STATA
TECHNICAL
BULLETIN

May 1998
STB-43

A publication to promote communication among Stata users

Contents of this issue — page

an66 STB-37–STB-42 available in bound format

Patricia Branton, Stata Corporation, stata@stata.com

The seventh year of the *Stata Technical Bulletin* (issues 37–42) has been reprinted in a bound book called *The Stata Technical Bulletin Reprints, Volume 7*. The volume of reprints is available from StataCorp for $25, plus shipping. Or, you may purchase all seven reprint volumes for $119, plus shipping, or just the most recent three volumes for $65, plus shipping. Authors of inserts in STB-37–STB-42 will automatically receive the book at no charge and need not order.

This book of reprints includes everything that appeared in issues 37–42 of the STB. As a consequence, you do not need to purchase the reprints if you saved your STBs. However, many subscribers find the reprints useful since they are bound in a convenient volume. Our primary reason for reprinting the STB, though, is to make it easier and cheaper for new users to obtain back issues. For those not purchasing the *Reprints*, note that *zz8* in this issue provides a cumulative index for the seventh year of the original STBs.

dm55 Generating sequences and patterns of numeric data: an extension to egen

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

A common task in Stata is the creation of repeating patterns and linear sequences of numeric data. The **seq** command of Cox (1997) makes this simple to do. In this insert we describe an extension to **egen** called **fill** for creating sequences. **fill()** can create ascending or descending lists, including negative numbers and complex repeating patterns.

As **fill()** is an extension to **egen**, the actual filename for the program is **_gfill.ado**. It should be placed in the personal ado directory and used just like any other **egen** function. This function will also be included in the next release of Stata. Until then, to view the help file for **fill()**, type **help fill** at the Stata command prompt.

Syntax

The syntax of **fill()** is

egen [*type*] *varname* **= fill(***numlist***)**

fill() follows much the same syntax as many other **egen** functions. The *numlist* must contain only numeric values. Alphanumeric characters are not supported.

Examples

To create a sequence of numbers, simply “show” **fill()** how the sequence should look. It must be a linear progression in order to produce the results you would expect. Geometric progressions are not understood.

```
. egen i=fill(1 2)

. egen x=fill(-10 -9)

. egen y=fill(1 3 5)

. egen z=fill(1 1 1 2 2 2)

              i          x          y          z
  1.          1        -10          1          1
  2.          2         -9          3          1
  3.          3         -8          5          1
  4.          4         -7          7          2
  5.          5         -6          9          2
  6.          6         -5         11          2
  7.          7         -4         13          3
  8.          8         -3         15          3
  9.          9         -2         17          3
 10.         10         -1         19          4
 11.         11          0         21          4
 12.         12          1         23          4
 13.         13          2         25          5
 14.         14          3         27          5
 15.         15          4         29          5
 16.         16          5         31          6
 17.         17          6         33          6
 18.         18          7         35          6
 19.         19          8         37          7
 20.         20          9         39          7
```

To produce repeating patterns, you must create the pattern definition by repeating the pattern twice in the numlist. For example, to produce a pattern such as 1, 3, 2, 4, 1, 3, 2, 4 ... issue commands like

```
. egen j=fill(1 3 2 4 1 3 2 4)
. egen k=fill(-2 -1 0 1 2 -2 -1 0 1 2)
. egen l=fill(1 1 2 2 1 1 2 2)
. egen m=fill(0 0 1 0 0 1)
            j         k         l         m
  1.        1        -2         1         0
  2.        3        -1         1         0
  3.        2         0         2         1
  4.        4         1         2         0
  5.        1         2         1         0
  6.        3        -2         1         1
  7.        2        -1         2         0
  8.        4         0         2         0
  9.        1         1         1         1
 10.        3         2         1         0
 11.        2        -2         2         0
 12.        4        -1         2         1
 13.        1         0         1         0
 14.        3         1         1         0
 15.        2         2         2         1
 16.        4        -2         2         0
 17.        1        -1         1         0
 18.        3         0         1         1
 19.        2         1         2         0
 20.        4         2         2         0
```

This is not a command one will likely use everyday, but in many circumstances `fill()` should make the generation of repeating sequences and patterns of numbers quite simple. In many cases, the `fill()` extension can be used in place of a `generate` command and complex expression using `_n` or `_N`.

Reference

Cox, N. J. 1997. dm44: Sequences of integers. *Stata Technical Bulletin* 37: 2–4.

dm56 | A labels editor for Windows and Macintosh

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

[*Editor's note: If you are running Windows* NT *or Macintosh, you may need to download the latest Stata executable from the Stata web site prior to running this command.*]

Stata provides three types of labels: Variable labels and the dataset label (each can be up to 31 characters long), and value labels (up to 8 characters long) mapped to nonnegative integers that can be assigned to one or more variables. This insert presents a visual tool, `labedit` for manipulating all three types of labels. As does its sibling, `notedit`, (Gleason 1998), `labedit` uses the dialog programming features new in Version 5.0 (for Windows and Macintosh only) and so is restricted to such platforms.

`labedit` [*varlist*] [`, novalue`]

is the syntax although it is easy to add a new option to Stata's top-level menu so that `labedit` can be launched with a mouse click or a keystroke shortcut; see Remark 7, below. If a *varlist* is present, only the variable labels and value label assignments for those variables can be edited; the `novalue` option is explained in Remark 1, below.

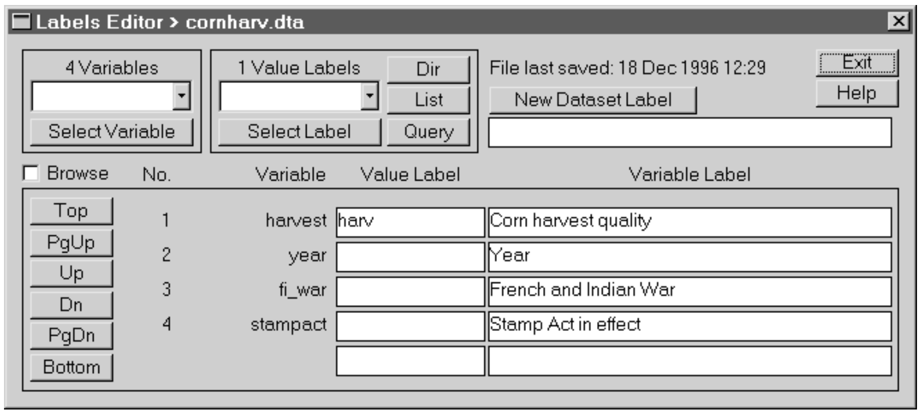
`labedit` operates in one of two modes: variables mode (shown in Figure 1) in which variable labels and value label assignments can be edited, and value label mode (shown in Figures 2 and 3) in which value label definitions can be edited. To illustrate, consider the dataset `cornharv.dta`, included with this insert:

```
. use cornharv, replace
(Corn Harvests in Eastern Mass.)
. describe
Contains data from cornharv.dta
  obs:            37                          Corn Harvests in Eastern Mass.
 vars:             4                          18 Dec 1996 12:29
 size:           333 (99.4% of memory free)   * _dta has notes
-------------------------------------------------------------------------------
   1. harvest     byte   %8.0g       harv       Corn harvest quality
   2. year        int    %9.0g                  Year
   3. fi_war      byte   %8.0g                  French and Indian War
   4. stampact    byte   %8.0g                  Stamp Act in effect
-------------------------------------------------------------------------------
Sorted by:  year
```

Each of the four variables has a variable label, and the variable `harvest` has been assigned a value label named `harv`, defined thus:

```
. lab list
harv:
           1 -Poor
           2 Poor
           3 +Poor
           4 -Ave
           5 Ave
           6 +Ave
           7 -Good
           8 Good
           9 +Good
```

Launching `labedit` on this dataset produces the dialog shown in Figure 1.



File last saved: 18 Dec 1996 12:29

No.	Variable	Value Label	Variable Label
1	harvest	harv	Corn harvest quality
2	year		Year
3	fi_war		French and Indian War
4	stampact		Stamp Act in effect

Figure 1. `labedit` in variables mode with `cornharv.dta`.

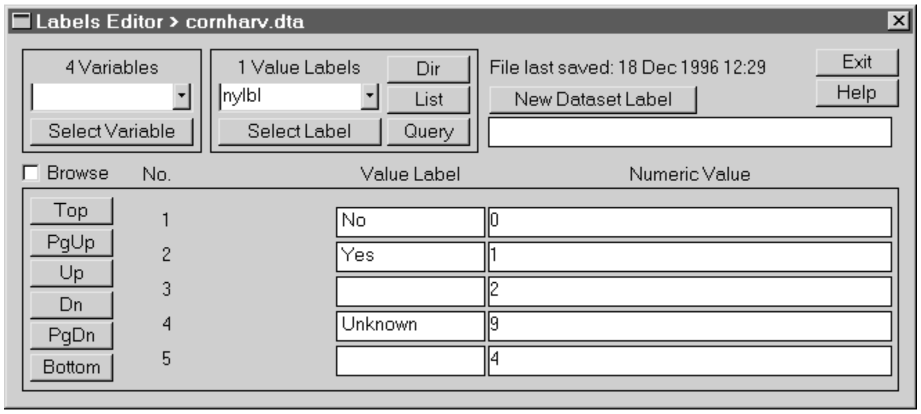
`labedit` begins in variables mode, with the edit window in the bottom portion of the dialog displaying the first five variable names and, beside them, the associated variable labels and assigned value labels. Typing a name into the edit box in the *Value Label* column assigns that value label to the associated variable; clearing the edit box removes an existing value label assignment. Similarly, the text in the edit box in the *Variable Label* column will be assigned as the variable's label; clearing that edit box removes an existing variable label. (Note that the edit boxes in the left and right stacks of the edit window will accept no more than 8 or 31 characters, respectively. Observe also that there will often be a leading blank character in those boxes, which must be removed to achieve the maximum length of 8 or 31 characters.)

But there is no **Save** button: The edit window autosaves its contents whenever any of the six scroll buttons (at the left edge of the window) is clicked, even if no actual scrolling occurs. In the situation of Figure 1, there are only four variables and hence the scroll buttons cannot move a new set of variables into the edit window; nevertheless, clicking any of the scroll buttons would update the variable labels and value label assignments for each of the four variables present in the window. If more than five variables are available so that actual scrolling is possible, the six scroll buttons traverse the variable list in a sensible manner. Clicking **Dn**, for example, shifts downward one position in the variable list, that is, moves the variables one position upward in the edit window, bringing a new variable into view and moving one variable out of the window. Clicking **PgDn** shifts five positions downward in the variable list, and clicking **Bottom** shifts so that the last variable appears in the bottom position of the edit window.

Further, any given variable can be located by typing its name in the edit box above the **Select Variable** button. Alternatively, clicking the downarrow at the right edge of the edit box presents a drop-down list of variable names, any one of which can be placed in the edit box by clicking. In either case, clicking the **Select Variable** button then brings the desired variable and its labels into view—in the top position of the edit window, if possible.

Clicking the **Select Label** button switches `labedit` into value label mode. In this mode, the right stack of edit boxes presents nonnegative integers and the left stack displays the associated value labels (if any). If a *Value Label* box is empty or contains the same integer that appears to its right, there is no label currently defined for that integer value. A new value label scheme can be defined by assigning it a name in the box above the **Select Label** button, and typing labels in the *Value Label* boxes next to the appropriate integers. The **Dn** and **PgDn** buttons can be used to bring a new set of consecutive integers into the *Numeric Value* boxes. However, any set of nonnegative integers can be typed into the *Numeric Value* boxes, in any order, and labels assigned to them in the *Value Label* boxes. Thus, it is not necessary to scroll downward to find a desired integer in order to assign it a label. And, just as in variables mode, the edit window autosaves its contents: clicking any scroll button updates the selected value label with the current contents of the two stacks of edit boxes.

To illustrate, consider the variables `fi_war` and `stampact` in `cornharv.dta`. These are indicator variables for which a value label that assigns `No` and `Yes` to the numeric values `0` and `1` might be useful. That can be done by typing the desired labels in the first two *Value Label* boxes, as shown in Figure 2.



File last saved: 18 Dec 1996 12:29

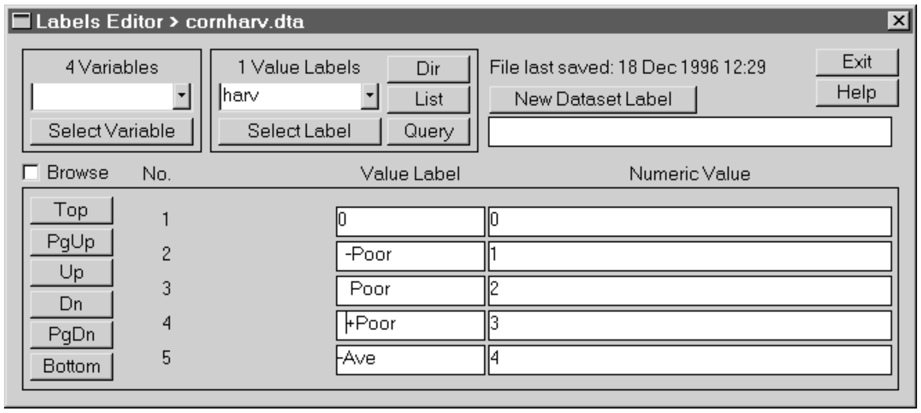
Value label: nylbl

No.	Value Label	Numeric Value
1	No	0
2	Yes	1
3		2
4	Unknown	9
5		4

Figure 2. Creating a new value label (`nylbl`) in value label mode.

But imagine that we also want to use the numeric value `9` to represent an uncertain or unknown status for an indicator variable. In any unused row of the edit window, type `Unknown` in the left edit box, and `9` in the right edit box. Then, type a name for this value label (say, `nylbl`) into the edit box above the **Select Label** button; Figure 2 shows the resulting dialog. Clicking any of the scroll buttons now creates a new value label named `nylbl` exactly as though the command `label define` had been used.

To edit existing value labels, type the label name into the edit box above the **Select Label** button and then click that button. Alternatively, click the downarrow at the right edge of the edit box and choose a value label name from the drop-down list that appears. Figure 3 results from selecting the value label `harv` in `cornharv.dta`. New labels can be added to the definition by typing the desired mapping(s) into any available pair(s) of edit boxes. Existing labels can be edited, or removed by clearing the relevant edit box. (Note that an existing value label will be dropped as soon as the last of its defined labels is cleared; this occurs automatically, and without notification.) To illustrate, recall that the *Value Label* edit boxes will accept no more than 8 characters. This makes it easy to create right-justified labels, if desired: Place the insertion caret at the left edge of the edit box, and press the space bar until the text will move no further. In Figure 3, the labels assigned to numeric values `1`, `2`, and `3` have been edited in this manner; clicking any scroll button would then update the value label `harv` accordingly.



File last saved: 18 Dec 1996 12:29

Value label: harv

No.	Value Label	Numeric Value
1	0	0
2	-Poor	1
3	Poor	2
4	+Poor	3
5	-Ave	4

Figure 3. Modifying the value label `harv` in `cornharv.dta`.

The remaining three buttons in the value label panel provide information about currently defined labels. Clicking **Dir** displays a directory of current label names on the *Results* screen, exactly as though the command `lab dir` had been given. Clicking the **List** button displays on the *Results* screen the definition of the selected value label, or of all defined value labels if there is no selected label—just as if the command `lab list` had been given. For example, clicking the **List** button in Figure 3 would (assuming that a scroll button has been clicked) display the following:

```
. harv:
                 1   -Poor
                 2    Poor
                 3   +Poor
                 4 -Ave
                 5 Ave
                 6 +Ave
                 7 -Good
                 8 Good
                 9 +Good
```

Finally, the **Query** button looks up labels assigned by the selected value label: type any set of nonnegative integers into the *Numeric Value* edit boxes, click on **Query**, and the labels assigned to those integers appear in the *Value Label* edit boxes. If the *Value Label* box matches the *Numeric Value* box, no label is currently assigned to that integer.

Remarks

1. By default, a drop-down list of defined value label names is created when `labedit` is launched. Doing so requires use of the `preserve` and `restore` commands, which can be relatively slow with very large datasets. The option `nolabel` bypasses construction of that list, which can make startup appreciably faster. Existing value labels can still be edited and new ones created; the `nolabel` option merely leads to an empty drop-down list of value label names.
2. Whether or not the drop-down list is created, it cannot be updated during an editing session. So, if a new value label is created or an existing value label is dropped, the drop-down list will not reflect that fact. Instead, use the **Dir** button to view the list of value labels that are defined at any point.
3. Value labels cannot be assigned to string variables; in variables mode, the *Value Label* box for a string variable contains (as a reminder) the text `<String>`.
4. The left-quote (‘`` ` ``’) and double-quote (‘`"`’) characters must be avoided. A ‘`` ` ``’ character forces some part of the text to be interpreted as a macro expansion, which typically causes that part to vanish. Instances of ‘`"`’ will usually have the same effect, but can produce other kinds of errors as well.
5. Clicking the *Browse* checkbox (just above the scroll buttons) disables the autosave action of the scroll buttons in both variables mode and value labels mode. With this setting, labels can be examined without fear of altering their current settings. But note that when *Browse* is set, any changes made in the two stacks of edit boxes are immediately lost as soon as a scroll button is clicked.
6. Beside the **Query** button there is an edit box that can be used to assign a new dataset label, by clicking the button just above it. But, as in Figures 1–3, an existing label will not appear in that box when `labedit` is launched because there is (at present) no simple way to query the dataset label.
7. Rather than typing `labedit` on the command line, some users may prefer to launch `labedit` by clicking on a menu item or pressing a keystroke combination. The included file `labmenu.do` makes the necessary arrangements. Typing `do labmenu` on the command line adds an item named **Labels** to Stata's top-level menu. Afterward, one mouse click launches `labedit`; Windows users can also use the *Alt-L* keystroke shortcut. To make this arrangement 'permanent', add the line '`do labmenu`' to the file `profile.do`; alternatively, copy the contents of `labmenu.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.

