details on the geometry for hidlin

In order to keep the number of drawing points reasonably low, it is required that the ratio of the x and y components of the eye position be the same as the ratio of the `xstep` to the `ystep`. Thus it is required that: $\mathrm{step}_x/\mathrm{step}_y = x_{eye}/y_{eye}$ (ignoring signs). However, when the information provided by the user does not satisfy that condition the program adapts the x- and y-step to meet the requirement, so that the user actually does not have to worry with this limitation.

The reason for implementing such a constraint is that all 2D-projected points now have fixed horizontal-axis coordinates (i.e., all points on the screen align on verticals). As a consequence, the next point to be drawn is easily identified as visible or invisible according to its position with respect to the current upper and lower edges of the graphed portion. These edges are permanently updated as new points are computed and, because they have fixed horizontal coordinates, their storage is very economical.

There are indeed $N_x + N_y + 1$ fixed coordinates, where N_x is the number of steps along x (i.e., $\mathrm{range}_x/\mathrm{step}_x$) and N_y the number of steps along y. Thus, even on a 500 by 500 grid, there are only 1001 points to memorize for the lower edge and 1001 for the upper edge!

When a curve becomes invisible, a partial segment is drawn from the last visible point to the approximate position where it disappears. This position is obtained, via linear interpolation, as the intersection of the previous visible edge and the segment joining the last point to the next (invisible) one. Naturally, a similar interpolation is performed when the curve goes from invisible to visible. These details are exhibited in Figure 3 (which incidentally was created entirely in Stata!).

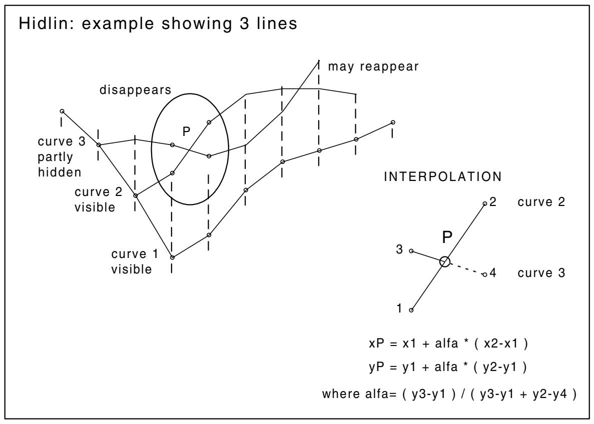

Hidlin: example showing 3 lines
may reappear
disappears
P
curve 3
partly
hidden
curve 2
visible
curve 1
visible
INTERPOLATION
2
curve 2
P
3
4
curve 3
1
xP = x1 + alfa * (x2-x1)
yP = y1 + alfa * (y2-y1)
where alfa= (y3-y1) / (y3-y1 + y2-y4)

Figure 3. Some details of the hidden line algorithm.

Syntax for altitude

`altitude` *func* `xinfo( #` [,] `#` [,] `# )` `yinfo( #` [,] `#` [,] `# )`
[, `nquant(`*#*`)` `symb(`*#*`)` `neg`
`xmargin(`*#*`)` `ymargin(`*#*`)` `text` `tmag(`*#*`)` `saving(`*filename*`)`]

The function is calculated on the grid defined by `xinfo(`*xlow xhigh xstep*`)`, `yinfo(`*ylow yhigh ystep*`)`, which are required.

The computed z-value (altitude) is then divided into `nquant` quantiles, which are displayed on the grid using symbol `symb`. The successive quantile levels are represented with different colors and increasing sizes of the symbol, so that the appearance truly yields contour effects. The legend relative to quantile colors and sizes is shown in the right margin.

Options

Most options are the same as for `hidlin`. `xinfo` and `yinfo` are required, while two options are specific for `altitude`:

`nquant` declares the desired number of quantiles (with a maximum of 20). The default is set to 16 because 8 pens are used (i.e., 8 colors), thus providing two full pen-color cycles.

`symb` specifies the choice of graphing symbol numbered as for `gph` (the default is 4):

0= dot	
1= large circle	4= small circle
2= square	5= diamond (not translucid)
3= triangle	6= plus

The lowest quantile is always represented by a dot so you always know where the surface is at its minimum.

Example: hlbivnor

The contour graph for the bivariate normal distribution can be produced using

```
. altitude bivnor, x(-3 3 .06) y(-3 3 .12) s(5)
(graph not shown)
```

Usually, satisfactory stepsizes are 100 steps along x and 50 steps along y, with certainly no need for more. Using the ranges of the example above, this might have been requested as `x(-3 3 6/100) y(-3 .3 6/50)` since expressions are permitted.

Note that the altitude representation provides a good indication of where to sit for viewing with `hidlin`. The x and y directions for Figures 1 and 2 are respectively: 2 along x, -3 along y and -2 along x, -4 along y, which are easy to visualize on the altitude plot. Only the z-direction of the eye may need repeated trials.

Syntax for makfun

`makfun` *x y* $\big[$, `x(`*#*|*matname*`)` `y(`*#*|*matname*`)` `z(`*zexpression*`)` `ground(`*#*`)` `smooth` $\big]$

where *x* and *y* are any two existing variables.

`makfun` generates a function $z = f(x,y)$ on an $[x,y]$-grid based on specified cutpoints or on quantiles of existing data in memory.

The data is preserved, but 3 matrices are created and left behind on exit; the x and y cutpoints are in *hlXcut* and *hlYcut*, and the z values are in *hlZfun*.

The aim was to provide an interface to the 3D-programs `hidlin` or `altitude` which can only plot functions, not data, but it turns out that `makfun` has general utility because *hlZfun* stores a nice 3D-summary of real data (smoothed if so desired) .

The graphical representation of this summary is obtained either from `hidlin` or `altitude` via the included `hlmakfun` function-declaration program as: `hidlin makfun,` ... or `altitude makfun,` ... (see the example below).

This `hlmakfun` program should not be modified, it knows how to use the saved matrices to reconstruct the surface and has the form of the `hlfun` required by `hidlin` or `altitude`.

Options

`x(`*#*|*matname*`)` indicates how the x variable is to be cut. The argument is either a number of quantiles (with a default of $\sqrt{\text{\_N}}/3$ in which case `xtile` is used) or a matrix containing the desired x cutpoints. Note that intervals are half-open $(\text{low}, \text{high}]$ as for `xtile` and with m intervals or quantiles there are $m-1$ cutpoints.

`y(`*#*|*matname*`)` is similar to the `x` option.

`z(`*zexpression*`)` indicates which summary is to be produced with choices being either `freq` or *statistic varname* (with a default of `freq`) with choices for *statistic* being the same as with `collapse`).

`ground(`*#*`)` is the z value to be used when an $[x,y]$ cell is empty and the *zexpression* cannot be evaluated. The default is 0.

`smooth` requests that computed z values be smoothed using surrounding cells, each with a weight of 1, and the target cell with a weight of 2 as shown below.

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{array}$$

For cells located outside the grid, or where a cell's contents are unknown (ground) a weight of 0 is used. Thus, for a corner cell at most 3 neighbors are available and for a cell elsewhere on an edge there are at most 5 neighbors. The original *hlZfun* matrix is copied into matrix *hlZfu1* and the smoothed summary is stored in *hlZfun*.

The bare call `makfun x y` uses $qx = qy = \min(\sqrt{\text{\_N}}/3, \texttt{matsize})$ quantiles and produces counts on the grid.

Example

For the auto data we study `mpg` in the quartiles of `price` by `foreign`. We begin by summarizing the grid variables:

```
. use auto
(1978 Automobile Data)
. summarize price
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      74    6165.257   2949.496       3291      15906
. tabulate foreign,nolabel
   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         52       70.27       70.27
          1 |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

Next, we use a matrix to define our own choice of cutpoints for `foreign` and use four quartiles for price:

```
. mat A=(0)
. makfun for pri, x(A) y(4) z(mean mpg)
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     mpg |       8     22.5845   4.769106   16.45455       30.5
```

This has generated 8 levels (2 × 4) and we may want to inspect the matrices that have been generated:

```
. mat dir
       hlZfun[2,4]
       hlYcut[3,1]
       hlXcut[1,1]
            A[1,1]
```

A first plot of the surface is shown in Figure 4 which was obtained by typing

```
. hidlin makfun, x(-1 1 .2)  y(3200 16000 200) e(1 -1000 -3) c
```

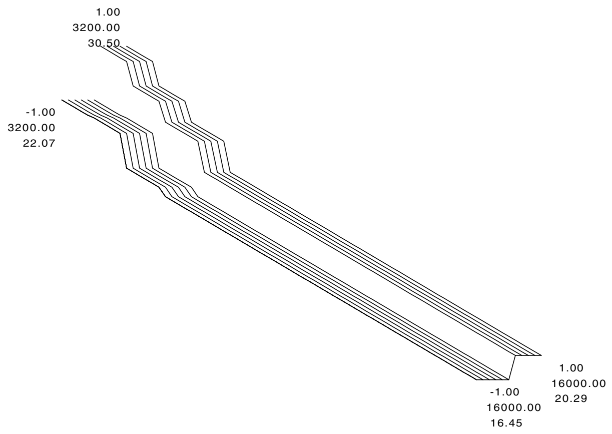


Figure 4. A first plot of the surface for the auto data.

The front (high quarter) is much too “long”; drawing from 3200 to 8000 should be adaquate. Now we adjust the grid and viewing location, show the box and coordinates and only draw the x lines, giving Figure 5:

```
. global xye x(-1 1 .025) y(3200 8000 50) e(1 -2000 -5)
. hidlin makfun, $xye b c l(x)
```

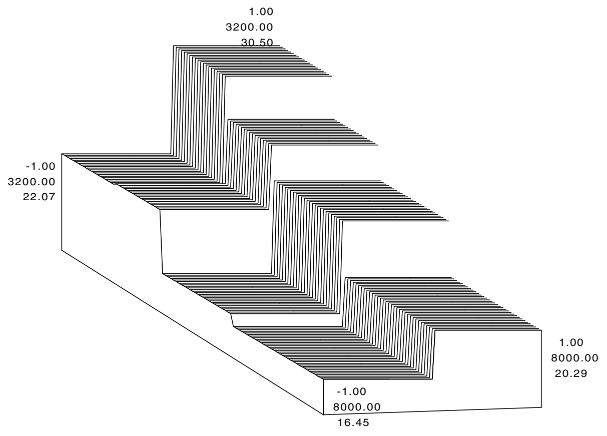


Figure 5. A better view of the surface in Figure 4.

Adding texts

In the previous graph, we would like to add some details with the `text` option. We first clear any texts in memory.

```
. hltex clean
. mat li hlZfun
hlZfun[2,4]
            c1         c2         c3         c4
r1  22.066668  22.214285  17.333334  16.454546
r2       30.5      27.25  24.571428  20.285715
```

Suppose we want to show the z levels on the graph. The utility `mat2mac` copies a row of a matrix into a global macro:

```
. mat2mac hlZfun 1 tx1 %6.1f        /* copy row 1 into tx1, format 6.1 */
tx1:      22.1   22.2   17.3   16.5
```

```
. mat2mac hlZfun 2 tx2 %6.1f        /* copy row 2 into tx2, format 6.1 */
tx2:      30.5   27.2   24.6   20.3
```

Next we define a number of texts:

```
. global hltexl mag(80). . . . Domestic . $tx1
. global hltexr mag(80). . . . Foreign . $tx2
. hidlin makfun, $xye b l(x) xm(15) ym(22) t tm(70)
. global hltext1 mag(100)Average  ´mpg´  in the quartiles of  ´price´  by  ´foreign´
. global hltexb2 hidlin  makfun,  $xye ...
. global hltexb1 ... box  lines(x) xmar(15)  ymar(22)  tex  tmag(70)
. global hltexc  GvM
```

Finally, we get Figure 6 by

```
. hidlin makfun, $xye b l(x) xm(15) ym(22) t tm(70)
```

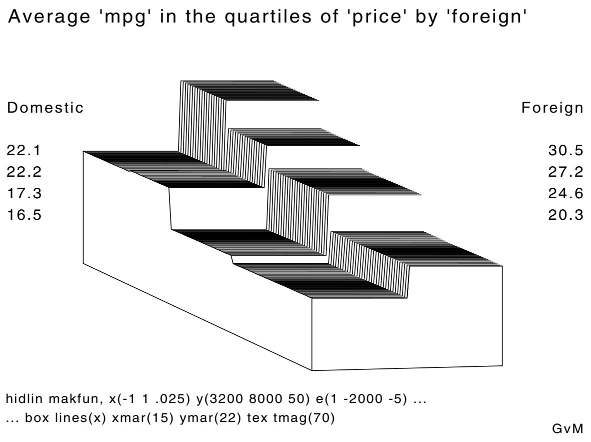


Figure 6. Using texts on graphs.

Stored Results

All results are stored in globals or matrices whose names start with *hl*.

hl-globals	
`hlprog`	progname for plotted function (`func`)
`hlXli, hlYLi`	0/1 indicating which curves are to be drawn
`hlXma, hlYma`	10 (default) or specified margin values (%)
`hlMag`	100 (default) or specified text magnification (%)
`hlXfa, hlYfa`	gph scaling factors
`hlXsh, hlYsh`	gph shifts
`hlX, hlY, hlE`	xinfo, yinfo, eye
`hlXn, hlYn, hln, hlNmax`	number of x-steps, y-steps, total, max
`hlBox, hlCoo, hlNeg, hltex`	indicators for corresponding options
hl-matrices	
`hlCx, hlCy`	[1x4] gph-coordinates of the corners
`hlCz`	[1x4] true z-value of the corners
`hlWk`	[2x3] working xinfo and yinfo (these may differ from the request because either the object has been oriented differently or because of geometric considerations)
`hlPj`	[2x3] projection matrix from 3D into 2D

Note that typing `hidlin clean` or `altitude clean` will remove all these stored results from memory. These commands invoke the utility `hlclean`.

References

Gould, W. 1991. gr3: Crude 3-dimensional graphics. *Stata Technical Bulletin* 2: 6–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 35–38.

——. 1993. gr3.1: Crude 3-dimensional graphics revisited. *Stata Technical Bulletin* 12: 12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 42–43.

gr31 Graphical representation of follow-up by time bands

Adrian Mander, MRC Biostatistical Research Unit, Cambridge, adrian.mander@mrc-bsu.cam.ac.uk

This command is a graphical addition to the `lexis` command introduced in STB-27 (Clayton and Hills 1995). The `lexis` command splits follow-up into different time bands and is used for the analysis of follow-up studies. It is more informative to draw the follow-up since all statistical analyses are incomplete without extensive preliminary analysis. Preliminary analysis usually can be plotting data or doing simple tabulations or summary statistics.

This command has the same basic syntax as the `lexis` command with a few additions. Follow-up starts from *timein* to *timeout* and is usually graphically represented by a straight line. With no additional information, follow-up is assumed to start from the same point in time, graphically this is assumed to be time 0. This can be offset using the `update()` option by another variable, e.g., date of birth perhaps or date of entry into the cohort.

For further information about Lexis diagrams, see Clayton and Hills (1993).

Syntax

```
grlexis2 timein fail fup [if exp] [, update(varname) event(evar) saving(filename) xlab(#,...,#)
         ylab(#,...,#) ltitle(string) btitle(string) title(string) symbol(#) nolines numbers
         noise box(#,#,#,#) ]
```

The variable *timein* contains the entry time for the time scale on which the observations are being expanded. The variable *fail* contains the outcome at exit; this may be coded 1 for failure or a more specific code for type of failure (such as icd) may be used; it must be coded 0 for censored observations. The variable *fup* contains the observation times.

Remarks

If there are missing values in the *timein* variable, that line is deleted. If there are missing values however in the *update* variable, then the *update* variable is set to zero. Proper handling of missing data values requires additional editing of the dataset or use of the `if` option.

On occasion, the command will be unable to cope with large x- or y-axis values and will overwrite the axis the label is on. In this case it is suggested that the `xlab` or `ylab` options be used.

Options

`update()` specifies variables which are entry times on alternative time scales. These variables will be appropriately incremented, thus allowing further expansion in subsequent calls.

`event()` If the subjects are followed up for the entire time but the events do not stop follow-up then the `event` option suppresses the drawing of a symbol at the end of a line and instead uses the values in *evar* to draw the symbols.

`saving(`*filename*`)` this will save the resulting graph in *filename*`.gph`. If the file already exists it is deleted and the new graph will be saved.

`xlab()` and `ylab()` have the same syntax as in the `graph` command. If these options are not specified, the labels will contain the maximum and minimum values on the axes.

`btitle()` and `ltitle()` are titles for the bottom and top-left portions of the graph. The default is the variable name in brackets followed by follow-up or *timein* depending on the axes.

`title()` adds a title at the top of the graph.

`symbol()` controls the plotting symbol of the failed events. The integer provided corresponds to

0 dot	
1 large circle	4 small circle
2 square	5 diamond
3 triangle	6 plus

`nolines` omits the lines that represent the follow up time.

`box` is the bounding box parameters. As usual, the lexis diagram has all the follow-up lines at 45 degrees from the horizontal. This can be achieved by setting up a square bounding box (for example, `box(2500,20000,2500,20000)`). The first number is top y-coordinate; second is bottom y-coordinate; third is left x-coordinate; and fourth is right x-coordinate.

`numbers` gives the option that instead of the failure codes, the line number will be plotted. For small datasets this may give clarity.

`noise` gives a little information about the two variables on the x and y axes.

Examples

The simplest plot displays follow-up offset by some starting time/date in the y direction. The automatic labeling of axes is between minimum and maximum points and they are labeled as time in and follow-up. In brackets the actual variables used are displayed.

```
. grlexis2 timein fail fup
```

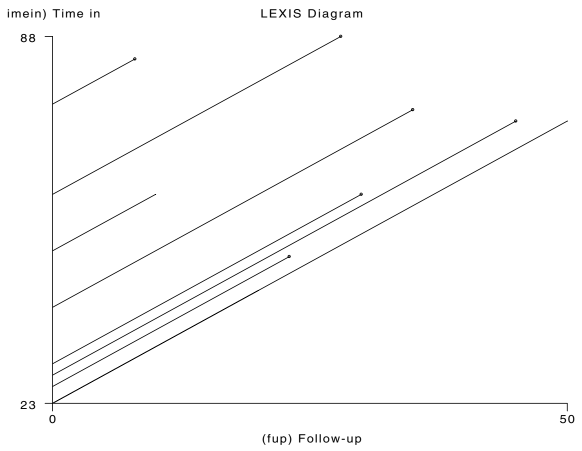


Figure 1

Taking the same dataset, the follow-up lines can now be offset in the x direction. Previously time-in may be the age of the person entering the cohort and datein may contain the dates that the subject entered the cohort. Now the follow-up times are more meaningful.

```
. grlexis2 timein fail fup, up(datein)
```

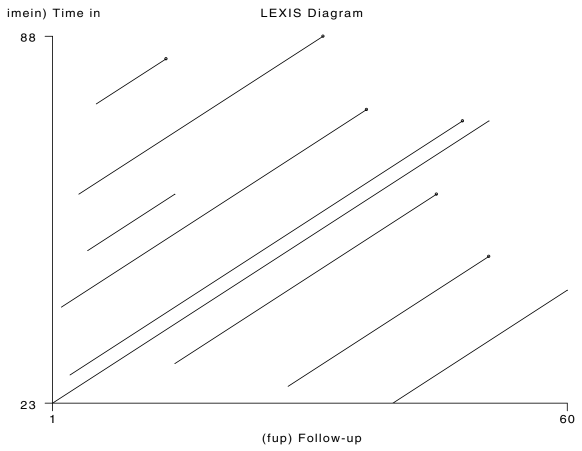


Figure 2

The following command just adds some of the options to improve the finished graph. `xlab` and `ylab` must have numbers as arguments to these options and no best guess system is used as in all other Stata graphs. The `numbers` option allows the drawing of text next to the events representing the line number in the dataset. The title options are similar to the usual graph options.

