Customizing Stata graphs made easy

Ben Jann

University of Bern, ben.jann@soz.unibe.ch

2018 London Stata Conference
London, September 6–7, 2018
Outline

1. Introduction
2. Overview of new Stata commands
3. Basic procedure
4. Composite settings
5. Summary
Stata’s graph schemes

- Stata provides a number of so-called **schemes** that define the overall look of graphs.
- Some examples are as follows.
- Load data:

  ```
  . sysuse auto, clear
  (1978 Automobile Data)
  . generate pfor = price if foreign==1 & rep78>=3
     (52 missing values generated)
  . generate pdom = price if foreign==0 & rep78>=3
     (32 missing values generated)
  . separate price if rep78>=3, by(rep78) shortlabel
  ```

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>display format</th>
<th>value label</th>
</tr>
</thead>
<tbody>
<tr>
<td>price3</td>
<td>int</td>
<td>%8.0gc</td>
<td>rep78 == 3</td>
</tr>
<tr>
<td>price4</td>
<td>int</td>
<td>%8.0gc</td>
<td>rep78 == 4</td>
</tr>
<tr>
<td>price5</td>
<td>int</td>
<td>%8.0gc</td>
<td>rep78 == 5</td>
</tr>
</tbody>
</table>
. set scheme s2color
. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))

. set scheme s1mono
. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))
. set scheme economist
. tw (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>     4 "Foreign" 5 6 7))

Ben Jann (University of Bern)

London, 06.09.2018
User contributed schemes

- The number of available schemes in official Stata is somewhat limited.
- Hence, some users took the effort to develop custom scheme files and make them publicly available.
- Examples are as follows.
Atz (2011)

. set scheme tufte
. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))
Bischof (2017a)

. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7)) scheme(538)

. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7)) scheme(538bw)
Bischof (2017b)

.set scheme plotplain
.two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))

.set scheme plotplainblind
.twon (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))
Bischof (2017b)

```
. set scheme plottig
. two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>   4 "Foreign" 5 6 7))
```

```
. set scheme plottigblind
. two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>   4 "Foreign" 5 6 7))
```

(note: clockdir zyx2legend_position not found in scheme, default attributes used)
Briatte (2013)

```stata
.set scheme burd
.two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>   4 "Foreign" 5 6 7))
```

```stata
.set scheme burd5
.graph hbar price, over(rep78) asyvars
>   stack percent over(foreign) legend(off)
```

Ben Jann (University of Bern)
Juul (2003)

.set scheme lean1
.two (lpoly ci pfor length, clstyle(p1line))
> (lpoly ci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))

.set scheme lean2
.two (lpoly ci pfor length, clstyle(p1line))
> (lpoly ci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))
Morris (2013, 2015)

. set scheme mrc
. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))

. set scheme tfl
. two (lpolyci pfor length, clstyle(p1line))
> (lpolyci pdom length, clstyle(p2line))
> (scatter price? length, pstyle(p3 p4 p5))
> , ytitle(Price) legend(order(2 "Domestic"
> 4 "Foreign" 5 6 7))

(note: anglestyle symbol not found in scheme, default)
(note: anglestyle symbol not found in scheme, default)
(note: anglestyle symbol not found in scheme, default)
(note: anglestyle symbol not found in scheme, default)
Newson (2005)

.set scheme rbn1mono
.two (lpolyci pfor length, clstyle(p1line))
>  (lpolyci pdom length, clstyle(p2line))
>  (scatter price? length, pstyle(p3 p4 p5))
>  , ytitle(Price) legend(order(2 "Domestic"
>  4 "Foreign" 5 6 7))

.set scheme rbn3mono
.two (lpolyci pfor length, clstyle(p1line))
>  (lpolyci pdom length, clstyle(p2line))
>  (scatter price? length, pstyle(p3 p4 p5))
>  , ytitle(Price) legend(order(2 "Domestic"
>  4 "Foreign" 5 6 7))
Personal schemes

These additional schemes might provide useful, but most likely none of them will do exactly what you want.

Therefore, some users also create their personal scheme (for example, by modifying one of the schemes above and storing it under a new name in an appropriate place in the local system).

A problem, however, is that what you want depends on context (properties of the data, type of analysis, nature of results, context in which graphs are used, audience to which the graphs are presented, . . .).

This means that you have to create a new scheme file each time you want to change some detail. This is very tedious and it is difficult to keep an overview.
Dynamic schemes

- My argument is that graph settings should be dynamic in the sense that they are defined in the do-file that creates the graphs.
- That is, graph settings should not be part of the local system, they should be part of the analysis script.
- This is much more convenient. It also has the advantage that everything needed to reproduce your graphs can be included in a single file.
- The new \texttt{grstyle} package supports such practice. It provides commands that let you change the graph settings on the fly. It works by maintaining a temporary scheme file in the background.
Outline

1. Introduction
2. Overview of new Stata commands
3. Basic procedure
4. Composite settings
5. Summary
Overview of new Stata commands

- There are two new packages: grstyle and palettes.

