graph twoway scatter — Two-way scatterplots

Description | Quick start | Menu | Syntax
Options | Remarks and examples | References | Also see

Description

`scatter` draws scatterplots and is the mother of all the `twoway` plottypes, such as `line` and `lfit` (see [G-2] **graph twoway line** and [G-2] **graph twoway lfit**).

`scatter` is both a command and a *plottype* as defined in [G-2] **graph twoway**. Thus the syntax for `scatter` is

```
. graph twoway scatter ...
. twoway scatter ...
. scatter ...
```

Being a plottype, `scatter` may be combined with other plottypes in the `twoway` family (see [G-2] **graph twoway**), as in,

```
. twoway (scatter ...) (line ...) (lfit ...) ...
```

which can equivalently be written as

```
. scatter ... || line ... || lfit ... || ...
```

Quick start

Scatterplot of `y1` versus `x`

```
twoway scatter y1 x
```

Same as above

```
scatter y1 x
```

Scatterplot overlaid with a line showing the linear relationship between `y1` and `x`

```
twoway scatter y1 x || lfit y1 x
```

Same as above, but suppress the legend

```
twoway scatter y1 x || lfit y1 x, legend(off)
```

Same as above, but with a separate graph area for each level of `catvar`

```
twoway scatter y1 x || lfit y1 x, by(catvar, legend(off))
```

Scatterplot overlaid with a lowess plot

```
twoway scatter y1 x || lowess y1 x
```

Scatterplot with different markers showing `catvar` $= 0$ and `catvar` $= 1$

```
twoway scatter y1 x if catvar==0 || scatter y1 x if catvar==1
```

Same as above, but use "Zero" and "One" in the legend to indicate the value of `catvar`

```
twoway scatter y1 x if catvar==0 || scatter y1 x if catvar==1, ///
  legend(order(1 "Zero" 2 "One"))
```

Make the markers red

```
scatter y1 x, mcolor(red)
```

Show markers as hollow circles with size proportional to frequency weight variable wvar

```
scatter y1 x [fweight=wvar], msymbol(Oh)
```

Add a marker at (15, 22) with label "Note this point"

```
scatter y1 x || scatteri 22 15 "Note this point"
```

Scatterplot of y1 and y2 versus x

```
scatter y1 y2 x
```

Specify the title "My Title" and the subtitle "My Subtitle"

```
scatter y1 y2 x, title("My Title") subtitle("My Subtitle")
```

Specify a two-line title

```
scatter y1 y2 x, title("My Somewhat" "Long Title")
```

Change the order of the plots in the legend to be y2 and y1

```
scatter y1 y2 x, legend(order(2 1))
```

Same as above, and control the appearance of the graph using the monochrome scheme stmono1

```
scatter y1 y2 x, legend(order(2 1)) scheme(stmono1)
```

Add random noise to the plot of y3 against x to make overlapping points more visible

```
scatter y3 x, jitter(5)
```

Menu

Graphics > Two-way graph (scatter, line, etc.)

Syntax

[twoway] scatter *varlist* [*if*] [*in*] [*weight*] [, *options*]

where *varlist* is

y_1 [y_2[...]] x

options	Description
marker_options	change look of markers (color, size, etc.)
marker_label_options	add marker labels; change look or position
colorvar_options	change color of markers based on values of a variable
connect_options	change look of lines or connecting method
jitter_options	jitter marker positions using random noise
axis_choice_options	associate plot with alternative axis
twoway_options	titles, legends, axes, added lines and text, by, regions, name, aspect ratio, etc.

aweights, fweights, and pweights are allowed; see **[U] 11.1.6 weight**.

marker_options	Description
msymbol(*symbolstylelist*)	shape of marker
mcolor(*colorstylelist*)	color and opacity of marker, inside and out
msize(*markersizestylelist*)	size of marker
msangle(*anglestyle*)	angle of marker symbol
mfcolor(*colorstylelist*)	inside or "fill" color and opacity
mlcolor(*colorstylelist*)	color and opacity of outline
mlwidth(*linewidthstylelist*)	thickness of outline
mlalign(*linealignmentstyle*)	outline alignment (inside, outside, center)
mlstyle(*linestylelist*)	overall style of outline
mstyle(*markerstylelist*)	overall style of marker
pstyle(*pstylelist*)	all the ...style() options above

The "*list*" specifiers shown above are available only when multiple y-variables are specified, with one specifier allowed per y-variable; otherwise, the standard specifiers listed in the option should be used.

marker_label_options	Description
mlabel(*varlist*)	specify marker variables
mlabstyle(*markerlabelstylelist*)	overall style of label
mlabposition(*clockposlist*)	where to locate label
mlabvposition(*varname*)	where to locate label 2
mlabgap(*sizelist*)	gap between marker and label
mlabangle(*anglestylelist*)	angle of label
mlabtextstyle(*textstylelist*)	overall style of text
mlabsize(*textsizestylelist*)	size of label
mlabcolor(*colorstylelist*)	color and opacity of label
mlabformat(*%fmtlist*)	format of label

The "*list*" specifiers shown above are available only when multiple y-variables are specified, with one specifier allowed per y-variable; otherwise, the standard specifiers listed in the option should be used.

colorvar_options	Description
colorvar(*colorvar*)	specify variable to control color
colordiscrete	treat *colorvar* as discrete instead of continuous
colorcuts(*numlist*)	specify list of cuts for *colorvar*
colorlevels(#)	specify number of levels for *colorvar*
colorvarminmax	include minimum and maximum of *colorvar* in the list of cuts created by colorcuts() and colorlevels()
colorrule([*crule*] [%# [*#]])	specify rule for creating colors for levels of *colorvar*
colorstart(*colorstyle*)	specify starting color for colorrule()
colorend(*colorstyle*)	specify ending color for colorrule()
colorlist(*colorstylelist*)	specify list of colors for each level of *colorvar*
colorfillonly	specify that the color for items based on *colorvar* not affect the outline
colorformissing(*colorstyle*)	specify the color for the level of missing values
colorkeysrange	draw keys in legend as ranged bars
coloruseplegend	use the contour-line plot legend instead of the contour plot legend

connect_options	Description
connect(*connectstylelist*)	how to connect points
sort[(*varlist*)]	how to order data before connecting
cmissing({y|n}...)	missing values are ignored
lpattern(*linepatternstylelist*)	line pattern (solid, dashed, etc.)
lwidth(*linewidthstylelist*)	thickness of line
lcolor(*colorstylelist*)	color and opacity of line
lalign(*linealignmentstyle*)	line alignment (inside, outside, center)
lstyle(*linestylelist*)	overall style of line
pstyle(*pstylelist*)	all the ...style() options above

