Title

stata.com

graph combine — Combine multiple graphs

Description

`graph combine` arrays separately drawn graphs into one.

Quick start

Combine stored graphs named `mygraph1` and `mygraph2` into a single figure
```
graph combine mygraph1 mygraph2
```

Combine graphs `mygraph1.gph` and `mygraph2.gph` that have been saved to disk using `graph save`
```
graph combine "mygraph1" "mygraph2"
```

Same as above
```
graph combine mygraph1.gph mygraph2.gph
```

Graph matrix with `g1` and `g2` in the first row and `g3` and `g4` in the second row
```
graph combine g1 g2 g3 g4
```

As above, but arrange graphs vertically with `g1` and `g2` in the first column and `g3` and `g4` in the second column
```
graph combine g1 g2 g3 g4, colfirst
```

As above, but omit `g4` and leave the second cell of the graph matrix empty
```
graph combine g1 g2 g3, holes(2)
```

Combine graphs `g1`–`g6` into a 3 rows by 2 columns graph matrix
```
graph combine g1 g2 g3 g4 g5 g6, rows(3)
```

Same as above
```
graph combine g1 g2 g3 g4 g5 g6, cols(2)
```

As above, but specify that the y axes of the individual subgraphs have the same scale
```
graph combine g1 g2 g3 g4 g5 g6, cols(2) ycommon
```

As above, and specify a common scale for the x axes of the subgraphs
```
graph combine g1 g2 g3 g4 g5 g6, cols(2) ycommon xcommon
```

As above, but rescale text and markers to half (0.5 times) their original size
```
graph combine g1 g2 g3 g4 g5 g6, cols(2) ycommon xcommon iscale(.5)
```

Use the *Stata Journal* scheme for the common portion of the graph and the subgraphs
```
graph combine g1 g2, scheme(sj) commonscheme
```

Specify that the margin between the subgraphs should be 0
```
graph combine g1 g2 g3 g4, imargin(0 0 0 0)
```

Menu

Graphics > Table of graphs

Syntax

```
graph combine name [ name ... ] [ , options ]
```

name	Description
simplename	name of graph in memory
name`.gph`	name of graph stored on disk
`"`*name*`"`	name of graph stored on disk

options	Description
`colfirst`	display down columns
`rows(#)` | `cols(#)`	display in # rows or # columns
`holes(`*numlist*`)`	positions to leave blank
`iscale(`[`*`]`#)`	size of text and markers
`altshrink`	alternate scaling of text, etc.
`imargin(`*marginstyle*`)`	margins for individual graphs
`ycommon`	give y axes common scales
`xcommon`	give x axes common scales
title_options	titles to appear on combined graph
region_options	outlining, shading, aspect ratio
`commonscheme`	put graphs on common scheme
`scheme(`*schemename*`)`	overall look
`nodraw`	suppress display of combined graph
`name(`*name*`, ...)`	specify name for combined graph
`saving(`*filename*`, ...)`	save combined graph in file

Options

`colfirst`, `rows(#)`, `cols(#)`, and `holes(`*numlist*`)` specify how the resulting graphs are arrayed. These are the same options described in [G-3] ***by_option***.

`iscale(#)` and `iscale(*#)` specify a size adjustment (multiplier) to be used to scale the text and markers used in the individual graphs.

By default, `iscale()` gets smaller and smaller the larger is G, the number of graphs being combined. The default is parameterized as a multiplier f(G)—$0 < \text{f}(G) < 1$, $\text{f}'(G) < 0$—that is used to multiply `msize()`, {`y`|`x`}`label(,labsize())`, etc., in the individual graphs.

If you specify `iscale(#)`, the number you specify is substituted for f(G). `iscale(1)` means that text and markers should appear the same size that they were originally. `iscale(.5)` displays text and markers at half that size. We recommend that you specify a number between 0 and 1, but you are free to specify numbers larger than 1.

If you specify `iscale(*#)`, the number you specify is multiplied by f(G), and that product is used to scale the text and markers. `iscale(*1)` is the default. `iscale(*1.2)` means that text and markers should appear at 20% larger than `graph combine` would ordinarily choose. `iscale(*.8)` would make them 20% smaller.

`altshrink` specifies an alternate method of determining the size of text, markers, line thicknesses, and line patterns. The size of everything drawn on each graph is as though the graph were drawn at full size, but at the aspect ratio of the combined individual graph, and then the individual graph and everything on it were shrunk to the size shown in the combined graph.

`imargin(`*marginstyle*`)` specifies margins to be put around the individual graphs. See [G-4] ***marginstyle***.

`ycommon` and `xcommon` specify that the individual graphs previously drawn by `graph twoway`, and for which the `by()` option was not specified, be put on common y or x axes scales. See *Combining twoway graphs* under *Remarks and examples* below.

