University of Bern Social Sciences Working Paper No. 30

Customizing Stata graphs made even easier

Ben Jann

April 12, 2018

http://ideas.repec.org/p/bss/wpaper/30.html
http://econpapers.repec.org/paper/bsswpaper/30.htm

Customizing Stata graphs made even easier

Ben Jann
University of Bern

Abstract.   In Jann (2017) I presented a command called `grstyle` that simplifies changing the default look of Stata graphs. The command, however, still relies on idiosyncratic scheme file syntax, which may not be well known to many users. In this paper I therefore present an extension called `grstyle set` that automizes the creation of sets of scheme file entries for a number of potentially useful adjustments, without much typing and without requiring much knowledge about scheme file syntax. Covered topics include, for example, the rendering of the background and coordinate system, the placement and look of the legend, the assignment of colors, symbols, and line patterns, and the assignment of relative or absolute sizes.

Keywords: st0001, `grstyle`, `grstyle set`, `graph`, graphics, scheme files

1   Introduction

Stata graphics are flexible, but customizing their appearance can be tedious. In consequence, many people stick to the default `s2color` scheme, which may not always be the best choice. The difficulty with customizing Stata graphics is that the settings determining the default look of graphs are kept in external files—so called scheme files—and that these scheme files use their own idiosyncratic language. Users who want to change the look of their graphs without adding a long list of options to each graph command have to create their own scheme file and store it in an appropriate place in the file system. In Jann (2017) I argued that style settings should be part of the script creating the graph rather than being kept in separate scheme files and I provided a command called `grstyle` that supports such practice.

However, `grstyle` still relies on scheme file syntax, which is different from other Stata syntax and is only rudimentarily documented (see `help scheme entries`). To make things easier I now present an extension called `grstyle set` that can be used to generate custom styles without knowing much about scheme file syntax.

To install the new software and view the help file, type:

```
. scc install grstyle, replace
. help grstyle set
```

2   Basic usage and syntax

First use `grstyle init` to initialize the custom style settings (see Jann 2017). After that, apply one or several `grstyle set` commands, and possibly some other `grstyle` commands, to record the desired settings. In case of conflict, later settings take prece-

dence over earlier settings. The syntax of `grstyle set` is

grstyle set *subcommand* [*arguments*] [, *options*] [: *elements*]

where *arguments*, *options*, and *elements* depend on *subcommand*. A full list of subcommands and their syntax is provided in the online help (see `help grstyle set`). Below, I focus on a selection of subcommands and options that I find most useful.

3   Subcommands and examples

3.1   Rendering of background and coordinate system

`grstyle set plain` applies a plain overall look to the graph: white background, no shading of by-labels or matrix labels, black titles, and grid lines in gray. The syntax is

grstyle set plain [, horizontal compact [no]grid minor dotted noextend
  box]

where

    `horizontal` sets the orientation of tick labels on the Y-axis to horizontal,

    `compact` reduces the symbols size, some font sizes, and the graph margin,

    `grid` turns all (major) grid lines on (horizontal, vertical, minimum, maximum),

    `nogrid` turns grid lines off,

    `minor` adds a minor grid (one minor grid line between each pair of major grid lines; `minor` only has an effect if `grid` is specified),
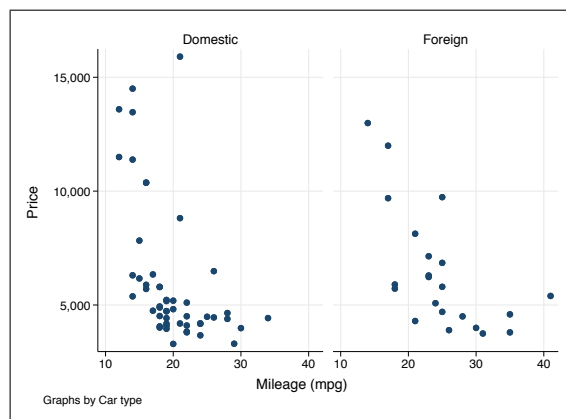
    `dotted` requests dotted grid lines,

    `noextend` draws axis lines only to the smallest and largest tick marks, and

    `box` draws a frame around the plot region. The default is to omit the frame.

An example is as follows:

```
. set scheme s2color
. grstyle init
. grstyle set plain, horizontal grid
. sysuse auto, clear
(1978 Automobile Data)
. scatter price mpg, by(foreign)
```

By default, Stata only prints horizontal grid lines and sometimes omits the lines at the minimum and the maximum. Option `grid` was specified to request that all grid lines be printed. Furthermore, Stata's default is to print the labels on the Y-axis vertically. Option `horizontal` changes the orientation of these labels to horizontal.