Reference

Gleason, J. R. 1998. dm57: A notes editor for Windows and Macintosh. *Stata Technical Bulletin* 43: 6–9.

dm57 | A notes editor for Windows and Macintosh

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

[*Editor's note: If you are running Windows* NT *or Macintosh, you may need to download the latest Stata executable from the Stata web site prior to running this command.*]

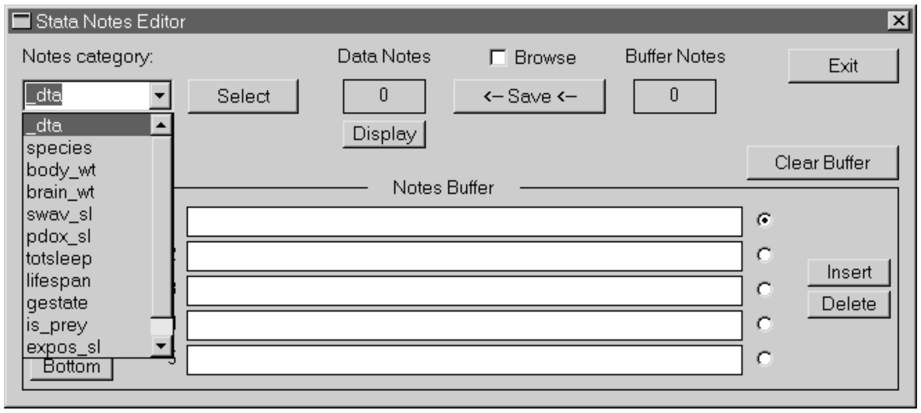
Since Version 4.0, Stata's `notes` command has provided a way to embed commentary in a dataset; see [R] **notes**. Notes can be used to document the source of the data, to explain details of data collection and variable definitions, or to record remarks during data analysis. But the `notes` command offers only the most rudimentary abilities: Notes can be created and associated with any existing variable or with the dataset (`_dta` notes), and existing notes can be displayed or dropped. But there is no way to edit the content of a note, to insert a new note between two old ones, or to change the ordering of notes, except to create new notes and drop old ones. Finally, `notes` does not renumber after dropping notes so that the note numbers quickly become fragmented.

This insert presents `notedit`, a visual editor for manipulating notes that improves upon the abilities of the `notes` command. `notedit` uses the dialog programming features new in Version 5.0 (for Windows and Macintosh only) and so is restricted to those platforms. The syntax is

```
notedit
```

which presents a dialog resembling those shown in Figures 1–3. (However, many users will find it convenient to add a new option to Stata's top-level menu so that `notedit` can be launched with a mouse click or a keystroke shortcut. See Remark 4, below.)

A typical use of `notedit` might begin by selecting a notes category, i.e., either `_dta` or the name of an existing variable. That is done by typing the category name into the edit field beneath the label *Notes Category*. Alternatively, click on the desired item in the drop-down list that appears when the down-arrow beside the edit box is clicked; Figure 1 shows this list for the data `mammals.dta` (included with this insert).



Notes category: _dta

Data Notes: 0

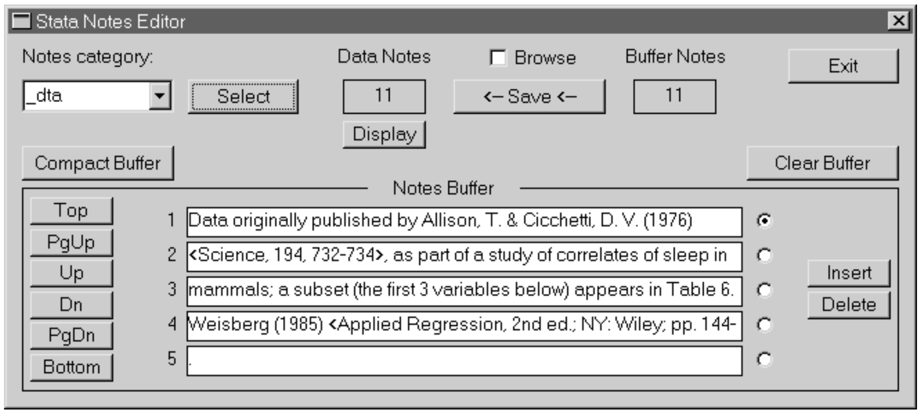
Buffer Notes: 0

_dta
species
body_wt
brain_wt
swav_sl
pdox_sl
totsleep
lifespan
gestate
is_prey
expos_sl

Figure 1. Selecting a notes category from the drop-down list.

Clicking the **Select** button then selects the category whose name appears in the edit box. If there are any notes associated with that category, they will be copied to a series of global macros—a buffer. The box marked *Data Notes* will display the number of notes currently in the dataset; the box marked *Buffer Notes* will show the number of the last non-blank note currently held in the buffer. The bottom portion of the dialog presents a scrollable window of depth five into the notes buffer; scrolling is controlled by a set of six navigation buttons at the left edge of the window.

Figure 2 shows the result of clicking **Select** in the situation depicted in Figure 1.



Notes category: _dta

Data Notes: 11

Buffer Notes: 11

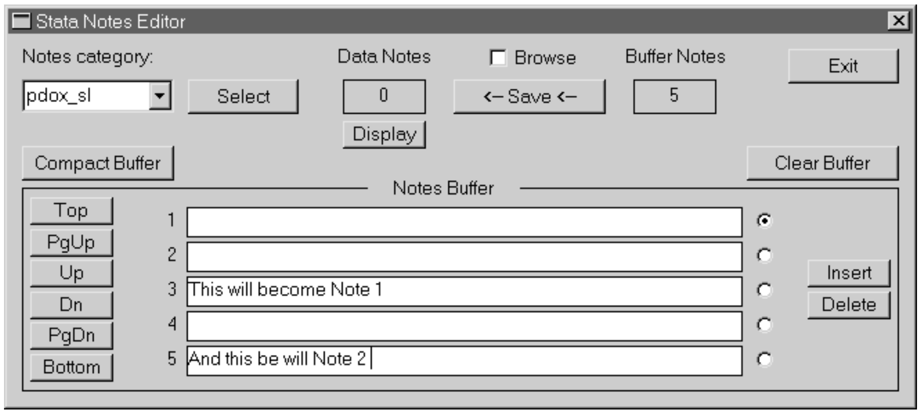
1 Data originally published by Allison, T. & Cicchetti, D. V. (1976)
2 <Science, 194, 732-734>, as part of a study of correlates of sleep in
3 mammals; a subset (the first 3 variables below) appears in Table 6.
4 Weisberg (1985) <Applied Regression, 2nd ed.; NY: Wiley; pp. 144-
5 .

Figure 2. The `notedit` dialog after selecting `_dta` notes in `mammals.dta`.

The first five of the 11 `_dta` notes are visible in the buffer window; the remaining six notes can be brought into view using the **Dn**, **PgDn**, or **Bottom** buttons. Each of the five slots in the buffer window is a functional edit box: Its contents can be replaced, deleted, overwritten, or cut, copied, and pasted to and from the clipboard—all in the usual manner. For example, on Windows 95, selecting text and right-clicking brings up a context menu that provides options for the cut, copy, and paste operations, among others. At the right of the buffer window are two buttons that operate on a single buffer slot. The **Insert** button creates a new, blank note, pushing existing notes downward in the buffer. The **Delete** button deletes a note and pulls existing notes upward to fill the gap. The bank of radio buttons controls the buffer position where the Insert or Delete action takes place.

Alternatively, one can simply begin with an empty buffer (clicking the **Clear Buffer** button arranges for that) and start typing notes into the buffer window slots. The category to which the notes will be attached can be selected later. There is no limit on the number of notes one can create, and each note can be as long as Stata's macros permit. (See [R] **limits**.) The contents of the notes buffer can be scrolled and altered at will; they do not become part of the dataset in memory until the **Save** button is clicked. Doing so copies the contents of the notes buffer to the dataset, replacing any previous notes associated with the selected category. For example, clicking **Clear Buffer** and then **Save** is an easy way to erase all notes for the selected category. Checking the **Browse** box disables the **Save** button, thereby turning `notedit` into a notes viewer.

In short, `notedit` tries to mimic the working style of a simple text editor; there are, however, some unavoidable differences. For one, the buttons at the left of the buffer window are required to navigate the notes buffer; you can't simply keep typing notes and pressing *Enter*, as you might in a text editor, and there is no vertical scroll bar. Typing a note into an empty slot does not immediately increment the displayed count of buffer notes, as it might in an editor. And an 'empty' slot in the buffer window in fact holds a single space character " "; slots `1`, `2`, and `4` in Figure 3 are examples.



Notes category: pdox_sl

Data Notes: 0

[ ] Browse

Buffer Notes: 5

1 [✓ ●]

2

3 This will become Note 1

4

5 And this be will Note 2

Figure 3. Clicking **Save** copies buffer notes 3 and 5 to the current data as `pdox_sl` notes 1 and 2.

So, typing `Here is a note` into such a slot typically forms the string `Here is a note`. Clicking **Insert**, **Delete**, **Compact Buffer**, **Save**, or any of the six navigation buttons forces an update of the buffer window, during which leading blank characters are discarded and the number of buffer notes is refreshed, even if no scrolling occurs. (Note that, as in Figure 3, the buffer note count may include blank notes.) Finally, just as with `notes`, buffer notes are purged of excess blanks when they are saved.

Then there are the matters of note numbering and entirely blank notes. Notes created and later modified by the `notes` command will generally not have consecutive numbers. For example, notes in a category will be numbered `1`, `3`, and `6` if six notes are saved in one session, and the second, fourth, and fifth of those notes are subsequently dropped by `notes`. But `notedit` automatically renumbers notes as they are loaded by the **Select** button; in the last example, the three notes will appear in buffer window slots `1`, `2`, and `3`, not in slots `1`, `3`, and `6`. On the other hand, the `notes` command does not create blank notes. So, while the `notedit` buffer can contain any number of blank notes, they will be purged during a Save operation. For example, clicking the **Save** button in the situation of Figure 3 will associate just two notes with the variable `pdox_sl`. That is, clicking **Save** first updates and compacts the notes buffer, and then copies the buffer to the current data, overwriting any previous notes in the selected category. Clicking the **Compact Buffer** button performs just the first step so that the buffer contains only the notes that will be saved, numbered consecutively.

Remarks

1. As just observed, `notedit` does not save blank notes. To separate groups of notes, create a note consisting of a single non-blank character such as '`.`' or '`_`', as in Note 5 of Figure 2.
2. As with `notes` the string `TS` will, if separated by blanks from other text, be translated to a date/time stamp. The translation occurs when the **Save** button is clicked. To see the result, reload the saved notes with the **Select** button, or use the **Display** button.

3. The **Display** button displays the current data notes (*not* the buffer notes) on the *Results* screen, exactly as though the command `notes` *cat* had been issued, where *cat* is the name of the selected category.

4. Rather than typing `notedit` on the command line, some users may prefer to launch `notedit` by clicking on a menu item or pressing a keystroke combination. The included file `notemenu.do` makes the necessary arrangements. Typing `do notemenu` on the command line adds an item named **Notes** to Stata's top-level menu. Afterward, one mouse click launches `notedit`; Windows users can also use the *Alt-N* keystroke shortcut. To make this arrangement permanent, add the line `do notemenu` to the file `profile.do`; alternatively, copy the contents of `notemenu.do` into `profile.do`. (See [U] **5.7**, **6.7**, or **7.6 Executing commands every time Stata is started**.)

5. The left-quote ('`') and double-quote ('"') characters must be avoided. A '`' character forces some part of the note to be interpreted as a macro expansion, which typically causes that part to vanish. Instances of '"' will produce an error message on the *Results* screen.

Acknowledgment

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

dm58 A package for the analysis of husband–wife data

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

This insert describes the package `hh`, a collection of programs that facilitate the analysis of 2-level data on individuals nested within households in long format (i.e., the data of husbands and wives are provided in separate records), without going through the slow process of `reshape`-ing the data between long and wide formats. (Note also that `reshape` involves some loss of information, e.g., notes associated with variables.) While `hh` was written for household data, it can be used to facilitate the analysis of data on other, similar asymmetric matches such as buyer supplier relationships.

The main features of the `hh` programs are explained in a sample session in which we use the dataset HIN95 ("Households in the Netherlands 1995") on the "internal organization of households". The dataset contains data on 1,533 couples (1,523 man–woman, 10 man–man/woman–woman) and 288 singles. The data on couples comprises information on individual characteristics of the spouses (e.g., individual labor market career, attitudes and opinions towards the relationship with their partner, etc.), as well as "couple data" (e.g., characteristics of their children, their joint housing history, division of labor, financial practices, etc.). For singles, the data comprises about the same individual characteristics as those for couples, but not the couple variables; additional variables contain information unique to singles.

An appropriate way to represent the data would probably use some kind of relational data base. Stata, however, only knows a rectangular data matrix as a data structure. Thus, we are forced to represent the data either as information on households, with individual information on spouses stored in different variables, or as information on individuals, with the household-level characteristics duplicated with both spouses. In Stata's `reshape` terminology, the first representation is the "wide format" and the second is the "long format". Both representations have their advantages and disadvantages. For instance, in the wide format, all data manipulation on individual-level characteristics has to be performed twice for the two partners. This is laborious and error prone (How can I be sure that the code for husbands and wives is consistent?). Moreover, the number of variables would be prohibitive in wide format (approximately 3,000 variables with HIN95). In the long format, however, memory and disk space is wasted by duplicating household-level information in the records of spouses. Comparison or aggregation of individual characteristics is often somewhat difficult—though `egen` provides many useful subfunctions. Moreover, `if/in` expressions may become hazardous since they select individuals rather than households. Finally, in most of our analyses, the unit of analysis is the household, not an individual, but individual characteristics (e.g., the wage rates of the partners) frequently enter the analysis.

The `hh` package is a modest and application-specific contribution to the reduction of the problems of the analysis of multi-level data using "traditional" nonrelational data management programs. It employs the long format. First we have to declare data to be household data by specifying a household identification variable and a sex variable (`hrespnr` and `rs002` for HIN95).

```
. hhset hrespnr rs002

288 individuals are partner-less
20 individuals in 10 homosexual couples
3046 individuals in 1523 heterosexual couples

household identification: hrespnr
sex identification:       rs002
selected case identifier  HH_idrop
```

Note that `hhset` deduces the number of singles (unique records for the household identifier) and gay/lesbian couples from the data declaration variables. These cases will be excluded "automatically" from later analyses. With `hhcorr`, we can now obtain the correlation between individual-level characteristics, say the frequencies of church-visiting of husband and wife. Note that **hhcorr** also lists summary individual information on husbands and wives.

```
. hhcorr rs005
                   Husband-Wife   ------ Husband -----   ------- Wife -------
Variable | #-of-HH  Correlation       Mean    Std.Dev        Mean    Std.Dev
---------+----------------------------------------------------------------------
rs005    |   1487       0.7986     1.778515   .9996761    1.890667   1.022779
```

Like many behavioral and attitudinal characteristics of couples, the intra-household correlation in religious behavior is quite high—approximately .80. We can also tabulate church-membership of husband and wife. All 2-way options of **tabulate** are supported by **hhtab**.

```
. hhtab  rs004, chi2
Husband x Wife tabulation of rs004  (denomination)
           | _Wife
  _Husband |  no relig   roman ca   dutch re   rereform      other |     Total
-----------+-------------------------------------------------------+----------
  no relig |       543         81         57         19          6 |       706
  roman ca |        58        310         12          3          0 |       383
  dutch re |        43         15        116          6          0 |       180
  rereform |        13          9          7        109          5 |       143
     other |         7          3          3          4         36 |        53
-----------+-------------------------------------------------------+----------
     Total |       664        418        195        141         47 |      1465
         Pearson chi2(16) = 2668.1872   Pr = 0.000
```

Indeed, spouses tend to have the same religion. In fact, **hhtab** recognizes individual-level and household-level variables.

```
. hhtab ha02 rs007
Note: ha02 does not vary within households
# of        |
additional  |
persons in  |
HH          |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        599       39.96       39.96
          1 |        300       20.01       59.97
          2 |        414       27.62       87.59
          3 |        152       10.14       97.73
          4 |         28        1.87       99.60
          5 |          4        0.27       99.87
          6 |          1        0.07       99.93
          8 |          1        0.07      100.00
------------+-----------------------------------
      Total |       1499      100.00
Husband x Wife tabulation of rs007  (likelihood voting if elections)
           | _Wife
  _Husband |  cert not   prob not   prob yes   cert yes |     Total
-----------+--------------------------------------------+----------
  cert not |        39         20         16         20 |        95
  prob not |        17         50         31         33 |       131
  prob yes |        17         51        126        115 |       309
  cert yes |        22         43        213        680 |       958
-----------+--------------------------------------------+----------
     Total |        95        164        386        848 |      1493
```

When using the hh package, it is still possible to correlate individual-level variables, such as the 2-way cross-tabulation of the church-visiting frequency of a person and her/his mother.

```
. tab rs005 rs022
church     | church visit mother
visit      |  never      1/year     1/month    1/week     parent d |     Total
-----------+-------------------------------------------------------+----------
  never    |       932        259         81        339          7 |      1618
  1/year   |       127        254        119        533         10 |      1043
  1/month  |        14         28         24        165          2 |       233
  1/week   |        29         24         14        311          5 |       383
-----------+-------------------------------------------------------+----------
     Total |      1102        565        238       1348         24 |      3277
```

A special type of `tabulate` is produced by `hhcount`.

```
. hhcount rs004
Husband x Wife tabulation of rs004
           | _Wife
  _Husband |        no        yes |     Total
-----------+----------------------+----------
        no |       992        122 |      1114
       yes |       111        298 |       409
-----------+----------------------+----------
     Total |      1103        420 |      1523
```

The programs in the `hh` package permit `if/in` restrictions of (husband–wife) statistics to subsamples of households.

```
. hhcorr rs005 if rs005>1
-if- and -in- should select both partners or none
r(198);
```

Note that Stata's `if/in` operates at the individual level, whereas `hhtab` operates at the level of households. `hhtab` recognized that the selected subsample violated household boundaries: In some households, only one of the partners was selected. What could, then, the intra-household correlation mean? (If the selected subsample had consisted of complete households, `hhcorr` would have proceeded to compute statistics.)

