Lecture5_ggplot

Kelly and Gallego

```
library(knitr)
options(width=60)
require(knitr)
# Set so that long lines in R will be wrapped:
opts_chunk$set(tidy.opts=list(width.cutoff=80),tidy=TRUE)
```

Ah, the swirling ocean of code that shows up in the wake of a popular, free, and open-source project like **R**. So many people write so many packages for so many things. There are packages for analyzing baseball data, for example, or tennis data, for that matter. Calculating carbonate chemistry, waveform analysis, organizing musical chord charts, exploring census data, and surely a thousand other things.

But making beautiful plots is something that **R** does well, and Hadley Wickham's package **ggplot2** is probably the best out there for doing it. You're already aware that this package exists, but today we're going to take it for a spin.

There's a whole theory of graphics underlying **ggplot2**, and you can learn a lot more about it here. But let's start with the helpfile for the package.

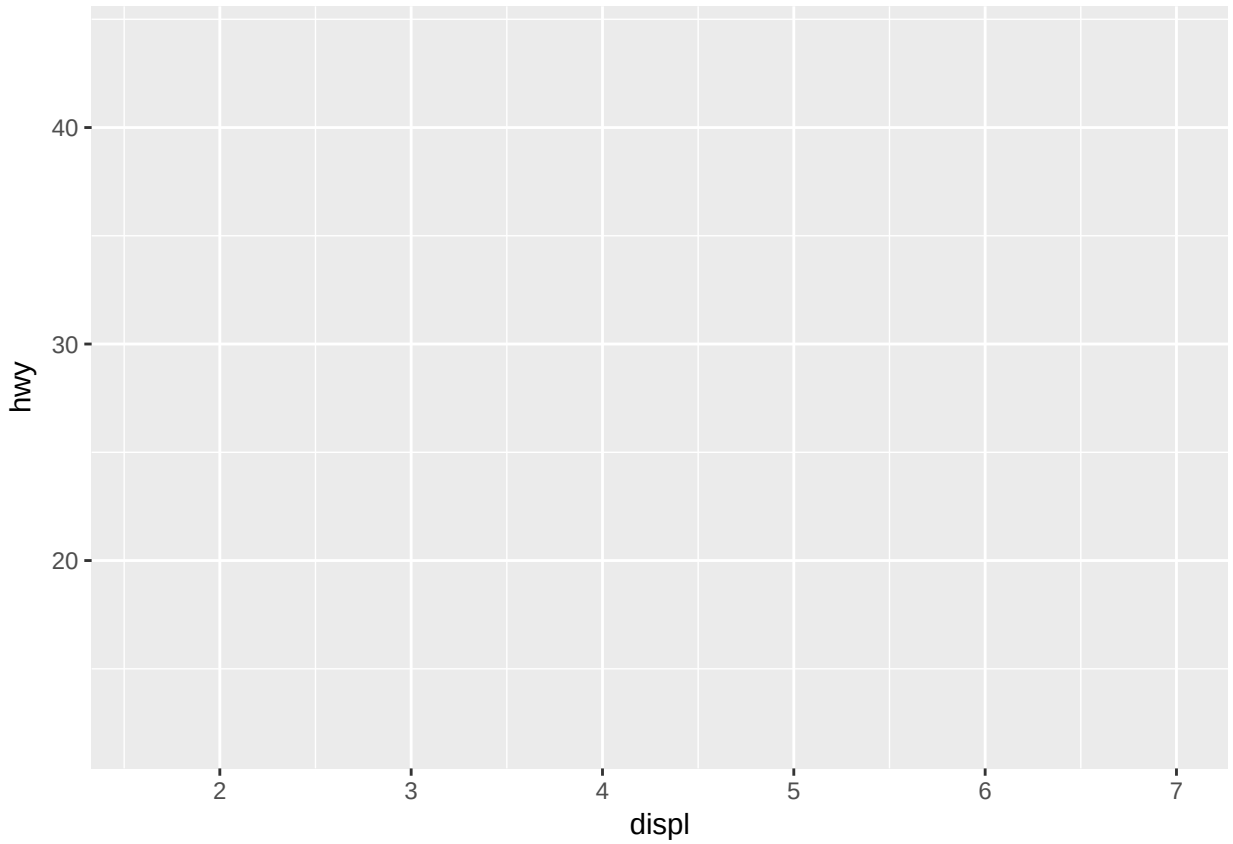
```
library(ggplot2)

`?`(ggplot)
```

The most confusing thing about **ggplot2** is that it doesn't use exactly the same syntax as the rest of **R**, for reasons that we won't go into (and that I don't really understand).

