A 2019 New Year's Resolution for Stata users: Make cleaner, prettier graphs

December 26, 2018

Nicholas T. Davis[1]

1 Introduction

Making graphs is the best part about being a social scientist.[2] The problem with graph-making, however, is that software interfaces are opaque. In the case of Stata, program .readme files, while often well-documented, usually lack informative notation regarding what the component syntax pieces actually do.[3] As a result, most people rely on the standard graphing defaults, which produce graphs that are not very aesthetically pleasing.

I spent much of 2018 trying to make better graphs. Because my sunk costs are in Stata, this short write-up is for people who have yet to make the pivot to R for philosophical or practical reasons. If you're an advanced Stata user, then perhaps this walkthrough isn't for you because you've already got this figured out. However, I have rarely seen Stata's graphing syntax annotated in ways that are useful to the non-expert, so perhaps this guide will be useful to a modest population of users nonetheless. In most cases, the syntax reviewed here is plug-and-play.

2 A simple blueprint for better fidelity

There are four things that one could do to make better graphs using Stata.

1. Download either plottig or plotplain by Dan Bischof. If .do files make you queasy, then these settings will at least clean up most of the clutter associated with the stock Stata settings. If you use these settings off-the-shelf, then cite the package. It's the least you can do.
2. Use `coefplot`. Read through Ben Jann's documentation of the coefplot package here. Coefplot is the singular flexible graphing utility that exists in the Stata workspace.[4] Ben also covers a lot of the different coding pieces that go into graphing. It's useful reading.
3. Understand Stata's general graphing syntax. See here for a good refresher.
4. Follow and iterate on the options listed in Section 4.

---

[1] Assistant Research Scientist, Public Policy Research Institute, Texas A&M University, nicholastdavis.com

[2] This has not been empirically verified.

[3] The exception to this is Ben Jann's documentation for `coefplot`, which is very, very clear.

[4] `Marginsplot` is fine, and I use it all the time. But, if you're presenting small multiples or multiple marginal effect estimates, then `coefplot` is often a more attractive route to pursue.

3 Off-the-shelf schemes

Let's begin with a simple example, which will require us to load a universally-available dataset into working memory. Type:

```
sysuse auto.dta
```

This syntax loads the ubiquitous "cars" dataset into Stata. Let's start with a basic description of the "length" variable, which is the length in inches of the cars in the dataset.

```
histogram weight
```

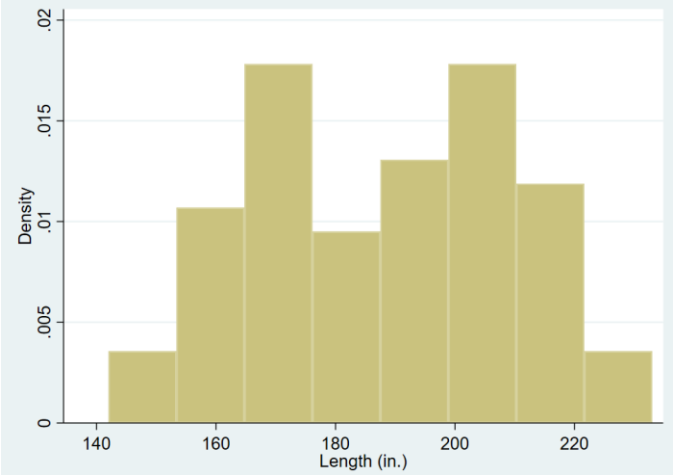

Figure 1. Distribution of car lengths using stock graphics package.

This is...not a pretty graph. Stata calls its default coloration, type, and scaling settings. The result is a visually poor blue-ish background, inelegant type, odd gridlines, and asymmetric scaling that doesn't look balanced.

We can do better simply by installing Dan's aforementioned "plottig" and "plotplain" schemes. You'll probably need to type:

```
findit plottig
```

Now, you'll want to select the "gr0070" package from the popup window and download it:

```
Web resources from Stata and other users

(contacting http://www.stata.com)

2 packages found (Stata Journal and STB listed first)
-----------------------------------------------------

gr0070 from http://www.stata-journal.com/software/sj17-3
    SJ17-3 gr0070. Provide graph schemes sensitive to color vision deficiency
    / Provide graph schemes sensitive to color vision / deficiency / by Daniel
    Bischof, Department of Political / Science, University of Zurich, Zurich,
    / Switzerland / Support:  bischof@ipz.uzh.ch / After installation, type
```

Next, you'll need to manually install the scheme's settings by typing:

```
set scheme plottig
```

Again, let's generate a histogram:

```
histogram length
```

The graphic below looks much better. It rids us of the blue background, replaces the color scheme with a nice monochromatic set of grays, and offers a sensible set of grid lines. If you only installed this scheme and went about your way, then you'd be better off than the modal graph.