The "*list*" specifiers shown above are available only when multiple y-variables are specified, with one specifier allowed per y-variable; otherwise, the standard specifiers listed in the option should be used.

jitter_options	Description
`jitter(#)`	perturb location of point
`jitterseed(#)`	random-number seed for `jitter()`

axis_choice_options	Description
`yaxis(#[#...])`	which y axis to use
`xaxis(#[#...])`	which x axis to use

twoway_options	Description
added_line_options	draw lines at specified y or x values
added_text_options	display text at specified (y, x) value
axis_options	labels, ticks, grids, log scales
title_options	titles, subtitles, notes, captions
legend_options	legend explaining what means what
`scale(#)`	resize text and markers
region_options	shading, outlining, and graph size
aspect_option	constrain aspect ratio of plot region
`scheme(`*schemename*`)`	overall look
`play(`*recordingname*`)`	play edits from *recordingname*
`by(`*varlist*`, ...)`	repeat for subgroups
`nodraw`	suppress display of graph
`name(`*name*`, ...)`	specify name for graph
`saving(`*filename*`, ...)`	save graph in file
advanced_options	difficult to explain

Options

marker_options specify how the points on the graph are to be designated. Markers are the ink used to mark where points are on a plot. Markers have shape, color, and size, and other characteristics. See [G-3] ***marker_options*** for a description of markers and the options that specify them.

`msymbol(O D S T + X o d s t smplus x)` is the default. `msymbol(i)` will suppress the appearance of the marker altogether.

marker_label_options specify labels to appear next to or in place of the markers. For instance, if you were plotting country data, marker labels would allow you to have "Argentina", "Bolivia", ..., appear next to each point and, with a few data, that might be desirable. See [G-3] ***marker_label_options*** for a description of marker labels and the options that control them.

By default, no marker labels are displayed. If you wish to display marker labels in place of the markers, specify `mlabposition(0)` and `msymbol(i)`.

colorvar_options specify that the color of the markers be determined by the levels of the numeric variable *colorvar*; see [G-3] ***colorvar_options***. *colorvar_options* are not allowed when plotting multiple ys against one x.

connect_options specify how the points are to be connected. The default is not to connect the points.

`connect()` specifies whether points are to be connected and, if so, how the line connecting them is to be shaped. The line between each pair of points can connect them directly or in stairstep fashion.

`sort` specifies that the data be sorted by the x variable before the points are connected. Unless you are after a special effect or your data are already sorted, do not forget to specify this option. If you are after a special effect, and if the data are not already sorted, you can specify `sort(`*varlist*`)` to specify exactly how the data should be sorted. Understand that specifying `sort` or `sort(`*varlist*`)` when it is not necessary will slow Stata down a little. You must specify `sort` if you wish to connect points, and you must specify the *twoway_option* `by()` with `total`.

`cmissing(y)` and `cmissing(n)` specify whether missing values are ignored when points are connected; whether the line should have a break in it. The default is `cmissing(y)`, meaning that there will be no breaks.

`lpattern()` specifies how the style of the line is to be drawn: solid, dashed, etc.

`lwidth()` specifies the width of the line.

`lcolor()` specifies the color and opacity of the line.

`lalign()` specifies the alignment of the line.

`lstyle()` specifies the overall style of the line.

`pstyle(`*pstyle*`)` specifies the overall style of the plot and is a composite of `mstyle()`, `mlabstyle()`, `lstyle()`, `connect()`, and `cmissing()`. The default is `pstyle(p1)` for the first plot, `pstyle(p2)` for the second, and so on. See *Appendix: Styles and composite styles* under *Remarks and examples*.

See [G-3] ***connect_options*** for more information on these and related options. See [G-4] **Concept: lines** for an overview of lines.

`jitter(`#`)` adds spherical random noise to the data before plotting. # represents the size of the noise as a percentage of the graphical area. This option is useful when plotting data which otherwise would result in points plotted on top of each other. See *Jittered markers* under *Remarks and examples*.

Commonly specified are `jitter(5)` or `jitter(6)`; `jitter(0)` is the default.

`jitterseed(`#`)` specifies the seed for the random noise added by the `jitter()` option. # should be specified as a positive integer. Use this option to reproduce the same plotted points when the `jitter()` option is specified.

axis_choice_options are for use when you have multiple x or y axes.
See [G-3] ***axis_choice_options*** for more information.

twoway_options include

added_line_options, which specify that horizontal or vertical lines be drawn on the graph; see [G-3] ***added_line_options***. If your interest is in drawing grid lines through the plot region, see *axis_options* below.

added_text_options, which specify text to be displayed on the graph (inside the plot region); see [G-3] ***added_text_options***.

axis_options, which allow you to specify labels, ticks, and grids. These options also allow you to obtain logarithmic scales; see [G-3] ***axis_options***.

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

legend_options, which allows specifying the legend explaining the symbols and line styles used; see [G-3] ***legend_options***.

`scale(`#`)`, which makes all the text and markers on a graph larger or smaller (`scale(1)` means no change); see [G-3] ***scale_option***.

region_options, which allow you to control the shading and outlining of the plot region and the graph size; see [G-3] ***region_options***.

aspect_option, which allow you to control the aspect ratio of the plot region; see [G-3] ***aspect_option***.

`scheme(`*schemename*`)`, which specifies the overall look of the graph; see [G-3] ***scheme_option***.

`play(`*recordingname*`)` applies the edits from *recordingname* to the graph, where *recordingname* is the name under which edits previously made in the Graph Editor have been recorded and stored. See *Graph Recorder* in [G-1] **Graph Editor**.