Essentially, though, you use it to first create a canvas `ggplot()`, and then you add elements to that canvas, piecemeal. You use the `+` symbol to tell the package that you are adding another element to the plot.

```
ggplot()  #just gives you the canvas
```

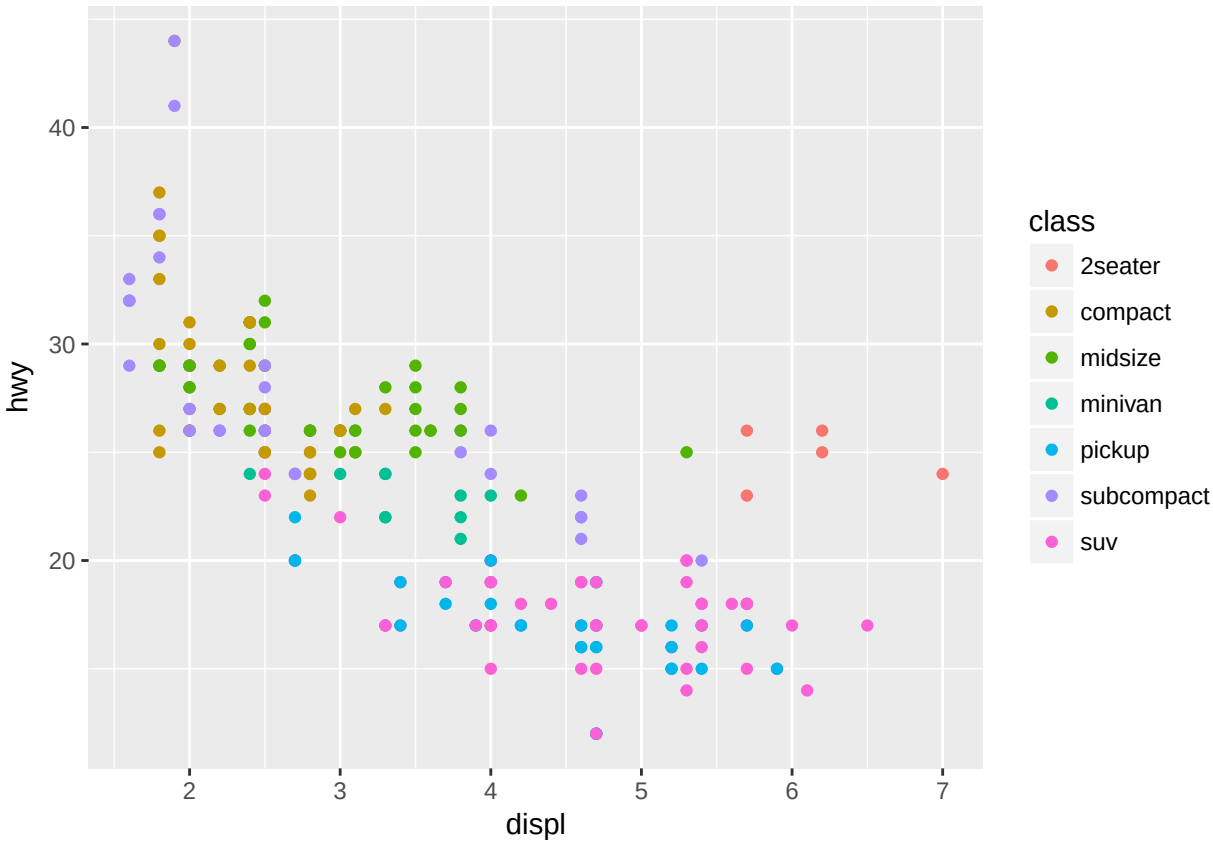
```r
ggplot(data = mpg,   #here, we're using a built-in dataset featuring info on different cars
  aes(x = displ, y = hwy , color = class) #the "aes" argument adds aesthetic properties to your plot. Y
  )
```



OK – so now we've told the package what we would like to plot (that is, what dataframe the data are coming from (mpg), the x and y variables, and that we want to vary the plot by color according to the class of the car being described.) But why is nothing plotted?

