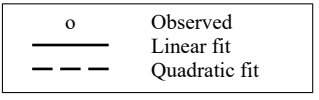
legend_options — Options for specifying legends

Description Quick start Syntax Options Remarks and examples Also see

Description

The `legend()` option allows you to control the look, contents, and placement of the legend. A sample legend is



The above legend has three *keys*. Each key is composed of a *symbol* and *descriptive text* describing the symbol (whatever the symbol might be, be it a marker, a line, or a color swatch).

The `plegend()` option is similar to the `legend()` option but is used with contour-line plots; see [G-2] **graph twoway contourline**. The look and placement of a contour-line plot legend is unique from the standard legend.

The `clegend()` option controls the look, contents, and placement of the legend in contour plots; see [G-2] **graph twoway contour**. The look and placement of a contour plot legend is unique from the standard legend. For details on `clegend()`, see [G-3] ***clegend_option***.

Quick start

Change the order of the plots in the legend to `y3`, `y2`, and `y1` when using a command that accepts multiple y variables

graph_command `y1 y2 y3 ..., ... legend(order(3 2 1))`

Same as above, and change the labels associated with each plot

graph_command `y1 y2 y3 ..., ... ///`
`legend(order(3 "y3 var" 2 "y2 var" 1 "y1 var"))`

Change the label of `y2` in the legend without changing the order

graph_command `y1 y2 y3 ..., ... legend(label(2 "y2 var"))`

Place all the legend keys and labels in 2 columns

graph_command `..., ... legend(cols(2))`

Place all the legend keys and labels in 1 row

graph_command `..., ... legend(rows(1))`

Do not display the legend

graph_command `..., ... legend(off)`

Use medium-sized font for legend key text

graph_command `..., ... legend(size(medium))`

Make font size on legend keys 20% larger than the default

graph_command `..., ... legend(size(*1.2))`

Make key size 50% wider than the default

graph_command ..., ... `legend(symxsize(*1.5))`

Place the legend to the right of the plot region, or at 3 o'clock using clock positions

graph_command ..., ... `legend(position(3))`

Place the legend inside the plot region in the upper right corner

graph_command ..., ... `legend(position(0) bplacement(neast))`

Syntax

legend_options	Description
`legend(`[*contents*] [*location*]`)`	standard legend, contents and location
`plegend(`[*contents*] [*location*]`)`	`contourline` legend, contents and location
`clegend(`[*suboptions*]`)`	`contour` plot legend; see [G-3] ***clegend_option***

`legend()`, `plegend()`, and `clegend()` are *merged-implicit*; see [G-4] **Concept: repeated options**.

where *contents* and *location* specify the contents and the location of the legends.

contents	Description
`order(`*orderinfo*`)`	which keys appear and their order
`label(`*labelinfo*`)`	override text for a key
`lastlabel(`*labelinfo*`)`	override text for a key, even if the text is modified by `order()`; apply specified text as the last label
`holes(`*numlist*`)`	positions in legend to leave blank
`all`	generate keys for all symbols
`style(`*legendstyle*`)`	overall style of legend
`cols(`#`)`	# of keys per line
`rows(`#`)`	or # of rows
[`no`]`colfirst`	"1, 2, 3" in row 1 or in column 1?
[`no`]`textfirst`	symbol-text or text-symbol?
`stack`	symbol/text vertically stacked
`rowgap(`*size*`)`	gap between lines
`colgap(`*size*`)`	gap between columns
`symplacement(`*compassdirstyle*`)`	alignment/justification of key's symbol
`keygap(`*size*`)`	gap between symbol-text
`symysize(`*size*`)`	height for key's symbol
`symxsize(`*size*`)`	width for key's symbol
`textwidth(`*size*`)`	width for key's descriptive text
`forcesize`	always respect `symysize()`, `symxsize()`, and `textwidth()`
`bmargin(`*marginstyle*`)`	outer margin around legend
textbox_options	appearance of key's descriptive text
title_options	titles, subtitles, notes, captions
`region(`*roptions*`)`	borders and background shading

`all` has no effect on `plegend()`.

location	Description
`off` or `on`	suppress or force display of legend
`position(`*clockposstyle*`)`	where legend appears
`ring(`*ringposstyle*`)`	where legend appears (detail)
`bplacement(`*compassdirstyle*`)`	placement of legend when positioned in the plotregion
`span`	"centering" of legend
`at(#)`	allowed with `by()` only

See *Where legends appear* under *Remarks and examples* below, and see *Positioning of titles* in [G-3] ***title_options*** for definitions of *clockposstyle* and *ringposstyle*.

orderinfo, the argument allowed by `legend(order())`, is defined as

{ # | - } ["*text*" ["*text*" ...]]

labelinfo, the argument allowed by `legend(label())`, is defined as

"*text*" ["*text*" ...]

roptions, the arguments allowed by `legend(region())`, include

roptions	Description
`style(`*areastyle*`)`	overall style of region
`color(`*colorstyle*`)`	line + fill color and opacity of region
`fcolor(`*colorstyle*`)`	fill color and opacity of region
`lstyle(`*linestyle*`)`	overall style of border
`lcolor(`*colorstyle*`)`	color and opacity of border
`lwidth(`*linewidthstyle*`)`	thickness of border
`lpattern(`*linepatternstyle*`)`	border pattern (solid, dashed, etc.)
`lalign(`*linealignmentstyle*`)`	border alignment (inside, outside, center)
`margin(`*marginstyle*`)`	margin between border and contents of legend

Options

`legend(`*contents*`,` *location*`)` defines the contents of the standard legend, along with how it is to look, and whether and where it is to be displayed.