`by(`*varlist*`, ...)`, which allows drawing multiple graphs for each subgroup of the data; see [G-3] ***by_option***.

`nodraw`, which prevents the graph from being displayed; see [G-3] ***nodraw_option***.

`name(`*name*`)`, which allows you to save the graph in memory under a name different from `Graph`; see [G-3] ***name_option***.

`saving(`*filename*`[ , asis replace ])`, which allows you to save the graph to disk; see [G-3] ***saving_option***.

See [G-3] ***twoway_options*** for complete details.

Remarks and examples

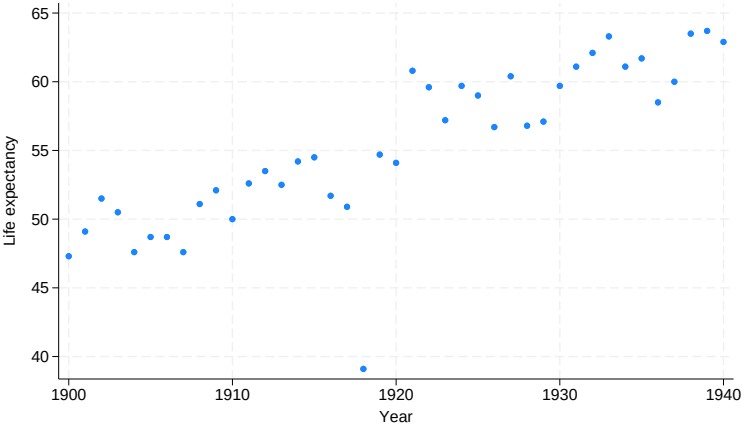
Remarks are presented under the following headings:

Typical use
Scatter syntax
The overall look for the graph
The size and aspect ratio of the graph
Titles
Axis titles
Axis labels and ticking
Grid lines
Added lines
Axis range
Log scales
Multiple axes
Markers
Weighted markers
Jittered markers
Connected lines
Graphs by groups
Saving graphs
Video example
Appendix: Styles and composite styles

Typical use

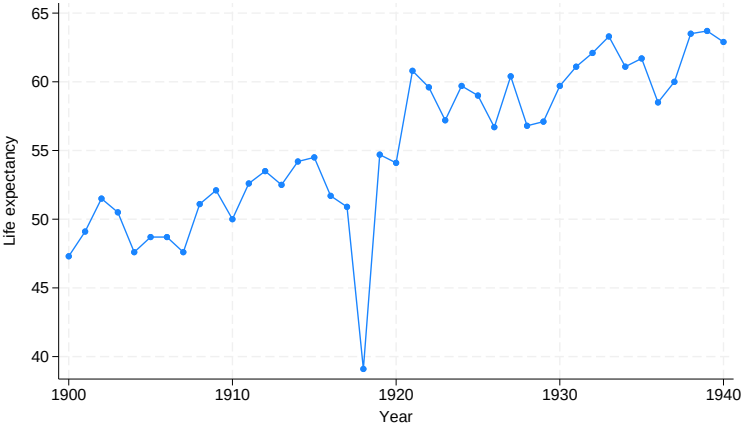
The scatter plottype by default individually marks the location of each point:

```
. use https://www.stata-press.com/data/r19/uslifeexp2
(US life expectancy, 1900-1940)
. scatter le year
```



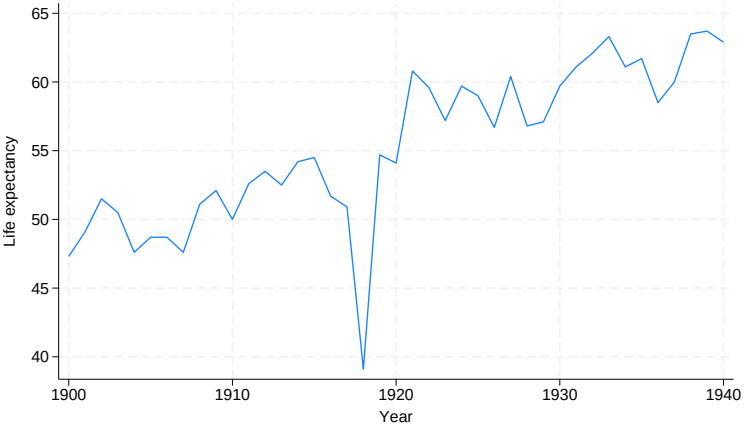
With the specification of options, you can produce the same effect as `twoway connected` (see [G-2] **graph twoway connected**),

```
. scatter le year, connect(l)
```



or twoway line:

```
. scatter le year, connect(l) msymbol(i)
```



In fact, all the other two-way plottypes eventually work their way back to executing scatter. scatter literally is the mother of all two-way graphs in Stata.

Scatter syntax

See [G-2] **graph twoway** for an overview of graph twoway syntax. Especially for graph twoway scatter, the only thing to know is that if more than two variables are specified, all but the last are given the interpretation of being *y* variables. For example,

```
. scatter y1var y2var xvar
```

would plot *y1var* versus *xvar* and overlay that with a plot of *y2var* versus *xvar*, so it is the same as typing

```
. scatter y1var xvar || scatter y2var xvar
```

If, using the multiple-variable syntax, you specify scatter-level options (that is, all options except *twoway_options* as defined in the syntax diagram), you specify arguments for *y1var*, *y2var*, ..., separated by spaces. That is, you might type

```
. scatter y1var y2var xvar, ms(O i) c(. l)
```

ms() and c() are abbreviations for msymbol() and connect(); see [G-3] ***marker_options*** and [G-3] ***connect_options***. In any case, the results from the above are the same as if you typed

```
. scatter y1var xvar, ms(O) c(.) || scatter y2var xvar, ms(i) c(l)
```

There need not be a one-to-one correspondence between options and *y* variables when you use the multiple-variable syntax. If you typed

```
. scatter y1var y2var xvar, ms(O) c(l)
```

then options ms() and c() will have default values for the second scatter, and if you typed

```
. scatter y1var y2var xvar, ms(O S i) c(l l l)
```

the extra options for the nonexistent third variable would be ignored.

