Customizing Stata graphs made easy

Ben Jann

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

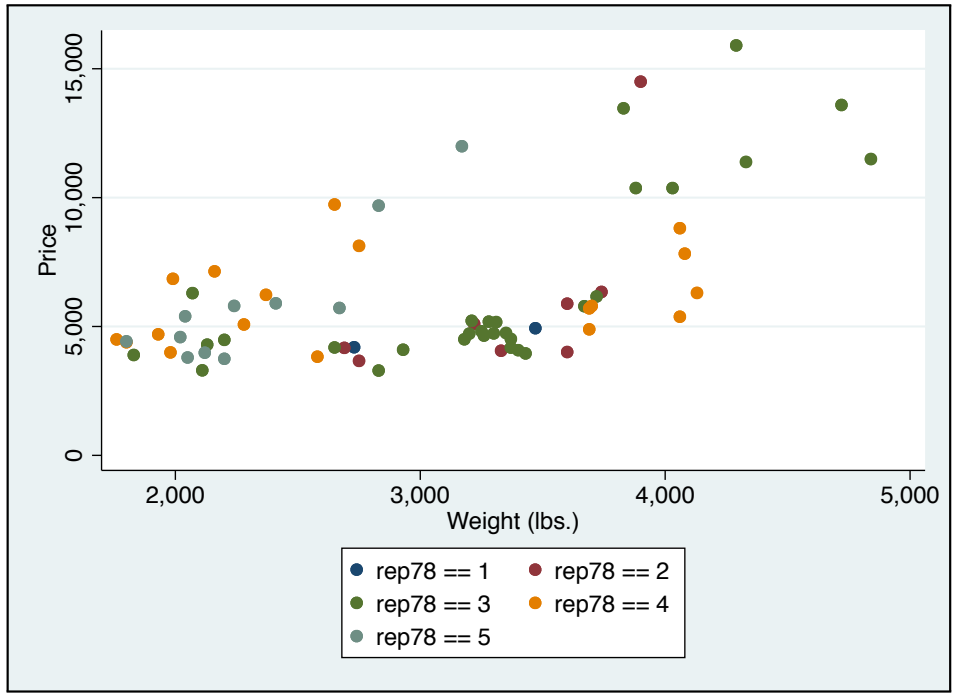
2018 German Stata Users Group meeting
University of Konstanz, June 22, 2018

Outline

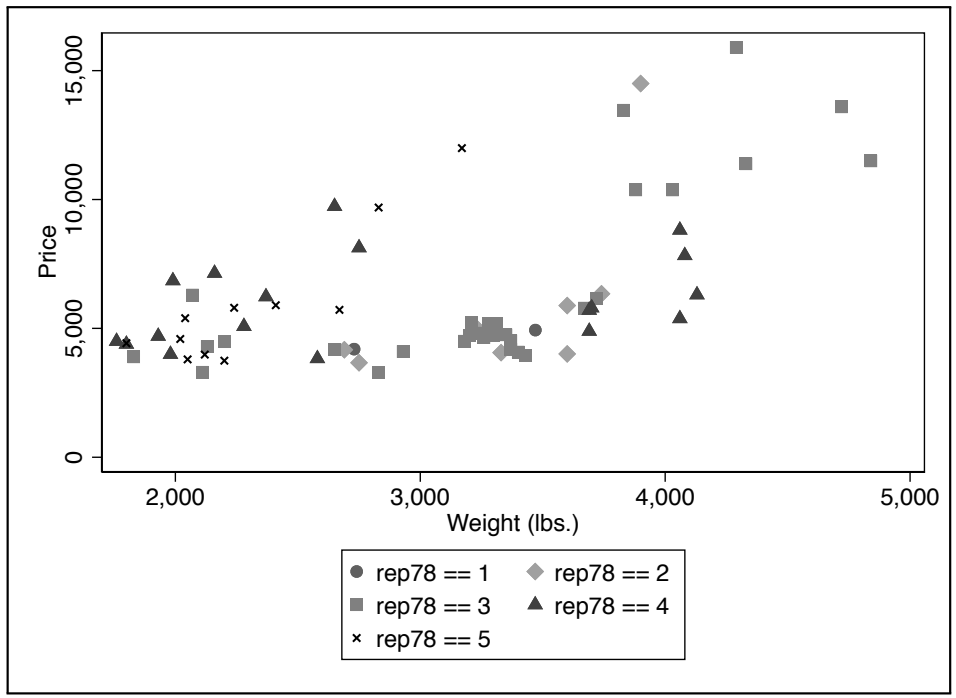
Stata's graph schemes

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

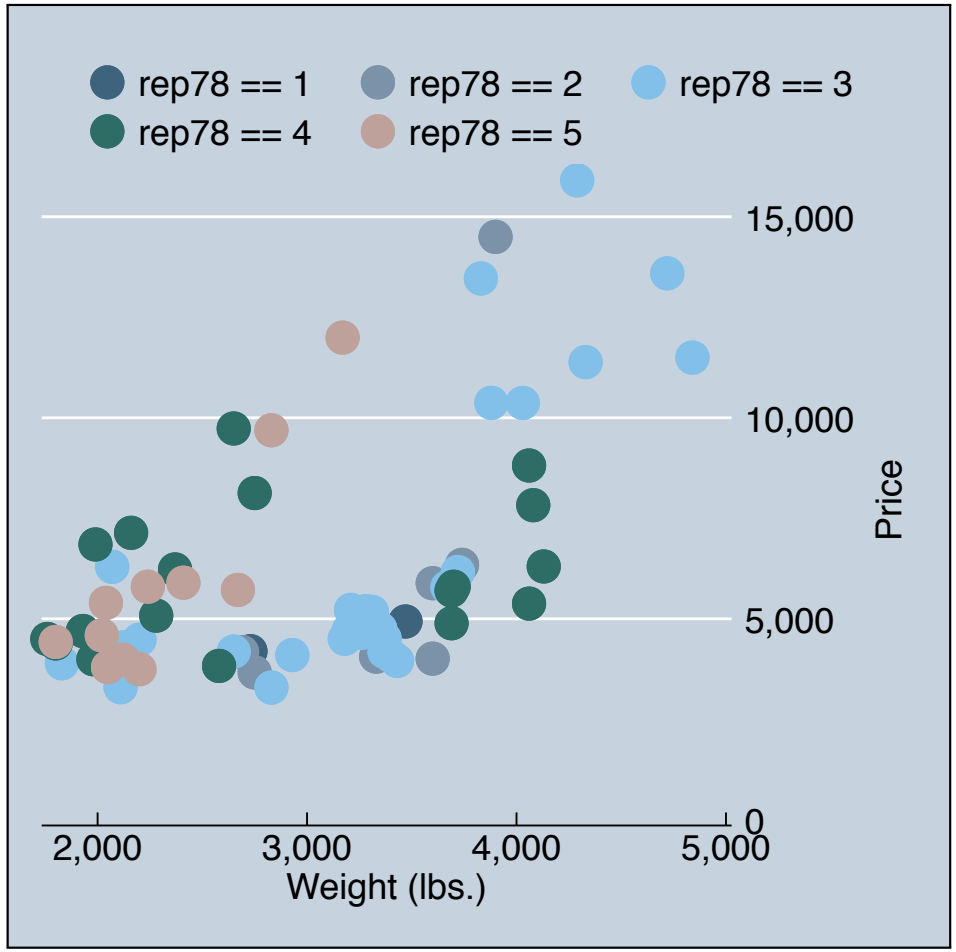
```
. sysuse auto, clear
(1978 Automobile Data)
. separate price, by(rep) shortlabel
   (output omitted)
. set scheme s2color
. scatter price? weight, ytitle(Price)
```




```
. set scheme s1mono
. scatter price? weight, ytitle(Price)
```



```
. set scheme economist
. scatter price? weight, ytitle(Price)
```

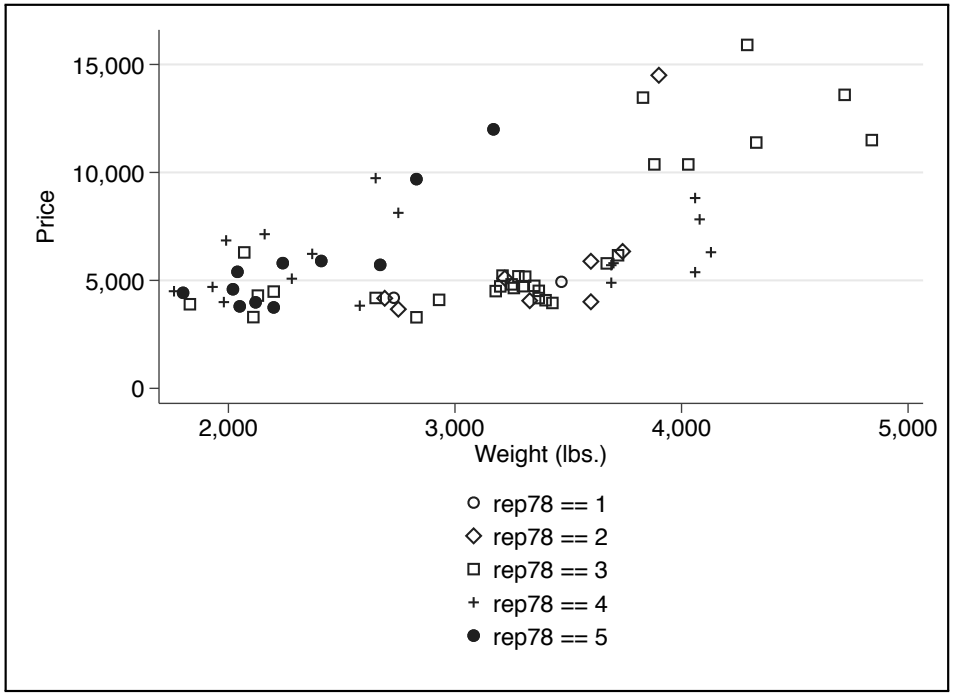


User contributed schemes

- 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.