In the `hh` programs, I implemented a simple mechanism to deal with selection of "clustered" observations via the selection of individuals. This method adds two switches (options): `any` and `all`. `any` specifies that all households are selected in which at least one of the members was selected by the `if/in` clauses.

```
. hhcorr rs005 if rs005>1, any
                   Husband-Wife   ------ Husband -----   ------- Wife -------
Variable | #-of-HH  Correlation       Mean    Std.Dev        Mean    Std.Dev
---------+--------------------------------------------------------------------
rs005    |    906       0.6457     2.266451   1.003873    2.453754   .9426846
```

Similarly, `all` specifies that only those households are selected in which all members are `if/in` selected.

```
. hhcorr rs005 if rs005>1, all
                   Husband-Wife   ------ Husband -----   ------- Wife -------
Variable | #-of-HH  Correlation       Mean    Std.Dev        Mean    Std.Dev
---------+--------------------------------------------------------------------
rs005    |    634       0.7986     2.678683   .8762791    2.758621   .8724166
```

Unsurprisingly, the number of selected households via the `all` option is always smaller (at least not larger) than via the `any` option.

Clearly, the commands `hhtab`, `hhcorr`, and `hhcount` cover only a small part of the statistical features of Stata. To use more general commands, we added a pre-command `hhi`. We start with an example. We want to compute the Spearman correlation between `rs005` (church-visiting frequency) of husbands and wives.

```
. hhi: spearman h.rs005 w.rs005
Hs005    <- Husband           of  rs005    church visit
Ws005    <- Wife              of  rs005    church visit

 Number of obs =    1487
Spearman's rho =       0.7366

Test of Ho: Hs005 and Ws005 independent
      Pr > |t| =       0.0000
```

The prefix functions `h.` and `w.` select the husband's and wife's values of the variable `rs005`. Other prefix functions are defined below. The `hhi` command could also have been used to obtain Pearson's moment correlation, rather than via `hhcorr`.

```
. hhi: corr h.rs005 w.rs005
Hs005    <- Husband           of  rs005    church visit
Ws005    <- Wife              of  rs005    church visit
(obs=1487)

        |    Hs005    Ws005
--------+------------------
   Hs005|   1.0000
   Ws005|   0.7986   1.0000
```

Both hhcorr and hhi give the same answer (.7986)—as they should. Note though that hhcorr allows variable lists and displays individual-level summary statistics. The hhi pre-command can be used for more advanced uses as well. For instance, we can easily analyze whether education (rs003) and age affect whether a couple owns rather than rents their apartment.

```
. replace hu04_a = hu04_a - 1            /* logistic requires 0/1 coding */
(3354 real changes made)

. gen age = 96-rybirth

. hhi: logistic hu04_a h.rs003 w.rs003 m.age a.age
Hs003    <- Husband            of  rs003    highest educ completed on quest
Ws003    <- Wife               of  rs003    highest educ completed on quest
Mge      <- Mean(Husband,Wife) of  age
Age      <- Abs(Husband-Wife)  of  age

Logit Estimates                                        Number of obs =   1501
                                                       chi2(4)       =  40.08
                                                       Prob > chi2   = 0.0000
Log Likelihood = -915.29162                            Pseudo R2     = 0.0214

------------------------------------------------------------------------------
  hu04_a | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   Hs003 |   .9060867   .0250159     -3.572   0.000       .8583593    .9564678
   Ws003 |   .9281081     .02866     -2.416   0.016       .8736017    .9860154
     Mge |   .9865684   .0055428     -2.407   0.016       .9757643    .9974921
     Age |   1.051562   .0204116      2.590   0.010       1.012307    1.092338
------------------------------------------------------------------------------
```

Note that hhi verifies that hu04_a indeed does not vary within households.

Detailed descriptions

The hh package comprises the following programs.

```
hhset     hrespnr sex [weight] [, partner(varname) verify]
hhtab     [varlist] [if exp] [in range] [, any all tabulate_options]
hhcount   [=]exp [if exp] [in range] [, any all tabulate_options]
hhcorr    [varlist] [if exp] [in range] [, any all label]
hhi       [if exp] [in range] [, any all add nolabel preserv keep] : any_cmd
hh_is
hh_slct   [if exp] [in range], gen(varname) [any all]
```

Description

hhset specifies the household identification variable and a variable specifying the sex of the respondents. Neither variable should contain missing values. hh expects the coding (1=Husband, 2=Wife). hhset generates an "internal variable" HH_idrop that marks the records that do not belong to a heterosexual couple. This information is stored in the data using characteristics associated with _dta named HH_name, and thus has to be supplied only once.

If the data on a household contains records on other people besides the "main couple" (typically, children, parents at old age, etc.) a partner variable should be specified that identifies the "main" couple by value 1 and all other household members by value 0. The hh package requires that a household contains 0 or 2 partners.

hhset typed without arguments displays the names of the key variables.

hhtab displays a husband-by-wife tabulation of the variables in *varlist*. Any tabulate option (e.g., cell nofreq) may be specified. A one-variable tab is displayed for variables that do not vary within households.

hhcount displays whether an expression is true for husband and wife as a 2×2 table if husbands and wives are different, otherwise, a 2×1 table is shown. In contrast with count, the expression should not be preceded with an if keyword, and the leading = is optional. Rather, the if and in clauses may be used to select a subset of households.

hhcorr lists the husband–wife (intra-household) correlations and marginal summary statistics for husbands and wives for the variables in varlist.

`hhi` facilitates mixed-level statistical analyses in which individual and household characteristics are combined. `hhi` is a pre-command like `xi:` or `by:` that operates on any Stata command with the regular syntax

`command` *varlist* [`if` *exp*] [`in` *range*] [*weight*] [, *options*]

Using `hhi`, it is possible to include in the *varlist* prefixed variables that define variables from individual-level data:

`h.`*varname*	husband's value of *varname*
`w.`*varname*	wife's value of *varname*
`m.`*varname*	mean of *varname* for husband and wife
`d.`*varname*	husband–wife difference for *varname*
`a.`*varname*	absolute difference between husband and wife in *varname*

If a variable is specified without a prefix, `hhi` verifies that it is indeed constant within all households. `hhi` generates new variable names by prepending or replacing the first character with the function prefix. `hhi` may yield incorrect results if the data contain variables with embedded periods in the variable names. You are strongly advised to provide `if/in` restrictions of the sample via `hhi`, and not via `any_cmd`. If your command includes weights, be aware that weights should be constant within households.

Utilities for hh-programmers

`hh_is` verifies that the dataset in memory is indeed in "HH (household) format." Otherwise it displays an error message.

`hh_slct` performs couple/household selection from the individual selection via `if/in`. The selected observations are marked by a variable specified via the required option `gen`.

General options

`any` specifies that households should be included in which at least one of the partners is selected by the `if` and/or `in` clauses.

`all` specifies that households should be included in which both partners are selected by the `if` and/or `in` clauses.

Options of hhset

`verify` specifies that the user is requested to verify the appropriateness of the value labels of the sex identifier.

Options of hhcorr

`label` specifies that variable labels are included in the display table instead of the means and standard deviations of the variables for husbands and wives.

Options of hhi

`add` specifies that the names of the newly generated variables are formed by prefixing the function-prefix to the name of the variable without dropping the first character of the variable name.

`nolabel` specifies that a table describing the variable names is displayed.

`preserv` specifies that the time-consuming process of saving the data in memory to disk is suppressed.

`keep` specifies that the constructed variables should be kept in the data. This allows the use of the standard Stata post-estimation commands. Note that you have to drop the constructed variables manually using the command `drop $HHI_VARS`, where `HHI_VARS` is a global macro that contains the variables constructed by `hhi`.

Remarks

The `hh` package was designed to facilitate working with the household data "Households in the Netherlands 1995" (HIN95), a survey held among approximately 1,500 couples and approximately 300 singles as part of the PIONIER program "The Management of Matches" that is supported by the Netherlands Organization for Scientific Research via research grant PGS 50-370 to W. Raub and J. Weesie.

ip25 | Parameterized Monte Carlo simulations: An enhancement to the simulation command

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

Stata contains a useful command `simul` that greatly simplifies Monte Carlo type simulations. In such simulations we usually seek to analyze the properties of stochastic (e.g., statistical) models that are too hard or too time-consuming to understand analytically. In my own applications of Monte Carlo simulations, I typically seek to derive—and possibly understand—how some "output" parameter of a stochastic model depends on some parameters of the model. For instance, how well is (a tail-probability of) the distribution of some statistic approximated by a limiting distribution? And, as a more detailed problem, how does the quality of the approximation depend on parameters of the model? Finding an answer to the latter type of question requires "parameterized simulations" in which a simulation is run repeatedly for a series of parameter values. This cannot easily be accomplished with `simul`.

I wrote `simul2`, an extension of the standard Stata command `simul` for Monte Carlo simulation, that adds the possibility to simulate the effects of varying one or more numeric parameters (inputs) on the results (outputs). `simul2` is fully backward compatible with `simul`. In the description below, I assume the reader has already read the section on `simul` in the Stata Reference Manual, so I will focus only on the extension for parameterized simulations.

Currently, `simul2` only supports "full-factorial parameterized simulations" (FFPS) in which *all* combinations of the specified levels of a series of parameters are simulated. FFPS is clearly only suitable for moderate numbers of parameters and moderate number of levels for each parameter: The number of conditions to be simulated equals the *product* of the number of levels of the parameters, and this rapidly "explodes"! If you need to simulate the effects of many parameters and/or your parameters take relatively many levels, FFPS is clearly not appropriate, and you have to turn to some cleverly constructed subsets of conditions using the theory of experimental designs, and you may have to turn to specialized simulation tools.

Syntax

`simul2` *progname*`,` `reps(`*#*`)` [`args(`*whatever*`)` `dots` `par(`*plist*`)` `report`]

where *progname* is the name of the program that performs a single simulation.

Options

`reps(`*#*`)` is not optional—it specifies the number of replications to be performed.

`args(`*whatever*`)` specifies any arguments to be passed to *progname* on invocation. The query call is then of the form "*progname* `?` *whatever*" and subsequent call of the form "*progname postname whatever*".

`dots` requests a dot be placed on the screen at the beginning of every call to *progname*, thus providing entertainment during long simulations.

`par(`*plist*`)` specifies a full-factorial parameterized simulation as one or more parameter specifications separated by commas. A parameter specification consists of a parameter name followed by a *numlist* object such as 1-5 or 0-1/0.10 (see `help` for `numlist` for details). The parameter names have to be included at the beginning of the list of variable names set in `S_1` when *progname* is called at the outset. Also, the parameter names have to be included, again at the beginning, in the list of results saved using `post`.

`report` specifies that a brief report is displayed with the interpretation of the parameters, including the total number of conditions to be simulated.

Examples

In the section on `simul` in the Stata Reference Manual, the simulation of a sample of 100 observations from a standard log-normal distribution is described. Using `simul2` it is easy to perform this simulation varying both the location and the scale parameters of the log-normal distribution. First, we have to modify the coding of the program `lnsim`. This requires

- adding the parameters loc and scale to the list of variable names to be saved.
- modifying the line that generates data from a log-normal distribution.
- including `loc` and `scale` on the line that posts the results.

These modifications yield the following program `lnsim`:

```
program define lnsim
        version 5.0
        if "`1'" == "?"
                global S_1 "loc scale mean var"
                exit

        drop _all
        set obs 100
        gen z = exp(loc + scale*invnorm(uniform()))
        summarize z
        post `1' loc scale _result(3) _result(4)
end
```

We can then run the parameterized simulation by specifying in a *numlist* object the levels of the parameters loc and scale. For `loc` we choose the equidistant values $-2, -1.5, -1, \ldots, 2$; for scale 1, 2, 3, 4.

```
. simul2 lnsim, reps(1000) dots par(loc -2-2/0.5, scale 1-4) report

Simulation program: lnsim

Parameter  #values  values
loc          9      -2 -1.5 -1 -.5 0 .5 1 1.5 2
scale        4      1 2 3 4

number of conditions:       36
replications per condition: 1000

[ -2,1] ..........  (output omitted)
[ -1.5,1] ..........  (output omitted)
[ -1,1] ..........  (output omitted)
  (output omitted)
[ 1.5,4] ..........  (output omitted)
[ 2,4] ..........  (output omitted)
```

The option `report` has produced the block describing the expanded *numlist*s. Note that the option `dots` now specifies that the conditions are described via the associated parameter values.

The dataset with simulated results now obtain 4 variables: `loc` and `scale` are the parameters (inputs), and `mean` and `var` the simulation results (outputs). Note that the dataset contains 36 x 1000 = 36000 observations.

```
. describe

Contains data
  obs:       36,000
 vars:            4                           31 Jul 1997 12:51
 size:      720,000 (98.2% of memory free)
-------------------------------------------------------------------------------
   1. loc         float  %9.0g
   2. scale       float  %9.0g
   3. mean        float  %9.0g
   4. var         float  %9.0g
-------------------------------------------------------------------------------
Sorted by:
```

Using Patrick Royston's `byvar` (Royston 1995), it is easy to obtain summary statistics over all simulation conditions:

```
. byvar loc scale: summ mean var

-> loc==-2 scale==1

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
    mean |    1000    .2243892   .0298248    .146899   .3920485
     var |    1000    .0863446   .0796194   .0164808   1.378216

-> loc==-2 scale==2

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
    mean |    1000      .98366    .554835   .2841944   6.421041
     var |    1000    30.97271   146.0815   .2207912   2912.247

  (output omitted)

-> loc==2 scale==4

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
    mean |    1000    19132.12   192154.3    135.838    4930243
     var |    1000    3.71e+12   8.45e+13   217416.6   2.43e+15
```

Reference

Royston, P. 1995. ip9: Repeat Stata command by variable(s). *Stata Technical Bulletin* 27: 3–5. Reprinted in *Stata Technical Bulletin Reprints* vol. 5, pp. 67–69.

sbe16.2 Corrections to the meta-analysis command

Stephen Sharp, London School of Hygiene and Tropical Medicine, stephen.sharp@lshtm.ac.uk
Jonathan Sterne, United Medical and Dental Schools, UK, j.sterne@umds.ac.uk

A few minor bugs have been discovered and corrected in this version of the `meta` command.

sg33.1 Enhancements for calculation of adjusted means and adjusted proportions

Joanne M. Garrett, University of North Carolina at Chapel Hill, FAX 919-966-2274, garrettj@med.unc.edu

After fitting a multiple linear regression model or a logistic regression model, we often want to report our results as covariate adjusted means or proportions (probabilities) by different categories of a nominal independent variable. Two programs which do this are `adjmean.ado` (adjusted means from linear models) and `adjprop.ado` (adjusted probabilities from logistic regression models) which were described in STB-24 (Garrett 1995). Now these programs have several enhancements which have come about from suggestions received from other users, as well as additions I use for my own work. These programs have the same names as the originals. Therefore, if you don't want to replace the old ones, you may want to rename them before loading these.

Summary of the changes

1. `level(`*#*`)` option added.
2. Standard errors (of the prediction) are printed.
3. Lowest category defaults to reference group.
4. An F test for the overall difference of the means is printed for a linear model; a likelihood ratio chi-square for the overall difference of proportions is printed for a logistic regression model.
5. Covariates can be set to any value for prediction after estimation of the model.
6. A second nominal X can be specified; includes test for interaction.
7. Two types of graphs are available to display results.

Description

`adjmean` fits a linear regression model (using `regress`) and uses the estimated coefficients to calculate means and confidence intervals for categories of one or two nominal independent variables. `adjprop` fits a logistic regression model (using logistic) and uses these estimated coefficients to calculate proportions (probabilities) and confidence intervals. Both programs adjust the estimates to the means (by default) of any covariates in the model. The means or probabilities and confidence intervals are always shown. Optionally, the model and/or a graph may be printed. Temporary dummy variables are generated automatically for the nominal variable(s), and interaction variables are generated if two X's are specified.

Syntax

The syntax of `adjmean` (for a multiple linear regression model) is

```
adjmean yvar [if exp], by(xvar1 [xvar2]) [adjust(covlist) model
        level(#) graph bar graph_options]
```

where *yvar* is a continuous outcome variable.

The syntax of `adjprop` (for a logistic regression model) is

```
adjprop yvar [if exp], by(xvar1 [xvar2]) [adjust(covlist) model
        level(#) graph bar graph_options]
```

where *yvar* is a binary outcome variable that must be coded 0 or 1.

Options (same for both commands)

`by(`*xvar1* [*xvar2*]`)` specifies one or two nominal variables that define the categories for the estimated means or probabilities. This option is required.

If one X (*xvar1*) is specified, temporary dummy variables are created for use in the model; the lowest category defaults to the reference group (it was the highest category in the earlier versions of the programs). This is consistent with Stata's `xi` command (although it doesn't use `xi`'s naming convention). When only one X is specified, the dummy variables are named `X2`, `X3`, `X4`, etc., where `X2` is the 2nd category of X, `X3` is the 3rd category, etc. An overall test of differences is printed, based on an F test for the linear model, or a likelihood-ratio test for the logistic model.

A second categorical X (*xvar2*) can be specified. Adjusted means (or probabilities) are estimated for all possible combinations of the X's. For example, suppose the outcome is systolic blood pressure and *xvar1* is race with three categories and *xvar2* is gender with two categories:

```
race:  1=white, 2=black, 3=Asian
gender:  1=male, 0=female
```

Mean systolic blood pressures (using `adjmean`) are calculated for six categories: white males, white females, black males, black females, Asian males, and Asian females. Had the outcome been a dichotomous variable for hypertension (1=hypertensive, 0=normal), the probability of being hypertensive (using `adjprop`) would be calculated for each of the six categories.

An overall test of differences is printed, and additionally, a test for interaction. The names for the dummy variables for *xvar1* are `X12`, `X13`, `X14`, etc., where the first number ("1") specifies dummy variables representing *xvar1*, and the second number designates which dummy variable. Dummy variables for *xvar2* are `X22`, `X23`, `X24`, etc. Interaction terms are created by multiplying combinations for the *xvar1* and *xvar2* dummy variables in the model. An F test for a linear model or the likelihood ratio test for a logistic model is printed to test an interaction effect.