```
. grlexis2 timein fail fup, up(datein) numbers xlab(0,30,60) ylab(0,20,40,60,10)
> ltitle(Time in) btitle(Follow-up) title(my graph)
```

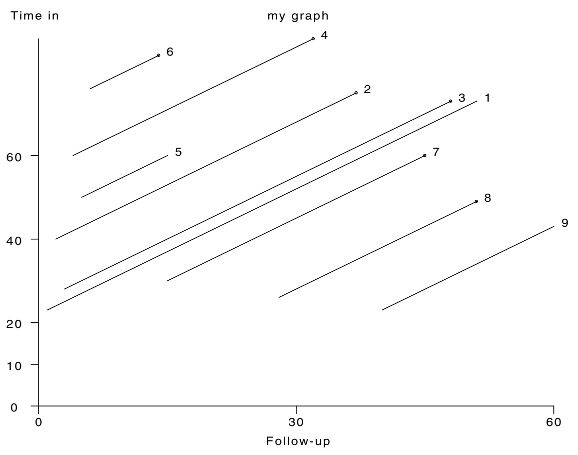


Figure 3

Take the same dataset but this time timein are the event times and everyone entered the cohort at birth. This means that the x-axis is time and the y-axis is age. The trick here is to generate a variable of zeroes. This insures all follow-up times will start from the y-axis ($y = 0$). Then each subjects' follow-up is offset in the x-direction by the entry time (in years).

```
. grlexis zero fail fup, up(timein) lti(Age) bti(Time) saving(gr3)
```

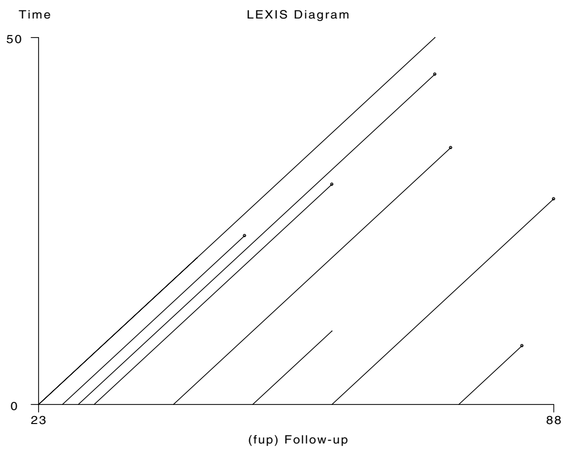


Figure 4

One possible disadvantage of the previous example is that after an event the subject may still be followed-up until the end of the study. However, previously if the follow-up is altered to the end of the study coverage time (in the command line this new variable is *fup2*), then all the events will be assumed to be at the end of the study. If the event occurs at earlier times, then a new variable is created that contains the amount of follow-up since study-entry until the first event that the subject had.

```
. grlexis zero fail fup2,up(timein) ev(event) lti(Age) bti(Time) saving(gr4)
```

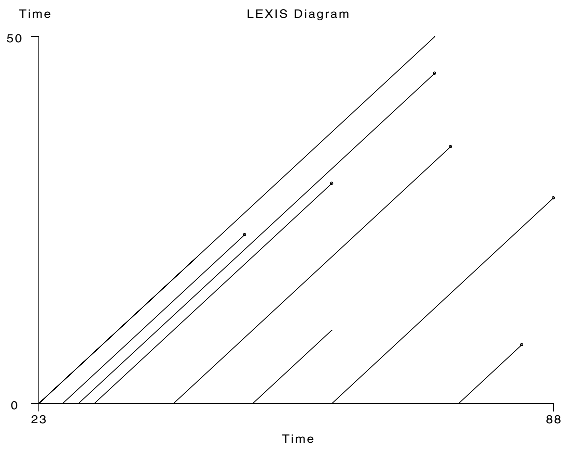


Figure 5

References

Clayton, D. and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

——. 1995. ssa7: Analysis of follow-up studies. *Stata Technical Bulletin* 27: 19–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 219–227.

ip14.1 Programming utility: numeric lists (correction and extension)

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

I recently encountered a bug in `numlist` (see Weesie 1997) that may occur in numeric lists with negative increments. In this new version of `numlist`, this bug was fixed. In addition, I improved the formatting of output to reduce the need for using explicit formatting. Finally, I replaced the relatively slow sorting algorithm with a Stata implementation of nonrecursive quicksort for numbers (see Wirth 1976).

The sorting commands (`qsort`, `qsortidx`) are actually separate utilities that may be of some interest by themselves to other Stata programmers. Some testing, however, indicated that a quicksort in Stata's macro language becomes quite slow for complex lists. For instance, sorting a macro with 1,000 random numbers took so long that I initially thought the main loop did not terminate due to some bug in my code. This is clearly stretching Stata's language beyond its design and purpose. I hope that Stata Corporation will one day apply its very fast sorting of numbers in variables to numbers in strings and macros.

References

Weesie, J. 1997. ip14: Programming utility: Numeric lists. *Stata Technical Bulletin* 35: 14–16. Reprinted in *Stata Technical Bulletin Reprints* vol. 6, pp. 68–70.

Wirth, N. 1976. *Algorithms + Data Structures = Programs*. Englewood Cliffs, NJ: Prentice–Hall.

ip26 Bivariate results for each pair of variables in a list

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

Syntax

`biv` *progname* [*varlist*] [*weight*] [`if` *exp*] [`in` *range*] [, `asy` `con` `echo` `header(`*header_string*`)` `hstart(`*#*`)` `own(`*own_command_name*`)` `pause` `quiet` *progname_options*]

Description

`biv` displays bivariate results from a Stata command or program *progname* for each pair of distinct variables in the *varlist*. For p variables there will be $p(p-1)/2$ such pairs if the order of the variables is immaterial and $p(p-1)$ otherwise.

The simplest cases are

```
. biv progname
. biv progname varlist
```

In the first case, the *varlist* defaults to `_all`.

Options

`asy` flags that the results of *progname var1 var2* may differ from those of *progname var2 var1* and that both sets are wanted.

`con` suppresses the new line that would normally follow the variable names echoed by `echo`. Thus other output (typically from the user's own program) may follow on the same line.

`echo` echoes the variable names to the monitor.

`header(`*header_string*`)` specifies a header string that will be displayed first before any bivariate results.

`hstart(`*#*`)` specifies the column in which the header will start. The default is column 21.

`own(`*own_command_name*`)` allows users to insert their own command dealing with the output from *progname*.

`pause` pauses output after each execution of *progname*. This may be useful, for example, under Stata for Windows when *progname* is `graph`.

`quiet` suppresses output from *progname*. This is likely to be useful only if the user arranges that output is picked up and shown in some way through the `own` option.

progname_options are the options of *progname*, if such exist.

Explanation

`biv` displays bivariate results for each pair of distinct variables in the list of variables supplied to it. It is a framework for some command or program (call it generically *progname*) that produces results for a pair of variables.

In some cases, perhaps the majority in statistical practice, the order of the variables is immaterial: for example, the correlation between x and y is identical to the correlation between y and x. In other cases the order of the variables is important: for example, the regression equation of y predicted from x is not identical to the converse. For the latter case, `biv` has an `asy` option (think asymmetric) spelling out that results are desired for x and y as well as for y and x.

`biv` might be useful in various ways. Here are some examples:

1. Some bivariate commands, such as `spearman` and `ktau`, take just two variables: thus if you want all the pairwise correlations, that could mean typing a large number of commands. (Actually, for `spearman`, there is another work-around: use `for` and `egen` to rank the set of variables in one fell swoop, and then use `correlate`.)
2. To get a set of scatter plots, you could use `graph, matrix`, but that may not be quite what you want. Perhaps you want each scatter plot to show variable names, or you would prefer a slow movie with each plot full size.
3. You write a program to do some bivariate work. `biv` provides the basic scaffolding of looping round a set of names to get all the distinct pairs. Thus your program can concentrate on what is specific to your problem.

`biv` also allows the minimal decoration of a one-line header; pausing after each run of *progname*; and suppressing the program's output and processing it with the user's own program.

Note that `biv` could lead to a lot of output if p is large and/or the results of *progname* are bulky. If p is 100, a modest number of variables in many fields, there are 4,950 pairs if the order of the variables is immaterial and 9,900 pairs otherwise. Users should consider the effects on paper and patience, their own and their institution's.

Examples

In the auto dataset supplied with Stata, we have several measures of vehicle size. Scatter plots for these would be obtained by

```
. biv graph length weight displ, xlab ylab
```

With Stata for Windows, the user may prefer

```
. biv graph length weight displ, xlab ylab pause
```

to allow enough time to peruse the graphs.

The patterns in these scatterplots are basically those of monotonic relationships, with some curvature. These might be expected not only from experience but also on dimensional grounds. With monotonicity and some lack of linearity, the Spearman correlation is a natural measure of strength of relationship.

```
. biv graph length displ weight
-> graph length displ
-> graph length weight
-> graph displ weight
. biv spearman length displ weight
-> spearman length displ
 Number of obs =      74
Spearman´s rho =       0.8525
Test of Ho: length and displ independent
      Pr > |t| =       0.0000
-> spearman length weight
 Number of obs =      74
Spearman´s rho =       0.9490
Test of Ho: length and weight independent
      Pr > |t| =       0.0000
-> spearman displ weight
 Number of obs =      74
Spearman´s rho =       0.9054
Test of Ho: displ and weight independent
      Pr > |t| =       0.0000
```

Finally, we show how to take more control over the output from *progname*. `concord` (Steichen and Cox 1998) calculates the concordance correlation coefficient, which measures the extent to which two variables are equal. This is the underlying question when examining different methods of measurement, or measurements by different people: when applied to the same phenomenon, the ideal is clearly that they are equal. (Two variables can be highly correlated, and close to some line $y = a + bx$, but equality is more demanding, as a should be 0, and b should be 1.)

`concord` takes two variables at a time, so `biv` is useful in getting all the pairwise concordance correlations.

An example dataset comes from Anderson and Cox (1978), who analyzed the measurement of soil creep (slow soil movement on hillslopes) at 20 sites near Rookhope, Weardale, in the Northern Pennines, England. The dataset is in the accompanying file `rookhope.dta`. The general slowness of the process (a few mm yr^{-1}) and the difficulty of obtaining a measurement without disturbance of the process combine to make measurement a major problem.

Six methods of measurement give 15 concordance correlation coefficients. Left to its own devices, `concord` would produce a fair amount of output for these 15, so we write a little program picking up the most important statistics from each run. The documentation for `concord` shows that the concordance correlation is left behind in global macro `S_2`, its standard error in global macro `S_3` and the bias correction factor in global macro `S_8`. The brief program `dispcon` uses `display` to print these out after each program run. The `biv` options used are `quiet`, to suppress the normal output from `concord`; `echo`, to echo variable names; `con`, to suppress the new line supplied by default, so that the output from `dispcon` will continue on the same line; and `header`, to supply a header string.

```
. program define dispcon
  1. version 5.0
  2. display %6.3f $S_2 %9.3f $S_3 %13.3f $S_8
  3. end

. describe using rookhope
Contains data                                      soil creep, Rookhope. Weardale
  obs:            20                               8 Jun 1998 12:53
 vars:             6
 size:           560
-------------------------------------------------------------------------------
   1. ip          float  %9.0g                     inclinometer pegs, mm/yr
   2. at          float  %9.0g                     Anderson´s tubes, mm/yr
   3. ap          float  %9.0g                     aluminium pillars, mm/yr
   4. yp          float  %9.0g                     Young´s pits, mm/yr
   5. dp          float  %9.0g                     dowelling pillars, mm/yr
   6. ct          float  %9.0g                     Cassidy´s tubes, mm/yr
-------------------------------------------------------------------------------
Sorted by:
. biv concord ct dp yp ip at ap  , qui echo con o(dispcon) he(rho_c se of rho_c
>  bias correction)
                    rho_c  se of rho_c bias correction
ct       dp         0.708    0.116       0.935
ct       yp         0.849    0.065       0.991
ct       ip         0.836    0.073       0.963
ct       at         0.819    0.079       0.965
ct       ap         0.754    0.102       0.945
dp       yp         0.857    0.064       0.957
dp       ip         0.923    0.034       0.992
dp       at         0.935    0.030       0.995
dp       ap         0.901    0.040       0.981
yp       ip         0.934    0.029       0.969
yp       at         0.933    0.031       0.980
yp       ap         0.867    0.056       0.944
ip       at         0.977    0.010       0.997
ip       ap         0.929    0.032       0.995
at       ap         0.936    0.027       0.983
```

The pattern of agreement between different measures is generally good, although Cassidy's tubes stand out as in relatively poor agreement with other methods.

References

Anderson, E. W. and N. J. Cox. 1978. A comparison of different instruments for measuring soil creep. *Catena* 5: 81–93.

Steichen, T. J. and N. J. Cox. 1998. sg84: Concordance correlation coefficient. *Stata Technical Bulletin* 43: 35–39.

ip27 Results for all possible combinations of arguments

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

Various commands in Stata allow repetition of a command or program for different arguments. In particular, `for` repeats one or more commands using items from one or more lists. The lists can be of variable names, of numbers, or of other items. An essential restriction on `for` is that the number of items must be the same in each list.

Thus

```
. for price weight mpg : gen l@ = log(mpg)
```

log-transforms each of the named variables, as does

```
. for price weight mpg \ logp logw logm, l(va) : gen @2 = log(@1)
```

The items in the first list match up one by one with those in the second list.

This insert describes a construct `cp` (think Cartesian product) that takes between one and five lists and some Stata command. With `cp` the number of items in each list is unrestricted, and `cp` executes the command in question for all combinations of items from the lists. For example, it can tackle problems of the form: do such-and-such for all $n_1 \times n_2$ pairs formed from the n_1 items in *list1* and the n_2 items in *list2*.

The construct is designed primarily for interactive use. The underlying programming is simply that of nested loops. Programmers should avoid building `cp` into their programs, as their own loops will typically work faster.

The restriction to five lists is not a matter of deep principle, but rather reflects a guess about what kinds of problems arise in practice.

`cp` could lead to a lot of output. Consider the effects on paper and patience, your own and your institution's.

Syntax

`cp` *list1* [\ *list2* [\ *list3*] ...] [, `noheader` `nostop` `pause`] : *stata_cmd*

Description

`cp` repeats *stata_cmd* using all possible n-tuples of arguments from between one and five lists: each possible argument from one list, each possible pair of arguments from two lists, and so forth.

In *stata_cmd*, `@1` indicates the argument from *list1*, `@2` the argument from *list2*, and so forth.

The elements of each list must be separated by spaces.

If there are n_j arguments in list j, k lists produce $\prod_{j=1}^{k} n_j$ sets of results.

The name `cp` is derived from Cartesian product.

Options

`noheader` suppresses the display of the command before each repetition.

`nostop` does not stop the repetitions if one of the repetitions results in an error.

`pause` pauses output after each execution of *stata_cmd*. This may be useful, for example, under Stata for Windows when `cp` is combined with `graph`.

Examples

With the auto data,

```
. cp 2 4 6 : xtile mpg@1 = mpg, n(@1)
```

executes

```
. xtile mpg2 = mpg, n(2)
. xtile mpg4 = mpg, n(4)
. xtile mpg6 = mpg, n(6)
```

and is close to

```
. for 2 4 6, l(n) : xtile mpg@1 = mpg, n(@1)
```

but

```
. cp length weight mpg \ 2 4 : xtile @1@2 = @1, n(@2)
```

which executes

```
. xtile length2 = length, n(2)
. xtile length4 = length, n(4)
. xtile weight2 = weight, n(2)
. xtile weight4 = weight, n(4)
. xtile mpg2 = mpg, n(2)
. xtile mpg4 = mpg, n(4)
```

(that is, two different numbers of quantile splits for three different variables) is more concise than the corresponding `for` statement:

```
. for length weight mpg : xtile @12 = @1, n(2) // xtile @14 = @1, n(4)
```

Another example is