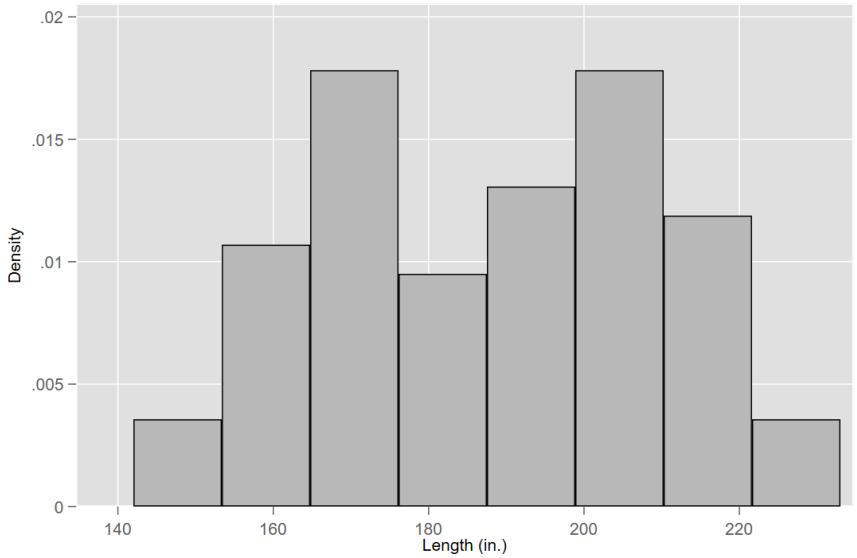

Figure 2. Distribution of car lengths using "plottig" scheme.

However, to be honest, the gray background is a little bit dark, and the gridlines add clutter to the visual presentation of the data. Instead, let's try the "plotplain" scheme. It should have been automatically included when you downloaded the earlier package. This time type:

```
set scheme plotplain
```

The graphic below replaces the gray background and exchanges it for a white one. It's a bit better, in my opinion, although I confess I don't love the faint dotted gridlines.

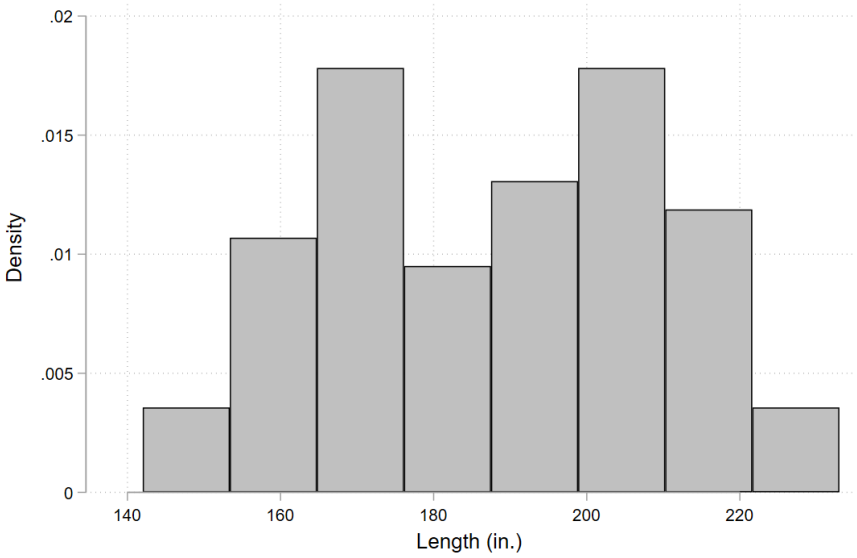

Figure 3. Distribution of car lengths using "plotplain" scheme.

In fact, while both of these examples are superior to Stata's default settings, there are still three issues that an off-the-shelf scheme won't address. First, the settings don't address blocky and inelegant font. Second, scaling issues persist – there is more white space on the eastern portion of the x-axis than there is on the western one, which means that the labeling comes across as asymmetrically centered. Finally, as we move into increasingly complex visualizations, other problems related to legends, plot colors, confidence intervals, etc. will manifest that must be addressed manually.

4 The magic is after the comma

Let's work on writing some code to generate a much cleaner visual presentation of the hard work you put in collecting and coding your data. In my opinion, there are two things that ensure a better graphic: clean, simple font and axes that are scaled proportionately.

Changing the font is easy. Simply generate a new histogram and in the editor window, you'll click on "Edit" and then "Preferences." From there, all you need to do is establish what font you want. Your options are endless, but try and avoid "serif" fonts because they look clunky and don't render as well. "Roboto light" looks good. So, too, does "Montserrat." Your machine likely doesn't have either font natively, but both fonts are available as true type font (.ttf) files online. Use google to search for them and install them after downloading them.

