

graph twoway — Twoway graphs[Description](#)[Menu](#)[Syntax](#)[Remarks and examples](#)

Description

`twoway` is a family of plots, all of which fit on numeric y and x scales.

Menu

Graphics > Twoway graph (scatter, line, etc.)

Syntax

```
[graph] twoway plot [if] [in] [, twoway_options]
```

where the syntax of *plot* is

```
[c] plotype varlist ..., options [d] [ll]
```

<i>plotype</i>	Description
<code>scatter</code>	scatterplot
<code>line</code>	line plot
<code>connected</code>	connected-line plot
<code>scatteri</code>	<code>scatter</code> with immediate arguments
<code>area</code>	line plot with shading
<code>bar</code>	bar plot
<code>spike</code>	spike plot
<code>dropline</code>	dropline plot
<code>dot</code>	dot plot
<code>rarea</code>	range plot with area shading
<code>rbar</code>	range plot with bars
<code>rspike</code>	range plot with spikes
<code>rcap</code>	range plot with capped spikes
<code>rcapsym</code>	range plot with spikes capped with symbols
<code>rscatter</code>	range plot with markers
<code>rline</code>	range plot with lines
<code>rconnected</code>	range plot with lines and markers
<code>pcspike</code>	paired-coordinate plot with spikes
<code>pccapsym</code>	paired-coordinate plot with spikes capped with symbols
<code>pccarrow</code>	paired-coordinate plot with arrows
<code>pccbarrow</code>	paired-coordinate plot with arrows having two heads
<code>pcscatter</code>	paired-coordinate plot with markers
<code>pci</code>	<code>pcspike</code> with immediate arguments
<code>pccarrowi</code>	<code>pccarrow</code> with immediate arguments
<code>tsline</code>	time-series plot
<code>tsrline</code>	time-series range plot
<code>contour</code>	contour plot with filled areas
<code>contourline</code>	contour lines plot
<code>mband</code>	median-band line plot
<code>mspline</code>	spline line plot
<code>lowess</code>	LOWESS line plot
<code>lfit</code>	linear prediction plot
<code>qfit</code>	quadratic prediction plot
<code>fpfit</code>	fractional polynomial plot
<code>lfitci</code>	linear prediction plot with CIs
<code>qfitci</code>	quadratic prediction plot with CIs
<code>fpfitci</code>	fractional polynomial plot with CIs
<code>function</code>	line plot of function
<code>histogram</code>	histogram plot
<code>kdensity</code>	kernel density plot
<code>lpoly</code>	local polynomial smooth plot
<code>lpolyci</code>	local polynomial smooth plot with CIs

For each of the above, see [G] **graph twoway** *plotype*, where you substitute for *plotype* a word from the left column.

twoway_options are as defined in [G-3] *twoway_options*.

The leading **graph** is optional. If the first (or only) *plot* is **scatter**, you may omit **twoway** as well, and then the syntax is

```
scatter ... [ , scatter_options ] [ || plot [ plot [ ... ] ] ]
```

and the same applies to **line**. The other *plotypes* must be preceded by **twoway**.

Regardless of how the command is specified, *twoway_options* may be specified among the *scatter_options*, *line_options*, etc., and they will be treated just as if they were specified among the *twoway_options* of the **graph twoway** command.

Remarks and examples

[stata.com](http://www.stata.com)

Remarks are presented under the following headings:

Definition

Syntax

Multiple if and in restrictions

twoway and plot options

Definition

Twoway graphs show the relationship between numeric data. Say that we have data on life expectancy in the United States between 1900 and 1940:

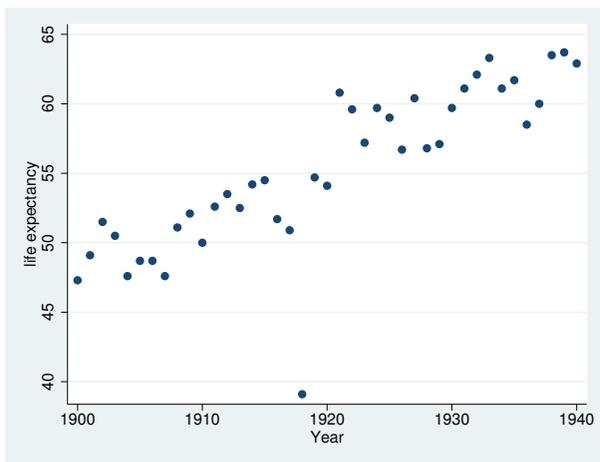
```
. use http://www.stata-press.com/data/r14/uslifeexp2
(U.S. life expectancy, 1900-1940)
. list in 1/8
```

	year	le
1.	1900	47.3
2.	1901	49.1
3.	1902	51.5
4.	1903	50.5
5.	1904	47.6
6.	1905	48.7
7.	1906	48.7
8.	1907	47.6