`adjust(`*covlist*`)` specifies an optional list of covariates. If this option is not specified, unadjusted estimates are reported. Values for the covariates default to their means. Alternatively, some (or all) covariates can be set to any value for prediction after estimation of the model. For example, the following statement estimates the systolic blood pressure means for each race category adjusted for age, gender, and cholesterol level:

```
. adjmean sbp, by(race) adjust(age gender chl)
```

If we wanted to calculate these means for someone who is 50 years old, male, with a cholesterol level of 250, we would write

```
. adjmean sbp, by(race) adjust(age=50 gender=1 chl=250)
```

`model` displays the output from `regress` or `logistic`. If this option is not requested, the model is not shown.

`level(`*#*`)` specifies the confidence level in percent for the confidence intervals. The default is 95%.

`graph` displays points for each adjusted mean or probability (y-axis) by *xvar1* (x-axis). If only *xvar1* is requested, confidence intervals are shown also. If *xvar2* is specified, the points are separated by categories of *xvar2*, with *xvar1* still defining the x-axis. This sounds more confusing than it is—see examples which follow.

`bar` displays a bar graph rather than point estimates when specified with graph.

graph_options are any of the options allowed with `graph`, `twoway`.

Examples

To illustrate how these programs work, let's take a look at a study of the cost of treating patients with acute low back pain (Carey et al. 1995). All patients ($n = 1406$) had low back pain (LBP) at entry into the study, and were followed for 6 months.

```
Contains data from LBP.DTA
 obs:        1,406                    Low Back Pain: Outcomes of Care
 vars:          10                    20 Jan 1998 13:28
-----------------------------------------------------------------------
  1. id          long   %9.0g            Patient ID
  2. provtype    byte   %9.0g    praclbl  Provider Type
  3. chiromd     byte   %9.0g    chirlbl  Chiropractor vs. MD
  4. age         byte   %8.0g            Patient Age in Years
  5. gender      byte   %8.0g    genlbl   Patient Gender
  6. cost        int    %10.0g           Total Cost of LBP Treatment
  7. status4w    byte   %9.0g    func4    Low Back Pain at 4 Weeks
  8. severity    byte   %8.0g            Severity of LBP
  9. severe3     byte   %9.0g    sevlbl   Severity of LBP
 10. sciatica    byte   %8.0g    yesno    Sciatica Present
-----------------------------------------------------------------------
```

Example 1

Does the cost of care for an episode of low back pain differ by the type of provider seen? Provider types are 1 for primary care physician, 2 for chiropractor, 3 for orthopedist, and 4 for HMO physician. To answer this, we would look at the mean cost for each provider type (`provtype`).

```
. adjmean cost, by(provtype)

*Unadjusted* Means and 95% Confidence Intervals

  Outcome:     Total Cost of LBP Treatment -- cost
  Nominal X:   Provider Type -- provtype
  Covariates:  (none)
```

```
 provtype      numobs       mean         se      lower      upper
     1:PC         538   398.0911    26.1085   346.9193   449.2628
  2:Chiro         514   626.3891   26.71108   574.0363   678.7419
  3:Ortho         162   627.5185   47.57903   534.2653   720.7717
    4:HMO         192   340.5104   43.70413   254.8519   426.1689
  Test for difference of 4 means:
    F(3, 1402) =   19.38
    Prob > F   <   0.0001
```

We see that the "unadjusted" mean costs differ by type of provider seen, with patients of chiropractors and orthopedists incurring the largest costs ($626 and $628), and HMO patients the least ($341). The overall F test shows a significant (p <0.0001) difference for the means. There are a number of other ways we could have gotten these same unadjusted means quite simply in Stata, e.g.,

```
. tabulate provtype, summarize(cost)
. oneway cost provtype, tabulate
```

Example 2

An argument might be made that the cost of treating LBP is higher for chiropractors and orthopedist because their patients have more severe pain. We rerun Example 1, but this time we will adjust for variables such as age, severity (a scale from 0 to 23, where 0 means no impairment and 23 means complete incapacitation), and sciatica (another measure of severity, where 1 means sciatica present, 0 means no sciatica). In addition to requesting the adjusted means, we request the model to be printed and a graph of the results.

```
. adjmean cost, by(provtype) adjust(age severity sciatica) model graph ylabel
xlabel(1,2,3,4)
  Source |       SS       df       MS                  Number of obs =    1401
---------+------------------------------               F(  6,  1394) =   48.04
   Model |  89651393.4     6  14941898.9               Prob > F      =  0.0000
Residual |   433565958  1394  311022.925               R-squared     =  0.1713
---------+------------------------------               Adj R-squared =  0.1678
   Total |   523217351  1400  373726.679               Root MSE      =  557.69
------------------------------------------------------------------------------
    cost |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      X2 |   248.1997   34.51434      7.191   0.000       180.4941    315.9054
      X3 |   200.9192   50.19444      4.003   0.000       102.4544     299.384
      X4 |  -24.06906   47.30386     -0.509   0.611      -116.8635    68.72537
     age |  -3.698749   1.149891     -3.217   0.001      -5.954453   -1.443045
severity |   26.00893   2.204246     11.799   0.000       21.68494    30.33293
sciatica |   238.0384   36.73184      6.480   0.000       165.9827     310.094
   _cons |   201.8666   57.91298      3.486   0.001       88.26064    315.4726
------------------------------------------------------------------------------
*Adjusted* Means and 95% Confidence Intervals
  Outcome:     Total Cost of LBP Treatment -- cost
  Nominal X:   Provider Type -- provtype
  Covariates:  age severity sciatica
 provtype      numobs    adjmean         se      lower      upper
     1:PC         536   386.3189   24.11297   339.0583   433.5794
  2:Chiro         513   634.5186   24.69707   586.1132    682.924
  3:Ortho         161    587.238    44.0196   500.9612   673.5149
    4:HMO         191   362.2498   40.61471   282.6465   441.8532
  Test for difference of 4 means:
    F(3, 1394) =   22.43
    Prob > F   <   0.0001
```
(*See Figure 1 below*)

The variable, `X2`, is the dummy variable which compares the adjusted mean cost for chiropractors (a `provtype` of 2) versus primary care physicians (a `provtype` of 1), `X3` compares orthopedists (a `provtype` of 3) to primary care, and `X4` compares HMO's (a `provtype` of 4) to primary care. The orthopedists' mean costs decreases from $628 (Example 1 result) to $587 after adjustment for age, severity of low back pain, and sciatica. Primary care physician–adjusted mean costs are lower as well, while chiropractors and HMO–adjusted costs increase. There is still an overall statistical difference in these means. Chiropractors and orthopedists charge more, even after adjusting for age and measures of severity. A graph of the adjusted means and 95% confidence intervals is in Figure 1. Had we preferred a bar graph, we could have typed

```
. adjmean cost, by(provtype) adjust(age severity sciatica) model ylabel graph bar
```
(*output omitted*)
(*See Figure 2 below*)

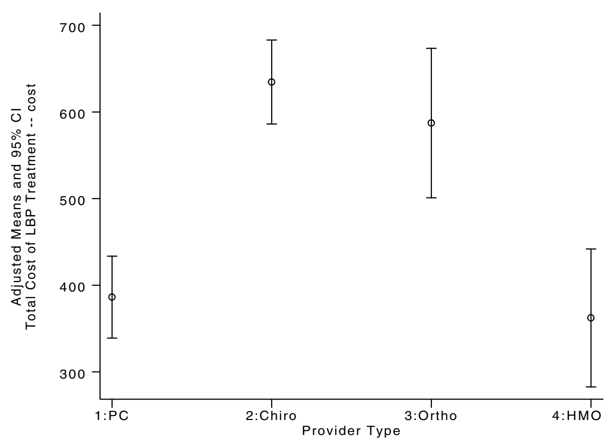


Figure 1. Adjusted means and 95% CI.

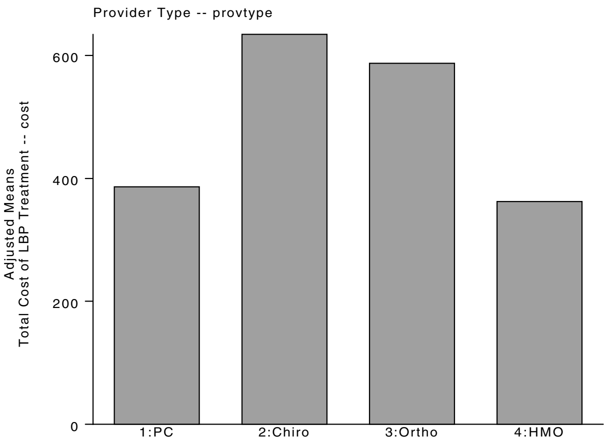


Figure 2. Adjusted means bar graph.

If you would like to test the difference of any pairs of means (not in the model), you can do so with the `test` statement after running `adjmean`. For example, suppose we wanted a test for a comparison between mean costs for chiropractors and orthopedists:

```
. test _b[X2] = _b[X3]
   ( 1)  X2 - X3 = 0.0
       F(  1,  1394) =    0.87
       Prob > F =    0.3499
```

Example 3

Suppose this time we want to compare costs for chiropractors versus any type of physician (`provtype` = 1, 3, and 4 combined), adjusted for age, severity, and sciatica. The variable is called `chiromd` (1=chiropractor, 0=MD). In addition, after fitting the model, let's calculate the predicted mean cost by chiropractor versus MD for someone who is 40 years old, has a low severity score (2), and no sciatica (0). The program does this by estimating the betas for the regression equation, then calculating the adjusted means by substituting specified values for the covariates into the equation.

```
. adjmean cost, by(chiromd) adjust(age=40 severity=2 sciatica=0)
*Adjusted* Means and 95% Confidence Intervals
  Outcome:     Total Cost of LBP Treatment -- cost
  Nominal X:   Chiropractor vs. MD -- chiromd
  Covariates:  age=40 severity=2 sciatica=0
  chiromd      numobs    adjmean         se      lower      upper
     0:MD         888   134.7998    27.6567   80.59362   189.0059
  1:Chiro         513   351.9856   31.48407   290.2779   413.6932
  Test for difference of 2 means:
    F(1, 1396) =   48.28
    Prob > F   <   0.0001
```

Overall, the mean costs are lower for these patients with less severe back pain, though we see that chiropractors have significantly higher costs ($352) than MD's ($135). Let's repeat Example 3 for patients who are the same age (40), but have more severe low back pain, say a high severity score of 20 and the presence of sciatica (1).

```
. adjmean cost, by(chiromd) adjust(age=40 severity=20 sciatica=1)
*Adjusted* Means and 95% Confidence Intervals
  Outcome:        Total Cost of LBP Treatment -- cost
  Nominal X:  Chiropractor vs. MD -- chiromd
  Covariates:     age=40 severity=20 sciatica=1
  chiromd      numobs    adjmean         se      lower      upper
     0:MD         888    847.096   37.21108   774.1641   920.0288
  1:Chiro         513   1064.282   40.35933   985.1794   1143.385
  Test for difference of 2 means:
    F(1, 1396) =   48.28
    Prob > F   <   0.0001
```

Not surprisingly, the cost of care increases substantially when the patients have more severe low back pain. Chiropractors continue to have significantly higher mean costs ($1,064) than MD's ($847).

Example 4

This example estimates the mean costs by categories of two nominal variables: **chiromd** (1=chiropractor, 0=MD) and **sciatica** (1=sciatica present, 0=no sciatica), adjusted for age and severity. The model will be printed, and a graph is requested, defaulting to points which will be connected.

```
. adjmean cost, by(chiromd sciatica) adjust(age severity) model graph ylabel xlabel(0,1) c(ll)
  Source |       SS       df       MS                  Number of obs =    1401
---------+------------------------------               F(  5,  1395) =   53.30
   Model |  83919668.0     5  16783933.6               Prob > F      =  0.0000
Residual |   439297683  1395  314908.733               R-squared     =  0.1604
---------+------------------------------               Adj R-squared =  0.1574
   Total |   523217351  1400  373726.679               Root MSE      =  561.17

------------------------------------------------------------------------------
    cost |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     X12 |   215.2707   35.58975      6.049   0.000       145.4555     285.086
     X22 |     236.85   47.37293      5.000   0.000       143.9201    329.7798
     I22 |    8.35065   74.12591      0.113   0.910      -137.0596    153.7609
     age |  -3.800583   1.151852     -3.300   0.001      -6.060132   -1.541035
severity |   26.23992   2.218988     11.825   0.000       21.88701    30.59283
   _cons |   234.9213   54.74507      4.291   0.000       127.5297    342.3128
------------------------------------------------------------------------------

*Adjusted* Means and 95% Confidence Intervals
  Outcome:      Total Cost of LBP Treatment -- cost
  Nominal X1:   Chiropractor vs. MD -- chiromd
  Nominal X2:   Sciatica Present -- sciatica
  Interaction:  chiromd * sciatica
  Covariates:   age severity

  chiromd  sciatica      numobs    adjmean         se      lower      upper
     0:MD      0:No         705   365.1132   21.21326   323.5359   406.6904
     0:MD     1:Yes         183   601.9631    42.1138   519.4216   684.5047
  1:Chiro      0:No         387   580.3839    28.6184   524.2928    636.475
  1:Chiro     1:Yes         126   825.5845   50.12164    727.348   923.8211

  Test for difference of 4 means:
    F(3, 1395) =   31.83
    Prob > F   <   0.0001

  Test for interaction of chiromd * sciatica:
    F(1, 1395) =    0.01
    Prob > F   =    0.9103
( See Figure 3 below)
```

The dummy variable for `chiromd` is `X12` and the dummy variable for `sciatica` is `X22`, with an interaction variable (`I22`) created between them. (Note: This would have been simpler to read if the variables had retained their original names, but often there are more than two categories per variable, and naming gets complicated. Most of the time, we're not interested in seeing the model, anyway. The default summary statistics usually give all the information we want.)

The highest mean cost is for patients with sciatica seeing chiropractors ($826), but costs are higher also for sciatica patients seeing physicians ($602). There is no evidence of interaction ($p = 0.9013$) which we can see from the regression table for `I22`, or from the F statistic at the bottom of the output. The graph (see Figure 3) confirms this. Although it may not be appropriate to connect the points of a nominal variable, it's done here to demonstrate that the `connect` option can work, as well as to show that two lines couldn't be much more parallel (a clear indication of no interaction).

Had we preferred a bar graph, we could have typed

```
. adjmean cost, by(chiromd sciatica) adjust(age severity) model graph bar ylabel
( See Figure 4 below)
```

(*Graphs on next page*)

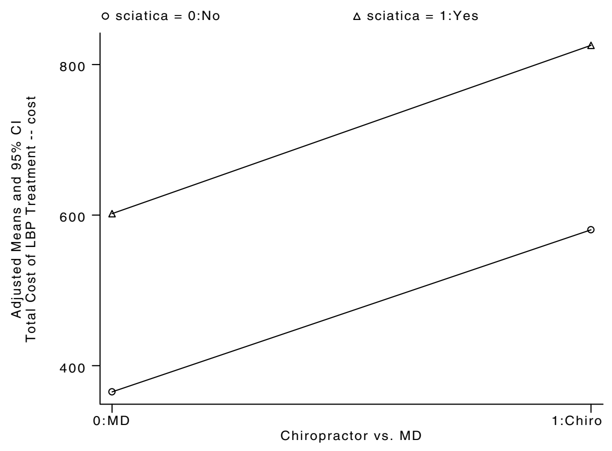


Figure 3. Mean cost of `chirmod` & `sciatica` adjusted for age & severity.

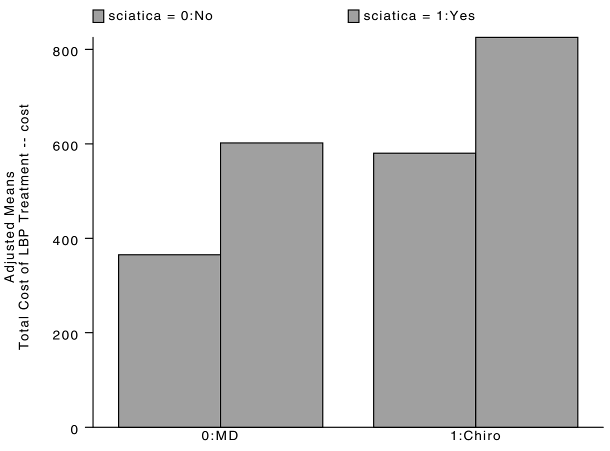


Figure 4. Bar graph version of Figure 3.

Example 5

Let's look at an example where there is significant interaction. When we examine the adjusted mean costs for categories of `chiromd` by gender (1=male, 0=female), we get the following results (model not printed):

```
. adjmean cost, by(chiromd gender) adjust(age severity sciatica) graph bar ylabel
  (output omitted)
*Adjusted* Means and 95% Confidence Intervals

  Outcome:      Total Cost of LBP Treatment -- cost
  Nominal X1:   Chiropractor vs. MD -- chiromd
  Nominal X2:   Patient Gender -- gender
  Interaction:  chiromd * gender
  Covariates:   age severity sciatica

  chiromd    gender      numobs    adjmean         se       lower      upper
     0:MD  0:Female         182   304.1542   41.50648     222.803   385.5054
     0:MD    1:Male         705   446.4079   20.99842    405.2517   487.5641
  1:Chiro  0:Female          60   881.6437   72.07227    740.3847   1022.903
  1:Chiro    1:Male         453   601.9128   26.28687    550.3915   653.4341

  Test for difference of 4 means:

    F(3, 1393) =   23.94
    Prob > F   <   0.0001

  Test for interaction of chiromd * gender:

    F(1, 1393) =   22.19
    Prob > F   <   0.0001
```

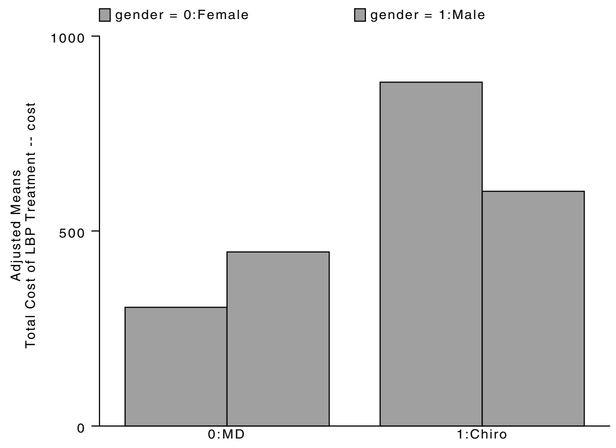


Figure 5. Mean costs by `chiromd` and `gender`.