After settling on your font, we can get down to working on the magic that occurs *after the comma*. In other words:

```
histogram length, all of the good stuff happens in here
```

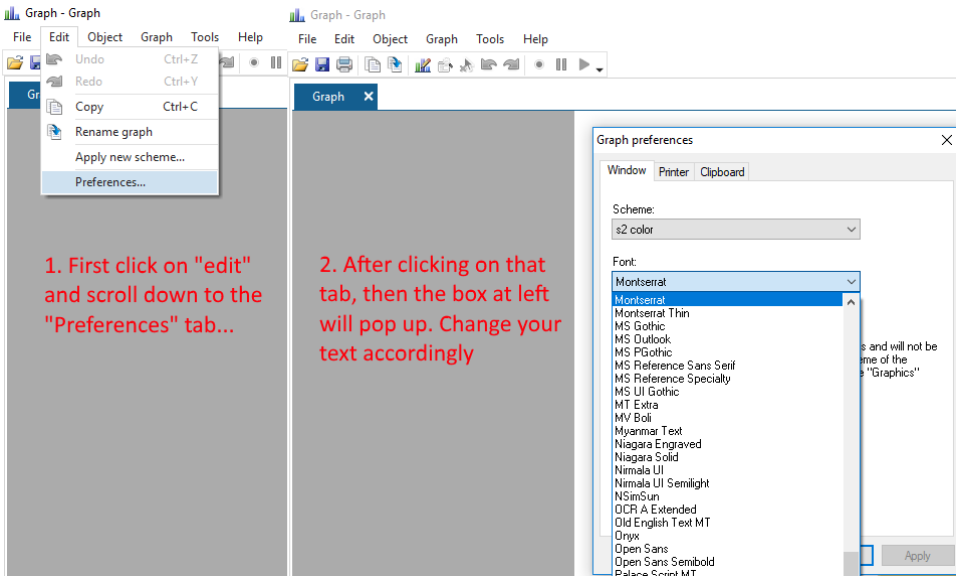

Figure 4. Changing font within the Stata graph editor

4.1 Scaling

Stata refuses to scale axes in ways that make any aesthetic sense. It basically just takes the minimum and maximum values across the range of your variables or estimates and plots those. This leaves an ungainly mess of an image because it messes with whitespace and leaves your labeling somewhat up to chance as it relates to the actual range of values.

Instead, let's try and reclaim the axes to work for us. The following command imposes the full range of values for which the x-axis should take: 130 to 245 inches. Doing this expands the white space on the right side of the graph so that it is roughly proportional to the left side. Further, the code also bumps up the y-axis values so that the base of the histogram doesn't overlap with the x-axis. Finally, the fourth line of code specifies the white background and adds a thin border around the pane.

Important note: the "///" tell Stata to execute the commands on that line and to then move to the next line of code. Stata terminates on the final line of code that does *not* have those dashes.

```
histogram length,                                   /// base graphing command
       xscale(r(130 245) lcolor())                  /// x-axis scaling
       yscale(r(-.001  .025) lcolor())              /// y-axis scaling
       plotregion(fcolor(white) lcolor(gs10))
```

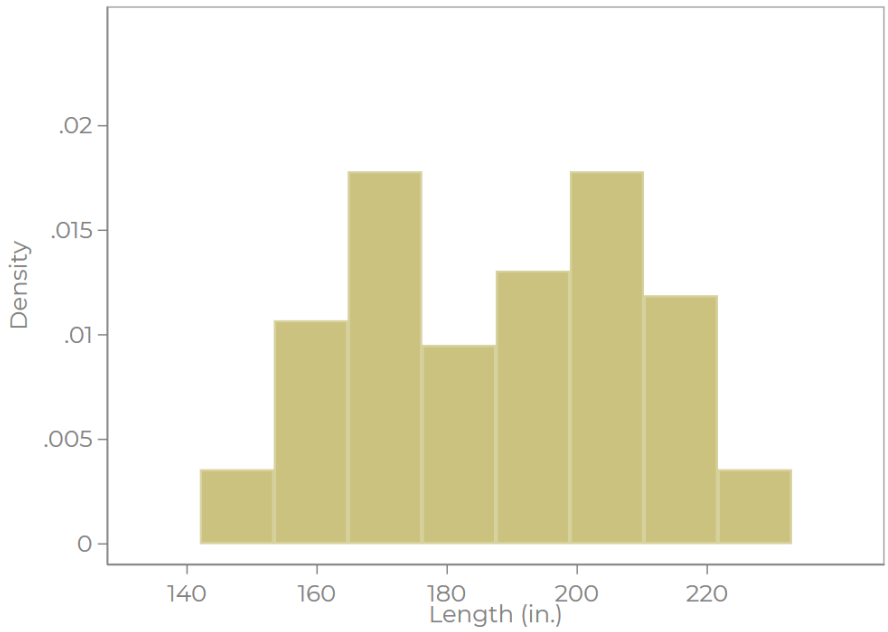

Figure 5. Distribution of car lengths with improved axis spacing.

The problem with this graph, however, is that the labelling is all screwed up. The decimals on the y-axis are not uniform, and the length on the x-axis doesn't cover the full span of the histogram bars. Further, the beige coloration is awful, and the axis labels are too hard to see. We can do better. Let's improve on this code above.