- The grstyle package contains commands to change graph settings from within a do-file.
  
  grstyle init  
  initialize the settings  
  grstyle scheme entry  
  add a single scheme entry  
  grstyle set ...  
  add composite settings  
  grstyle type  
  view the settings  
  grstyle clear  
  clear the settings

- The palettes package contains commands to manage colors, marker symbols, and line patterns. These commands are used by grstyle, but they can also be used separately.
  
  colorpalette  
  retrieve colors  
  symbolpalette  
  retrieve marker symbols  
  linepalette  
  retrieve line patterns
Outline

1. Introduction

2. Overview of new Stata commands

3. Basic procedure

4. Composite settings

5. Summary
Basic procedure

The basic procedure is to first call `grstyle init`, then add the desired settings using a series of `grstyle` commands, and then create the graphs:

```
set scheme schemename
grstyle init
grstyle ...
grstyle ...
  :
graph command
graph command
  :
grstyle clear
```
Example

Here's the graph of before, using the s2color scheme (Stata’s default):

```
.set scheme s2color
.two (lpolyci pfor length, clstyle(p1line))
>  (lpolyci pdom length, clstyle(p2line))
>  (scatter price? length, pstyle(p3 p4 p5))
>  , ytitle(Price) legend(order(2 "Domestic"
>        4 "Foreign" 5 6 7))
```

Now let’s change how things look.
Example

Step 1: get rid of background color, use horizontal labels on the Y axis, make CIs transparent

```bash
. grstyle init
. grstyle color background white
. grstyle anglestyle vertical_tick horizontal
. grstyle color ci_area gs12%50
. grstyle color ci_arealine gs12%0
. two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>       4 "Foreign" 5 6 7))
```

The settings added by `grstyle` are scheme entries. Their syntax is:

```
attribute element style
```

See `help scheme entries` for details.
Step 2: change the rendering of the grid

- `grstyle color major_grid dimgray`
- `grstyle linewidth major_grid thin`
- `grstyle yesno draw_major_hgrid yes`
- `grstyle yesno grid_draw_min yes`
- `grstyle yesno grid_draw_max yes`
- `two (lpolyci pfor length, clstyle(p1line))`
  >  `(lpolyci pdom length, clstyle(p2line))`
  >  `(scatter price? length, pstyle(p3 p4 p5))`
  >  `, ytitle(Price) legend(order(2 "Domestic"
  >  4 "Foreign" 5 6 7))`
Example

Step 3: move the legend to the right and remove the frame

```
. grstyle clockdir legend_position 4
. grstyle numstyle legend_cols 1
. grstyle linestyle legend none
. two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>   4 "Foreign" 5 6 7))
```
Example

Revert back to original:

```
 . grstyle clear
 . two (lpolyci pfor length, clstyle(p1line))
 >   (lpolyci pdom length, clstyle(p2line))
 >   (scatter price? length, pstyle(p3 p4 p5))
 >   , ytitle(Price) legend(order(2 "Domestic"
 >       4 "Foreign" 5 6 7))
```

![Graph with price on the y-axis and length on the x-axis, showing two sets of data points with different styles and legends.](image-url)
Outline

1. Introduction
2. Overview of new Stata commands
3. Basic procedure
4. Composite settings
5. Summary
Composite settings

- I assume you got the idea.
- But you probably ask yourself:
  - “Wasn’t the goal to make things easy? This still looks pretty complicated to me.”
- Yes, scheme entry syntax (see help scheme entries) is unfamiliar and hard to remember.
- This is why there is a command called grstyle set that can be used to generate scheme entries for some frequently used settings.
- grstyle set has various subcommands to determine the style of background and coordinate system, the legend, confidence areas, colors, symbols, line pattern, and sizes.
Plain background, horizontal labels, full grid, non-extended axis lines, transparent confidence areas:

```stata
. set scheme s2color
. grstyle init
. grstyle set plain, horizontal grid noextend
. grstyle set ci
. two (lpolyci pfor length, clstyle(p1line))
  >  (lpolyci pdom length, clstyle(p2line))
  >  (scatter price? length, pstyle(p3 p4 p5))
  >  , ytitle(Price) legend(order(2 "Domestic"
  >  4 "Foreign" 5 6 7))
```

![Graph example](attachment:image.png)
Contents of temporary scheme file

```
   . grststyle type
#include s2color
  color background white
  color plotregion none
  linestyle plotregion none
  color grid dimgray
  color major_grid dimgray
  color minor_grid dimgray
  linewidth grid thin
  linewidth major_grid thin
  linewidth minor_grid thin
yesno draw_major_vgrid yes
yesno draw_major_hgrid yes
yesno draw_minor_vgrid yes
yesno draw_minor_hgrid yes
yesno grid_draw_min yes
yesno grid_draw_max yes
  color heading black
  color textbox none
  color bylabel_outline none
  color mat_label_box none
anglestyle vertical_tick horizontal
  yesno extend_axes_low no
  yesno extend_axes_high no
  yesno extend_axes_full_low no
  yesno extend_axes_full_high no
  color ci_area "gs12%50"
  color ci_arealine "gs12%0"
  color ci2_area "ltkhaki%50"
  color ci2_arealine "ltkhaki%0"
```