Within chiropractors, mean costs are higher for females ($882) than males ($602), whereas within physicians, the relationship is reversed—females: $304, males: $446. The test for interaction is highly significant (p <0.0001), and the bar graph (see Figure 5) illustrates the relationship visually.

Example 6

If we repeat the last example using our original four categories for provider (`provtype`), we find that females have higher mean costs than males if they visit chiropractors, but have about the same or lower costs than males for primary care, orthopedic, or HMO physicians. This is easiest to see in Figure 6.

```
. adjmean cost, by(provtype gender) adjust(age severity sciatica) graph bar ylabel
  (output omitted )
*Adjusted* Means and 95% Confidence Intervals

  Outcome:      Total Cost of LBP Treatment -- cost
  Nominal X1:   Provider Type -- provtype
  Nominal X2:   Patient Gender -- gender
  Interaction:  provtype * gender
  Covariates:   age severity sciatica

 provtype    gender      numobs    adjmean         se      lower      upper
     1:PC  0:Female          89   268.8453   58.88354   153.4357   384.2549
     1:PC    1:Male         447   409.9181   26.25605   358.4572    461.379
  2:Chiro  0:Female          60    882.098   71.71287   741.5433   1022.653
  2:Chiro    1:Male         453   601.6091   26.15592   550.3444   652.8737
  3:Ortho  0:Female          25   569.0792   110.9323    351.656   786.5024
  3:Ortho    1:Male         136   590.6166   47.57316   497.3749   683.8583
    4:HMO  0:Female          68   254.8119   67.46792   122.5772   387.0466
    4:HMO    1:Male         122   419.2326   50.38327   320.4832    517.982

  Test for difference of 8 means:

    F(7, 1389) =   12.94
    Prob > F   <   0.0001

  Test for interaction of provtype * gender:

    F(3, 1389) =    7.34
    Prob > F   <   0.0001
```

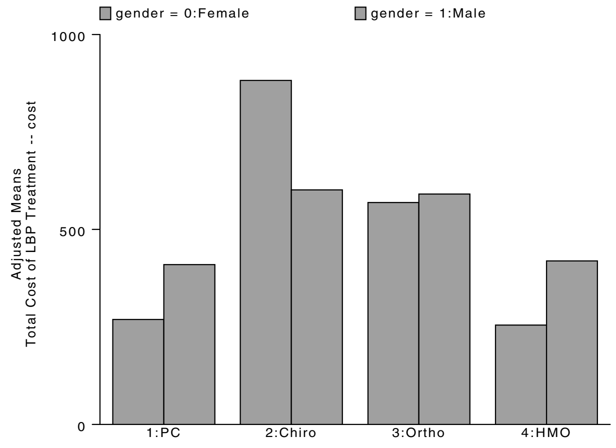


Figure 6. Mean costs adjusted by provider type and gender.

Example 7

In the previous examples, we looked at adjusted means (costs), so let's look at a couple of examples where the outcome is dichotomous rather than continuous, and we want to estimate adjusted probabilities using logistic regression (`adjprop`). Another outcome we considered in this study was how quickly patients got over their back pain. Of course, each provider type was sure their patients would get better quicker. We created a variable, `status4w`, which measured whether patients still had back pain at four weeks (`status4w`: 1=not better, 0=better). So, our logistic regression model predicts the probability of still having low back pain after four weeks. Since patients with less pain at baseline may have improved more quickly, we will adjust for `age`, `severity`, and `sciatica`. The following statement includes a request to display the logistic regression model results and a bar graph by provider type:

```
. adjprop status4w, by(provtype) adj(age severity sciatica) model graph bar ylabel
Logit Estimates                                         Number of obs =   1401
                                                        chi2(6)       = 170.75
                                                        Prob > chi2   = 0.0000
Log Likelihood = -582.29855                             Pseudo R2     = 0.1279
```

```
------------------------------------------------------------------------------
status4w | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      X2 |   .9653365   .1661172     -0.205   0.838       .6889728    1.352556
      X3 |   1.015132   .2483003      0.061   0.951       .6285202    1.639553
      X4 |   1.240407   .2876763      0.929   0.353        .787323     1.95423
     age |   1.007679   .0057617      1.338   0.181       .9964497    1.019036
sciatica |    1.54771   .2516177      2.687   0.007       1.125398    2.128497
severity |   1.145833   .0144185     10.818   0.000       1.117919    1.174444
------------------------------------------------------------------------------

*Adjusted* Probabilities and 95% Confidence Intervals

  Outcome:     Low Back Pain at 4 Weeks -- status4w
  Nominal X:   Provider Type -- provtype
  Covariates:  age sciatica severity

 provtype      numobs    adjprob         se      lower      upper
     1:PC         536   .1408429   .1300396   .1127274   .1745909
  2:Chiro         513   .1366278   .1322635   .1088233   .1701798
  3:Ortho         161     .14267   .2193117   .0976924   .2036809
    4:HMO         191   .1689809   .2025572   .1202704   .2322121

  Likelihood ratio test for difference of 4 probabilities:

    LR Chi2(3)    =    1.16
    Prob > Chi2   =    0.7635
```

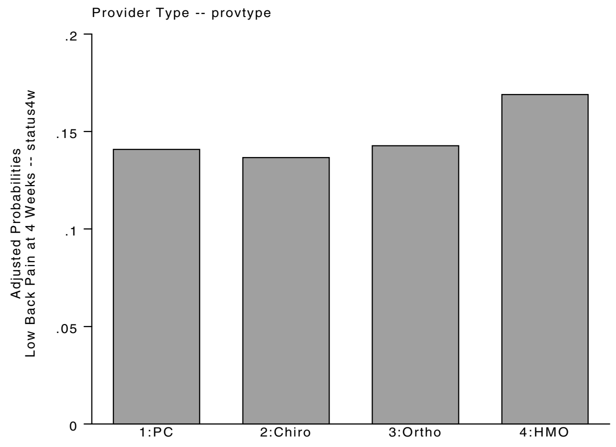


Figure 7. Adjusted probabilities by provider type.

We see odds ratios for the model (coefficients are available if we follow this statement with `logit`). For instance, `X2` is the odds ratio of not being better at 4 weeks for chiropractic patients compared to primary care patients. The adjusted probabilities are similar to each other, with about 14.1% of primary care, 13.7% of chiropractic, 14.3% of orthopedic, and 16.9% of HMO patients not better at 4 weeks. Although it looks like more of the HMO patients in Figure 7 still have low back pain, the differences between provider types is not statistically significant (p =0.7635). The bottom line—no matter who you go see to treat low back pain, you'll get better in about the same amount of time. (Actually, an alternative strategy suggests you'll do just as well and save a lot of money if you avoid seeing anyone.)

Example 8

Finally, let's look at the probabilities of not being better (`status4w`) for `provtype` and the severity score divided into three categories (`severe3`: 1=none/mild, 2=moderate, 3=severe), adjusted for `age` and `sciatica`. A bar graph is displayed.

```
. adjprop status4w, by(provtype severe3) adj(age sciatica) graph bar ylabel

*Adjusted* Probabilities and 95

  Outcome:      Low Back Pain at 4 Weeks -- status4w
  Nominal X1:   Provider Type -- provtype
  Nominal X2:   Severity of LBP -- severe3
  Interaction:  provtype * severe3
  Covariates:   age sciatica

 provtype    severe3      numobs    adjprob         se      lower      upper
     1:PC     1:Mild         146   .0487137   .3879754   .0233787   .0987282
     1:PC      2:Mod         144   .1050477   .2737591   .0642291   .1671729
     1:PC   3:Severe         246   .3161212   .1382498   .2606449   .3773784
  2:Chiro     1:Mild         138   .0369936   .4560503   .0154716   .0858438
  2:Chiro      2:Mod         168   .1444471   .2182219   .0991646   .2056861
```

```
2:Chiro   3:Severe        207   .2603206   .1591271   .2048608    .324664
3:Ortho     1:Mild         35    .028247   1.015757   .0039544    .175481
3:Ortho      2:Mod         48   .1085458   .4736061   .0459162   .2355144
3:Ortho   3:Severe         78    .303043     .24695   .2113409   .4136648
  4:HMO     1:Mild         61   .0366448   .7199808   .0091911   .1349348
  4:HMO      2:Mod         51   .1491321   .4084734   .0729646   .2807324
  4:HMO   3:Severe         79   .3492898   .2377596   .2519644   .4610386
Likelihood ratio test for difference of 12 probabilities:
  LR Chi2(11)    =  128.30
  Prob > Chi2    <   0.0001
Likelihood ratio test of interaction for provtype * severe3:
  LR Chi2(6)     =    3.19
  Prob > Chi2    =    0.7845
```

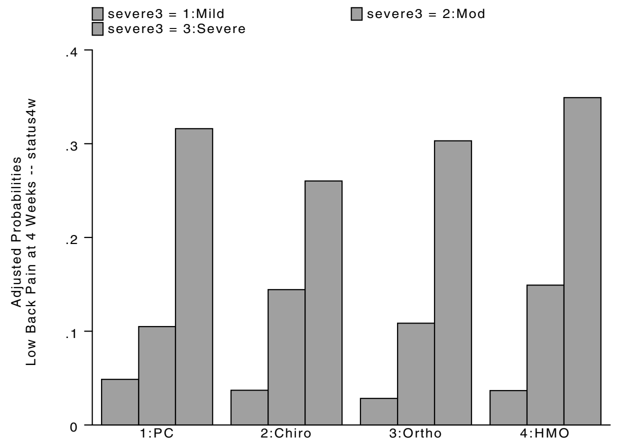


Figure 8. Probability of low back pain at 4 weeks adjusted by provider type and severity of LBP.

We have fit a logistic regression model with three dummy variables for `provtype`, two dummy variables for `severe3`, and the interaction terms between them. There is no evidence of interaction ($p = 0.7845$). In the bar graph (Figure 8), we see very little difference in the probability of not recovering by four weeks across provider type, but a distinct difference by degree of severity. Only 3% to 5% of patients who had mild pain at baseline have not recovered by four weeks, 10% to 15% of those with moderate pain have not recovered, and 26% to 35% of those with severe pain are still not better.

References

Garrett, J. M. 1995. sg33: Calculation of adjusted means and adjusted proportions. *Stata Technical Bulletin* 24: 22–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 161–165.

Carey, T. S., J. M. Garrett, A. Jackman, C. McLaughlin, and J. Fryer. 1995. The outcomes and costs of care for acute low back pain among patients seen by primary care practitioners, chiropractors and orthopedic surgeons. *New England Journal of Medicine* 333(14): 913–917.

sg81 Multivariable fractional polynomials

Patrick Royston, Imperial College School of Medicine, UK, proyston@rpms.ac.uk
Gareth Ambler, Imperial College School of Medicine, UK, gambler@rpms.ac.uk

[*Editor's note: This insert requires that you install the official updates—the* `stata` *directory.*]

Introduction

A fractional polynomial (FP) is a polynomial whose powers are integers or fractions which may be positive, zero, or negative. FPs are extremely useful in regression models because they offer greater flexibility than ordinary polynomials. The Stata command `fracpoly` fits regression models with a single predictor transformed to a fractional polynomial, with any other predictors untransformed. The procedure is suitable when we are interested in a single continuous covariate and we wish to control for other covariates linearly. The ado-file `mfracpol` described here may be used to model two or more continuous covariates simultaneously using FPs.

Multivariable fractional polynomial models are especially useful when one wishes to preserve the continuous nature of the predictor variables in a regression model, but suspects that some or all of the relationships are nonlinear. Using a 'backfitting' algorithm, `mfracpol` finds a fractional polynomial transformation for each predictor in turn, while holding the functional forms of the other predictors temporarily fixed. The algorithm converges when the functional forms of the predictors (i.e. the FP powers) do not change. Variable selection by backward elimination is available as an option within the procedure.

To demonstrate the versatility of the method we give examples using the well-known `auto` dataset supplied with Stata, and a leg ulcer dataset previously analyzed by Smith et al. (1992) and Royston & Altman (1994). The former is assumed to have normal errors and the latter is a survival-time dataset.

The ado-file `mfracpol` is a regression-like command with the following basic syntax:

`mfracpol` *regression_cmd yvar xvars* [*weight*] [`if` *exp*] [`in` *range*] `,` [`df(`*df_list*`)` `sel`ect`(`*select_list*`)` `al`pha`(`*alpha_list*`)` *minor_options*]

Full details are given in the section *Syntax of the* `mfracpol` *command*.

Fractional polynomials were introduced by Royston and Altman (1994), and are described in detail in [R] **fracpoly**. The backfitting algorithm is described by Royston and Altman (1994) with refinements (implemented here) by Sauerbrei and Royston (1998).

Fractional polynomials

As noted earlier, a fractional polynomial in x is a polynomial whose powers may be fractional. The polynomial may contain any number of terms but models with one or two terms are suitable for most practical purposes. This is especially true when we have several predictors and there is a danger of overfitting. Theoretically, the powers may take any real value, but by default `fracpoly` uses values from the reduced set $\{-2, -1, -0.5, 0, 0.5, 1, 2, 3\}$ where the power 0 refers to the $\log$ transformation. If a power p is repeated, the FP contains terms in x^p and $x^p \log x$.

Fractional polynomials are useful when we wish to transform a predictor to try to improve the fit of a linear regression model. The best degree-one FP model has a single power of x, which is found by fitting models corresponding to each power in the above set in turn and choosing the model with the lowest deviance (defined as minus twice the maximized log likelihood). The power 1 corresponds to the linear model. The best degree-two FP model has 2 terms in x and is found by fitting models which correspond to every possible pair of powers in the set. Two parameters (degrees of freedom) are considered to be used by every term in a fractional polynomial model: one for each regression coefficient and one for each power. This is an approximation in that the set of possible powers is finite, rather than infinite as would be the case if arbitrary powers were allowed.

Backfitting algorithm and variable selection procedure

Before backfitting starts, `mfracpol` determines the order in which the predictors are to be processed. By default, all predictors are entered as linear into a multiple regression model and the processing order is determined from the p values of the Wald statistics for removing each variable singly from the model. The predictor with the smallest p value (denoted by x_1) is processed first, that with the second smallest p value is dealt with second, and so on. The aim is to model relatively unimportant variables after important variables, which may help reduce potential model-fitting difficulties associated with collinearity or “concurvity.” As always with multiple regression, however, the inclusion of highly correlated predictors may cause difficulties. For example, the final FP chosen for a pair of correlated variables may depend on which variable happens to be modeled first in the backfitting algorithm, that is, on the significance of each variable in a multiple regression model as described above.

The present paragraph describes the default case in which all variables are to be included in a final model—no variable selection is applied. At the initial cycle of the algorithm, all predictors which have not yet been considered for transformation are entered into the regression model as linear. The first step is to consider whether there is any benefit in transforming x_1 to an FP. This is accomplished by comparing the model with a second-degree FP in x_1 with that with a first-degree FP in x_1, using a deviance-based significance test. The deviance difference between the models is compared with the appropriate percentile of a χ^2 distribution with two degrees of freedom. (In the case of normal-errors models, an F test is used.) If the corresponding p value is less than the FP selection level, controlled by the `alpha()` option, a second-degree FP is chosen. If not, then the first-degree FP model is compared with a linear model for x_1 by comparing the deviance difference with a χ^2 distribution with one degree of freedom. If the p value is less than the selection level, again given by `alpha()`, the first-degree FP is chosen, otherwise no transformation is applied to x_1. The `alpha()` selection level may be specified for each variable independently. The default value is 0.05.

Variable selection is controlled by the `select()` option and may be specified for each variable independently. The default selection level is 1, and since a p value is always less than or equal to 1, all variables are then included in the final model. Any variable whose value of `select()` is less than 1 is considered for elimination from the model as follows. The backfitting algorithm proceeds as above to the point at which x_1 is considered. Before comparing the second and first-degree FPs, the deviance difference between the model with a second-degree FP in x_1 and that with x_1 omitted is compared with a χ^2 distribution with 4 degrees of freedom. If the corresponding p value is greater than the selection level according to the `select()` option, x_1 is dropped from the model for the rest of the cycle. Otherwise, the procedure described above for determining the FP transformation is applied. On the next cycle, x_1 is reconsidered in the same way. Convergence occurs when the set of variables in the model and their FP transformations are stable.

The above description assumes that all predictors are assigned 4 degrees of freedom, which is the maximum permitted by `mfracpol`. If, for example, x_1 is assigned 2 degrees of freedom the variable selection test is based on a comparison of the first-degree FP model with that with x_1 omitted, and uses a χ^2 distribution with 2 degrees of freedom (or F test for normal errors models). If x_1 survives, the model with a first-degree FP transformation is compared with that with x_1 untransformed. Similarly, if x_1 is initially assigned 1 degree of freedom, the only test performed is that comparing the model with x_1 untransformed (i.e. linear) with that with x_1 omitted. The latter would apply to all binary predictors, for example.

Comments on selection levels

We reemphasize that the FP procedure described in the previous section uses two independent selection levels, controlled by the `select()` and `alpha()` options respectively. The `select()` option determines whether a predictor is eligible to be dropped from the model, a selection level of one prohibiting exclusion. A low selection level such as 0.001 will result in simpler models but with a possible loss of power in detecting influential variables, and hence possibly in biased predictions. A higher value will produce more complex models with an increased chance of overfitting and poorer predictive accuracy. This tension is an example of the classical trade-off between bias and variance. Sauerbrei (1998) discusses variable selection issues in some detail. Note that the variable selection test is affected by the initial number of degrees of freedom chosen for a variable, controlled by the `df()` option. It is possible for a variable to be included if it is specified to have 2 degrees of freedom but excluded if 4 degrees of freedom are specified. This can occur if the true functional form is well approximated by a first-degree FP and the data are noisy. Power is lost by performing the exclusion test with 4 degrees of freedom instead of the more appropriate value of 2 degrees of freedom. If the relationship between a predictor and the outcome is expected or required to be monotonic, there is a case for specifying 2 degrees of freedom; otherwise, we recommend allowing more general second-degree FP functions by specifying 4 degrees of freedom.