```
. cp mpg price \ length displ weight : regress @1 @2
```

which executes

```
. regress mpg length
. regress mpg displ
. regress mpg weight
. regress price length
. regress price displ
. regress price weight
```

As the number of combinations of items increases, so also does the benefit from the conciseness of `cp` become more evident.

Acknowledgment

Thanks to Fred Wolfe for helpful stimuli and responses.

sbe18.1	Update of sampsi

Paul T. Seed, United Medical & Dental School, UK, p.seed@umds.ac.uk

The `sampsi` command (Seed 1997) has been updated. I have improved the error messages slightly. It now checks for correlations outside -1 to $+1$, and correctly reports when correlation `r1` is needed.

References

Seed, P. T. 1997. Sample size calculations for clinical trials with repeated measures data. *Stata Technical Bulletin* 40: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 121–125.

sbe24.1	Correction to funnel plot

Michael J. Bradburn, Institute of Health Sciences, Oxford, UK, m.bradburn@icrf.icnet.uk
Jonathan J. Deeks, Institute of Health Sciences, Oxford, UK, j.deeks@icrf.icnet.uk
Douglas G. Altman, Institute of Health Sciences, Oxford, UK, d.altman@icrf.icnet.uk

A slight error in the `funnel` command introduced in STB-44 has been corrected.

sg84.1	Concordance correlation coefficient, revisited

Thomas J. Steichen, RJRT, FAX 336-741-1430, steicht@rjrt.com
Nicholas J. Cox, University of Durham, UK, FAX (011) 44-91-374-2456, n.j.cox@durham.ac.uk

Description

`concord` computes Lin's (1989) concordance correlation coefficient, ρ_c, for agreement on a continuous measure obtained by two persons or methods and provides an optional graphical display of the observed concordance of the measures. `concord` also provides statistics and optional graphics for Bland and Altman's (1986) limits-of-agreement, *loa*, procedure. The *loa*, a data-scale assessment of the degree of agreement, is a complementary approach to the relationship-scale approach of ρ_c.

This insert documents enhancements and changes to `concord` and provides the syntax needed to use new features. A full description of the method and of the operation of the original command and options is given in Steichen and Cox (1998). This revision does not change the implementation of the underlying statistical methodology or modify the original operating characteristics of the program.

Syntax

`concord` *var1 var2* [*weight*] [`if` *exp*] [`in` *range*] [`,` `summary` `graph(ccc|loa)` `snd(`*snd_var*[`, replace`]`)` `noref` `reg` `by(`*by_var*`)` `level(`*#*`)` *graph_options*]

New and modified options

`noref` suppresses the reference line at $y = 0$ in the *loa* plot. This option is ignored if `graph(loa)` is not requested.

`reg` adds to the *loa* plot a regression line that fits the paired differences to the pairwise means. This option is ignored if `graph(loa)` is not requested.

`snd(`*snd_var*[`, replace`]`)` saves the standard normal deviates produced for the normal plot generated by `graph(loa)`. The values are saved in variable *snd_var*. If *snd_var* does not exist, it is created. If *snd_var* exists, an error will occur unless `replace` is also specified. This option is ignored if `graph(loa)` is not requested.

graph_options have been modified slightly. To accommodate the optional regression line, the default graph options for `graph(loa)` now include `connect(lll.l)`, `symbol(...o.)` and `pen(35324)` for the lower confidence interval limit, the mean difference, the upper confidence interval limit, the data points, and the regression line (if requested) respectively, along with default titles and labels. (The user is still not allowed to modify the default graph options for the normal probability plot, but additional labeling can be added.)

Explanation

Publication of the initial version of `concord` resulted in a number of requests for new or modified features. Further, a few deficiencies in the initial implementation were identified by users. The resulting changes are embodied in this version.

First, the initial implementation did not allow special characters, such as parentheses, to be included in the labels on the optional graphs. Such characters are now allowed when part of an existing label string (i.e., those labels created via Stata's `label` command). Because of parser limitations, special characters are not allowed, and will result in an error, when included in a passed option string.

Second, Bland and Altman (1986) suggested that the *loa* confidence interval would be valid provided the differences follow a normal distribution and are independent of the magnitude of the measurement. They argued that the normality assumption should be valid provided that the magnitude of the difference is independent of the magnitude of the individual measures. They proposed that these assumptions be checked visually using a plot of the casewise differences against the casewise means of the two measures and by a normal probability plot for the differences. In a 1995 paper they proposed an additional visual tool that was not implemented in the *loa* plot in the initial release of `concord`: that is, a regression line fitting the paired differences to the pairwise means. This additional tool is provided now when option `reg` is specified. As indicated above, the default graph options were changed to accommodate the regression line.

Third, as a direct result of adding the regression line, the reference line at $y = 0$ in the *loa* plot was made optional. Specification of option `noref` will suppress this reference line and reduce potential visual clutter.

Fourth, option `snd()` has been added to allow the user to save the standard normal deviates produced for the normal plot generated by `graph(loa)`. As indicated above, the values can be saved in either a new or existing variable. Following Stata convention, option modifier `replace`, which cannot be abbreviated, must be provided to overwrite an existing variable.

Lastly, it has come to our attention that two of the p values printed in the output were not explained in the initial insert. These p values are for the test of null hypothesis H_o: $\rho_c = 0$. The first of these p values results when the test is performed using the asymptotic point estimate and variance. The second occurs when Fisher's z-transformation is first applied. While not particularly applicable to the measurement question at hand, failure to reject this hypothesis indicates serious disconcordance between the measures. The more interesting hypothesis, H_o: $\rho_c = 1$, is not testable. The user is advised to use the confidence intervals for ρ_c to assess the goodness of the relationship.

Saved Results

The system S_# macros are unchanged.

Acknowledgments

We thank Richard Hall and Richard Goldstein for comments that motivated many of these changes and additions.

References

Bland, J. M. and D. G. Altman. 1986. Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet* I: 307–310.

——. 1995. Comparing methods of measurement: why plotting difference against standard is misleading. *Lancet* 346: 1085–1087.

Lin, L. I-K. 1989. A concordance correlation coefficient to evaluate reproducibility. *Biometrics* 45: 255–268.

Steichen, T. J. and N. J. Cox. 1998. sg84: Concordance correlation coefficient. *Stata Technical Bulletin* 43: 35–39.

sg89.1 Correction to the adjust command

Kenneth Higbee, Stata Corporation, khigbee@stata.com

The `adjust` command (Higbee 1998) has been improved to handle a larger number of variables in the variable list. Previously it would produce an uninformative error message when the number of variables was large.

Reference

Higbee, K. T. 1998. sg89: Adjusted predictions and probabilities after estimation. *Stata Technical Bulletin* 44: 30–37.

sg90 Akaike's information criterion and Schwarz's criterion

Aurelio Tobias, Institut Municipal d'Investigacio Medica (IMIM), Barcelona, atobias@imim.es
Michael J. Campbell, University of Sheffield, UK, m.j.campbell@sheffield.ac.uk

Introduction

It is well known that nested models, estimated by maximum likelihood, can be compared by examining the change in the value of -2log-likelihood on adding, or deleting, variables in the model. This is known as the likelihood-ratio test (McCullagh and Nelder 1989). This procedure is available in Stata using the `lrtest` command. To assess the goodness comparisons between nonnested models, two statistics can be used: the Akaike's information criterion (Akaike 1974) and Schwarz's criterion (Schwarz 1978). Neither are currently available in Stata.

[*Editor's note: A previous implementation of these tests was made available in Goldstein 1992.*]

Criteria for assessing model goodness of fit

Akaike's information criterion (AIC) is an adjustment to the -2log-likelihood score based on the number of parameters fitted in the model. For a given dataset, the AIC is a goodness-of-fit measure that can be used to compare nonnested models. The AIC is defined as follows: AIC $= -2$log-likelihood $+ 2 \times p$, where p is the total number of parameters fitted in the model. Lower values of the AIC statistic indicate a better model, and so, we aim to get the model that minimizes the AIC.

Schwarz's criterion (SC) provides a different way to adjust the -2log-likelihood for the number of parameters fitted in the model and for the total number of observations. The SC is defined as SC $= -2$log-likelihood $+ p \times \ln(n)$, where p is, again, the total number of parameters in the model, and n the total number of observations in the dataset. As for the AIC, we aim to fit the model that provides the lowest SC.

We should note that both the AIC and SC can be used to compare disparate models, although care should be taken because there are no formal statistical tests to compare different AIC or SC statistics. In fact, models selected can only be applied to the current dataset, and possible extensions to other populations of the best model chosen based on the AIC or SC statistics must be done with caution.

Syntax

The `mlfit` command works after fitting a maximum-likelihood estimated model with one of the following commands: `clogit`, `glm`, `logistic`, `logit`, `mlogit`, or `poisson`. The syntax is

```
mlfit
```

Example

We tested the mlfit command using a dataset from Hosmer and Lemeshow (Hosmer and Lemeshow 1989) on 189 births at a US hospital with the main interest being in low birth weight. The following ten variables are available in the dataset:

low	birth weight less than 2.5 kg (0/1)
age	age of mother in years
lwt	weight of mother (lbs) last menstrual period
race	white/black/other
smoke	smoking status during pregnancy (0/1)
ptl	number of previous premature labours
ht	history of hypertension (0/1)
ui	has uterine irritability (0/1)
ftv	number of physician visits in first semester
bwt	actual birth weight

We will fit a logistic regression model. The first model includes age, weight of mother at last menstrual period (lwt), race and smoke as predictor variables for the birth weight (low). As we can see, age is a nonstatistically significant variable.

```
. xi: logit low age lwt i.race smoke
i.race              Irace_1-3    (naturally coded; Irace_1 omitted)

Iteration 0:  Log Likelihood =  -117.336
Iteration 1:  Log Likelihood =-107.59043
Iteration 2:  Log Likelihood =-107.29007
Iteration 3:  Log Likelihood =-107.28862
Iteration 4:  Log Likelihood =-107.28862

Logit Estimates                                        Number of obs =    189
                                                       chi2(5)       =  20.09
                                                       Prob > chi2   = 0.0012
Log Likelihood = -107.28862                            Pseudo R2     = 0.0856

------------------------------------------------------------------------------
     low |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |  -.0224783   .0341705     -0.658   0.511      -.0894512    .0444947
     lwt |  -.0125257   .0063858     -1.961   0.050      -.0250417   -9.66e-06
 Irace_2 |   1.231671   .5171518      2.382   0.017       .2180725     2.24527
 Irace_3 |   .9432627   .4162322      2.266   0.023       .1274626    1.759063
   smoke |   1.054439   .3799999      2.775   0.006       .3096526    1.799225
   _cons |   .3324516   1.107673      0.300   0.764      -1.838548    2.503451
------------------------------------------------------------------------------

. mlfit
 Criteria for assessing logit model fit
 Akaike´s Information Criterion (AIC) and Schwarz´s Criterion (SC)
------------------------------------------------------------------------------
 AIC           SC           | -2 Log Likelihood    Num.Parameters
----------------------------+-------------------------------------------------
 226.57724     246.02772    |  214.57724             6
------------------------------------------------------------------------------
```

Now, we drop age from the model and include history of hypertension (ht) and uterine irritability.

(*Continued on next page*)

```
. xi: logit low lwt i.race smoke ht ui
i.race              Irace_1-3    (naturally coded; Irace_1 omitted)

Iteration 0:  Log Likelihood =  -117.336
Iteration 1:  Log Likelihood =-102.68681
Iteration 2:  Log Likelihood =-102.11335
Iteration 3:  Log Likelihood =-102.10831
Iteration 4:  Log Likelihood =-102.10831

Logit Estimates                                       Number of obs =    189
                                                      chi2(6)       =  30.46
                                                      Prob > chi2   = 0.0000
Log Likelihood = -102.10831                           Pseudo R2     = 0.1298

------------------------------------------------------------------------------
     low |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     lwt |  -.0167325   .0068034     -2.459   0.014      -.0300669    -.003398
 Irace_2 |   1.324562   .5214669      2.540   0.011       .3025055    2.346618
 Irace_3 |   .9261969   .4303893      2.152   0.031       .0826495    1.769744
   smoke |   1.035831   .3925611      2.639   0.008       .2664256    1.805237
      ht |   1.871416   .6909051      2.709   0.007       .5172672    3.225565
      ui |    .904974   .4475541      2.022   0.043        .027784    1.782164
   _cons |   .0562761   .9378604      0.060   0.952      -1.781897    1.894449
------------------------------------------------------------------------------

. mlfit

Criteria for assessing logit model fit
 Akaike´s Information Criterion (AIC) and Schwarz´s Criterion (SC)

------------------------------------------------------------------------------
 AIC           SC           | -2 Log Likelihood    Num.Parameters
----------------------------+-------------------------------------------------
 218.21662     240.90885    |  204.21662             7
------------------------------------------------------------------------------
```

Although both models are nonnested, we should use the AIC and/or SC statistics to compare them. As we can see from the results, the second model presents considerably lower AIC and SC values (AIC = 218.22, SC = 240.91) than the first model (AIC = 226.58, SC = 246.03). As we said earlier, there is no way to test if the difference between the two AICs is statistically significant.

The correctness of the implementation of calculation methods of AIC and SC statistics was assessed by doing the example with S-PLUS and SAS, respectively, and identical results were obtained.

Saved results

`mlfit` saves the following results in the S_ macros:

`S_E_aic`	AIC statistic
`S_E_sc`	SC statistic
`S_E_ll`	log-likelihood value
`S_E_ll2`	−2 log-likelihood value
`S_E_np`	number of parameters fitted in the model
`S_E_nobs`	number of observations

Acknowledgments

This work was done while Aurelio Tobias was visiting the Institute of Primary Care, University of Sheffield, UK. Aurelio Tobias was funded by a grant from the British Council and from the Research Stimulation Fund by ScHARR (University of Sheffield).

References

Akaike, H. 1974. A new look at statistical model identification. *IEEE Transactions on Automatic Control* 19: 716–722.

Goldstein, R. 1992. srd12: Some model selection statistics. *Stata Technical Bulletin* 6: 22–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 194–199.

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

McCullagh, P. and J. A. Nelder. 1989. *Generalized Linear Models*. 2d ed. London: Chapman and Hall.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

sg91 Robust variance estimators for MLE Poisson and negative binomial regression

Joseph Hilbe, Arizona State University, hilbe@asu.edu

Poisson and negative binomial regression are two of the more important routines used for modeling discrete response data. Although the Poisson model has been the standard method used to model such data, researchers have known that the assumptions upon which the model are based are rarely met in practice. In particular, the Poisson model assumes the equality of the mean and the variance of the response. Hence, the Poisson model assumes that cases enter each respective cell count in a uniform manner. Although the model is fairly robust to deviations from the assumptions, researchers now have software available to model overdispersed Poisson data. Notably, negative binomial regression has become the method of choice for modeling such data. In effect, the negative binomial assumes that cases enter each count cell with a gamma shape defined by α, the ancillary or heterogeneity parameter. Note that the variance of the Poisson model is simply μ, whereas the variance for the two parameter gamma is μ^2/α. Since the variance of the negative binomial is $\mu + k\mu^2$ where $k = 1/\alpha$, the negative binomial can be conceived as a Poisson-gamma mixture model with the assumption of log-likelihood criterion of case independence still retained.

The use of the Huber–White robust standard error sandwich estimator allows one to model data which may otherwise violate the log-likelihood assumption of the independence of cases. Essentially, the method adjusts the standard errors to accommodate extra correlation in the data. The parameter estimates are left unadjusted. Because the robust estimator can be used in such a manner for correlated data, it has been the standard method used by GEE programs to display standard errors.

Stata has implemented the `robust` and `cluster` options with many of its regression routines, but they have not yet been made part of Stata's `poisson` and `nbreg` ado-files. I have written programs which otherwise emulate the `poisson` and `nbreg` commands, but add the `robust`, `cluster`, and `score` options. The programs, called `poisml` and `nbinreg`, were written using Stata's ML capabilities. `poisml` calls `mypoislf.ado` and `nbinreg` calls `nbinlf.ado` for log-likelihood calculations.

Robust estimators are themselves calculated using the so-called `score` function. In Stata, this is defined as the derivative of the log-likelihood function with respect to B, the parameter estimates. When determining the score for an ancillary parameter, such as is the case with the negative binomial, one must calculate the derivative of the LL function with respect to that parameter, here termed α. For the negative binomial LL function, which includes α in two of its $\log -\gamma$ functions, the derivative necessitates the use of the digamma function ψ. The latter is not one of Stata's built-in functions; hence, a good approximation is needed. I have used a simple numerical approximation to the derivative; it appears to work quite well over the region of values used in calculating scores.

If we define $\delta = \exp(xb + o)$, the scores used for the Poisson and negative binomial with offset (log exposure) o are given by

Poisson	$y - \delta$
Negative binomial	$\dfrac{y-\delta}{1+a\delta}$
	$(-1/a)\left(\dfrac{a(\delta-y)}{1+a\delta} - \ln(1+a\delta) + \psi(y+1/a) - \psi(1/a)\right)$

Unlike Stata's `nbreg` command, `nbinreg` initializes values using Poisson estimates. This takes a little longer, but it may be worth it.

Example

We illustrate the use of `poisml` and `nbinreg` commands on the 1996 Arizona Medpar data on DRG 475 (the dataset is included on the diskette in the file `medpar.dta`).

```
. poisml los hmo died white age80 type2 type3, nolog irr
Poisson Estimates                                 Number of obs    =    1495
                                                  Model chi2(6)    =  947.07
                                                  Prob > chi2      =  0.0000
Log Likelihood =  -6834.6053315                   Pseudo R2        =  0.0648
------------------------------------------------------------------------------
     los |        IRR   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     hmo |   .9316981   .0223249     -2.953   0.003       .8889537    .9764978
    died |    .784677   .0143389    -13.270   0.000       .7570707      .81329
   white |   .8731055   .0239723     -4.942   0.000       .8273626    .9213775
   age80 |   .9826527   .0201582     -0.853   0.394       .9439271    1.022967
   type2 |   1.270891   .0268037     11.366   0.000       1.219427    1.324526
   type3 |   2.105107   .0553468     28.312   0.000       1.999377    2.216429
------------------------------------------------------------------------------
```

Using the `glm` command confirms there is overdispersion; the deviance-based dispersion is nearly six times greater than it should be for an appropriate Poisson model.

```
. glm los hmo died white age80 type2 type3, nolog f(poi)

Residual df  =      1488                               No. of obs =      1495
Pearson X2   =  9244.928                               Deviance   =  7954.061
Dispersion   =  6.212989                               Dispersion =  5.345471

Poisson distribution, log link
------------------------------------------------------------------------------
     los |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     hmo |  -.0707465   .0239615     -2.953   0.003      -.1177102   -.0237828
    died |  -.2424831   .0182736    -13.270   0.000      -.2782986   -.2066675
   white |  -.1356989   .0274563     -4.942   0.000      -.1895123   -.0818855
   age80 |  -.0174995   .0205141     -0.853   0.394      -.0577064    .0227073
   type2 |   .2397179   .0210905     11.366   0.000       .1983814    .2810545
   type3 |   .7443665   .0262917     28.312   0.000       .6928358    .7958972
   _cons |   2.391268   .0275976     86.648   0.000       2.337177    2.445358
------------------------------------------------------------------------------
```

Next we accommodate overdispersion by the `robust` and `cluster` options in `poisml` and `nbinreg`. Clustering by provider may account for some of the overdispersion.

```
. poisml los hmo died white age80 type2 type3, nolog robust cluster(provnum) irr

Poisson Estimates                                  Number of obs    =    1495
                                                   Model chi2(6)    =  947.07
                                                   Prob > chi2      =  0.0000
Log Likelihood =  -6834.6053315                    Pseudo R2        =  0.0648

                          (standard errors adjusted for clustering on provnum)
------------------------------------------------------------------------------
         |             Robust
     los |        IRR   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     hmo |   .9316981   .0480179     -1.373   0.170       .8421819    1.030729
    died |    .784677   .0510591     -3.726   0.000       .6907216    .8914128
   white |   .8731055   .0620901     -1.908   0.056       .7595114    1.003689
   age80 |   .9826527   .0620691     -0.277   0.782       .8682285    1.112157
   type2 |   1.270891    .079558      3.829   0.000       1.124146    1.436791
   type3 |   2.105107   .4528658      3.460   0.001       1.380886    3.209156
------------------------------------------------------------------------------


. nbinreg los hmo died white age80 type2 type3, nolog irr

Negative Binomial Estimates                        Number of obs    =    1495
                                                   Model chi2(6)    =  151.20
                                                   Prob > chi2      =  0.0000
Log Likelihood =  -4780.8947983                    Pseudo R2        =  0.0156

------------------------------------------------------------------------------
     los |        IRR   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
los      |
     hmo |   .9352302   .0494308     -1.267   0.205        .843197    1.037309
    died |   .7917555   .0323761     -5.710   0.000       .7307758    .8578237
   white |   .8854497    .060065     -1.793   0.073        .775215     1.01136
   age80 |   .9840357   .0458491     -0.345   0.730       .8981542    1.078129
   type2 |   1.272802     .06397      4.800   0.000         1.1534    1.404564
   type3 |   2.069855   .1562708      9.636   0.000       1.785153    2.399961
---------+--------------------------------------------------------------------
lnalpha  |
------------------------------------------------------------------------------
   alpha     .4339545   [lnalpha]_cons = ln(alpha)
                  (LR test against Poisson, chi2(1) =  4107.417 P =  0.0000)
```

Now we look at negative binomial regression with robust and clustering effect of provider. Note that in this case the p-values are quite similar to that of the robust Poisson model likewise adjusted by the clustering effect

```
. nbinreg los hmo died white age80 type2 type3, nolog irr robust clust(provnum)

Negative Binomial Estimates                        Number of obs    =    1495
                                                   Model chi2(6)    =  151.20
                                                   Prob > chi2      =  0.0000
Log Likelihood =  -4780.8947983                    Pseudo R2        =  0.0156
```