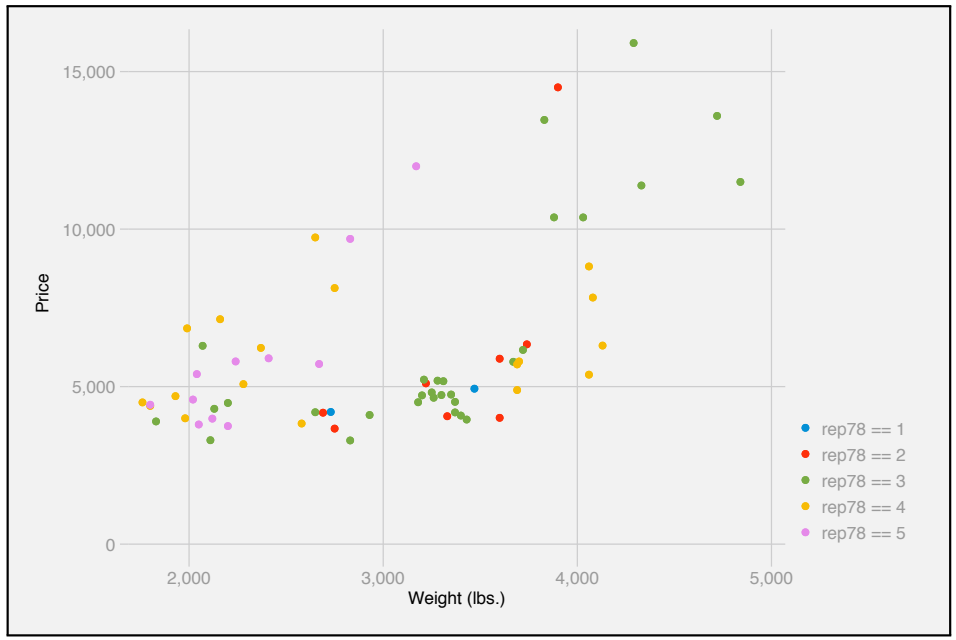
Atz (2011)

```
. set scheme tufte
. scatter price? weight, ytitle(Price)
```

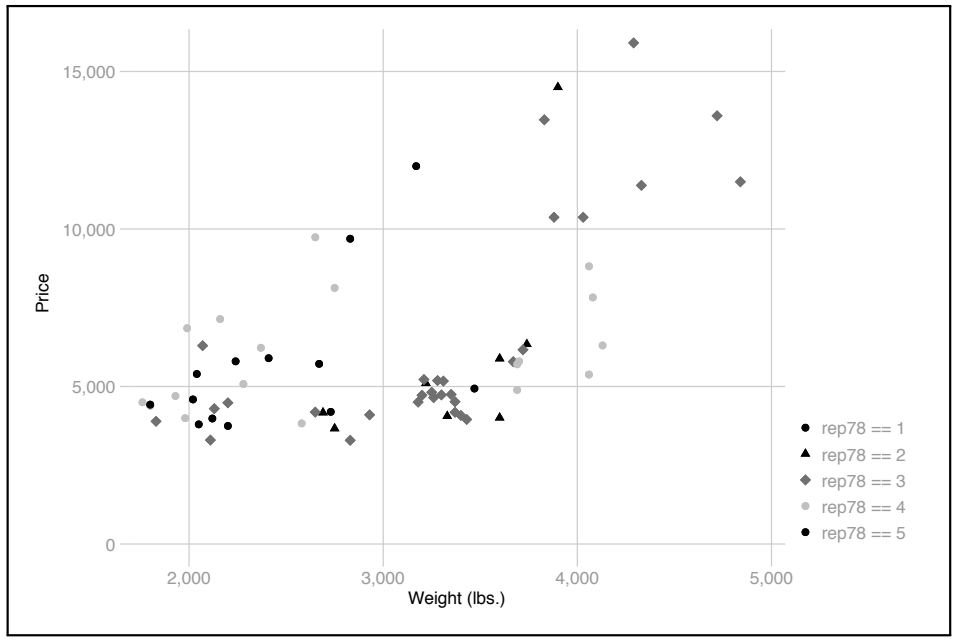


Bischof (2017a)

```
. scatter price? weight, ytitle(Price) ///
>     scheme(538)
```

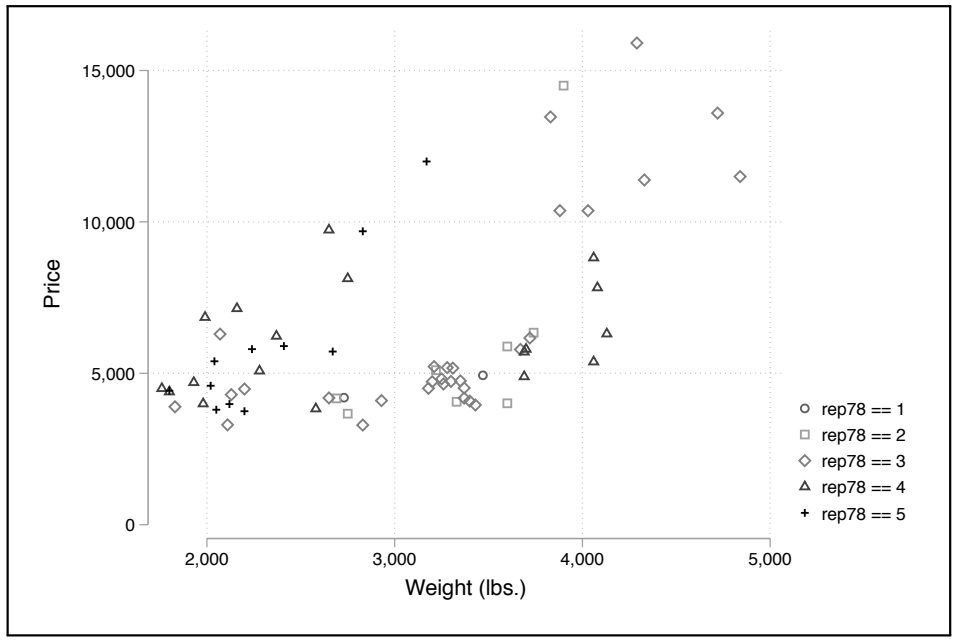



```
. scatter price? weight, ytitle(Price) ///
>     scheme(538bw)
```

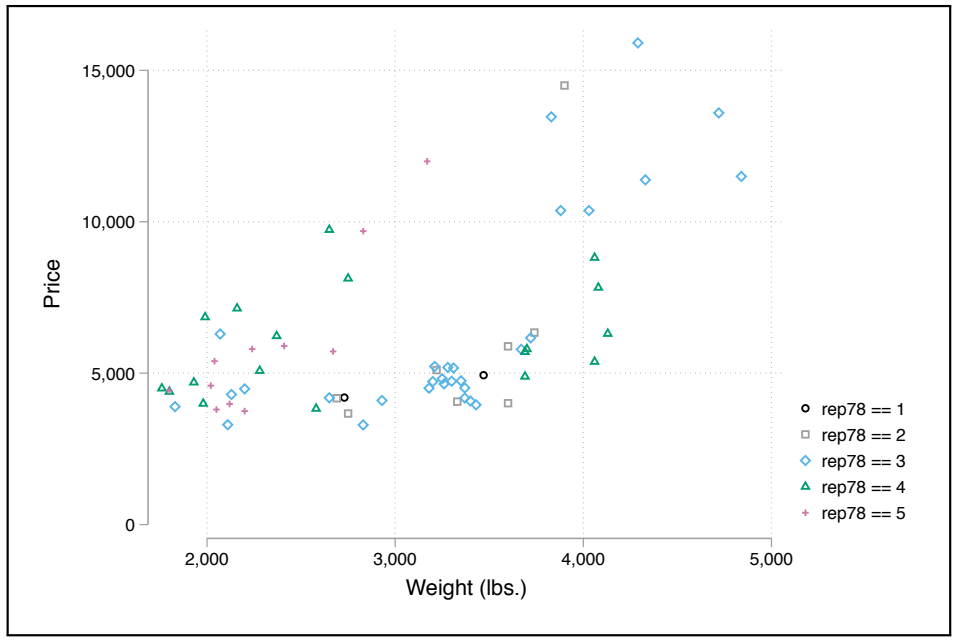


Bischof (2017b)

```
. set scheme plotplain
. scatter price? weight, ytitle(Price)
```

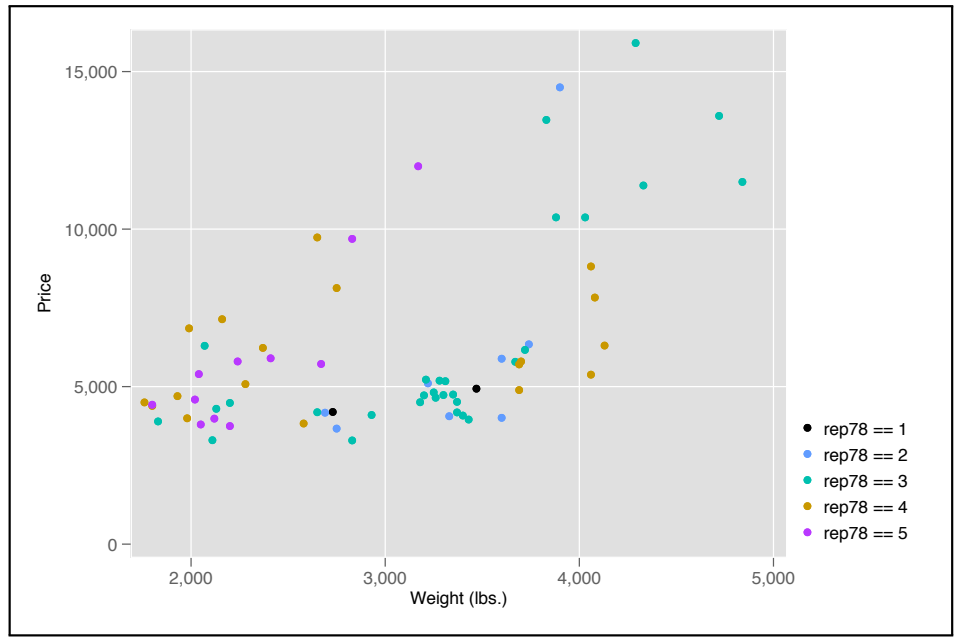



```
. set scheme plotplainblind
. scatter price? weight, ytitle(Price)
```

