STATA TECHNICAL BULLETIN

March 1997
STB-36

A publication to promote communication among Stata users

Contents of this issue

gr16.1 Convex hull plots

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

Syntax

The syntax for the `chplot` command is

`chplot` *yvar xvar* [`if` *exp*] [`in` *range*] [`, hull(`*#*`) hv(`*hullvar*`) movie` *graph_options*]

Note: `chplot` requires the `conhull` and `condraw` programs which must be installed from the gr16 directory of the STB 23 disk (January 1995).

Options

`hull(`*#*`)` specifies the convex hull required. The default is 1, the outermost hull.

`hv(`*hullvar*`)` specifies a variable to hold information about the hulls and is only needed if `_hull1` is a variable in the data set.

`movie` specifies that the final graph (for `hull` $>$ 1) is to be preceded by a movie of the hulls up to `hull`. Thus, `hull(2) movie` means that the final graph includes hull 2 and that hulls 1 and 2 will be shown separately beforehand.

graph_options are any options allowed with `graph, twoway`. See help on `graph`. Defaults are `c(.l) s(oi) t1(" ")` with `l1` and `b1` indicating y and x variables as usual.

Explanation

The convex hull programs `conhull` and `condraw` of Gray and McGuire (1995) are a very useful adjunct to the graphics facilities of Stata. The program here, `chplot`, is a supplement to those programs designed to streamline two frequent needs, namely to get either a quick or a presentable graph of a particular convex hull on a standard scatter plot. `chplot` in fact calls `conhull` and `condraw`, which do all the hard work, so all that is offered is a different interface that will ease some tasks.

The use of `chplot` varies from the use of `conhull` and `condraw` in the following details:

1. `conhull` and `condraw` require the user to specify the x variable before the y variable, contrary to the convention of `graph, twoway` that will be familiar to experienced Stata users. `chplot` uses the same convention as `graph, twoway`; that is, the y variable before the x variable.
2. `conhull` and `condraw` leave behind extra variables and extra observations specifying the hulls and allowing graphical closure of the hulls. These extras have to be dropped from the data set each time other convex hulls are drawn in the same Stata session. `chplot` handles these details by using temporary variables and leaving the data unchanged. A partial exception is that `condraw` uses a stub, by default `hv`, for a set of variables, e.g. `hv1`, `hv2`, ... I do not know a way to specify a temporary stub in Stata; one problem is that all temporary variables have names 8 characters long underneath the names that the programmer employs. I have left `condraw` and `conhull` unchanged and merely used a stub `_hull`. A guess at user practices is that the varname `_hull1` is less likely in user data sets than the varname `hv1`. If `_hull1` is in the data set when `chplot` is invoked, an error message is issued and the `hv` option must be used to specify another new variable. However, `_hull1` is not left behind by `chplot`, so that it is in effect used as a temporary variable.
3. `chplot` allows the user to specify `if` and `in`.
4. `chplot` allows the user to specify the `graph, twoway` options for a presentable graph at the same time as invoking `condraw` and `conhull`.
5. `chplot` has a simpler syntax for the tasks described, especially for the casual user.
6. `chplot` does not allow separate hulls to be drawn for each level of a categorical variable.
7. `chplot` does not allow two or more hulls to be drawn on the same graph.
8. `chplot` does not allow the user to examine the subsets of the data defined by each convex hull.

Items 1–5 are suggested to be advantages of `chplot`, while items 6–8 are limitations.

Examples

We will work with data on 158 glacial cirques from the English Lake District (Evans and Cox 1995), found in the accompanying file `cirques.dta`. Glacial cirques are hollows excavated by glaciers that are open downstream, bounded upstream by the crest of a steep slope, and accurate in plan around a more gently sloping floor. More informally, they are sometimes described as "armchair-shaped". They are common in mountain areas that have or have had glaciers present.

Whether cirque shape changes with size is one question of interest to geomorphologists. Given data on cirque length and width, the outermost convex hull is simply displayed by

```
. chplot length width
```

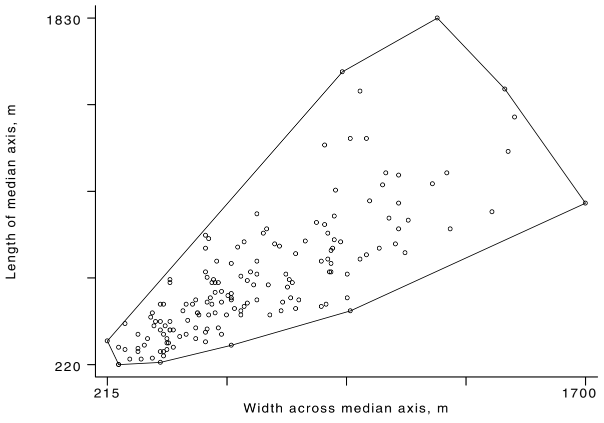


Figure 1. Convex hull plot: natural scale.

The next step might be to use logarithmic scales and to add some sensible labels by

```
. chplot length width, xlog ylog xlabel(200,500,1000,2000) ylabel(200,500,1000,2000)
```

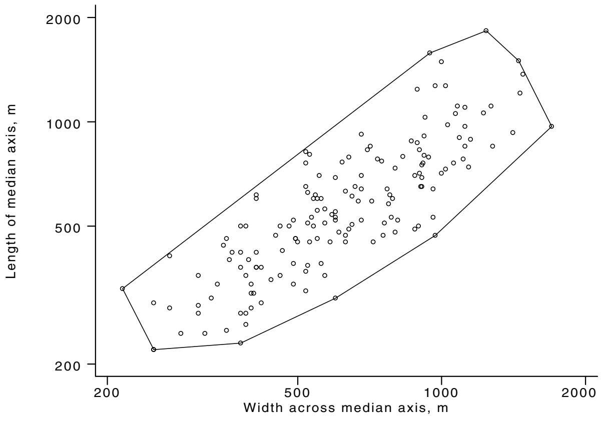


Figure 2. Convex hull plot: logarithmic scale.

Here we are exploiting the congenial fact that, for this example, taking the convex hull and logarithmic transformation of both variables are commutative; the result is the same whichever you do first. Note, however, that this is necessarily true only for affine transformations and must be checked otherwise.

References

Evans, I. S. and N. J. Cox. 1995. The form of glacial cirques in the English Lake District, Cumbria. *Zeitschrift für Geomorphologie* 39: 175–202.

Gray, J. P. and T. McGuire. 1995. gr16: Convex hull programs. *Stata Technical Bulletin* 23: 11–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 59–66.

gr24	Easier bar charts

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

Syntax

The syntax for the `vbar` command is

> `vbar` *varlist* [*weight*] [`if` *exp*] [`in` *range*] [`,` `sort(`*sortvar*`)` `rsort(`*sortvar*`)`
> `laby(`*labelvar*`)` `by(`*byvar*`)` `category` `sepbars` `perc` `prop` `pbyg` `pbyv` *bar_options*]

Options

`sort(`*sortvar*`)` means that bars are to be plotted in ascending order of *sortvar*, highest values on the right. If weights are used and *sortvar* is numeric, then the order is calculated using the weights. `sort` may not be combined with `rsort` or `by`.

`rsort(`*sortvar*`)` means that bars are to be plotted in descending order of *sortvar*, highest values on the left. If weights are used and *sortvar* is numeric, then the order is calculated using the weights. `rsort` may not be combined with `sort` or `by`.

`laby(`*labelvar*`)` means that bars are to be plotted with labels from *labelvar*, normally but not necessarily a string variable containing text. `laby` may not be combined with `by`.

`by(`*byvar*`)` means what it does with `graph, bar`: bars are plotted in groups according to the values of *byvar*. Note that `by` may not be combined with `sort`, `rsort`, or `laby`.

`category` applies when *varlist* contains a single categorical variable and it is desired to show the count or frequency in each category. If the number of categories is 6 or fewer, a set of temporary indicator variables will be generated, and the bars will touch and have (by default) different colors and shadings. If the number of categories is 7 or more, the categories will be plotted as separate bars for a single variable, with the same color and shading, so long as `by` has not been invoked. However, if separate bars are preferred when the number of categories is 6 or fewer, use the `sepbars` option in addition.

`sepbars` is explained just above and is restricted to overriding a default behavior with the `category` option.

`perc` means that results are to be reported as percents. The default is percents of the grand total of all the variables in the *varlist*. See also `pbyg` and `pbyv`. `perc` and `prop` are mutually exclusive.

`prop` means that results are to be reported as proportions or fractions between 0 and 1. The default is proportions of the grand total of all the variables in the *varlist*. See also `pbyg` and `pbyv`. `prop` and `perc` are mutually exclusive.

`pbyg` means that percents or proportions (one must be specified) are of the total of all the values in each group, defined by one value of the `by()` variable. `pbyg` and `pbyv` are mutually exclusive.

`pbyv` means that percents or proportions (one must be specified) are of the total of all the values for each variable in the *varlist*. `pbyv` and `pbyg` are mutually exclusive.

bar_options are other options allowed with `graph, bar`. See [R] **graph** or on-line help for `graph`, including details on `means`, `stack`, `shading()`, and `alt`.

Explanation

Simple bar charts can be surprisingly awkward in Stata. The basic reason for this is that `graph, bar` has a built-in tendency to add up whatever is fed to it. This is fine so long as what you want plotted are sums of values in their original units, or means, which are one step away with the `means` option. If you want something else, some preprocessing is required, which makes some basic tasks rather complicated, especially for users new to Stata.

I have written `vbar` as a way of automating the most common kinds of preprocessing. `vbar` is intended to be just `graph, bar` with some added bells and whistles.

First, however, I will commend `hist`. If you find `graph, bar` frustrating, check out `hist`, which gives a histogram for categorical variables. `hist` may be the answer to your question.

There are three main problems that `vbar` is designed to tackle. They sometimes arise in combination, especially the first and the second.

Problem 1: Percents and proportions

Suppose you want results plotted in percents (sum 100) or proportions (sum 1). It is necessary to transform your variables so that their values do indeed add up to the appropriate sum, 100 or 1. In STB-14, Felicia Knaul (1993) asked how to create stacked bar charts of percents. The answer to her question was six command lines creating the percents before they are fed to **graph, bar**. With **vbar**, the answer to her problem would be

```
. vbar P*, stack by(age) title("Work and school status by age") perc pbyg
```

where the options **perc** and **pbyg** call for percents and for those percents to be calculated for each group (p by g).

Let us look at another similar example, especially for those without access to the insert by Knaul. Smith (1984) studied the geography of crime in an area of Birmingham, England. She asked 531 residents in an ethnically mixed area whether they had taken any action to avoid victimization. Her table can be read into Stata and a bar chart drawn using **vbar** (see Figure 1).

```
. input int (yes no) str8 race
1. 41  51   "W.Indian"
2. 129 53   "Asian"
3. 182 75   "white"
4. end
. vbar yes no, by(race) ylabel
```

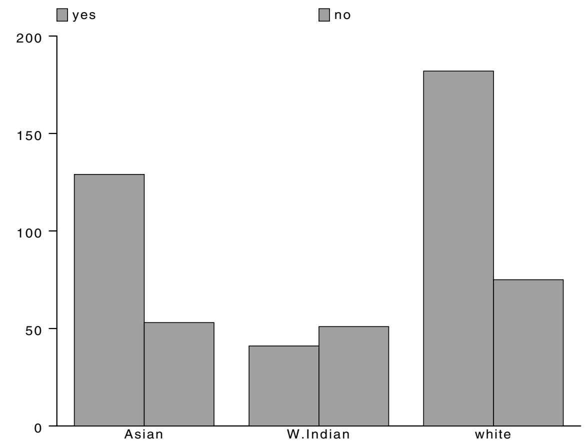


Figure 1. vbar chart.