4.2 Manual labeling

In addition to the code above, we need to do a better job explicitly setting the coloration parameters. We also need to add in some standardized labelling. Building on the above code, we add in the literal values that we want the x- and y-axes to take, along with specifying their coloration. We also darken the labels and set the bar color to black. The code to generate this is below.[5]

```
histogram length,                                                                    /// base graphing command
    xscale(r(130 245) lcolor())                                                      /// x-axis scaling
    yscale(r(-.001  .02) lcolor())                                                   /// y-axis scaling
    plotregion(fcolor(white) lcolor(gs10) lwidth(med))                               /// background coloration and outline
    xlabel(135 145 155 165 175 185 195 205 215 225 235 245,                          /// x-axis label #s
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)                     /// x-axis labels coloration
    ylabel(0.000 "0.000" 0.005 "0.005" 0.010 "0.010" 0.015 "0.015" 0.020 "0.020",   /// y-axis label #s
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)                     /// y-axis labels coloration
    xtitle("Length (inches)", color(black))                                          /// x-axis title
    ytitle("Density", color(black))                                                  /// y-axis title
    bcolor(black)                                                                    /// setting bar color
    title(" ")
```

[5] Section 6 reproduces all of this code in text format so you can copy and paste it into your own do files.

That code produces the following graph:

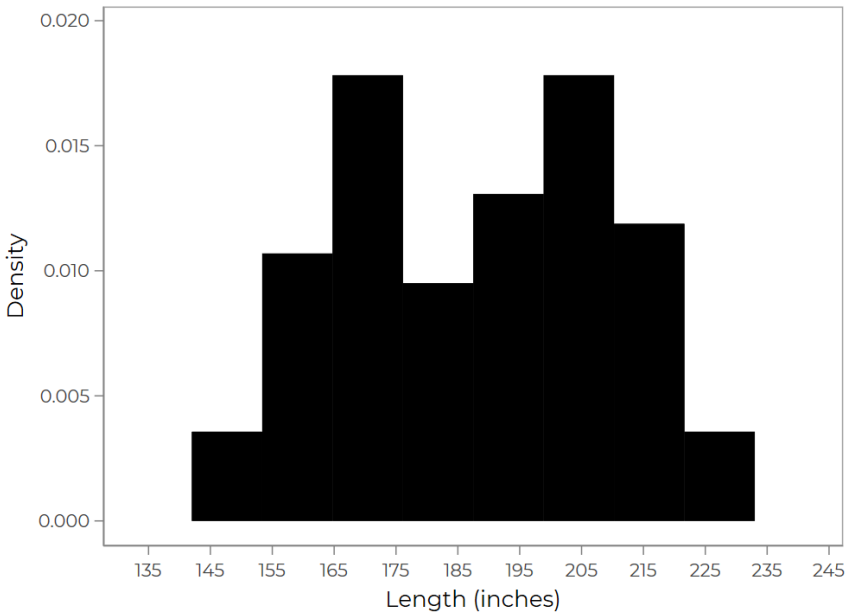

Figure 6. Distribution of car lengths with improved scaling and coloration

If you wanted white bars with black outlines, for example, then you would have changed the coloration options to: `bcolor(white) bl(black)`. That would give you a super minimalist graphic as follows:

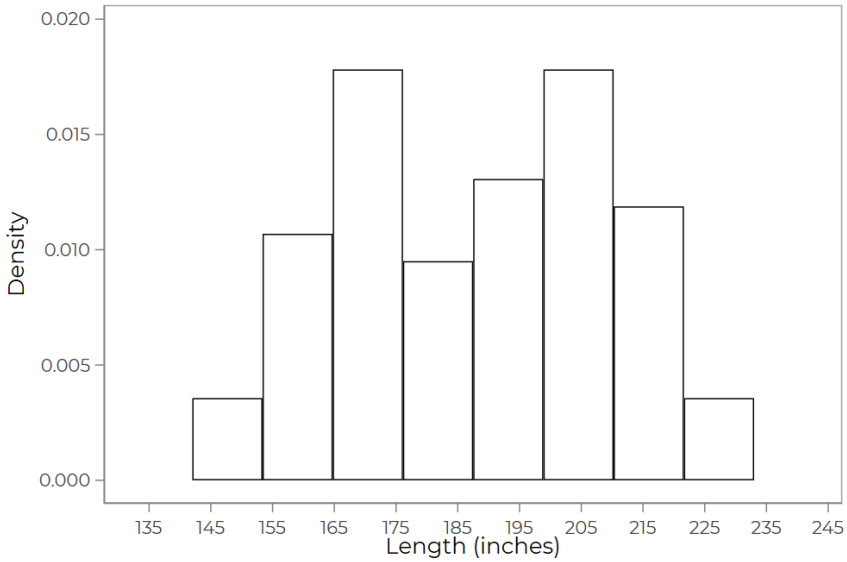

Figure 7. Distribution of car lengths with improved axis spacing and minimalist coloration

4.3 Basic annotated code

These changes are not difficult to implement. The following code can be added to most any graphing syntax. It tinkers with the graph space, itself, setting background coloration, x- and y-axis scaling, labels, and the like. It's flexible and can be added to graphs built with either `marginsplot` or `coefplot`. (Note: the "///" informs Stata to keep reading the code on the next line; on the terminal line of code, you should *not* include "///.")

```
* annotated basic code that can be universally applied:

xscale(r(# #) lcolor(color))                                    /// x-axis scaling
yscale(r(# #) lcolor(color))                                    /// y-axis scaling
plotregion(fcolor(color) lcolor(color) lwidth(__))              /// background coloration and outline
xlabel(# "label" # "label",                                     /// x-axis label #s
labcolor(color) tlcolor(color) tlwidth(__) labsize(__) nogrid) /// x-axis labels coloration
ylabel((# "label" # "label",                                    /// y-axis label #s
labcolor(color) tlcolor(color) tlwidth(__) labsize(__) nogrid) /// y-axis labels coloration
xtitle(" ", color(color))                                       /// x-axis title
ytitle(" ", color(color))                                       /// y-axis title
bcolor(color) blcolor(color)                                    /// setting bar color
title(" ")


* where: # # = a range of numbers from smaller to larger, # = whatever number(s) you choose
(often this will be a range of numbers associated with your quantities of interest), color =
whatever color you choose, and ___ = width or size
```

However, while this syntax is useful, to manipulate post-estimation plots requires a more involved set of commands. For this, we'll need to include some more code.