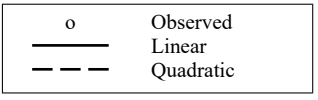
`plegend(`*contents*`,` *location*`)` defines the contents of the legend for a contour-line plot (see [G-2] **graph twoway contourline**), along with how it is to look, and whether and where it is to be displayed.

Content suboptions for use with legend( ) and plegend( )

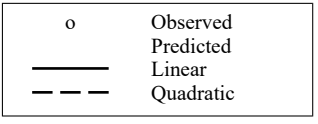
`order(`*orderinfo*`)` specifies which keys are to appear in the legend and the order in which they are to appear.

`order(# # ...)` is the usual syntax. `order(1 2 3)` would specify that key 1 is to appear first in the legend, followed by key 2, followed by key 3. `order(1 2 3)` is the default if there are three keys. If there were four keys, `order(1 2 3 4)` would be the default, and so on. If there were four keys and you specified `order(1 2 3)`, the fourth key would not appear in the legend. If you specified `order(2 1 3)`, first key 2 would appear, followed by key 1, followed by key 3.

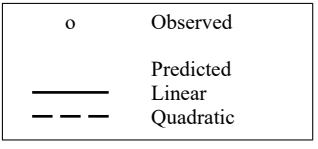
A dash specifies that text be inserted into the legend. For instance, `order(1 2 - "`*text*`" 3)` specifies key 1 appear first, followed by key 2, followed by the text *text*, followed by key 3. Imagine that the default key were



Specifying `order(1 - "Predicted" 2 3)` would produce



and specifying `order(1 - " " "Predicted" 2 3)` would produce



Note carefully the specification of a blank for the first line of the text insertion; we typed `" "` and not `""`. Typing `""` would insert nothing.

You may also specify quoted text after # to override the descriptive text associated with a symbol. Specifying `order(1 "Observed 1992" - " " "Predicted" 2 3)` would change "Observed" in the above to "Observed 1992". It is considered better style, however, to use the `label()` suboption to relabel symbols.

`label(#"`*text*`" ["`*text*`" ...])` specifies the descriptive text to be displayed next to the #th key. Multiline text is allowed. Specifying `label(1 "Observed 1992")` would change the descriptive text associated with the first key to be "Observed 1992". Specifying `label(1 "Observed" "1992-1993")` would change the descriptive text to contain two lines, "Observed" followed by "1992–1993".

The descriptive text of only one key may be changed per `label()` suboption. Specify multiple `label()` suboptions when you wish to change the text of multiple keys.

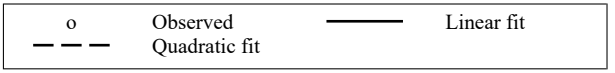
`lastlabel(#"`*text*`" ["`*text*`" ...])` specifies the descriptive text to be displayed next to the #th key. This text is the last label to be applied and overrides any text specified in `order()` or `label()`. This suboption is useful for modifying the descriptive text for a key that has already been modified by the `order()` suboption. If the text for a key is modified by the `order()` suboption, you cannot change the label with the `label()` suboption, but you can with `lastlabel()`. For example, when you estimate margins at unique values with `margins catvar1, over(catvar2)` and then plot the values with `marginsplot`, `marginsplot` uses the `order()` suboption to create the legend. If you would like to change the text for one of the resulting keys, you can do so with `lastlabel()` but not with `label()`.

`lastlabel()` uses the key index that appears on the graph instead of the original index. For example, if you specified `order(5 "key5" 4 "key4")`, which places the fifth key first in the legend, `lastlabel(1 "newtext1")` would change "key5" to "newtext1".

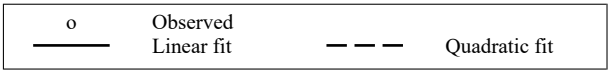
The descriptive text of only one key may be changed via the `lastlabel()` suboption. Specify multiple `lastlabel()` suboptions when you wish to change the text of multiple keys.

`holes(`*numlist*`)` specifies where gaps appear in the presentation of the keys. `holes()` has an effect only if the keys are being presented in more than one row and more than one column.

Consider a case in which the default key is



Specifying `holes(2)` would result in



Here `holes(2)` would have the same effect as specifying `order(1 - " " 2 3)`, and as a matter of fact, there is always an `order()` command that will achieve the same result as `holes()`. `order()` has the added advantage of working in all cases.

`all` specifies that keys be generated for all the plots of the graph, even when the same symbol is repeated. The default is to generate keys only when the symbols are different, which is determined by the overall style. For example, in

```
. scatter ylow yhigh x, pstyle(p1 p1) || ...
```

there would be only one key generated for the variables `ylow` and `yhigh` because they share the style `p1`. That single key's descriptive text would indicate that the symbol corresponded to both variables. If, on the other hand, you typed

```
. scatter ylow yhigh x, pstyle(p1 p1) legend(all) || ...
```

then separate keys would be generated for `ylow` and `yhigh`.

In the above example, do not confuse our use of `scatter`'s option `pstyle()` with `legend()`'s suboption `legend(style())`. The `pstyle()` option sets the overall style for the rendition of the points. `legend()`'s `style()` suboption is documented directly below.

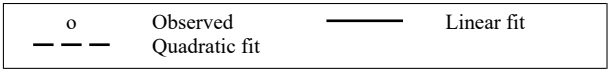
`all` has no effect on `plegend()`.

`style(`*legendstyle*`)` specifies the overall look of the legend—whether it is presented horizontally or vertically, how many keys appear across the legend if it is presented horizontally, etc. The options listed below allow you to change each attribute of the legend, but `style()` is the starting point.