Example 1: A model with normal errors

We now use `mfracpol` to fit multivariable fractional polynomial models for the `auto` dataset. The dataset contains details on prices, mileages, weights and other characteristics of 74 cars in the year 1978. It was compiled and published by Chambers et al. (1983). We choose first to use a simple normal-errors model for fuel consumption (`mpg`) as the response, with `weight`, car type (`foreign`) and cylinder displacement (`displ`) as the predictors. `foreign` is a binary variable indicating whether the car is foreign (1) or American made (0). We allow `weight` and `displ` to have 4 degrees of freedom and `foreign` 1 degree of freedom using the option `df(4,foreign:1)`. In other words, second-degree FP transformations are considered for `weight` and `displ`, whereas foreign is entered untransformed.

We fit the model using the command

```
. mfracpol regress mpg weight foreign displ, df(4, foreign:1) alpha(0.05) select(1)
```

We use the default value of 1 for the `select()` option which ensures that all predictors are included in the final model. We also use the default value `alpha(0.05)`for the FP selection level. The backfitting algorithm for this model converges after the first cycle. The following results are obtained:

```
Deviance for model with all terms linear =   388.327, 74 observations

         --m = 1 vs linear--    --- m = 2 vs m = 1 ---  Final     P(FP models)
Variable  Power  Dev.diff.  P    Powers   Dev.diff.  P   Powers    m = 1  m = 2
-------------------------------------------------------------------------------
weight     -1     7.270  0.009   -2,-2      0.585  0.767 -1        0.000  0.000
foreign    1                                             1         0.031
displ      -2     2.503  0.128   -2,3       0.704  0.727 1         0.141  0.342

Fractional polynomial fitting algorithm converged after 1 cycle.

Final multivariable fractional polynomial model for mpg
-------------------------------------------------------

Variable |    -----Initial-----          -----Final-----
         |   df     Select   Alpha    Status    df    Powers
---------+------------------------------------------------
  weight |    4     1.0000   0.0500     in       2     -1
 foreign |    1     1.0000   0.0500     in       1     1
   displ |    4     1.0000   0.0500     in       1     1
---------+------------------------------------------------

  Source |       SS       df       MS                Number of obs =      74
---------+------------------------------             F(  3,    70) =   53.02
   Model |  1696.79491     3  565.598304             Prob > F      =  0.0000
Residual |  746.664549    70  10.6666364             R-squared     =  0.6944
---------+------------------------------             Adj R-squared =  0.6813
   Total |  2443.45946    73  33.4720474             Root MSE      =   3.266
```

```
------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
weight_1 |   48.06922   7.211376      6.666   0.000       33.68658    62.45185
 foreign |  -2.380441   1.078488     -2.207   0.031      -4.531417   -.2294645
   displ |  -.0097729   .0076711     -1.274   0.207      -.0250724    .0055265
   _cons |   6.866634   3.821799      1.797   0.077      -.7557029    14.48897
------------------------------------------------------------------------------
Deviance:  381.058.
```

We see in the first table that **weight** initially undergoes an FP transformation with powers $-2,-2$. The FP-transformed predictor is tested for inclusion in the model, using an F test with 4 numerator degrees of freedom, which results in the p value of 0.000 given in the column **P(FP models): m=2**. The selection level of 1 forces all the predictors into the model so this predictor is included. The program also displays the result of an inclusion test for the first-degree FP transformation, which is based on 2 degrees of freedom. The degree two and degree one (-1) transformations are compared resulting in a p value of 0.767 which is greater than the FP-selection level of 0.05. The degree-two transformation is statistically not significantly better than a degree-one FP, so the latter is preferred. This is compared with a linear term in **weight**, for which the p value is 0.009 so the degree-one FP is chosen.

The binary predictor **foreign** is then processed and kept in the model. By convention it has power 1. Next, the predictor **displ** is considered. The p value for inclusion is only 0.342 but the predictor is kept in the model as no selection is taking place. Both the FP selection tests reject the more complicated transformation so the predictor is left untransformed. At this stage, the backfitting algorithm finishes since **mfracpol** realizes that the functional forms of the predictors will not change on the next iteration.

The second table summarizes the initial settings and the final functional forms of each predictor. In particular, the column labeled "Status" indicates whether the predictor is included in the final model or not. In this case the final model is

$$\texttt{mpg} = \beta_0 + \beta_1 * \texttt{weight}^{-1} + \beta_2 * \texttt{displ} + \beta_3 * \texttt{foreign} + \varepsilon$$

This final model is fitted and the appropriate output from the **regress** command is displayed. The fitted functions for each predictor may be plotted using version 2.1.0 (or later) of **fracplot**, released in STB-41. The command

```
fracplot weight, xlab ylab
```

produces the graph in Figure 1. We see that the nonlinear function seems to fit the data well.

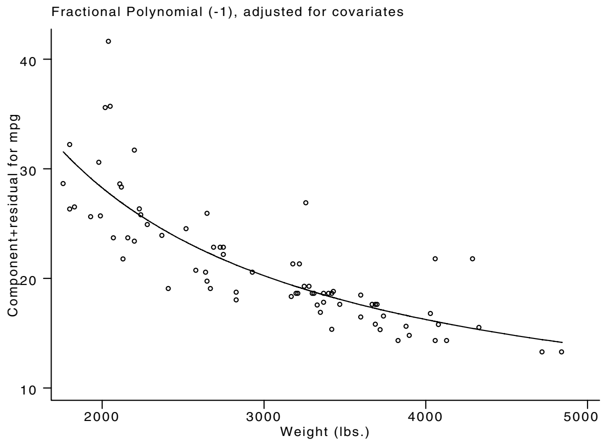


Figure 1. The first model fit to the auto data.

We now see what happens when we enter all the predictors in the model and assign selection level 0.05 to each of them. However, for illustrative purposes, we will assign a selection level of 1 to **foreign** to ensure that it is included in the final model. The predictor **rep78** is categorical and records each car's repair record for 1978. We split this into 4 indicator variables **r78_2-r78_5**, where **r78_2** is 1 if **rep78** is 2, and so on. We fit the model using

```
. mfracpol regress mpg displ foreign length weight turn trunk hdroom
> gratio price r78_2-r78_5, df(4,foreign r78_2-r78_5:1) select(0.05,
> foreign:1)
```

On the third and final cycle of the algorithm, the output is

```
          --m = 1 vs linear--    --- m = 2 vs m = 1 ---  Final     P(FP models)
Variable  Power  Dev.diff.  P    Powers   Dev.diff.  P   Powers    m = 1  m = 2
-------------------------------------------------------------------------------
foreign   1                                              1         0.002
gratio    3       1.979  0.184   3,3        2.646  0.317 .         0.085  0.126
length    2       0.208  0.665   -2,-2      1.134  0.606 1         0.002  0.010
r78_5     1                                              1         0.006
displ     -2     10.527  0.002   -2,3       0.354  0.855 -2        0.004  0.023
turn      -2      0.509  0.501   -2,-2      0.050  0.979 .         0.191  0.509
weight    .5      0.015  0.906   -2,-2      0.465  0.817 .         0.606  0.845
price     -2      0.930  0.363   3,3        1.012  0.644 .         0.277  0.490
r78_3     1                                              .         0.355
trunk     -1      0.396  0.553   1,2        0.385  0.846 .         0.661  0.886
r78_2     1                                              .         0.689
r78_4     1                                              .         0.496
hdroom    3       0.042  0.847   2,3        0.413  0.836 .         0.982  0.982
Fractional polynomial fitting algorithm converged after 3 cycles.
Final multivariable fractional polynomial model for mpg
-------------------------------------------------------

Variable |    -----Initial-----         -----Final-----
         |   df     Select   Alpha    Status    df    Powers
---------+-------------------------------------------------
   displ |   4     0.0500   0.0500     in      2     -2
 foreign |   1     1.0000   0.0500     in      1     1
  length |   4     0.0500   0.0500     in      1     1
  weight |   4     0.0500   0.0500     out     0
    turn |   4     0.0500   0.0500     out     0
   trunk |   4     0.0500   0.0500     out     0
  hdroom |   4     0.0500   0.0500     out     0
  gratio |   4     0.0500   0.0500     out     0
   price |   4     0.0500   0.0500     out     0
   r78_2 |   1     0.0500   0.0500     out     0
   r78_3 |   1     0.0500   0.0500     out     0
   r78_4 |   1     0.0500   0.0500     out     0
   r78_5 |   1     0.0500   0.0500     in      1     1
---------+-------------------------------------------------

  Source |       SS       df       MS              Number of obs =      69
---------+------------------------------           F(  4,    64) =   46.80
   Model |  1743.96449     4  435.991122           Prob > F      =  0.0000
Residual |   596.23841    64  9.31622516           R-squared     =  0.7452
---------+------------------------------           Adj R-squared =  0.7293
   Total |   2340.2029    68  34.4147485           Root MSE      =  3.0522

------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 displ_1 |   7.006063   1.983264      3.533   0.001       3.044037    10.96809
 foreign |  -3.776967   1.180833     -3.199   0.002      -6.135953   -1.417982
  length |  -.1217434    .032669     -3.727   0.000      -.1870071   -.0564796
   r78_5 |   3.327015   1.166615      2.852   0.006       .9964348    5.657596
   _cons |   41.42275   6.911346      5.993   0.000       27.61575    55.22975
------------------------------------------------------------------------------
Deviance:  344.614.
```

We see that most of the variables are dropped from the model since the inclusion tests resulted in p values greater than the selection level of 0.05. Also, we note that the -2 power is chosen for the predictor `displ`. We view the nonlinear relationship between displacement and fuel consumption using

```
. fracplot displ, xlab ylab
```

The resulting graph is shown in Figure 2. This closely resembles the relationship between weight and fuel consumption in Figure 1, which is not surprising since the linear correlation between `weight`$^{-1}$ and `displ`$^{-2}$ is 0.94. We also notice that after adjusting for displacement and length, American cars are more economical than foreign ones, and that those with a category-5 repair record are more economical than others.

(*Graph on next page*)

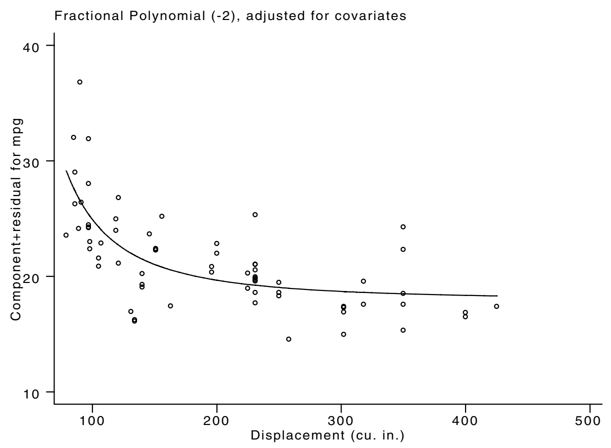


Figure 2. The second model fit to the `auto` data.

Example 2: Survival data

Multivariable fractional polynomials were shown by Sauerbrei and Royston (1998) and Sauerbrei et al. (1998) to be useful when modeling survival times in breast cancer. In this example we use `mfracpol` to fit Cox proportional hazards models (Cox 1972) to data where the response is healing time. The data arise from a randomized controlled clinical trial of two treatments of leg ulcer in 200 patients (Smith et al., 1992). We carry out a complete-case analysis on 183 patients with 92 events and 91 censored observations. The response `y` is the number of days from diagnosis to complete healing. The censoring variable is `cens`, and the continuous predictors are initial ulcerated area (`area`), number of months between onset of the ulcer and first treatment (`mnths`), `age`, diastolic blood pressure (`dbp`), `height`, ankle pressure (`ankp`), and `weight`. In addition there are two binary indicator variables which denote the differential treatment effect (`trt`), and the presence or absence of deep vein involvement (`dvi`).

We are interested in constructing a prognostic score to predict a tendency for fast healing given the relevant covariates. Initially, we include all the predictors using

```
. use legulcer
. mfracpol cox y area mnths age dbp height ankp weight trt dvi,
> dead(cens) df(4,trt dvi:1) select(0.05)
```

The `dead()` option is used with the `cox` regression command to specify the censoring variable. The third and final cycle of the algorithm produces the following output:

```
           --m = 1 vs linear--    --- m = 2 vs m = 1 ---  Final     P(FP models)
Variable  Power  Dev.diff.  P     Powers   Dev.diff.  P   Powers    m = 1  m = 2
--------------------------------------------------------------------------------
area       .5     3.067  0.080   0,.5       0.127  0.938 1         0.000  0.000
dvi        1                                             1         0.006
age        -2     2.382  0.123   -2,-2      0.753  0.686 1         0.012  0.049
dbp        -2     1.336  0.248   -2,-2      0.867  0.648 .         0.061  0.166
trt        1                                             .         0.255
mnths      0     12.756  0.000   .5,.5      2.494  0.287 0         0.000  0.001
ankp       3      0.159  0.690   3,3        0.815  0.665 .         0.283  0.503
height     3      0.164  0.685   3,3        0.500  0.779 .         0.633  0.842
weight     3      0.449  0.503   2,2        0.287  0.866 .         0.499  0.795
Fractional polynomial fitting algorithm converged after 3 cycles.
Final multivariable fractional polynomial model for y
---------------------------------------------------

Variable |    -----Initial-----          -----Final-----
         |   df     Select   Alpha    Status    df    Powers
---------+--------------------------------------------------
    area |    4     0.0500   0.0500     in       1     1
   mnths |    4     0.0500   0.0500     in       2     0
     age |    4     0.0500   0.0500     in       1     1
     dbp |    4     0.0500   0.0500     out      0
  height |    4     0.0500   0.0500     out      0
    ankp |    4     0.0500   0.0500     out      0
  weight |    4     0.0500   0.0500     out      0
     trt |    1     0.0500   0.0500     out      0
     dvi |    1     0.0500   0.0500     in       1     1
---------+--------------------------------------------------
```

```
Cox Regression -- entry time 0                          Number of obs =    183
                                                        chi2(4)       = 101.47
                                                        Prob > chi2   = 0.0000
Log Likelihood = -364.46471                             Pseudo R2     = 0.1222
------------------------------------------------------------------------------
       y |
    cens |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    area |   -.001604   .0003375     -4.753   0.000      -.0022654   -.0009426
 mnths_1 |  -.4032784    .106552     -3.785   0.000      -.6121166   -.1944403
     age |  -.0229402   .0087782     -2.613   0.009      -.0401452   -.0057352
     dvi |  -.5857493   .2165408     -2.705   0.007      -1.010162   -.1613371
------------------------------------------------------------------------------
Deviance:  728.929.
```

At convergence, just 4 predictors remain: `area`, `age`, `mnths`, and `dvi`. Of these, `mnths` is log transformed (in fact, it is log(`mnths+1`) to avoid zeroes). The effect of this variable is illustrated in Figure 3.

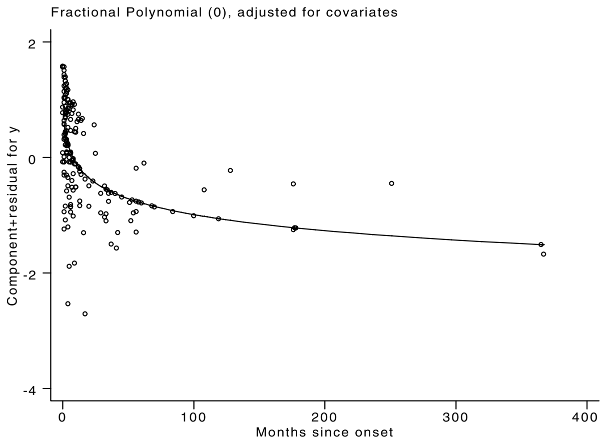


Figure 3. Model fit to leg ulcer data.

We see that an increase in the number of months since the onset of the ulcer is related to a decrease in the log relative "hazard" of healing, hence to an increased healing time as we would expect. The relationship between the log relative hazard and `mnths` is nonlinear and appears to level off for patients with long intervals before receiving treatment. For example, the model predicts the healing time to be similar if treatment was initiated after 100 or after 200 months from onset, but to be very much quicker for short intervals.

The above model differs from that obtained by Royston and Altman (1994) using an earlier version of the multivariable FP algorithm with the same data. Their model included `area`, `age`, `dbp` and `mnths` with powers of 0.5, −2, 1 and 0 respectively, and `dvi`, and had a deviance of 718.39. The reason for the discrepancy is that they used selection level 0.1 for both `select()` and `alpha()`. In addition, they did not apply an overall variable inclusion test, as implemented in `mfracpol`. The effect is to increase the Type I error rate of the selection procedure beyond the nominal level. If we use the command

```
. mfracpol cox y area mnths age dbp height ankp weight trt dvi,
> dead(cens) df(2,trt dvi:1) alpha(0.1) select(0.1)
```

we obtain Royston and Altman (1994)'s final model.

Syntax of the mfracpol command

`mfracpol` *regression_cmd yvar xvars* [*weight*] [`if` *exp*] [`in` *range*] ,
[*major_options minor_options regression_cmd_options*]

where *regression_cmd* may be

`regress, fit, logit, logistic, cox, poisson`.

All weight types supported by *regression_cmd* are allowed.

Options

The *major_options* (most used options) are

`df(`*df_list*`)` `alpha(`*alpha_list*`)` `select(`*select_list*`)`.

The *minor_options* (less used options) are

`catzero(`*varlist*`) cycles(`*#*`) powers(`*list*`) zero(`*varlist*`)`.

regression_cmd_options are options appropriate to the regression command in use.