4.4 A more involved example using `marginsplot`

Taking the above logic, let's work on producing an analysis that illustrates the marginal effect of car length on car price that accounts for the distribution of real car lengths in the dataset. In addition, let's layer a histogram of car lengths onto our predicted marginal effects of length on price. Here, we'll begin with a simple bivariate regression

```
reg price length
```

Next, we'll explore how different lengths are related to price. Let's explore intervals of 10 inches.

```
quietly margins, at(length=(150(10)230))
```

And, finally, we'll call `marginsplot` to create our graph. `Marginsplot` will use the Stata defaults that we decided were ugly, so we'll need to add the above code, plot and confidence interval syntax, and the `addplot` feature to create a pretty nice looking graphic. The full syntax looks like this:[6]

[6] See page 14 for syntax in text format.

```
*approximating a rugplot
reg price length
margins, at(length=(150(10)230))
marginsplot,
    level(95)
    plotregion(fcolor(white) lcolor(gs10) lwidth(med))
    xlabel(,
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)
    ylabel(4000 "$4,000" 6000 "$6,000" 8000 "$8,000" 10000 "$10,000",
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)
    xtitle("Length (inches)", color(black))
    ytitle("Car price ($)", color(black))
    recastci(rspike)
    ciopts(lcolor(black))
    plotlopts(mcolor(white) mlcolor(black) lcolor(black))
    title(" ")
        addplot(hist length,
        blcolor(black) fcolor(black)
        discrete
        yaxis(2)
        yscale(alt lcolor() axis(2))
        ylabel(0 " " 20 " " 40 " " 60 " " 80 " " 100 " ",
        labcolor() axis(2) tlcolor(black) tlwidth(thin) labsize(small) tl(0))
        ytitle(" ", axis(2))
        legend(order( 2 "Predicted price" 3 "Car lengths")))
```

```
/// base graphing command
/// confidence intervals
/// background coloration and outline
/// x-axis label #s
/// x-axis labels coloration
/// y-axis label #s
/// y-axis labels coloration
/// x-axis title
/// y-axis title
/// confidence interval display options
/// confidence interval color
/// plot colors
/// master graph title
/// adding and overlaying distribution of length
/// bar line and fill colors
/// tells stata to graph actual distribution of length
/// calls for 2nd y-axis, which we'll suppress
/// 2nd y-axis scaling
/// minimizing the height of histogram
/// label options for 2nd axis, removes ticks / labels
/// no title on 2nd axis
```

That code creates a graphic that superimposes a very rough "rug plot" beneath the estimates. Stata includes no native rug plot feature, so you need to get a little creative by suppressing the "height" of the histogram. This is done by making sure the second y-axis is scaled using large numbers and removing the ticks from the second axis. It is important to note that the "axis(2)" option is vital because otherwise the `addplot` would overwrite the information on the first y-axis, which conveys prices.