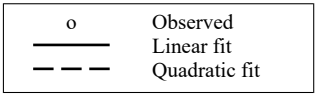
You need not specify `style()` just because there is something you want to change. You specify `style()` when another style exists that is exactly what you desire or when another style would allow you to specify fewer changes to obtain what you want.

See [G-4] ***legendstyle*** for a list of available legend styles.

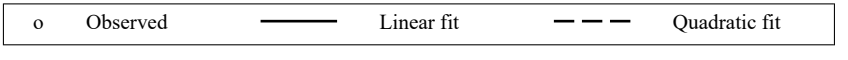
`cols(#)` and `rows(#)` are alternatives; they specify in how many columns or rows (lines) the keys are to be presented. The usual default is `cols(2)`, which means that legends are to take two columns:



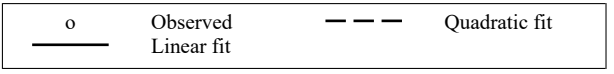
`cols(1)` would force a vertical arrangement,



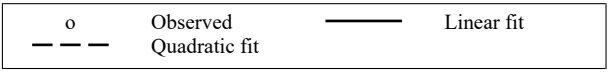
and `rows(1)` would force a horizontal arrangement:



`colfirst` and `nocolfirst` determine whether, when the keys are presented in multiple columns, keys are to read down or to read across, resulting in this

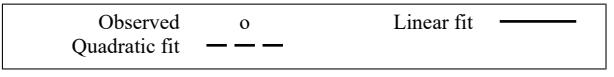


or this

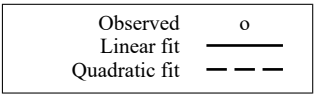


The usual default is `nocolfirst`, so `colfirst` is the option.

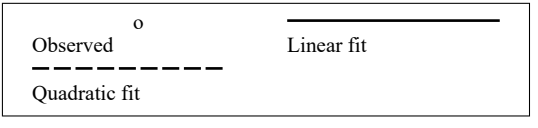
`textfirst` and `notextfirst` specify whether the keys are to be presented as descriptive text followed by the symbol or the symbol followed by descriptive text. The usual default is `notextfirst`, so `textfirst` is the option. `textfirst` produces keys that look like this



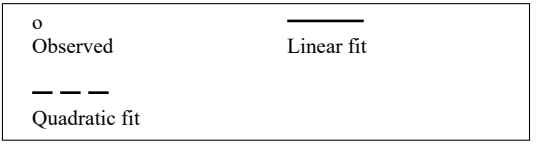
and `textfirst cols(1)` produces



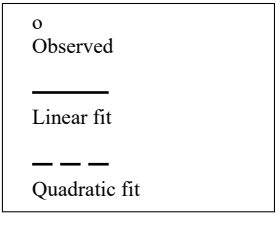
`stack` specifies that the symbol-text be presented vertically with the symbol on top (or with the descriptive text on top if `textfirst` is also specified). `legend(stack)` would produce



`legend(stack symplacement(left) symxsize(13) forcesize rowgap(4))` would produce



`stack` tends to be used to produce single-column keys. `legend(cols(1) stack symplacement(left) symxsize(13) forcesize rowgap(4))` produces

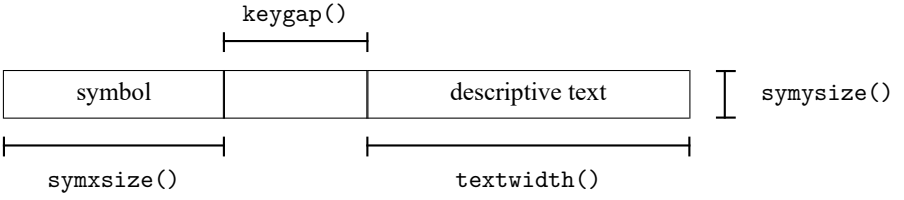


This is the real use of `stack`: to produce narrow, vertical keys.

`rowgap(`*size*`)` and `colgap(`*size*`)` specify the distance between lines and the distance between columns. The defaults are `rowgap(1.4)` and `colgap(4.9)`. See [G-4] ***size***.

`symplacement(`*compassdirstyle*`)` specifies how symbols are justified in the key. The default is `symplacement(center)`, meaning that they are vertically and horizontally centered. The two most commonly specified alternatives are `symplacement(right)` (right alignment) and `symplacement(left)` (left alignment). See [G-4] ***compassdirstyle*** for other alignment choices.

`keygap(`*size*`)`, `symysize(`*size*`)`, `symxsize(`*size*`)`, and `textwidth(`*size*`)` specify the height and width to be allocated for the key and the key's symbols and descriptive text:



The defaults are

`symxsize()`	13
`keygap()`	2
`textwidth()`	according to longest descriptive text line
`symysize()`	according to height of font (*)

(*) The size of the font is set by the *textbox_option* `size(`*size*`)`; see *textbox_options* below.

Markers are placed in the symbol area, centered according to `symplacement()`.

Lines are placed in the symbol area vertically according to `symplacement()` and horizontally are drawn to length `symxsize()`.

Color swatches fill the `symysize()` × `symxsize()` area.

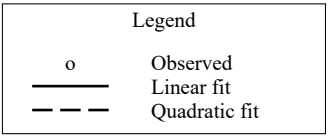
See [G-4] ***size*** for information on specifying sizes.

`forcesize` causes the sizes specified by `symysize()` and `symxsize()` to be respected. If `forcesize` is not specified, once all the symbols have been placed for all the keys, the symbol area is compressed (or expanded) to be no larger than necessary to contain the symbols.