Furthermore, the following example illustrates the use of option `nogrid` to turn grid lines off and option `noextend` to request that axis lines only be drawn to the smallest and largest tick:

```
. grstyle init
. grstyle set plain, nogrid noextend
. scatter price mpg, by(foreign)
```
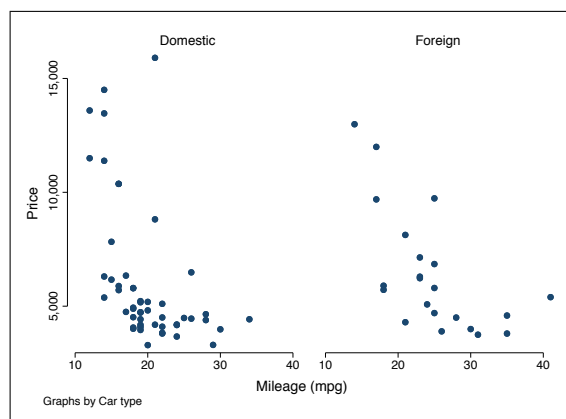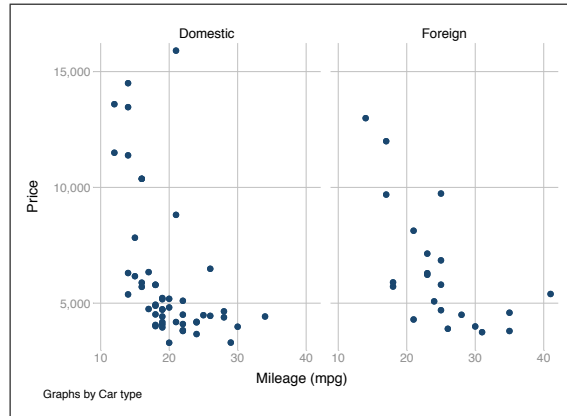
A second command is `grstyle set mesh`. It also applies a plain background, but renders the coordinate plane as a mesh (or inverted mesh) of gridlines. Its syntax is

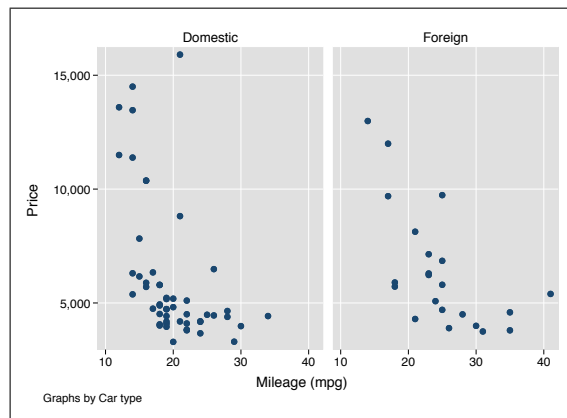grstyle set [i]mesh [, <u>horizontal</u> <u>compact</u> <u>minor</u>]

where `horizontal`, `compact`, and `minor` are as above. An example is as follows:

```
. grstyle init
. grstyle set mesh, horizontal
. scatter price mpg, by(foreign)
```

Alternatively, to use an inverted mesh, type `imesh` instead of `mesh`:

```
. grstyle init
. grstyle set imesh, horizontal
. scatter price mpg, by(foreign)
```

3.2   Placement and rendering of the legend

`grstyle set legend` determines the position and some style aspects of the legend. The syntax is

`grstyle set legend` [*clockpos*] [`, nobox stacked inside klength(`*size*`)`]

where *clockpos* sets the position of the legend according to [G] ***clockposstyle*** and

>   `nobox` removes the box around the legend (including background color),
>
>   `stacked` stacks keys and labels (and makes some adjustments to the gaps between keys),
>
>   `inside` places the legend inside the plot region, and

klength(*size*) determines the length of legend keys, where *size* is a size specification according to [G] ***textsizestyle***. By default, *size* is set to huge. This is considerably shorter than the factory setting, which is equivalent to klength(13).

The default *clockpos* is 6, which places the legend at the bottom. For example, type grstyle set legend 4 to place the legend at the lower right side of the plot. If the legend is placed at the side (positions 2, 3, 4, 7, 8, and 9), the number of legend columns will be restricted to 1. Here is an example in which the legend is placed inside the plot region (option inside) at the top right (position 2):

```
. grstyle init
. grstyle set plain, horizontal grid
. grstyle set legend 2, inside
. sysuse auto, clear
(1978 Automobile Data)
. separate price, by(foreign)
>     shortlabel
  (output omitted)
. scatter price0 price1 mpg
```
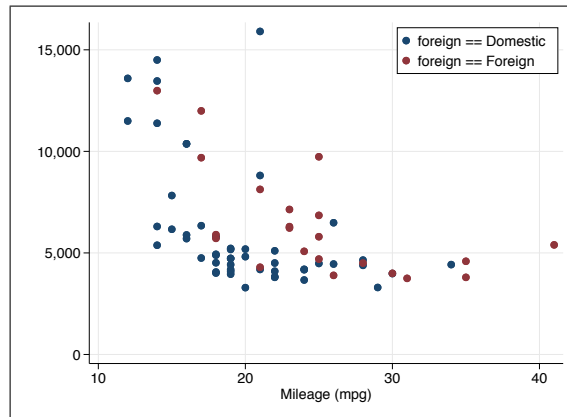
In the following example the legend is placed at the lower right side of the plot region (position 4) and the frame around the legend is removed (option nobox):

```
. grstyle init
. grstyle set imesh, horizontal
>     compact minor
. grstyle set legend 4, nobox
. scatter price0 price1 mpg
```
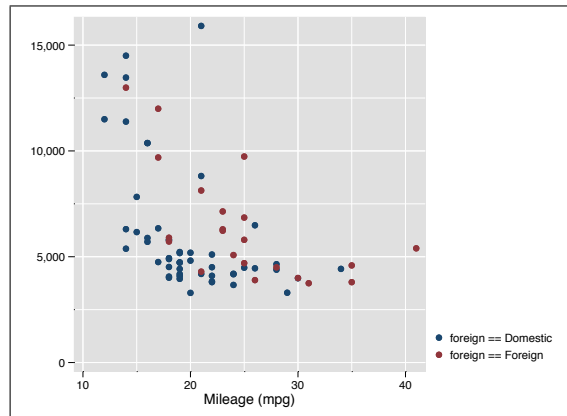
3.3   Assigning colors