Alternative backgrounds

Mesh of grid lines without axes:

\begin{verbatim}
. grstyle init
. grstyle set ci
. grstyle set mesh, horizontal compact minor
. two (lpolyci pfor length, clstyle(p1line))
  (lpolyci pdom length, clstyle(p2line))
  (scatter price? length, pstyle(p3 p4 p5))
  , ytitle(Price) legend(order(2 "Domestic"
   > 4 "Foreign" 5 6 7))
\end{verbatim}

Inverted mesh:

\begin{verbatim}
. grstyle init
. grstyle set ci
. grstyle set imesh, horizontal minor
. two (lpolyci pfor length, clstyle(p1line))
  (lpolyci pdom length, clstyle(p2line))
  (scatter price? length, pstyle(p3 p4 p5))
  , ytitle(Price) legend(order(2 "Domestic"
   > 4 "Foreign" 5 6 7))
\end{verbatim}
The legend

Place legend at top-left inside plot region:

```
. grstyle init
. grstyle set ci
. grstyle set plain, grid
. grstyle set legend 10, inside
. two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>   4 "Foreign" 5 6 7))
```

Remove frame and place on the right:

```
. grstyle init
. grstyle set ci
. grstyle set imesh, horizontal
. grstyle set legend 4, nobox
. two (lpolyci pfor length, clstyle(p1line))
>   (lpolyci pdom length, clstyle(p2line))
>   (scatter price? length, pstyle(p3 p4 p5))
>   , ytitle(Price) legend(order(2 "Domestic"
>   4 "Foreign" 5 6 7))
```
Colors

... previous example with different colors

```
  . grstyle set color hue, n(5) opacity(60)
  . grstyle set symbolsize large
  . two (lpolyci pfor length, clstyle(p1line))
  > (lpolyci pdom length, clstyle(p2line))
  > (scatter price? length, pstyle(p3 p4 p5))
  > , ytitle(Price) legend(order(2 "Domestic"
  > 4 "Foreign" 5 6 7))
```

... yet another set of colors

```
  . grstyle set color RdYlGn, n(5)
  . graph hbar price, over(rep78) asyvars
  > stack percent over(foreign) legend(off)
```

(if scheme entries are defined repeatedly, Stata will always use the last definition)
Colors

- A large number of color palettes is available.
  - palettes from official Stata’s scheme files
  - palettes from user contributed scheme files
  - HCL (Hue-Chroma-Luminance) and HSV (Hue-Saturation-Value) color generators
  - palette collections such as ColorBrewer (Brewer et al. 2003)
  - etc.

- Can also specify custom colors in RGB, CMYK, HSV, HCL, or Hex (web colors).
- Can adjust intensity and opacity (Stata 15).
- See http://repec.sowi.unibe.ch/stata/palettes/
Symbols and line patterns

Using the default sequence:

. grstyle init
. grstyle set plain, grid
. grstyle set color Set1
. grstyle set symbol
. grstyle set lpattern
. twoway (connected price? weight, sort),
>   ytitle(Price)

Specifying a custom sequence:

. grstyle set symbol t s d
. grstyle set lpattern "-.." "--." "_"
. twoway (connected price? weight, sort),
>   ytitle(Price)
A specific feature of `grstyle set` is that it can set absolute sizes (inch, pt, cm, or mm).

Sizes in Stata graphs are always relative. The procedure therefore is to first use `grstyle set graphsize` to determine the size of the graph.

After that, use `grstyle set size` to set the size of text etc. There are also commands for symbol sizes, line widths, and margins.

Of course, the target sizes will only be preserved as long as you do not change the graph size.
Sizes

. grstyle init
. grstyle set plain, grid
. grstyle set color d3
. grstyle set legend 2
. grstyle set graphsize 6cm 9cm
. grstyle set size 10pt: heading
. grstyle set size 8pt: subheading axis_title
. grstyle set size 6pt: tick_label key_label
. grstyle set symbolsize 3 4 5, pt
. grstyle set linewidth .8pt: plineplot
. grstyle set linewidth .4pt: legend axisline tick major_grid
. grstyle set linewidth 0: pmark
. grstyle set margin zero
. 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 "3pt" 2 "4pt" 3 "5pt" 4 ".8pt"))
sizes
Contents

1 Introduction

2 Overview of new Stata commands

3 Basic procedure

4 Composite settings

5 Summary
Summary

- *grstyle* provides a tool to change graph settings from within a do-file. Manual editing of scheme files is no longer needed.
- *grstyle set* is a powerful convenience command to define a wide variety of style settings without having to know much about scheme entry syntax.
- *grstyle* allows modular assembling of a scheme file; multiple calls to *grstyle set* and *grstyle scheme entry* can be freely combined.
- *colorpalette* provides a versatile color management system.
- More information and examples:
  - [http://repec.sowi.unibe.ch/stata/grstyle](http://repec.sowi.unibe.ch/stata/grstyle)
  - [http://repec.sowi.unibe.ch/stata/palettes](http://repec.sowi.unibe.ch/stata/palettes)
References


References