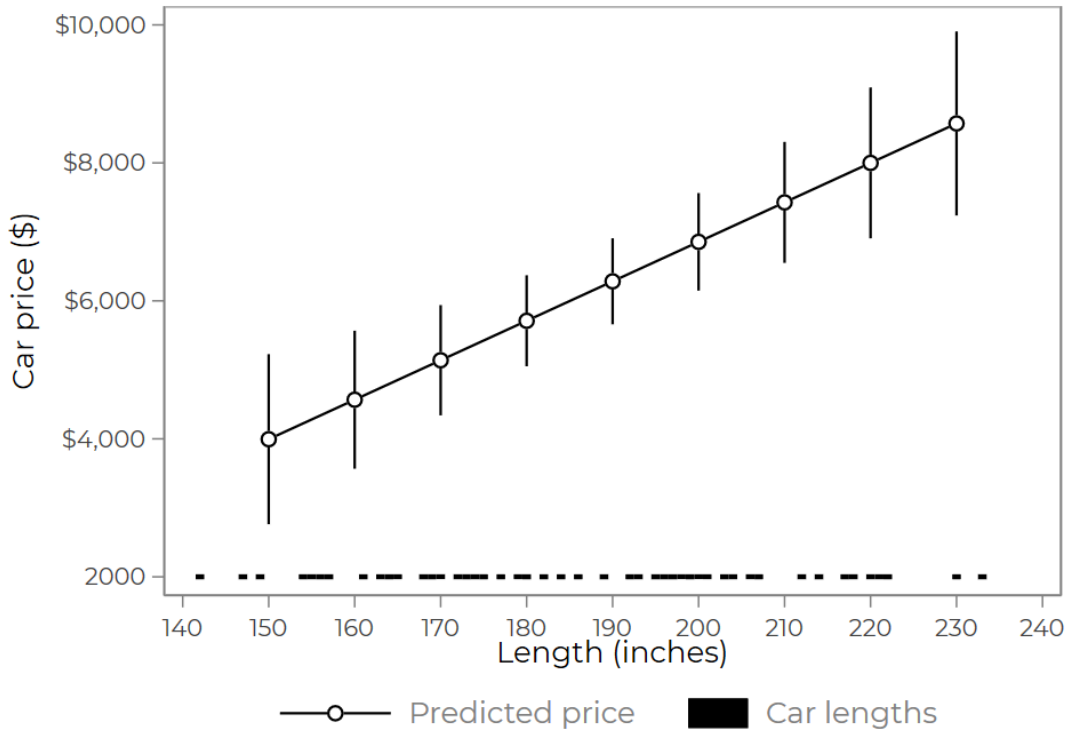

Figure 8. The effect of car length on car prices with rug plot.

If we instead want a more traditional histogram superimposed onto the graphic, then we simply need to re-arrange some of the options. The code below gets us a little closer to that common graphic:[7]

```
*overlaying histogram
reg price length
margins, at(length=(150(10)230))
marginsplot,                                                    /// base graphing command
    level(95)                                                   /// confidence intervals
    plotregion(fcolor(white) lcolor(gs10) lwidth(med))          /// background coloration and outline
    xlabel(,                                                    /// x-axis label #s
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid) /// x-axis labels coloration
    ylabel(4000 "$4,000" 6000 "$6,000" 8000 "$8,000" 10000 "$10,000", /// y-axis label #s
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid) /// y-axis labels coloration
    xtitle("Length (inches)", color(black))                     /// x-axis title
    ytitle("Car price ($)", color(black))                       /// y-axis title
    recastci(rspike)                                            /// confidence interval display options
    ciopts(lcolor(black))                                       /// confidence interval color
    plotopts(mcolor(white) mlcolor(black) lcolor(black))        /// plot colors
    title(" ")                                                  /// master graph title
        addplot(hist length,                                    /// adding histogram marginsplot graph
        blcolor(gs10) fcolor(gs10)                              /// bar line and fill colors
        percent                                                 /// histogram bins in "percent" rather than "discrete" (actual)
        yaxis(2)                                                /// calls for 2nd y-axis
        yscale(alt lcolor() axis(2))                            /// scaling on 2nd y-axis
        ylabel(0 "0%" 20 "20%" 40 "40%" 60 " " 80 " " 100 " ",  /// labels on 2nd y-axis
        labcolor() axis(2) tlcolor(black) tlwidth(thin) labsize(small)) /// label options on 2nd y-axis
        ytitle(" ", axis(2))                                    /// telling Stata to remove title on 2nd y-axis
        legend(order( 2 "Predicted price" 3 "Distribution of car lengths")))
```

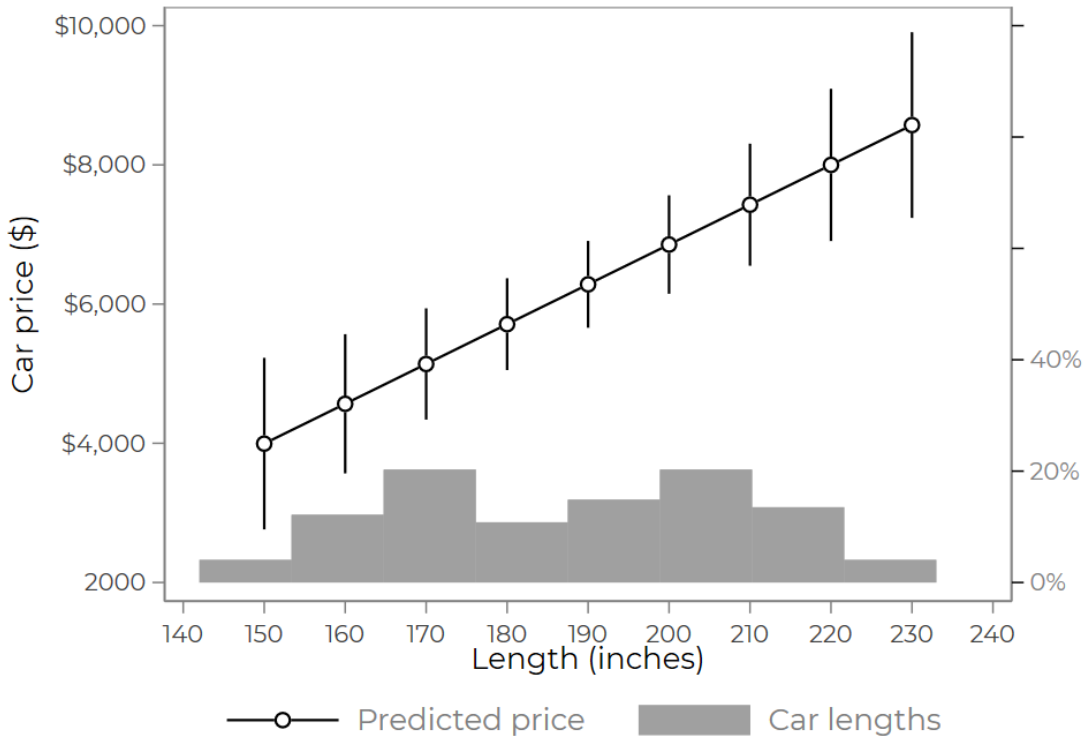

Figure 9. The effect of car length on car prices with histogram of car lengths

[7] See page 15 for syntax in text format.