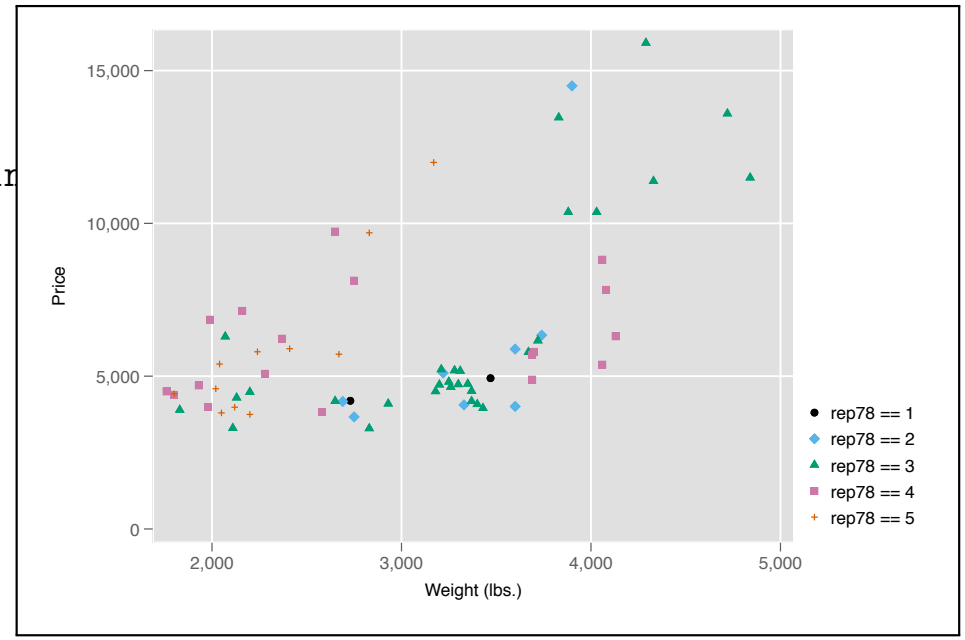


Bischof (2017b)

```
. set scheme plottig
. scatter price? weight, ytitle(Price)
```

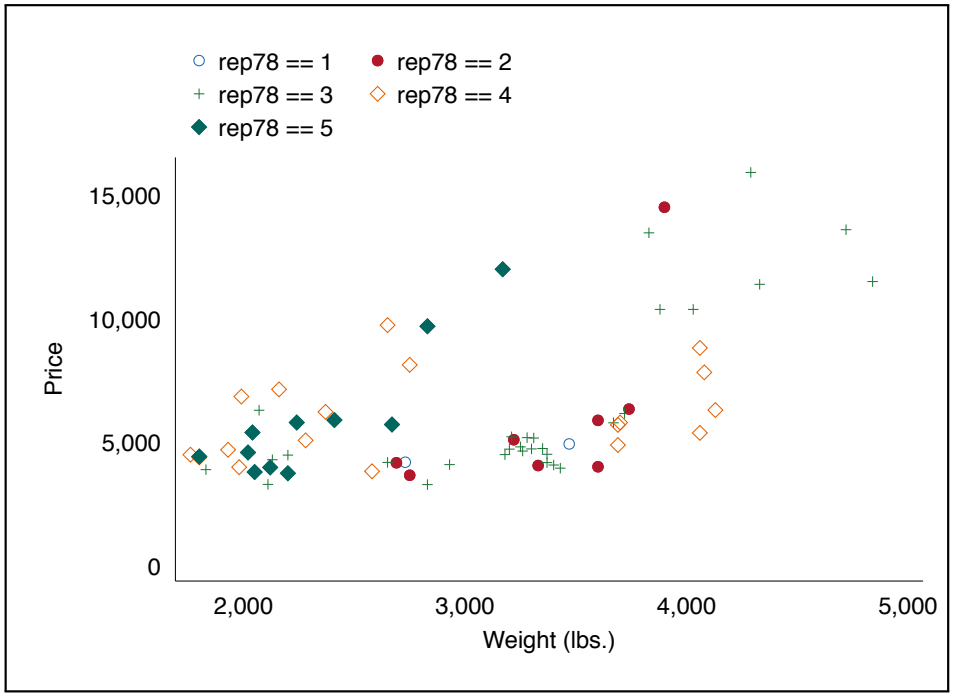



```
. set scheme plottigblind
. scatter price? weight, ytitle(Price)
(note:  clockdir zyx2legend_position not found in
```

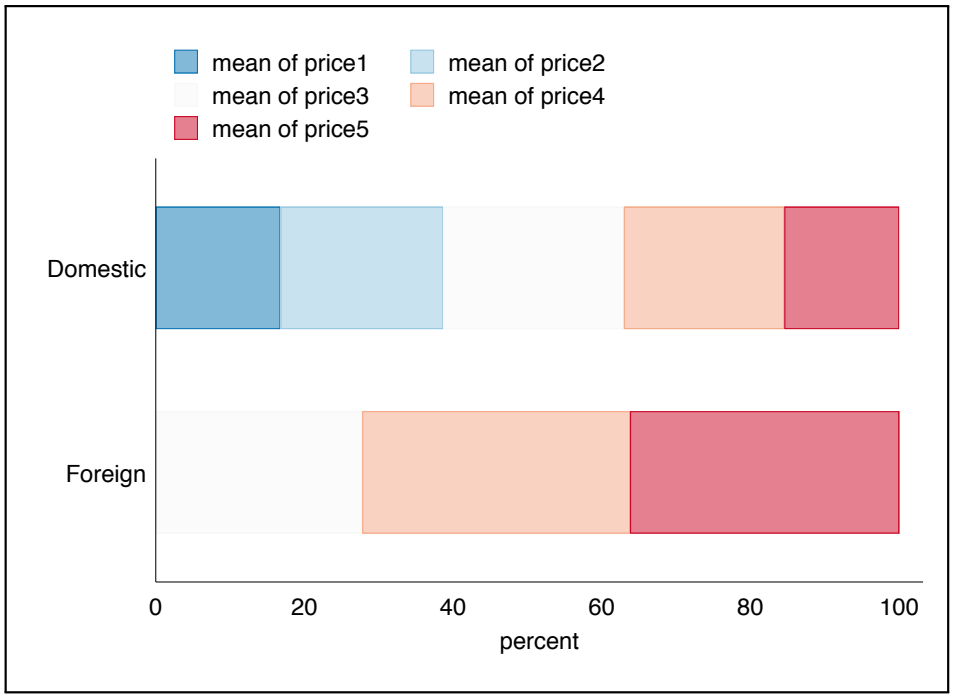


Briatte (2013)

```
. set scheme burd
. scatter price? weight, ytitle(Price)
```

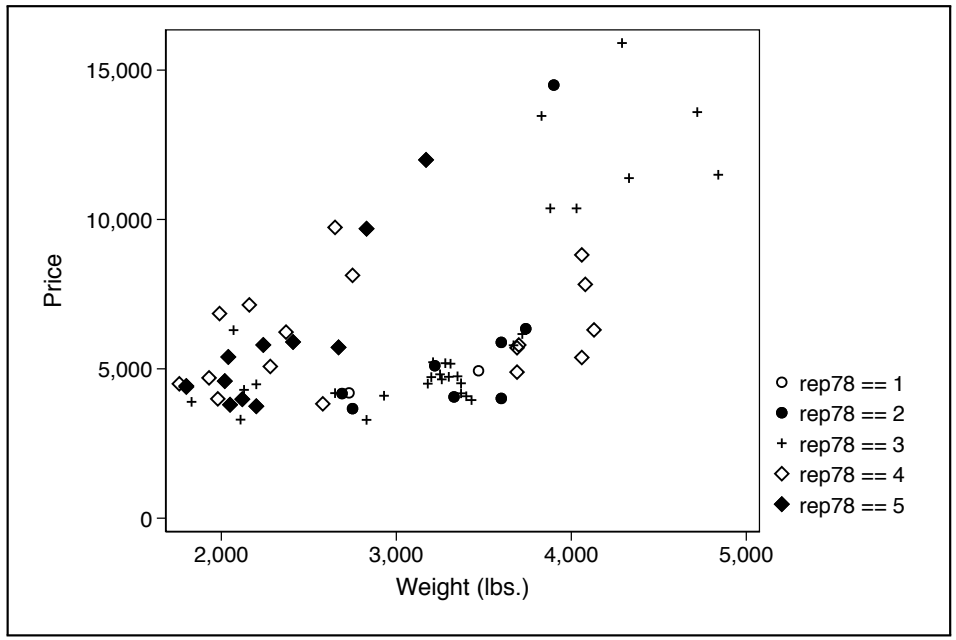



```
. set scheme burd5
. graph hbar price?, stack percent over(foreign)
```