Major options

`df(`*df_list*`)` sets the degrees of freedom for each predictor. The degrees of freedom (not counting the regression constant, `_cons`) are twice the degree of the fractional polynomial, so for example an *xvar* fitted as a second-degree FP ($m = 2$) has 4 degrees of freedom. The first item in *df_list* may be either # or *varlist*:#. Subsequent items must be *varlist*:#. Items are separated by commas and *varlist* is specified in the usual way for variables. With the first type of item, the degrees of freedom for all predictors are taken to be #. With the second type of item, all members of *varlist* (which must be a subset of *xvars*) have # degrees of freedom. For example `df(2, weight:4)` allows weight 4 degrees of freedom and all other predictors 2 degrees of freedom. The default is 1 degree of freedom for each predictor.

`alpha(`*alpha_list*`)` sets the significance level for testing between FP models of different degree. The rules for *alpha_list* are the same as for *df_list* in the `df()` option. Each selection level must lie between 0 and 1 and the default selection level is 0.05 for all predictors.

`select(`*select_list*`)` sets the significance levels for variable selection. A variable is dropped if its removal causes a nonsignificant increase in deviance at the relevant selection level. The rules for *select_list* are the same as for *df_list* in the `df()` option. Each selection level must lie between 0 and 1. The default selection level of 1 forces all relevant predictors into the model.

*regression_cmd_option*s are options appropriate to *regression_cmd*, such as `dead(`*deadvar*`)` for `cox`.

Minor options

`catzero(`*varlist*`)` creates a binary variable corresponding to zero values for each member of *varlist*, where *varlist* must be a subset of *xvars*. Each binary variable is linked to the corresponding member of *varlist* and both are tested together for inclusion in the model. An application is the assessment of the effect of cigarette smoking, where we may believe that non-smokers are qualitatively different from smokers. See also the `zero()` option below

`cycles(`*#*`)` is the maximum number of iteration cycles permitted. The default value is 5.

`powers(`*list*`)` is the set of fractional polynomial powers to be used. The default set is $\{-2\ -1\ -0.5\ 0\ 0.5\ 1\ 2\ 3\}$ where 0 refers to log.

`zero(`*varlist*`)` allows negative and zero values of members of *varlist* to be ignored when FP transformations are applied. It is conventional for the user to assess the effect of these values through the use of indicator variables, such as described in the `catzero()` option above. By default, any variables with nonpositive values are subjected to a preliminary linear transformation to avoid negative and zero values (see [R] **fracpoly**). *varlist* must be a subset of *xvars*.

Saved results

`mfracpol` saves in the `S_#` macros:

`S_1`	names of variables in final model, including FP-transformed variables if any
`S_2`	deviance of the final model

Note

This software requires the `fracpoly` family of functions which were released with STB-41.

Acknowledgment

We thank Willi Sauerbrei for helpful comments on an earlier draft. The research received financial support from project grant number 045512/Z/95/Z from The Wellcome Trust.

References

Chambers, J. M., W. S. Cleveland, B. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth International Group.

Cox, D. R. 1972. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society, Series B* 74: 187–220.

Royston, P. and D. G. Altman. 1994. Regression using fractional polynomials of continuous covariates: parsimonious parametric modelling (with discussion). *Applied Statistics* 43: 429–467.

Sauerbrei, W. 1998. The use of resampling methods to simplify regression models in medical statistics. *Applied Statistics*, submitted.

Sauerbrei, W. and P. Royston. 1998. Building multivariable prognostic and diagnostic models: transformation of the predictors using fractional polynomials. *Journal of the Royal Statistical Society, Series A*, submitted.

Sauerbrei, W., P. Royston, H. Bojar, C. Schmoor, M. Schumacher, and the German Breast Cancer Study Group (GBSG). 1998. Modelling the effects of standard prognostic factors in node positive breast cancer. Submitted.

Smith, J. M., C. J. Dore, A. Charlett, and J. D. Lewis. 1992. A randomized trial of Biofilm dressing for venous leg ulcers. *Phlebology* 7: 108–113.

sg82 Fractional polynomials for st data

Patrick Royston, Imperial College School of Medicine, UK, proyston@rpms.ac.uk

Introduction

Fractional polynomials were introduced by Royston and Altman (1994) and implemented in Stata 5.0 (see [R] **fracpoly**). The Stata command `fracpoly` constructs models with a fractional polynomial transformation of a single covariate. The new command `mfracpol` (Royston and Ambler 1998), allows models with fractional polynomials applied to an arbitrary number of covariates. However, neither `fracpoly` nor `mfracpol` may be used with Cox regression and `st` data, though both are compatible with the `cox` command. The present insert remedies that deficiency by providing two new programs, `stfracp` and `stmfracp`, which may be seen as convenience tools.

Syntax

`stfracp` *varlist* [`if` *exp*] [`in` *range*] `,` *stcox_options fracpoly_options*

`stmfracp` *varlist* [`if` *exp*] [`in` *range*] `,` *stcox_options mfracpol_options*

Both `stfracp` and `stmfracp` require survival data and therefore `stset` must be used beforehand.

Options

stcox_options are any of the `stcox` options.

fracpoly_options are any of the `fracpoly` options.

mfracpol_options are any of the `mfracpol` options.

`stfracp` or `stmfracp` without arguments replays the most recently estimated model.

Examples

```
. stset time cens
. stfracp age, degree(1)
. stfracp age weight, degree(1)
. stmfracp age weight bmi, df(2, sex:1) select(0.05)
```

References

Royston, P. and D. G. Altman. 1994. Regression using fractional polynomials of continuous covariates: parsimonious parametric modelling (with discussion). *Applied Statistics* 43: 429–467.

Royston, P. and G. Ambler. 1998. sg81: Multivariable fractional polynomials. *Stata Technical Bulletin* 43: 24–32.

sg83 Parameter estimation for the Gumbel distribution

Manuel G. Scotto, University of Lisbon, manolo@naval05.ist.utl.pt
Aurelio Tobias, Institut Municipal d'Investigacio Medica (IMIM), Barcelona, atobias@imim.es

Introduction

It is well known that a main advantage of Stata is that it can be fully programmable in an easy way. Stata is used by researchers of all disciplines, and thanks to the *Stata Technical Bulletin* we can find specific applications for several research fields, such as Biostatistics and Epidemiology (*sbe*), Time Series and Econometrics (*sts*), Social Science and Psychometrics (*sss*), or Quality Control (*sqc*). In this insert we present a program that can be used in Environmental Statistics; namely, the estimation of the parameters of the Gumbel distribution.

The Gumbel distribution

In the analysis of extreme value data, such as wind speeds, ozone levels and river heights, the Generalized Extreme Value (GEV) distribution with cumulative density function

$$\mathrm{F}(X;\gamma,\lambda,\delta)=\exp\left\{-\left[1-\gamma\frac{X-\lambda}{\delta}\right]^{1/\gamma}\right\}\qquad\text{where } 1-\gamma\frac{X-\lambda}{\delta}>0$$

plays a fundamental role. The parameter γ is called the shape parameter and may be used to model a wide range of tail behaviors. There are three particular forms of F corresponding to $\gamma > 0$ (Frechet, Type II), $\gamma < 0$ (Weibull, Type III) and $\gamma = 0$, being interpreted as the limit as $\gamma \to 0$, widely called the Gumbel or Type I distribution with cumulative distribution function (see Gumbel 1958)

$$\mathrm{F}(x;\lambda,\delta)=\exp\left\{-\exp\left(-\frac{x-\lambda}{\delta}\right)\right\}$$

with location parameter λ varying in the real line and the scale parameter δ being nonnegative. The Gumbel distribution has been usually applied to analyze extreme levels of air pollution (Leadbetter et al. 1983), and the sea levels with particular attention to questions concerning trend and periodicity (Smith 1986).

Estimation procedure

Standard statistical methodology from parametric theory is available if our data consist of a sample $X = (X_1,\dots,X_n)$ of an independent and identically distributed sequence of Gumbel distributed random variables depending on the unknown parameters $\theta = (\lambda,\delta)$. We can estimate the parameter vector θ by maximum likelihood (MLE). For the Gumbel distribution, the log-likelihood function is defined as

$$\ell((\lambda,\delta);X)=-n\log\delta-\frac{1}{\delta}\sum_{j=1}^{n}X_i+\frac{n\lambda}{\delta}-\sum_{j=1}^{n}\exp\left\{-\frac{X_i-\lambda}{\delta}\right\}$$

Differentiating with respect to λ and δ yields the likelihood equations given by

$$\begin{aligned}\widehat{\lambda}=&-\widehat{\delta}\log\frac{1}{n}\sum_{j=1}^{n}\exp\{-X_i/\widehat{\delta}\}\\ \widehat{\delta}=&\bar{X}-\frac{\sum_{j=1}^{n}\exp\{-X_i/\widehat{\delta}\}X_i}{\sum_{j=1}^{n}\exp\{-X_i/\widehat{\delta}\}}\end{aligned}$$

where $\bar{X}$ is the sample mean. Clearly, the second equation above only depends on δ, and thus can be solved through an iterative procedure such as the Newton–Raphson method. This method defines the $(i+1)$th iteration as follows:

$$\delta_{i+1}=\delta_i-\frac{f(\delta_i)}{f'(\delta_i)},\qquad i=0,1,2,\dots$$

with

$$f(\delta)=\delta-\bar{X}-\frac{\sum_{j=1}^{n}\exp\{-X_i/\delta\}X_i}{\sum_{j=1}^{n}\exp\{-X_i/\delta\}}$$

and

$$f'(\delta)=\frac{\partial}{\partial\delta}f(\delta)=1-\left[\frac{(\sum_{j=1}^{n}\exp\{-X_i/\delta\}X_i^2)(\sum_{j=1}^{n}\exp\{-X_i/\delta\})-(\sum_{j=1}^{n}\exp\{-X_i/\delta\}X_i)^2}{(\delta\sum_{j=1}^{n}\exp\{-X_i/\delta\})^2}\right]$$

Before starting the iterative process, it is necessary to devise a criteria for stopping the iterative search for δ. A practical criterion dictates stopping as soon as further iterations fail to change the parameter values significantly; that is, given a small number ε, we accept $\widehat{\delta}_{i+1}$ as the MLE solution provided $|\widehat{\delta}_{i+1}-\widehat{\delta}_i|\le\varepsilon$. Finally, all optimization methods require that one supply an initial value for the parameter. In this case we use as initial value $\sqrt{6s^2}/\pi$ with s^2 the sample variance. Once we obtain $\widehat{\delta}$ as estimate of δ, we can substitute it in the equation for $\widehat{\lambda}$.

Syntax

The syntax for the `gumbel` command is as follows:

`gumbel` *varname* [`if` *exp*] [`in` *range*] [, `iterate(`*#*`)` `ltol(`*#*`)`]

Options

`iterate(`*#*`)` specifies the maximum number of iterations allowed in estimating the parameters; `iterate(100)` is the default.

`ltol(`*#*`)` specifies the convergence criterion, ε, between iterations; `ltol(1e-5)` is the default.

Example

We fitted the Gumbel distribution to the annual maximum sea levels in Venice during the period 1981–82 (data from Smith 1986).

```
. describe

Contains data from smith.dta
  obs:            51
 vars:             1
 size:           408
-------------------------------------------------------------------------------
   1. seal        float  %9.0g                annual maximum sea levels
-------------------------------------------------------------------------------

. summarize seal, detail
                  annual maximum sea levels
-------------------------------------------------------------
      Percentiles      Smallest
 1%        71.03          71.03
 5%        80.24          76.86
10%        84.19          80.24       Obs                  51
25%        93.33          82.81       Sum of Wgt.          51

50%       101.54                      Mean           104.8722
                        Largest       Std. Dev.      18.45836
75%       115.02          138.2
90%       126.92         139.08       Variance       340.7111
95%       139.08         143.59       Skewness       1.200928
99%       173.57         173.57       Kurtosis       5.491322
-------------------------------------------------------------
```

Graphically these annual maximum sea levels have the following sample distribution:

```
. kdensity seal
```

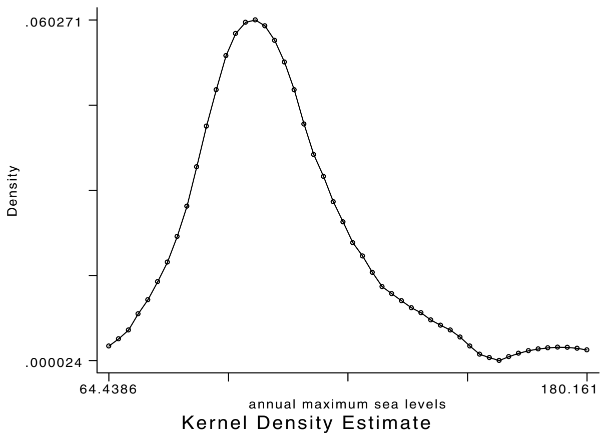


Figure 1. Kernel density estimate of annual maximum sea levels in Venice, for the period 1981–82.

Smith fitted the Gumbel distribution, obtaining $\widehat{\lambda} = 96.8$ and $\widehat{\delta} = 14.5$ for λ and δ. Using the `gumbel` command we obtained

```
. gumbel seal
Variable |  Delta         Lambda        Num.Iterations
---------+--------------------------------------------
seal     |  14.513113     96.690773     3
```

Saved results

gumbel saves the following results in the S_# macros:

S_E_la	estimation of λ parameter
S_E_de	estimation of δ parameter
S_E_it	number of iterations to convergence
S_E_ltol	convergence criterion
S_E_nobs	number of observations

References

Gumbel, E. J. 1958. *Statistics of Extremes*. New York: Columbia University Press.

Leadbetter, M. R., G. Lindgren, and H. Rootzen. 1983. *Extremes and Related Properties of Random Sequences and Processes*. Berlin: Springer.

Smith, R. L. 1986. Extreme value theory based on the r largest annual events. *Journal of Hydrology* 86: 27–43.

sg84 Concordance correlation coefficient

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 for agreement on a continuous measure obtained by two persons or methods. The Lin coefficient combines measures of both precision and accuracy to determine whether the observed data deviate significantly from the line of perfect concordance (i.e., the line at 45°). Lin's coefficient increases in value as a function of the nearness of the data's reduced major axis to the line of perfect concordance (the accuracy of the data) and of the tightness of the data about its reduced major axis (the precision of the data). The Pearson correlation coefficient, ρ, the bias-correction factor, C_b, and the equation of the reduced major axis are reported to show these components. Note that the concordance correlation coefficient, ρ_c, can be expressed as the product of ρ, the measure of precision, and C_b, the measure of accuracy. The optional concordance graph plots the observed data, the reduced major axis of the data, and the line of perfect concordance as a graphical display of the observed concordance of the measures.

concord also provides statistics and optional graphics for Bland and Altman's limits-of-agreement, LOA, procedure (1986). The LOA, a data-scale assessment of the degree of agreement, is a complementary approach to the relationship-scale approach of Lin.

The user provides the pairs of measurements for a single property as observations in variables *var1* and *var2*. Frequency weights may be specified and used. Missing values (if any) are deleted in a casewise manner.

Syntax

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

Options

graph(ccc) requests a graphical display of the data, the line of perfect concordance and the reduced major axis of the data. The reduced major axis or SD line goes through the intersection of the means and has slope given by the sign of Pearson's *r* and the ratio of the standard deviations. The SD line serves as a summary of the center of the data. (For more information on the SD line, see Freedman et al., 1998, Ch. 8.)

graph(loa) requests a graphical display of the limits-of-agreement, the mean difference, and the data presented as paired differences plotted against pairwise means. A normal plot for the differences is also displayed.

summary requests summary statistics.

by(*by_var*) produces separate results for groups of observations defined by *by_var*.

`level(#)` specifies the confidence level, in percent, for confidence intervals. The default is `level(95)` or as set by `set level`.

graph_options are those allowed with `graph, twoway`. Setting `t1title(.)` blanks out the default `t1title`. The default graph options for `graph(ccc)` are `connect(.l) symbol(o.) pen(22)` for the data points and SD line, respectively, along with default titles and labels. The default graph options for `graph(loa)` are `connect(lll.) symbol(...o) pen(3532)` for the confidence interval limits, the mean difference, and the data points, respectively, along with default titles and labels. (The user is not allowed to modify the graph options for the normal probability plot.)

Explanation

A frequent problem in science is the comparison of two or more sets of measurements of what is supposedly the same property. These sets of measurements may be produced by different people, different instruments, and so forth. It is then interesting to know how similar or different are the measurements: are they close enough to be taken as interchangeable, or different enough to signal a measurement problem? Taking two sets as two paired variables Y_1 and Y_2, the statistical question at issue is whether $Y_1 = Y_2$. Given a scatter plot with equal scales for Y_1 and Y_2, the ideal case is clearly that data points would all lie on a line through the origin at 45°.

The standard Pearson correlation coefficient r may be used for comparison of two sets of measurements. However, the definition of correlation implies that perfect correlation $r = \pm 1$ arises if the data for two variables Y_1 and Y_2 all lie on some straight line $Y_1 = a + bY_2$. This is a more general ideal, insofar as the intercept a need not equal 0 and the gradient b need not equal 1, as implied by $Y_1 = Y_2$. It is perfectly possible to achieve a very good correlation in which the corresponding straight line has $a \neq 0$ or $b \neq 1$, but these cases signal measurement problems just as surely as does a failure to fit a straight line altogether.

To help distinguish these different problems, Lin (1989) proposed a *concordance correlation coefficient*, which is implemented in this insert. In essence, it compares the agreement between two paired sets of measurements by measuring variation from a 45° line through the origin. The remainder of this section paraphrases key parts of Lin's paper, which also includes details on Monte Carlo simulations of estimators and examples based on some medical data.

