Syntax

```
_graph box yvars [if] [in] [weight] [ , options ]

_graph hbox yvars [if] [in] [weight] [ , options ]
```

where `yvars` is a `varlist`

**options**

<table>
<thead>
<tr>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>group_options</code></td>
<td>groups over which boxes are drawn</td>
</tr>
<tr>
<td><code>yvar_options</code></td>
<td>variables that are the boxes</td>
</tr>
<tr>
<td><code>boxlook_options</code></td>
<td>how the boxes look</td>
</tr>
<tr>
<td><code>legending_options</code></td>
<td>how variables are labeled</td>
</tr>
<tr>
<td><code>axis_options</code></td>
<td>how numerical y axis is labeled</td>
</tr>
<tr>
<td><code>title_and_other_options</code></td>
<td>titles, added text, aspect ratio, etc.</td>
</tr>
</tbody>
</table>

Each is defined below.

**group_options**

<table>
<thead>
<tr>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>over(varname[, over_subopts])</code></td>
<td>categories; option may be repeated</td>
</tr>
<tr>
<td><code>nofill</code></td>
<td>omit empty categories</td>
</tr>
<tr>
<td><code>missing</code></td>
<td>keep missing value as category</td>
</tr>
<tr>
<td><code>allcategories</code></td>
<td>include all categories in the dataset</td>
</tr>
</tbody>
</table>

**yvar_options**

<table>
<thead>
<tr>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ascategory</code></td>
<td>treat <code>yvars</code> as first <code>over()</code> group</td>
</tr>
<tr>
<td><code>asyvars</code></td>
<td>treat first <code>over()</code> group as <code>yvars</code></td>
</tr>
<tr>
<td><code>cw</code></td>
<td>calculate variable statistics omitting missing values of any variable</td>
</tr>
</tbody>
</table>
### boxlook_options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nooutsides</td>
<td>do not plot outside values</td>
</tr>
<tr>
<td>box(#, barlook_options)</td>
<td>look of #th box</td>
</tr>
<tr>
<td>pcycle(#)</td>
<td>box styles before pstyles recycle</td>
</tr>
<tr>
<td>intensity([*]#)</td>
<td>intensity of fill</td>
</tr>
<tr>
<td>lintensity([*]#)</td>
<td>intensity of outline</td>
</tr>
<tr>
<td>medtype(line</td>
<td>cline</td>
</tr>
<tr>
<td>medline(line_options)</td>
<td>look of line if medtype(cline)</td>
</tr>
<tr>
<td>medmarker(marker_options)</td>
<td>look of marker if medtype(marker)</td>
</tr>
<tr>
<td>cwhiskers</td>
<td>use custom whiskers</td>
</tr>
<tr>
<td>lines(line_options)</td>
<td>look of custom whiskers</td>
</tr>
<tr>
<td>alsize(#)</td>
<td>width of adjacent line; default is 67</td>
</tr>
<tr>
<td>capsize(#)</td>
<td>height of cap on adjacent line; default is 0</td>
</tr>
<tr>
<td>marker(#, marker_options marker_label_options)</td>
<td>look of #th marker and label for outside values</td>
</tr>
<tr>
<td>outergap([*]#)</td>
<td>gap between edge and first box and between last box and edge</td>
</tr>
<tr>
<td>boxgap(#)</td>
<td>gap between boxes; default is 33</td>
</tr>
</tbody>
</table>


### legending_options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>legend_options</td>
<td>control of yvar legend</td>
</tr>
<tr>
<td>nolabel</td>
<td>use yvar names, not labels, in legend</td>
</tr>
<tr>
<td>yvaroptions(over_subopts)</td>
<td><code>over_subopts</code> for yvars; seldom specified</td>
</tr>
<tr>
<td>showyvars</td>
<td>label yvars on x axis; seldom specified</td>
</tr>
</tbody>
</table>

See [G-3] `legend_options`.

