Title

stata.com

graph display — Display graph stored in memory

Description

`graph display` redisplays a graph stored in memory.

Quick start

Display graph `mygraph` stored in memory

```
graph display mygraph
```

Increase the size of all text, markers, and line widths by 50%

```
graph display mygraph, scale(1.5)
```

Resize `mygraph` to 3 inches by 2 inches

```
graph display mygraph, ysize(2) xsize(3)
```

Apply the *Stata Journal* scheme to the overall look of `mygraph`

```
graph display mygraph, scheme(sj)
```

Same as above, but for the graph currently in the Graph window

```
graph display, scheme(sj)
```

Menu

Graphics > Manage graphs > Make memory graph current

Syntax

```
graph display [name] [, options]
```

If *name* is not specified, the name of the current graph—the graph displayed in the Graph window—is assumed.

options	Description
`ysize(#)`	change height of graph (in inches)
`xsize(#)`	change width of graph (in inches)
`margins(`*marginstyle*`)`	change outer margins
`scale(#)`	resize text, markers, and line widths
`scheme(`*schemename*`)`	change overall look

Options

`ysize(#)` and `xsize(#)` specify in inches the height and width of the entire graph (also known as the *available area*). The defaults are the original height and width of the graph. These two options can be used to change the aspect ratio; see *Changing the size and aspect ratio* under *Remarks and examples* below.

`margins(`*marginstyle*`)` specifies the outer margins: the margins between the outer graph region and the inner graph region as shown in the diagram in [G-3] ***region_options***. See *Changing the margins and aspect ratio* under *Remarks and examples* below, and see [G-4] ***marginstyle***.

`scale(#)` specifies a multiplier that affects the size of all text, markers, and line widths in a graph. `scale(1)` is the default, and `scale(1.2)` would make all text and markers 20% larger. See [G-3] ***scale_option***.

`scheme(`*schemename*`)` specifies the overall look of the graph. The default is the original scheme with which the graph was drawn. See *Changing the scheme* under *Remarks and examples* below, and see [G-3] ***scheme_option***.

Remarks and examples

stata.com

See [G-2] **graph manipulation** for an introduction to the graph manipulation commands.

Remarks are presented under the following headings:

Changing the size and aspect ratio

Under *Controlling the aspect ratio* in [G-3] ***region_options***, we compared

```
. use https://www.stata-press.com/data/r18/auto
(1978 automobile data)
. scatter mpg weight
```

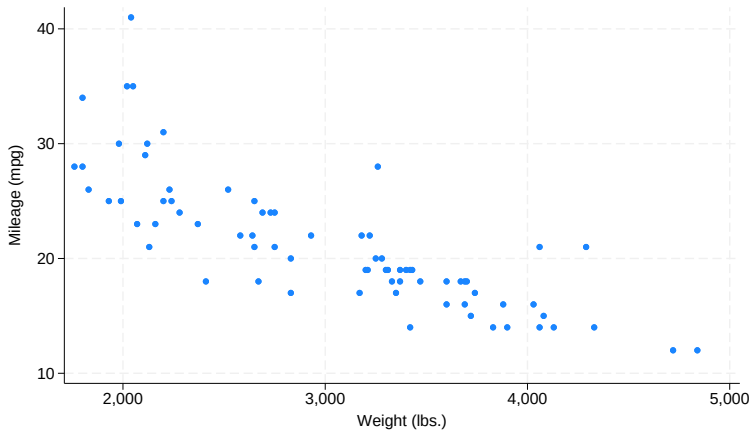
with

```
. scatter mpg weight, ysize(5)
```

We used the scheme `s2gcolor` in that entry, but we will use the default scheme `stgcolor` in the examples below.

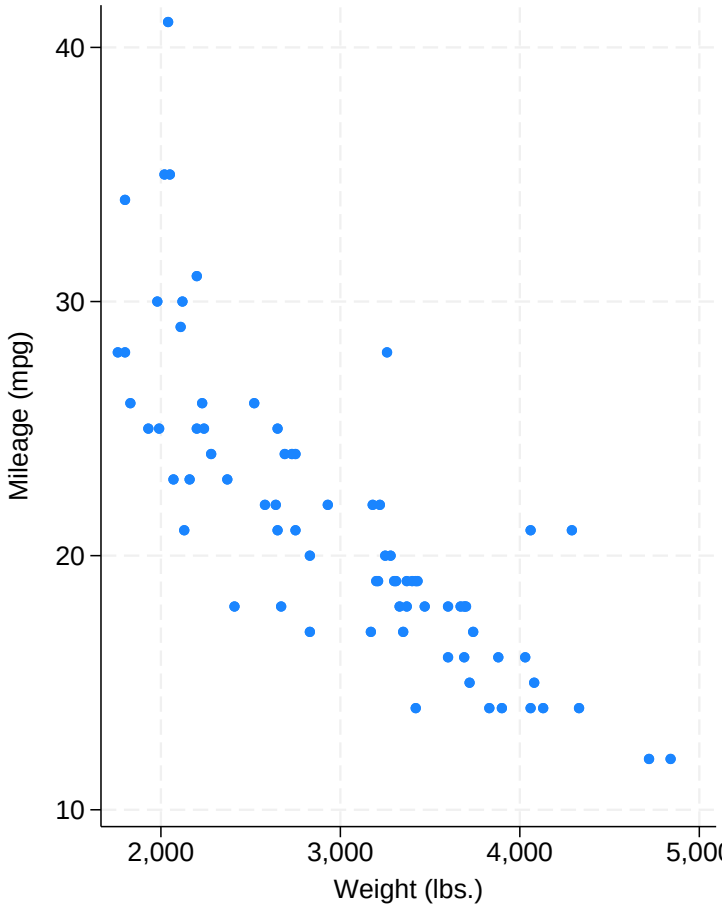
Note that we do not need to reconstruct the graph merely to change the `ysize()` or `xsize()`. We could start with some graph

```
. scatter mpg weight
```