If you wish to specify the default for one of the y variables, you may specify period (.):

```
. scatter y1var y2var xvar, ms(. O) c(. l)
```

There are other shorthands available to make specifying multiple arguments easier; see [G-4] ***stylelists***.

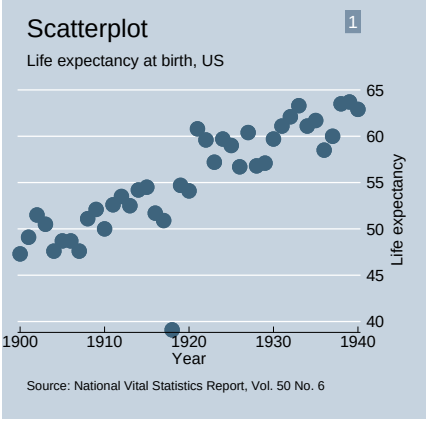
Because multiple variables are interpreted as multiple y variables, to produce graphs containing multiple x variables, you must chain together separate `scatter` commands:

```
. scatter yvar x1var, ... || . scatter yvar x2var, ...
```

The overall look for the graph

The overall look of the graph is mightily affected by the scheme, and there is a `scheme()` option that will allow you to specify which scheme to use. We showed earlier the results of `scatter le year`. Here is the same graph repeated using the `economist` scheme:

```
. use https://www.stata-press.com/data/r19/uslifeexp2, clear
(US life expectancy, 1900-1940)
. scatter le year,
        title("Scatterplot")
        subtitle("Life expectancy at birth, US")
        note("1")
        caption("Source: National Vital Statistics Report,
         Vol. 50 No. 6")
        scheme(economist)
```



See [G-4] **Schemes intro**.

The size and aspect ratio of the graph

The size and aspect ratio of the graph are controlled by the *region_options* `ysize(#)` and `xsize(#)`, which specify the height and width in inches of the graph. For instance,

```
. scatter yvar xvar, xsize(4) ysize(4)
```

would produce a 4 × 4 inch square graph. See [G-3] ***region_options***.

Titles

By default, no titles appear on the graph, but the *title_options* `title()`, `subtitle()`, `note()`, `caption()`, and `legend()` allow you to specify the titles that you wish to appear, as well as to control their position and size. For instance,

```
. scatter yvar xvar, title("My title")
```

would draw the graph and include the title "My title" (without the quotes) at the top. Multiple-line titles are allowed. Typing

```
. scatter yvar xvar, title("My title" "Second line")
```

would create a two-line title. The above, however, would probably look better as a title followed by a subtitle:

```
. scatter yvar xvar, title("My title") subtitle("Second line")
```

In any case, see [G-3] ***title_options***.

Axis titles

Titles do, by default, appear on the y and x axes. The axes are titled with the variable names being plotted or, if the variables have variable labels, with their variable labels. The *axis_title_options* `ytitle()` and `xtitle()` allow you to override that. If you specify

```
. scatter yvar xvar, ytitle("")
```

the title on the y axis would disappear. If you specify

```
. scatter yvar xvar, ytitle("Rate of change")
```

the y-axis title would become "Rate of change". As with all titles, multiple-line titles are allowed:

```
. scatter yvar xvar, ytitle("Time to event" "Rate of change")
```

See [G-3] ***axis_title_options***.

Axis labels and ticking

By default, approximately five major ticks and labels are placed on each axis. The *axis_label_options* `ylabel()` and `xlabel()` allow you to control that. Typing

```
. scatter yvar xvar, ylabel(#10)
```

would put approximately 10 labels and ticks on the y axis. Typing

```
. scatter yvar xvar, ylabel(0(1)9)
```

would put exactly 10 labels at the values 0, 1, …, 9.

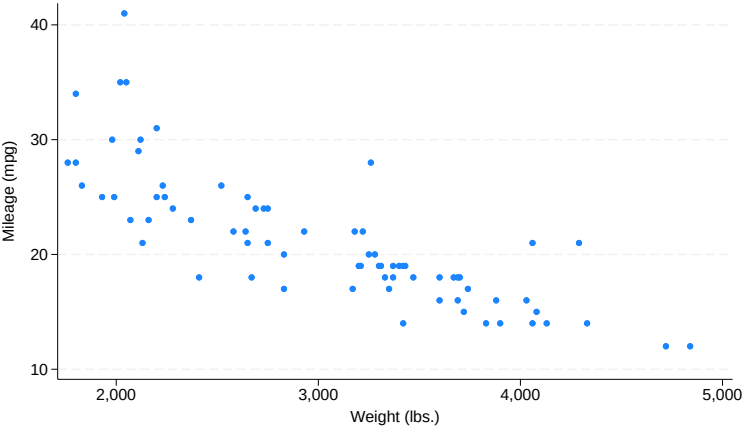
`ylabel()` and `xlabel()` have other features, and options are also provided for minor labels and minor ticks; see [G-3] ***axis_label_options***.

Grid lines

With the `st` family of schemes—see [G-4] **Scheme st**—grid lines are included in both the y and x axes by default. You can specify the `xlabel(,nogrid)` and `ylabel(,nogrid)` options to suppress the grid lines.

Grid lines are considered an extension of ticks and are specified as suboptions inside the *axis_label_options* `ylabel()` and `xlabel()`. For instance, below we suppress the grid lines on the x axis:

```
. use https://www.stata-press.com/data/r19/auto, clear
(1978 automobile data)
. scatter mpg weight, xlabel(, nogrid)
```



In the above example, the grid lines are placed at the same values as the default ticks and labels, but you can control that, too. See [G-3] ***axis_label_options***.