`bmargin(`*marginstyle*`)` specifies the outer margin around the legend. That is, it specifies how close other things appearing near to the legend can get. Also see suboption `margin()` under *Suboptions for use with legend(region( ))* below for specifying the inner margin between the border and contents. See [G-4] ***marginstyle*** for a list of margin choices.

textbox_options affect the rendition of the descriptive text associated with the keys. These are described in [G-3] ***textbox_options***. One of the most commonly specified *textbox_options* is `size(`*size*`)`, which specifies the size of font to be used for the descriptive text.

title_options allow placing titles, subtitles, notes, and captions on legends. For instance, `legend(col(1) subtitle("Legend"))` produces



Note our use of `subtitle()` and not `title()`; `title()`s are nearly always too big. See [G-3] ***title_options***.

`region(`*roptions*`)` specifies the border and shading of the legend. You could remove the border around the legend by specifying `legend(region(lstyle(none)))` (thus doing away with the line) or `legend(region(lcolor(none)))` (thus making the line invisible). You could also give the legend a gray background tint by specifying `legend(region(fcolor(gs5)))`. See *Suboptions for use with legend(region( ))* below.

Suboptions for use with legend(region( ))

`style(`*areastyle*`)` specifies the overall style of the region in which the legend appears. The other suboptions allow you to change the region's attributes individually, but `style()` provides the starting point. See [G-4] ***areastyle*** for a list of choices.

`color(`*colorstyle*`)` specifies the color and opacity of the background of the legend and the line used to outline it. See [G-4] ***colorstyle*** for a list of color choices.

`fcolor(`*colorstyle*`)` specifies the background (fill) color and opacity for the legend. See [G-4] ***colorstyle*** for a list of color choices.

`lstyle(`*linestyle*`)` specifies the overall style of the line used to outline the legend, which includes its pattern (solid, dashed, etc.), its thickness, and its color. The other suboptions listed below allow you to change the line's attributes individually, but `lstyle()` is the starting point. See [G-4] ***linestyle*** for a list of choices.

`lcolor(`*colorstyle*`)` specifies the color and opacity of the line used to outline the legend. See [G-4] ***colorstyle*** for a list of color choices.

`lwidth(`*linewidthstyle*`)` specifies the thickness of the line used to outline the legend. See [G-4] ***linewidthstyle*** for a list of choices.

`lpattern(`*linepatternstyle*`)` specifies whether the line used to outline the legend is solid, dashed, etc. See [G-4] ***linepatternstyle*** for a list of choices. When `lpattern()` is specified, the line alignment is always `center`; thus, `lalign()` is ignored.

`lalign(`*linealignmentstyle*`)` specifies whether the line used to outline the area is inside, outside, or centered. See [G-4] ***linealignmentstyle*** for a list of alignment choices.

`margin(`*marginstyle*`)` specifies the inner margin between the border and the contents of the legend. Also see `bmargin()` under *Content suboptions for use with legend( ) and plegend( )* above for specifying the outer margin around the legend. See [G-4] ***marginstyle*** for a list of margin choices.

Location suboptions for use with legend( )

`off` and `on` determine whether the legend appears. The default is `on` when more than one symbol (meaning marker, line style, or color swatch) appears in the legend. In those cases, `legend(off)` will suppress the display of the legend.

`position(`*clockposstyle*`)`, `ring(`*ringposstyle*`)`, and `bplacement(`*compassdirstyle*`)` override the default location of the legend, which is to the right of the plot region for the default scheme, `stcolor`. `position()` specifies a direction [*sic*] according to the hours on the dial of a 12-hour clock, and `ring()` specifies the distance from the plot region.

`ring(0)` is defined as being inside the plot region itself and allows you to place the legend inside the plot. `ring(`k`)`, $k > 0$, specifies positions outside the plot region; the larger the `ring()` value, the farther away from the plot region the legend is. `ring()` values may be integers or nonintegers and are treated ordinally.

When `ring(0)` is specified, `bplacement()` further specifies where in the plot region the legend is placed. `bplacement(seast)` places the legend in the southeast (lower-right) corner of the plot region.

`position(12)` puts the legend directly above the plot region (assuming `ring()` > 0), `position(3)` directly to the right of the plot region, and so on.

See *Where legends appear* under *Remarks and examples* below and *Positioning of titles* in [G-3] ***title_options*** for more information on the `position()` and `ring()` suboptions.

`span` specifies that the legend be placed in an area spanning the entire width (or height) of the graph rather than an area spanning the plot region. This affects whether the legend is centered with respect to the plot region or the entire graph. See *Spanning* in [G-3] ***title_options*** for more information on `span`.

`at(#)` is for use only when the *twoway_option* `by()` is also specified. It specifies that the legend appear in the #th position of the $R \times C$ array of plots, using the same coding as `by(..., holes())`. See *Use of legends with by( )* under *Remarks and examples* below, and see [G-3] ***by_option***.