### axis_options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yalternate</td>
<td>put numerical y axis on right (top)</td>
</tr>
<tr>
<td>xalternate</td>
<td>put categorical x axis on top (right)</td>
</tr>
<tr>
<td>yreverse</td>
<td>reverse y axis</td>
</tr>
<tr>
<td>axis_scale_options</td>
<td>y-axis scaling and look</td>
</tr>
<tr>
<td>axis_label_options</td>
<td>y-axis labeling</td>
</tr>
<tr>
<td>ytitle(...)</td>
<td>y-axis titling</td>
</tr>
</tbody>
</table>

**Title and Other Options**

- **text(...)**: add text on graph; x range [0, 100]
- **yline(...)**: add y lines to graph
- **aspect_option**: constrain aspect ratio of plot region
- **std_options**: titles, graph size, saving to disk
- **by(varlist, ...)**: repeat for subgroups


**Description**

The **over_subopts**—used in over(varname, over_subopts) and, on rare occasion, in yvaroptions(over_subopts)—are

<table>
<thead>
<tr>
<th>over_subopts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>add total group</td>
</tr>
<tr>
<td>relabel(# &quot;text&quot; ...)</td>
<td>change axis labels</td>
</tr>
<tr>
<td>label( cat_axis_label_options )</td>
<td>rendition of labels</td>
</tr>
<tr>
<td>axis( cat_axis_line_options )</td>
<td>rendition of axis line</td>
</tr>
<tr>
<td>gap([*]#)</td>
<td>gap between boxes within over() category</td>
</tr>
<tr>
<td>sort(varname)</td>
<td>put boxes in prespecified order</td>
</tr>
<tr>
<td>sort(#)</td>
<td>put boxes in median order</td>
</tr>
<tr>
<td>descending</td>
<td>reverse default or specified box order</td>
</tr>
</tbody>
</table>


aweights, fweights, and pweights are allowed; see [U] 11.1.6 weight and see note concerning weights in [D] collapse.

**Menu**

Graphics > Box plot

**Description**

`graph box` draws vertical box plots. In a vertical box plot, the y axis is numerical, and the x axis is categorical.

```
. graph box y1 y2, over(cat_var)
```

- y1, y2 must be numeric; statistics are shown on the y axis
- `cat_var` may be numeric or string; it is shown on categorical x axis

first group | second group
---|---
2 | 4
4 | 6
6 | 8

---

---
The encoding and the words used to describe the encoding are

- adjacent line <= outside values
- whiskers <= upper adjacent value
- box <= 75th percentile (upper hinge)
- whiskers <= median
- box <= 25th percentile (lower hinge)
- adjacent line <= lower adjacent value
- whiskers <= outside value

`graph hbox` draws horizontal box plots. In a horizontal box plot, the numerical axis is still called the `y` axis, and the categorical axis is still called the `x` axis, but `y` is presented horizontally, and `x` vertically.

```
. graph hbox y1 y2, over(cat_var)
```

The same conceptual layout as for `graph box`:
- `y1`, `y2` appear on `y` axis
- `cat_var` appears on `x` axis

**Options**

Options are presented under the following headings:

- `group_options`
- `yvar_options`
- `boxlook_options`
- `legending_options`
- `axis_options`
- `title_and_other_options`
- `Suboptions for use with over() and yvaroptions()`
group_options

over(varname[ , over_subopts]) specifies a categorical variable over which the yvars are to be repeated. varname may be string or numeric. Up to two over() options may be specified when multiple yvars are specified, and up to three over()s may be specified when one yvar is specified; see Examples of syntax under Remarks and examples below.

nofill specifies that missing subcategories be omitted. See the description of the nofill option in [G-2] graph bar.