Colors can be assigned using command grstyle set color. Its syntax is

grstyle set <u>c</u>olor $\big[$*argument*$\big]$ $\big[$, <u>p</u>lots(*numlist*) *palette_opts* $\big]$ $\big[$: *elements* $\big]$

where *argument* is

   palette $\big[$ $\big[$, *palette_options*$\big]$ / $\big[$ *palette* $\big[$, *palette_options*$\big]$ / ... $\big]$ $\big]$

palette is a color palette or list of colors as described below and

   plots(*numlist*) specifies the plot numbers for numbered elements, and

   palette_opts are palette options such as n() to set the palette size, select() to select
   elements from a palette, reverse to use the palette in reverse order, intensity()
   to apply color intensity adjustment, or opacity() to set opacity (which requires
   Stata 15). See Jann (2018) for details on these options.

elements is a space-separated list of scheme entry elements to be set (see help scheme
entries). Examples are background for the background color, plotregion for the color
of the plot region, or xyline for the color of added lines. See the entry on grstyle set
color in help grstyle set for a more extensive list of possible elements. If *elements*
is omitted, the overall default colors for plot objects such as markers, lines, bars, etc.
are set (elements p1, p2, etc.).

❏ **Technical note**

   Numbered elements, that is, elements that apply to specific plots, can be specified as
p#$\big[$*stub*$\big]$. Examples are p# for the overall default colors, p#lineplot for the colors used
in line plots, or p#label for the colors of marker labels. Such syntax will be expanded
to a set of numbered elements (e.g., p1, p2, etc.) using the numbers provided by the
plots() option or, if plots() is omitted, using numbers 1 to n, where n is the number
of provided colors. The values will be assigned one by one to the expanded elements
(recycling the list of colors if necessary; and starting over for each set if multiple sets
are specified). Naturally, numbered elements can also be specified directly as individual
elements, e.g., p1label, p2label, etc. Furthermore, note that numbered elements
always also have an unnumbered form, p$\big[$*stub*$\big]$, that can be used to set the default for
all plots. For example, to use the same color for marker labels in all plots you can simply
set plabel instead of p1label, p2label, etc. However, numbered elements will take
precedence over unnumbered elements. That is, if, say, p3label has been set, setting
plabel will have no effect on marker labels in the 3rd plot.

                                                                                      ❏

**Using a predefined color palette or a color generator**

If *argument* is omitted, grstyle set color uses the same colors as are assigned to
elements p1 to p15 in Stata's s2color scheme ([G] **scheme s2**). Alternatively, specify
one of the color palettes or color generators provided by the colorpalette command
(see Jann 2018).