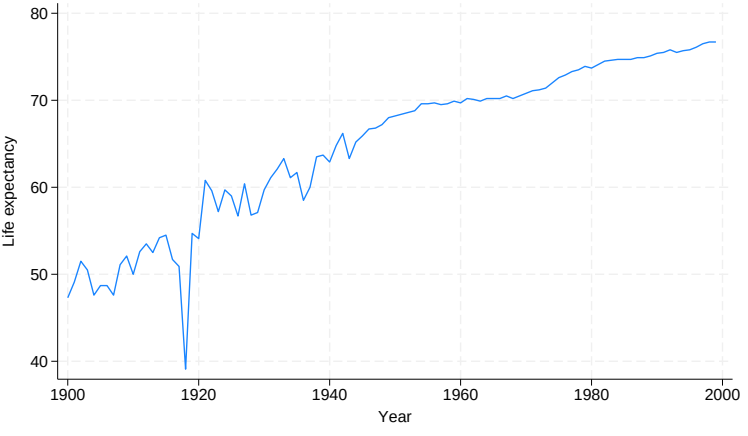
Remarks and examples

Remarks are presented under the following headings:

When legends appear

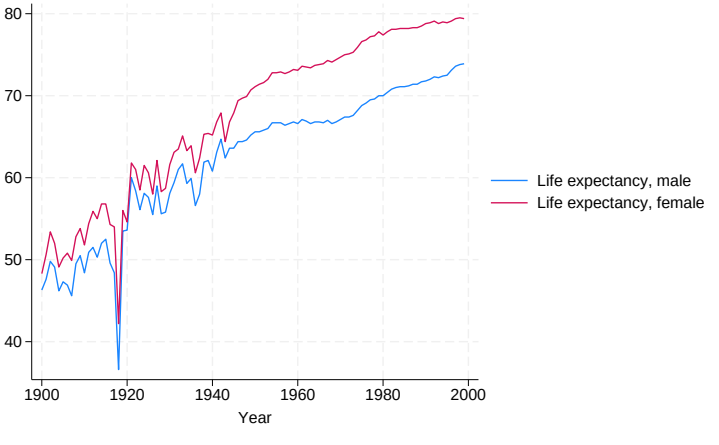
Standard legends appear on the graph whenever more than one symbol is used, where symbol is broadly defined to include markers, lines, and color swatches (such as those used to fill bars). When you draw a graph with only one symbol on it, such as no legend appears.

```
. use https://www.stata-press.com/data/r19/uslifeexp
(US life expectancy, 1900-1999)
. line le year
```



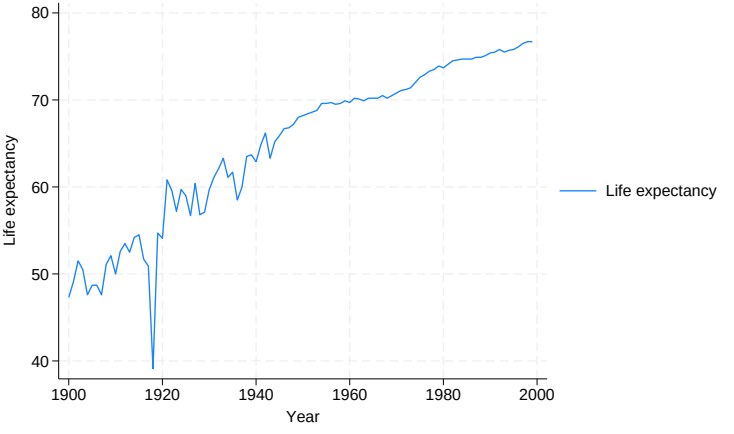
When there is more than one symbol, a legend is added:

```
. line le_m le_f year
```



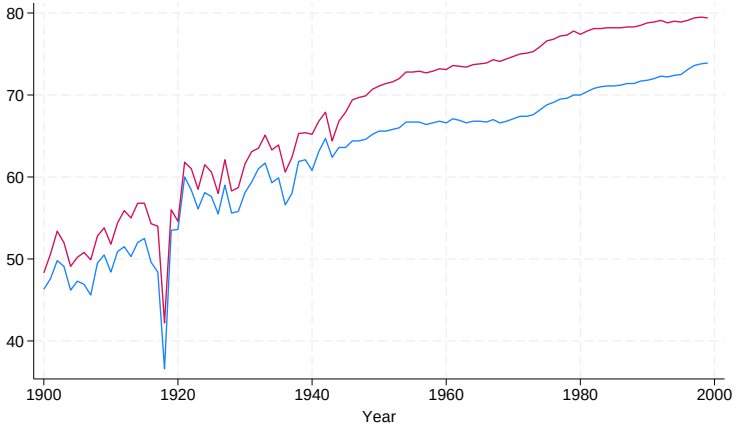
Even when there is only one symbol, a legend is constructed. It is merely not displayed. Specifying `legend(on)` forces the display of the legend:

```
. line le year, legend(on)
```



Similarly, when there is more than one symbol and you do not want the legend, you can specify `legend(off)` to suppress it:

```
. line le_m le_f year, legend(off)
```



The contents of legends

By default, the descriptive text for legends is obtained from the variable's variable label; see [D] **label**. If the variable has no variable label, the variable's name is used. In

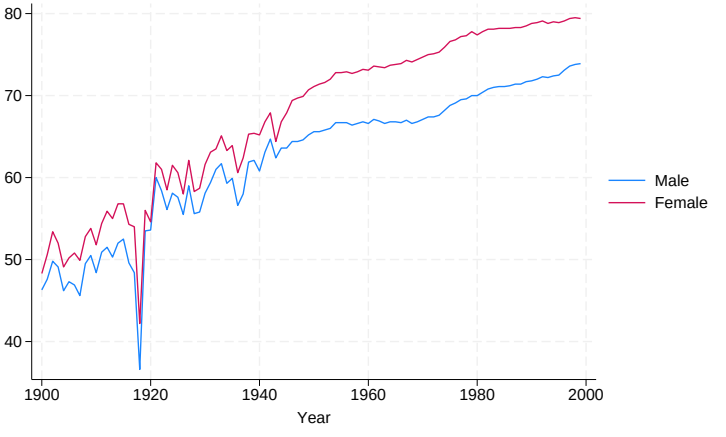
```
. line le_m le_f year
```

the variable `le_m` had previously been labeled "Life expectancy, male", and the variable `le_f` had been labeled "Life expectancy, female". In the legend of this graph, repeating "life expectancy" is unnecessary. The graph would be improved if we changed the labels on the variables:

```
. label var le_m "Male"
. label var le_f "Female"
. line le_m le_f year
```