These options have no effect when applied to the categorical axes of `bar`, `box`, and `dot` graphs. Also, when `twoway` graphs are combined with `bar`, `box`, and `dot` graphs, the options affect only those graphs of the same type as the first graph combined.

title_options allow you to specify titles, subtitles, notes, and captions to be placed on the combined graph; see [G-3] ***title_options***.

region_options allow you to control the aspect ratio, size, etc., of the combined graph; see [G-3] ***region_options***. Important among these options are `ysize(#)` and `xsize(#)`, which specify the overall size of the resulting graph. It is sometimes desirable to make the combined graph wider or longer than usual.

`commonscheme` and `scheme(`*schemename*`)` are for use when combining graphs that use different schemes. By default, each subgraph will be drawn according to its own scheme.

`commonscheme` specifies that all subgraphs be drawn using the same scheme and, by default, that scheme will be your default scheme; see [G-4] **Schemes intro**.

`scheme(`*schemename*`)` specifies that the *schemename* be used instead; see [G-3] ***scheme_option***.

`nodraw` causes the combined graph to be constructed but not displayed; see [G-3] ***nodraw_option***.

`name(`*name*`[ , replace ])` specifies the name of the resulting combined graph. `name(Graph, replace)` is the default. See [G-3] ***name_option***.

`saving(`*filename*`[ , asis replace ])` specifies that the combined graph be saved as *filename*. If *filename* is specified without an extension, `.gph` is assumed. `asis` specifies that the graph be saved in as-is format. `replace` specifies that, if the file already exists, it is okay to replace it. See [G-3] ***saving_option***.

Remarks and examples

stata.com

Remarks are presented under the following headings:

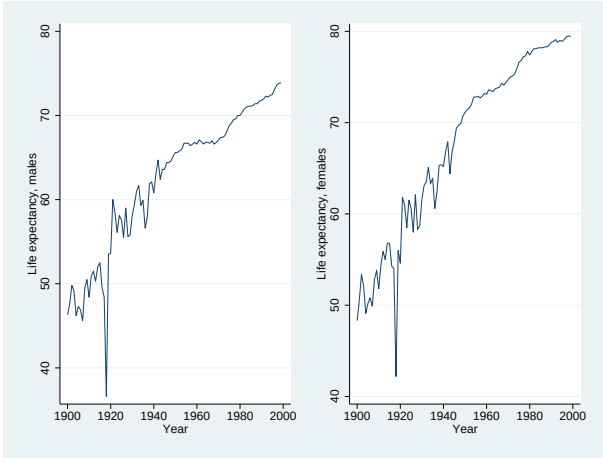
Typical use

We have previously drawn

```
. use https://www.stata-press.com/data/r17/uslifeexp
(U.S. life expectancy, 1900-1999)
. line le_male   year, saving(male)
. line le_female year, saving(female)
```

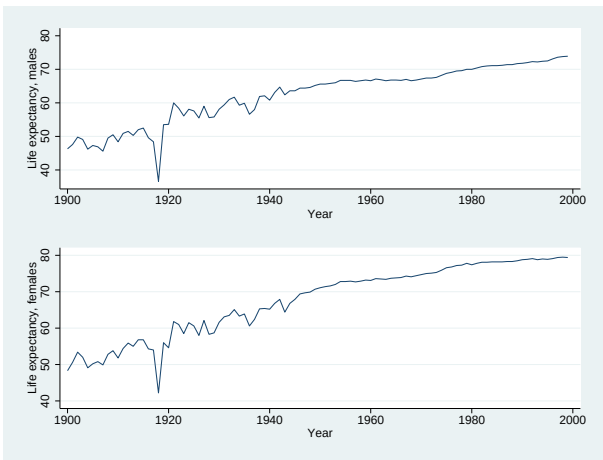
We now wish to combine these two graphs:

```
. gr combine male.gph female.gph
```



This graph would look better combined into one column and if we specified `iscale(1)` to prevent the font from shrinking:

```
. gr combine male.gph female.gph, col(1) iscale(1)
```



Typical use with memory graphs

In both the above examples, we explicitly typed the `.gph` suffix on the ends of the filenames:

```
. gr combine male.gph female.gph
. gr combine male.gph female.gph, col(1) iscale(1)
```

We must do that, or we must enclose the filenames in quotes:

```
. gr combine "male" "female"
. gr combine "male" "female", col(1) iscale(1)
```

If we did neither, `graph combine` would assume that the graphs were stored in memory and would then have issued the error that the graphs could not be found. Had we wanted to do these examples by using memory graphs rather than disk files, we could have substituted `name()` for saving on the individual graphs

```
. use https://www.stata-press.com/data/r17/uslifeexp, clear
(U.S. life expectancy, 1990-1999)
. line le_male   year, name(male)
. line le_female year, name(female)
```