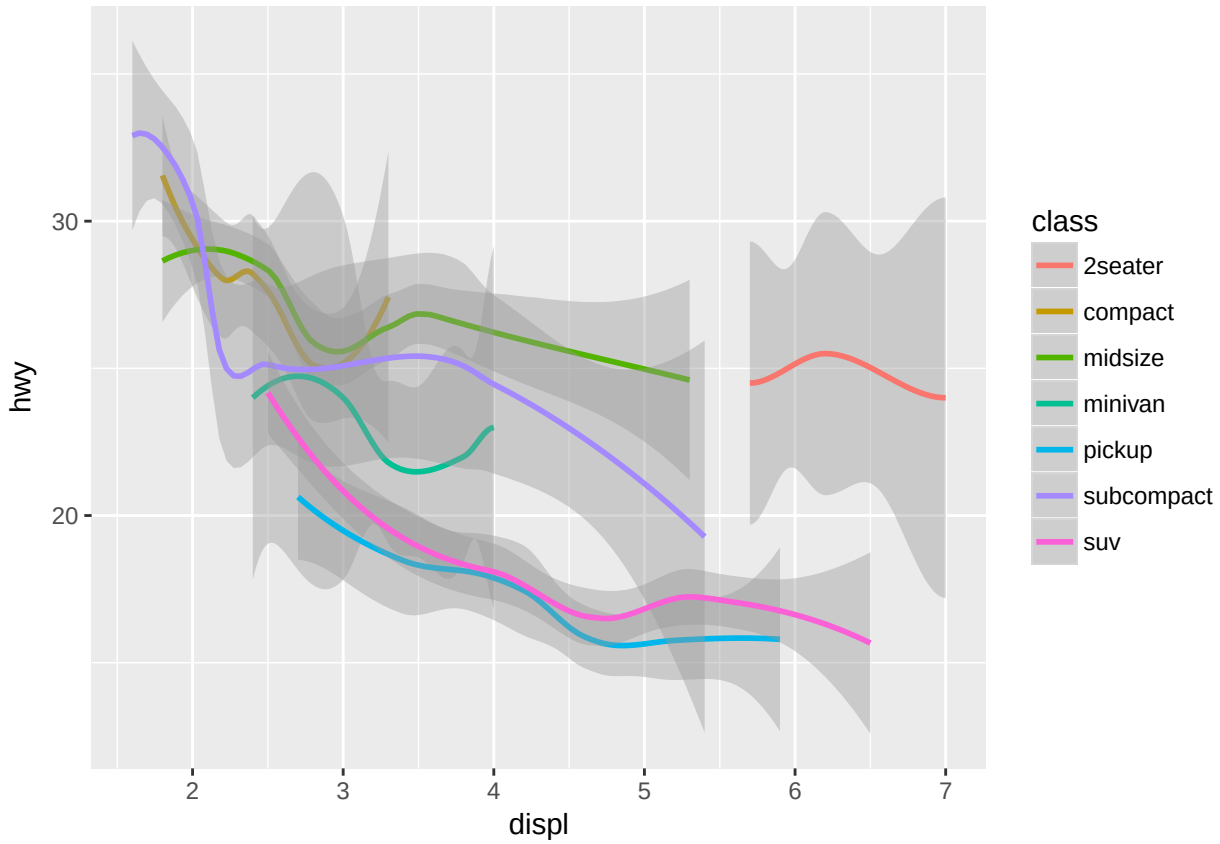
Because **ggplot2** is very needy. You actually need to tell it you want to plot POINTS, in this case. You do this with another layer of the graphic, called a *geom* (which is just basically the kind of plot you want... scatterplot, barplot, etc)

```
ggplot(data = mpg,
  aes(x = displ, y = hwy , color = class)
  ) +  #same code as above, but now with...
  geom_point()  #to tell ggplot that you want to plot points
```