This could also have been done by

```
. sort race
. graph yes no, bar by(race) ylabel
```

However, where **vbar** excels is in the next step, standardizing so that we look at percents (see Figure 2):

```
. vbar yes no, by(race) ylabel perc pbyg l2(percent)
```

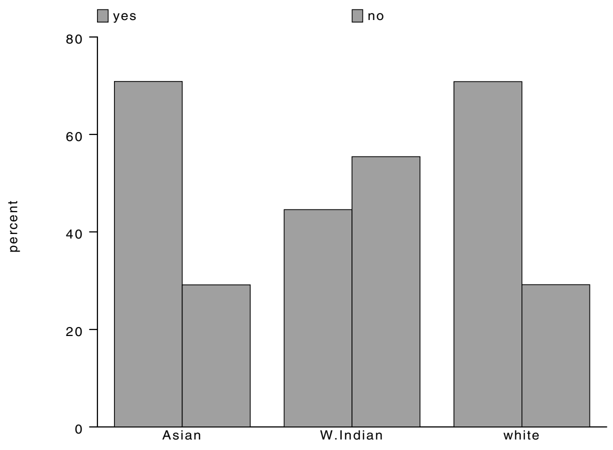


Figure 2. vbar chart with percents.

The option `stack` could have been added with both `graph, bar`, and `vbar`.

Another option allowed with `vbar` is `pbyv` (percent or proportion for each variable). If `perc` or `prop` (which calls for proportions) is called without `pbyv` or `pbyg`, the default is that they are calculated with reference to the total of all the variables in the *varlist*, which is naturally equivalent to `pbyv` if there is just one variable.

Problem 2: Categorical variables

Suppose you have a categorical variable coded numerically with more or less arbitrary codes. `graph, bar` would treat each code literally, that is to say, numerically, so that if a variable were coded 1, 2, and so forth, all the 1's would be added up, all the 2's added up, and so forth, so that each bar would represent the frequency of the value, multiplied by the arbitrary code. In the automobile data distributed with Stata, `rep78` is such a categorical variable with codes 1 through 5.

`hist` is an answer to this problem and plots frequencies (as, in a cruder way, does `tabulate, plot`). Yet with `hist`, the bars touch, which you may not want, perhaps because you are sensitive about giving the impression of an underlying continuous scale. Note also that `hist, by()` is not similar to `graph, bar by()`.

A solution which is rather Stata-ish is to use `tabulate, gen()` to generate a set of indicator variables (values 1 or 0) that when added up by `graph, bar` will give the desired frequencies. Experienced Stata users will do this as a conditioned reflex, but to those learning Stata it can seem a trifle arcane. `vbar` automates this indicator variable line of attack when called with the `category` option. The attack fails if the number of categories exceeds 6, because `graph, bar` will not allow more than 6 variables. There is another line of attack that is then tried, which has the consequence that it plots separate bars, so long as `by` has not been used as an option. Separate bars may or may not be what is preferred. If there are 6 or fewer categories, invoking the `sepbars` option will override the default. In the automobile example, the `category` and `sepbars` options produce the following plot:

```
. vbar rep78, category sepbars ylabel
```

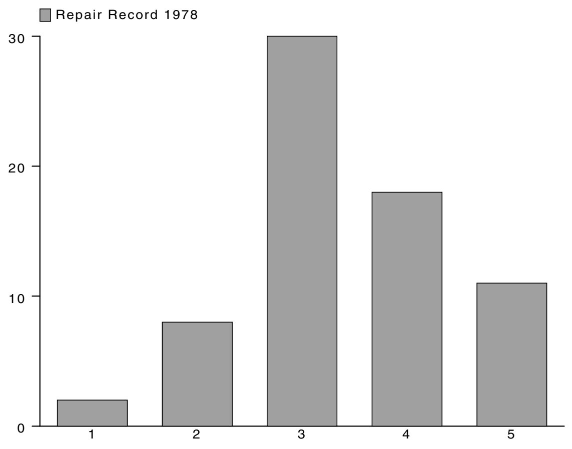


Figure 3. vbar with category option.

With the Smith data looked at earlier, the assumption was that we were reading in the counts from a summary table. What is also likely with such data is that we do not have the summary counts but the raw data in a categorical variable, say `answer`, taking values `yes` or `no`. The bar chart would then be produced by

```
. vbar answer, category by(race) ylabel perc pbyg
```

Note that `vbar` only allows one variable in the *varlist* with the category option. So

```
. vbar a, category
```

is allowed, but not

```
. vbar a b, category
```

The best reasons for this limitation are that users do not often seem to want the latter form, and that if they ask for it, they really would be better served by

```
. vbar a, by(b) category
```

or

```
. vbar b, by(a) category
```

instead, which is fine. Note that in the last example the `category` option specifies that `b` is a categorical variable. `a` is treated as one automatically, as is true of `graph, bar`. Under Stata 5.0, which `vbar` assumes, both `a` and `b` can be string variables as well as numeric, because of a change to `tabulate`.

Problem 3: Ordered, labeled bar charts

Consider the United States census data distributed with Stata. You might want a bar graph of divorces for each state, labeled with state identifiers. The full names would lead to a very messy graph, but two-letter identifiers (in lower case) would be just about tolerable. Let us assume that `st2` contains those identifiers (in lower case), such as tx (a little state somewhere in the South). We have included with this insert a file called `census1.dta` containing the variables `state` and `divorce` from `census.dta` as well as the variable `st2` with the two letter identifiers.

```
. graph div, bar by(st2)
```

requires a preceding

```
. sort st2
```

and produces an alphabetically ordered bar chart, which may be what you want. On the other hand,

```
. graph div, bar by(div)
```

requires a preceding

```
. sort div
```

and produces a numerically ordered bar chart. Incidentally, this works properly only because the 50 states have 50 unique values for `div`, with no ties. You would not always be so lucky. What if you want both the state labeling and the numerical ordering? It can be done by `graph, bar` with preprocessing, and it is done on request by `vbar`. `vbar` separates the ordering and the labeling by another variable. The cost, however, is that the values of the string variable become value labels, and so cannot be more than 8 characters long. There may well be some trick to get round this limit.

```
. vbar div, sort(div) laby(st2) ylabel
. vbar div, rsort(div) laby(st2) ylabel
```

would produce bar charts with the divorces increasing or decreasing from left to right, respectively (see Figure 4 for the result of the first of these commands), and labeled by the state identifiers. Any ties in either the `div` (values) or the `st2` (labels) variables would have been handled properly, and separate bars shown for separate, but equal, values.

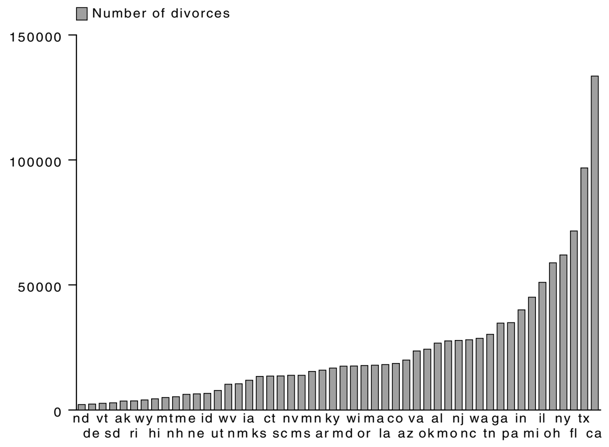


Figure 4. vbar with laby() option.

So far we have considered the case in which `laby()` takes a string variable as argument. The option may also be used with a numeric variable which is either labeled (its labels are used) or unlabeled (its values are used as labels). The same length limit of 8 characters applies.

Note that the sorting in `vbar` is alphabetic, or reverse alphabetic, with string variables, and takes account of any weights specified with numeric variables. I can imagine cases in which the user wants the bar heights to reflect the weights, but not the sorting variable. Note that they need not be the same variable. That cannot be handled by `vbar`. It would make the syntax and programming more complicated for what is, I guess, an unusual case that can be tackled by something like

```
. gen bar = foo*weight
. vbar bar, sort(bazz)
```

In the above, the user wishes to apply weights to `foo`, which together determine the bar heights, but not to the sorting variable `bazz`.

It is not an error to call the `sort` or `rsort` option without `laby`. In that case, `_n` is used to label the sorted bars.

References

Knaul, F. 1993. qs5: How to create stacked bar charts of percentages. *Stata Technical Bulletin* 14: 12–13. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 71–72.

Smith, S. J. 1984. Crime and the structure of social relations. *Transactions, Institute of British Geographers* 9: 427–442.

gr25	Spike plots for histograms, rootograms, and time-series plots

Nicholas J. Cox, University of Durham, UK, FAX (011) 44-91-374-2456, n.j.cox@durham.ac.uk
Anthony R. Brady, Public Health Laboratory Service Statistics Unit, UK, tbrady@phls.co.uk

Syntax

The syntax for the `spikeplt` command is

`spikeplt` *varname* [*weight*] [`if` *exp*] [`in` *range*] [`, round(`*#*`) frac root zero(`*#*`)` *graph_options*]

Options

`round(`*#*`)` uses the `round()` function to round *varname* to the nearest multiple of #. See the help on functions or [U] **20.3.5** in the manual. In other words, `round` specifies the bin width or class interval.

`frac` specifies that the vertical scale is to show fractions. This convention is the opposite of that for `graph, histogram`.

`root` specifies that the vertical scale is to show square roots of frequencies (J. W. Tukey's "rootogram").

`zero(`*#*`)` specifies a constant other than 0 as a level from which spikes are drawn vertically.

graph_options are options allowed with `graph, twoway`. `by()` is allowed, but `total` is trapped with a warning, as the total graph produced with `by(`*byvar*`) total` would superimpose spikes, not add them vertically.

The defaults for these graph options are

`l2title` of "Frequency" (or "Fraction", if `frac` is used, or "Root of frequency", if `root` is used);

`b2title` of the variable label of *varname* (or *varname* itself, if the label is not defined);

`c(||)`, so that each bin or value is shown by a vertical spike;

`s(ii)`, so that invisible point symbols are used.

Explanation

A standard idiom with `graph` is

`graph` y_1 y_2 x `, c(||) sy(ii)`

which connects y_1 and y_2 by vertical lines on a scatter (twoway) plot of y_1 and y_2 against x, in this case without visible point symbols. From that it is a small step to see that setting y_2 to a constant (e.g., 0 or some other reference level) produces some useful graphs, and another small step to see that some prior calculation gives you a kind of histogram, with separate spikes, not touching bars.

`spikeplt` has been written to automate the production of such graphs, so that users can, as far as possible, get what they want with a single command. The name "spike plot" was suggested partly by the "spike chart" of Berry (1996, 14).

Histograms

Some data analysts producing histograms seem to prefer spikes to bars, as a matter of logic or of taste, especially with discrete variables. Examples are found in Plackett (1971) and Evans, Hastings, and Peacock (1993). Even if people want a conventional histogram with touching bars, the limit in `graph, histogram` of 50 bins sometimes proves frustrating. In our experience, this is often when there is some fine structure in the frequency distribution that is of interest, even if it is in some way spurious or pathological. Examples come from demographic data on human ages; people prefer to state certain ages (such as multiples of 10 or 5 years) as a matter of vanity, ignorance, biases in memory, and so forth. The number of possible ages is clearly about 100 and the fine structure of the distribution can only be seen well if each is a separate bin. For such tasks, `spikeplt` offers an alternative to `graph, histogram`.

A more pervasive issue is the size of the data set. In broad terms, the details of a histogram should be less affected by quirks of sampling as the number of values n increases. Hence with larger n, more bins can be justified, although it is difficult to make this precise in a general manner. Suggested rules of thumb for the number of bins include those discussed by Emerson and Hoaglin (1983) and in [R] **graph histogram**. With a rule of $2\sqrt{n}$, more than 50 bins would be needed for $n > 625$; with a $10\ \log_{10} n$ rule, for $n > 100{,}000$; and with Sturges' rule $1 + \log_2 n$, for $n > 2^{49} \approx 5.6 \times 10^{14}$. Another rule, suggested half-seriously by the current first author, is $3\sqrt[3]{n}$, for which the threshold is 4,630. The compromise rule discussed in [R] **graph histogram** of

$$\min(\sqrt{n}, 10\ \log_{10} n)$$

leads to a threshold of 100,000. Without taking these rules more literally than they deserve, we note simply that some, but not all, imply more than 50 bins for data set sizes that are frequently encountered.

Note that in Stata 5.0, it is possible, at the cost of some programming, to create your own alternative histograms with `gph`. These could have more than 50 bins and they could even have unequal bin widths.