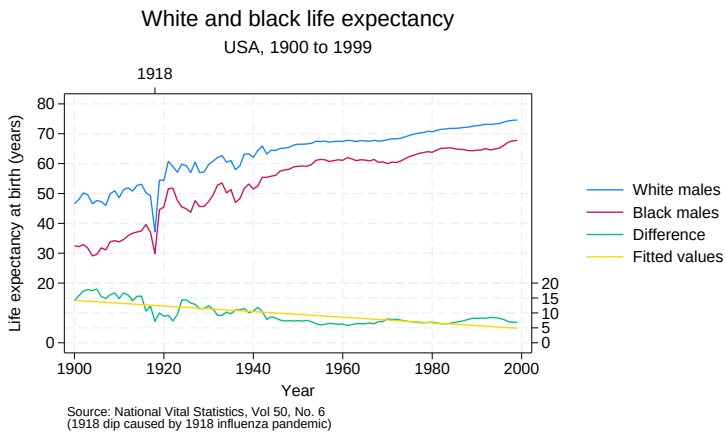
and then we could redisplay it with different `ysize()` and `xsize()` or both values:

```
. graph display, ysize(5)
```

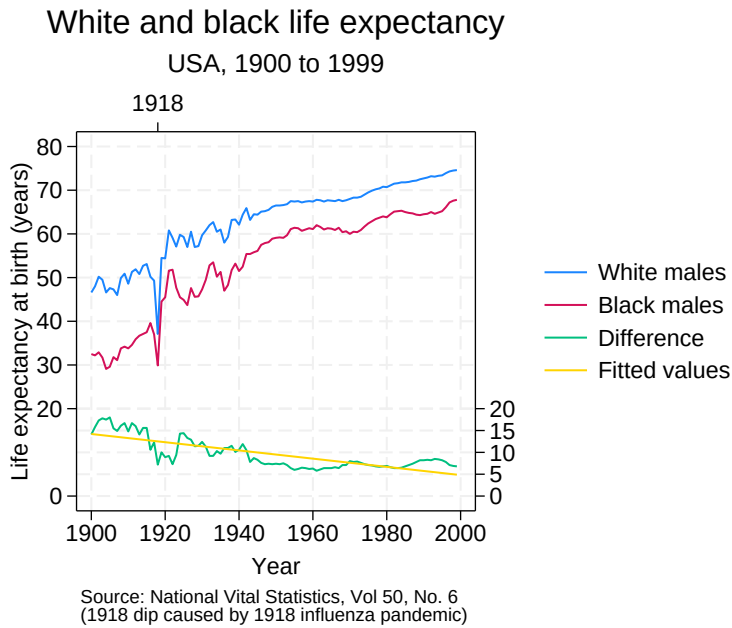


In this way, we can quickly find the best `ysize()` and `xsize()` values. This works particularly well when the graph we have drawn required many options:

```
. use https://www.stata-press.com/data/r18/uslifeexp, clear
(U.S. life expectancy, 1900-1999)
. generate diff = le_wm - le_bm
. label var diff "Difference"
. line le_wm year, yaxis(1 2) xaxis(1 2)
  || line le_bm year
  || line diff  year
  || lfit diff  year
  ||,
     ytitle("",        axis(2))
     xtitle("",        axis(2))
     xlabel(1918,      axis(2))
     ylabel(0(5)20,    axis(2) grid)
     ylabel(0 20(10)80             )
     ytitle("Life expectancy at birth (years)")
     title("White and black life expectancy")
     subtitle("USA, 1900 to 1999")
     note("Source: National Vital Statistics, Vol 50, No. 6"
          "(1918 dip caused by 1918 influenza pandemic)")
     legend(label(1 "White males") label(2 "Black males"))
```



```
. graph display, ysize(3.5)
```



Also, we can change sizes of graphs we have previously drawn and stored on disk:

```
. graph use ...
. graph display, ysize(...) xsize(...)
```

You may not remember what `ysize()` and `xsize()` values were used (they are `ysize(4.5)` and `xsize(7.5)` from the `stcolor` scheme). Then use `graph describe` to describe the file; it reports the `ysize()` and `xsize()` values; see [G-2] **graph describe**.