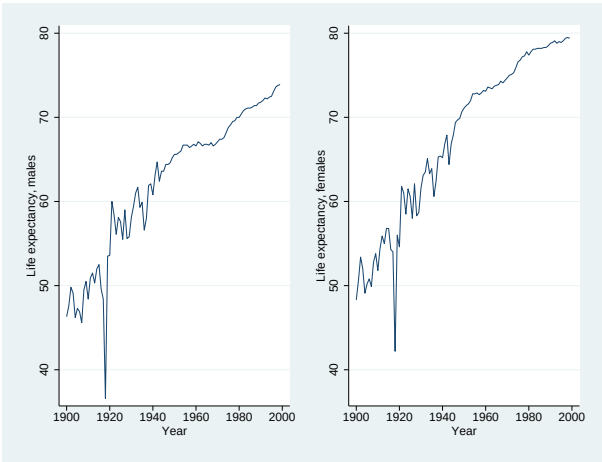
and then we could type the names without quotes on the `graph combine` commands:

```
. gr combine male female
. gr combine male female, col(1) iscale(1)
```

Combining twoway graphs

In the first example of *Typical use*, the y axis of the two graphs did not align: one had a minimum of 40, whereas the other was approximately 37. Option `ycommon` will put all `twoway` graphs on a common y scale.

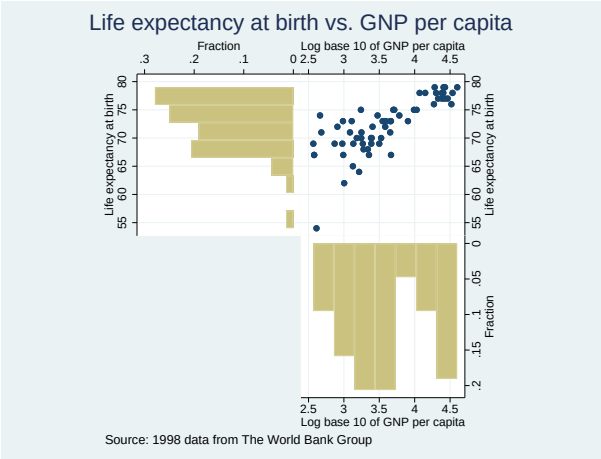
```
. use https://www.stata-press.com/data/r17/uslifeexp, clear
(U.S. life expectancy, 1990-1999)
. line le_male   year, saving(male)
. line le_female year, saving(female)
. gr combine male.gph female.gph, ycommon
```



Advanced use

```
. use https://www.stata-press.com/data/r17/lifeexp, clear
(Life expectancy, 1998)
. generate loggnp = log10(gnppc)
(5 missing values generated)
. label var loggnp "Log base 10 of GNP per capita"
. scatter lexp loggnp,
        ysca(alt) xsca(alt)
        xlabel(, grid gmax)      saving(yx)
```

```
. twoway histogram lexp, fraction
        xsca(alt reverse) horiz  saving(hy)
. twoway histogram loggnp, fraction
        ysca(alt reverse)
        ylabel(,nogrid)
        xlabel(,grid gmax)       saving(hx)
. graph combine hy.gph yx.gph hx.gph,
        hole(3)
        imargin(0 0 0 0) graphregion(margin(l=22 r=22))
        title("Life expectancy at birth vs. GNP per capita")
        note("Source:  1998 data from The World Bank Group")
```



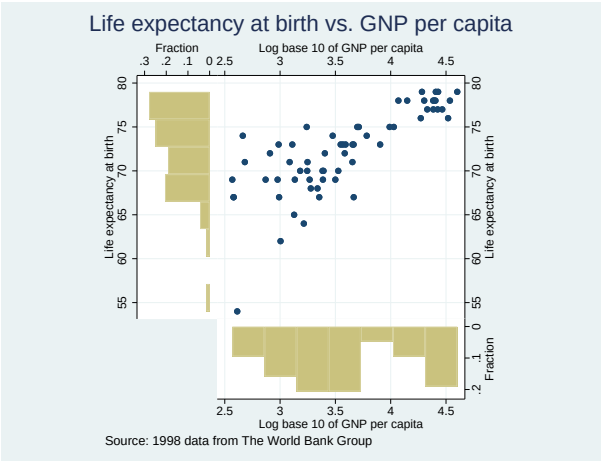
Note the specification of

```
imargin(0 0 0 0) graphregion(margin(l=22 r=22))
```

on the `graph combine` statement. Specifying `imargin()` pushes the graphs together by eliminating the margins around them. Specifying `graphregion(margin())` makes the graphs more square—to control the aspect ratio.