4.5 An example using "coefplot"

While marginsplot is extremely useful, it is less flexible than coefplot, which allows us to plot multiple post-estimates. We can tailor the above code to work with the coefplot commands, which are related, though subtlety different. Here, we will run a multiple regression of different car features and then plot the marginal effect of those items on car price.[8]

```
reg price length01 weight01 mpg01 displacement01 foreign01
est store price
coefplot price,                                                    /// coefplot package
    drop(_cons)                                                    /// drop the constant, coefplot will always auto-plot
    level(95)                                                      /// confidence interval level
    plotregion(fcolor(white) lcolor(gs10) lwidth(med))             /// background coloration and outline
    xlabel(-10000 "-$10,000" 0 "$0" 10000 "$10,000" 20000 "$20,000", /// x-axis label #s
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)   /// x-axis labels coloration
    ylabel(,                                                       /// y-axis label #s
    labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)   /// y-axis labels coloration
    xtitle("Marginal effect, min-to-max", color(black))            /// x-axis title
    ytitle(, color(black))                                         /// y-axis title
    title(" ")                                                     /// graph title " " = not title
    xline(0, lp(dash) lcolor(red))                                 /// red dashed line at 0
    mcolor(white) mlcolor(black)                                   /// point estimate color options
    ciopts(lcolor(black))                                          /// CI color options
    coeflabels( length01 = "Length" weight01 = "Weight" mpg01 = "MPG" displacement01 = "Displacement" foreign01 = "Foreign") /// re-labeling
    legend(off)
```

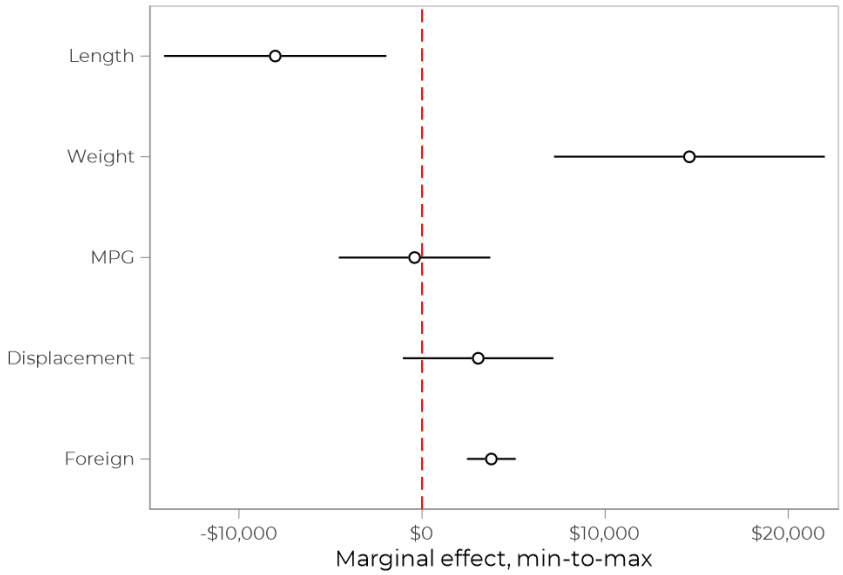

Figure 10. Marginal effect of car-related features on car price. Estimates convey effect of moving from minimum to maximum values on respective covariate. Equation models car price as function of these five items; it is a silly model.

[8] See page 16 for syntax in text format.

5 Conclusion

Constructing better graphs is not terribly difficult. Dan Bischof and Ben Jann have made this pretty easy. If you're a Luddite, then use an off-the-shelf package like plottig or plotplain to sex up your graphs. With minimal tinkering of the Stata defaults, however, you can take advantage of the flexibility of both marginsplot and coefplot to make more compelling, clearer graphs. And once you have your style template nailed down, then it is very easy to replicate across projects.

Go forth.

Make 2019 the "Year of the Beautiful Stata Graph."