Rootograms

`spikeplt` allows the production of the "rootograms" suggested by J. W. Tukey *circa* 1965. The idea is to show not frequencies, but their square roots. Frequencies, as counted variables, tend to have variability that is stabilized by a root transformation, at least approximately. Note also that the square root of a normal or Gaussian density is a multiple of another normal or Gaussian density. Hence if the normal is the reference distribution, we are looking for the same shape on a rootogram, and experience in assessing histograms for approximate normality can be applied directly in assessing rootograms. However, taking the root is only the first step in Tukey's procedure, and we do not implement his hanging or suspended rootograms. See Tukey (1965, 1972, 1977), Tukey and Wilk (1965), or Velleman and Hoaglin (1981).

Consider dotplots

In several ways, spike plots are comparable to the dotplots provided by Stata's `dotplot` command, which can also be very useful in showing the fine structure of data distributions. Apart from the obvious difference that dots represent individual values and spikes represent bin frequencies, a further difference is that dotplots are tuned by indirectly changing the number of bins, whereas spike plots are tuned by directly changing the bin width. For many problems, we prefer dotplots to spike plots, and conversely.

Time-series plots

Some kinds of time series lend themselves quite well to spike plots. Daily rainfalls for a year are often plotted as a series of vertical spikes. On such graphs, wet and dry spells come out well, even for British stations, where according to a U.S. myth it is usually raining. For other time series, setting y_2 to some average level shows clearly periods above and below average.

Examples

In practice, frequency distribution problems for `spikeplt` come in two forms. First, the data for a variable are to be summarized as a frequency distribution. This requires basically

```
spikeplt varname
```

so long as the bin width or class interval is the same as the resolution of the data, or

```
spikeplt varname, round(#)
```

if rounding is required. The number specified in place of # is the bin width or class interval.

Second, the data for two variables show values and frequencies. This form requires the use of weights. For example, Mosteller, Fienberg, and Rourke (1983, 83) give data on the age structure of the population of Ghana reported in the 1960 census, rounded down to the nearest thousand, and taken from the U.N. *Demographic Yearbook* for 1962 (p. 189). These data can be found in the file `ghanaage.dta`. For ages from 0 (under 1 year) to 90 years, the frequencies are the numbers at each age. Note that 15000 people were 91 and over or of unknown age. 91 bins are too many for `graph, histogram`, which is why we are using `spikeplt`.

The command is then

```
. spikeplt age [w=pop], l2("Population in 000") gap(5) ylabel xlabel(0,20,40,60,80) xtick(10,30,50,70,90)
```

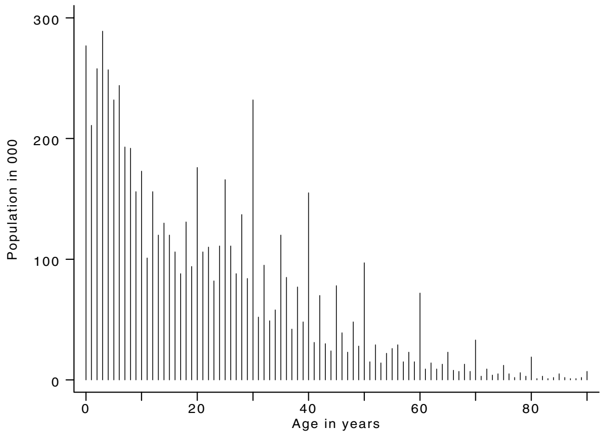


Figure 1. Spike plot of age.

The resulting graph shown in Figure 1 shows heaping of ages, which is a well-known phenomenon in demography. It is easy to pick out preferences for ages that are multiples of 10 and of 5. In addition, ages ending with 2, 4, 6, and 8 are generally more frequent and ages ending with 1, 3, 7, and 9 generally less frequent than would be expected if there were a relatively smooth underlying distribution. Further comment would require some anthropological or sociological knowledge on the significance of various numbers among this population: is 30, for example, a particularly important age to achieve for some or all groups in Ghana? Another example of age heaping for the female population of Mexico is given by Mosteller and Tukey (1977, 476–477). Apart from its occurrence in demography, which appears to have been recognized since at least the 18th century (Westergaard 1932, 77), preference for certain digits in reported numbers has been identified in several sciences (Cox 1991). Spike plots showing the frequencies of all possible results are a natural tool in recognition of such biases.

Time-series plots that are spike plots also require the use of weights. The usual situation is that each time occurs just once in the data. Reference levels other than 0 are often useful, such as a mean or median.

To illustrate using `spikeplt` for time series, we consider the yearly temperature means for the world and each hemisphere given by Parker and Jones (1991), expressed as deviations from the 1951–80 mean. The data are given in `gltemp.dta`. A spike plot representation of the Southern hemisphere series is given in Figure 2 and is obtained by

```
. spikeplt year [w=shtemp], l1("temperature deviation from 1951-80 mean") l2("deg C") gap(5)
```

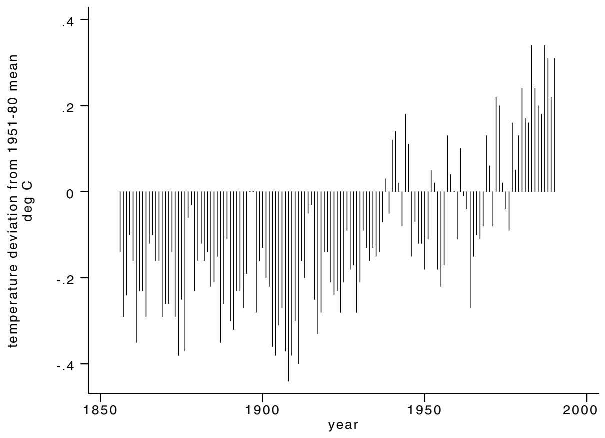


Figure 2. Spike plot of temperature showing deviations from 1951–80 mean

The pattern of recent warming stands out quite well. Note that we must override the default `l2title` of “Frequency”, which makes no sense in this example.

Note that when printing the result of using `spikeplt`, users may want to draw the spikes with pen thicknesses larger than the default.

References

Berry, D. A. 1996. *Statistics: A Bayesian Perspective.* Belmont, CA: Duxbury Press.

Cox, N. J. 1991. Human factors. *Nature* 353: 597.

Emerson, J. D. and D. C. Hoaglin. 1983. Stem-and-leaf displays. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 7–32. New York: John Wiley.

Evans, M., N. Hastings, and B. Peacock. 1993. *Statistical Distributions.* New York: John Wiley.

Mosteller, F., S. E. Fienberg, and R. E. K. Rourke. 1983. *Beginning Statistics with Data Analysis.* Reading, MA: Addison–Wesley.

Mosteller, F. and J. W. Tukey. 1977. *Data Analysis and Regression.* Reading, MA: Addison–Wesley.

Parker, D. E. and P. D. Jones. 1991. Global warmth in 1990. *Weather* 46: 302–311.

Plackett, R. L. 1971. *An Introduction to the Theory of Statistics.* Edinburgh: Oliver & Boyd.

Tukey, J. W. 1965. The future of processes of data analysis. Reprinted in *The Collected Works of John W. Tukey, Volume IV: Philosophy and Principles of Data Analysis: 1965–1986*, ed. L. V. Jones, 517–547 (1986). Monterey, CA: Wadsworth & Brooks/Cole.

Tukey, J. W. 1972. Some graphic and semigraphic displays. In *Statistical Papers in Honor of George W. Snedecor*, ed. T. A. Bancroft and S. A. Brown, 293–316. Ames, IA: Iowa State University Press.

Tukey, J. W. 1977. *Exploratory Data Analysis.* Reading, MA: Addison–Wesley, Ch. 17.

Tukey, J. W. and M. B. Wilk. 1965. Data analysis and statistics: Principles and practice. Reprinted in *The Collected Works of John W. Tukey, Volume V: Graphics: 1965–1985*, ed. W. S. Cleveland, 23–29 (1988). Pacific Grove, CA: Wadsworth & Brooks/Cole.

United Nations. 1962. *Demographic Yearbook 1962.* New York: United Nations.

Velleman, P. F. and D. C. Hoaglin. 1981. *Applications, Basics, and Computing of Exploratory Data Analysis.* Boston, MA: Duxbury Press, Ch. 9.

Westergaard, H. 1932. *Contributions to the History of Statistics.* London: P. S. King.

ip16 Using dialog boxes to vary program parameters

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

One of the most exciting new features of Stata 5.0 for Windows is the ability to use dialog boxes to provide a graphical user interface to Stata commands and programs. While dialog boxes can be used in a variety of ways, in this insert we describe how they can be used to rapidly vary the value of a parameter such as a smoothing parameter or a bandwidth in a kernel density estimator and have for each value of the parameter a new graph appear in the graphics window. This use of dialog boxes makes it possible to emulate using a "slider" in a language such as `Xlisp`.

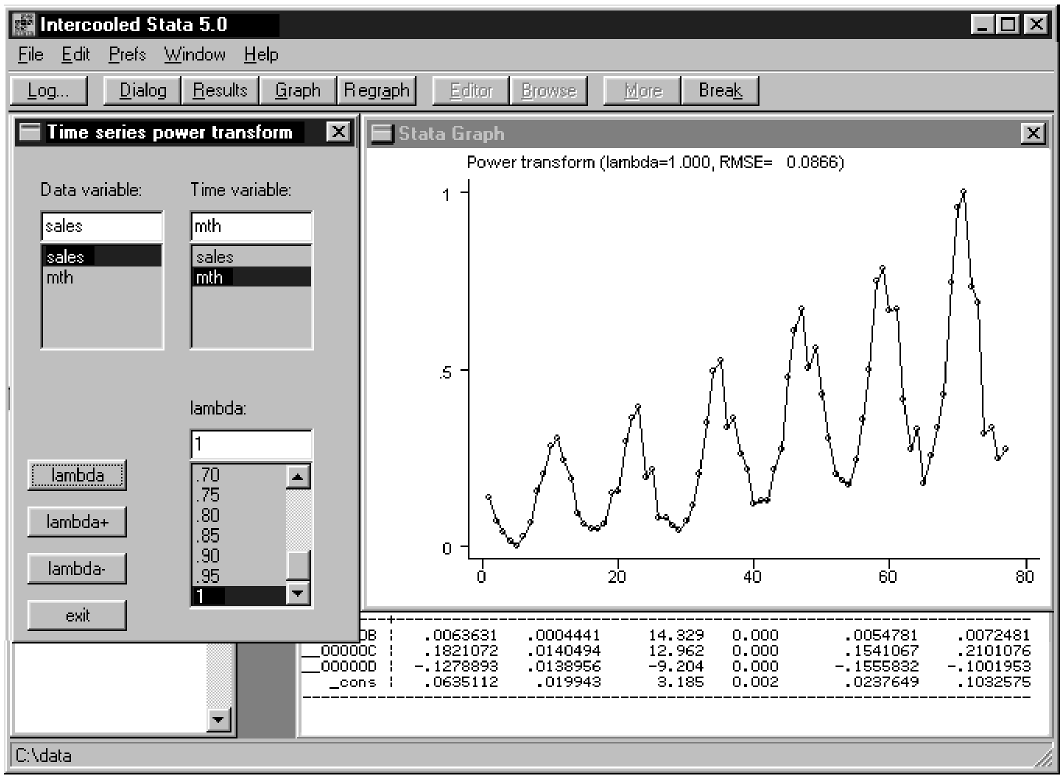
To illustrate what we mean, consider Figure 1 where we are using a program we have written called `ptransdb` ("power transform dialog box") to try to find a suitable power transform for a time series of monthly sales data (discussed by Chatfield and Prothero (1973) and Newton (1988, 233), and included with this insert as `sales.dta`). If we denote the data by $X_1, \ldots, X_n$ (with $n = 77$ for the sales data), we seek a value of the exponent λ giving us a transformed data set (with constant variance across time) $Y_1, \ldots, Y_n$ by

$$Y_t = \begin{cases} X_t^\lambda & \lambda \neq 0 \\ \log(X_t) & \lambda = 0 \end{cases}$$

Using such a power transform for stabilizing variance is a standard initial step in many methods of time series analysis; see Box and Jenkins (1970), for example.

In Figure 1, we see a dialog box labeled "Time series power transform" as well as a Stata Graph window with the plot of original sales data (except that it has been standardized to be in the interval $[0, 1]$ by subtracting its minimum value and dividing by its range, a practice we do throughout this insert so that series for different values of λ are comparable).

We have resized the Stata window and the Graph window so that both the dialog box and Graph window are visible. We have also made the foreground color black and the background color white in the graphics window so that Figure 1 would show up well when printed.