missing specifies that missing values of the over() variables be kept as their own categories, one for .a, another for .a, etc. The default is to ignore such observations. An over() variable is considered to be missing if it is numeric and contains a missing value or if it is string and contains “

allcategories specifies that all categories in the entire dataset be retained for the over() variables. When if or in is specified without allcategories, the graph is drawn, completely excluding any categories for the over() variables that do not occur in the specified subsample. With the allcategories option, categories that do not occur in the subsample still appear in the legend, and zero-height bars are drawn where these categories would appear. Such behavior can be convenient when comparing graphs of subsamples that do not include completely common categories for all over() variables. This option has an effect only when if or in is specified or if there are missing values in the variables. allcategories may not be combined with by().

yvar_options

ascategory specifies that the yvars be treated as the first over() group. The important effect of this is to move the captioning of the variables from the legend to the categorical x axis. See the description of ascategory in [G-2] graph bar.

asyvars specifies that the first over() group be treated as yvars. The important effect of this is to move the captioning of the first over group from the categorical x axis to the legend. See the description of asyvars in [G-2] graph bar.

cw specifies casewise deletion. If cw is specified, observations for which any of the yvars are missing are ignored. The default is to calculate statistics for each box by using all the data possible.

boxlook_options

nooutsides specifies that the outside values not be plotted or used in setting the scale of the y axis.

box(#, barlook_options) specifies the look of the yvar boxes. box(1, ...) refers to the box associated with the first yvar, box(2, ...) refers to the box associated with the second, and so on.

You specify barlook_options. Those options are borrowed from graph bar for boxes. The most useful barlook_option is color(colorstyle), which sets the color of the box. For instance, you might specify box(1, color(green)) to make the box associated with the first yvar green. See [G-4] colorstyle for a list of color choices and see [G-3] barlook_options for information on the other barlook_options.

pcycle(#) specifies how many variables are to be plotted before the pstyle (see [G-4] pstyle) of the boxes for the next variable begins again at the pstyle of the first variable—p1box (with the boxes for the variable following that using p2box and so on). Put another way: # specifies how quickly the look of boxes is recycled when more than # variables are specified. The default for most schemes is pcycle(15).
intensity(#) and intensity(*#) specify the intensity of the color used to fill the inside of the box. intensity(#) specifies the intensity, and intensity(*#) specifies the intensity relative to the default.

By default, the box is filled with the color of its border, attenuated. Specify intensity(*#), # < 1, to attenuate it more and specify intensity(*#), # > 1, to amplify it.

Specify intensity(0) if you do not want the box filled at all. If you are using a scheme that draws the median line in the background color such as s2mono, also specify option medtype(line) to change the median line to be in the color of the outline of the box.

lintensity(#) and lintensity(*#) specify the intensity of the line used to outline the box. lintensity(#) specifies the intensity, and lintensity(*#) specifies the intensity relative to the default.

By default, the box is outlined at the same intensity at which it is filled or at an amplification of that, which depending on your chosen scheme; see [G-4] schemes intro. If you want the box outlined in the darkest possible way, specify intensity(255). If you wish simply to amplify the outline, specify intensity(*#), # > 1, and if you wish to attenuate the outline, specify intensity(*#), # < 1.

medtype(), medline(), and medmarker() specify how the median is to be indicated in the box.

medtype(line) is the default. A line is drawn across the box at the median. Here options medline() and medmarker() are irrelevant.

medtype(cline) specifies a custom line be drawn across the box at the median. The default custom line is usually a different color. You can, however, specify option medline(line_options) to control exactly how the line is to look; see [G-3] line_options.

medtype(marker) specifies a marker be placed in the box at the median. Here you may also specify option medmarker(marker_options) to specify the look of the marker; see [G-3] marker_options.

cwhiskers, lines(line_options), alsize(#), and capsize(#) specify the look of the whiskers.

cwhiskers specifies that custom whiskers are desired. The default custom whiskers are usually dimmer, but you may specify option lines(line_options) to specify how the custom whiskers are to look; see [G-3] line_options.

alsize(#) and capsize(#) specify the width of the adjacent line and the height of the cap on the adjacent line. You may specify these options whether or not you specify cwhiskers. alsize() and capsize() are specified in percentage-of-box-width units; the defaults are alsize(67) and capsize(0). Thus the adjacent lines extend two-thirds the width of a box and, by default, have no caps. Caps refer to whether the whiskers look like

```
this    or this
\-      [-]
```