Controlling the aspect ratio of subgraphs

The above graph can be converted to look like this



by adding `fysize(25)` to the drawing of the histogram for the x axis,

```
. twoway histogram loggnp, fraction
        ysca(alt reverse)
        ylabel(0(.1).2, nogrid)
        xlabel(, grid gmax)       saving(hx)
        fysize(25)                                          ← new
```

and adding `fxsize(25)` to the drawing of the histogram for the y axis:

```
. twoway histogram lexp, fraction
        xsca(alt reverse) horiz
                                  saving(hy)
        fxsize(25)                                          ← new
```

The `graph combine` command remained unchanged.

The *forced_size_options* `fysize()` and `fxsize()` are allowed with any graph, their syntax being

forced_size_options	Description
`fysize(`*size*`)`	use only percent of height available
`fxsize(`*size*`)`	use only percent of width available

There are three ways to control the aspect ratio of a graph:

1. Specify the *region_options* `ysize(#)` and `xsize(#)`; # is specified in inches.
2. Specify the *region_option* `graphregion(margin(`*marginstyle*`))`.
3. Specify the *forced_size_options* `fysize(`*size*`)` and `fxsize(`*size*`)`.

Now let us distinguish between

a. controlling the aspect ratio of the overall graph, and

b. controlling the aspect ratio of individual graphs in a combined graph.

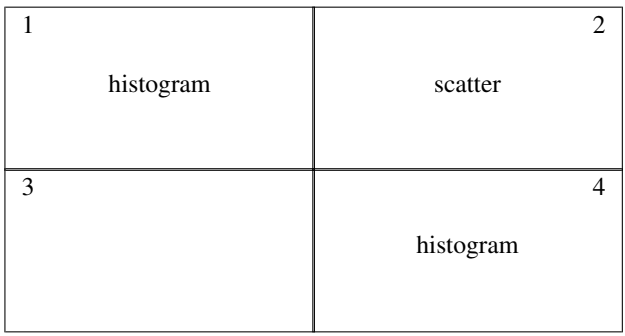
For problem (a), methods (1) and (2) are best. We used method (2) when we constructed the overall combined graph above—we specified `graphregion(margin(l=22 r=22))`. Methods 1 and 2 are discussed under *Controlling the aspect ratio* in [G-3] ***region_options***.

For problem (b), method (1) will not work, and methods (2) and (3) do different things.

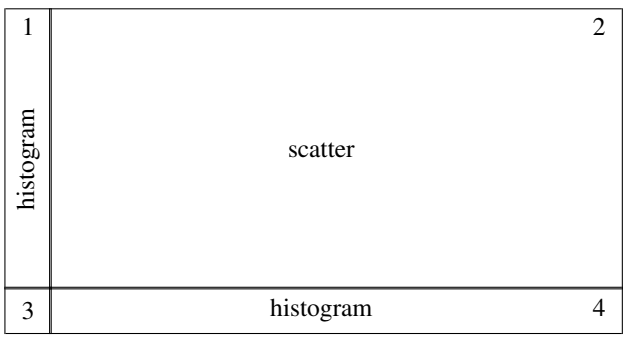
Method (1) controls the physical size at which the graph appears, so it indirectly controls the aspect ratio. `graph combine`, however, discards this physical-size information.

Method (2) is one way of controlling the aspect ratio of subgraphs. `graph combine` honors margins, assuming that you do not specify `graph combine`'s `imargin()` option, which overrides the original margin information. In any case, if you want the subgraph long and narrow, or short and wide, you need only specify the appropriate `graphregion(margin())` at the time you draw the subgraph. When you combine the resulting graph with other graphs, it will look exactly as you want it. The long-and-narrow or short-and-wide graph will appear in the array adjacent to all the other graphs. Each graph is allocated an equal-sized area in the array, and the oddly shaped graph is drawn into it.

Method (3) is the only way you can obtain unequally sized areas. For the combined graph above, you specified `graph combine`'s `imargin()` option and that alone precluded our use of method (2), but most importantly, you did not want an array of four equally sized areas:



We wanted



The *forced_size_options* allowed us to achieve that. You specify the *forced_size_options* `fysize()` and `fxsize()` with the commands that draw the subgraphs, not with `graph combine`. Inside the parentheses, you specify the percentage of the graph region to be used. Although you could use `fysize()` and `fxsize()` to control the aspect ratio in ordinary cases, there is no reason to do that. Use `fysize()` and `fxsize()` to control the aspect ratio when you are going to use `graph combine`

and you want unequally sized areas or when you will be specifying `graph combine`'s `imargin()` option.

References

Ängquist, L. 2014. Stata tip 117: graph combine—Combining graphs. *Stata Journal* 14: 221–225.

Cox, N. J. 2020. Stata tip 139: The by() option of graph can work better than graph combine. *Stata Journal* 20: 1016–1027.

Also see

[G-2] **graph use** — Display graph stored on disk

[G-2] **graph save** — Save graph to disk

[G-3] ***saving_option*** — Option for saving graph to disk

[G-4] **Concept: gph files** — Using gph files