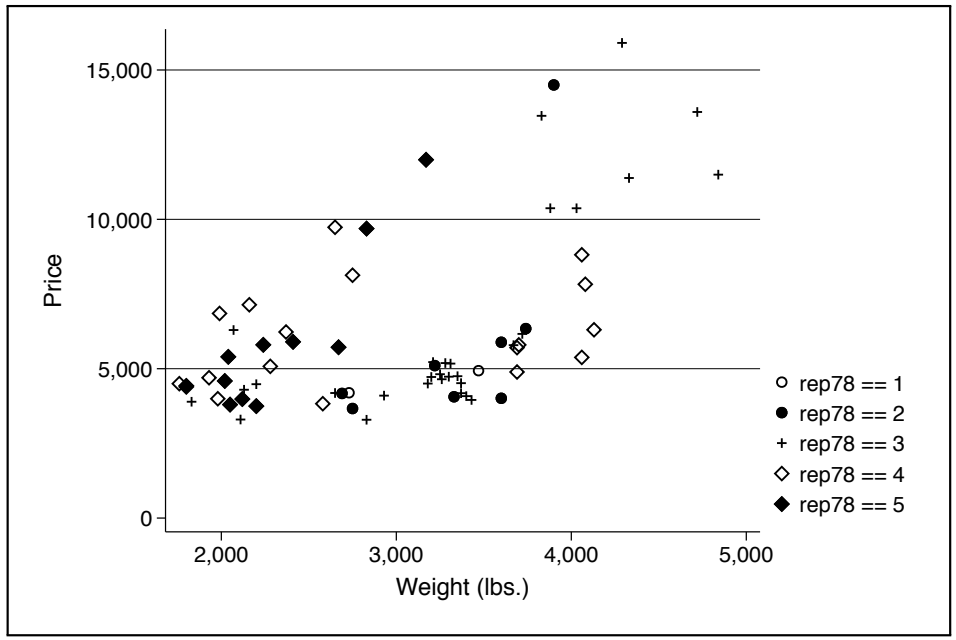


Juul (2003)

```
. set scheme lean1
. scatter price? weight, ytitle(Price)
```

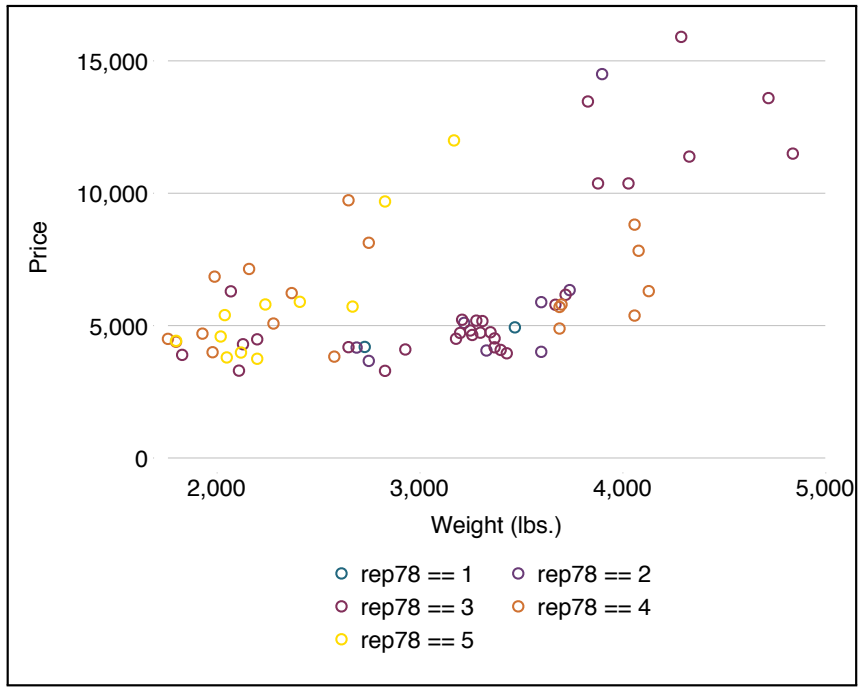



```
. set scheme lean2
. scatter price? weight, ytitle(Price)
```

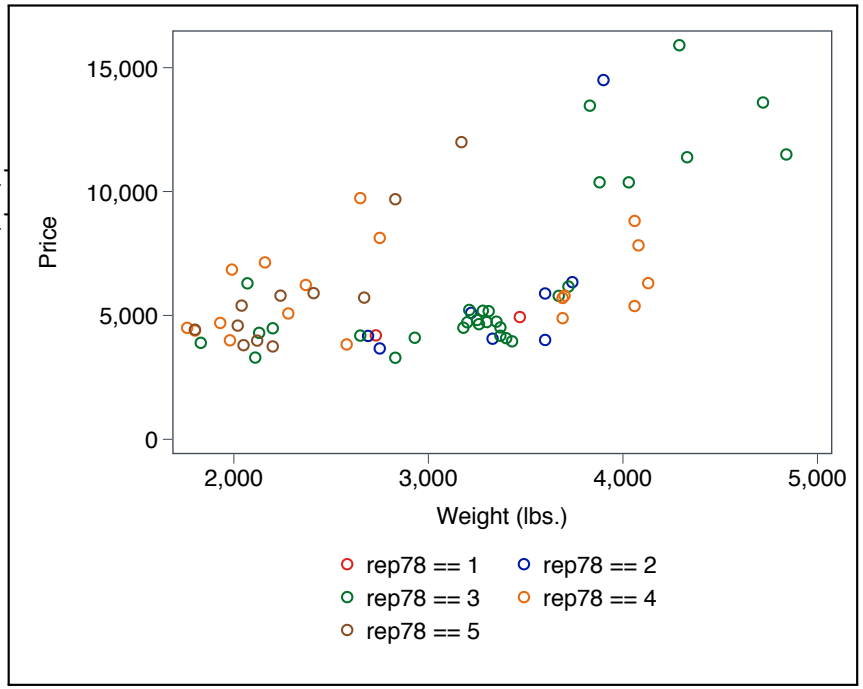


Morris (2013, 2015)

```
. set scheme mrc
. scatter price? weight, ytitle(Price)
```

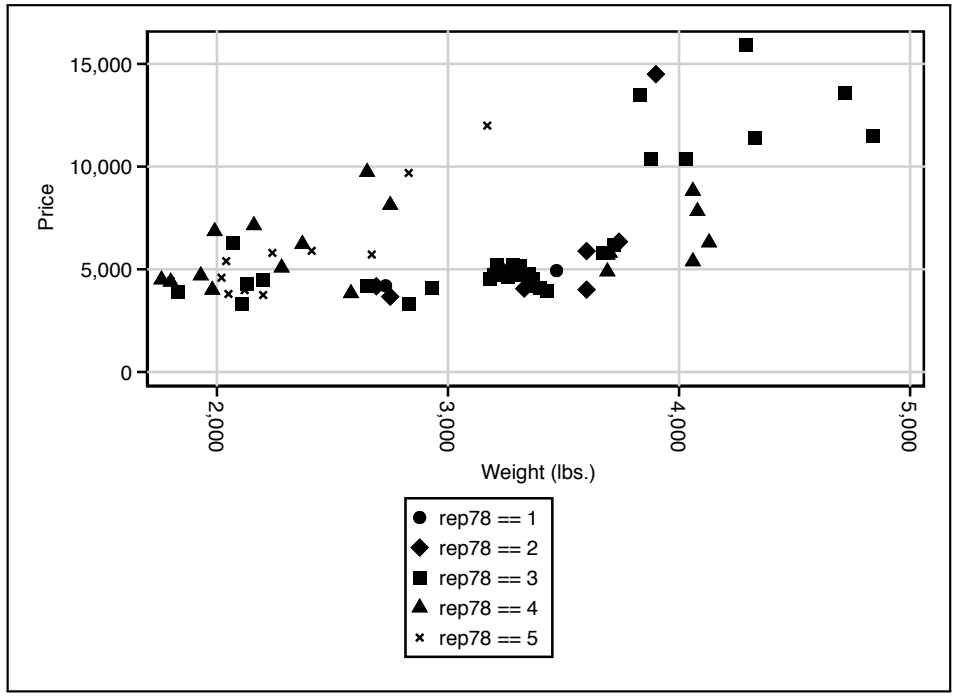



```
. set scheme tfl
. scatter price? weight, ytitle(Price)
(note:  anglestyle symbol not found in scheme, defaul
(note:  anglestyle symbol not found in scheme, defaul
```

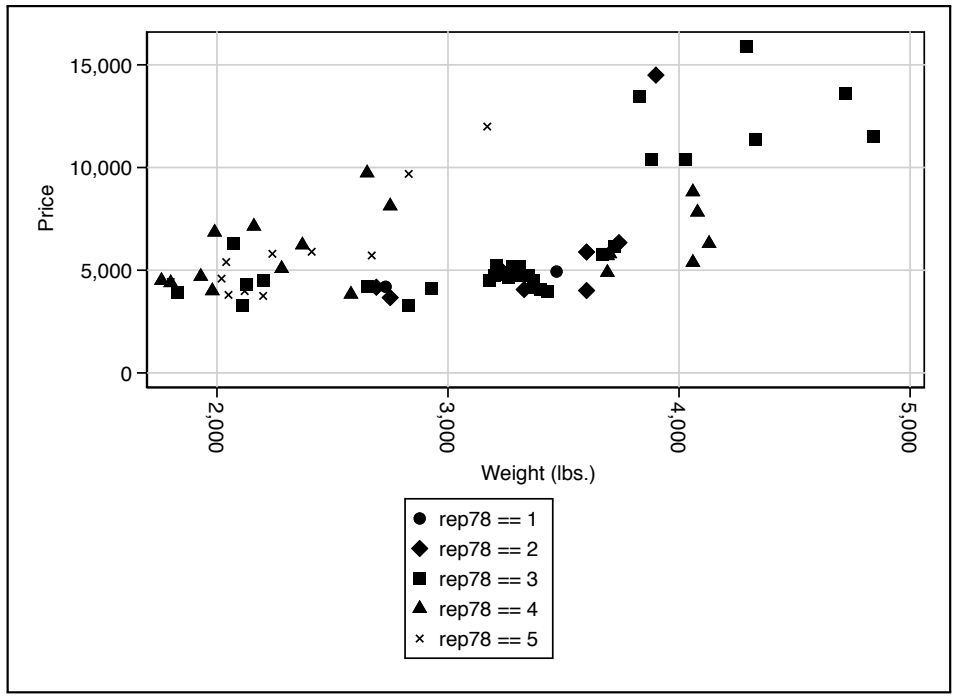


Newson (2005)

```
. set scheme rbn1mono
. scatter price? weight, ytitle(Price)
```




```
. set scheme rbn3mono
. scatter price? weight, ytitle(Price)
```



