Title

stata.com

***advanced_options* —** Rarely specified options for use with graph twoway

Description Syntax Options Remarks and examples Also see

Description

The *advanced_options* are not so much advanced as they are difficult to explain and are rarely used. They are also invaluable when you need them.

Syntax

title_options	Description
`pcycle(#)`	plots before pstyles recycle
`yvarlabel(`*quoted_strings*`)`	respecify y-variable labels
`xvarlabel(`*quoted_string*`)`	respecify x-variable label
`yvarformat(`*%fmt* [...]`)`	respecify y-variable formats
`xvarformat(`*%fmt*`)`	respecify x-variable format
`yoverhangs`	adjust margins for y-axis labels
`xoverhangs`	adjust margins for x-axis labels
`recast(`*newplottype*`)`	treat plot as *newplottype*

The above options are *rightmost*; see [G-4] **concept: repeated options**.

where *quoted_string* is one quoted string and *quoted_strings* are one or more quoted strings, such as

"*plot 1 label*"

"*plot 1 label*" "*plot 2 label*"

newplottype	Description
scatter	treat as graph twoway scatter
line	treat as graph twoway line
connected	treat as graph twoway connected
bar	treat as graph twoway bar
area	treat as graph twoway area
spike	treat as graph twoway spike
dropline	treat as graph twoway dropline
dot	treat as graph twoway dot
rarea	treat as graph twoway rarea
rbar	treat as graph twoway rbar
rspike	treat as graph twoway rspike
rcap	treat as graph twoway rcap
rcapsym	treat as graph twoway rcapsym
rline	treat as graph twoway rline
rconnected	treat as graph twoway rconnected
rscatter	treat as graph twoway rscatter
pcspike	treat as graph twoway pcspike
pccapsym	treat as graph twoway pccapsym
pcarrow	treat as graph twoway pcarrow
pcbarrow	treat as graph twoway pcbarrow
pcscatter	treat as graph twoway pcscatter

newplottypes in each grouping (scatter through dot, rarea though rscatter, and pcspike through pcscatter) should be recast only among themselves.

Options

pcycle(#) specifies how many plots are drawn before the pstyle (see [G-4] ***pstyle***) of the next plot begins again at p1, with the plot after the next plot using p2, and so on. The default # for most schemes is pcycle(15).

yvarlabel(*quoted_strings*) and xvarlabel(*quoted_string*) specify strings that are to be treated as if they were the variable labels of the first, second, ..., y variables and of the x variable.

yvarformat(%*fmt*) and xvarformat(%*fmt*) specify display formats that are to be treated as if they were the display formats of the first, second, ..., y variables and of the x variable.

yoverhangs and xoverhangs attempt to adjust the graph region margins to prevent long labels on the y or x axis from extending off the edges of the graph. Only the labels for the smallest and largest tick values on the axes are considered when making the adjustment. yoverhangs and xoverhangs are ignored if by() is specified; see [G-3] ***by_option***.

recast(*newplottype*) specifies the new plottype to which the original graph twoway *plottype* command is to be recast; see [G-2] **graph twoway** to see the available *plottype*s.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Use of yvarlabel( ) and xvarlabel( )

When you type, for instance,

```
. scatter mpg weight
```

the axes are titled using the variable labels of `mpg` and `weight` or, if the variables have no variable labels, using the names of the variables themselves. Options `yvarlabel()` and `xvarlabel()` allow you to specify strings that will be used in preference to both the variable label and the name.

```
. scatter mpg weight, yvarl("Miles per gallon")
```

would label the y axis "Miles per gallon" (omitting the quotes), regardless of how variable `mpg` was labeled. Similarly,

```
. scatter mpg weight, xvarl("Weight in pounds")
```

would label the x axis "Weight in pounds", regardless of how variable `weight` was labeled.

Obviously, you could specify both options.

In neither case will the actual variable label be changed. Options `yvarlabel()` and `xvarlabel()` treat the specified strings as if they were the variable labels. `yvarlabel()` and `xvarlabel()` are literal in this treatment. If you specified `xvarlabel("")`, for instance, the variable label would be treated as if it were nonexistent, and thus the variable name would be used to title the x axis.

What makes these two options "advanced" is not only that they affect the way axes are titled but also that they substitute the specified strings for the variable labels wherever the variable label might be used. Variable labels are also used, for instance, in the construction of legends (see [G-3] ***legend_options***).

Use of yvarformat( ) and xvarformat( )

Options `yvarformat()` and `xvarformat()` work much like `yvarlabel()` and `xvarlabel()`, except that, rather than overriding the variable labels, they override the variable formats. If you type

```
. scatter mpg weight, yvarformat(%9.2f)
```

the values on the y axis will be labeled 10.00, 20.00, 30.00, and 40.00 rather than 10, 20, 30, and 40.

Use of recast( )

`scatter`, `line`, `histogram`, ... —the word that appears directly after `graph twoway`—is called a *plottype*. Plottypes come in two forms: *base plottypes* and *derived plottypes*.

Base plottypes plot the data as given according to some style. `scatter` and `line` are examples of base plottypes.

Derived plottypes do not plot the data as given but instead derive something from the data and then plot that according to one of the base plottypes. `histogram` is an example of a derived plottype. It derives from the data the values for the frequencies at certain x ranges, and then it plots that derived data using the base plottype `graph twoway bar`. `lfit` is another example of a derived plottype. It takes the data, fits a linear regression, and then passes that result along to `graph twoway line`.

`recast()` is useful when using derived plottypes. It specifies that the data are to be derived just as they would be ordinarily, but rather than passing the derived data to the default base plottype for plotting, they are passed to the specified base plottype.

For instance, if we typed

```
. twoway lfit mpg weight, pred(resid)
```

we would obtain a graph of the residuals as a line plot because the `lfit` plottype produces line plots. If we typed

```
. twoway lfit mpg weight, pred(resid) recast(scatter)
```

we would obtain a scatterplot of the residuals. `graph twoway lfit` would use `graph twoway scatter` rather than `graph twoway line` to plot the data it derives.

`recast(`*newplottype*`)` may be used with both derived and base plottypes, although it is most useful when combined with derived plots.

❑ Technical note

The syntax diagram shown for `scatter` in [G-2] **graph twoway scatter**, although extensive, is incomplete, and so are all the other plottype syntax diagrams shown in this manual.

Consider what would happen if you specified

```
. scatter ... , ... recast(bar)
```

You would be specifying that `scatter` be treated as a `bar`. Results would be the same as if you typed

```
. twoway bar ... , ...
```

but let's ignore that and pretend that you typed the `recast()` version. What if you wanted to specify the look of the bars? You could type

```
. scatter ... , ... bar_options recast(bar)
```

That is, `scatter` allows `graph twoway bar`'s options, even though they do not appear in `scatter`'s syntax diagram. Similarly, `graph twoway bar` allows all of `scatter`'s options; you might type

```
. twoway bar ..., ... scatter_options recast(scatter)
```

The same is true for all other pairs of base plottypes, with the result that all base plottypes allow all base plottype options. The emphasis here is on base: the derived plottypes do not allow this sharing.

If you use a base plottype without `recast()` and if you specify irrelevant options from other base types, that is not an error, but the irrelevant options are ignored. In the syntax diagrams for the base plottypes, we have listed only the options that matter under the assumption that you do not specify `recast`.

❑

Also see

[G-2] **graph twoway** — Twoway graphs