   Here is an example using the economist palette ([G] **scheme economist**), the

rainbow palette from the ptol collection (Tol 2012), or the Greens palette from the ColorBrewer collection (Brewer et al. 2003; Brewer 2016), respectively:
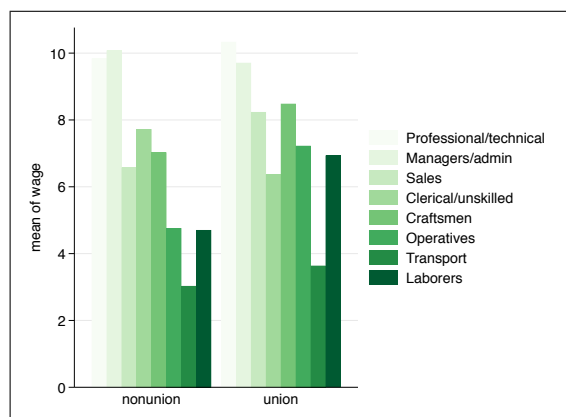
```
. grstyle init
. grstyle set plain, horizontal
. grstyle set legend 3, nobox
. grstyle set color economist
. sysuse nlsw88, clear
(NLSW, 1988 extract)
. graph bar wage if occ<9, over(occ)
>     asyvars over(union)
```
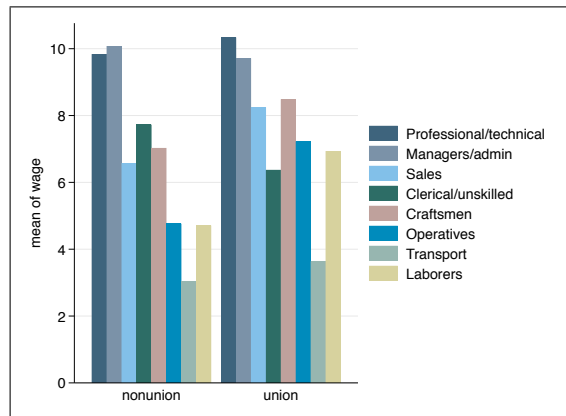


```
. grstyle set color ptol, rainbow
>     n(8) reverse
. graph bar wage if occ<9, over(occ)
>     asyvars over(union)
```
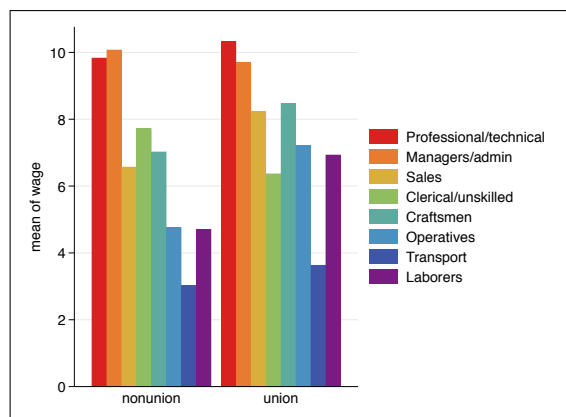


```
. grstyle set color Greens, n(8)
. graph bar wage if occ<9, over(occ)
>     asyvars over(union)
```

Option `n(8)` has been specified in the 2nd and 3rd variant because 8 color are
required for the graph. Like many of the palettes provided by `colorpalette`, the `ptol`
and `Greens` palettes are adaptive in the sense that the returned colors depend on how
many colors are requested. The same is true, for instance, for the HCL (Hue-Chroma-
Luminance) color generator that returns colors with evenly spaced hues around the color
wheel (the `hue` palette), as illustrated in the following example:

```
. grstyle init
. grstyle set imesh, horizontal
>     compact minor
. grstyle set legend 4, nobox stack
. grstyle set color hue, n(4)
. sysuse uslifeexp, clear
(U.S. life expectancy, 1900-1999)
. lab var le_wfemale "white females"
. lab var le_wmale   "white males"
. lab var le_bfemale "black females"
. lab var le_bmale   "black males"
. line le_wfemale le_wmale le_bfemale
>     le_bmale year, lwidth(*1.5 ..)
>     ytitle(Life expectancy)
```
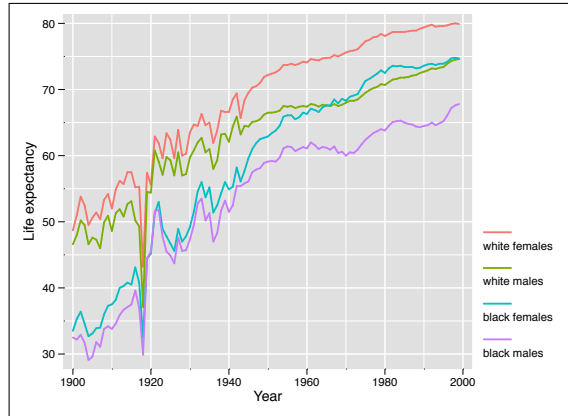

Specifying custom colors

Instead of using a named palette you can provide custom colors by specifying a list of
colorstyles (named colors, RBG codes, CMYK codes, or HSV codes; see [G] ***colorstyle***).
In addition, `grstyle set` also supports colors specified as hex triplets (such as `#87ceeb`
for sky blue) or as HCL codes (such as `"hcl 135 100 65"` for green); see Jann (2018) for
details. Here is an example using hex codes of semantic colors of buttons in Bootstrap
v3.3 (getbootstrap.com/docs/3.3):

```
. grstyle init
. grstyle set plain
. grstyle set color #286090 #449d44
>     #31b0d5 #ec971f #c9302c
. drop _all
. input primary success info warning
>     danger

      primary    success      info
> warning      danger
  1. 12 14 2 8 5
  2. end
. graph pie primary success info
>     warning danger, legend(off)
>     plabel(_all name, size(*1.5)
>         color(white))
```
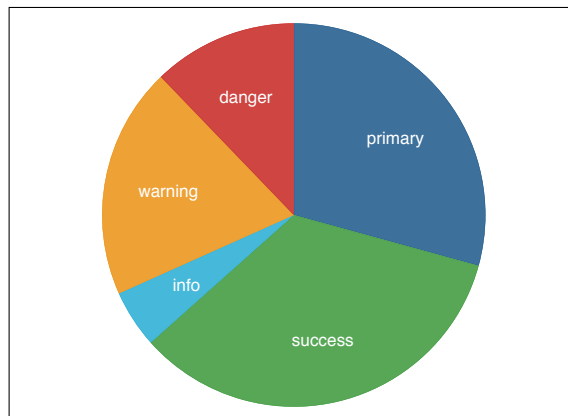
Making markers transparent (requires Stata 15)