Intercooled Stata 5.0
File Edit Prefs Window Help
Log... Dialog Results Graph Regraph Editor Browse More Break
Time series power transform
Data variable:
Time variable:
sales
mth
lambda:
1
.70
.75
.80
.85
.90
.95
lambda
lambda+
lambda-
exit
Stata Graph
Power transform (lambda=1.000, RMSE= 0.0866)
C:\data

```
---+------------------------------------------------------------------------
  0B |   .0063631   .0004441     14.329   0.000       .0054781    .0072481
__00000C |   .1821072   .0140494     12.962   0.000       .1541067    .2101076
__00000D |  -.1278893   .0138956     -9.204   0.000      -.1555832   -.1001953
   _cons |   .0635112    .019943      3.185   0.002       .0237649    .1032575
----------------------------------------------------------------------------
```

Figure 1. Dialog box for time series power transform (lambda = 1).

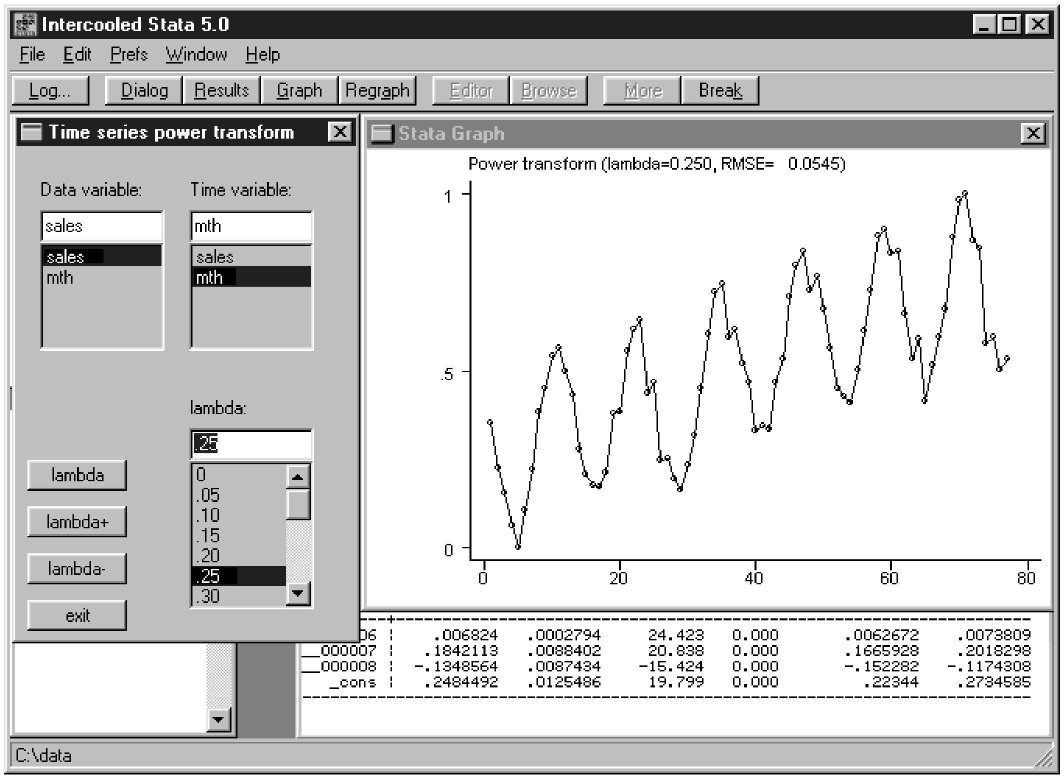
The dialog box has list boxes for the user to choose the data and time variables (the user is expected to input the data prior to using `ptransdb`). The user can then try a value of λ by choosing a value in the list box labeled “lambda” (for simplicity the choices range from zero to one in steps of 0.05) and then clicking on the button labeled “lambda” (when `ptransdb` begins it uses the value one so that the standardized original data is plotted).

Alternatively, one can rapidly increase or decrease the value of λ by clicking on the buttons labeled “lambda+” and “lambda−”, respectively. Each time such a click is done, the graphics window displays the new standardized transformed data set. This allows a user to almost animate the graphs of the successive power transforms (unfortunately one must actually click to get each graph, thus slowing down the animation).

Note that the standardization is important here as it keeps the vertical axis from changing which would disturb the visual impression of smoothly changing the value of λ. Clicking on the button labeled “exit” ends `ptransdb` and control of Stata is returned to the command line.

In Figure 2, we display the result of using $\lambda = 0.25$, that is, the fourth root transform recommended by Chatfield and Prothero for these data. Notice that the value of λ is inserted into the caption above each graph as is a number called “RMSE”. This is the root mean square error in regressing the standardized transformed data on the sum of a linear trend and a cosine plus sine of period 12.

For the sales data the value of λ minimizing this variance is arguably the best value to stabilize variance across time. Except for this number, `ptransdb` can be used for any time series. In our discussion below of how to write a program such as `ptransdb`, we show how this regression can be easily removed from the program.


Intercooled Stata 5.0
File Edit Prefs Window Help
Log... Dialog Results Graph Regraph Editor Browse More Break
Time series power transform
Data variable:
Time variable:
sales
mth
lambda:
.25
lambda
lambda+
lambda-
exit
0 .05 .10 .15 .20 .25 .30
Stata Graph
Power transform (lambda=0.250, RMSE= 0.0545)
1
.5
0
0 20 40 60 80
C:\data

```
---------+--------------------------------------------------------------------
      06 |    .006824   .0002794     24.423   0.000       .0062672    .0073809
__000007 |   .1842113   .0088402     20.838   0.000       .1665928    .2018298
__000008 |  -.1348564   .0087434    -15.424   0.000       -.152282   -.1174308
   _cons |   .2484492   .0125486     19.799   0.000         .22344    .2734585
------------------------------------------------------------------------------
```

Figure 2. Dialog box for time series power transform (lambda = 0.25).

Writing programs using dialog boxes to vary parameters

In this section, we present an annotated version of the file `ptransdb.ado` containing the `ptransdb` program. See [R] **window control** for complete details on the elements of Stata dialog boxes.

First we define the program and get the current list of variables into the global variable `DB_var`:

```
program define ptransdb
        version 5.0
        local varlist opt
        parse "`*´"
        global DB_var "`varlist´"
```

Next we use Stata's `window control ssimple` command to define the three list boxes in the dialog box:

```
        global DB_one  "Data variable:"
        window control static          DB_one  10 10 50 9
        global DB_data " "
        window control ssimple         DB_var  10 20 50 50 DB_data
        global DB_two  "Time variable:"
        window control static          DB_two  70 10 50 9
        global DB_time " "
        window control ssimple         DB_var  70 20 50 50 DB_time
        global DB_l    "lambda:"
        window control static          DB_l    70 80  50 9
        global DB_lams "0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 .55 .60"
        global DB_lams "$DB_lams .65 .70 .75 .80 .85 .90 .95 1"
        global DB_lam  "1"
        window control ssimple         DB_lams 70 90 50 60 DB_lam
```

Now place the four buttons used in the dialog box and define the global macros used to hold the actions to take for each button (note that the actions for all but "exit" are to call the program `bcdr` defined below):

```
                window control button "lambda"      5 100 40 10 DB_lc
                window control button "lambda+"     5 115 40 10 DB_lp
                window control button "lambda-"     5 130 40 10 DB_lm
                window control button "exit"        5 145 40 10 DB_ex

                global DB_lc   "bcdr 0"
                global DB_lp   "bcdr 1"
                global DB_lm   "bcdr -1"
                global DB_ex   "exit 3000"
```

Finally, issue the command `wdlg` that actually forms the dialog box:

```
                wdlg "Time series power transform" 10 10 140 170
        end
```

Another program, `bcdr`, does the actual calculation and graphing. Note that `bcdr` has one argument which can be -1 for "lambda$-$," $+1$ for "lambda$+$," and 0 for "lambda." Since it is called only by `ptransdb`, we need not worry about checking arguments or any of the usual programming problems.

```
        program define bcdr
                version 5.0

                local arg `1´
```

Now get the data variable and time variable into the tempvar's `x` and `t` and update the value of λ:

```
                tempvar x t
                gen `x´ = $DB_data
                gen `t´ = $DB_time

                if `arg´ == -1 { global DB_lam = max(0,$DB_lam-.025)}
                if `arg´ ==  1 { global DB_lam = min(1,$DB_lam+.025)}
```

Next do the called-for transform and do the standardizing:

```
                if $DB_lam == 0 { replace `x´=log(`x´)}
                else            { replace `x´=`x´^$DB_lam}

                sum `x´
                replace `x´ = (`x´-_result(5))/(_result(6)-_result(5))
```

This section does the regression and can be easily removed:

```
                tempvar cs sn
                gen `cs´ = cos(2*_pi*_n/12)
                gen `sn´ = sin(2*_pi*_n/12)
                regress `x´ `t´ `cs´ `sn´
                local s2 : display %9.4f _result(9)
```

Now do the graph, noting that to remove the regression, one need only change the argument of the `t1` option in `graph`:

```
                local lam : display %5.3f $DB_lam
                label var `x´ " "
                label var `t´ " "
                graph `x´ `t´, xlab ylab c(l) /*
                */    t1("Power transform (lambda=`lam´, RMSE=`s2´)")
        end
```

References

Box, G. E. P. and G. M. Jenkins. 1970. *Time Series Analysis, Forecasting, and Control.* San Francisco: Holden–Day.

Chatfield, C. and D. L. Prothero. 1973. Box–Jenkins seasonal forecasting: Problems in a case study. *Journal of the Royal Statistical Society, Series A* 136: 295–336.

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

sbe13.2 Correction to age-specific reference intervals (“normal ranges”)

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

We have discovered that our routine, `xriml`, for calculating age-specific reference intervals (Wright and Royston, 1997) has an error in the estimation of the standard errors of centiles (option `se`) when used in conjunction with modeling the coefficient of variation (option `cv`). Confidence limits based on this routine are too narrow. An amended version is available with this edition of STB. `xriml` is correct in all other respects.

Reference

Wright, E. and P. Royston. 1997. sbe13: Age-specific reference intervals (“normal ranges”). *Stata Technical Bulletin* 34: 24–34.

sbe14 Odds ratios and confidence intervals for logistic regression models with effect modification

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

This insert describes `effmod`, an easy-to-use program for anyone who dislikes having to hand calculate odds ratios and confidence intervals for logistic regression models with significant interaction terms, or, even worse, have to explain to students how to do it. After years of teaching non-statistician medical researchers how to use logistic regression, and watching their eyes glaze over when I got to the formula for calculating the variance estimate for a linear combination of betas, I decided to make life easier on all of us and write a program that automates the process.

After writing `effmod`, I discovered the `lincom` command in Stata 5.0, which will also display odds ratios and confidence intervals (see [R] `lincom` in the Stata Reference Manual). `lincom` and `effmod` produce the same results, but they differ in their syntax. `lincom` requires you to specify the appropriate linear combinations of the estimators; `effmod` uses a syntax based on descriptive terms familiar to anyone who has studied epidemiology.

Which program is better depends entirely on user preference. `effmod` is geared toward a non-mathematically inclined audience and has the advantage of being specified and displayed using epidemiological terminology, with a summary of the variables and stratum-specific values used for the odds ratios. However, `lincom` can be explained in similar terms, and, if specified appropriately, is simple to use. Following each example in this insert I will include the statement needed to duplicate the results using `lincom`. However, remember that `lincom` must follow the estimation of a model.

Background

The general form for the calculation of the odds ratio from the estimates of the logistic regression model is

$$\mathrm{OR}_{X^*,X} = \exp\left(\sum \beta_i (X_i^* - X_i)\right)$$

where X^* and X represent values for one group and a comparison group, respectively.

In many epidemiologic studies, the focus is on one main study factor (“exposure”) which frequently is coded as 1 for “exposed” and 0 for “unexposed”. For instance, suppose X_1 were the exposure variable:

$$\begin{array}{ll} & \text{Exposure Status} \\ & \quad\downarrow \\ \text{Let} & X^* = (1, X_2, X_3, \ldots, X_k) \\ & X \;\;= (0, X_2, X_3, \ldots, X_k) \end{array}$$

Then

$$\mathrm{OR} = \exp\left(\beta_1(1-0) + \beta_2(X_2^* - X_2) + \cdots + \beta_k(X_k^* - X_k)\right) = e^{\beta_1}$$

Our fairly complicated odds ratio formula reduces to a simple exponentiation of the beta coefficient for the “exposure” variable (which Stata is kind enough to present for us on the `logistic` output).