If you want caps, try capsize(5).

marker(#, marker_options marker_label_options) specifies the marker and label to be used to display the outside values. See [G-3] marker_options and [G-3] marker_label_options.

outergap(*#) and outergap(#) specify the gap between the edge of the graph to the beginning of the first box and the end of the last box to the edge of the graph.
outergap(*#) specifies that the default be modified. Specifying outergap(*1.2) increases the gap by 20%, and specifying outergap(*.8) reduces the gap by 20%.

outergap(#) specifies the gap as a percentage-of-box-width units. outergap(50) specifies that the gap be half the box width.

boxgap(#) specifies the gap to be left between yvar boxes as a percentage-of-box-width units. The default is boxgap(33).

booxgap() affects only the yvar boxes. If you want to change the gap for the first, second, or third over() group, specify the over_subopt gap() inside the over() itself; see Suboptions for use with over() and yvaroptions() below.

legending_options

legend_options allows you to control the legend. If more than one yvar is specified, a legend is produced. Otherwise, no legend is needed because the over() groups are labeled on the categorical x axis. See [G-3] legend_options, and see Treatment of multiple yvars versus treatment of over() groups under Remarks and examples below.

nolabel specifies that, in automatically constructing the legend, the variable names of the yvars be used in preference to their labels.

yvaroptions(over_subopts) allows you to specify over_subopts for the yvars. This is seldom done.

showyvars specifies that, in addition to building a legend, the identities of the yvars be shown on the categorical x axis. If showyvars is specified, it is typical to also specify legend(off).

axis_options

yalternate and xalternate switch the side on which the axes appear.

Used with graph box, yalternater moves the numerical y axis from the left to the right; xalternate moves the categorical x axis from the bottom to the top.

Used with graph hbox, yalternater moves the numerical y axis from the bottom to the top; xalternate moves the categorical x axis from the left to the right.

If your scheme by default puts the axes on the opposite sides, then yalternater and xalternater reverse their actions.

yreverse specifies that the numerical y axis have its scale reversed so that it runs from maximum to minimum.

axis_scale_options specify how the numerical y axis is scaled and how it looks; see [G-3] axis_scale_options. There you will also see option xscale() in addition toyscale(). Ignore xscale(), which is irrelevant for box plots.

axis_label_options specify how the numerical y axis is to be labeled. The axis_label_options also allow you to add and suppress grid lines; see [G-3] axis_label_options. There you will see that, in addition to options ylabel(), ytick(), ..., ymtick(), options xlabel(), ..., xmtick() are allowed. Ignore the x*() options, which are irrelevant for box plots.

ytitle() overrides the default title for the numerical y axis; see [G-3] axis_title_options. There you will also find option xtitle() documented, which is irrelevant for box plots.
**title_and_other_options**

`text()` adds text to a specified location on the graph; see [G-3] *added_text_options*. The basic syntax of `text()` is

```
  text(##_y #_x "text")
```

`text()` is documented in terms of twoway graphs. When used with box plots, the “numeric” `x` axis is scaled to run from 0 to 100.

`yline()` adds horizontal (box) or vertical (hbox) lines at specified `y` values; see [G-3] *added_line_options*. The `xline()` option, also documented there, is irrelevant for box plots. If your interest is in adding grid lines, see [G-3] *axis_label_options*.

`aspect_option` allows you to control the relationship between the height and width of a graph’s plot region; see [G-3] *aspect_option*.

`std_options` allow you to add titles, control the graph size, save the graph on disk, and much more; see [G-3] *std_options*.

`by(varlist, ...)` draws separate plots within one graph; see [G-3] *by_option* and see *Use with by()* under Remarks and examples below.