We can also specify the `label()` suboption to change the descriptive text. We obtain the same visual result without relabeling our variables:

```
. line le_m le_f year, legend(label(1 "Male") label(2 "Female"))
```



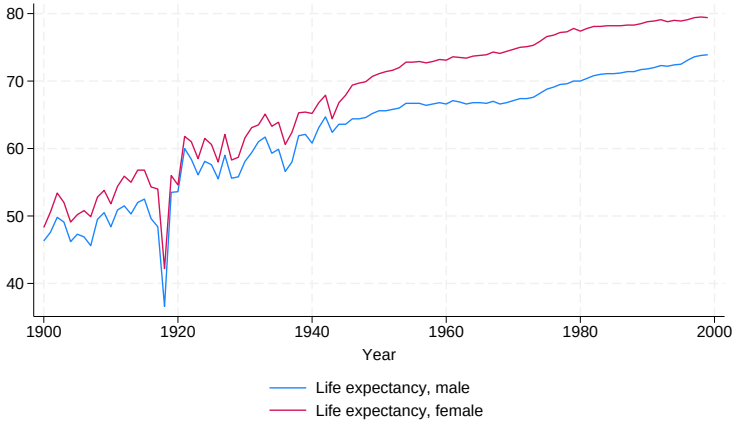
The descriptive text for contour-line legends is the values for the contour lines of the z variable.

Where legends appear

By default, legends appear to the right of the plot region at `position(3) ring(4)`. Suboptions `position()` and `ring()` specify the location of the legend. `position()` specifies on which side of the plot region the legend appears—`position(3)` means 3 o'clock—and `ring()` specifies the distance from the plot region—`ring(4)` means farther out than the *title_option* `b2title()`; see [G-3] ***title_options***.

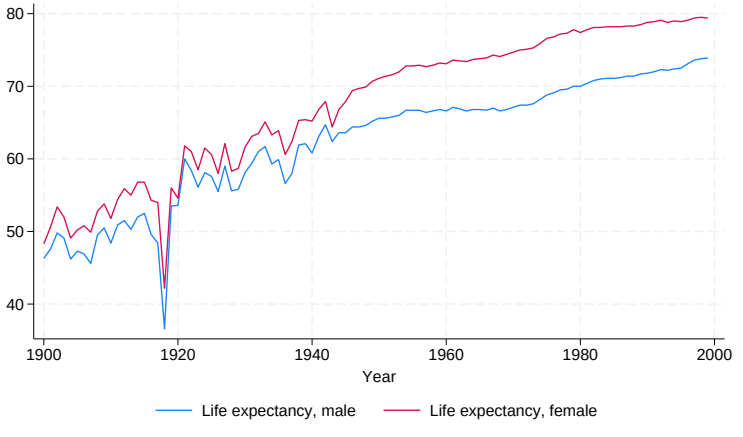
If we specify `legend(position(6))`, the legend will be moved to the 6 o'clock position:

```
. line le_m le_f year, legend(pos(6))
```



The legend was moved to the bottom of the graph. In this case, we may want to specify the `col(2)` option:

```
. line le_m le_f year, legend(pos(6) col(2))
```



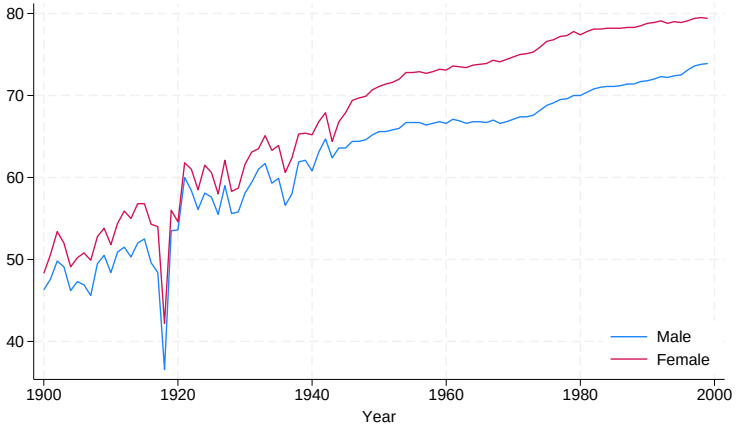
As a matter of syntax, we could have typed the above command with two `legend()` options

```
. line le_m le_f year, legend(pos(6)) legend(col(2))
```

instead of one combined: `legend(pos(6) col(2))`. We would obtain the same results either way.