The formula for the 95% confidence interval (which Stata also calculates and prints for us) is

$$\exp(\beta_1 \pm 1.96(s.e._{\beta_1}))$$

Had the exposure variable been coded as something other than 1 and 0, we would need to multiply the beta coefficient by the difference that we want to compare before exponentiating. For instance, suppose age in years was our “exposure.” If we exponentiate beta (or use the odds ratio calculation we find on the printout), we are looking at the odds ratio for a one year change in age, e.g., a 39 year-old versus a 38 year-old. It might be more informative to report a 10-year difference in age, say comparing a 50 year-old to a 40 year-old. Our odds ratio and 95% confidence interval formulas then become

$$\text{OR} = \exp(\beta_1(50-40)) = e^{10\beta_1} \qquad 95\%\ \text{CI} = \exp(\beta_1(10) \pm 1.96(s.e._{\beta_1})(10))$$

Note that one multiplies the standard error in the confidence interval formula by the same multiple used for beta. A dead giveaway that someone has forgotten to do so is a tiny confidence interval around a reasonable sized odds ratio.

This is still fairly straightforward. However, things start getting messy when there is significant effect modification, which is the epidemiologists' term for interaction between the exposure and another variable—the effect modifier. What we are saying is the odds ratio changes depending on the value of the effect modifier or effect modifiers. In essence, we are stratifying our odds ratio by categories of the effect modifiers. The general formula for the odds ratio reduces, but any terms which include interactions with the exposure variable remain. For example, suppose we are interested in the odds ratio of developing coronary heart disease (yes $\texttt{chd} = 1$; no $\texttt{chd} = 0$) for people with hypertension ($\texttt{hpt} = 1$) versus people with normal blood pressure ($\texttt{hpt} = 0$), and we find there is significant interaction between hypertension and age, as well as hypertension and sex. The logistic model (written in the log odds form) might look like this:

$$\text{logit}\,\Pr(\texttt{chd} = 1|X_i) = \alpha + \beta_1(\texttt{hpt}) + \beta_2(\texttt{age}) + \beta_3(\texttt{sex}) + \beta_4(\texttt{hpt} \times \texttt{age}) + \beta_5(\texttt{hpt} \times \texttt{sex})$$

The formula for the odds ratio with the two interaction terms would be

$$\text{OR} = \exp(\beta_1(1-0) + \beta_4(1-0)(\texttt{age}) + \beta_5(1-0)(\texttt{sex})) = \exp(\beta_1 + \beta_4(\texttt{age}) + \beta_5(\texttt{sex}))$$

Next we would substitute values of age and sex and use the estimated betas to solve the equation to get odds ratios for the `hpt–chd` comparison. For example, we might want the `hpt–chd` odds ratio for 50 year-old males, or for 60 year-old females.

Okay so far, but what about the confidence intervals for these odds ratios? Many journals are requiring confidence intervals rather than p-values when we report odds ratios, and now we need separate confidence intervals for each odds ratio (we may have several odds ratios representing the categories of our effect modifiers). Not only do we need more confidence intervals, the variance estimate for the formula no longer is the simple variance of a single beta. It's now a more complicated formula for a linear combination of betas. The general form for the 95% confidence interval (with interaction) is $\exp(l \pm 1.96\sqrt{\text{Var}(l)})$ where $l = \sum_{i=1}^{k} a_i Y_i$ and

$$\text{Var}(l) = \sum_{i=1}^{k} a_i^2 \text{Var}(Y_i) + 2\sum_{i<j} a_i a_j \text{Cov}(Y_i, Y_j)$$

The good news is the terms involving variables other than the exposure and the effect modifiers drop out of this equation. Additionally, if the exposure is coded as 1 and 0, the equations for l and $\text{Var}(l)$ become

$$l = \beta + \sum_{j=1}^{k} \beta_j M_j$$

$$\text{Var}(l) = \text{Var}(\beta) + \sum_{j=1}^{k} M_j^2 \text{Var}(\beta_j) + 2\sum_{j=1}^{k} M_j \text{Cov}(\beta, \beta_j) + 2\sum_{j<j'} M_j M_{j'} \text{Cov}(\beta_j, \beta_{j'})$$

where β = beta for the exposure variable, β_j = betas for the j (up to k) interaction terms, and M_j = values for the j effect modifiers.

For the 2 interaction term example ($\texttt{hpt} \times \texttt{age}$ and $\texttt{hpt} \times \texttt{sex}$), we would have

$$l = \beta_1 + \beta_4(\texttt{age}) + \beta_5(\texttt{sex})$$

$$\begin{aligned}\text{Var}(l) = \text{Var}(\beta_1) + (\texttt{age})^2\text{Var}(\beta_4) + (\texttt{sex})^2\text{Var}(\beta_5) + 2(\texttt{age})\text{Cov}(\beta_1, \beta_4) \\ + 2(\texttt{sex})\text{Cov}(\beta_1, \beta_5) + 2(\texttt{age})(\texttt{sex})\text{Cov}(\beta_4, \beta_5)\end{aligned}$$

Now we pick off the appropriate estimates from the variance–covariance matrix, substitute in a value for age and sex, and solve the equation. All this just to get one of the confidence intervals for one odds ratio. We must repeat the calculation for other combinations of age and sex. Of course, if we had started with only one interaction term (rather than two), the variance estimate would reduce quite a bit (to 3 terms). Additionally, had the single effect modifier been a dichotomous 0–1 variable, the equation would reduce further to the familiar e^{β_1} for the 0 category. Would you still rather not be bothered? Then `effmod` will help.

Description

`effmod` calculates user specified stratum-specific odds ratios and confidence intervals for logistic regression models which include interaction terms between an exposure and effect modifiers. If the model has been specified previously or `effmod` is repeated with new stratum values, the current model estimates are used; otherwise a new model is fit.

Syntax

`effmod` *dvar evar* [`if` *exp*] `,` `covariate(`*cov_list*`)` `interact(`*interaction #*`)`
[`ediff(`*#*`)` `model` `level(`*#*`)`]

dvar is the “disease” variable (dichotomous outcome) and should be coded as 1 and 0.

evar is the “exposure” variable and can be nominal, ordinal, interval, or continuous.

Options required

`covariate(`*cov_list*`)` is the list of confounders in the model.

`interact(`*interaction #*`)` is the list of interaction variables in the model plus a stratum value for each effect modifier; for more than one interaction term, the form is `interact(`*interaction*$_1$ *#* *interaction*$_2$ *#* . . . `)`.

For example, suppose there are two interactions represented by the variables `hptxsex` (hypertension by sex, where sex=1 for males and sex=0 for females) and `hptxage` (hypertension by age). To calculate the odds ratio and 95% CI for 60 year-old males, specify

```
interact(hptxsex 1 hptxage 60)
```

and for 60 year-old females

```
interact(hptxsex 0 hptxage 60)
```

and so on.

Options allowed

`ediff(`*#*`)` is the difference of values for “exposed” versus “unexposed.” For example, to compare age = 50 versus age = 40, `ediff` would be specified as: `ediff(10)`. The default for `ediff` is 1, which is equivalent to exposed = 1 versus unexposed = 0.

`model` displays the logistic regression table.

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

Examples

The set of examples to illustrate `effmod` come from a case–control study (Garrett et al. 1993) of the relationship between the “exposure”—a rare form of a gene (`hras`, for those readers familiar with genetics) and the “disease”—incidence of breast cancer (`brcancer`). (I apologize in advance to the author of this study for taking liberties with the data to illustrate some points, but, since the author is related to me, I hope he won’t mind.) The data are stored in `hras.dta`. Several of the variables with their descriptions and coding are in the following table:

Variable	Definition	Coding
`brcancer`	Breast cancer diagnosis (“outcome”)	1 = case 0 = control
`hras`	Type of gene (“exposure”)	1 = rare form of hras 0 = common form
`race`	Race of patient	1 = black women 0 = white women
`bmi`	Body mass index—a measure of obesity	continuous (higher #s mean heavier)
`pabiopsy`	Past history of breast biopsy	1 = yes 0 = no
`menopaus`	Has been through menopause	1 = post-menopause 0 = pre-menopause

In addition, there are interaction terms between `hras` and `race` (`hrasxrac`), `hras` and `bmi` (`hrasxbmi`), and `bmi` and `menopaus` (`bmixmeno`).

In the first example, `effmod` calculates the odds ratio and 95% confidence interval for rare hras and breast cancer with one interaction term between `hras` and `race` (`hrasxrac`) for black women (`race` $= 1$), controlling for `race`, `bmi`, and `pabiopsy`. It requests that the logistic regression table be printed (`model`), and accepts the default for `ediff` (1) since “exposed” (`hras` $= 1$) minus “unexposed” (`hras` $= 0$) equals 1. The model is

$$\operatorname{logit}\Pr(\texttt{brcancer} = 1) = \alpha + \beta_1(\texttt{hras}) + \beta_2(\texttt{race}) + \beta_3(\texttt{bmi}) + \beta_4(\texttt{pabiopsy}) + \beta_5(\texttt{hrasxrac})$$

With the commands

```
. logistic brcancer hras race bmi pabiopsy hrasxrac
. lincom hras + hrasxrac
```

`logistic` will fit the model and produce the usual table of odds ratios and confidence intervals for the odds ratios and `lincom` will produce:

```
 ( 1)  hras + hrasxrac = 0.0
------------------------------------------------------------------------------
brcancer | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     (1) |    4.20202   2.343971      2.574   0.010       1.408123    12.53936
------------------------------------------------------------------------------
```

Using `effmod` gets the results of both `logistic` and `lincom` using a syntax my non-statistician students like:

```
. effmod brcancer hras, cov(race bmi pabiopsy) int(hrasxrac 1) model
Logit Estimates                                       Number of obs =    706
                                                      chi2(5)       =  74.52
                                                      Prob > chi2   = 0.0000
Log Likelihood = -447.32742                           Pseudo R2     = 0.0769
------------------------------------------------------------------------------
brcancer | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    hras |    1.15649   .5195732      0.324   0.746       .4794298    2.789709
    race |   .5881098   .1808482     -1.726   0.084         .32189    1.074507
     bmi |   1.056875   .0169562      3.448   0.001       1.024159    1.090637
pabiopsy |    3.21892   .5240456      7.181   0.000       2.339554    4.428812
hrasxrac |   3.633424   2.605469      1.799   0.072       .8911144     14.8149
------------------------------------------------------------------------------
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    hras
 Confounders: race bmi pabiopsy
 Interaction Terms and Stratum Values:
              hrasxrac: 1
 Exposed-Unexposed= 1
   l= 1.4355653
   Var(l)= .31116333
 Odds Ratio (95% CI) for brcancer vs. hras:  4.202  (1.408, 12.539)
```

Following the model results, `effmod` reports a summary table of information (a way to check to make sure the variables were specified as expected, particularly if the model is not printed), including "*l*", the variance of "*l*", and the final odds ratio and confidence interval for the stratum requested. In this case, the stratum for black women is specified as "`hrasxrac 1`." The odds ratio tells us that black women with the rare form of the hras gene are 4.2 times as likely to develop breast cancer as black women with the common form of the gene. The 95% confidence interval (1.4 to 12.5) does not include 1.0 (which would mean no association), thus, the odds ratio is statistically significant.

Now, let's repeat the example for white women ("`hrasxrac 0`"):

```
. effmod brcancer hras, cov(race bmi pabiopsy) int(hrasxrac 0)
OR and 95% CI for a Logistic Regression Model with Interaction
---------------------------------------------------------------
 Disease:     brcancer
 Exposure:    hras
 Confounders: race bmi pabiopsy
 Interaction Terms and Stratum Values:
              hrasxrac: 0
 Exposed-Unexposed= 1
   l= .14538979
   Var(l)= .20184104
 Odds Ratio (95% CI) for brcancer vs. hras:  1.156  (0.479, 2.790)
```

The corresponding `lincom` command is

```
. lincom hras
```

This time the odds ratio tells us that among white women there is no association between having the rare form of hras and breast cancer—the odds ratio is close to 1.0 and the confidence interval includes 1.0, meaning it is nonsignificant. There was no need to print the model again since the estimates did not change from the previous example.