Making marker symbols transparent is a bit tricky. If you just make colors transparent, the marker outline will look different than the marker fill because the outline is printed on top of the fill. A solution is to make the outline fully transparent. To set the fill colors of markers, use scheme entry elements `p1markfill`, `p2markfill`, etc.; for the outlines use `p1markline`, `p2markline`, etc. Here is an example using the `Set1` palette from the ColorBrewer collection (Brewer et al. 2003; Brewer 2016):

```
. grstyle init
. grstyle set imesh
. grstyle set legend 4, nobox
. grstyle set color Set1, opacity(50)
>     : p#markfill
. grstyle set color Set1, opacity(0)
>     : p#markline
. grstyle symbolsize p huge
. sysuse auto, clear
(1978 Automobile Data)
. separate price, by(foreign)
>     shortlabel
  (output omitted )
. scatter price0 price1 length
```
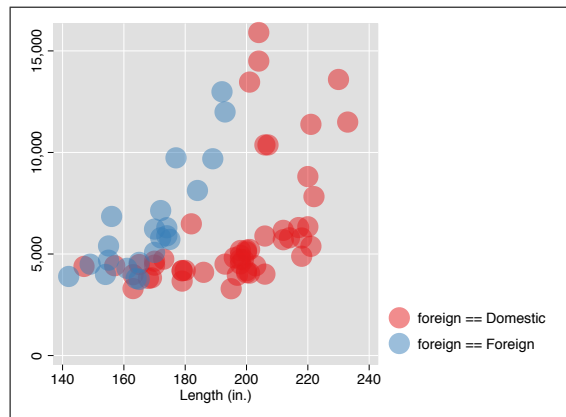

The `opacity()` option was used to determine the degree of transparency. An opacity value of 100 means fully opaque (no transparency), a value of 0 means fully transparent (invisible).

Changing the color intensity

Color intensity can be set using the `intensity()` option when calling `grstyle set color`. For some elements, however, color intensity can also be set as a separate scheme entry using `grstyle set intensity`. The syntax is

grstyle set <u>intensity</u> $\big[$ *intlist* $\big]$ $\big[$, <u>p</u>lots(*numlist*) $\big]$ $\big[$: *elements* $\big]$

where *intlist* is a space-separated list of intensity specifications according to [G] ***intensitystyle*** (the default is 100, i.e., full intensity), option `plots()` specifies the plot numbers for numbered elements, and *elements* is a space-separated list of scheme entry elements to be set. Examples elements are `pie` for the fill intensity of pie slices or `histogram` for the fill intensity of histogram bars. See the entry on `grstyle set intensity` in `help grstyle set` for a more extensive list of possible elements. If *elements* is omitted, the intensity of bars and areas in two-way graphs is set (element `p`, or, if `plots()` is specified or multiple intensities are provided, elements `p1`, `p2`, etc.).

The following example uses the `Accent` palette from the ColorBrewer collection

(Brewer et al. 2003; Brewer 2016), but then sets the fill intensity of bars for `graph bar`
([G] **graph bar**) to 30 percent:

```
. grstyle init
. grstyle set plain, horizontal
. grstyle set color Accent: p#bar
>     p#barline
. grstyle set intensity 30: bar
. sysuse citytemp, clear
(City Temperature Data)
. graph bar (mean) tempjuly tempjan,
>     over(region) bargap(-30)
```
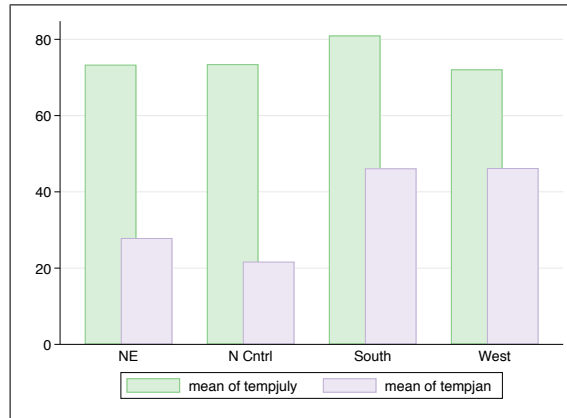