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 `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

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 a color palette
`symbolpalette`	retrieve a symbol palette
`linepalette`	retrieve a line pattern palette

Outline

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

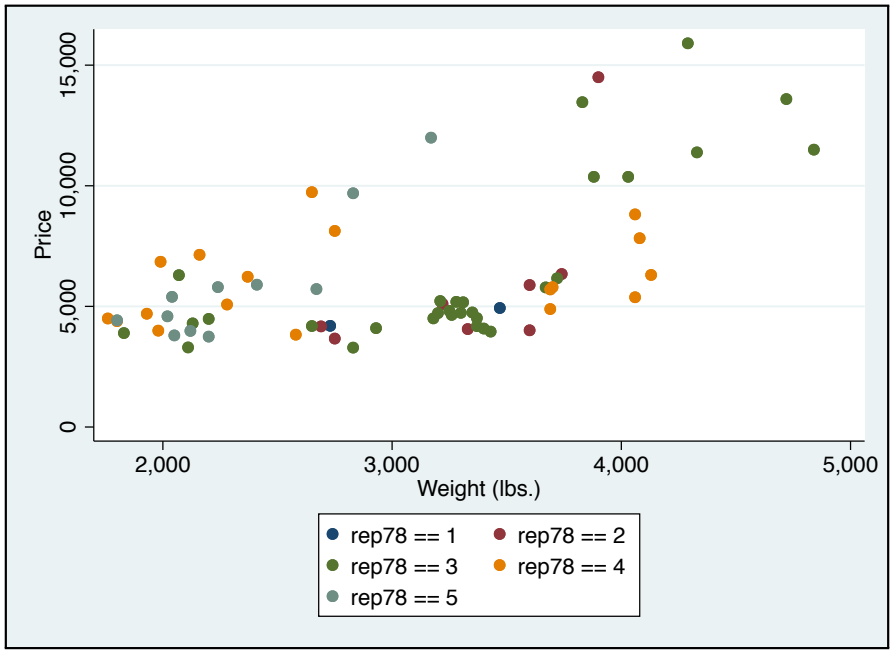
  ⋮

  `grstyle clear`

Example

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

```
. set scheme s2color
. scatter price? weight, ytitle(Price)
```

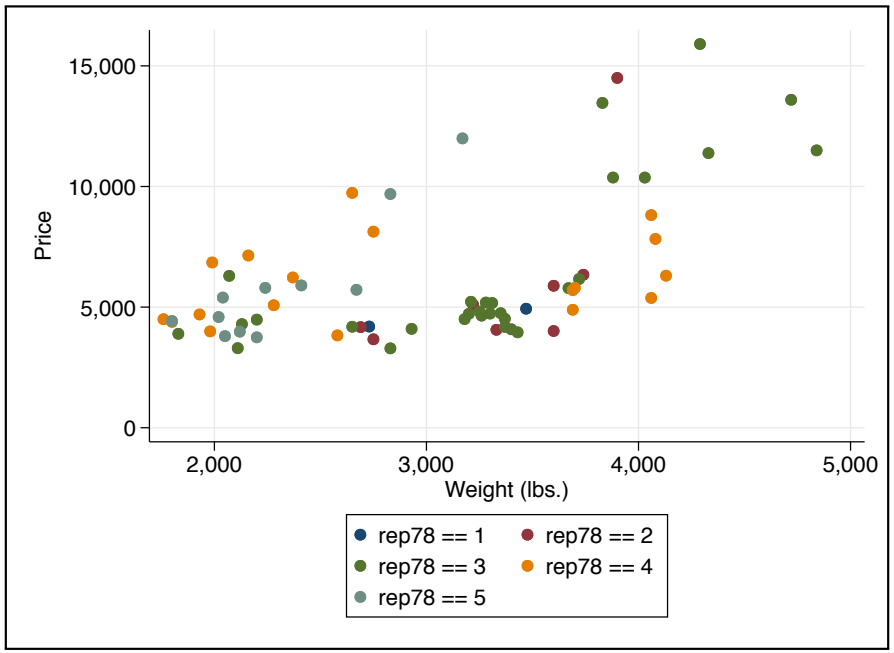


- Now let's change how things look.

Example

- Step 1: get rid of background color, change the rendering of the grid, use horizontal labels on the Y axis

```
. grstyle init
. grstyle color background white
. 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
. grstyle anglestyle vertical_tick horizontal
. scatter price? weight, ytitle(Price)
```



- The settings added by `grstyle` use scheme entry syntax. The syntax of scheme entries is

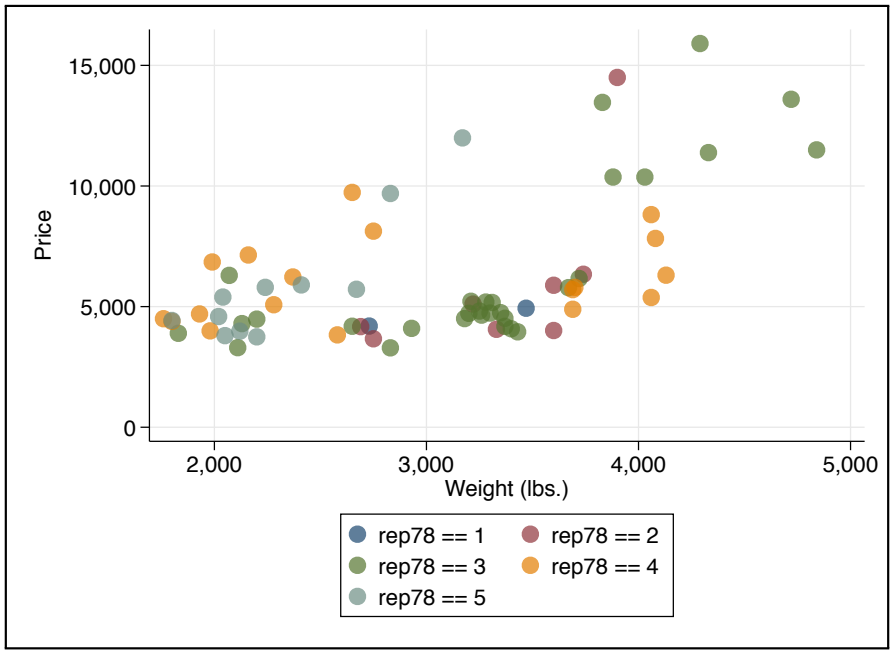
  attribute element style

  See `help scheme entries`.

Example

- Step 2: use larger marker symbols and make them transparent

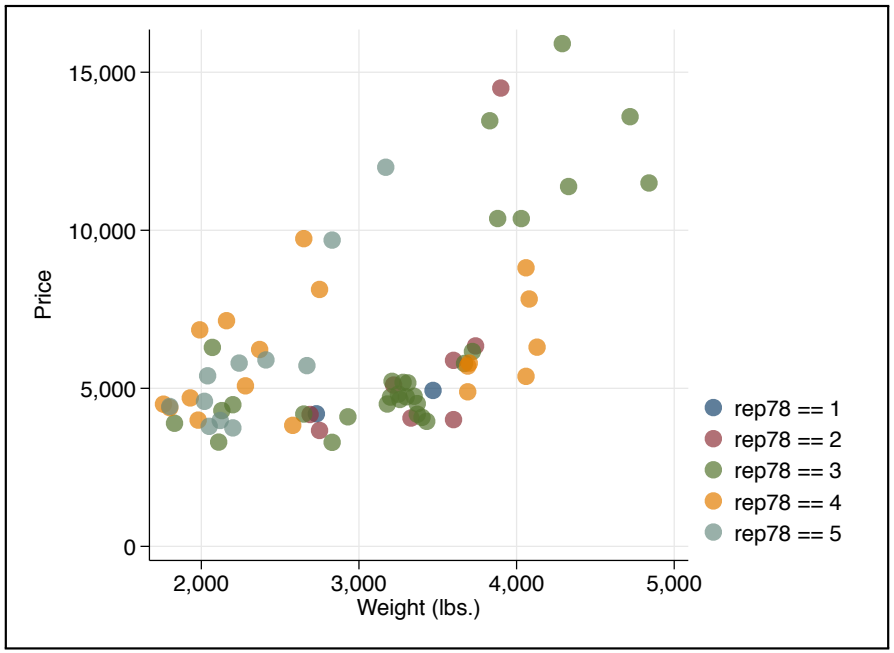
```
. grstyle symbolsize p large
. grstyle color p1markline navy%0
. grstyle color p1markfill navy%70
. grstyle color p2markline maroon%0
. grstyle color p2markfill maroon%70
. grstyle color p3markline forest_green%0
. grstyle color p3markfill forest_green%70
. grstyle color p4markline dkorange%0
. grstyle color p4markfill dkorange%70
. grstyle color p5markline teal%0
. grstyle color p5markfill teal%70
. scatter price? weight, ytitle(Price)
```