For the next examples, let's suppose that in addition to the `hras` and `race` interaction, we think there is interaction between `hras` and `bmi` (`hrasxbmi`). Again, `race` is dichotomous, specified as 0 (white women) or 1 (black women), but `bmi` is continuous, so we must select some values of `bmi` for our strata. I have selected `bmi` = 23 (a normal value for body mass index for women) and `bmi` = 28 (getting rather zaftig—for non-New Yorkers, that means "plump"). Four examples of `effmod` with two interaction terms follow:

`race` = 1 and `bmi` = 23

`race` = 1 and `bmi` = 28

`race` = 0 and `bmi` = 23

`race` = 0 and `bmi` = 28

For the first example, we have

```
. effmod brcancer hras, cov(race bmi pabiopsy) int(hrasxrac 1 hrasxbmi 23) model
Logit Estimates                                         Number of obs =    706
                                                        chi2(6)       =  77.91
                                                        Prob > chi2   = 0.0000
Log Likelihood = -445.63177                             Pseudo R2     = 0.0804
------------------------------------------------------------------------------
brcancer | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    hras |   12.40228   16.86845      1.851   0.064       .8625319    178.3316
    race |   .5529998   .1725018     -1.899   0.058       .3000562    1.019172
     bmi |   1.067554     .01819      3.836   0.000       1.032491    1.103807
pabiopsy |   3.242319   .5297073      7.200   0.000       2.353924    4.466004
hrasxrac |   5.728345   4.340338      2.304   0.021       1.297412    25.29183
hrasxbmi |   .9100992   .0460674     -1.861   0.063       .8241431     1.00502
------------------------------------------------------------------------------
```

```
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    hras
 Confounders: race bmi pabiopsy
 Interaction Terms and Stratum Values:
              hrasxrac: 1
              hrasxbmi: 23
 Exposed-Unexposed= 1
   l= 2.0966696
   Var(l)= .45142941
 Odds Ratio (95% CI) for brcancer vs. hras:  8.139  (2.181, 30.373)
```

We see from the model that both interactions are statistically significant, or nearly so. But are the individual stratum-specific odds ratios significant? Black women with a normal value (23) for body mass index are 8.1 times as likely to develop breast cancer if they have the rare form of hras. The confidence interval (2.18 to 30.37) does not include 1.0, thus the odds ratio is significant.

For `race` = 1 and `bmi` = 28:

```
. effmod brcancer hras, cov(race bmi pabiopsy) int(hrasxrac 1 hrasxbmi 28)
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    hras
 Confounders: race bmi pabiopsy
 Interaction Terms and Stratum Values:
              hrasxrac: 1
              hrasxbmi: 28
 Exposed-Unexposed= 1
   l= 1.6256615
   Var(l)= .32253505
 Odds Ratio (95% CI) for brcancer vs. hras:  5.082  (1.670, 15.468)
```

Black women with a high value (28) for body mass index are 5.1 times as likely to develop breast cancer if they have the rare form of hras. The odds ratio is significant.

For `race` = 0 and `bmi` = 23:

```
. effmod brcancer hras, cov(race bmi pabiopsy) int(hrasxrac 0 hrasxbmi 23)
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    hras
 Confounders: race bmi pabiopsy
 Interaction Terms and Stratum Values:
              hrasxrac: 0
              hrasxbmi: 23
 Exposed-Unexposed= 1
   l= .35124306
   Var(l)= .21327909
 Odds Ratio (95% CI) for brcancer vs. hras:  1.421  (0.575, 3.513)
```

White women with a normal value (23) for bmi are 1.4 times as likely to develop breast cancer if they have rare hras, but the odds ratio is not significant.

For `race` = 0 and `bmi` = 28:

```
. effmod brcancer hras, cov(race bmi pabiopsy) int(hrasxrac 0 hrasxbmi 28)
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    hras
 Confounders: race bmi pabiopsy
 Interaction Terms and Stratum Values:
              hrasxrac: 0
              hrasxbmi: 28
 Exposed-Unexposed= 1
   l= -.11976513
   Var(l)= .21619827
 Odds Ratio (95% CI) for brcancer vs. hras:  0.887  (0.357, 2.207)
```

White women with a high value (28) for bmi are no more likely to develop breast cancer if they have the rare form of hras.

The equivalent `lincom` commands for the previous four examples are

```
. lincom  hras + hrasxrac + 23*hrasxbmi
. lincom  hras + hrasxrac + 28*hrasxbmi
. lincom  hras + 23*hrasxbmi
. lincom  hras + 28*hrasxbmi
```

Finally, let's look at some examples where the "exposure" is continuous, rather than a 0–1 dichotomous variable. Suppose `bmi` is our exposure variable, i.e., we are interested in the relationship between body mass index and breast cancer. And, we discover that there is effect modification between `bmi` and menopause status (post-menopause: `menopaus` = 1 and pre-menopause: `menopaus` = 0—don't ask what happened to the group of women experiencing menopause—for sake of illustration, let's assume that menopause is an instantaneous event). We'll keep the model simple, and look only at the outcome of breast cancer with `bmi`, `menopaus`, and the `bmi` by `menopaus` interaction (`bmixmeno`). First, let's compare odds ratios for a one-unit change in `bmi`, stratified by post-menopausal women, and then stratified by pre-menopausal women.

```
. effmod brcancer bmi, cov(menopaus) int(bmixmeno 1) model
Logit Estimates                                        Number of obs =    700
                                                       chi2(3)       =  17.78
                                                       Prob > chi2   = 0.0005
Log Likelihood = -470.76854                            Pseudo R2     = 0.0185
------------------------------------------------------------------------------
brcancer | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     bmi |   1.026446   .0182239      1.470   0.142       .9913416    1.062793
menopaus |   .2501185   .2093135     -1.656   0.098       .0485078    1.289675
bmixmeno |    1.06519   .0339957      1.979   0.048       1.000601    1.133948
------------------------------------------------------------------------------
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    bmi
 Confounders: menopaus
 Interaction Terms and Stratum Values:
              bmixmeno: 1
 Exposed-Unexposed= 1
   l= .08925487
   Var(l)= .00070336
 Odds Ratio (95% CI) for brcancer vs. bmi:  1.093  (1.038, 1.152)
```

The corresponding `lincom` command is

```
. lincom bmi + bmixmeno
```

Thus, among post-menopausal women, there is a significant increase in breast cancer with increasing `bmi` (the confidence interval does not include 1.0). However, since we are looking at a one-unit change in `bmi`, at first glance we might conclude erroneously that an odds ratio as small as 1.09 implies body mass index doesn't have much to do with breast cancer incidence.

Next, we look at

```
. effmod brcancer bmi, cov(menopaus) int(bmixmeno 0)
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    bmi
 Confounders: menopaus
 Interaction Terms and Stratum Values:
              bmixmeno: 0
 Exposed-Unexposed= 1
```

```
      l= .02610197
      Var(l)= .00031522
    Odds Ratio (95% CI) for brcancer vs. bmi:  1.026  (0.991, 1.063)
```

with corresponding `lincom` command

```
. lincom bmi
```

which shows that among pre-menopausal women, there is no evidence of increase in breast cancer with increasing `bmi` (the confidence interval includes 1.0)

To make our odds ratios look a little more substantive, we can use the `ediff(#)` option to calculate the odds ratio for a larger change in `bmi`. For instance, the next two examples repeat the previous two, using a 10 point difference on the `bmi` scale. The two `lincom` commands are

```
. lincom 10*bmi + 10*bmixmeno
. lincom 10*bmi
```

Now `effmod` for the first of these two example:

```
. effmod brcancer bmi, cov(menopaus) int(bmixmeno 1) ediff(10)
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    bmi
 Confounders: menopaus
 Interaction Terms and Stratum Values:
              bmixmeno: 1
 Exposed-Unexposed= 10 (shows the difference in bmi)
   l= .89254874
   Var(l)= .00070336
 Odds Ratio (95% CI) for brcancer vs. bmi:  2.441  (1.452, 4.106)
```

This shows us that we can say that post-menopausal women are 2.4 times as likely to develop breast cancer with a 10 point increase in `bmi`. Again, we conclude this odds ratio is significant. If the relationship is significant for a one-unit change in `bmi`, it will be for a 10-unit change (or 1000-unit change, for that matter—I once had a journal reviewer tell me that since the confidence interval was so close to 1.0 for a one-unit change of a continuous effect modifier, it was sure to cross 1.0 if I tried to look at a 10-unit change).

For the second example of using `ediff` we have

```
. effmod brcancer bmi, cov(menopaus) int(bmixmeno 0) ediff(10)
OR and 95% CI for a Logistic Regression Model with Interaction
--------------------------------------------------------------
 Disease:     brcancer
 Exposure:    bmi
 Confounders: menopaus
 Interaction Terms and Stratum Values:
              bmixmeno: 0
 Exposed-Unexposed= 10
   l= .26101966
   Var(l)= .00031522
 Odds Ratio (95% CI) for brcancer vs. bmi:  1.298  (0.917, 1.839)
```

Although the odds ratio for a 10-unit change in `bmi` is slightly larger than the example where we did not use `ediff(10)`, as expected, the confidence interval remains nonsignificant. Among pre-menopausal women, there is no significant relationship between body mass index and breast cancer.

References

Garrett, P. A., B. S. Hulka, Y. L. Kim, and R. A. Farber. 1993. HRAS protooncogene polymorphism and breast cancer. *Cancer Epidemiology, Biomarkers & Prevention* 2: 131–138.

Kleinbaum, D. G., L. L. Kupper, and H. Morgenstern. 1982. *Epidemiologic Research: Principles and Quantitative Methods.* New York: Van Nostrom Reinhold.

sg67 Univariate summaries with boxplots

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

Univariate summaries (means, standard deviations, etc.) are perhaps the quantities most often examined during data analysis. This is partly because these summaries serve so many purposes, for example, to familiarize oneself with the data, to aid in understanding other computations, or to act as canonical data descriptors in written reports.

Stata's `summarize` command provides the components of a univariate summary, but its presentation is not always ideal for the purpose at hand. For instance, `summarize` *varlist* displays the mean, standard deviation, minimum, and maximum of a set of variables in a left-to-right fashion that allows many such sets of results to be viewed at once. But one might wish to describe each variable by its five-number summary (minimum, 25th percentile, median, 75th percentile, and maximum). `summarize` *varlist*`, detail` will compute the required values (along with many others), but present them in a style that consumes about 15 lines of output for each variable.

As another example, `summarize` typically shows the mean and standard deviation in a `g` format that displays 7 significant digits. This can be desirable, but not when the goal is to extract, visually, or by cut-and-paste, the means and standard deviations of several variables for inclusion in a written report; then, one might prefer those values to be aligned on their decimal points, followed by a small, fixed number of decimal places.

Of course, there are several other ways of displaying univariate summaries in Stata. For example, the `table` command (new in Stata 5.0) provides very general and flexible formatting of tables of summaries, though it is not especially convenient for interactive data analysis. This insert presents a new command (`univar`) that offers a streamlined display of univariate summaries including, optionally, text-mode boxplots.

Syntax

`univar` *varlist* [*weight*] [`if` *exp*] [`in` *range*] [`,` `boxplot` `byvar(`*bylist*`)` `dec(`*#*`)` `fmt({f|g})` `lstwise` `se`]

As with `summarize`, `aweight`s and `fweight`s are allowed.