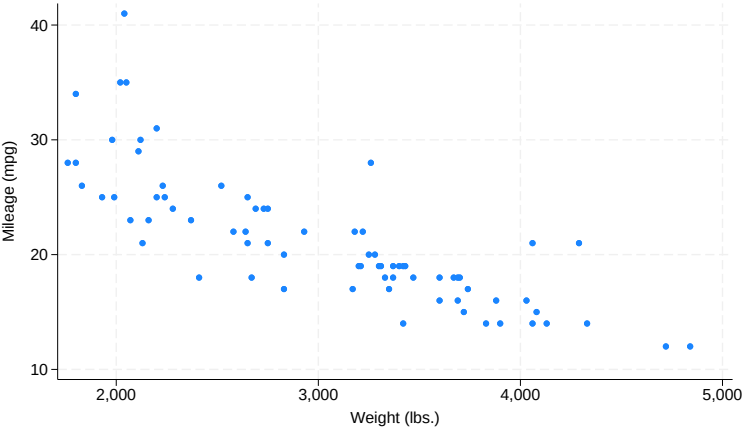
Added lines

Lines may be added to the graph for emphasis by using the *added_line_options* `yline()` and `xline()`; see [G-3] ***added_line_options***.

Axis range

The extent or range of an axis is set according to all the things that appear on it—the data being plotted and the values on the axis being labeled or ticked. In the graph that just appeared above,

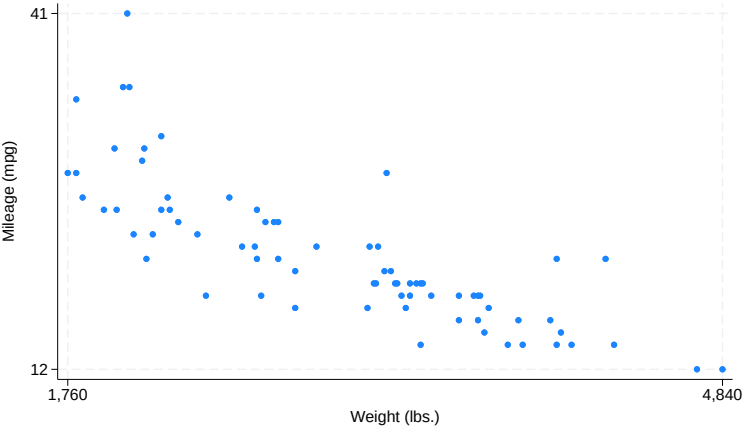
```
. use https://www.stata-press.com/data/r19/auto, clear
(1978 automobile data)
. scatter mpg weight
```



variable `mpg` varies between 12 and 41 and yet the y axis extends from 10 to 41. The axis was extended to include $10 < 12$ because the value 10 was labeled. Variable `weight` varies between 1,760 and 4,840; the x axis extends from 1,760 to 5,000. This axis was extended to include $5{,}000 > 4{,}840$ because the value 5,000 was labeled.

You can prevent axes from being extended by specifying the `ylabel(minmax)` and `xlabel(minmax)` options. `minmax` specifies that only the minimum and maximum are to be labeled:

```
. scatter mpg weight, ylabel(minmax) xlabel(minmax)
```



In other cases, you may wish to widen the range of an axis. This you can do by specifying the `range()` descriptor of the *axis_scale_options* `yscale()` or `xscale()`. For instance,

```
. scatter mpg weight, xscale(range(1000 5000))
```

would widen the x axis to include 1,000–5,000. We typed out the name of the option, but most people would type

```
. scatter mpg weight, xscale(r(1000 5000))
```

`range()` can widen, but never narrow, the extent of an axis. Typing

```
. scatter mpg weight, xscale(r(1000 4000))
```

would not omit cars with `weight`$>$ 4000 from the plot. If that is your desire, type

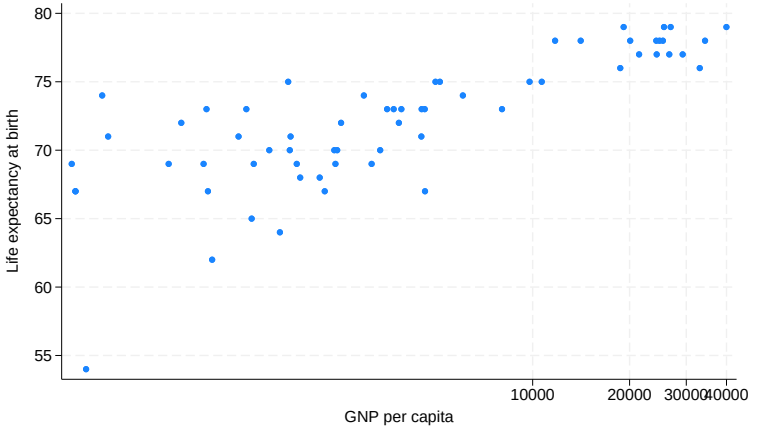
```
. scatter mpg weight if weight<=4000
```

See [G-3] ***axis_scale_options*** for more information on `range()`, `yscale()`, and `xscale()`; see [G-3] ***axis_label_options*** for more information on `ylabel(minmax)` and `xlabel(minmax)`.

Log scales

By default, arithmetic scales for the axes are used. Log scales can be obtained by specifying the `log` suboption of `yscale()` and `xscale()`. For instance,

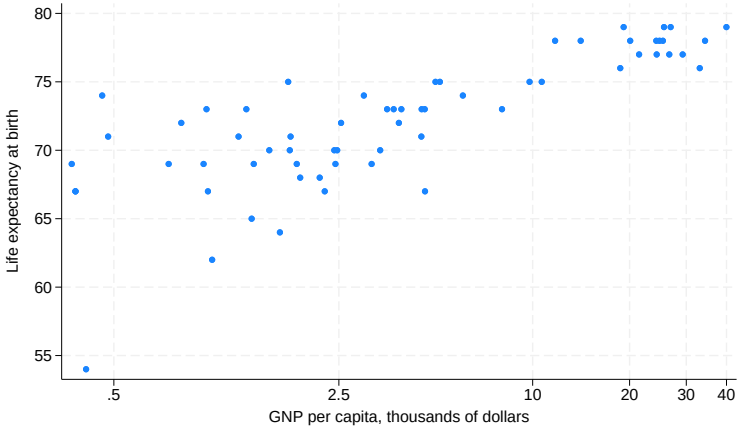
```
. use https://www.stata-press.com/data/r19/lifeexp, clear
(Life expectancy, 1998)
. scatter lexp gnppc, xscale(log)
```



The important option above is `xscale(log)`, which caused `gnppc` to be presented on a log scale.