Changing the margins and aspect ratio

We can change the size of a graph or change its margins to control the aspect ratio; this is discussed in *Controlling the aspect ratio* of [G-3] ***region_options***, which gives the example

```
scatter mpg weight, by(foreign, total graphregion(margin(l+10 r+10)))
```

This too can be done in two steps:

```
. scatter mpg weight, by(foreign, total)
. graph display, margins(l+10 r+10)
```

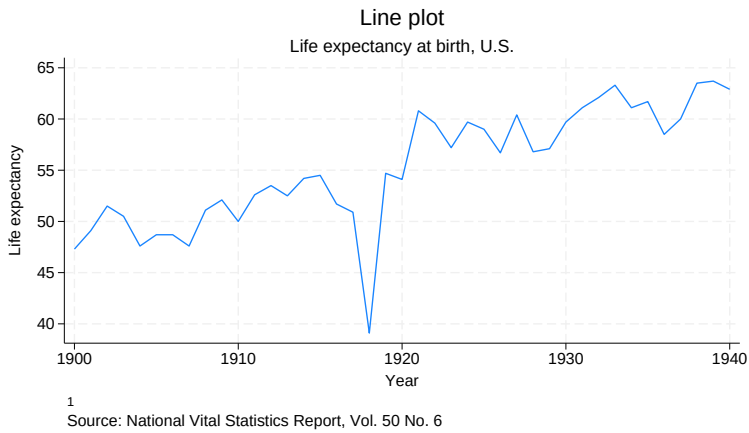
`graph display`'s `margin()` option corresponds to `graphregion(margin())` used at the time we construct graphs.

Changing the scheme

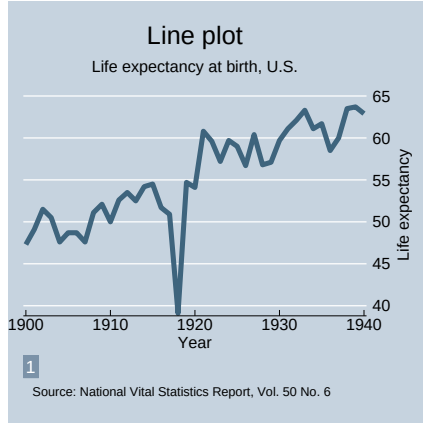
Schemes determine the overall look of a graph, such as where axes, titles, and legends are placed and the color of the background; see [G-4] **Schemes intro**. Changing the scheme after a graph has been constructed sometimes works well and sometimes works poorly.

Here is an example in which it works well:

```
. use https://www.stata-press.com/data/r18/uslifeexp2, clear
(U.S. life expectancy, 1900-1940)
. line le year, sort
        title("Line plot")
        subtitle("Life expectancy at birth, U.S.")
        note("1")
        caption("Source: National Vital Statistics Report,
         Vol. 50 No. 6")
```



```
. graph display, scheme(economist)
```



The above example works well because no options were specified to move from their default location things such as axes, titles, and legends, and no options were specified to override default colors. The issue is simple: if we draw a graph and say, "Move the title from its default location to over here", over here may be a terrible place for the title once we change schemes. Or if we override a color and make it magenta, magenta may clash terribly.

The above does not mean that the graph command need be simple. The example shown in *Changing the size and aspect ratio* above,

```
. line le_wm year, yaxis(1 2) xaxis(1 2)
  || line le_bm year
  || line diff  year
  || lfit diff  year
  ||,
     ylabel(0(5)20, axis(2) gmin angle(horizontal))
     ylabel(0 20(10)80,     gmax angle(horizontal))
     ytitle("", axis(2))
     xlabel(1918, axis(2)) xtitle("", axis(2))
     ytitle("Life expectancy at birth (years)")
     title("White and black life expectancy")
     subtitle("USA, 1900 to 1999")
     note("Source: National Vital Statistics, Vol 50, No. 6"
          "(1918 dip caused by 1918 influenza pandemic)")
     legend(label(1 "White males") label(2 "Black males"))
```

moves across schemes just fine, the only potential problem being our specification of `angle(horizontal)` for labeling the two y axes. That might not look good with some schemes.

If you are concerned about moving between schemes, when you specify options, specify style options in preference to options that directly control the outcome. For example, to have two sets of points with the same color, specify the `mstyle()` option rather than changing the color of one set to match the color you currently see of the other set.

There is another issue when moving between styles that have different background colors. Styles are said to have naturally white or naturally black background colors; see [G-4] **Schemes intro**. When you move from one type of scheme to another, if the colors were not changed, colors that previously stood out would blend into the background and vice versa. To prevent this, `graph display` changes all the colors to be in accordance with the scheme, except that `graph display` does not change colors you specify by name (for example, you specify `mcolor(magenta)` or `mcolor("255 0 255")` to change the color of a symbol).

We recommend that you do not use `graph display` to change graphs from having naturally black to naturally white backgrounds. As long as you print in monochrome, `print` does an excellent job translating black to white backgrounds, so there is no need to change styles for that reason. If you are printing in color, we recommend that you change your default scheme to a naturally white scheme; see [G-2] **set scheme**.

Also see

[G-2] **graph manipulation** — Graph manipulation commands

[G-2] **graph replay** — Replay multiple graphs