Before explaining the options, we first demonstrate the default behavior of `univar` using the data set `kcal.dta` supplied by Clayton and Hills (1995):

```
. use kcal, replace
Heart disease and diet survey

. univar toteng height weight
                                        -------------- Quantiles --------------
Variable       n     Mean     S.D.      Min      .25      Mdn      .75      Max
-------------------------------------------------------------------------------
  toteng     337  2828.87   441.75  1748.43  2536.69  2802.98  3109.66  4395.75
  height     332   173.36     6.41   152.40   168.91   172.97   177.80   190.50
  weight     333    72.40    11.28    10.43    64.64    72.80    79.83   106.14
-------------------------------------------------------------------------------
```

By contrast, the default response of `summarize` is

```
. summarize toteng height weight
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
  toteng |     337    2828.872   441.7528    1748.43    4395.75
  height |     332    173.3642   6.409235      152.4      190.5
  weight |     333    72.39953    11.2766   10.43273   106.1417
```

Two differences are apparent: `univar` shows the complete five-number summary in horizontal style, and prints results in `f` rather than `g` format. The latter choice displays the values aligned on their decimal points with a fixed number of decimal places, the conventional style of presenting numbers in text. The former choice permits a more compact and intuitive presentation of five-number summaries than the `detail` option of `summarize`:

```
. summarize toteng, detail
                 total energy (kcals/day)
-------------------------------------------------------------
      Percentiles      Smallest
 1%      1876.13        1748.43
 5%      2168.86        1854.02
10%      2311.24         1858.8       Obs                 337
25%      2536.69        1876.13       Sum of Wgt.         337

50%      2802.98                      Mean           2828.872
                        Largest       Std. Dev.      441.7528
75%      3109.66        4063.02
90%      3366.61        4234.06       Variance       195145.5
95%      3595.05        4256.81       Skewness       .4430436
99%      4063.02        4395.75       Kurtosis       3.506768
```

Options

`boxplot` draws a text-mode boxplot for each *varlist* variable. Stata can, of course, draw boxplots using the **`graph`** command, but a somewhat coarser boxplot built of text characters can still be helpful, especially if displayed beside the numerical values being portrayed.

`byvar(`*bylist*`)` requests summaries at each unique set of values of the variables in *bylist*, which is analogous to attaching the `by` *bylist*`:` prefix to the **`summarize`** command.

`dec(`*#*`)` controls the number of decimal places used to display the summary values; for example, `dec(4)` switches to `%8.4f` format. By default, **`univar`** uses `%8.2f` format to display all values except the number of observations n.

`fmt` chooses between `f` and `g` output formats; for example, supplying the options `dec(0)` and `fmt(g)` chooses the format `%8.0g` for the summary values, a style similar to the default output of **`summarize`**.

`lstwise` requests listwise deletion of missing values (an observation is ignored if any of the *varlist* variables is missing); the default is to use all available observations for each variable (variable-wise deletion).

`se` requests that the standard error of the mean ($s/\sqrt{n}$) be printed in place of the sample standard deviation (s).

Examples

To illustrate, we continue looking at the data in `kcal.dta`:

```
. univar toteng height weight, box by(job loweng)
-> job=0 loweng=0
                                        ----------:::::::::|:::::::::----------
Variable       n     Mean     S.D.      Min      .25      Mdn      .75      Max
-------------------------------------------------------------------------------
          -----:::::::::::|::::::::::------------------------------------------
  toteng      54  3155.26   343.26  2755.00  2873.95  3114.83  3336.00  4256.81
          --------------------------:::::::::::::|:::::::----------------------
  height      54   172.28     5.36   157.73   167.89   172.72   175.51   184.15
          -----------::::::::::::::::|::::::-----------------------------------
  weight      54    73.91     9.95    57.20    64.64    75.18    78.70   101.83
-------------------------------------------------------------------------------
```

(*output for four bygroups omitted*)

```
-> job=2 loweng=1
                                        ----------:::::::::|:::::::::----------
Variable       n     Mean     S.D.      Min      .25      Mdn      .75      Max
-------------------------------------------------------------------------------
          ------------------------------------------::::::::::|::::::::--------
  toteng      68  2463.41   225.89  1748.43  2365.93  2512.83  2631.58  2749.70
          ----------------------------------::::::::|:::::::-------------------
  height      64   175.77     7.43   153.67   172.08   176.53   180.34   190.50
          ----------------------------------------::::|:::::-------------------
  weight      64    72.96    13.17    10.43    66.68    72.92    79.22   106.14
-------------------------------------------------------------------------------
```

The above command produces a summary for each of the six combinations of `job` and `loweng`, and draws a boxplot for each five-number summary calculated. The boxplots map the range of each variable onto a fixed width in the output; the median is drawn with the character "`|`", the remainder of the box with "`:`", and the whiskers with "`-`". **`univar`** also draws a glyph at the top of each summary table to serve as a reminder of this representation.

Adding the `se` and `lstwise` options then gives this result:

```
. univar toteng height weight, box by(j low) lst se

-> job=0 loweng=0
                                        ----------:::::::::|:::::::::----------
Variable       n     Mean     S.E.      Min      .25      Mdn      .75      Max
-------------------------------------------------------------------------------
          -----:::::::::::|::::::::::------------------------------------------
  toteng      54  3155.26    46.71  2755.00  2873.95  3114.83  3336.00  4256.81
          --------------------------:::::::::::::|:::::::----------------------
  height      54   172.28     0.73   157.73   167.89   172.72   175.51   184.15
          -----------::::::::::::::::|::::::-----------------------------------
  weight      54    73.91     1.35    57.20    64.64    75.18    78.70   101.83
-------------------------------------------------------------------------------
```

(*output for four bygroups omitted*)

```
-> job=2 loweng=1
                                        ----------:::::::::|:::::::::----------
Variable       n     Mean     S.E.      Min      .25      Mdn      .75      Max
-------------------------------------------------------------------------------
          ------------------------------------------::::::::::|::::::::--------
  toteng      64  2467.44    28.62  1748.43  2372.61  2516.92  2631.58  2749.70
          ----------------------------------::::::::|:::::::-------------------
  height      64   175.77     0.93   153.67   172.08   176.53   180.34   190.50
          ----------------------------------------::::|:::::-------------------
  weight      64    72.96     1.65    10.43    66.68    72.92    79.22   106.14
-------------------------------------------------------------------------------
```

Remarks

1. The `byvar` option differs from the `by:` prefix available to `summarize` in that the data need not be sorted by the *bylist* variables; `univar` sorts the data as necessary and then restores the original ordering before exiting. In addition, the `byvar` option may be combined with an `in` clause, whereas the `by:` prefix may not.
2. However, `univar` uses the `by:` prefix to implement the `byvar` option. Hence, the *bylist* can have at most ten variables, which may be of either numeric or string type, and may include missing values or null strings.
3. `univar` runs on Stata Version 4.0, but Version 5.0 or newer is required to display value labels for *bylist* variables.
4. `univar`'s output is 79 characters wide, the same width used by Stata's `.hlp` files.
5. The characters "`|`", "`:`", and "`-`" will produce reasonable boxplots in most fixed pitch fonts. However, these characters are set by local macros at the top of the file `univar.ado`, and are easily redefined, if desired. A similar comment applies to the color used to draw the boxplots.

Acknowledgment

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

Reference

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

sg68 Goodness-of-fit statistics for multinomial distributions

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

This insert describes a command `gof` that computes goodness-of-fit statistics for multinomial distributed observations (obs) and expected values (exp). The expected values may be derived from an estimated model for the multinomial probability distribution, e.g., a loglinear model, or be strictly theoretical. The gof-statistics are the members of the 1-parameter Cressie–Read family (1984, 1988) $\mathrm{CR}(\mathrm{obs}, \mathrm{exp}, \lambda)$ of discrepancy measures,

$$\mathrm{CR}(\mathrm{obs}, \mathrm{exp}, \lambda) = \frac{2}{\lambda(\lambda+1)} \sum \mathrm{obs}[(\mathrm{obs}/\mathrm{exp})^{\lambda} - 1]$$

where the summation is over all "cells," i.e., the "response categories" of the multinomial distribution.

The Cressie–Read family contains many well known goodness-of-fit statistics as special cases. For instance, Pearson's χ^2,

$$\chi^2 = \sum (\mathrm{obs} - \mathrm{exp})^2/\mathrm{exp}$$

belongs to the Cressie–Read family with $\lambda = 1$. The deviance or likelihood-ratio statistic LR,

$$\mathrm{LR} = 2 \sum \mathrm{obs} \times \log(\mathrm{obs}/\mathrm{exp})$$

is embedded in the Cressie–Read family as the (continuous) limiting value in $\lambda = 0$. Other measures such as Freeman–Tukey's statistic ($\lambda = -1/2$), the Kullback–Leibler information distance (entropy) ($\lambda = -1$), and Neyman's modified χ^2 ($\lambda = -2$; note that in modern terminology we would refer to Neyman's χ^2 as a Wald statistic) are similarly special cases of the general form. Finally, Cressie and Read's recommended statistic ($\lambda = 2/3$) is obviously a member of the family.

If the expected values (exp) are true or at least efficiently estimated, all members of the CR family are asymptotically (central) χ^2 distributed. Under standard regularity conditions, the degrees of freedom of $\mathrm{CR}(\,.\,,\,.\,,\lambda)$ can be expressed as "the number of cells $-$ 1" for theoretical expected values and "the number of cells $-$ 1 $-$ the number of imposed restrictions" for estimated expected values. Thus, all CR statistics are first-order efficient. Based on a.o. higher-order asymptotic developments and Monte Carlo experimentation, Cressie and Read (1988) recommend the application of a nonstandard statistic, $\mathrm{CR}(\,.\,,\,.\,,\frac{2}{3})$.

Syntax

The syntax of `gof` is

`gof` *obs exp* [`if` *exp*] [`in` *range*] [`, x2 cr g2 lr ft kl x2n lambda(`*numeric_list*`)`
`by(`*varlist*`) df(`*#*`) list n(`*#*`) plot scale saving(`*filename*`)`]

Note: The `lambda()` option requires the `numlist` program which must be installed from the ip14 directory of the STB 35 disk (January 1997).

Options to select lambda

`x2 cr g2 lr ft kl x2n` specify the members of the Cressie–Read family to be displayed (`g2` and `lr` are synonymous). More than one of these options may be specified. Formally,

λ	description	abbreviation	formula
1.0	Pearson's χ^2	`x2`	$\sum(\mathrm{obs}-\mathrm{exp})^2/\mathrm{exp}$
2/3	C & R's recommended statistic	`cr`	$\frac{9}{5}\sum \mathrm{obs}((\mathrm{obs}/\mathrm{exp})^{2/3}-1)$
0.0	log-likelihood ratio (deviance)	`lr` or g2	$2\sum \mathrm{obs}\times\log(\mathrm{obs}/\mathrm{exp})$
−0.5	Freeman–Tukey's statistic	`ft`	$4\sum(\sqrt{\mathrm{obs}}-\sqrt{\mathrm{exp}})^2$
−1.0	Kullback–Leibler information	`kl`	$2\sum \mathrm{exp}\times\log(\mathrm{exp}/\mathrm{obs})$
−2.0	Neyman's modified χ^2	`x2n`	$\sum(\mathrm{obs}-\mathrm{exp})^2/\mathrm{obs}$

`lambda(`*numeric_lists*`)` specifies a range of powers for the Cressie–Read statistics. See on-line help for `numlist` (Weesie 1997) for the definition of *numeric_lists*. Note that `numlist` must be installed separately for this option to work.

Specifying none of these options implies all of `x2 cr lr ft x2n`, or, stated differently, it is the same as specifying `lambda(-2 -1 -.5 0 .667 1)`.

Other options

`by(`*varlist*`)` specifies a list of variables on which to aggregate obs and exp (join "cells" with the same values on varlist) before computing goodness-of-fit statistics.

`df(`*#*`)` specifies the degrees of freedom used in the computations of chi-squared-based approximate significance levels.

`list` specifies that the table with statistic values is displayed. This option is effective only in combination with `plot`.

`n(`*#*`)` specifies that the variables obs and exp are expressed as proportions with the total number of observations equal to *#*.

`plot` specifies that a statistic-by-lambda plot is displayed. If `df` is specified, horizontal lines at the 90%, 95%, and 99% critical values of the (central) chi-squared-distribution is shown.

`scale` specifies that the expected values may be scaled so that they sum to the number of observations. Otherwise obs and exp should have equal sums (within a .001 multiplicative margin).

`saving(`*filename*`)` specifies the name of a file to save the statistic-by-lambda plot.

Examples

We have estimated a model with 8 parameters, including the constant, on the data in the accompanying file `gof.dta`. The data are assumed to follow a multinomial distribution. In this case, we estimated a loglinear model in GLM. The data and estimated expected counts and a variable `region`, to be used below, are

```
. list obs exp region
          obs        exp     region
  1.        7   2.484035          1
  2.       10   10.51335          1
  3.       11   12.43314          2
  4.       10    12.9811          2
  5.       17   14.66245          3
(output omitted)
 39.       14   14.14302         20
 40.        6   5.875558         20
```

The deviance (likelihood ratio statistic against the saturated model) can be obtained as

```
. gof obs exp, df(32) lr
Cressie-Read multinomial goodness-of-fit (32 df; 40 cells)
     %cells(exp<1) =  0.075    %cells(exp<5) =  0.225
     %cells(obs<1) =  0.000    %cells(obs<5) =  0.250
L( 0.00)             LR =    37.945   p = 0.2166
```

Note that the output of `gof` contains the proportions of cells with low observed and expected counts.