The values 30,000 and 40,000 are overprinted. We could improve the graph by typing

```
. generate gnp000 = gnppc/1000
(5 missing values generated)
. label var gnp000 "GNP per capita, thousands of dollars"
. scatter lexp gnp000, xsca(log) xlab(.5 2.5 10(10)40)
```



See [G-3] ***axis_options***.

Multiple axes

Graphs may have more than one y axis and more than one x axis. There are two reasons to do this: you might include an extra axis so that you have an extra place to label special values or so that you may plot multiple variables on different scales. In either case, specify the `yaxis()` or `xaxis()` option. See [G-3] ***axis_choice_options***.

Markers

Markers are the ink used to mark where points are on the plot. Many people think of markers in terms of their shape (circles, diamonds, etc.), but they have other properties, including, most importantly, their color and size. The shape of the marker is specified by the `msymbol()` option, its color by the `mcolor()` option, and its size by the `msize()` option.

By default, solid circles are used, and the color will be different for each y variable; see *marker_options* under *Options* for the remaining details, if you care. In any case, when you type

```
. scatter yvar xvar
```

results are as if you typed

```
. scatter yvar xvar, msymbol(O)
```

You can vary the symbol used by specifying other `msymbol()` arguments. Similarly, you can vary the color and size of the symbol by specifying the `mcolor()` and `msize()` options. See [G-3] ***marker_options***.

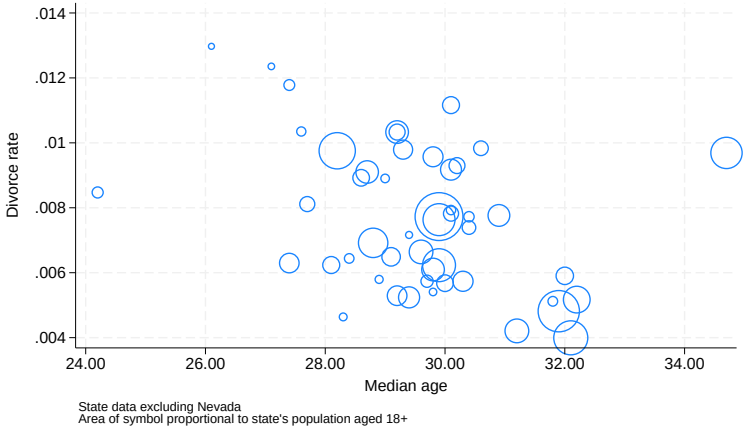
In addition to the markers themselves, you can request that the individual points be labeled. These marker labels are numbers or text that appear beside the marker symbol—or in place of it—to identify the points. See [G-3] ***marker_label_options***.

Weighted markers

If weights are specified—see [U] 11.1.6 weight—the size of the marker is scaled according to the size of the weights. aweights, fweights, and pweights are allowed and all are treated the same; iweights are not allowed because scatter would not know what to do with negative values. Weights affect the size of the marker and nothing else about the plot.

Below we use US state–averaged data to graph the divorce rate in a state versus the state's median age. We scale the symbols to be proportional to the population size:

```
. use https://www.stata-press.com/data/r19/census, clear
(1980 Census data by state)
. generate drate = divorce / pop18p
. label var drate "Divorce rate"
. scatter drate medage [w=pop18p] if state!="Nevada", msymbol(Oh)
    note("State data excluding Nevada"
         "Area of symbol proportional to state's population aged 18+")
```



Note the use of the msymbol(Oh) option. Hollow scaled markers look much better than solid ones.

scatter scales the symbols so that the sizes are a fair representation when the weights represent population weights. If all the weights except one are 1,000 and the exception is 999, the symbols will all be of almost equal size. The weight 999 observation will not be a dot and the weight 1,000 observation giant circles as would be the result if the exception had weight 1.

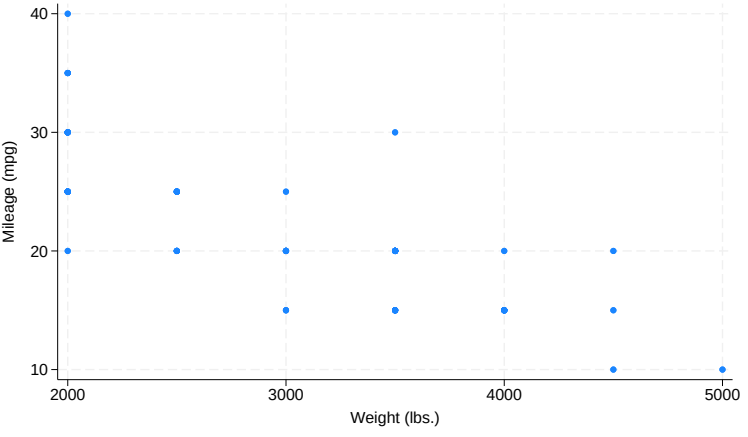
Weights are ignored when the mlabel() option is specified. See [G-3] *marker_label_options*.

Jittered markers

scatter will add spherical random noise to your data before plotting if you specify jitter(#), where # represents the size of the noise as a percentage of the graphical area. This can be useful for creating graphs of categorical data when, were the data not jittered, many of the points would be on top of each other, making it impossible to tell whether the plotted point represented one or 1,000 observations.