`ring()`—the suboption that specifies the distance from the plot region—is seldom specified, but, when it is specified, `ring(0)` is the most useful. `ring(0)` specifies that the legend be moved inside the plot region:

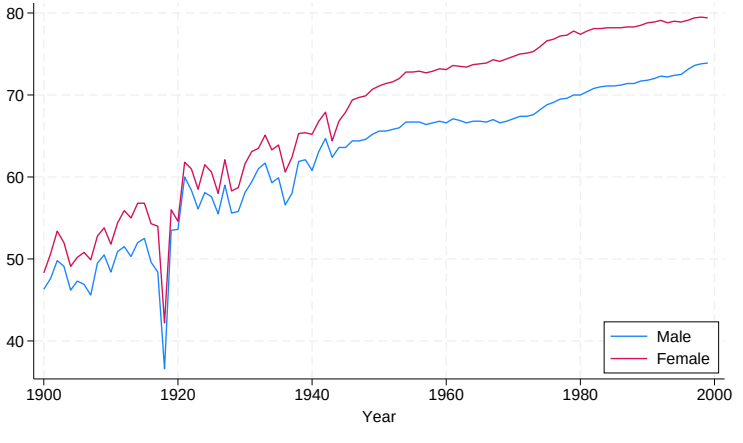
```
. line le_m le_f year, legend(pos(5) ring(0)
                       label(1 "Male") label(2 "Female"))
```



Our use of `position(5) ring(0)` put the legend inside the plot region, at 5 o'clock, meaning in the bottom right corner. Had we specified `position(2) ring(0)`, the legend would have appeared in the top left corner.

We might now add a border to the legend:

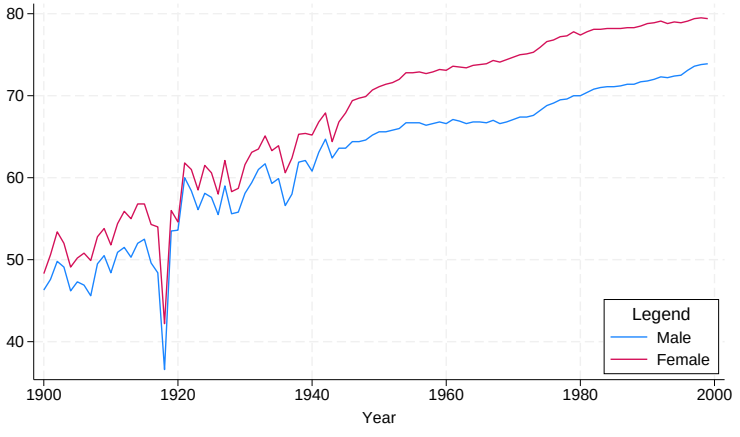
```
. line le_m le_f year, legend(pos(5) ring(0)
                       label(1 "Male") label(2 "Female")
                       region(lcolor(black)))
```



Putting titles on legends

Legends may include titles:

```
. line le_m le_f year, legend(pos(5) ring(0)
                       label(1 "Male") label(2 "Female")
                       region(lcolor(black))
                       subtitle("Legend"))
```



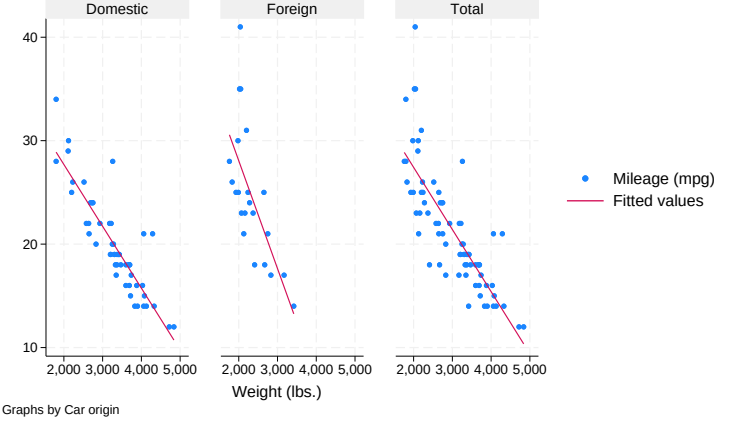
Above we specified `subtitle()` rather than `title()` because, when we tried `title()`, it seemed too big.

Legends may also contain `notes()` and `captions()`; see [G-3] ***title_options***.

Use of legends with by()

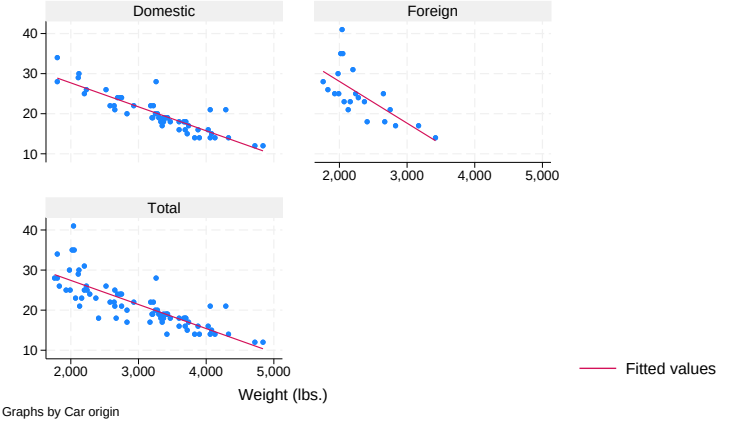
If you want the legend to be located in the default location, no special action need be taken when you use `by()`:

```
. use https://www.stata-press.com/data/r19/auto, clear
(1978 automobile data)
. scatter mpg weight || lfit mpg weight ||, by(foreign, total row(1))
```



If, however, you wish to move the legend, you must distinguish between `legend(`*contents*`)` and `legend(`*location*`)`. The former must appear outside the `by()`. The latter appears inside the `by()`. Below, we specify `order(2)` to indicate that we only want the second key in the legend; the first key is for the scatterplot:

```
. scatter mpg weight || lfit mpg weight ||,
          legend(order(2))
          by(foreign, total legend(pos(4)))
```