To obtain the default list of statistics, we run

```
. gof obs exp, df(32)
Cressie-Read multinomial goodness-of-fit (32 df; 40 cells)
L(-2.00)      Neyman´s X2 =    53.612   p = 0.0097
L(-1.00)    Kullback´s KL =    40.671   p = 0.1399
L(-0.50)    Freeman-Tukey =    38.436   p = 0.2009
L( 0.00)               LR =    37.945   p = 0.2166
L( 0.67)      Cressie-Read =    39.638   p = 0.1661
L( 1.00)     Pearson´s X2 =    41.535   p = 0.1206
```

Note that the p values of the statistics vary. Using Neyman's χ^2 we would reject the model with expected values `exp` at any significance level below 1%. With the other statistics, we would not reject the model. These conflicting conclusions are somewhat disturbing. We are concerned that some of the conclusions that we want to draw are not very robust with respect to (1) auxiliary assumptions, such as the link-function in a GLM model, or (2) arbitrary decisions, such as the selection of test-statistic in a class with similar asymptotic properties with little known about small-sample properties. Of course, the program `gof` can be abused to shop around for a statistic that "proves" whatever we want to do. It is clear that such an application of `gof` has nothing to do with good statistics or good science.

It is possible to collapse cells on a variable before computing the goodness-of-fit statistics with the `by` option. A relatively high proportion of cells with low expected counts is often a reason to collapse cells. Note that you have to manually modify the appropriate degrees of freedom.

```
. gof obs exp, df(12) by(reg)
Cressie-Read multinomial goodness-of-fit (12 df; 20/40 cells)
l(-2.00)     Neyman´s X2 =    17.302   p = 0.1386
l(-1.00)   Kullback´s KL =    16.954   p = 0.1513
l(-0.50)   Freeman-Tukey =    16.910   p = 0.1530
l( 0.00)              LR =    16.952   p = 0.1514
l( 0.67)    Cressie-Read =    17.140   p = 0.1444
l( 1.00)    Pearson´s X2 =    17.292   p = 0.1389
```

Finally, it is often quite convenient to inspect the Cressie–Read family in a statistic-by-lambda plot. This plot, containing the critical significance levels at 90%, 95% and 99%, is obtained via the `plot` option

```
. gof obs exp, df(32) lambda(-2.2-2.21/0.1) plot
Cressie-Read multinomial goodness-of-fit (32 df; 40 cells)
```

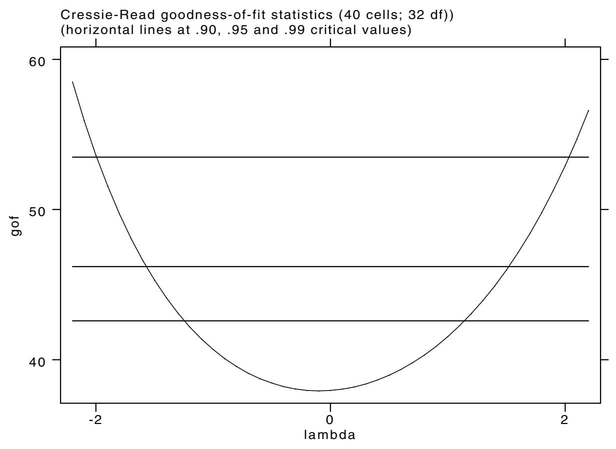


Figure 1. Cressie–Read goodness-of-fit statistics.

Acknowledgment

The command `gof` is an extensive update of an early version in the ETS Kit, a collection of Stata commands written for Stata 2.1 by the late Albert Verbeek, Professor of Statistics at the Department of Social Sciences at Utrecht University, and myself.

References

Cressie, N. A. C. and T. R. C. Read. 1984. Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society*, *Series B* 46: 440–464.

Read, T. R. C. and N. A. C. Cressie. 1988. *Goodness-of-Fit Statistics for Discrete Multivariate Data*. New York: Springer Verlag.

Weesie, J. 1997. ip14: Programming utility: Numeric lists. *Stata Technical Bulletin* 35: 14–16.

sg69 Immediate Mann–Whitney and binomial effect-size display

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

A perennial complaint of users of statistics is that the results are not "meaningful" in the real world. The two programs presented here are the first of what will eventually be several inserts presenting translations from statistical tests to what may be more meaningful indices.

The first index presented, the Mann–Whitney score, has previously been presented in `ranksum2` (Goldstein 1994). The presentation there was a simple reporting of part of what makes up the test. Here, we expand the use of the score to many other tests, including pre-post studies, comparisons of proportions, matched-pairs t tests, and independent groups t tests. This statistic is only presented as an immediate statistic, as it is meant to be used directly after some statistical test or procedure.

The interpretation of this score is that it shows the proportion of pairs in which cases of one type (e.g., one group such as the experimental versus the control group, males versus females, etc.) that have a higher value for the dependent variable than do cases of the other type. Pairs are simply defined as the number of people in group 1 times the number of people in group 2; for example, if there are 15 people in the experimental group and 19 in the control group, there are $285 = 15 \times 19$ pairs. Then, a value of 0.27 means that in 27% of those 285 pairs, the member of the experimental group has more of something (e.g., dollars, pain, days in hospital, etc.) than does the member of the control group. It also means, for a randomly chosen pair, the probability is 0.27 that the person from the experimental group will have a higher value on the dependent variable than will the person from the control group. This value is the same as the area under the ROC curve as shown in `ranksum2`.

The second measure presented, the binomial effect-size display, translates the results of any correlation, chi-squared, t test, or F test to a simple effect size.

Immediate Mann–Whitney Statistic

`mwstati` $\#_1$ $\#_2$ $[\#_3]$ [`, pp` `proportion` `matched` `independent`]

gives the Mann–Whitney statistic ("probability that a randomly selected member of one group will have a better result than a randomly selected member of the other group") for the following cases.

Options

`pp`, pre-post studies, $\#_2$ is the total number of subjects in the study while $\#_1$ is the number of subjects who improved; thus, if there were 30 subjects and 22 of them improved, the user would enter `mwstati 22 30, pp`.

`proportion`, comparisons of proportions, $\#_1$ is proportion of "treatment" group who improve, while $\#_2$ is proportion of "control" group who improve; that is, enter, for each group, the proportion who had whatever event is of interest (whatever the event is; e.g., promotion, termination, survival, remission, relapse, etc.); these proportions are easily available from Stata's tabulate command by asking for column or row percentages (whichever is appropriate in your setup); for example: `mwstati 15 18, proportion`.

`matched`, matched pairs t tests, $\#_1$ is average difference, while $\#_2$ is standard deviation of differences; this information is provided in the standard output from Stata's `ttest` command.

`independent`, independent groups t tests, $\#_1$ is difference of means; $\#_2$ is variance for one group, while $\#_3$ is variance for other group. Note that the above asks for variances, while the `ttest` shows standard deviations (the variance is just the square of the standard deviation).

Note that this statistic is given at the very end of `ranksum2` for a nonparametric comparison (via the `ranksum` test).

Remarks

In addition to its use in helping to interpret statistical results, this statistic can also be used, with caution, to "adjust" the results from studies that are of lower quality than desired. For example, Colditz et al. (1989) suggest that studies that use sequential assignment, rather than random assignment, should have their Mann–Whitney statistics reduced by 0.15 and that non-double-blind randomized studies should be decreased by 0.11. Another example (Colditz et al. 1988a) is that when there is a standard therapy but the control group is a placebo group, the Mann–Whitney statistic should be decreased by 0.10!

In other words, studies using sequential assignment have been shown to be biased (as compared with randomized studies) and the amount of bias is about 15%; that is, non-randomized studies tend to overestimate the proportion of pairs in which one group does better by about 15%, compared with randomized studies. Similarly, use of a placebo arm in a study, when there is a standard therapy, tends to result in a bias of about 10%. Note that these percentages are based on an examination of a number of studies in certain medical areas from the 1980s; the results might differ in other areas (e.g., schizophrenia) or at other times.

The formulas used are from Colditz et al. (1988b); in that article they also present a formula for obtaining a combined score across several measures, weighting each by the inverse of their standard deviations. This requires the sample size for each group.

Examples of the use of the Mann–Whitney test statistic

The first example is from the Stata Manual (and is the same as the second example used for BESD, below). The only difference between the first two examples is the sign of the statistic, with the results summing to 1.0.

```
. mwstati -1.75 0.7797, m
Mann-Whitney statistic for this situation is: 0.0124
```

We would interpret this to mean that in only 1% of the with-and-without treatment pairs would the without treatment car have greater mileage. Since there are only two groups, this implies that in about 99% of the pairs, the car with treatment would have greater mileage; we can check on this by entering the same information as above, except that the negative sign on the t statistic is left off:

```
. mwstati 1.75 0.7797, m
Mann-Whitney statistic for this situation is: 0.9876
```

The next example is also from the Stata Manual.

```
. mwstati 1.75 .6209 .8798, i
Mann-Whitney statistic for this situation is: 0.9234
```

This final example shows the same information, but now using the separate variances. The Mann–Whitney score goes down from about 99% to about 92% showing some effect of our assumption of equal variances.

Binomial effect-size display

besd *test_value* $\left[\ \#_n\ \#_{df}\ \right]$ [, **c**hi **t** **f**]

If there is no option then the first argument should be the correlation and nothing else is needed.

Many people use statistics that appear to have obvious meanings, such as the correlation coefficient, or an effect size (e.g., the difference in means divided by the common standard deviation). However, it is not always obvious how "important" the value of these is in the real world.

"The BESD displays the change in success rate (e.g., survival rate, improvement rate, etc.) attributable to a new treatment procedure. For example, an r of .32 ... is said to account for "only 10% of the variance"; however, the BESD shows that this proportion of variance accounted for is equivalent to increasing the success rate from 34% to 66%, which would mean, for example, reducing an illness rate or a death rate from 66% to 34%." (Rosenthal and Rubin 1982, 166).

Examples of the use of the BESD

The first example is from Rosenthal and Rubin (1982, 167):

```
. besd .32
Difference in ´success´ rates = 0.320
´Experimental success´ rate = 0.660  & ´control success´ rate = 0.340
```

Thus, a correlation of 0.32 is equivalent to increasing the success rate from 34% to 66%, certainly an important accomplishment.

The next example is from the Stata Manual's entry on [R] **ttest**.

```
. besd 2.244 11, t
Difference in ´success´ rates = 0.560
´Experimental success´ rate = 0.780  & ´control success´ rate = 0.220
```

In this example, we see that a relatively small t statistic implies a large real-world difference in success rates, one that would make most of us ecstatic.

References

Colditz, G. A., J. N. Miller, and F. Mosteller. 1988a. The effect of study design on gain in evaluations of new treatments in medicine and surgery. *Drug Information Journal* 22: 343–352.

Colditz, G. A., J. N. Miller, and F. Mosteller. 1988b. Measuring gain in the evaluation of medical technology. *International Journal of Technology Assessment* 4: 637–42.

Colditz, G. A., J. N. Miller, and F. Mosteller. 1989. How study design affects outcomes in comparisons of therapy, I: Medical. *Statistics in Medicine* 8: 441–54.

Goldstein, R. 1994. The overlapping coefficient and an "improved" rank-sum statistic. *Stata Technical Bulletin* 22: 12–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 132–136.

Rosenthal, R. and D. B. Rubin. 1982. A simple, general purpose display of the magnitude of experimental effect. *Journal of Educational Psychology* 74: 166–169.

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.

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 66554254
Fax:	+972 66554254
Email:	sasconsl@actcom.co.il
Countries served:	Israel

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

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

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

Company:	Smit Consult
Address:	Doormanstraat 19 Postbox 220 5150 AE Drunen Netherlands
Phone:	+31 416-378 125
Fax:	+31 416-378 385
Email:	j.a.c.m.smit@smitcon.nl
Countries served:	Netherlands

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

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