Example

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

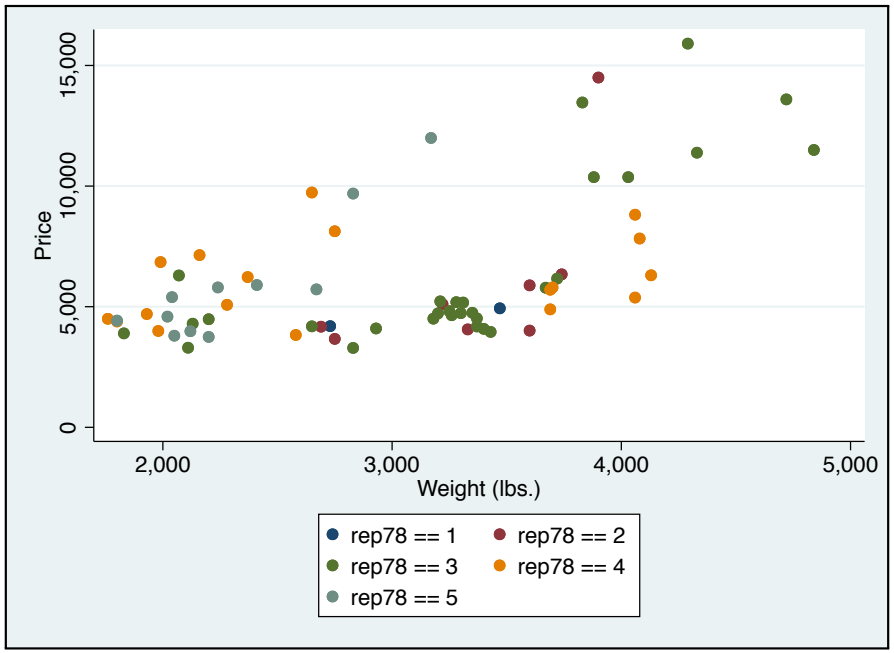
```
. grstyle clockdir legend_position 4
. grstyle numstyle legend_cols 1
. grstyle linestyle legend none
. scatter price? weight, ytitle(Price)
```



Example

- Revert back to original.

```
. grstyle clear
. scatter price? weight, ytitle(Price)
```



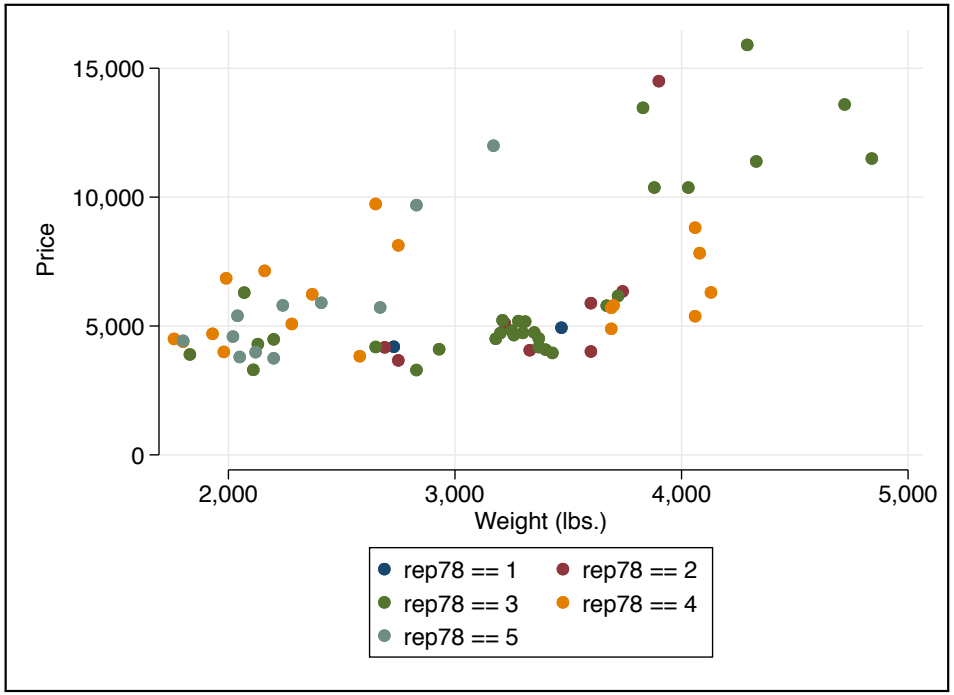
Outline

Composite settings

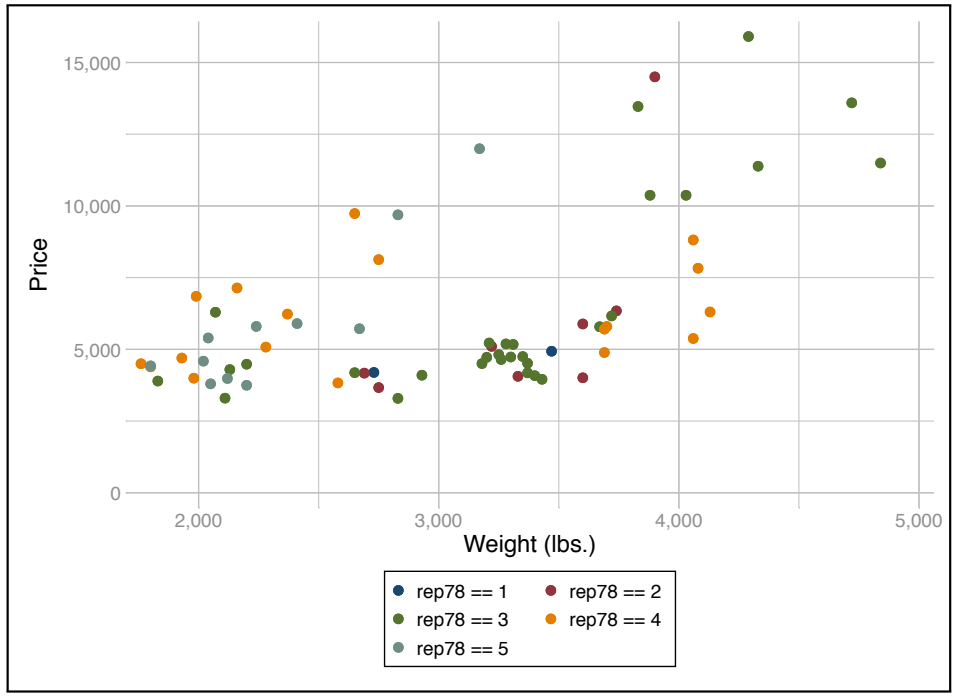
- I assume you got the idea.
- But you probably ask yourself:
  - "Wasn't the goal to make things easy? This still looks pretty complicated to me."
- Yes, scheme entry syntax (see `help scheme entries`) is unfamiliar and hard to remember.
- This is why there is a command called `grstyle set` that can 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.

Background and coordinate system

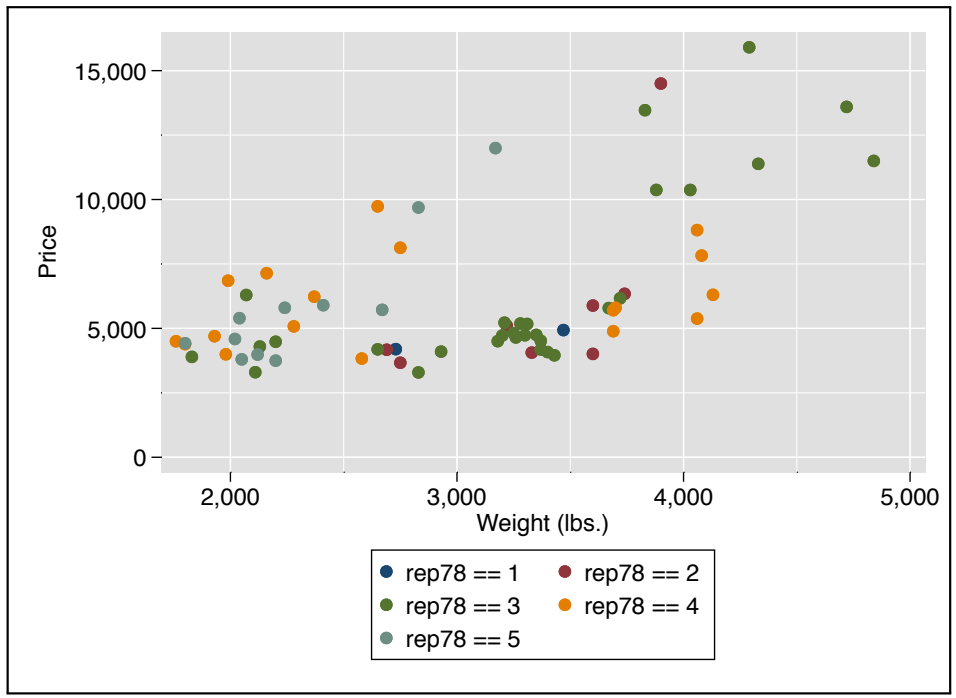
```
. grstyle init
. grstyle set plain, horizontal grid noextend
. scatter price? weight, ytitle(Price)
```



```
. grstyle init
. grstyle set mesh, horizontal compact minor
. scatter price? weight, ytitle(Price)
```