6 Code for figures

```
clear

sysuse auto

* Figure 7
* example histogram
histogram length,                                                       /// base graphing command
       xscale(r(130 245) lcolor())                                      /// x-axis scaling
       yscale(r(-.001  .02) lcolor())                                   /// y-axis scaling
       plotregion(fcolor(white) lcolor(gs10) lwidth(med))               /// background coloration and outline
       xlabel(135 145 155 165 175 185 195 205 215 225 235 245,          /// x-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)     /// x-axis labels coloration
       ylabel(0.000 "0.000" 0.005 "0.005" 0.010 "0.010" 0.015 "0.015" 0.020 "0.020",   /// y-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)     /// y-axis labels coloration
       xtitle("Length (inches)", color(black))                          /// x-axis title
       ytitle("Density", color(black))                                  /// y-axis title
       bcolor(white) bl(black)                                          /// setting bar color
       title(" ")
```

```
*Figure 8
*approximating a rugplot
reg price length
margins, at(length=(150(10)230))
marginsplot,                                                    /// base graphing command
       level(95)                                                /// confidence intervals
       plotregion(fcolor(white) lcolor(gs10) lwidth(med))       /// background coloration and outline
       xlabel(,                                                 /// x-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)       /// x-axis labels coloration
       ylabel(4000 "$4,000" 6000 "$6,000" 8000 "$8,000" 10000 "$10,000",  /// y-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)       /// y-axis labels coloration
       xtitle("Length (inches)", color(black))                          /// x-axis title
       ytitle("Car price ($)", color(black))                            /// y-axis title
       recastci(rspike)                                                 /// confidence interval display options
       ci1opts(lcolor(black))                                           /// confidence interval color
       plot1opts(mcolor(white) mlcolor(black) lcolor(black))            /// plot colors
       title(" ")                                               /// master graph title
              addplot(hist length,                              /// adding and overlaying distribution of length
              blcolor(black) fcolor(black)                      /// bar line and fill colors
              discrete                                          /// tells stata to graph actual distribution of length
              yaxis(2)                                          /// calls for 2nd y-axis, which we'll suppress
              yscale(alt lcolor() axis(2))                      /// 2nd y-axis scaling
              ylabel(0 " " 20 " " 40 " " 60 " " 80 " " 100 " ",           /// minimizing the height of histogram
              labcolor() axis(2) tlcolor(black) tlwidth(thin) labsize(small) tl(0))    /// label options for 2nd axis
              ytitle(" ", axis(2))                                                     /// no title on 2nd axis
              legend(order( 2 "Predicted price" 3 "Car lengths")))
```

```
*Figure 9
*overlaying histogram
reg price length
margins, at(length=(150(10)230))
marginsplot,                                                    /// base graphing command
       level(95)                                                /// confidence intervals
       plotregion(fcolor(white) lcolor(gs10) lwidth(med))       /// background coloration and outline
       xlabel(,                                                 /// x-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)       /// x-axis labels coloration
       ylabel(4000 "$4,000" 6000 "$6,000" 8000 "$8,000" 10000 "$10,000",  /// y-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)       /// y-axis labels coloration
       xtitle("Length (inches)", color(black))                            /// x-axis title
       ytitle("Car price ($)", color(black))                        /// y-axis title
       recastci(rspike)                                             /// confidence interval display options
       ci1opts(lcolor(black))                                       /// confidence interval color
       plot1opts(mcolor(white) mlcolor(black) lcolor(black))        /// plot colors
       title(" ")                                                   /// master graph title
              addplot(hist length,                      /// adding histogram marginsplot graph
              blcolor(gs10) fcolor(gs10)                /// bar line and fill colors
              percent                                   /// histogram bins in "percent" rather than "discrete" (actual)
              yaxis(2)                                  /// calls for 2nd y-axis
              yscale(alt lcolor() axis(2))              /// scaling on 2nd y-axis
              ylabel(0 "0%" 20 "20%" 40 "40%" 60 " " 80 " " 100 " ",            /// labels on 2nd y-axis
              labcolor() axis(2) tlcolor(black) tlwidth(thin) labsize(small))   /// label options on 2nd y-axis
              ytitle(" ", axis(2))                                  /// telling Stata to remove title on 2nd y-axis
              legend(order( 2 "Predicted price" 3 "Car lengths")))
```

```
* Figure 10
* coefplot example
* first recode vars to 0 to 1 to make plots clean
foreach x in length weight mpg displacement foreign {
sum `x'
gen `x'01 = (`x'-r(min))/(r(max)-r(min))
}

reg price length01 weight01 mpg01 displacement01 foreign01
est store price
coefplot price,                                          /// coefplot package
       drop(_cons)                                       /// drop the constant, coefplot will always auto-plot
       level(95)                                         /// confidence interval level
       plotregion(fcolor(white) lcolor(gs10) lwidth(med))              /// background coloration and outline
       xlabel(-10000 "-$10,000" 0 "$0" 10000 "$10,000" 20000 "$20,000", /// x-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)    /// x-axis labels coloration
       ylabel(,                                          /// y-axis label #s
       labcolor(gs4) tlcolor() tlwidth(thin) labsize(small) nogrid)    /// y-axis labels coloration
       xtitle("Marginal effect, min-to-max", color(black))             /// x-axis title
       ytitle(, color(black))                                          /// y-axis title
       title(" ")                                                      /// graph title " " = not title
       xline(0, lp(dash) lcolor(red))                                  /// red dashed line at 0
       mcolor(white) mlcolor(black)                                    /// point estimate color options
       ciopts(lcolor(black))                                           /// CI color options
       coeflabels( length01 = "Length" weight01 = "Weight" mpg01 = "MPG" displacement01 = "Displacement" foreign01 =
       "Foreign")                                                      /// re-labeling
       legend(off)
```