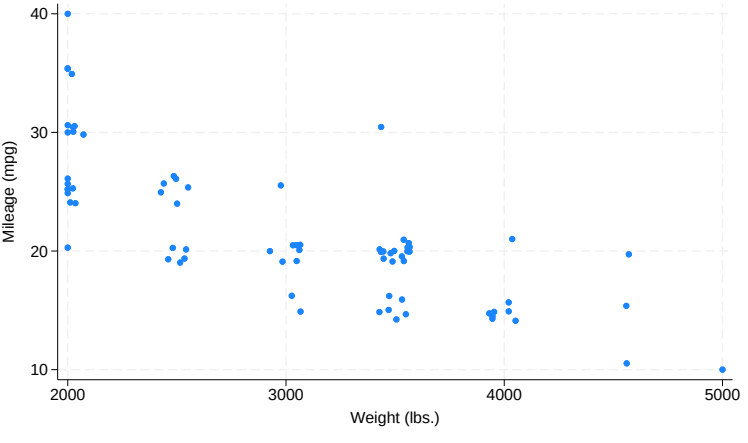
For instance, in a variation on `auto.dta` used below, `mpg` is recorded in units of 5 mpg, and `weight` is recorded in units of 500 pounds. A standard scatter has considerable overprinting:

```
. use https://www.stata-press.com/data/r19/autornd, clear
(1978 automobile data)
. scatter mpg weight
```



There are 74 points in the graph, even though it appears because of overprinting as if there are only 19. Jittering solves that problem:

```
. scatter mpg weight, jitter(7)
```



Connected lines

The connect() option allows you to connect the points of a graph. The default is not to connect the points.

If you want connected points, you probably want to specify connect(l), which is usually abbreviated c(l). The l means that the points are to be connected with straight lines. Points can be connected in other ways (such as a stairstep fashion), but usually c(l) is the right choice. The command

```
. scatter yvar xvar, c(l)
```

will plot *yvar* versus *xvar*, marking the points in the usual way, and drawing straight lines between the points. It is common also to specify the sort option,

```
. scatter yvar xvar, c(l) sort
```

because otherwise points are connected in the order of the data. If the data are already in the order of *xvar*, the sort is unnecessary. You can also omit the sort when creating special effects.

connect() is often specified with the msymbol(i) option to suppress the display of the individual points:

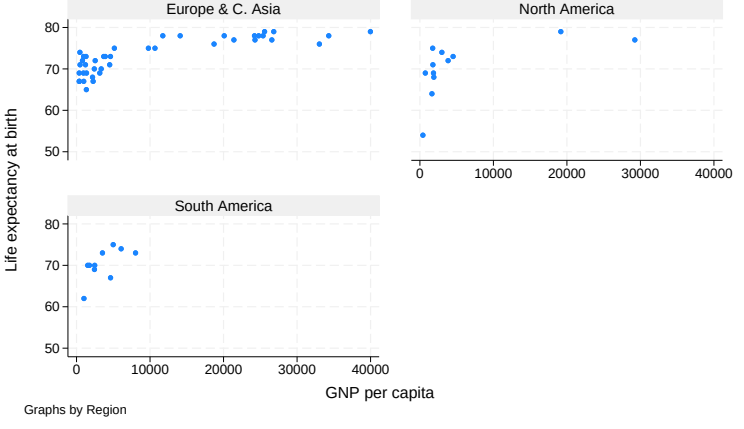
```
. scatter yvar xvar, c(l) sort m(i)
```

See [G-3] ***connect_options***.

Graphs by groups

Option by() specifies that graphs are to be drawn separately for each of the different groups and the results arrayed into one display. Below we use country data and group the results by region of the world:

```
. use https://www.stata-press.com/data/r19/lifeexp, clear
(Life expectancy, 1998)
. scatter lexp gnppc, by(region)
```



Variable region is a numeric variable taking on values 1, 2, and 3. Separate graphs were drawn for each value of region. The graphs were titled "Europe & C. Asia", "North America", and "South America" because numeric variable region had been assigned a value label, but results would have been the same had variable region been a string directly containing "Europe & C. Asia", "North America", and "South America".

See [G-3] ***by_option*** for more information on this useful option.

Saving graphs

To save a graph to disk for later printing or reviewing, include the saving() option,

```
. scatter ..., ... saving(filename)
```

or use the graph save command afterward:

```
. scatter ...
. graph save filename
```

See [G-3] ***saving_option*** and [G-2] **graph save**. Also see [G-4] **Concept: gph files** for information on how files such as *filename*.gph can be put to subsequent use.

Video example

Basic scatterplots in Stata

Appendix: Styles and composite styles

Many options end in the word style, including mstyle(), mlabstyle(), and lstyle(). Option mstyle(), for instance, is described as setting the "overall look" of a marker. What does that mean?

How something looks—a marker, a marker label, a line—is specified by many detail options. For markers, option msymbol() specifies its shape, mcolor() specifies its color and opacity, msize() specifies its size, and so on.

A *style* specifies a composite of related option settings. If you typed the mstyle(p1) option, you would be specifying a whole set of values for msymbol(), mcolor(), msize(), and all the other m*() options. p1 is called the name of a style, and p1 contains the settings.

Concerning mstyle() and all the other options ending in the word style, throughout this manual you will read statements such as

> Option *whatever*style() specifies the overall look of *whatever*, such as its *(insert list here)*. The other options allow you to change the attributes of a *whatever*, but *whatever*style() is the starting point.
>
> You need not specify *whatever*style() just because there is something you want to change about the look of a *whatever*, and in fact, most people seldom specify the *whatever*style() option. You specify *whatever*style() when another style exists that is exactly what you desire or when another style would allow you to specify fewer changes to obtain what you want.

Styles actually come in two forms called *composite styles* and *detail styles*, and the above statement applies only to composite styles and appears only in manual entries concerning composite styles. Composite styles are specified in options that end in the word style. The following are examples of composite styles:

```
mstyle(symbolstyle)
mlstyle(linestyle)
mlabstyle(markerlabelstyle)
lstyle(linestyle)
pstyle(pstyle)
```

The following are examples of detail styles:

```
mcolor(colorstyle)
mlwidth(linewidthstyle)
mlabsize(textsizestyle)
lpattern(linepatternstyle)
```

In the above examples, distinguish carefully between option names such as `mcolor()` and option arguments such as *colorstyle*. *colorstyle* is an example of a detail style because it appears in the `mcolor()` option, and the option name does not end in the word style.

Detail styles specify precisely how an attribute of something looks, and composite styles specify an "overall look" in terms of detail-style values.