3.4   Transparent confidence intervals (requires Stata 15)

Confidence interval can be made transparent using `grstyle set color` with the `opacity()`
option. Alternatively, for convenience, use `grstyle set ci`. The syntax is

grstyle set ci $\big[$*colors*$\big]$ $\big[$, <u>op</u>acity(*numlist*) *palette_opts* $\big]$

where *colors* is a list of colors or a color palette as described in Section 3.3, option
`opacity()` sets the level of opacity (0: fully transparent; 100: fully opaque), and
palette_opts are palette options as described in Section 3.3. Specify one or two colors,
depending on whether you want only the primary, or the primary and the secondary
confidence intervals to be affected. If no colors are specified, `gs12` is used for primary
confidence intervals and `ltkhaki` for secondary confidence intervals (same colors as in
the `s2color` scheme). The default opacity is 50. Specify two opacity values if you want
to use different transparency levels for primary and secondary confidence intervals. An
example is as follows:

```
. grstyle init
. grstyle set plain, horizontal grid
. grstyle set color Dark2
. grstyle set legend 10, inside nobox
. grstyle set ci
. sysuse auto, clear
(1978 Automobile Data)
. twoway
> (scatter price length if foreign==0)
> (scatter price length if foreign==1)
> (lpolyci price length if foreign==0,
>     clstyle(p1))
> (lpolyci price length if foreign==1,
>     clstyle(p2))
> , legend(order(2 "Foreign"
>               1 "Domestic"))
```
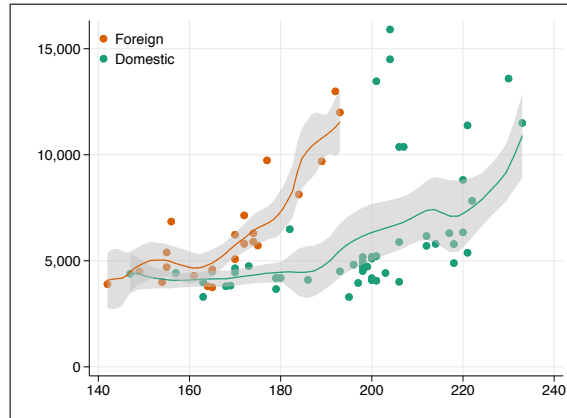
3.5 Assigning marker symbols and line patterns

The syntax to assign marker symbols and line patterns is

grstyle set symbol [*palette*] [, plots(*numlist*) *palette_opts*] [: *elements*]

grstyle set lpattern [*palette*] [, plots(*numlist*) *palette_opts*] [: *elements*]

where *palette* is a symbol palette or a line pattern palette provided by commands
symbolpalette and linepalette (Jann 2018), or list of symbols or line patterns as
documented in [G] ***symbolstyle*** and [G] ***linepatternstyle***. If *palette* is omitted, the
same sequence of symbols and line patterns as in Stata's monochrome schemes is used.
Options are as follows:

   plots(*numlist*) specifies the plot numbers for numbered elements.

    palette_opts are palette options such as n() to set the palette size, select() to select
        elements from a palette, or reverse to use the palette in reverse order. See Jann
        (2018) for details.

elements is a space-separated list of scheme entry elements to be set (see help scheme
entries). Examples are matrix to set the marker symbol used in matrix graphs,
plineplot to set line pattern for line plots, pline to set line pattern for connected-line
plots, or xyline to set the line pattern for added lines. See the sections on grstyle set
symbol and grstyle set lpattern in help grstyle set for more extensive lists of
possible elements. If *elements* is omitted, the overall default symbols and line patterns
are set (element p, or, if plots() is specified or multiple symbols or line patterns are
provided, elements p1, p2, etc.).

    Stata's default color schemes use filled circles for marker symbols and solid lines
for all plots, which makes the graphs hard to read on black and white printouts. To
improve readability, you could change the symbols and line patterns as in the following

example:

```
. grstyle init
. grstyle set plain
. grstyle set color Set1
. grstyle set symbol
. grstyle set lpattern
. sysuse auto, clear
(1978 Automobile Data)
. separate price, by(foreign)
>     shortlabel
  (output omitted)
. two connect price0 price1 weight,
>     ytitle(Price) sort
```
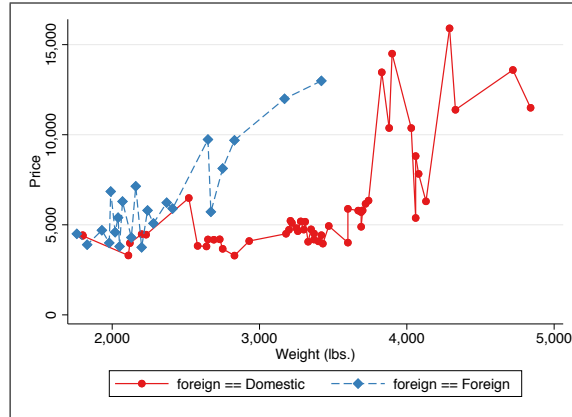
If you are not happy with the default sequence of symbols and line patterns, use one of the alternative palettes provided by `symbolpalette` and `linepalette` (Jann 2018) or specify an own sequence as in the following example:

```
. grstyle set symbol T S
. grstyle set lpattern "-." "--.."
. grstyle set color white, p(1 2)
>     : p#markfill
. two connect price0 price1 weight,
>     ytitle(Price) sort
```
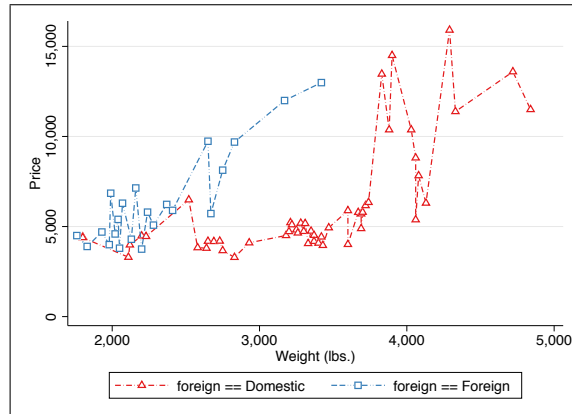
3.6   Assigning sizes