```
                          (standard errors adjusted for clustering on provnum)
------------------------------------------------------------------------------
         |               Robust
     los |        IRR   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
los      |
     hmo |   .9352302   .0477849     -1.311   0.190       .8461103    1.033737
    died |   .7917555   .0474362     -3.897   0.000       .7040335    .8904076
   white |   .8854497   .0641497     -1.679   0.093       .7682374    1.020545
   age80 |   .9840357   .0629949     -0.251   0.802       .8679997    1.115584
   type2 |   1.272802   .0785426      3.909   0.000       1.127806    1.436439
   type3 |   2.069855   .4302092      3.500   0.000       1.377278    3.110699
---------+--------------------------------------------------------------------
lnalpha  |
------------------------------------------------------------------------------
   alpha     .4339545   [lnalpha]_cons = ln(alpha)
                  (LR test against Poisson, chi2(1) =  4107.417 P =  0.0000)
```

References

Hilbe, J. 1994. sg16.5: Negative binomial regression. *Stata Technical Bulletin* 18: 2–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 84–88.

sg92 Logistic regression for data including multiple imputations

Christopher Paul, RAND, Santa Monica, California, cpaul@rand.org

One of the factors limiting the use of more sophisticated (and less biased) methods for dealing with missing data is sheer practicality. Here I hope to help decrease the entry cost for the use of multiple imputation by providing code that allows the use of multiple imputation data for logistic regression. Note that this program does not do the imputations; it just automates the combination and correction of coefficients and standard errors for multiply imputed data in the logistic regression context. For a complete discussion of multiple imputation, see Rubin (1987).

Logistic regression for data including multiple imputations

`implogit` *depvar varlist* [*weight*] [`if` *exp*] [`in` *range*] `, impvars(`*no. of vars*`) impno(`*no. of imputations*`)`
 [`robust cluster(`*varname*`) level(`*#*`) or`]

depvar may not contain multiple imputation data.

varlist may contain multiple imputation data, with the following conditions: variables containing imputed data must be the last variables in *varlist*; and imputed variables must have variable names following special conventions; that is, they must have fewer than 5 characters, and have `_01`–`_xx` appended to them, where `xx` is the number of imputations done. Even though an imputed variable may have many individual variables to repres ent it, include it only once in *varlist*. For example, suppose the variable `incom` was imputed 5 times. It should be labeled `incom_01 incom_02 incom_03 incom_04 incom_05`. In the command, list only one, say, `incom_01`. The program will iterate through the numbers and take care of the rest. If you r variables are not named properly, the program will not work!

`fweights` and `pweights` are allowed.

`implogit` does not share features with all estimation commands. Because of the external variance adjustments implicit in the corrections to the standard errors and the programmer's limited skill in matrix algebra, this program does not post a full variance-covariance estimate. Post-estimation commands that rely solely on `_b` and `_se` are available. Any commands requiring the off-diagonal elements of the variance–covariance estimate either will not work or will be wrong. Try `matrix get(vce)`, and you will see what is missing.

`implogit`, typed without argument, does not replay the previous results.

Description

`implogit` uses the Rubin (1987) corrections of coefficients and standard errors for logistic regressions with data that contain multiple imputations. Multiple imputation variables must be ordered in a specific way and named in a special fashion; see *varlist* above.

`implogit` proceeds by performing k logistic regressions (where k is the number of imputations done), cycling through the different imputations in each regression. Results are saved, and, when done, coefficients are averaged and standard errors are corrected. Results are then reported. Standard errors are corrected based on the following formula, in which k is the number of imputations done, and i runs from 1 to k:

$$\text{Var}(\widehat{\beta}) = \overline{SE(\beta)} + \left(1+\frac{1}{k}\right)\frac{\sum(\widehat{\beta}_2+\bar{\beta}_i)^2}{k-1}$$

In most regards, `implogit` behaves as the standard Stata `logit` command. The procedure reports unexponentiated coefficients and their corrected standard errors.

Options

`impvars(`*no. of vars*`)` indicates the number of variables included that contain multiple imputations. They must be the last variables specified in *varlist*. If your model contains 2 variables for which data have been imputed, `impvars(2)` should be specified, and they should be the last 2 variables in *varlist*. `impvars` default is 1.

`impno(`*no. of imputations*`)` is the number of imputations done for each imputed variable and thus the number of iterations of regressions (k) that will be required. If there is more than one variable with multiple imputations, they all must have the same number of imputations.

`robust` specifies the Huber/White/sandwich estimator of variance is to be used in place of the traditional calculation; see [U] **26.10 Obtaining robust variance estimates**. `robust` combined with `cluster()` allows observations which are not independent within `cluster`.

`cluster(`*varname*`)` specifies that the observations are independent across groups (clusters) but not necessarily within groups. *varname* specifies to which group each observation belongs; e.g., `cluster(personid)` in data with repeated observations on individuals. See [U] **26.10 Obtaining robust variance estimates**. `cluster()` can used with `pweights` to produce estimates for unstratified cluster-sampled data, but see `help svylogit` for a command especially designed for survey data. Specifying `cluster()` implies `robust`.

`or` reports odds ratios (exponentiated coefficients) and their standard errors, etc., rather than unexponentiated coefficients.

Examples

Data for these examples are from the RAND/UCLA Los Angeles Mammography Promotion in Churches Program baseline survey, conducted in 1995. Data were sampled in clusters, based on churches (`chid`). The outcome of interest is respondent compliance with AMA cancer screening guidelines for breast cancer (`comply`). More than 18% of respondents failed to report income. Income was imputed using a Bayesian bootstrap method (not shown or discussed here). In this example, all variables are (0/1) indicator variables. Income greater than $10,000 (`inc10`) is imputed 10 times and stored as `inc10_01–inc10_10`.

```
. use stbimp, clear
. implogit comply  partner dr_enthu hisp dropout dr_hisp dr_asian dr1_more noinsr
> inc10_01,impvars(1) impno(10) cluster(chid)

Iteration no. 1 complete.
Iteration no. 2 complete.
  (output omitted )
Iteration no. 9 complete.
Iteration no. 10 complete.

Logistic Regression using imputed values.
 Coefficients and Standard Errors Corrected
N = 1141

Log Likelihood for component regression no.1= -650.85724.
Log Likelihood for component regression no.2= -650.95298.
  (output omitted )
Log Likelihood for component regression no.9= -650.91123.
Log Likelihood for component regression no.10= -651.05008.
------------------------------------------------------------------------------
  comply |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 partner |   .3343246   .1163444      2.874   0.018       .0711354    .5975138
dr_enthu |   .8877968   .1437799      6.175   0.000       .5625442     1.21305
    hisp |  -.4692397   .2663255     -1.762   0.112       -1.07171    .1332306
 dropout |  -.5092944   .2159917     -2.358   0.043      -.9979015   -.0206872
 dr_hisp |  -.5585531   .2153716     -2.593   0.029      -1.045758   -.0713486
dr_asian |  -.3505409   .1724349     -2.033   0.073      -.7406157    .0395339
dr1_more |   .4855365   .1842637      2.635   0.027        .068703    .9023701
  noinsr |  -.8983906   .2974062     -3.021   0.014       -1.57117   -.2256111
inc10_01 |   .1070703   .2001414      0.535   0.606      -.3456811    .5598216
------------------------------------------------------------------------------
```

Here is the same regression with odds being reported.

```
. implogit comply  partner dr_enthu hisp dropout dr_hisp dr_asian dr1_more noinsr
> inc10_01,impvars(1) impno(10) cluster(chid) or
Iteration no. 1 complete.
Iteration no. 2 complete.
  (output omitted)
Iteration no. 9 complete.
Iteration no. 10 complete.
Logistic Regression using imputed values.
 Coefficients and Standard Errors Corrected
N = 1141
Log Likelihood for component regression no.1= -650.85724.
Log Likelihood for component regression no.2= -650.95298.
  (output omitted)
Log Likelihood for component regression no.9= -650.91123.
Log Likelihood for component regression no.10= -651.05008.
------------------------------------------------------------------------------
  comply |       Odds   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 partner |   1.396997   .1625327      2.874   0.018       1.073727    1.817594
dr_enthu |   2.429771   .3493521      6.175   0.000       1.755132    3.363727
    hisp |   .6254777   .1665807     -1.762   0.112       .3424225    1.142513
 dropout |   .6009195   .1297936     -2.358   0.043       .3686522    .9795253
 dr_hisp |   .5720362   .1232004     -2.593   0.029       .3514255    .9311373
dr_asian |    .704307   .1214471     -2.033   0.073       .4768203    1.040326
dr1_more |   1.625047   .2994372      2.635   0.027       1.071118    2.465439
  noinsr |   .4072245   .1211111     -3.021   0.014       .2078019    .7980284
inc10_01 |   1.113012   .2227599      0.535   0.606       .7077381     1.75036
------------------------------------------------------------------------------
```

Saved Results

All typical S_E_ global macros are available, as are _b and _se vectors. S_E_ll, S_E_l0 and S_E_prs2 are based on the unadjusted results of the last of the k component logistic regressions, as the programmer is not aware of the consequence to log-likelihood from multiple imputation.

Acknowledgments

This work was performed while working on the RAND/UCLA Los Angeles Mammography Promotion in Churches Project. The project and the time spent developing this program were supported by the National Institutes of Health, the National Cancer Institute, Award No. 1 R01 CA60132 (SAF). Thanks to both Daniel McCaffrey and Naihua Duan for their consultation and advice regarding the mathematics of the statistics computed here.

Reference

Rubin, D. B. 1987. *Multiple Imputation for Nonresponse in Surveys.* John Wiley & Sons: New York.

sg93 Switching regressions

Frederic Zimmerman, Stanford University, zimmer@leland.stanford.edu

Introduction

Consider the following system, in which observed values of the dependent variable come from one of two regression regimes, but where it is unknown *a priori* which regime produced any given observation. Here a third equation (a classification equation) determines whether a given observed dependent variable comes from regression regime 1 or regression regime 2:

$$\begin{aligned} y_{i1} &= x_i\beta_1 + u_{i1} \qquad u_1 \sim N\left(0,\sigma_1^2\right) \\ y_{i2} &= x_i\beta_2 + u_{i2} \qquad u_2 \sim N\left(0,\sigma_2^2\right) \\ y_{i3} &= x_i\beta_3 + u_{i3} \qquad u_3 \sim N\left(0,\sigma_3^2\right) \\ y_i &= \begin{cases} y_{i1} & \text{if } y_{i3} < 0 \\ y_{i2} & \text{if } y_{i3} \geq 0 \end{cases} \end{aligned}$$

Where the outcomes $\mathbf{y}$ are observed; $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$ are latent. The task then is to use the independent data $\mathbf{X}$ and the observed outcome variables $\mathbf{y}$ to estimate the unknown parameters for all three equations. Call the first two equations component equations and the third equation a classification equation. `switchr` produces estimates of β_1, β_2, and β_3, and of σ_1 and σ_2, which may be equal or different. The classification variance, σ_3^2, is unidentified, and assumed to be 1. Such models have been used to explore dual labor markets, the effects of school lunch programs (some kids eat them; some don't), and unobserved cultural differences in gender bias.

Syntax

`switchr` *eq1 eq2* [*weight*] [`if` *exp*] [`,` `cluster(`*string*`)` `strata(`*string*`)` `sigequal` `tol(`*real*`)` `itout(`*integer*`)` `noisyn(`*integer*`)`]

`pweight`s are allowed.

Description

The user defines the two component equations (*eq1*), which must be identical; and the classification equation (*eq2*), which is different.The dependent variable in the classification equation is the classification variable. An initial guess of the classification variable must be provided by the user for each observation. `switchr` returns the estimated classification vector ($\mathbf{y}_3$) with the same name as the initial guess of the classification vector. Since the estimated probabilities of observations belonging to the first component regime are not generally just zero or one, the elements of $\mathbf{y}_3$ fall throughout the interval $[0, 1]$.

Options

`cluster` and `strata` invoke the `_robust` variance calculations by using `svyreg`.

`sigequal` forces the variance of the two component regressions to be the same.

`tol` specifies a threshold such that, when the largest relative change among the coefficients from one iteration to the next is less than `tol()`, estimates are declared to have converged. The default is `tol(1e-4)`.

`itout()` and `noisyn()` control reporting of progress on the iterations and of intermediate output. `itout(`*#*`)` specifies that every # iterations, the convergence criterion, the mean and mean change of the classification vector, and the time taken for that iteration be reported. The default is `itout(20)`. `noisyn(`*#*`)` specifies that every # iterations, the regression results on the two component regressions be reported. In addition, results for a modified version of the classification vector are reported. These classifications results are accurate up to a constant (across coefficients) scaling parameter. The default is `noisyn(16000)`.

Remarks

To obtain these estimates, `switchr` maximizes the likelihood function through the EM algorithm of Dempster, Laird, and Rubin (1977), as further articulated by Hartley (1978). This iterative method first estimates the classification vector, i.e., the probability that a given observation is in the first component regression. This estimate is obtained by reweighting the probabilities based on the errors of the observations in the two component regressions. The updated probabilities are then used to weight observations in each of the two separate component regressions. This iterative procedure eventually converges to the maximum-likelihood estimates of the above three-equation regression model.

Exact standard errors in a switching regression are not available. This addition provides good approximate standard errors. The program can be used iteratively to obtain bootstrapped standard errors. See Douglas (1996).

Because the likelihood surface is not globally concave, several local maxima are possible. `switchr` can be used iteratively to find the local maximum with the highest likelihood value, by searching from different starting values of the classification vector. If σ_1 and σ_2 are allowed to differ (the default), problems can arise in which one variance is driven to zero, with a very small number of observations sorted into the corresponding component regime. One solution to this problem is to use the `sigequal` option to force the two variances to be the same.

Users should be aware of several caveats in using this method. First, estimation of switching regressions with unobserved regime separation is valid only when errors are independent, and identically distributed. The independence of errors across equations can be a strong assumption. Second, standard errors are not exact, and will in general be biased downward. Finally, convergence is not monotonic in the convergence criterion measure (relative change in the coefficient estimates). Therefore, it is prudent to be conservative in the convergence criterion.

This program uses `elapse` to help with the timing of the iterations; see Zimmerman (1998).

The full complement of programs needed for `switchr` is `switchr.ado`, `switchr.hlp`, `elapse.ado`, `elapse.hlp`, `_sw_lik.ado`, and `_sw_lik2.ado`.

Saved Results

S_E_ll log-likelihood value of the joint likelihood function

Example

In this example, we calculate a switching regression model of child height-for-age z scores on a number of variables. Children's height-for-age is hypothesized to depend on the explanatory variables in different ways according to unobservable cultural factors and time constraints in the household.

```
. gen double Guesprob = 0
. quietly summarize lnpce, detail
. replace Guesprob = 1 if lnpce > _result(10)
. eq main : haz docdist nursdist phardist father mother gender lnpce crop_siz urban rural
. eq regime : Guesprob lnpce members maxfe1 maxfe2 highfe urban rural KZN Ecape NorP reticul elect
. switchr main regime [pweight=rsweight] , cl(clustnum) noisyn(200) sige
```

Here is the regression for the switching equation:

```
On iter 1250 the mean absolute change in the probability vector is : 0.00001
Average of the probability vector  is: 0.194
On iteration 1250 greatest diff is: -0.000192 on phardist in the second main eq n
Log-likelihood is : -5562.2071

Note: subpop() subpopulation is same as full population
subpop() = 1 indicates observation in subpopulation
subpop() = 0 indicates observation not in subpopulation

Survey linear regression

pweight:  rsweight                              Number of obs    =      2864
Strata:   <one>                                 Number of strata =         1
PSU:      clustnum                              Number of PSUs   =       191
                                                Population size  =   2623753
                                                F(  12,    179)  =  53542.88
Subpopulation no. of obs =      2864            Prob > F         =    0.0000
Subpopulation size       =   2623753            R-squared        =    0.9852
------------------------------------------------------------------------------
Guesprob |      Coef.    Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   lnpce |   -.173144    .0084625    -20.460   0.000     -.1898365   -.1564514
 members |   .0150417     .001121     13.418   0.000      .0128304     .017253
  maxfe1 |   -.004693    .0054297     -0.864   0.389     -.0154031    .0060172
  maxfe2 |  -.0027249    .0004576     -5.954   0.000     -.0036276   -.0018223
  highfe |   .1936183    .0198772      9.741   0.000        .15441    .2328265
   urban |  -3.940108    .0110183   -357.596   0.000     -3.961842   -3.918374
   rural |  -4.789063    .0144696   -330.973   0.000     -4.817605   -4.760521
     KZN |  -.9713577    .0119679    -81.164   0.000     -.9949648   -.9477507
   ECape |  -.4252319    .0162149    -26.225   0.000     -.4572162   -.3932476
    NorP |  -.6130657    .0163161    -37.574   0.000     -.6452496   -.5808817
 reticul |  -.2623947    .0136669    -19.199   0.000     -.2893531   -.2354364
   elect |  -.0620349    .0084462     -7.345   0.000     -.0786953   -.0453744
   _cons |   4.361922    .0535379     81.474   0.000      4.256317    4.467527
------------------------------------------------------------------------------
```

Here is the first component regression (header omitted):

```
------------------------------------------------------------------------------
     haz |      Coef.    Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 docdist |   .1222716    .0129649      9.431   0.000      .0966979    .1478452
nursdist |  -.0472442    .0111013     -4.256   0.000     -.0691419   -.0253466
phardist |   -.010398     .007309     -1.423   0.156     -.0248153    .0040192
  father |  -.1235845    .1454306     -0.850   0.397     -.4104505    .1632815
  mother |   .0475992    .1847994      0.258   0.797     -.3169228    .4121212
  gender |   .2337818    .1124614      2.079   0.039      .0119485    .4556151
   lnpce |    .630567    .1122685      5.617   0.000      .4091142    .8520198
crop_siz |   .3970924    .2174543      1.826   0.069     -.0318424    .8260272
   urban |  -.6485643    .1608106     -4.033   0.000     -.9657677   -.3313609
   rural |   3.369163    .3003281     11.218   0.000      2.776758    3.961569
   _cons |   -4.69402    .6332944     -7.412   0.000     -5.943211   -3.444829
------------------------------------------------------------------------------
```

And now the second component regression (header omitted):

```
------------------------------------------------------------------------------
     haz |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 docdist | -.0020019    .0034663    -0.578   0.564     -.0088392    .0048354
nursdist |  .0020252    .0033815     0.599   0.550     -.0046449    .0086953
phardist |  .0008634    .0019992     0.432   0.666     -.0030801     .004807
  father | -.0394215    .0750347    -0.525   0.600     -.1874295    .1085865
  mother |    .21741    .1089093     1.996   0.047      .0025832    .4322368
  gender |  .1480176    .0693646     2.134   0.034      .0111939    .2848413
   lnpce |  .3266183    .0553563     5.900   0.000      .2174265    .4358102
crop_siz | -.0409482    .0330227    -1.240   0.217     -.1060864      .02419
   urban |  6.306667    .2095572    30.095   0.000      5.893309    6.720024
   rural |  5.946315    .2007892    29.615   0.000      5.550253    6.342377
   _cons | -9.339175    .3630126   -25.727   0.000     -10.05523   -8.623123
------------------------------------------------------------------------------
```

References

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B 39: 1–38.

Douglas, S. 1996. Bootstrap confidence intervals in a switching regressions model. *Economics Letters* 53: 7–15.

Goldfeld, S. M. and R. E. Quandt. 1976. *Studies in Non-Linear Estimation*. Cambridge: Ballinger.

Hartley, M. J. 1978. Comment. *Journal of the American Statistical Association* 73: 738–741.

Zimmerman, F. 1998. ip24: Timing portions of a program. *Stata Technical Bulletin* 41: 8–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, p. 92.

svy7 Two-way contingency tables for survey or clustered data

William M. Sribney, Stata Corporation, wsribney@stata.com

The syntax for the `svytab` command is