**Suboptions for use with over() and yvaroptions()**

`total` specifies that, in addition to the unique values of `over(varname)`, a group be added reflecting all the observations. When multiple `over()`s are specified, `total` may be specified in only one of them.

`relabel(# "text" ...)` specifies text to override the default category labeling. See the description of the `relabel()` option in [G-2] *graph bar* for more information about this useful option.

`label(cat_axis_label_options)` determines other aspects of the look of the category labels on the `x` axis. Except for `label(labcolor())` and `label(labsize())`, these options are seldom specified; see [G-3] *cat_axis_label_options*.

`axis(cat_axis_line_options)` specifies how the axis line is rendered. This is a seldom specified option. See [G-3] *cat_axis_line_options*.

`gap(#) and gap(*#)` specify the gap between the boxes in this `over()` group. `gap(#)` is specified in percentage-of-box-width units, so `gap(67)` means two-thirds the width of a box. `gap(*#)` allows modifying the default gap. `gap(*1.2)` would increase the gap by 20% and `gap(*.8)` would decrease the gap by 20%.

To understand the distinction between `over(..., gap())` and option `boxgap()`, consider

```
  . graph box before after, boxgap(...) over(sex, gap(...))
```

`boxgap()` sets the distance between the before and after boxes. `over(..., gap())` sets the distance between the boxes for males and females. Similarly, in

```
  . graph box before after, boxgap(...) 
      over(sex, gap(...)) 
      over(agegrp, gap(...))
```

`over(sex, gap())` sets the gap between males and females, and `over(agegrp, gap())` sets the gap between age groups.
sort(varname) and sort(#) control how the boxes are ordered. See How boxes are ordered and Reordering the boxes under Remarks and examples below.

sort(varname) puts the boxes in the order of varname; see Putting the boxes in a prespecified order under Remarks and examples below.

sort(#) puts the boxes in order of their medians. # refers to the yvar number on which the ordering should be performed; see Putting the boxes in median order under Remarks and examples below.

descending specifies that the order of the boxes—default or as specified by sort()—be reversed.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Examples of syntax
Treatment of multiple yvars versus treatment of over() groups
How boxes are ordered
Reordering the boxes
Putting the boxes in a prespecified order
Putting the boxes in median order
Use with by()
Video example
History

Also see [G-2] graph bar. Most of what is said there applies equally well to box plots.

Introduction

graph box draws vertical box plots:

```
. use http://www.stata-press.com/data/r13/bplong
    (fictional blood pressure data)
. graph box bp, over(when) over(sex)
    ytitle("Systolic blood pressure")
    title("Response to Treatment, by Sex")
    subtitle("(120 Preoperative Patients)" "")
    note("Source: Fictional Drug Trial, StataCorp, 2003")
```

![Graph of response to treatment, by sex](image)
Examples of syntax

Below we show you some *graph box* commands and tell you what each would do:

```
graph box bp
One big box showing statistics on blood pressure.

graph box bp_before bp_after
Two boxes, one showing average blood pressure before, and the other, after.

graph box bp, over(agegrp)
    #_of_agegrp boxes showing blood pressure for each age group.

graph box bp_before bp_after, over(agegrp)
    2 × #_of_agegrp boxes showing blood pressure, before and after, for each age group. The grouping would look like this (assuming three age groups):
```

![Box plots example](image-url)
Treatment of multiple yvars versus treatment of over() groups

Consider two datasets containing the same data but organized differently. The datasets contain blood pressure before and after an intervention. In the first dataset, the data are organized the wide way; each patient is an observation. A few of the data are

<table>
<thead>
<tr>
<th>patient</th>
<th>sex</th>
<th>agegrp</th>
<th>bp_before</th>
<th>bp_after</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Male</td>
<td>30-45</td>
<td>143</td>
<td>153</td>
</tr>
<tr>
<td>2</td>
<td>Male</td>
<td>30-45</td>
<td>163</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>Male</td>
<td>30-45</td>
<td>153</td>
<td>168</td>
</tr>
</tbody>
</table>
In the second dataset, the data are organized the long way; each patient is a pair of observations. The corresponding observations in the second dataset are