Suppose that pairs of samples $(Y_{i1}, Y_{i2}),\ i = 1, \dots, n$, are independently selected from a bivariate population with means μ_1 and μ_2, covariance matrix

$$\begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$$

and Pearson correlation coefficient $\rho = \sigma_{12}/\sigma_1\sigma_2$. The degree of concordance between Y_1 and Y_2 can be characterized by the expectation of the squared difference

$$\begin{aligned} E[(Y_1 - Y_2)^2] &= (\mu_1 - \mu_2)^2 + \sigma_1^2 + \sigma_2^2 - 2\sigma_{12} \\ &= (\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2 - 2(1-\rho)\sigma_1\sigma_2, \end{aligned}$$

which is also twice the expected squared perpendicular deviation from the 45° line. If Y_1 and Y_2 were in perfect agreement, this expectation would be 0. To express concordance in relative terms, the concordance correlation coefficient was proposed by Lin (1989) as

$$\begin{aligned} \rho_c &= 1 - \frac{E[(Y_1 - Y_2)^2]}{\sigma_1^2 + \sigma_2^2 + (\mu_1 - \mu_2)^2} \\ &= 1 - \frac{\text{Expected squared perpendicular deviation from } 45^\circ \text{ line}}{\text{Expected squared perpendicular deviation from } 45^\circ \text{ line if } Y_1,\ Y_2 \text{ uncorrelated}} \\ &= \frac{2\sigma_{12}}{\sigma_1^2 + \sigma_2^2 + (\mu_1 - \mu_2)^2} \\ &= \rho C_b, \end{aligned}$$

where

$$\begin{aligned} C_b &= 2/(v + 1/v + u^2), \\ v &= \sigma_1/\sigma_2 = \text{scale shift, and} \\ u &= (\mu_1 - \mu_2)/\sqrt{\sigma_1\sigma_2} = \text{location shift relative to the scale.} \end{aligned}$$

C_b is a bias correction factor that measures how far the best-fit line deviates from the 45° line. The maximum value of $C_b = 1$ means no deviation from the line: the further C_b is from 1, the greater the deviation from the line. In other words, C_b is a measure of accuracy, whereas ρ is a measure of precision.

The concordance correlation coefficient, ρ_c, which combines ρ and C_b, possesses the following properties and characteristics:

1. $-1 \leq -|\rho| \leq \rho_c \leq |\rho| \leq 1$.
2. $\rho_c = 0$ if and only if $\rho = 0$.
3. $\rho_c = \rho$ if and only if $\mu_1 = \mu_2$ and $\sigma_1 = \sigma_2$.
4. $\rho_c = \pm 1$ if and only if
   (a) $(\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2 + 2\sigma_1\sigma_2(1 \pm \rho) = 0$, or equivalently,
   (b) $\rho = \pm 1,\ \sigma_1 = \sigma_2$ and $\mu_1 = \mu_2$, or equivalently,
   (c) each pair of readings is in perfect agreement (for the case $\rho_c = 1$)
   or in perfect reversed agreement (for the case $\rho_c = -1$).

For n independent pairs of sample data, we estimate the concordance correlation by using the sample counterparts of the population parameters; i.e., let

$$\widehat{\rho}_c = \frac{2S_{12}}{S_1^2 + S_2^2 + (\overline{Y}_1 - \overline{Y}_2)^2}$$

where

$$\overline{Y}_j = \frac{1}{n}\sum_{i=1}^{n} Y_{ij}, \qquad S_j^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_{ij} - \overline{Y}_j)^2, \quad j = 1,2, \qquad \text{and} \quad S_{12} = \frac{1}{n}\sum_{i=1}^{n}(Y_{i1} - \overline{Y}_1)(Y_{i2} - \overline{Y}_2)$$

If the data for Y_1 and Y_2 come from a bivariate normal (Gaussian) distribution, it can be shown that $\widehat{\rho}_c$ has an asymptotic normal distribution with mean ρ_c and variance

$$\sigma^2_{\widehat{\rho}_c} = \frac{1}{n-2}\left[(1-\rho^2)\rho_c^2(1-\rho_c^2)/\rho^2 \ + \ 4\rho_c^3(1-\rho_c)u^2/\rho \ - \ 2\rho_c^4 u^4/\rho^2\right]$$

which can be used to obtain (symmetric) confidence intervals for ρ_c. However, for small samples a better approximation to normality is gained by an inverse hyperbolic tangent transformation (R. A. Fisher's z-transformation), yielding

$$\widehat{z} = \text{arc tanh}(\widehat{\rho}_c) = \frac{1}{2}\ln\frac{1+\widehat{\rho}_c}{1-\widehat{\rho}_c}$$

Its variance

$$\sigma^2_{\widehat{z}} = \frac{\sigma^2_{\widehat{\rho}_c}}{(1-\rho_c^2)^2}$$

can be used to obtain confidence intervals for $\widehat{z}$. When transformed back to the ρ_c scale, such confidence intervals are realistically asymmetric and necessarily within the open interval $(-1, 1)$.

Limits of agreement

Bland and Altman (1986) considered the problem of comparing two methods of clinical measurement and rejected standard correlation methods. Their principal objection was that correlation methods change the question from "*how well do the methods agree?*" to "*how well are they related?*" These authors correctly point out that the Pearson correlation, a measure of strength of linear relationship, is unaffected by changes in location or scale. They argue that insensitivity to such changes is inappropriate when assessing agreement, explaining, for example, that one cannot blindly mix or substitute measures when one measure is uniformly twice the other measure. Likewise, they point out logical flaws in the use of regression, t testing, and intraclass correlation.

To avoid these problems, Bland and Altman proposed a simple data-scale assessment of agreement based on the paired differences between the readings of the two clinical measurements. Given paired sets of data $(Y_{i1}, Y_{i2}),\ i = 1, \ldots, n$, compute the paired differences, $d_i = Y_{i1} - Y_{i2}$, and their mean, $\bar{d}$, and standard deviation, s_d. Then, provided the differences follow a normal distribution and are independent of the magnitude of the measurement, compute the confidence interval for the difference between single measurements on the same subject by the two methods as $(\bar{d} - Z_{1-\alpha/2}s_d, \bar{d} + Z_{1-\alpha/2}s_d)$, where $Z_{1-\alpha/2}$ is the $1 - \alpha/2$ critical point of the normal distribution. Bland and Altman label this confidence interval as the *limits of agreement*. They suggest that the normality assumption should be valid provided that the magnitude of the difference is independent of the magnitude of the individual measures. They propose 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. Their procedure and suggested plots are implemented in this insert.

Note that although Bland and Altman (1986) rejected standard correlation methods for the assessment of agreement between clinical measurements, this rejection did not apply to Lin's concordance correlation coefficient, a statistic which was first proposed some three years later in 1989. Lin's coefficient addresses the deficiencies of correlation methods raised by Bland and Altman and should be considered *complementary* to their limits-of-agreement procedure. The limits-of-agreement procedure assesses agreement on the data scale, whereas the concordance correlation coefficient addresses agreement on the relationship scale. Neither approach alone gives a complete analysis of the quality of agreement but, in combination, they reveal much about a comparison. The limits of agreement indicate how close the two methods fall on the data scale but provide only weak indication of deviations from perfect concordance or of precision relative to range. In contrast, the concordance correlation coefficient gives no assessment on the data scale but provides strong indications on a standardized relationship scale about precision and deviation from perfect concordance.

Example

Bland and Altman (1986) provide data assessing peak expiratory flow rate (PEFR), measured in liters per minute by two competing meters (Wright and Mini), for 17 subjects. The following is the output from `concord`.

```
. concord Wright Mini, summary

Concordance correlation coefficient (Lin, 1989)

 rho_c   se(rho_c)   obs    [   95% CI   ]  p-value     CI type
---------------------------------------------------------------
 0.943     0.029      17     0.887  0.999    0.000   asymptotic
                             0.850  0.979    0.000  z-transform

Pearson's r =  0.943  Pr(r = 0) = 0.000  C_b = rho_c/r =  0.999
reduced major axis:   slope =     1.028   intercept =   -14.908

        difference              95% limits of agreement
   average    std. dev.          (Bland & Altman, 1986)
---------------------------------------------------------------
     2.118      38.765             -73.861     78.096

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
  Wright |      17    450.3529   116.3126        178        656
    Mini |      17    452.4706   113.1151        259        658
```

Here, $\rho_c = 0.943$ (ρ_c is denoted `rho_c` in the printout) indicates significant concordance and has a confidence interval that is narrow and near 1.0. The Pearson r is identical (to three digits) to ρ_c and the bias correction factor, C_b, (denoted `C_b` in the printout) is essentially 1, indicating that the reduced major axis of the data effectively falls on the line of perfect concordance (a slope of 1 and an intercept of 0). The printout reports that the actual reduced major axis of the data has a slope of 1.03 and an intercept of -14.9 (since the centroid of these data is far from zero, 15 units of deviation at the intercept is trivial). Therefore these data show strong concordance with primarily random deviations from perfection. The limits-of-agreement analysis indicates that one should expect the two measures to provide flow rates that are consistently within ± 80 liters per minute of each other.

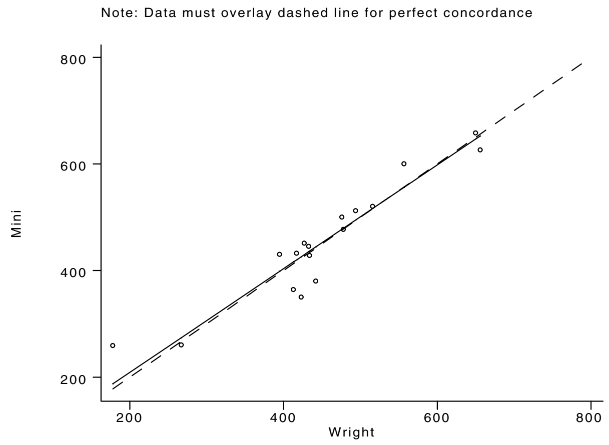


Figure 1. The `ccc` graph for Bland and Altman data.

The optional `ccc` graph of the data, shown as Figure 1, provides visual confirmation of this interpretation. The *precision* of the data is apparent, since all data points fall relatively near the reduced major axis (as was suggested by the high Pearson correlation), and the *accuracy* is evident, since the reduced major axis falls essentially on the 45° line (as was suggested by the bias correction factor). Lin's ρ_c, which combines measures of both precision and accuracy, clearly indicates the strong concordance shown in the plot.

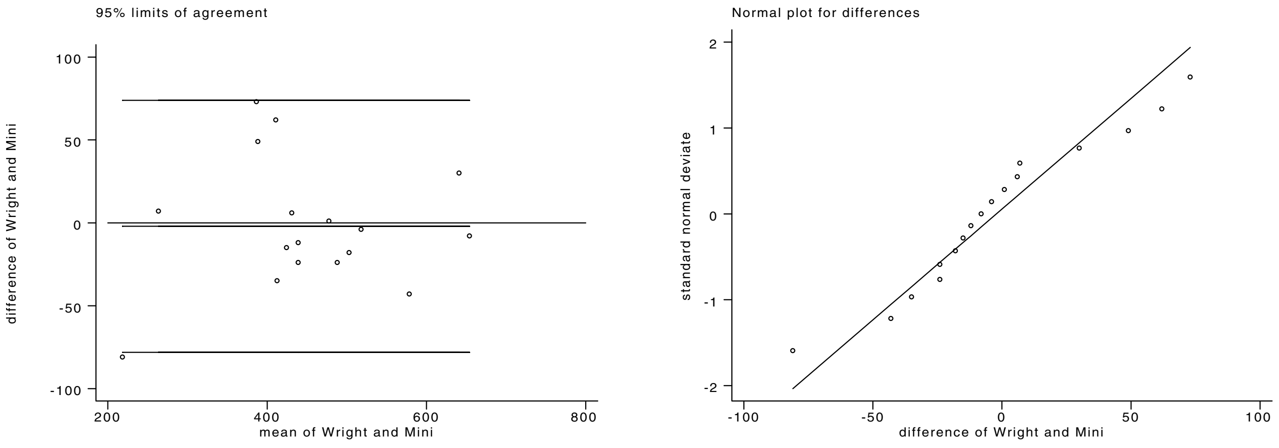


Figure 2. The `loa` graphs for Bland and Altman data

If instead, the optional `loa` graphs of the data are requested, a visual display of the limits of agreement are provided laid over a plot of paired differences against pairwise means (Figure 2, left panel). This plot shows no tendency for the magnitude of the difference to be dependent on the magnitude of the individual measures. Likewise, the normal probability plot for differences (Figure 2, right panel) shows no obvious nonnormality.

Saved results

If the `by()` option is not used, `concord` saves in the system `S_#` macros:

`S_1`	number of observations compared	`S_7`	upper CI limit (z-transform)
`S_2`	concordance correlation coefficient, $\hat{\rho}_c$	`S_8`	bias-correction factor, C_b
`S_3`	standard error of $\hat{\rho}_c$, $\sigma_{\hat{\rho}_c}$	`S_9`	mean difference
`S_4`	lower CI limit (asymptotic)	`S_10`	standard deviation of mean difference
`S_5`	upper CI limit (asymptotic)	`S_11`	lower limit-of-agreement value
`S_6`	lower CI limit (z-transform)	`S_12`	upper limit-of-agreement value

References

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

Freedman, D., R. Pisani, and R. Purves. 1998. *Statistics.* New York: Norton.

Lin, L. I-K. 1989. A concordance correlation coefficient to evaluate reproducibility. *Biometrics* 45: 255–68.

zz8 Cumulative Index for STB-37–STB-42

[an] Announcements

STB-37	2	an64	STB-31–STB-36 available in bound format
STB-37	2	an65	Memorium for Stewart West

[stata] Official Updates

STB-39	2	stata47	lookup now indexes on-line FAQs
STB-39	4	stata48	Updated reshape
STB-40	2	stata49	Interrater agreement
STB-40	8	stata50	Changes to ttest and sdtest
STB-41	2	stata51	Exact McNemar test added to mcc and mcci commands
STB-42	2	stata52	Origin/noOrigin option added to sts graph command

[dm] Data Management

STB-37	2	dm44	Sequences of integers
STB-37	4	dm45	Changing string variables to numeric
STB-37	6	dm46	Enhancement to the sample command
STB-37	7	dm47	Verifying value label mappings
STB-38	2	dm48	An enhancement of reshape
STB-39	17	dm49	Some new matrix commands
STB-40	9	dm50	Defining variables and recording their definitions
STB-40	10	dm51	Defining and recording variable orderings
STB-40	12	dm52	Executing a command on a subset of the data
STB-41	2	dm53	Detection and deletion of duplicate observations
STB-42	3	dm54	Capturing comments from data dictionaries

[gr] Graphics

STB-41	4	gr16.2	Corrections to condraw.ado
STB-40	12	gr24.1	Easier bar charts: correction
STB-40	12	gr25.1	Spike plots for histograms, rootograms and time series plots: update
STB-37	9	gr26	Bin smoothing and summary on scatter plots
STB-41	4	gr27	An adaptive variable span running line smoother
STB-42	4	gr28	A graphical procedure to test equality of variances

[ip] Instruction on Programming

STB-37	12	ip17	While loops from the command line
STB-37	17	ip18	A command for randomly resampling a dataset
STB-39	20	ip19	Using expressions in Stata commands
STB-40	13	ip20	Checking for sufficient memory to add variables
STB-40	13	ip21	Storing commands in the keyboard buffer (Windows and Macintosh only)
STB-40	13	ip22	Parsing options with embedded parentheses
STB-41	7	ip23	Expansion and display of if expressions
STB-41	8	ip24	Timing portions of a program

[sbe] Biostatistics & Epidemiology

STB-40	16	sbe13.3	Correction to age-specific reference intervals (“normal ranges”)
STB-38	4	sbe15	Age-specific reference intervals for normally distributed data
STB-38	9	sbe16	Meta-analysis
STB-42	6	sbe16.1	New syntax and output for the meta-analysis command
STB-39	22	sbe17	Discrete time proportional hazards regression
STB-40	16	sbe18	Sample size calculations for clinical trials with repeated measures data
STB-41	9	sbe19	Tests for publication bias in meta-analysis
STB-41	15	sbe20	Assessing heterogeneity in meta-analysis: the Galbraith plot
STB-42	8	sbe21	Adjusted population attributable fractions from logistic regression
STB-42	13	sbe22	Cumulative meta-analysis
STB-42	16	sbe23	Meta-analysis regression

[sed] Exploratory Data Analysis

STB-41	17	sed9.1	Pointwise confidence intervals for running

[sg] General Statistics

STB-42	22	sg42.2	Displaying predicted probabilities from probit or logit regression
STB-41	23	sg44.1	Correction to random number generators
STB-41	23	sg53.2	Stata-like commands for complementary log-log regression
STB-38	14	sg70	Interquantile and simultaneous-quantile regression
STB-38	22	sg71	Routines to maximize a function
STB-39	32	sg72	Newey–West standard errors for probit, logit, and poisson models
STB-40	18	sg73	Table making programs
STB-40	23	sg74	Symmetry and marginal homogeneity test / Transmission-Disequilibrium Test (TDT)
STB-41	23	sg75	Geometric means and confidence intervals

STB-42	24	sg76	An approximate likelihood-ratio test for ordinal response models
STB-42	28	sg77	Regression analysis with multiplicative heteroscedasticity
STB-42	32	sg78	Simple and multiple correspondence analysis in Stata
STB-42	38	sg79	Generalized additive models
STB-42	43	sg80	Indirect standardization

[smv] Multivariate Analysis

STB-38	26	smv3.2	Enhancements to discriminant analysis
STB-37	22	smv7	Inference on principal components

[snp] Nonparametric Methods

STB-38	27	snp13	Nonparametric assessment of multimodality for univariate data
STB-42	47	snp14	A two-sample multivariate nonparametric test

[ssa] Survival Analysis

STB-37	24	ssa9	Cox proportional hazards model with the exact calculation for ties
STB-37	26	ssa9.1	Survival analysis subroutine for programmers
STB-40	27	ssa10	Analysis of follow–up studies with Stata 5.0
STB-41	25	ssa10.1	Update to analysis of follow–up studies with Stata 5.0
STB-41	25	ssa11	Survival analysis with time-varying covariates

[svy] Survey Sample

STB-40	39	svy6	Versions of mlogit, ologit, and oprobit for survey data

[sxd] Experimental Design

STB-41	43	sxd1	Random allocation of treatments in blocks

[tt] Teaching

STB-41	46	tt7	Random walk tutorial

[zz] Not elsewhere classified

STB-37	30	zz7	Cumulative Index for STB-31–STB-36

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:	andres.gil@usa.net
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:	0931 62 7255
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/mhendekli/unidost.htm
Countries served:	Turkey