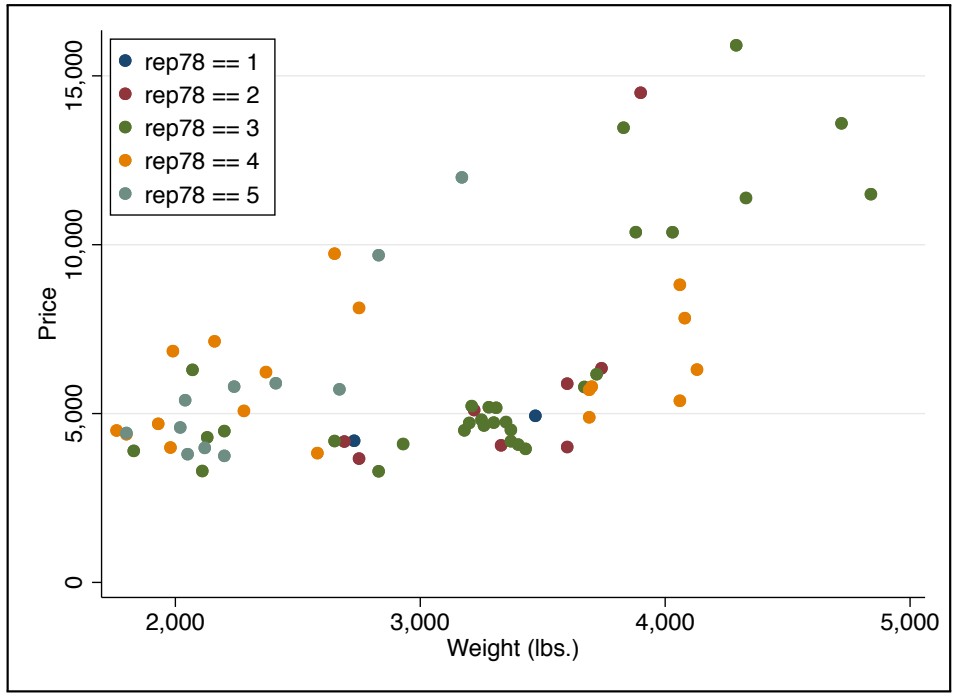



```
. grstyle init
. grstyle set imesh, horizontal minor
. scatter price? weight, ytitle(Price)
```

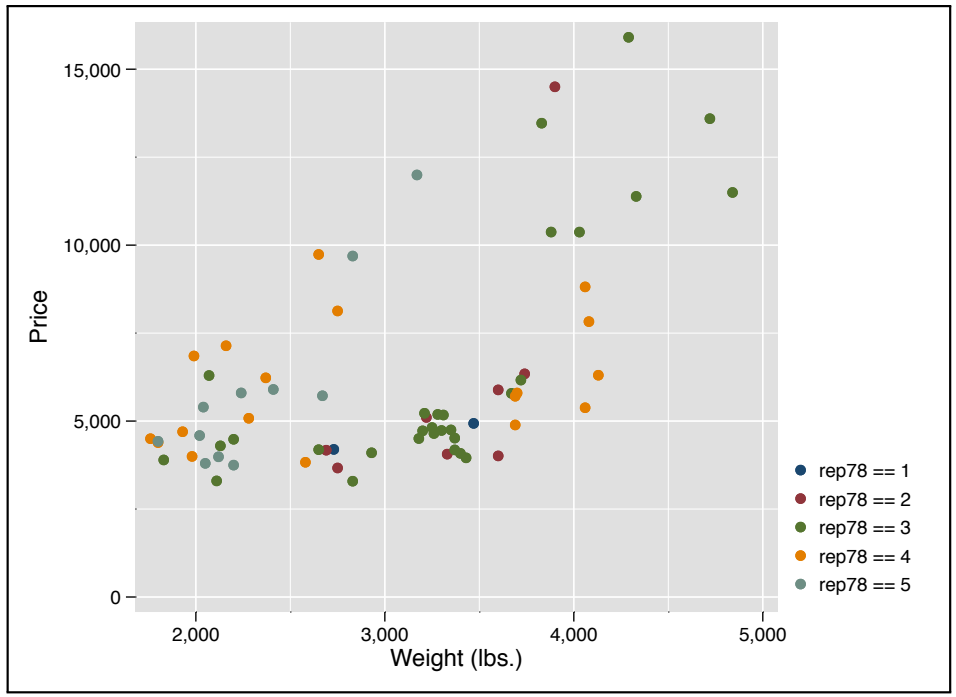


The legend

```
. grstyle init
. grstyle set plain
. grstyle set legend 10, inside
. scatter price? weight, ytitle(Price)
```

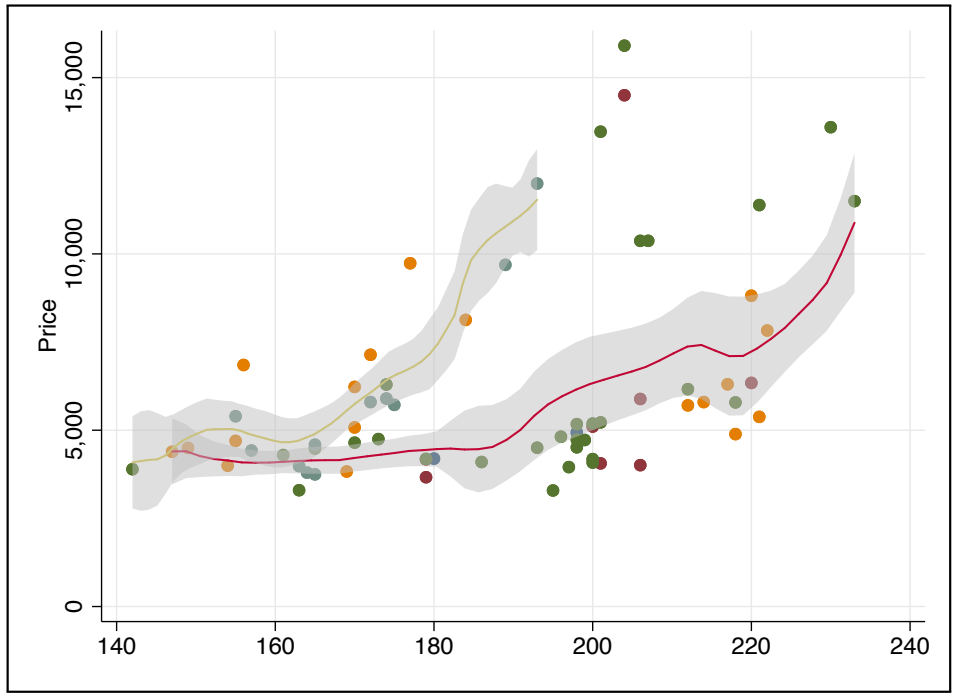



```
. grstyle init
. grstyle set imesh, horizontal compact minor
. grstyle set legend 4, nobox
. scatter price? weight, ytitle(Price)
```



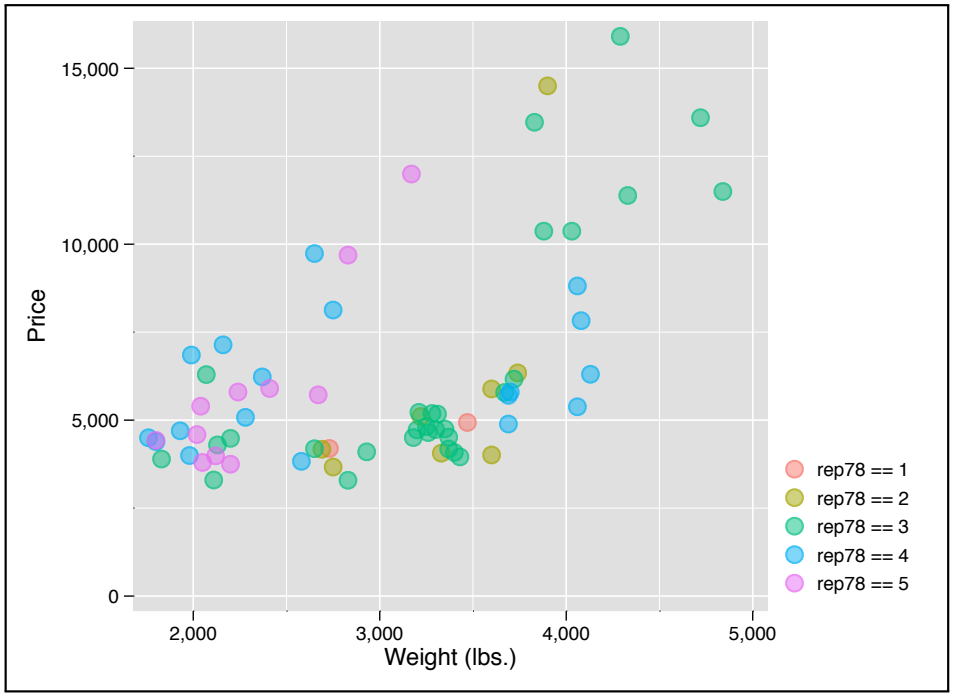
Transparent CIs

```
. grstyle init
. grstyle set plain, grid
. grstyle set ci
. twoway (scatter price? length) ///
>     (lpolyci price length if foreign==0) ///
>     (lpolyci price length if foreign==1) ///
>     , ytitle(Price) legend(off)
```

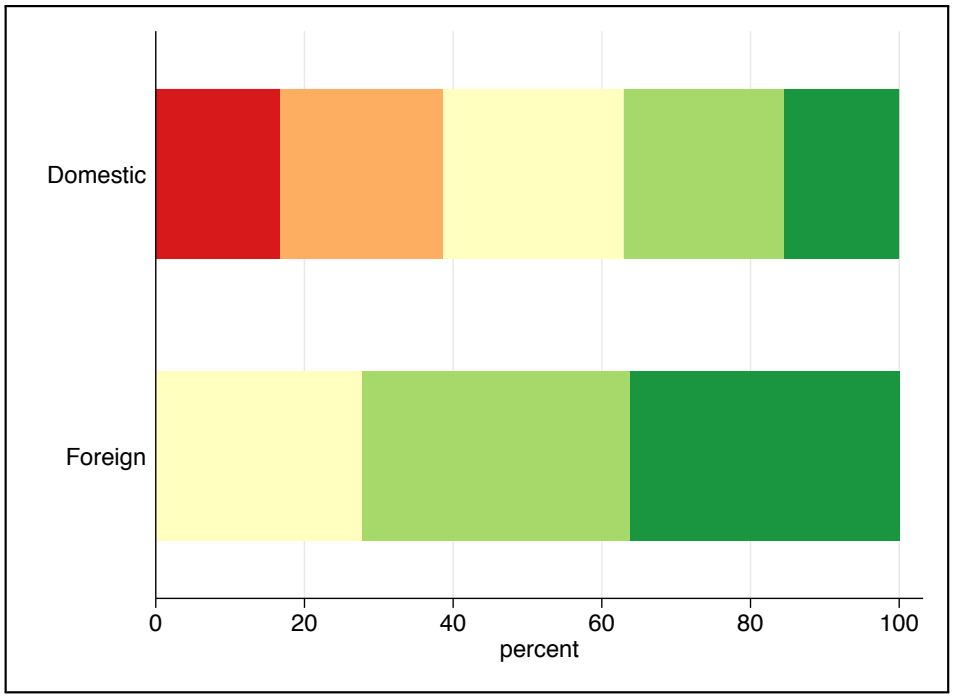


Colors

```
. grstyle init
. grstyle set imesh, horizontal compact minor
. grstyle set legend 4, nobox
. grstyle set color hue, n(5) opacity(50)
. grstyle set symbolsize large
. scatter price? weight, ytitle(Price)
```




```
. grstyle init
. grstyle set plain
. grstyle set color RdYlGn, n(5)
. graph hbar price?, stack percent ///
>     over(foreign) legend(off)
```