4 graph twoway — Twoway graphs

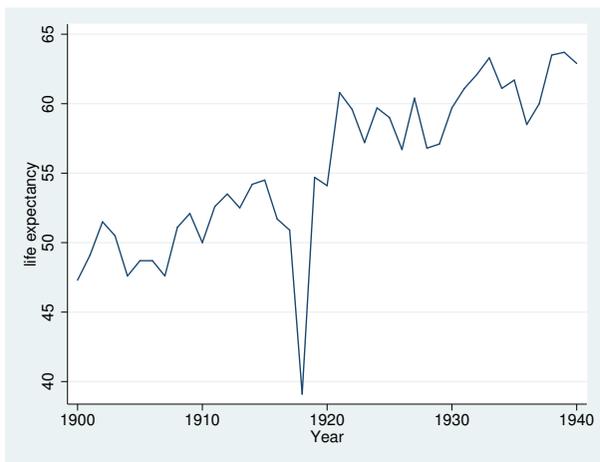
We could graph these data as a twoway scatterplot,

```
. twoway scatter le year
```



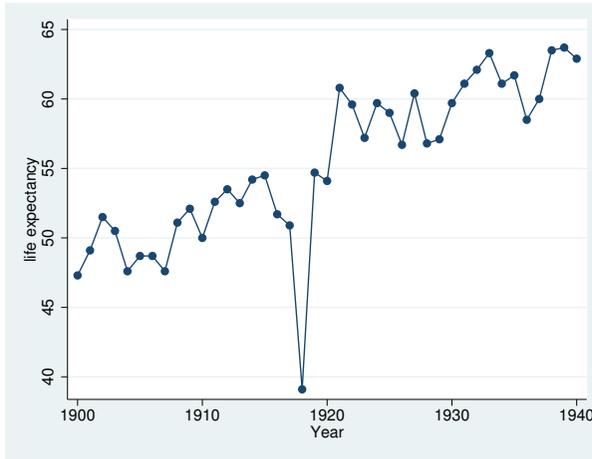
or we could graph these data as a twoway line plot,

```
. twoway line le year
```



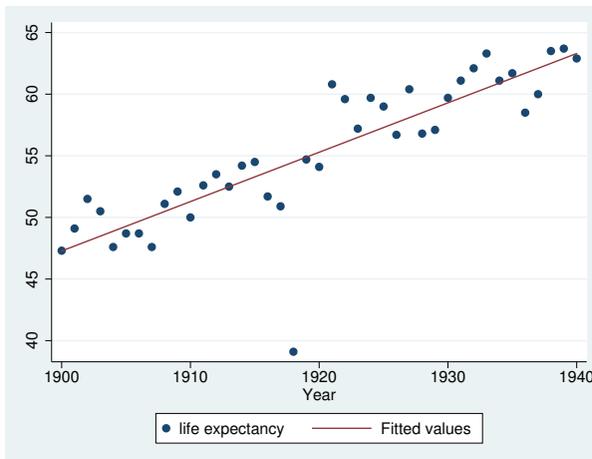
or we could graph these data as a twoway connected plot, marking both the points and connecting them with straight lines,

```
. twoway connected le year
```



or we could graph these data as a scatterplot and put on top of that the prediction from a linear regression of le on year,

```
. twoway (scatter le year) (lfit le year)
```



or we could graph these data in many other ways.

These all are examples of twoway graphs. What distinguishes a twoway graph is that it fits onto numeric y and x axes.

Each of what we produced above is called a *graph*. What appeared in the graphs are called *plots*. In the first graph, the plottype was a scatter; in the second, the plottype was a line; in the third, the plottype was connected; and in the fourth, there were two plots: a scatter combined with a line plot of a linear fit.

`twoway` provides many different plottypes. Some, such as `scatter` and `line`, simply render the data in different ways. Others, such as `lfit`, transform the data and render that. And still others, such as `function`, actually make up data to be rendered. This last class makes it easy to overlay $y = x$ lines or $y = f(x)$ functions on your graphs.

By the way, in case you are wondering, there are no errors in the above data. In 1918, there was an outbreak of influenza known as the 1918 Influenza Pandemic, which in the United States, was the worst epidemic ever known and which killed more citizens than all combat deaths of the 20th century.

Syntax

If we want to graph y_1 versus x and y_2 versus x , the formal way to type this is

```
. graph twoway (scatter y1 x) (scatter y2 x)
```

If we wanted y_1 versus x plotted with solid circles and y_2 versus x plotted with hollow circles, formally we would type

```
. graph twoway (scatter y1 x, ms(0)) (scatter y2 x, ms(0h))
```

If we wanted y_1 versus x plotted with solid circles and wanted a line graph for y_2 versus x , formally we would type

```
. graph twoway (scatter y1 x, ms(0)) (line y2 x, sort)
```

The `sort` option is included under the assumption that the data are not already sorted by x .