<table>
<thead>
<tr>
<th>patient</th>
<th>sex</th>
<th>agegrp</th>
<th>when</th>
<th>bp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Male</td>
<td>30-45</td>
<td>Before</td>
<td>143</td>
</tr>
<tr>
<td>1</td>
<td>Male</td>
<td>30-45</td>
<td>After</td>
<td>153</td>
</tr>
<tr>
<td>2</td>
<td>Male</td>
<td>30-45</td>
<td>Before</td>
<td>163</td>
</tr>
<tr>
<td>2</td>
<td>Male</td>
<td>30-45</td>
<td>After</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>Male</td>
<td>30-45</td>
<td>Before</td>
<td>153</td>
</tr>
<tr>
<td>3</td>
<td>Male</td>
<td>30-45</td>
<td>After</td>
<td>168</td>
</tr>
</tbody>
</table>

Using the first dataset, we might type

```
. use http://www.stata-press.com/data/r13/bpwide, clear
    (fictional blood press data)
. graph box bp_before bp_after, over(sex)
```
Using the second dataset, we could type

```
. use http://www.stata-press.com/data/r13/bplong, clear
. graph box bp, over(when) over(sex)
```

The two graphs are virtually identical. They differ in that

<table>
<thead>
<tr>
<th>multiple <code>yvars</code></th>
<th>over( ) groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>boxes different colors</td>
<td>yes</td>
</tr>
<tr>
<td>boxes identified via . . .</td>
<td>legend</td>
</tr>
</tbody>
</table>

Option `ascategory` will cause multiple `yvars` to be presented as if they were the first `over()` group, and option `asyvars` will cause the first `over()` group to be presented as if they were multiple `yvars`. Thus

```
. graph box bp, over(when) over(sex) asyvars
```

would produce the first chart and

```
. graph box bp_before bp_after, over(sex) ascategory
```

would produce the second.

**How boxes are ordered**

The default is to place the boxes in the order of the `yvars` and to order each `over(varname)` group according to the values of `varname`. Let us consider some examples:

```
graph box bp_before bp_after
Boxes appear in the order specified, `bp_before` and `bp_after`.
```

```
graph box bp, over(when)
Boxes are ordered according to the values of variable `when`.
```

If variable `when` is a numeric, the lowest `when` number comes first, followed by the next lowest, and so on. This is true even if variable `when` has a value label. Say that `when = 1` has been labeled “Before” and `when = 2`, labeled “After”. The boxes will be in the order Before followed by After.
If variable `when` is a string, the boxes will be ordered by the sort order of the values of the variable (that is, alphabetically, but with capital letters placed before lowercase letters). If variable `when` contains “Before” and “After”, the boxes will be in the order After followed by Before.

```plaintext
graph box bp_before bp_after, over(sex)
Boxes appear in the order specified, bp_before and bp_after, and are repeated for each sex, which will be ordered as explained above.

graph box bp_before bp_after, over(sex) over(agegrp)
Boxes appear in the order specified, bp_before and bp_after, repeated for sex ordered on the values of variable sex, repeated for agegrp ordered on the values of variable agegrp.
```

### Reordering the boxes

There are two ways you may wish to reorder the boxes:

1. You want to control the order in which the elements of each `over()` group appear. String variable `when` might contain “After” and “Before”, but you want the boxes to appear in the order Before and After.

2. You wish to order the boxes according to their median values. You wish to draw the graph

```plaintext
  . graph box wage, over(industry)
```

and you want the industries ordered by `wage`.

We will consider each of these desires separately.