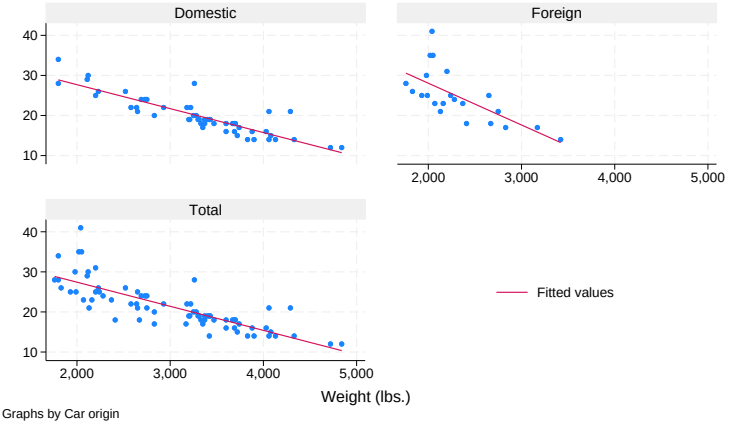
`legend(order(2))` was placed in the command just where we would place it had we not specified `by()` but that `legend(pos(4))` was moved to be inside the `by()` option. We did that because the `order()` suboption is documented under *contents* in the syntax diagram, whereas `position()` is documented under *location*. The logic is that, at the time the individual plots are constructed, they must know what style of key they are producing. The placement of the key, however, is something that happens when the overall graph is assembled, so you must indicate to `by()` where the key is to be placed. Were we to forget this distinction and simply to type

```
. scatter mpg weight || lfit mpg weight ||,
          legend(order(2) pos(4))
          by(foreign, total)
```

the `pos(4)` suboption would have been ignored.

Another *location* suboption is provided for use with `by()`: `at(#)`. You specify this option to tell `by()` to place the legend inside the $R \times C$ array it creates:

```
. scatter mpg weight || lfit mpg weight ||,
          legend(order(2))
          by(foreign, total legend(at(4) pos(0)))
```



In the above, we specified `at(4)` to mean that the key was to appear in the fourth position of the 2×2 array, and we specified `pos(0)` to move the key to the middle (0 o'clock) position within the cell.

In some cases, particularly when we have more than two graphs per row, it can be helpful to put the legend at the bottom of the graph. We could do this by specifying `pos(6)`.

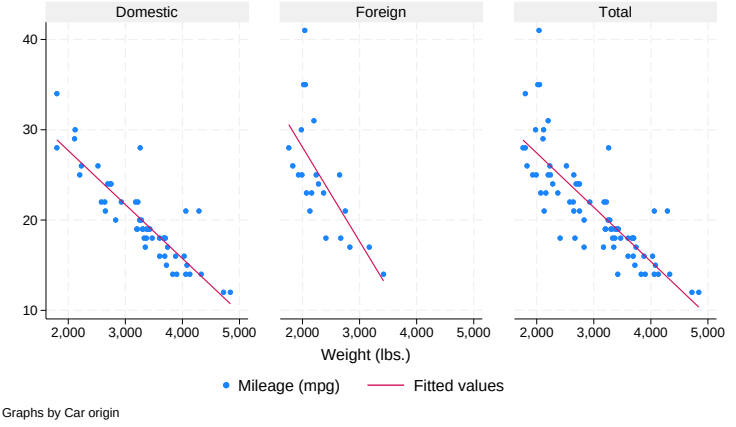
```
. scatter mpg weight || lfit mpg weight ||,
          by(foreign, total row(1) legend(pos(6)))
```

Alternatively, we can change the scheme to one that places the legend at the bottom (in addition to other changes). For instance, we could type

```
. scatter mpg weight || lfit mpg weight ||,
          by(foreign, total row(1)) scheme(stcolor_alt)
```

A third option, which we demonstrate here, is to specify the *bystyle* `altleg` in the `style()` option to move the legend to the bottom.

```
. scatter mpg weight || lfit mpg weight ||,
          by(foreign, total row(1) style(altleg))
```



Finally, if you wish to suppress the legend, you must specify the `legend(off)` inside the `by()` option:

```
. scatter mpg weight || lfit mpg weight ||,
          by(foreign, total row(1) legend(off))
```

Problems arising with or because of legends

There are three potential problems associated with legends:

1. Text may flow outside the border of the legend box.
2. The presence of the legend may cause the title of the y axis to run into the values labeled on the axis.
3. The presence of the legend may cause the graph to be too narrow.

The first problem arises because Stata uses an approximation to obtain the width of a text line. One solution is to specify `region(margin())` to add margin space around the legend:

```
. graph ..., ... legend(region(margin(marginstyle)))
```

Other solutions are available, such as `rows()` and `cols()`; see *Syntax*.

The second problem arises when the key is in the 6 o'clock location (`position(6)`) and the descriptive text for one or more of the keys is long. In `position(6)`, the borders of the key are supposed to line up with the borders of the plot region. Usually the plot region is wider than the key, so the key is expanded to fit below it. When the key is wider than the plot region, however, it is the plot region that is widened. As the plot region expands, it will eat away at whatever is at its side, namely, the y axis labels and title. Margins will disappear. In extreme cases, the title will be printed on top of the labels, and the labels themselves may end up on top of the axis!

The solution to this problem is to shorten the descriptive text, either by using fewer words or by breaking the long description into multiple lines. Use the `legend(label(#` *"text"*`))` option to modify the longest line of the descriptive text.

The third problem arises when the key is in the 3 o'clock location (`position(3)`) and the descriptive text for one or more of the keys is long. One solution is similar to shorten the descriptive text as described above. Alternatively, the legend can be moved to the bottom by specifying `position(6)` or specifying a scheme such as `stcolor_alt`, which places the legend at the bottom.

Also see

[G-3] ***title_options*** — Options for specifying titles