We have shown the formal way to type each of our requests, but few people would type that. First, most users omit the `graph`:

```
. twoway (scatter y1 x) (scatter y2 x)
. twoway (scatter y1 x, ms(0)) (scatter y2 x, ms(0h))
. twoway (scatter y1 x, ms(0)) (line y2 x, sort)
```

Second, most people use the `||`-separator notation rather than the `()`-binding notation:

```
. twoway scatter y1 x || scatter y2 x
. twoway scatter y1 x, ms(0) || scatter y2 x, ms(0h)
. twoway scatter y1 x, ms(0) || line y2 x, sort
```

Third, most people now omit the `twoway`:

```
. scatter y1 x || scatter y2 x
. scatter y1 x, ms(0) || scatter y2 x, ms(0h)
. scatter y1 x, ms(0) || line y2 x, sort
```

And finally, most people quickly realize that `scatter` allows us to plot more than one y variable against the same x variable:

```
. scatter y1 y2 x
. scatter y1 y2 x, ms(0 0h)
. scatter y1 x, ms(0) || line y2 x, sort
```

The third example did not change: in that example, we are combining a scatterplot and a line plot. Actually, in this particular case, there is a way we can combine that, too:

```
. scatter y1 y2 x, ms(0 i) connect(. l)
```

That we can combine `scatter` and `line` just happens to be an oddity of the examples we picked. It is important to understand that there is nothing wrong with any of the above ways of typing our request, and sometimes the wordier syntaxes are the only way to obtain what we want. If we wanted to graph y_1 versus x_1 and y_2 versus x_2 , the only way to type that is

```
. scatter y1 x1 || scatter y2 x2
```

or to type the equivalent in one of the wordier syntaxes above it. We have to do this because `scatter` (see [G-2] [graph twoway scatter](#)) draws a scatterplot against one x variable. Therefore, if we want two different x variables, we need two different `scatter`s.

In any case, we will often refer to the `graph twoway` command, even though, when we give the command, we will seldom type the `graph`, and mostly, we will not type the `twoway` either.

Multiple if and in restrictions

Each *plot* may have its own *if* *exp* and *in* *range* restrictions:

```
. twoway (scatter mpg weight if foreign, msymbol(0))
      (scatter mpg weight if !foreign, msymbol(0h))
```

Multiple *plots* in one `graph twoway` command draw one graph with multiple things plotted in it. The above will produce a scatter of `mpg` versus `weight` for foreign cars (making the points with solid circles) and a scatter of `mpg` versus `weight` for domestic cars (using hollow circles).

Also, the `graph twoway` command itself can have *if* *exp* and *in* *range* restrictions:

```
. twoway (scatter mpg weight if foreign, msymbol(0))
      (scatter mpg weight if !foreign, msymbol(0h)) if mpg>20
```

The `if mpg>20` restriction will apply to both `scatter`s.

We have chosen to show these two examples with the `()`-binding notation because it makes the scope of each *if* *exp* so clear. In `||`-separator notation, the commands would read

```
. twoway scatter mpg weight if foreign, msymbol(0) ||
      scatter mpg weight if !foreign, msymbol(0h)
```

and

```
. twoway scatter mpg weight if foreign, msymbol(0) ||
      scatter mpg weight if !foreign, msymbol(0h) || if mpg>20
```

or even

```
. scatter mpg weight if foreign, msymbol(0) ||
      scatter mpg weight if !foreign, msymbol(0h)
```

and

```
. scatter mpg weight if foreign, msymbol(0) ||
      scatter mpg weight if !foreign, msymbol(0h) || if mpg>20
```

We may specify `graph twoway` restrictions only, of course:

```
. twoway (scatter mpg weight) (lfit mpg weight) if !foreign
. scatter mpg weight || lfit mpg weight || if !foreign
```

twoway and plot options

`graph twoway` allows options, and the individual *plots* allow options. For instance, `graph twoway` allows the `saving()` option, and `scatter` (see [G-2] [graph twoway scatter](#)) allows the `msymbol()` option, which specifies the marker symbol to be used. Nevertheless, we do not have to keep track of which option belongs to which. If we type

```
. scatter mpg weight, saving(mygraph) msymbol(0h)
```

the results will be the same as if we more formally typed

```
. twoway (scatter mpg weight, msymbol(Oh)), saving(mygraph)
```

Similarly, we could type

```
. scatter mpg weight, msymbol(Oh) || lfit mpg weight, saving(mygraph)
```

or

```
. scatter mpg weight, msymbol(Oh) saving(mygraph) || lfit mpg weight
```

and, either way, the results would be the same as if we typed

```
. twoway (scatter mpg weight, msymbol(Oh))  
         (lfit mpg weight), saving(mygraph)
```

We may specify a `graph twoway` option “too deeply”, but we cannot go the other way. The following is an error:

```
. scatter mpg weight || lfit mpg weight ||, msymbol(Oh) saving(mygraph)
```

It is an error because we specified a `scatter` option where only a `graph twoway` option may be specified, and given what we typed, there is insufficient information for `graph twoway` to determine for which *plot* we meant the `msymbol()` option. Even when there is sufficient information (say that option `msymbol()` were not allowed by `lfit`), it would still be an error. `graph twoway` can reach in and pull out its options, but it cannot take from its options and distribute them back to the individual *plots*.