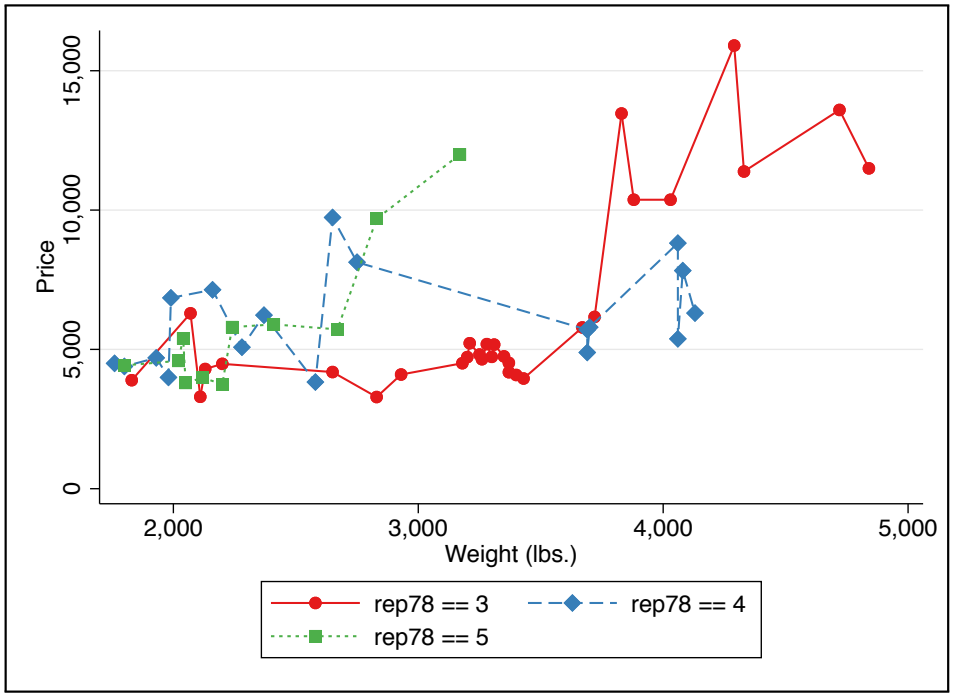


Colors

- 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

```
. grstyle init
. grstyle set plain
. grstyle set color Set1
. grstyle set symbol
. grstyle set lpattern
. twoway (connected price3-price5 weight, sort),
>     ytitle(Price)
```



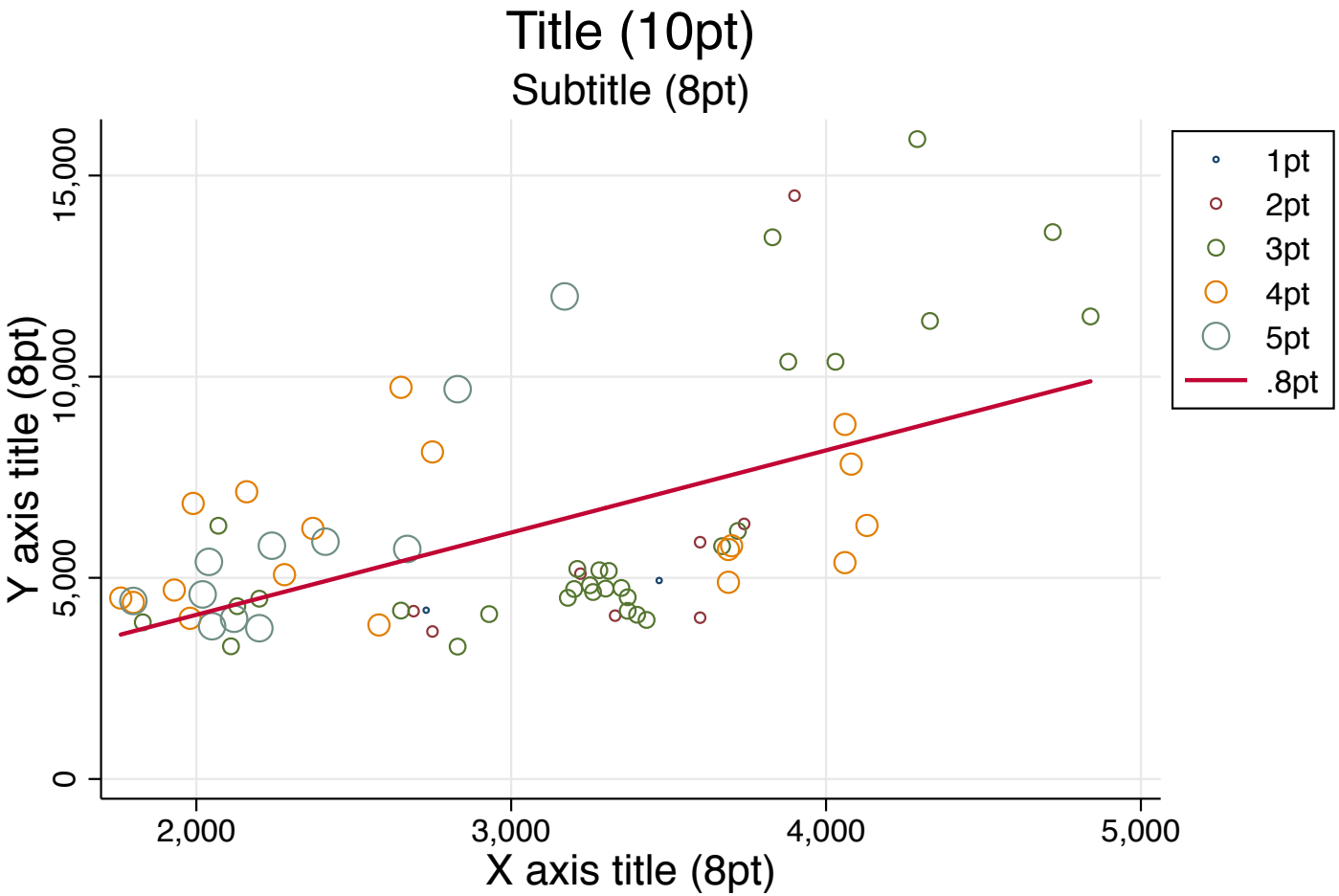
Sizes

- 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 legend 2
. grstyle set symbol Oh
. 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 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
. 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"))
```

Sizes


Title (10pt)
Subtitle (8pt)
Y axis title (8pt)
0
5,000
10,000
15,000
2,000
3,000
4,000
5,000
X axis title (8pt)
1pt
2pt
3pt
4pt
5pt
.8pt

Contents

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.
- `colorpalette` provides a versatile color management system.
- More information and examples:
  - http://repec.sowi.unibe.ch/stata/grstyle
  - http://repec.sowi.unibe.ch/stata/palettes

References

- Atz, U. 2011. SCHEME_TUFTE: Stata module to provide a Tufte-inspired graphics scheme. Available from https://ideas.repec.org/c/boc/bocode/s457285.html.
- Bischof, D. 2017a. G538SCHEMES: module to provide graphics schemes for http://fivethirtyeight.com. Available from http://ideas.repec.org/c/boc/bocode/s458404.html.
- Bischof, D. 2017b. New graphic schemes for Stata: plotplain and plottig. The Stata Journal 17(3): 748–759.
- Brewer, C. A., G. W. Hatchard, M. A. Harrower. 2003. ColorBrewer in Print: A Catalog of Color Schemes for Maps. Cartography and Geographic Information Science 30(1): 5–32.
- Briatte, F. 2013. SCHEME-BURD: Stata module to provide a ColorBrewer-inspired graphics scheme with qualitative and blue-to-red diverging colors. Available from http://ideas.repec.org/c/boc/bocode/s457623.html.
- Buchanan, B. 2015. BREWSCHEME: Stata module for generating customized graph scheme files. Available from http://ideas.repec.org/c/boc/bocode/s458050.html.

References

- Juul, S. 2003. Lean mainstream schemes for Stata 8 graphics. The Stata Journal 3(3): 295-301.
- Morris, T. 2013. SCHEME-MRC: Stata module to provide graphics scheme for UK Medical Research Council. Available from http://ideas.repec.org/c/boc/bocode/s457703.html.
- Morris, T. 2015. SCHEME-TFL: Stata module to provide graph scheme, based on Transport for London's corporate colour pallette. Available from http://ideas.repec.org/c/boc/bocode/s458103.html.
- Newson, R. 2005. SCHEME_RBN1MONO: Stata module to provide minimal monochrome graphics schemes. Statistical Software Components S456505, Boston College Department of Economics. Available from https://ideas.repec.org/c/boc/bocode/s456505.html.