### Putting the boxes in a prespecified order

You have drawn the graph

```plaintext
  . graph box bp, over(when) over(sex)
```

Variable `when` is a string containing “Before” and “After”. You wish the boxes to be in that order. To do that, you create a new numeric variable that orders the group as you would like:

```plaintext
  . generate order = 1 if when=="Before"
  . replace order = 2 if when=="After"
```

You may name the variable and create it however you wish, but be sure that there is a one-to-one correspondence between the new variable and the `over()` group’s values. You then specify `over()`’s `sort(varname)` option:

```plaintext
  . graph box bp, over(when, sort(order)) over(sex)
```

If you want to reverse the order, you may specify the `descending` suboption:

```plaintext
  . graph box bp, over(when, sort(order) descending) over(sex)
```

### Putting the boxes in median order

You have drawn the graph

```plaintext
  . graph hbox wage, over(industry)
```

and now wish to put the boxes in median order, lowest first. You type

```plaintext
  . graph hbox wage, over( industry, sort(1) )
```
If you wanted the largest first, you would type

```
. graph hbox wage, over(industry, sort(1) descending)
```

The 1 in `sort(1)` refers to the first (and here only) `yvar`. If you had multiple `yvars`, you might type

```
. graph hbox wage benefits, over(industry, sort(1))
```

and you would have a chart showing `wage` and `benefits` sorted on `wage`. If you typed

```
. graph hbox wage benefits, over(industry, sort(2))
```

the graph would be sorted on `benefits`.

**Use with `by()`**

`graph box` and `graph hbox` may be used with `by()`, but in general, you will want to use `over()` in preference to `by()`. Box charts are explicitly categorical and do an excellent job of presenting summary statistics for multiple groups in one chart.

A good use of `by()`, however, is when the graph would otherwise be long. Consider the graph

```
. use http://www.stata-press.com/data/r13/nlsw88, clear
   (NLSW, 1988 extract)
. graph hbox wage, over(ind) over(union)
```

In the above graph, there are 12 industry categories and two union categories, resulting in 24 separate boxes. The graph, presented at normal size, would be virtually unreadable. One way around that problem would be to make the graph longer than usual,

```
. graph hbox wage, over(ind) over(union) ysize(7)
```

See *Charts with many categories* in [G-2] `graph bar` for more information about that solution. The other solution would be to introduce union as a `by()` category rather than an `over()` category:

```
. graph hbox wage, over(ind) by(union)
```
Below we do precisely that, adding some extra options to produce a good-looking chart:

```
.graph hbox wage, over(ind, sort(1)) nooutside
    ytitle(""")
    by(
        union,
        title("Hourly wage, 1988, woman aged 34–46", span)
        subtitle(" ")
    )
```

The title options were specified inside the `by()` so that they would not be applied to each graph separately; see [G-3] _by_option_.

**Video example**

Box plots in Stata

**History**

Box plots have been used in geography and climatology, under the name “dispersion diagrams”, since at least 1933; see Crowe (1933). His figure 1 shows all the data points, medians, quartiles, and octiles by month for monthly rainfalls for Glasgow, 1868–1917. His figure 2, a map of Europe with several climatic stations, shows monthly medians, quartiles, and octiles.

**Methods and formulas**

For a description of box plots, see Cleveland (1993, 25–27).

Summary statistics are obtained from `summarize`; see [R] _summarize_.

The upper and lower adjacent values are as defined by Tukey (1977):

Let $x$ represent a variable for which adjacent values are being calculated. Define $x_{(i)}$ as the $i$th ordered value of $x$, and define $x_{[25]}$ and $x_{[75]}$ as the 25th and 75th percentiles.
Define $U$ as $x_{[75]} + \frac{3}{2}(x_{[75]} - x_{[25]})$. The upper adjacent value is defined as $x_i$, such that $x_{(i)} \leq U$ and $x_{(i+1)} > U$.

Define $L$ as $x_{[25]} - \frac{3}{2}(x_{[75]} - x_{[25]})$. The lower adjacent value is defined as $x_i$, such that $x_{(i)} \geq L$ and $x_{(i-1)} < L$.

References


Also see

[G-2] graph bar — Bar charts

[R] lv — Letter-value displays

[R] summarize — Summary statistics