A final group of subcommands is concerned with setting the overall size of graphs, the size of text and markers, the thickness of lines, and the width of margins. In Stata, sizes of graph objects are determined relative to the overall size of the graph (the minimum of the height and the width of the graph, to be precise). This makes it difficult to control the size of objects in absolute terms (e.g., to ensure that a certain text object uses, say, a 10pt font). As a key feature, `grstyle set` provides an easy way to assign absolute sizes.

Overall graph size

The syntax to set the overall graph size is

`grstyle set graphsize` $[$ *height* $[$ *width* $]]$

where the units for *height* and *width* can be <u>inch</u> (inches), `pt` (points; 1 inch = 72 points), `cm` (centimeters; 1 inch = 2.54 centimeters), or `mm` (millimeters; 1 inch = 25.4 millimeters). `inch` is assumed if the units are omitted. The default height is 4 inches, the default width is 5.5 inches.

For example, to set the size of graphs to 10 by 12 centimeters, you could type:

```
. grstyle set graphsize 10cm 12cm
```

❏ **Technical note**

Apart from setting the graph size, `grstyle set graphsize` also creates a global macro called GRSTYLE_RSIZE. The macro contains information that will be used by `grstyle set` when assigning absolute sizes (see below).

❏

Relative and absolute sizes of objects

The commands to set the size of text objects, the size of markers symbols, the thickness of lines, and the width of margins are

`grstyle set size` $[$ *sizelist* $]$ $[$, *units* <u>p</u>`lots`(*numlist*) $]$ $[$: *elements* $]$

`grstyle set symbolsize` $[$ *sizelist* $]$ $[$, *units* <u>p</u>`lots`(*numlist*) $]$ $[$: *elements* $]$

`grstyle set linewidth` $[$ *sizelist* $]$ $[$, *units* <u>p</u>`lots`(*numlist*) $]$ $[$: *elements* $]$

`grstyle set margin` $[$ *sizelist* $]$ $[$, *units* <u>p</u>`lots`(*numlist*) $]$ $[$: *elements* $]$

where *sizelist* is a space-separated list of absolute (in `pt`, `mm`, `inch`, or `cm`) or relative sizes (according to [G] ***textsizestyle***, [G] ***markersizestyle***, [G] ***linewidthstyle***, and [G] ***marginstyle***, respectively; excluding multiplication syntax and excluding *margin-exp*). Options are as follows:

units and can be `pt`, `mm`, <u>inch</u>, or `cm` and determines how numbers without units are interpreted. If *units* is omitted, numbers without units will be interpreted as relative sizes.

`plots`(*numlist*) specifies the plot numbers for numbered elements.

elements is a space-separated list of scheme entry elements to be set. See `help grstyle set` for details on possible elements.

The procedure to assign absolute sizes is to first set the overall graph size using `grstyle set graphsize`, which defines the reference value used by `grstyle set` for the computation of absolute sizes. After that, apply `grstyle set size` etc. with size specifications in absolute units. If `grstyle set graphsize` has not been run, `grstyle set` will determine absolute sizes with respect to a 4 inch graph size.