```
svytab varname1 varname2 [weight] [if exp] [in range] [,
      strata(varname) psu(varname) fpc(varname) subpop(varname) srssubpop
      tab(varname) missing
      cell count row column obs se ci deff deft
      { proportion | percent } nolabel nomarginals format(%fmt) vertical level(#)
      pearson lr null wald llwald noadjust ]
```

`pweight`s and `iweight`s are allowed. See [U] **18.1.6 weight** in the *Stata User's Guide.*

When any of `se`, `ci`, `deff`, or `deft` are specified, only one of `cell`, `count`, `row`, or `column` can be specified. If none of `se`, `ci`, `deff`, or `deft` are specified, any or all of `cell`, `count`, `row`, and `column` can be specified.

`svytab` typed without arguments redisplays previous results. Any of the options on the last three lines of the syntax diagram (`cell` through `noadjust`) can be specified when redisplaying with the following exception: `wald` must be specified at run time.

Warning: Use of `if` or `in` restrictions will not produce correct statistics and variance estimates for subpopulations in many cases. To compute estimates for a subpopulation, use the `subpop()` option.

[*Editor's note: The ado-files for this command can be found in the stata directory on the STB-45 diskette.*]

Introduction

This command produces two-way tabulations with tests of independence for complex survey data or for other clustered data.

Do not be put off by the long list of options for `svytab`. It is a simple command to use. Using the `svytab` command is just like using `tabulate` to produce two-way tables for ordinary data. The main difference is that `svytab` will compute a test of independence that is appropriate for a complex survey design or for clustered data. Here is an example:

```
. svyset strata stratid
. svyset psu psuid
. svyset pweight finalwgt
. svytab race diabetes
------------------------------------------------------------------------------
pweight:  finalwgt                               Number of obs      =     10349
Strata:   stratid                                Number of strata   =        31
PSU:      psuid                                  Number of PSUs     =        62
                                                 Population size    = 1.171e+08
------------------------------------------------------------------------------
----------+-------------------------
          |        Diabetes
     Race |      no      yes    Total
----------+-------------------------
    White |    .851    .0281    .8791
    Black |   .0899    .0056    .0955
    Other |   .0248  5.2e-04    .0253
          |
    Total |   .9658    .0342        1
----------+-------------------------
  Key:  cell proportions
  Pearson:
    Uncorrected   chi2(2)         =   21.3483
    Design-based  F(1.52,47.26)   =   15.0056     P = 0.0000
```

The test of independence that is displayed by default is based on the usual Pearson χ^2 statistic for two-way tables. To account for the survey design, the statistic is turned into an F statistic with noninteger degrees of freedom using the methods developed by Rao and Scott (1981, 1984). The theory behind this correction is quite advanced, and I will only sketch out a little of it later in this article. But understanding the theory behind the correction is not necessary to use and interpret the statistic. Just interpret the p-value next to the "Design-based" F statistic as you would the p-value for the Pearson χ^2 statistic for "ordinary" data; i.e., data that are assumed independent and identically distributed (iid).

Complicating this "simple" command is the fact that I did not just implement one statistic for the test of independence, but rather four statistics with two variants of each, for a total of eight statistics. This was done because the literature on the subject has not indicated that there is a single superior choice. However, I evaluated these eight in simulations, along with several other variants of these statistics, and every statistic except the one chosen for the default of this command had a black mark against it. So it made the choice of a default very easy. Based on these simulations, I would advise a practical researcher to just use the default statistic and never bother with any of the others.

The options that give these eight statistics are `pearson` (the default), `lr`, `null` (a toggle for displaying variants of the `pearson` and `lr` statistics), `wald`, `llwald`, and `noadjust` (a toggle for displaying variants of the `wald` and `llwald` statistics). The options `wald` and `llwald` with `noadjust` yield the statistics developed by Koch et al. (1975), which have been implemented in the CROSSTAB procedure of the SUDAAN software (Shah et al. 1997, Release 7.5). Based on simulations, which are detailed later in this article, I recommend that these two unadjusted statistics only be used for comparative purposes.

Other than the survey design options (the first row of options in the syntax diagram) and the test statistic options (the last row of options in the diagram), most of the other options merely relate to different choices for what can be displayed in the body of the table. By default, cell proportions are displayed, but it is likely that in many circumstances, it makes more sense to view row or column proportions or weighted counts.

Standard errors and confidence intervals can optionally be displayed for weighted counts or cell, row, or column proportions. The confidence intervals are constructed using a logit transform so that their endpoints always lie between 0 and 1. Associated design effects (deff and deft) can be viewed for the variance estimates. The mean generalized deff (Rao and Scott 1984) is also displayed when `deff` or `deft` is requested; this deff is essentially a design effect for the asymptotic distribution of the test statistic; see the discussion of it below.

Options

The survey design options `strata()`, `psu()`, `fpc()`, `subpop()`, and `srssubpop` are the same as those for the `svymean` command; see [R] **svyset** and [R] **svymean** in the *Stata Reference Manual* for a description of these options.

`tab(`*varname*`)` specifies that counts should instead be cell totals of this variable and proportions (or percentages) should be relative to (i.e., weighted by) this variable. For example, if this variable denotes income, then the cell "counts" are instead totals of income for each cell, and the cell proportions are proportions of income for each cell. See the *Methods and formulas* section at the end of this article.

`missing` specifies that missing values of *varname*$_1$ and *varname*$_2$ are to be treated as another row or column category, rather than be omitted from the analysis (the default).

`cell` requests that cell proportions (or percentages) be displayed. This is the default if none of `count`, `row`, or `column` are specified.

`count` requests that weighted cell counts be displayed.

`row` or `column` requests that row or column proportions (or percentages) be displayed.

`obs` requests that the number of observations for each cell be displayed.

`se` requests that the standard errors of either cell proportions (the default), weighted counts, or row or column proportions be displayed. When `se` (or `ci`, `deff`, or `deft`) is specified, only one of `cell`, `count`, `row`, or `column` can be selected. The standard error computed is the standard error of the one selected.

`ci` requests confidence intervals for either cell proportions, weighted counts, or row or column proportions.

`deff` or `deft` requests that the design-effect measure deff or deft be displayed for either cell proportions, counts, or row or column proportions. See [R] **svymean** for details. The mean generalized deff is also displayed when `deff` or `deft` is requested; see the following discussion for an explanation of mean generalized deff.

`proportion` or `percent` requests that proportions (the default) or percentages be displayed.

`nolabel` requests that variable labels and value labels be ignored.

`nomarginals` requests that row and column marginals not be displayed.

`format(%`*fmt*`)` specifies a format for the items in the table. The default is `%6.0g`. See [U] **19.5 Formats: controlling how data is displayed**.

`vertical` requests that the endpoints of confidence intervals be stacked vertically on display.

`level(`*#*`)` specifies the confidence level (i.e., nominal coverage rate), in percent, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] **26.4 Specifying the width of confidence intervals**.

`pearson` requests that the Pearson χ^2 statistic be computed. By default, this is the test of independence that is displayed. The Pearson χ^2 statistic is corrected for the survey design using the second-order corrections of Rao and Scott (1984) and converted into an F statistic. One term in the correction formula can be calculated using either observed cell proportions or proportions under the null hypothesis (i.e., the product of the marginals). By default, observed cell proportions are used. If the `null` option is selected, then a statistic corrected using proportions under the null is displayed as well. See the following discussion for details.

`lr` requests that the likelihood-ratio test statistic for proportions be computed. Note that this statistic is not defined when there are one or more zero cells in the table. The statistic is corrected for the survey design using exactly the same correction procedure that is used with the `pearson` statistic. Again, either observed cell proportions or proportions under the null can be used in the correction formula. By default, the former is used; specifying the `null` option gives both the former and the latter. Neither variant of this statistic is recommended for sparse tables. For nonsparse tables, the `lr` statistics are very similar to the corresponding `pearson` statistics.

`null` modifies the `pearson` and `lr` options only. If it is specified, two corrected statistics are displayed. The statistic labeled `D-B (null)` (`D-B` stands for design-based) uses proportions under the null hypothesis (i.e., the product of the marginals) in the Rao and Scott (1984) correction. The statistic labeled merely `Design-based` uses observed cell proportions. If `null` is not specified, only the correction that uses observed proportions is displayed. See the following discussion for details.

`wald` requests a Wald test of whether observed weighted counts equal the product of the marginals (Koch et al. 1975). By default, an adjusted F statistic is produced; an unadjusted statistic can be produced by specifying `noadjust`. The unadjusted F statistic can yield extremely anticonservative p-values (i.e., p-values that are too small) when the degrees of freedom of the variance estimates (the number of PSUs minus the number of strata) are small relative to the $(R-1)(C-1)$ degrees of freedom of the table (where R is the number of rows and C is the number of columns). Hence, the statistic produced by `wald` and `noadjust` should not be used for inference except when it is essentially identical to the adjusted statistic; it is only made available to duplicate the results of other software.

`llwald` requests a Wald test of the log-linear model of independence (Koch et al. 1975). Note that the statistic is not defined when there are one or more zero cells in the table. The adjusted statistic (the default) can produce anticonservative p-values, especially for sparse tables, when the degrees of freedom of the variance estimates are small relative to the degrees of freedom of the table. Specifying `noadjust` yields a statistic with more severe problems. Neither the adjusted nor the unadjusted statistic is recommended for inference; the statistics are only made available for pedagogical purposes and to duplicate the results of other software.

`noadjust` modifies the `wald` and `llwald` options only. It requests that an unadjusted F statistic be displayed in addition to the adjusted statistic.

Note: `svytab` uses the `tabdisp` command (see [R] **tabdisp**) to produce the table. As such, it has some limitations. One is that only five items can be displayed in the table. If you select too many items, you will be warned immediately. You may have to view additional items by redisplaying the table while specifying different options.

Examples of display options

The example shown earlier is from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981). The `strata`, `psu`, and `pweight` variables are first set using the `svyset` command rather than specifying them as options to `svytab`; see [R] **svyset** for details.

The default table displays only cell proportions, and this makes it very difficult to compare the incidence of diabetes in white versus black versus "other" racial groups. It would be better to look at row proportions. This can be done by redisplaying the results (i.e., the command is reissued without specifying any variables) with the `row` option.

```
. svytab, row
------------------------------------------------------------------------------
pweight:  finalwgt                               Number of obs      =     10349
Strata:   stratid                                Number of strata   =        31
PSU:      psuid                                  Number of PSUs     =        62
                                                 Population size    = 1.171e+08
------------------------------------------------------------------------------

----------+--------------------
          |      Diabetes
     Race |    no    yes  Total
----------+--------------------
    White |  .968   .032      1
    Black |  .941   .059      1
    Other | .9797  .0203      1
          |
    Total | .9658  .0342      1
----------+--------------------
  Key:  row proportions

  Pearson:
    Uncorrected   chi2(2)         =   21.3483
    Design-based  F(1.52,47.26)   =   15.0056     P = 0.0000
```

This table is much easier to interpret. A larger proportion of blacks have diabetes than do whites or persons in the "other" racial category. Note that the test of independence for a two-way contingency table is equivalent to the test of homogeneity of row (or column) proportions. Hence, we can conclude that there is a highly significant difference between the incidence of diabetes among the three racial groups. We may now wish to compute confidence intervals for the row proportions. If we try to redisplay specifying `ci` along with `row`, we get the following result:

```
. svytab, row ci
confidence intervals are only available for cells
to compute row confidence intervals, rerun command with row and ci options
r(111);
```

There are limits to what `svytab` can redisplay. Basically, any of the options relating to variance estimation (i.e., `se`, `ci`, `deff`, and `deft`) must be specified at run time along with the single item (i.e., `count`, `cell`, `row`, or `column`) for which one wants standard errors, confidence intervals, deff, or deft. So to get confidence intervals for row proportions, one must rerun the command. We do so below requesting not only `ci` but also `se`.

```
. svytab race diabetes, row se ci format(%7.4f)
----------------------------------------------------------------------------
pweight:  finalwgt                                Number of obs      =     10349
Strata:   stratid                                 Number of strata   =        31
PSU:      psuid                                   Number of PSUs     =        62
                                                  Population size    = 1.171e+08
----------------------------------------------------------------------------

----------+-------------------------------------------------
          |                  Diabetes
     Race |              no              yes            Total
----------+-------------------------------------------------
    White |          0.9680           0.0320           1.0000
          |        (0.0020)         (0.0020)
          | [0.9638,0.9718]  [0.0282,0.0362]
          |
    Black |          0.9410           0.0590           1.0000
          |        (0.0061)         (0.0061)
          | [0.9271,0.9523]  [0.0477,0.0729]
          |
    Other |          0.9797           0.0203           1.0000
          |        (0.0076)         (0.0076)
          | [0.9566,0.9906]  [0.0094,0.0434]
          |
    Total |          0.9658           0.0342           1.0000
          |        (0.0018)         (0.0018)
          | [0.9619,0.9693]  [0.0307,0.0381]
----------+-------------------------------------------------
  Key:  row proportions
        (standard errors of row proportions)
        [95% confidence intervals for row proportions]

  Pearson:
    Uncorrected   chi2(2)         =   21.3483
    Design-based  F(1.52,47.26)   =   15.0056     P = 0.0000
```

In the above table, we specified a `%7.4f` format rather than use the default `%6.0g` format. Note that the single format applies to every item in the table. If you do not want to see the marginal totals, you can omit them by specifying `nomarginal`. If the above style for displaying the confidence intervals is obtrusive—and it can be in a wider table—you can use the `vertical` option to stack the endpoints of the confidence interval one over the other and omit the brackets (the parentheses around the standard errors are also omitted when `vertical` is specified). If you would rather have results expressed as percentages, like the `tabulate` command, use the `percent` option. If you want to play around with these display options until you get a table that you are satisfied with, first try making changes to the options on redisplay (i.e., omit the cross-tabulated variables when you issue the command). This will be much faster if you have a large dataset.

The standard errors computed by `svytab` are exactly the same as those produced by the other `svy` commands that can compute proportions: namely, `svymean`, `svyratio`, and `svyprop`. Indeed, `svytab` calls the same driver program that `svymean` and `svyratio` use. For instance, the estimate of the proportion of African-Americans with diabetes (the second proportion in the second row of the preceding table) is simply a ratio estimate, which we can duplicate using `svyratio`:

```
. gen black = (race==2)

. gen diablk = diabetes*black
(2 missing values generated)