Composite styles sometimes contain other composite styles as members. For instance, when you specify the `mstyle()` option—which specifies the overall look of markers—you are also specifying an `mlstyle()`—which specifies the overall look of the lines that outline the shape of the markers. That does not mean you cannot specify the `mlstyle()` option, too. It just means that specifying `mstyle()` implies an `mlstyle()`. The order in which you specify the options does not matter. You can type

```
. scatter ..., ... mstyle(...) ... mlstyle(...) ...
```

or

```
. scatter ..., ... mlstyle(...) ... mstyle(...) ...
```

and, either way, `mstyle()` will be set as you specify, and then `mlstyle()` will be reset as you wish. The same applies for mixing composite-style and detail-style options. Option `mstyle()` implies an `mcolor()` value. Even so, you may type

```
. scatter ..., ... mstyle(...) ... mcolor(...) ...
```

or

```
. scatter ..., ... mcolor(...) ... mstyle(...) ...
```

and the outcome will be the same.

The grandest composite style of them all is `pstyle(`*pstyle*`)`. It contains all the other composite styles and `scatter` (`twoway`, in fact) makes great use of this grand style. When you type

```
. scatter y1var y2var xvar, ...
```

results are as if you typed

```
. scatter y1var y2var xvar, pstyle(p1 p2) ...
```

That is, *y1var* versus *xvar* is plotted using `pstyle(p1)`, and *y2var* versus *xvar* is plotted using `pstyle(p2)`. It is the `pstyle(p1)` that sets all the defaults—which marker symbols are used, what color they are, etc.

The same applies if you type

```
. scatter y1var xvar, ... || scatter y2var xvar, ...
```

y1var versus *xvar* is plotted using `pstyle(p1)`, and *y2var* versus *xvar* is plotted using `pstyle(p2)`, just as if you had typed

```
. scatter y1var xvar, pstyle(p1) ... || scatter y2var xvar, pstyle(p2) ...
```

The same applies if you mix `scatter` with other plottypes:

```
. scatter y1var xvar, ... || line y2var xvar, ...
```

is equivalent to

```
. scatter y1var xvar, pstyle(p1) ... || line y2var xvar, pstyle(p2) ...
```

and

```
. twoway (..., ...) (..., ...), ...
```

is equivalent to

```
. twoway (..., pstyle(p1) ...) (..., pstyle(p2) ...), ...
```

which is why we said that it is `twoway`, and not just `scatter`, that exploits `scheme()`.

You can put this to use. Pretend that you have a dataset on husbands and wives and it contains the variables

`hinc`	husband's income
`winc`	wife's income
`hed`	husband's education
`wed`	wife's education

You wish to draw a graph of income versus education, drawing no distinctions between husbands and wives. You type

```
. scatter hinc hed || scatter winc wed
```

You intend to treat husbands and wives the same in the graph, but in the above example, they are treated differently because `msymbol(O)` will be used to mark the points of `hinc` versus `hed` and `msymbol(D)` will be used to designate `winc` versus `wed`. The color of the symbols will be different, too.

You could address that problem in many different ways. You could specify the `msymbol()` and `mcolor()` options (see [G-3] ***marker_options***), along with whatever other detail options are necessary to make the two scatters appear the same. Being knowledgeable, you realize you do not have to do that. There is, you know, a composite style that specifies this. So you get out your manuals, flip through, and discover that the relevant composite style for the marker symbols is `mstyle()`.

Easiest of all, however, would be to remember that `pstyle()` contains all the other styles. Rather than resetting `mstyle()`, just reset `pstyle()`, and whatever needs to be set to make the two plots the same will be set. Type

```
. scatter hinc hed || scatter winc wed, pstyle(p1)
```

or, if you prefer,

```
. scatter hinc hed, pstyle(p1) || scatter winc wed, pstyle(p1)
```

You do not need to specify `pstyle(p1)` for the first plot, however, because that is the default.

As another example, you have a dataset containing

`mpg`	Mileage ratings of cars
`weight`	Each car's weight
`prediction`	A predicted mileage rating based on weight

You wish to draw the graph

```
. scatter mpg weight || line prediction weight
```

but you wish the appearance of the line to "match" that of the markers used to plot `mpg` versus `weight`. You could go digging to find out which option controlled the line style and color and then dig some more to figure out which line style and color goes with the markers used in the first plot, but much easier is simply to type

```
. scatter mpg weight || line prediction weight, pstyle(p1)
```

References

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and Y. Feng. 2025. Binscatter regressions. *Stata Journal* 25: 3–50.

Cox, N. J. 2005a. Stata tip 24: Axis labels on two or more levels. *Stata Journal* 5: 469.

———. 2005b. Stata tip 27: Classifying data points on scatter plots. *Stata Journal* 5: 604–606.

———. 2025. Stata tip 162: Add marginal rugs using marker symbols or axis ticks. *Stata Journal* 25: 491–497.

Friendly, M., and D. Denis. 2005. The early origins and development of the scatterplot. *Journal of the History of the Behavioral Sciences* 41: 103–130. https://doi.org/10.1002/jhbs.20078.

Musau, A. 2020. Stata tip 136: Between-group comparisons in a scatterplot with weighted markers. *Stata Journal* 20: 489–492.

Pinna, M. 2022. Binned scatterplots with marginal histograms: binscatterhist. *Stata Journal* 22: 430–445.

Royston, P., and N. J. Cox. 2005. A multivariable scatterplot smoother. *Stata Journal* 5: 405–412.

Tibbles, M., and E. Melse. 2023. A note on creating inset plots using graph twoway. *Stata Journal* 23: 265–275.

Winter, N. J. G. 2005. Stata tip 23: Regaining control over axis ranges. *Stata Journal* 5: 467–468.

Also see

[G-2] **graph twoway** — Two-way graphs

[G-3] ***axis_choice_options*** — Options for specifying the axes on which a plot appears

[G-3] ***connect_options*** — Options for connecting points with lines

[G-3] ***marker_label_options*** — Options for specifying marker labels

[G-3] ***marker_options*** — Options for specifying markers

[G-3] *twoway_options* — Options for two-way graphs

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on citing Stata documentation.