An example is as follows:

```
. grstyle init
. grstyle set plain, grid
. grstyle set legend 2
. grstyle set symbol Oh
. grstyle set graphsize 9cm 6cm
. grstyle set size 10pt: heading
. grstyle set size 8pt: subheading axis_title
. grstyle set size 6pt: tick_label key_label
. grstyle set symbolsize 1 2 3 4 5, pt
. grstyle set linewidth .8pt: plineplot
. grstyle set linewidth .4pt: pmark legend
>     axisline tick major_grid
. grstyle set margin zero
. quietly sysuse auto, clear
. quietly separate price, by(rep) shortlabel
. twoway (scatter price? weight)
>     (lfit price weight),
>     title("Title (10pt)")
>     subtitle("Subtitle (8pt)")
>     xtitle("X axis title (8pt)")
>     ytitle("Y axis title (8pt)")
>     legend(order(1 "1pt" 2 "2pt" 3 "3pt"
>         4 "4pt" 5 "5pt" 6 ".8pt"))
```
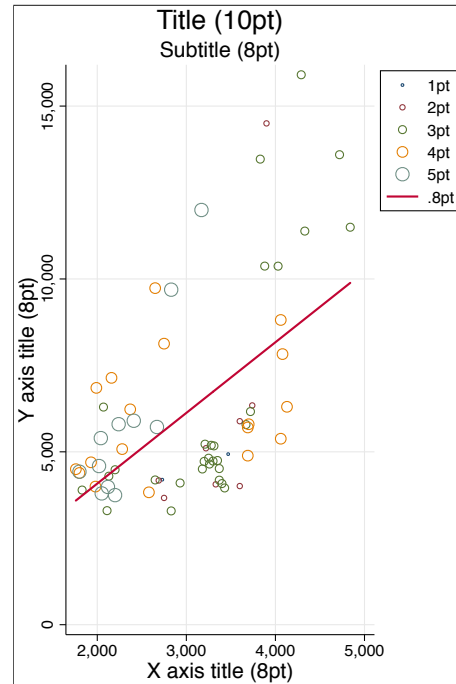

The text size of the title is set to 10 points (`heading`), subtitle (`subheading`) and axis titles (`axis_title`) are set to 8 points, tick labels (`tick_label`) and legend key labels (`key_label`) are set to 6 points. Marker symbols are set to sizes from 1 to 5 points for the 1st to the 5th plot. The line thickness for line plots (`plineplot`) is set to 0.8 points, the line thickness of marker symbol outlines (`pmark`), the legend outline (`legend`), the axis lines (`axisline`), the ticks (`tick`), and the grid lines (`major_grid`) are set to 0.4 points (the same line thickness is used by LaTeX for the frame around the graph). Finally, the margin of the graph region is set to zero.

❏ **Technical note**

Sizes in Stata graphs are always relative to the physical graph size. This implies that the requested absolute sizes will only be achieved as long as you do not change the graph size.

❏

3.7 Providing custom subcommands

An own collection of graph settings can be provided by defining a program called grstyle_set_*myname*. *myname* will then be available to grstyle set like any other subcommand.

When calling your program, grstyle set provides global macro GRSTYLE_FH containing the file handle of the temporary scheme file to be written to. Use file write write with this handle to add your scheme entries. For example, the following program would set the default approximate number of labeled ticks:

```
program grstyle_set_nticks
    syntax [, n(int 5) ]
    file write $GRSTYLE_FH "numticks_g horizontal_major `n'" _n
    file write $GRSTYLE_FH "numticks_g vertical_major   `n'" _n
end
```
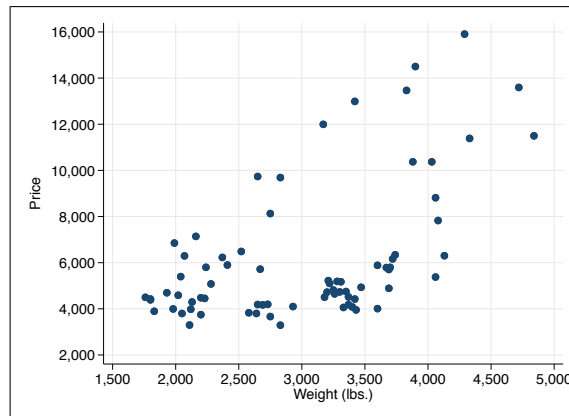
After defining the program (or storing the program in file grstyle_set_nticks.ado), it can be used as follows:

```
. grstyle init
. grstyle set plain, horizontal grid
. grstyle set nticks, n(10)
. sysuse auto, clear
(1978 Automobile Data)
. scatter price weight
```


4   References

Brewer, C. A. 2016. *Designing Better Maps. A Guide for GIS Users.* 2nd ed. Redlands, CA: Esri Press.

Brewer, C. A., G. W. Hatchard, and M. A. Harrower. 2003. ColorBrewer in Print: A Catalog of Color Schemes for Maps. *Cartography and Geographic Information Science* 30(1): 5–32.

Jann, B. 2017. Customizing Stata graphs made easy. University of Bern Social Sciences Working Papers 29, available from http://ideas.repec.org/p/bss/wpaper/29.html.

———. 2018. Color palettes for Stata graphics. University of Bern Social Sciences Working Papers 31, available from http://ideas.repec.org/p/bss/wpaper/31.html.

Tol, P. 2012. Colour Schemes. SRON Technical Note, Doc. no. SRON/EPS/TN/09-002.
    Available from https://personal.sron.nl/~pault/colourschemes.pdf.

About the author

Ben Jann is Professor of Sociology at the University of Bern, Switzerland. His research interests include social-science methodology, statistics, social stratification, and labor market sociology. He is principle investigator of TREE, a large-scale multi-cohort panel study in Switzerland on transitions from education to employment (www.tree.unibe.ch).