. svyratio diablk/black

Survey ratio estimation

pweight:  finalwgt                                  Number of obs    =     10349
Strata:   strata                                    Number of strata =        31
PSU:      psu                                       Number of PSUs   =        62
                                                    Population size  = 1.171e+08
----------------------------------------------------------------------------
      Ratio       |  Estimate    Std. Err.   [95% Conf. Interval]        Deff
------------------+---------------------------------------------------------
  diablk/black    |  .0590349    .0061443    .0465035    .0715662    .6718049
----------------------------------------------------------------------------
```

Although the standard errors are exactly the same (which they must be since the same driver program computes both), the confidence intervals are slightly different. The `svytab` command produced the confidence interval $[\,0.0477, 0.0729\,]$, and `svyratio` gave $[\,0.0465, 0.0716\,]$. The difference is due to the fact that `svytab` uses a logit transform to produce confidence intervals whose endpoints are always between 0 and 1. This transformation also shifts the confidence intervals slightly toward the null (i.e., 0.5), which is beneficial since the untransformed confidence intervals tend to be, on average, biased away from the null. See the *Methods and formulas* section at the end of this article for details.

The Rao and Scott correction for chi-squared tests

Two statistics commonly used for iid data for the test of independence of $R \times C$ tables (R rows and C columns) are the Pearson χ^2 statistic

$$X^2_{\rm P} = n \sum_{r=1}^{R}\sum_{c=1}^{C} \left(\hat{p}_{rc} - \hat{p}_{0rc}\right)^2 / \hat{p}_{0rc} \tag{1}$$

and the likelihood-ratio χ^2 statistic

$$X^2_{\rm LR} = 2n \sum_{r=1}^{R}\sum_{c=1}^{C} \hat{p}_{0rc} \ln\left(\hat{p}_{rc}/\hat{p}_{0rc}\right) \tag{2}$$

where n is the total number of observations, $\hat{p}_{rc}$ is the estimated proportion for the cell in the rth row and cth column of the table, and $\hat{p}_{0rc}$ is the estimated proportion under the null hypothesis of independence; i.e., $\hat{p}_{0rc} = \hat{p}_{r\cdot}\hat{p}_{\cdot c}$, the product of the row and column marginals: $\hat{p}_{r\cdot} = \sum_{c=1}^{C} \hat{p}_{rc}$ and $\hat{p}_{\cdot c} = \sum_{r=1}^{R} \hat{p}_{rc}$. For iid data, both of these statistics are distributed asymptotically as $\chi^2_{(R-1)(C-1)}$. Note that the likelihood-ratio statistic is not defined when one or more of the cells in the table are empty. The Pearson statistic, however, can be calculated when one or more cells in the table are empty—the statistic may not have good properties in this case, but the statistic still has a computable value.

One can estimate the variance of $\hat{p}_{rc}$ under the actual survey design. For instance, in Stata, this variance can be computed using linearization methods using `svymean` or `svyratio`. With this variance estimate available, the question then becomes how to use it to estimate the true distribution of the statistics $X^2_{\rm P}$ and $X^2_{\rm LR}$. Rao and Scott (1981, 1984) showed that, asymptotically, $X^2_{\rm P}$ and $X^2_{\rm LR}$ are distributed as

$$X^2 \sim \sum_{k=1}^{(R-1)(C-1)} \delta_k W_k \tag{3}$$

where the W_k are independent χ^2_1 variables and the δ_k are the eigenvalues of

$$\Delta = (\widetilde{\mathbf{X}}_2' \mathbf{V}_{\rm srs} \widetilde{\mathbf{X}}_2)^{-1} (\widetilde{\mathbf{X}}_2' \mathbf{V} \widetilde{\mathbf{X}}_2) \tag{4}$$

where $\mathbf{V}$ is the variance of the $\hat{p}_{rc}$ under the survey design and $\mathbf{V}_{\rm srs}$ is the variance of the $\hat{p}_{rc}$ that one would have if the design were simple random sampling; namely, $\mathbf{V}_{\rm srs}$ has diagonal elements $p_{rc}(1-p_{rc})/n$ and off-diagonal elements $-p_{rc}p_{st}/n$.

$\widetilde{\mathbf{X}}_2$ takes a bit of explaining. Rao and Scott do their development in a log-linear modeling context. So consider $[\,\mathbf{1}\,|\,\mathbf{X_1}\,|\,\mathbf{X_2}\,]$ as predictors for the cell counts of the $R \times C$ table. The $\mathbf{X_1}$ matrix of dimension $RC \times (R+C-2)$ contains the $R-1$ "main effects" for the rows and the $C-1$ "main effects" for the columns. The $\mathbf{X_2}$ matrix of dimension $RC \times (R-1)(C-1)$ contains the row and column "interactions." Hence, fitting $[\,\mathbf{1} \mid \mathbf{X}_1 \mid \mathbf{X}_2\,]$ gives the fully saturated model (i.e., fits the observed values perfectly) and $[\,\mathbf{1} \mid \mathbf{X}_1\,]$ gives the independence model. The $\widetilde{\mathbf{X}}_2$ matrix is the projection of $\mathbf{X}_2$ onto the orthogonal complement of the space spanned by the columns of $\mathbf{X}_1$, where the orthogonality is defined with respect to $\mathbf{V}_{\rm srs}$; i.e., $\widetilde{\mathbf{X}}_2' \mathbf{V}_{\rm srs} \mathbf{X}_1 = \mathbf{0}$.

I do not expect anyone to understand from this brief description how this leads to equation (3); see Rao and Scott (1984) for the proofs. However, even without a full understanding, one can get a feeling for Δ. It is like a "ratio" (although remember that it is a matrix) of two variances. The variance in the "numerator" involves the variance under the true survey design, and the variance in the "denominator" involves the variance assuming that the design was simple random sampling. Recall that the design effect deff for the estimated proportions is defined as deff $= V(\hat{p}_{rc})/V_{\rm srs}(\hat{p}_{rc})$ (see the *Methods and Formulas* section of the [R] **svymean** entry in the *Stata Reference Manual*). Hence, Δ can be regarded as a design effects matrix, and Rao and Scott call its eigenvalues, the δ_k's, the "generalized design effects." Simplistically, one can view the iid statistics $X^2_{\rm P}$ and $X^2_{\rm LR}$ as being "too big" by a "factor" of Δ for true survey design.

It is easy to compute an estimate for Δ using estimates for $\mathbf{V}$ and $\mathbf{V}_{\rm srs}$. But, unfortunately, equation (3) is not practical for the computation of a p-value. However, one can compute simple first-order and second-order corrections based on it. A first-order correction is based on downweighting the iid statistics by the average eigenvalue of $\widehat{\Delta}$; namely, one computes

$$X^2_{\rm P}(\hat{\delta}_\cdot) = X^2_{\rm P}/\hat{\delta}_\cdot \qquad \text{and} \qquad X^2_{\rm LR}(\hat{\delta}_\cdot) = X^2_{\rm LR}/\hat{\delta}_\cdot$$

where $\hat{\delta}_{\cdot}$ is the mean generalized deff

$$\hat{\delta}_{\cdot} = \frac{1}{(R-1)(C-1)} \sum_{k=1}^{(R-1)(C-1)} \delta_k$$

These corrected statistics are asymptotically distributed as $\chi^2_{(R-1)(C-1)}$.

A better second-order correction can be obtained by using the Satterthwaite approximation to the distribution of a weighted sum of χ^2_1 variables. Here the Pearson statistic becomes

$$X^2_{\mathrm{P}}(\hat{\delta}_{\cdot}, \hat{a}) = \frac{X^2_{\mathrm{P}}}{\hat{\delta}_{\cdot}(\hat{a}^2+1)} \tag{5}$$

where $\hat{a}$ is the coefficient of variation of the eigenvalues:

$$\hat{a}^2 = \frac{\sum \hat{\delta}_k^2}{(R-1)(C-1)\hat{\delta}_{\cdot}^2} - 1$$

Since $\sum \hat{\delta}_k = \mathrm{tr}\,\widehat{\Delta}$ and $\sum \hat{\delta}_k^2 = \mathrm{tr}\,\widehat{\Delta}^2$, equation (5) can be written in an easily computable form as

$$X^2_{\mathrm{P}}(\hat{\delta}_{\cdot}, \hat{a}) = \frac{\mathrm{tr}\,\widehat{\Delta}}{\mathrm{tr}\,\widehat{\Delta}^2}\, X^2_{\mathrm{P}} \tag{6}$$

These corrected statistics are asymptotically distributed as χ^2_d with

$$d = \frac{(R-1)(C-1)}{\hat{a}^2+1} = \frac{(\mathrm{tr}\,\widehat{\Delta})^2}{\mathrm{tr}\,\widehat{\Delta}^2} \tag{7}$$

i.e., a χ^2 with, in general, noninteger degrees of freedom. The likelihood-ratio X^2_{LR} statistic can also be given this second-order correction in an identical manner.

Two wrinkles in the Rao and Scott correction formula

We would be done if it were not for two outstanding issues. First, how should one compute the variance estimate $\widehat{\mathbf{V}}_{\mathrm{srs}}$, which appears in equation (4) and in the computation of $\widetilde{\mathbf{X}}_2'$? This may seem like a stupid question, since as we said earlier, $\mathbf{V}_{\mathrm{srs}}$ has diagonal elements $p_{rc}(1-p_{rc})/n$ and off-diagonal elements $-p_{rc}p_{st}/n$; but note that here p_{rc} is the true, not estimated, proportion. Hence, the question is, what to use to estimate p_{rc}: the observed proportions $\hat{p}_{rc}$ or the proportions estimated under the null hypothesis of independence $\hat{p}_{0rc} = \hat{p}_{r\cdot}\hat{p}_{\cdot c}$? Rao and Scott (1984, p. 53) write "In practice, [the asymptotic significance level of $X^2(\delta_{\cdot}, a)$] is estimated by using $\widehat{\mathbf{V}}$ for $\mathbf{V}$ and $\hat{\pi}$ (or $\hat{\hat{\pi}}$) for π." In our notation, π_{rc} is p_{rc}, $\hat{\pi}_{rc}$ is the observed $\hat{p}_{rc}$, and $\hat{\hat{\pi}}_{rc}$ is the estimate under the null, $\hat{p}_{0rc}$. So Rao and Scott leave this as an open question.

Because of the question of whether to use $\hat{p}_{rc}$ or $\hat{p}_{0rc}$ to compute $\widehat{\mathbf{V}}_{\mathrm{srs}}$, `svytab` can compute both corrections. By default, when the `null` option is not specified, only the correction based on $\hat{p}_{rc}$ is displayed. If `null` is specified, two corrected statistics and corresponding p-values are displayed, one computed using $\hat{p}_{rc}$ and the other using $\hat{p}_{0rc}$. Simulations evaluating the performance of each of these correction methods are discussed later in this article.

The second wrinkle to be ironed out concerns the degrees of freedom resulting from the variance estimate $\widehat{\mathbf{V}}$ of the cell proportions under the survey design. The customary degrees of freedom for t statistics resulting from this variance estimate are $\nu = n_{\mathrm{PSU}} - L$, where n_{PSU} is the total number of PSUs and L is the total number of strata. Hence, one feels that one should use something like ν degrees of freedom as the denominator degrees of freedom in an F statistic.

Thomas and Rao (1987) explore this issue, not for two-way contingency tables, but for simple goodness-of-fit tests. They took the first-order corrected $X^2_{\mathrm{P}}(\hat{\delta}_{\cdot})$ and turned it into an F statistic by dividing it by its degrees of freedom $d_0 = (R-1)(C-1)$. The F statistic was taken to have numerator degrees of freedom equal to d_0, and denominator degrees of freedom equal to νd_0. This is much more liberal than using just ν as denominator degrees of freedom. Rao and Thomas (1989) mention the possibility of implementing a similar F statistic for two-way tables, but state that "its properties have not been studied." Note that Rao and Thomas's discussion concerns F statistics produced from the first-order corrected $X^2(\hat{\delta}_{\cdot})$ statistics, and not from the second-order corrected $X^2(\hat{\delta}_{\cdot}, \hat{a})$ statistics. But there is no reason that one should not consider similar transforms of the second-order corrected $X^2(\hat{\delta}_{\cdot}, \hat{a})$ statistics into F statistics.

Fuller et al. (1986) in the PC CARP software take the stricter approach with the second-order corrected $X^2_{\rm P}(\hat\delta_\cdot, \hat a)$ and turn it into an F statistic with ν denominator degrees of freedom. This strictness is partially balanced by the fact that rather than $d = (\mathrm{tr}\,\widehat\Delta)^2/\mathrm{tr}\,\widehat\Delta^2$ numerator degrees of freedom, they use

$$d = \min\{(\mathrm{tr}\,\widehat\Delta)^2/t_3,\ (R-1)(C-1)\} \qquad \text{where} \qquad t_3 = \mathrm{tr}\,\widehat\Delta^2 - (\mathrm{tr}\,\widehat\Delta)^2/\nu \tag{8}$$

Hence, d becomes slightly bigger in this formulation—trivially so when ν is large, but noticeably bigger when ν is small. It should also be noted that Fuller et al. use the null hypothesis estimates $\hat p_{0rc}$ to compute $\widehat{\mathbf V}_{\rm srs}$.

In a later section, we evaluate Fuller et al.'s variant along with the F statistic suggested by Rao and Thomas (1989) for the second-order corrected statistic. The results were surprising—or at least, they surprised me.

Wald statistics

Prior to the work by Rao and Scott (1981, 1984), Wald tests for the test of independence for two-way tables were developed by Koch et al. (1975). Two Wald statistics have been proposed. The first is similar to the Pearson statistic in that it is based on

$$\hat Y_{rc} = \hat N_{rc} - \hat N_{r\cdot}\hat N_{\cdot c}/\hat N_{\cdot\cdot}$$

where $\hat N_{rc}$ is the estimated weighted counts for the r,cth cell, and $r = 1, \dots, R-1$ and $c = 1, \dots, C-1$. The delta-method can be used to estimate the variance of $\hat{\mathbf Y}$ (which is $\hat Y_{rc}$ stacked into a vector), and a Wald statistic can be constructed in the usual manner:

$$W = \hat{\mathbf Y}'\big(\mathbf J_{\mathbf N}\widehat{\mathbf V}(\hat{\mathbf N})\mathbf J_{\mathbf N}'\big)^{-1}\hat{\mathbf Y} \qquad \text{where} \qquad \mathbf J_{\mathbf N} = \partial\hat{\mathbf Y}/\partial\hat{\mathbf N}'$$

Another Wald statistic can be developed based on the log-linear model with predictors $[\,\mathbf 1 \mid \mathbf X_1 \mid \mathbf X_2\,]$ that was mentioned earlier. This Wald statistic is

$$W_{\rm LL} = \big(\mathbf X_2'\ln\hat{\mathbf p}\big)'\big(\mathbf X_2'\mathbf J_{\mathbf p}\widehat{\mathbf V}(\hat{\mathbf p})\mathbf J_{\mathbf p}'\mathbf X_2\big)^{-1}\big(\mathbf X_2'\ln\hat{\mathbf p}\big)$$

where $\mathbf J_{\mathbf p}$ is the matrix of first derivatives of $\ln\hat{\mathbf p}$ with respect to $\hat{\mathbf p}$, which is, of course, just a matrix with $\hat p_{rc}^{-1}$ on the diagonal and zero elsewhere. Note that this log-linear Wald statistic is undefined when there is a zero proportion in the table.

These Wald statistics can be converted to F statistics in one of two ways. One method is the standard manner: divide by the χ^2 degrees of freedom $d_0 = (R-1)(C-1)$ to get an F statistic with d_0 numerator degrees of freedom and $\nu = n_{\rm PSU} - L$ denominator degrees of freedom. This is the form of the F statistic suggested by Koch et al. (1975) and implemented in the CROSSTAB procedure of the SUDAAN software (Shah et al. 1997, Release 7.5), and it is the method used by `svytab` when the `noadjust` option is specified along with `wald` or `llwald`.

Another technique is to adjust the F statistic by using

$$F_{\rm adj} = (\nu - d_0 + 1)W/(\nu d_0) \qquad \text{with} \qquad F_{\rm adj} \sim F(d_0, \nu - d_0 + 1)$$

This is the adjustment that is done by default by `svytab`. Note that `svytest` and the other `svy` estimation commands produce adjusted F statistics by default, using exactly the same adjustment procedure. See Korn and Graubard (1990) for a justification of the procedure.

As Thomas and Rao (1987) point out, the unadjusted F statistics give anticonservative p-values (i.e., under the null, the statistics are "significant" more often than they should be) when ν is not large. Thomas and Rao (1987) explore the behavior of the adjusted F statistic for the log-linear Wald test in simulations for simple goodness-of-fit tests, and they find that even the adjusted statistic is anticonservative when ν is small. This conclusion for the log-linear Wald statistic is borne out by my simulations, which are detailed in the next section. The adjusted "Pearson" Wald statistic, however, behaves reasonably under the null in most cases.

Evaluating the statistics via simulations

At this point in the article, you the reader are at the same point where I was after I read all the papers and coded up a bunch of the possible statistics. I had a number of questions:

1. Should one use $\hat p_{rc}$ or $\hat p_{0rc}$ to compute $\widehat{\mathbf V}_{\rm srs}$ in the second-order Rao and Scott correction for $X^2_{\rm P}$ and $X^2_{\rm LR}$?
2. It is next to certain that one should adjust for the degrees of freedom $\nu = n_{\rm PSU} - L$ of the design-based variance estimate $\widehat{\mathbf V}$ by converting the corrected χ^2 statistic into an F statistic. But what should one use: the ν denominator degrees of

freedom as Fuller et al. (1986) have implemented in PC CARP or something like the more liberal νd degrees of freedom suggested by Rao and Thomas (1989)?

3. How anticonservative are the unadjusted Wald statistics (Koch et al. 1975)? Just a bit or a lot? Is the adjustment really necessary?
4. How powerful are the Wald statistics compared with the Rao and Scott corrected statistics? Are they about the same or are some statistics clearly better than others?
5. There is yet another possibility for testing the independence of two-way tables, and that is to use `svymlog`. Indeed, this was the suggestion that I made to Stata users while I worked on `svytab`. How good or bad was this recommendation?
6. Is one statistic clearly better than the others? Can researchers just use one statistic all the time and ignore all the others?

Since the theory behind all of these statistics is asymptotic, recourse to theory almost certainly will not help answer any of these questions. Simulations are obviously the thing to do. I did several simulations that attempted to evaluate the statistics under a variety of different conditions: for sparse tables and for nonsparse tables; for small and for large variance degrees of freedom ν; under the null hypothesis of independence to evaluate Type I error; and under an alternative hypothesis of nonindependence to evaluate power.

There is one point about simulations for tests of independence I cannot emphasize strongly enough: It is essentially impossible to do simulations that definitively prove anything in general. This is because there is an infinite number of distributions that one could assume for the underlying true cell proportions. Contrast this to the case of, for example, a maximum-likelihood estimator in which one can use the assumed likelihood function as the distribution for the simulations.

Hence, evaluation of the statistics should focus on failure. If a statistic shows undesirable properties (e.g., anticonservative p-values under the null, or poor power under an alternative) for a particular simulation, then it would be reasonable to assume that the statistic may fail in this manner in at least some similar situations with real data.

Another shortcoming of the following simulations must be mentioned. These statistics are intended for survey data. A common paradigm for the theoretical development of many survey estimators is that of sampling from a fixed finite population. Under this paradigm, the properties of an estimator are evaluated by considering repeated sampling from the same fixed finite population under the same survey design. Hence, to do simulations following this paradigm, one should first generate a finite population from an assumed distribution of the population (i.e., a "superpopulation") , and then repeatedly draw samples from the finite population. This whole process should then be repeated many times. This is clearly an arduous task that would require weeks of computer time. I chose to do something much simpler.

Setting up the simulations

For the simulations, I generated clusters and observations within clusters using a simple parametric model. I did not simulate stratified sampling or sampling weights. Hence, my simulations represent simple random sampling with replacement of clusters from an infinite population of clusters.

To produce two categorical variables z_1 and z_2 for the tabulation, I first generated underlying latent variables y_1 and y_2 according to a "random-effects" model:

$$y_{kij} = u_i + e_{ij} \quad \text{where} \quad u_i \sim N(0, \rho_\text{c}) \quad \text{and} \quad e_{ij} \mid u_i \sim N(0, 1-\rho_\text{c})$$

Hence, unconditionally, $y_k \sim N(0,1)$ and the intra-cluster correlation is ρ_c. To simulate the null hypothesis of independence, y_1 and y_2 were independently generated. To simulate an alternative hypothesis of nonindependence, y_1 and y_2 were given correlations of roughly ρ_a by first generating independent y_1' and y_2' and then setting

$$\begin{aligned} y_1 &= y_1' \\ y_2 &= \rho_\text{a}\, y_1' + \sqrt{1-\rho_\text{a}^2}\, y_2' \end{aligned}$$

To generate z_k with M categories, y_k was categorized based on fixed normal quantiles. For the nonsparse tables simulations, quantiles were derived from equally spaced probabilities:

$$z_k = q \quad \text{if and only if} \quad \Phi^{-1}((q-1)/M) < y_k \le \Phi^{-1}(q/M) \quad \text{where} \quad q = 1, 2, \ldots, M$$

where Φ is the standard cumulative normal. For sparse tables simulations, the quantile cut points were varied so that one category was infrequent. This frequency was chosen so that zero cells were generated in tables roughly half of the time.

$R \times C$ tables were generated with $2 \leq R, C \leq 5$. The intra-cluster correlation was $\rho_c = 0.25$ for all reported simulations. For the simulations representing small variance degrees of freedom, 20 clusters ($\nu = 19$) ranging in size from 30 to 70 observations where chosen, for a total of approximately 1,000 observations. The cluster sizes were randomly generated once, and then left fixed for subsequent simulations. With $\nu = 19$ and table degrees of freedom $d_0 = (R-1)(C-1)$ ranging from 1 to 16, we should be able to see what happens when d_0 approaches ν. Note that $\nu = 19$ is not unrealistic. The NHANES II data shown earlier have $\nu = 31$.

For the simulations representing large variance degrees of freedom, 200 clusters ($\nu = 199$) ranging in size from 3 to 10 observations where chosen, for a total of approximately 1300 observations.

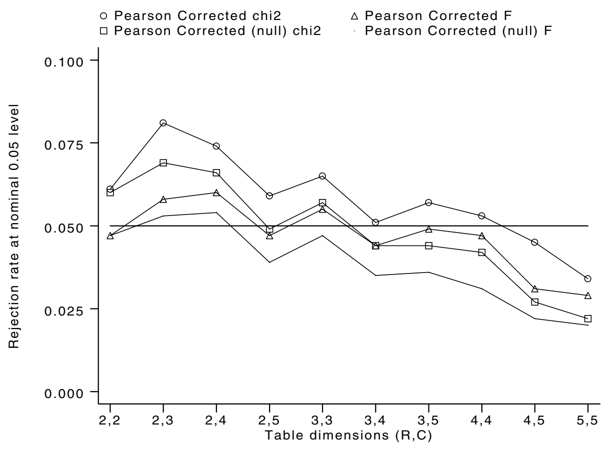
The approximate correlation ρ_a between the latent variables that was used to generate an alternative hypothesis was chosen based on R and C and on the type of simulation (sparse or nonsparse, small or large ν) to give power of around 0.5 to 0.9 for most of the statistics. Because of this arbitrariness, only the relative power of the statistics can be assessed. That is, absolute assessments of power across different simulations cannot be made.

For all simulations, 1,000 replications were run.

Simulations of the null for nonsparse and sparse tables for small ν

Simulations of the null hypothesis of independence for small ν should be a good trial of the denominator degrees of freedom for the Rao and Scott corrected F statistics and also a good trial of the unadjusted versus adjusted Wald statistics.

Figure 1 shows the observed rejection rate at a nominal 0.05 level for four variants of the Rao and Scott corrected Pearson statistic. Shown are the two second-order corrected χ^2 statistics $X^2_{\mathrm{P}}(\hat{\delta}_., \hat{a})$ from equation (5). The one labeled "null" is corrected using $\hat{p}_{0rc}$ to compute $\widehat{\mathbf{V}}_{\mathrm{srs}}$ in design effects matrix $\widehat{\Delta}$, and I refer to this as the null-corrected statistic. The other one uses $\hat{p}_{rc}$, and I refer to this as the default-corrected statistic since it is `svytab`'s default.



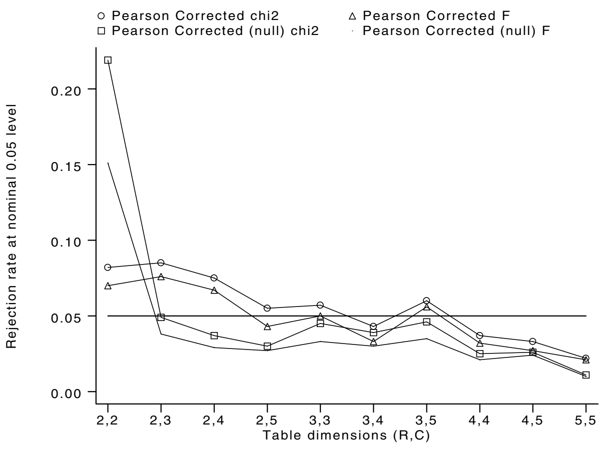


Figure 1. Variants of the Pearson statistic under null for nonsparse (left) and sparse (right) tables (variance df=19).

Also shown in Figure 1 are these two χ^2 statistics turned into F statistics with denominator degrees of freedom νd according to the suggestion of Rao and Thomas (1989). The left graph of the nonsparse tables simulation shows that the corrected χ^2 statistics are anticonservative (i.e., declare significance too frequently) for the smaller tables. It is surprising, however, how well the χ^2 statistics do for the larger tables where $d_0 = (R-1)(C-1)$ approaches ν. The corrected F statistics do better for the smaller nonsparse tables. For the nonsparse tables, the null-corrected F statistic is noticeably more conservative than the default-corrected statistic, except for the $R = 2, C = 2$ table.

The right graph of Figure 1 shows a sparse tables simulation for the same statistics. An anomaly appears for the null-corrected statistics for the $R = 2, C = 2$ table: the rejection rate of the simulation is > 0.15 at a nominal 0.05 level. After the $R = 2, C = 2$ table, the null-corrected statistics dive to levels more conservative than the default-corrected statistics.

Similar graphs (Figure 2) can be made for the corresponding four variants of the corrected likelihood-ratio statistics. The graph for the nonsparse tables is essentially identical to the left graph of Figure 1. This is not surprising since the uncorrected Pearson statistic is usually numerically similar to the uncorrected likelihood-ratio statistic when tables are not sparse.

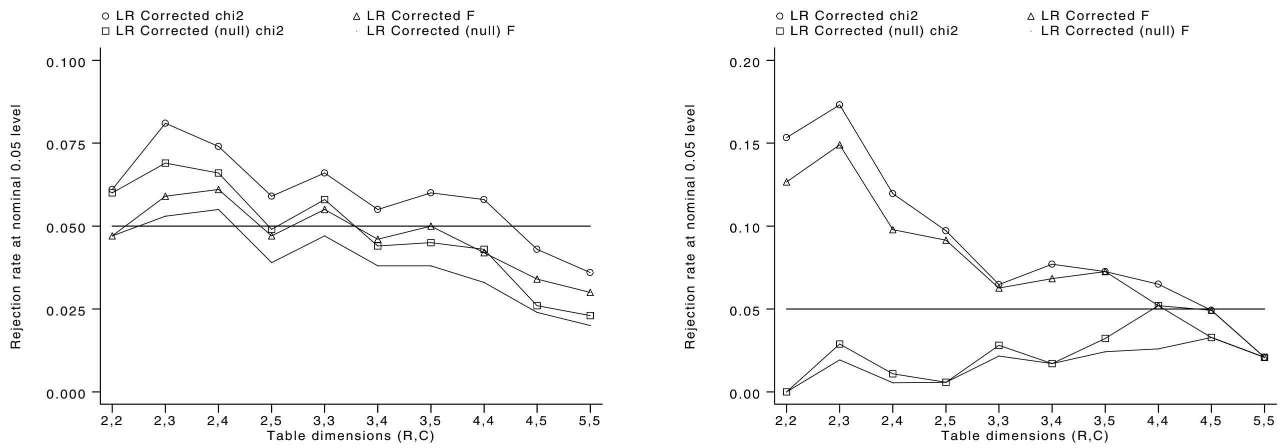


Figure 2. Variants of the likelihood-ratio statistic under null for nonsparse (left) and sparse (right) tables (variance df=19).

In the sparse tables simulations for the likelihood-ratio statistics (right graph of Figure 2), there are many instances in which these statistics are not defined because of a zero cell. Of those sparse tables for which the statistics could be computed, the null-corrected likelihood-ratio statistics display severe conservatism and the default-corrected likelihood-ratio statistics exhibit unsatisfactory anticonservatism, especially for the smaller tables (e.g., rejection rates > 0.1 for $R = 2, C = 2$ and $R = 2, C = 3$).

The Fuller et al. PC CARP variant

Figure 3 compares the statistic implemented by Fuller et al. (1986) in the PC CARP software with the default-corrected Pearson F statistic (`svytab`'s default statistic). The left graph of Figure 3 shows that the Fuller et al. statistic is quite conservative for all except the smaller tables. Recall that the Fuller et al. statistic uses ν denominator degrees of freedom, whereas the Pearson F statistic uses νd.

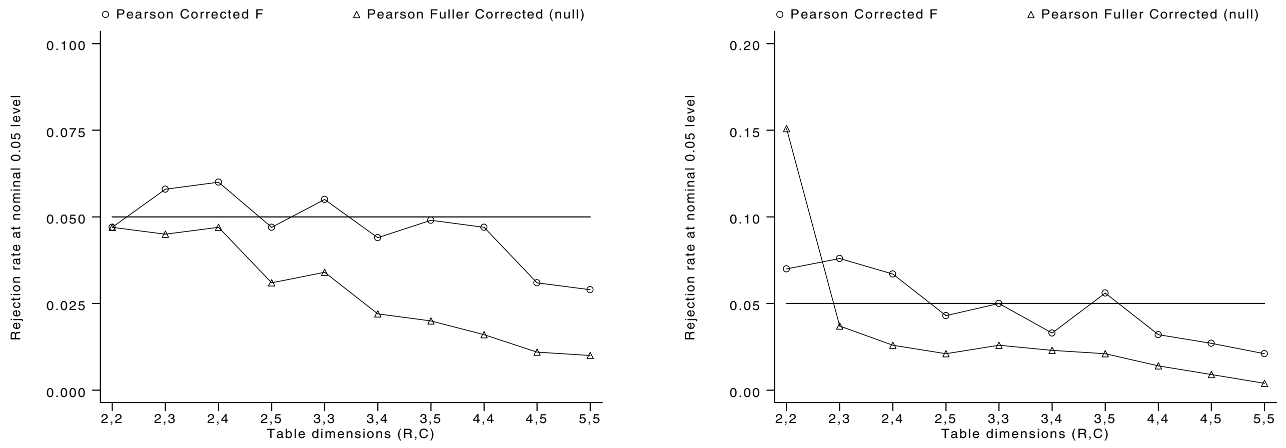


Figure 3. Fuller et al. (1986) correction vs. `svytab`'s default Pearson under null for nonsparse (left) and sparse (right) tables (variance df=19).

The right graph of Figure 3 shows the sparse tables simulation. Since the Fuller et al. variant is a null-corrected statistic, it also exhibits the same anomalous anticonservatism for the $R = 2, C = 2$ table as do the null-corrected Pearson statistics in the right graph of Figure 1. One could apply Fuller et al. formulas to the default correction (and I have done so in my simulations), but this also leads to appreciable conservatism for the larger tables. The ν denominator degrees of freedom of the Fuller et al. variant simply appear to be too strict.

One could increase the denominator degrees of freedom of the Fuller et al. variant to νd, but since the numerator degrees of freedom for the Fuller et al. variant (equation (8)) are larger than the numerator degrees of freedom calculated according to Rao and Scott's (1984) formulas, this leads to a uniformly anticonservative statistic.

So, although the theory behind the Fuller et al. (1986) variant appears compelling, simulations show it to be problematic.

Unadjusted and adjusted Wald statistics

Figure 4 shows the behavior of the unadjusted Wald statistics (Koch et al. 1975) under the null hypothesis of independence for nonsparse and sparse tables for $\nu = 19$. Note the scale of the left axis. As the table degrees of freedom $d_0 = (R-1)(C-1)$ approach the variance degrees of freedom $\nu = 19$, the rejection rate approaches 1. Clearly, this behavior makes an unadjusted Wald statistic unsuitable for use in these situations.

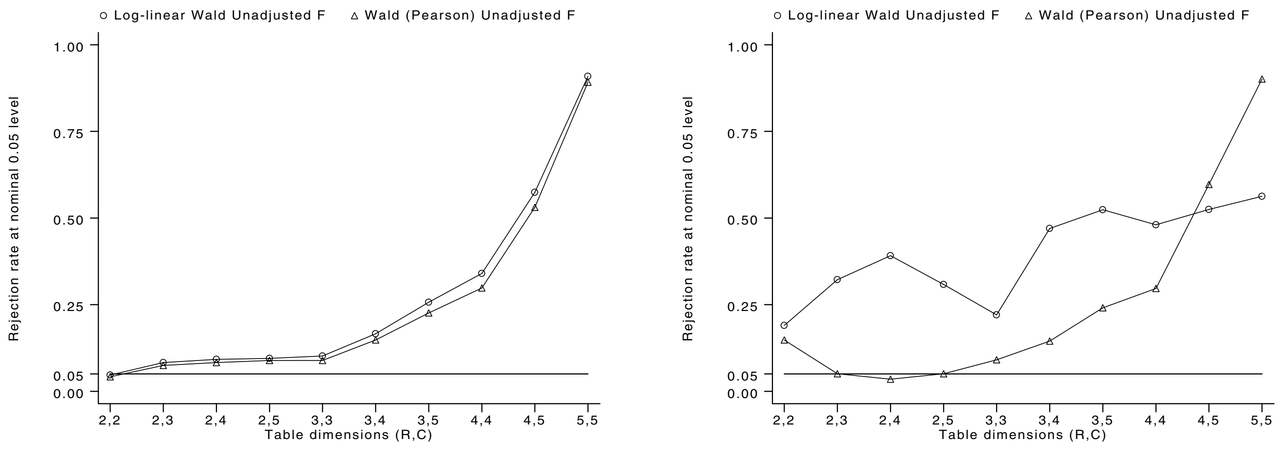


Figure 4. Unadjusted Wald statistics under null for nonsparse (left) and sparse (right) tables (variance df=19).

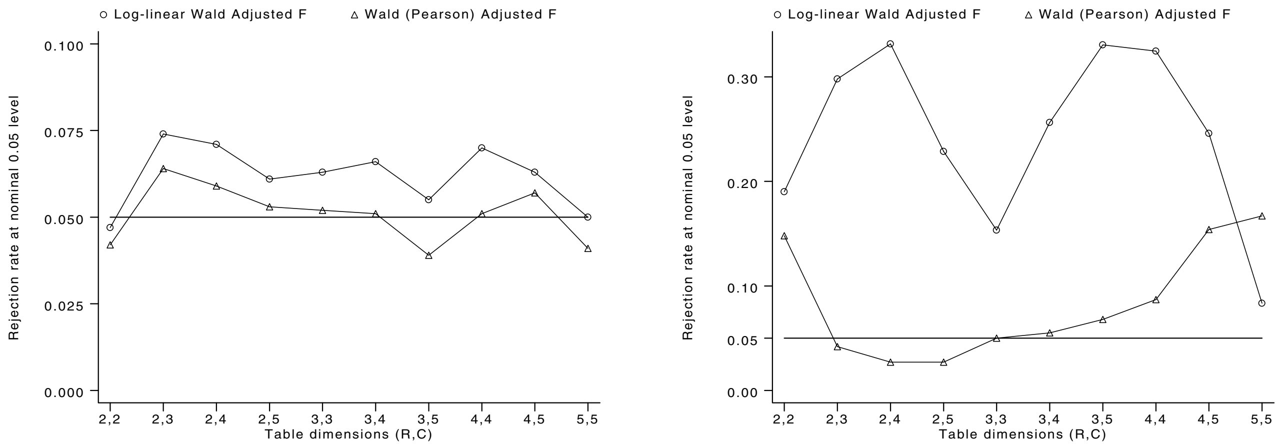


Figure 5. Adjusted Wald statistics under null for nonsparse (left) and sparse (right) tables (variance df=19).

The adjusted "Pearson" Wald statistic (Figure 5) behaves much better. For nonsparse tables, the observed rejection rate is close to the nominal 0.05 level. For sparse tables, the statistic is very anticonservative for the $R = 2, C = 2$ table like the null-corrected statistics, and also very anticonservative for some of the larger tables.

In agreement with the observations of Thomas and Rao (1987), the adjusted log-linear Wald statistic is almost uniformly anticonservative, and is especially poor for sparse tables. Note that the simulation results for the log-linear Wald statistic are effectively based on fewer replications for sparse tables (Figures 3 and 4, right graphs) since the statistic was often undefined because of a zero cell in the table.

Using `svymlog` for a test of independence

The `svymlog` command can be used to conduct a test of independence for two-way tables. One merely does the following:

```
. xi: svymlog z1 i.z2
. svytest I*
```

For those readers not familiar with `xi`, note that the `I*` variables are dummies for the `z2` categories. Note also that `svytest`, by default, computes an adjusted F statistic.

This command sequence using `svymlog` was included in all the simulations. The results are easy to describe: for every nonsparse tables simulation, `svymlog` yielded the same results as the adjusted log-linear Wald statistic. For sparse tables, `svymlog` was uniformly and seriously anticonservative.

Simulations of the null for nonsparse and sparse tables for large ν

Since all of the theory for the various statistics is asymptotic as the number of PSUs (clusters) goes to infinity, one expects that all the statistics should be well-behaved under the null—for nonsparse tables, at least. Simulations of nonsparse tables with $\nu = 199$ bear this out for all statistics except for the unadjusted Wald statistics, which are surprisingly anticonservative for some of the larger tables; see the left graph of Figure 6.

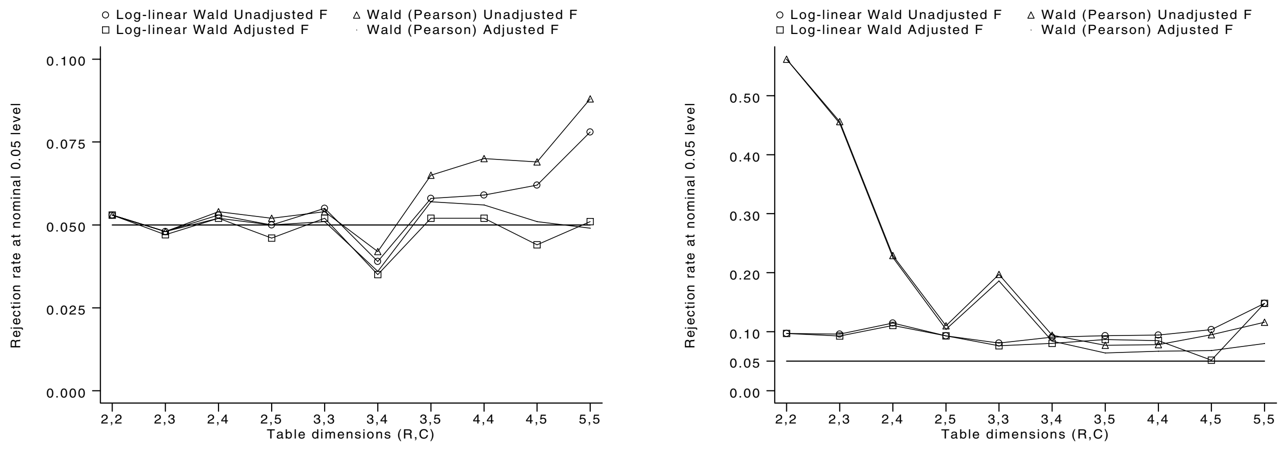


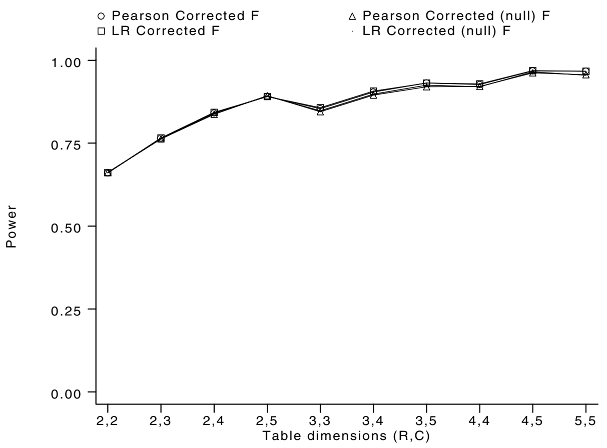
Figure 6. Unadjusted and adjusted Wald statistics under null for large variance degrees of freedom (df=199) for nonsparse (left) and sparse (right) tables.

Sparse tables, however, are a different story; see the right graph of Figure 6. All of the Wald statistics exhibit severe problems. The "Pearson" Wald statistic, both unadjusted and adjusted, is extremely anticonservative for the smaller tables. Inexplicably, its behavior is worse here than it is when the variance degrees of freedom are small. The log-linear Wald statistic is again uniformly anticonservative.

The null-corrected Rao and Scott statistics also exhibit some anomalies for these sparse tables simulations. The null-corrected Pearson is extremely anticonservative for the $R = 2, C = 2$ table (rejection rate > 0.50 at a nominal 0.05 level), but conservative for the other tables. The null-corrected likelihood-ratio statistic appears to be extremely conservative (rejection rate < 0.02) for those sparse tables in the simulation that did not contain a zero cell. The default-corrected statistics, however, do much better. The default-corrected Pearson has a rejection rate of 0.04–0.06 for all tables in the large variance degrees of freedom simulation of sparse tables.

Power for nonsparse and sparse tables for small ν

Figure 7 shows the relative power of the default- and null-corrected Pearson and likelihood-ratio F statistics for small variance degrees of freedom ($\nu = 19$). Again, it must be kept in mind that only relative comparisons among the statistics can be made, since the alternative hypothesis was changed as the table size changed. The left graph of Figure 7 shows that these four statistics all have essentially identical power for nonsparse tables.



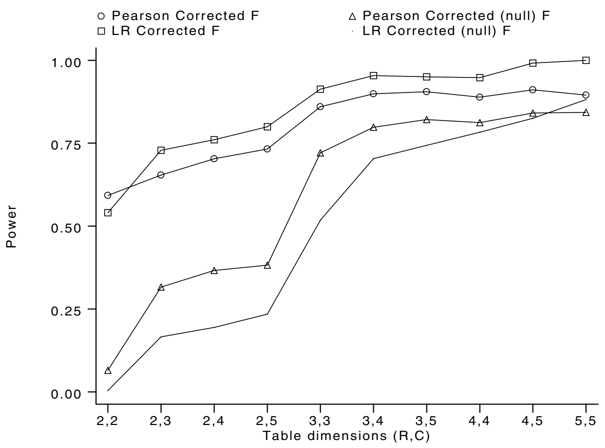
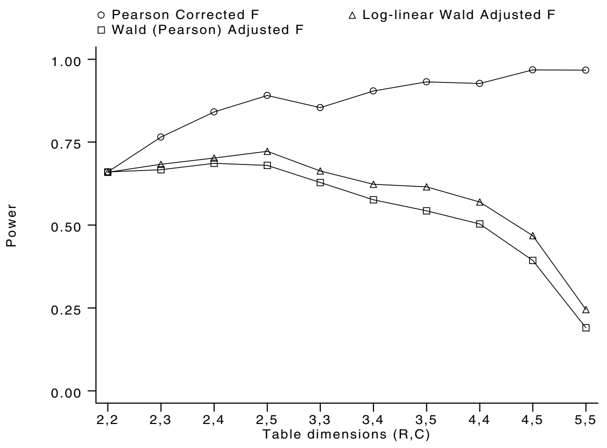


Figure 7. Power of corrected Pearson and likelihood-ratio statistics for nonsparse (left) and sparse (right) tables (variance df=19).

For sparse tables, the null-corrected statistics exhibit extremely poor power for $R = 2, C = 2$. Indeed, the rejection rate for the null-corrected Pearson F statistic was *lower* for this alternative hypothesis than it was under the null hypothesis for an otherwise similar simulation (Figure 1, right graph). This is puzzling. I can only say that the behavior of the null-corrected statistics appears to be very erratic for sparse $R = 2, C = 2$ tables in these simulations. The default-corrected statistics show no such erratic behavior for sparse $R = 2, C = 2$ tables. The default-corrected Pearson, for example, has a slightly inflated rejection rate under the null (0.07), but good power under this alternative (> 0.50).

Figure 8 compares the power of the adjusted Wald statistics with that of the default-corrected Pearson F statistic for small variance degrees of freedom ($\nu = 19$). For nonsparse tables, the default-corrected Pearson is more powerful than the adjusted Wald statistics, except for the $R = 2, C = 2$ table for which the powers are the same. The differences are particularly dramatic for larger tables. Sparse tables (Figure 8, right graph) also show this fall-off for larger tables.



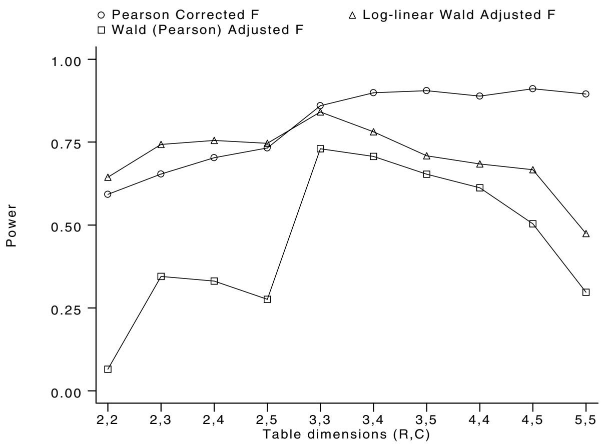


Figure 8. Power of adjusted Wald statistics versus `svytab`'s default Pearson variant for nonsparse (left) and sparse (right) tables (variance df=19).

It is not reasonable to include unadjusted Wald statistics in power comparisons for small variance degrees of freedom since they do such a bad job of controlling Type I error under the null (Figure 4) in this situation. However, it should be noted that despite this fact, the power of the default-corrected Pearson F statistic is either better than or about the same as the power of the unadjusted Wald statistics.

Power for nonsparse and sparse tables for large ν

Power for all statistics was similar for nonsparse tables when the variance degrees of freedom were large ($\nu = 199$). All of the Rao and Scott corrected statistics had essentially identical power for nonsparse tables. These statistics held a slight advantage of about 5% over the adjusted Wald statistics for all but the smallest tables, where the power was the same.

The sparse tables simulation again provoked erratic behavior for some of the statistics in the smaller tables. Again, the default-corrected Pearson F statistic was superior. Its power was either about the same or significantly better than the other statistics.

Conclusions

Rather surprisingly, the simulations yielded unequivocal answers to the questions that I posed earlier.

1. Should one use $\hat{p}_{rc}$ (default-corrected) or $\hat{p}_{0rc}$ (null-corrected) in the second-order Rao and Scott correction? The answer appears to be $\hat{p}_{rc}$. Using $\hat{p}_{0rc}$ results in some erratic behavior for sparse tables—at least in the simulations I ran. For nonsparse tables, it does not matter which is used; they both yield similar statistics.

2. What should one use: the ν denominator degrees of freedom as Fuller et al. (1986) have implemented in PC CARP or something like the more liberal νd degrees of freedom suggested by Rao and Thomas (1989)? The Fuller et al. variant is too conservative for all but the smallest tables. The Rao and Thomas suggestion appears better, although it produces slightly anticonservative statistics for the smallest tables.

3. How anticonservative are the unadjusted Wald statistics (Koch et al. 1975)? They can be dangerously anticonservative. For large tables and small variance degrees of freedom (i.e., a small to moderate number of PSUs), they can reject the null hypothesis most of the time even though the null hypothesis of independence is true. Only when the variance degrees of freedom are very large ($>$ 1000) and the unadjusted and adjusted statistics are approximately equal would the use of the unadjusted statistic be justified. The adjusted log-linear Wald statistics appear to be uniformly anticonservative for small to moderate variance degrees of freedom. The "Pearson" Wald statistic (the `wald` option of `svytab`) appears to have reasonable properties under the null hypothesis in this situation. However, like the Rao and Scott null-corrected statistics, it can exhibit erratic behavior in small sparse tables.

4. How powerful are the Wald statistics compared with the Rao and Scott corrected statistics? The Rao and Scott corrected statistics are significantly more powerful than the adjusted Wald statistics for small to moderate variance degrees of freedom, especially for large tables.

5. How good is `svymlog` for a test of independence? At best, it is as good as the adjusted log-linear Wald statistic, which is to say fairly bad.

6. Is one statistic clearly better than the others? All of the statistics except the default-corrected Pearson F statistic collected black marks in the simulations. The default-corrected Pearson F statistic was slightly anticonservative for small tables when the variance degrees of freedom were small, but other than that it did remarkably well. It exhibited reasonably true Type I error for small sparse tables when other statistics behaved erratically. However, these simulations only used one particular model of sparse tables, so it would be unwarranted to generalize this observation to all sparse tables. The corrected likelihood-ratio F statistic is an equally good choice for nonsparse tables; but it is no better, so there is no reason to ever use it over the corrected Pearson F statistic. Hence, I recommend the use of the default `pearson` statistic of `svytab` in all situations. Conversely, I recommend ignoring all the other statistics in all situations, unless you have a particular pedagogical interest in them.

Methods and formulas

We assume here that readers are familiar with the *Methods and Formulas* section of the [R] **svymean** entry of the *Stata Reference Manual.*

For a table of R rows by C columns with cells indexed by r, c, let

$$y_{(rc)hij} = \begin{cases} 1 & \text{if the } hij\text{th element of the data is in the } r,c\text{ th cell} \\ 0 & \text{otherwise} \end{cases}$$

where $h = 1, \ldots, L$, indexes strata; $i = 1, \ldots, n_h$, indexes PSUs; and $j = 1, \ldots, m_{hi}$, indexes elements in the PSU. Weighted cell counts (the `count` option) are

$$\hat{N}_{rc} = \sum_{h=1}^{L} \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} w_{hij}\, y_{(rc)hij}$$

where w_{hij} are the weights, if any. If a variable x_{hij} is specified with the `tab()` option, $\hat{N}_{rc}$ becomes

$$\hat{N}_{rc} = \sum_{h=1}^{L} \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} w_{hij}\, x_{hij}\, y_{(rc)hij}$$

Let

$$\hat{N}_{r\cdot} = \sum_{c=1}^{C} \hat{N}_{rc}\,, \qquad \hat{N}_{\cdot c} = \sum_{r=1}^{R} \hat{N}_{rc}\,, \qquad \text{and} \qquad \hat{N}_{\cdot\cdot} = \sum_{r=1}^{R}\sum_{c=1}^{C} \hat{N}_{rc}$$

Estimated cell proportions are $\hat{p}_{rc} = \hat{N}_{rc}/\hat{N}_{\cdot\cdot}$. Estimated row proportions are $\hat{p}_{\text{row}\,rc} = \hat{N}_{rc}/\hat{N}_{r\cdot}$. Estimated column proportions are $\hat{p}_{\text{col}\,rc} = \hat{N}_{rc}/\hat{N}_{\cdot c}$.

$\hat{N}_{rc}$ is a total and the proportion estimators are ratios, and their variances can be estimated using linearization methods as outlined in [R] **svymean**. `svytab` computes the variance estimates using the same driver program that `svymean`, `svyratio`, and `svytotal` use. Hence, `svytab` produces exactly the same standard errors as these commands would.

Confidence intervals for proportions are calculated using a logit transform so that the endpoints lie between 0 and 1. Let $\hat{p}$ be an estimated proportion and $\hat{s}$ an estimate of its standard error. Let $f(\hat{p}) = \ln(\hat{p}/(1-\hat{p}))$ be the logit transform of the proportion. In this metric, an estimate of the standard error is $f'(\hat{p})\hat{s} = \hat{s}/\hat{p}(1-\hat{p})$. Thus, a $100(1-\alpha)\%$ confidence interval in this metric is $\ln(\hat{p}/(1-\hat{p})) \pm t_{1-\alpha/2,\nu}\,\hat{s}/\hat{p}(1-\hat{p})$, where $t_{1-\alpha/2,\nu}$ is the $(1-\alpha/2)$th quantile of Student's t distribution. The endpoints of this confidence interval are transformed back to the proportion metric using $f^{-1}(y) = \exp(y)/(1+\exp(y))$. Hence, the displayed confidence intervals for proportions are

$$f^{-1}\left(\ln\left(\frac{\hat{p}}{1-\hat{p}}\right) \pm \frac{t_{1-\alpha/2,\nu}\,\hat{s}}{\hat{p}(1-\hat{p})}\right)$$

Confidence intervals for weighted counts are untransformed and are identical to the intervals produced by `svytotal`.

The Rao and Scott (1984) second-order correction to the Pearson statistic X^2_{P} (equation (1)) and the likelihood-ratio statistic X^2_{LR} (equation (2)) is calculated by first estimating the design effects matrix

$$\widehat{\Delta} = \left(\mathbf{C}'\mathbf{D}_{\hat{\mathbf{p}}}^{-1}\widehat{\mathbf{V}}_{\text{srs}}\mathbf{D}_{\hat{\mathbf{p}}}^{-1}\mathbf{C}\right)^{-1}\left(\mathbf{C}'\mathbf{D}_{\hat{\mathbf{p}}}^{-1}\widehat{\mathbf{V}}\mathbf{D}_{\hat{\mathbf{p}}}^{-1}\mathbf{C}\right)$$

This formulation is different from that of equation (4), but it yields numerically identical results for the correction and is easier to compute. Here $\mathbf{C}$ is a contrast matrix that is any $RC \times (R-1)(C-1)$ full-rank matrix that is orthogonal to $[\,\mathbf{1}\,|\,\mathbf{X}_1]$ (i.e., $\mathbf{C}'\mathbf{1} = \mathbf{0}$ and $\mathbf{C}'\mathbf{X}_1 = \mathbf{0}$), where $\mathbf{X}_1$ is a $RC \times (R+C-2)$ matrix of row and column "main effects"; see the description following equation (4). $\widehat{\mathbf{V}}$ is the variance estimate for the cell proportions under the survey design (i.e., the same estimate that would be produced by `svyratio`). $\widehat{\mathbf{V}}_{\text{srs}}$ is the variance estimate assuming simple random sampling. $\mathbf{D}_{\hat{\mathbf{p}}}$ is a diagonal matrix with the estimated proportions on the diagonal.

The observed proportions $\hat{p}_{rc}$ or the proportions under the null hypothesis of independence, $\hat{p}_{0rc} = \hat{p}_{r\cdot}\hat{p}_{\cdot c}$, can be used for the computation of $\widehat{\mathbf{V}}_{\text{srs}}$ and $\mathbf{D}_{\hat{\mathbf{p}}}$. For the former choice, $\widehat{\mathbf{V}}_{\text{srs}}$ becomes a matrix with diagonal elements $\hat{p}_{rc}(1-\hat{p}_{rc})/n$ and off-diagonal elements $-\hat{p}_{rc}\hat{p}_{st}/n$. The former choice is used by default for the `pearson` and `lr` statistics. The latter is used when the `null` option is specified.

Note that when the $\hat{p}_{rc}$ estimates are used and one of the estimates is zero, the corresponding variance estimate is also zero; hence, the corresponding element for $\mathbf{D}_{\hat{\mathbf{p}}}^{-1}$ is immaterial for the computation of $\widehat{\Delta}$.

Rao and Scott's (1984) second-order correction is used to produce F statistics:

$$F_{\text{P}} = \frac{X^2_{\text{P}}}{\text{tr}\,\widehat{\Delta}} \quad \text{with} \quad F_{\text{P}} \sim F(d, \nu d) \quad \text{where} \quad d = \frac{(\text{tr}\,\widehat{\Delta})^2}{\text{tr}\,\widehat{\Delta}^2} \quad \text{and} \quad \nu = \left(\sum_{h=1}^{L} n_h\right) - L$$

The denominator degrees of freedom are based on a suggestion of Rao and Thomas (1989). The correction for the likelihood-ratio statistic X^2_{LR} is identical.

The Wald statistics computed by `svytab` with the `wald` and `llwald` options are detailed in the earlier section titled *Wald statistics.*

References

Fuller, W. A., W. Kennedy, D. Schnell, G. Sullivan, H. J. Park. 1986. PC CARP. Ames, IA: Statistical Laboratory, Iowa State University.

Koch, G. G., D. H. Freeman, Jr., and J. L. Freeman. 1975. Strategies in the multivariate analysis of data from complex surveys. *International Statistical Review* 43: 59–78.

Korn, E. L. and B. I. Graubard. 1990. Simultaneous testing of regression coefficients with complex survey data: use of Bonferroni t statistics. *The American Statistician* 44: 270–276.

McDowell, A., A. Engel, J. T. Massey, and K. Maurer. 1981. Plan and operation of the Second National Health and Nutrition Examination Survey, 1976–1980. *Vital and Health Statistics* 15(1). National Center for Health Statistics, Hyattsville, MD.

Rao, J. N. K. and A. J. Scott. 1981. The analysis of categorical data from complex sample surveys: chi-squared tests for goodness of fit and independence in two-way tables. *Journal of the American Statistical Association* 76: 221–230.

——. 1984. On chi-squared tests for multiway contingency tables with cell proportions estimated from survey data. *Annals of Statistics* 12: 46–60.

Rao, J. N. K. and D. R. Thomas. 1989. Chi-squared tests for contingency tables. In *Analysis of Complex Surveys*, ed. C. J. Skinner, D. Holt, and T. M. F. Smith, Ch. 4, 89–114. New York: John Wiley & Sons.

Shah, B. V., B. G. Barnwell, and G. S. Bieler. 1997. *SUDAAN User's Manual, Release 7.5.* Research Triangle Park, NC: Research Triangle Institute.

Thomas, D. R. and J. N. K. Rao. 1987. Small-sample comparisons of level and power for simple goodness-of-fit statistics under cluster sampling. *Journal of the American Statistical Association* 82: 630–636.

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, *stata*, to the manufacturers of Stata for their exclusive use.

Guidelines for authors

The Stata Technical Bulletin (STB) is a journal that is intended to provide a forum for Stata users of all disciplines and levels of sophistication. The STB contains articles written by StataCorp, Stata users, and others.

Articles include new Stata commands (ado-files), programming tutorials, illustrations of data analysis techniques, discussions on teaching statistics, debates on appropriate statistical techniques, reports on other programs, and interesting datasets, announcements, questions, and suggestions.

A submission to the STB consists of

1. An insert (article) describing the purpose of the submission. The STB is produced using plain TeX so submissions using TeX (or LaTeX) are the easiest for the editor to handle, but any word processor is appropriate. If you are not using TeX and your insert contains a significant amount of mathematics, please FAX (409–845–3144) a copy of the insert so we can see the intended appearance of the text.

2. Any ado-files, `.exe` files, or other software that accompanies the submission.

3. A help file for each ado-file included in the submission. See any recent STB diskette for the structure a help file. If you have questions, fill in as much of the information as possible and we will take care of the details.

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

5. Files containing the graphs to be included in the insert. If you have used STAGE to edit the graphs in your submission, be sure to include the `.gph` files. Do not add titles (e.g., “Figure 1: ...”) to your graphs as we will have to strip them off.

The easiest way to submit an insert to the STB is to first create a single “archive file” (either a `.zip` file or a compressed `.tar` file) containing all of the files associated with the submission, and then email it to the editor at `stb@stata.com` either by first using `uuencode` if you are working on a Unix platform or by attaching it to an email message if your mailer allows the sending of attachments. In Unix, for example, to email the current directory and all of its subdirectories:

```
tar -cf - . | compress | uuencode xyzz.tar.Z > whatever
mail stb@stata.com < whatever
```

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:	+972-4-9541153
Fax:	+972-6-6554254
Email:	sasconsl@actcom.co.il
Countries served:	Israel

Company:	Dittrich & Partner Consulting
Address:	Prinzenstrasse 2 D-42697 Solingen Germany
Phone:	+49-212-3390 200
Fax:	+49-212-3390 295
Email:	evhall@dpc.de
URL:	http://www.dpc.de
Countries served:	Germany, Austria, Italy

For the most up-to-date list of Stata distributors, visit http://www.stata.com

Company:	IEM
Address:	P.O. Box 2222 PRIMROSE 1416 South Africa
Phone:	27-11-8286169
Fax:	27-11-8221377
Email:	iem@hot.co.za
Countries served:	South Africa, Botswana, Lesotho, Namibia, Mozambique, Swaziland, Zimbabwe

Company:	MercoStat Consultores
Address:	9 de junio 1389 CP 11400 MONTEVIDEO Uruguay
Phone:	598-2-613-7905
Fax:	+Same
Email:	mercost@adinet.com.uy
Countries served:	Uruguay, Argentina, Brazil, Paraguay

Company:	Metrika Consulting
Address:	Mosstorpsvagen 48 183 30 Taby Stockholm Sweden
Phone:	+46-708-163128
Fax:	+46-8-7924747
Email:	sales@metrika.se
Countries served:	Sweden, Baltic States, Denmark, Finland, Iceland, Norway

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:	info@ritme.com
URL:	http://www.ritme.com
Countries served:	France, Belgium, Luxembourg, Switzerland

Company:	Smit Consult
Address:	Doormanstraat 19 5151 GM Drunen Netherlands
Phone:	+31-416-378 125 +31-416-378 385
Email:	J.A.C.M.Smit@smitcon.nl
URL:	http://www.smitconsult.nl
Countries served:	Netherlands

(*List continued on next page*)

International Stata Distributors

(*Continued from previous page*)

Company:	Survey Design & Analysis Services P/L
Address:	249 Eramosa Road West Moorooduc VIC 3933 Australia
Phone:	+61-3-59788329
Fax:	+61-3-59788623
Email:	sales@survey-design.com.au
URL:	http://survey-design.com.au
Countries served:	Australia, New Zealand

Company:	Timberlake Consultants
Address:	47 Hartfield Crescent West Wickham Kent BR4 9DW United Kingdom
Phone:	+44 181 4620495
Fax:	+44 181 4620493
Email:	info@timberlake.co.uk
URL:	http://www.timberlake.co.uk
Countries served:	United Kingdom, Eire

Company:	Timberlake Consulting S.L.
Address:	Calle Montecarmelo n°36 Bajo 41011 Seville Spain
Phone:	+34.5.428.40.94
Fax:	+34.5.428.40.94
Email:	timberlake@zoom.es
Countries served:	Spain

Company:	Timberlake Consultores, Lda.
Address:	Praceta do Comércio, 13 - 9° Dto. Quinta Grande 2720 Alfragide Portugal
Phone:	+351 (0)1 471 9337
Fax:	+351 (0)1 471 9337
Telemóvel:	+351 (0)931 6272 55
Email:	timberlake.co@mail.telepac.pt
Countries served:	Portugal

Company:	Unidost A.S.
Address:	Rihtim Cad. Polat Han D:38 Kadikoy 81320 ISTANBUL Turkey
Phone:	+90-216-4141958
Fax:	+90-216-3368923
Email:	info@unidost.com
URL:	http://abone.turk.net/unidost
Countries served:	Turkey