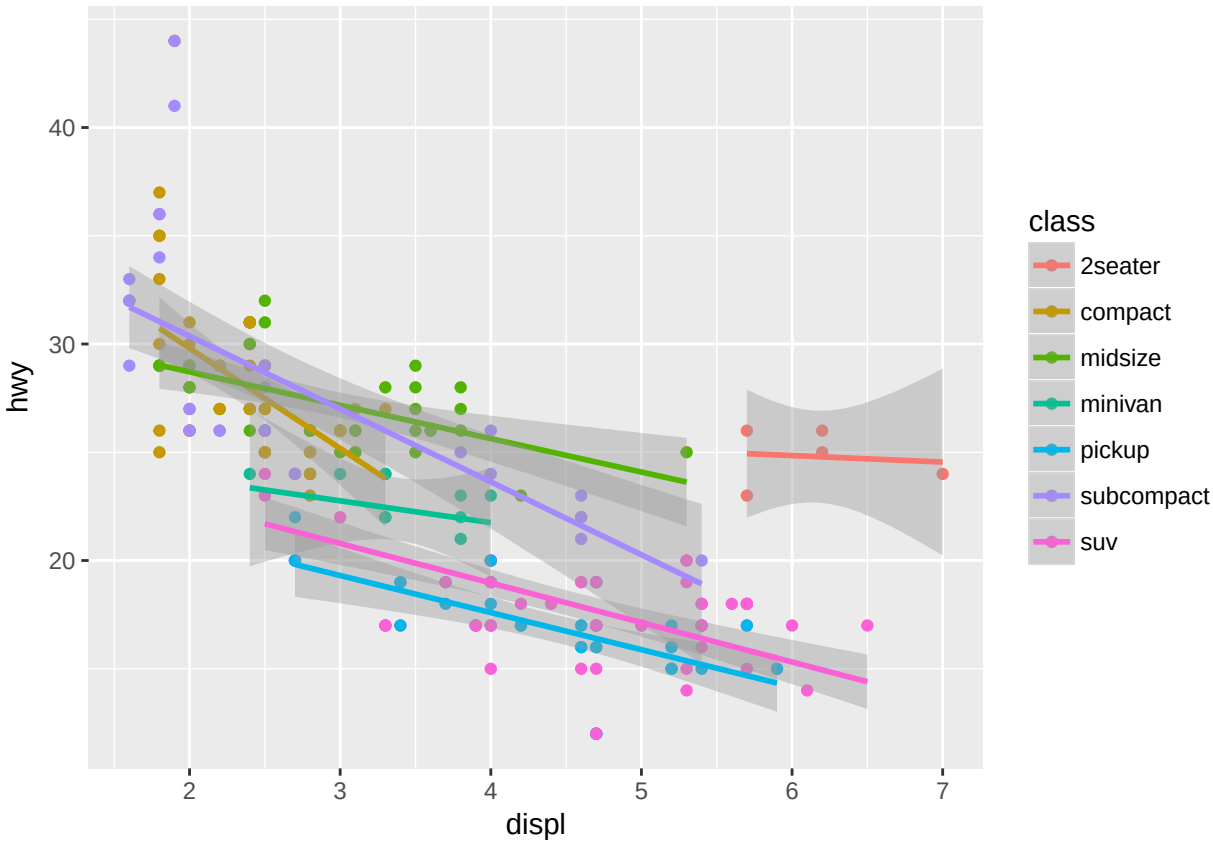


Now, isn't that nice? And in fact, we can make lots of different plots using similar code.

```
baseinfo <- ggplot(data = mpg, aes(x = displ, y = hwy, color = class))  #I'll save all of this as an ob

baseinfo + geom_smooth()  #for example, to plot smoothed lines and some semblance of confidence interva
```

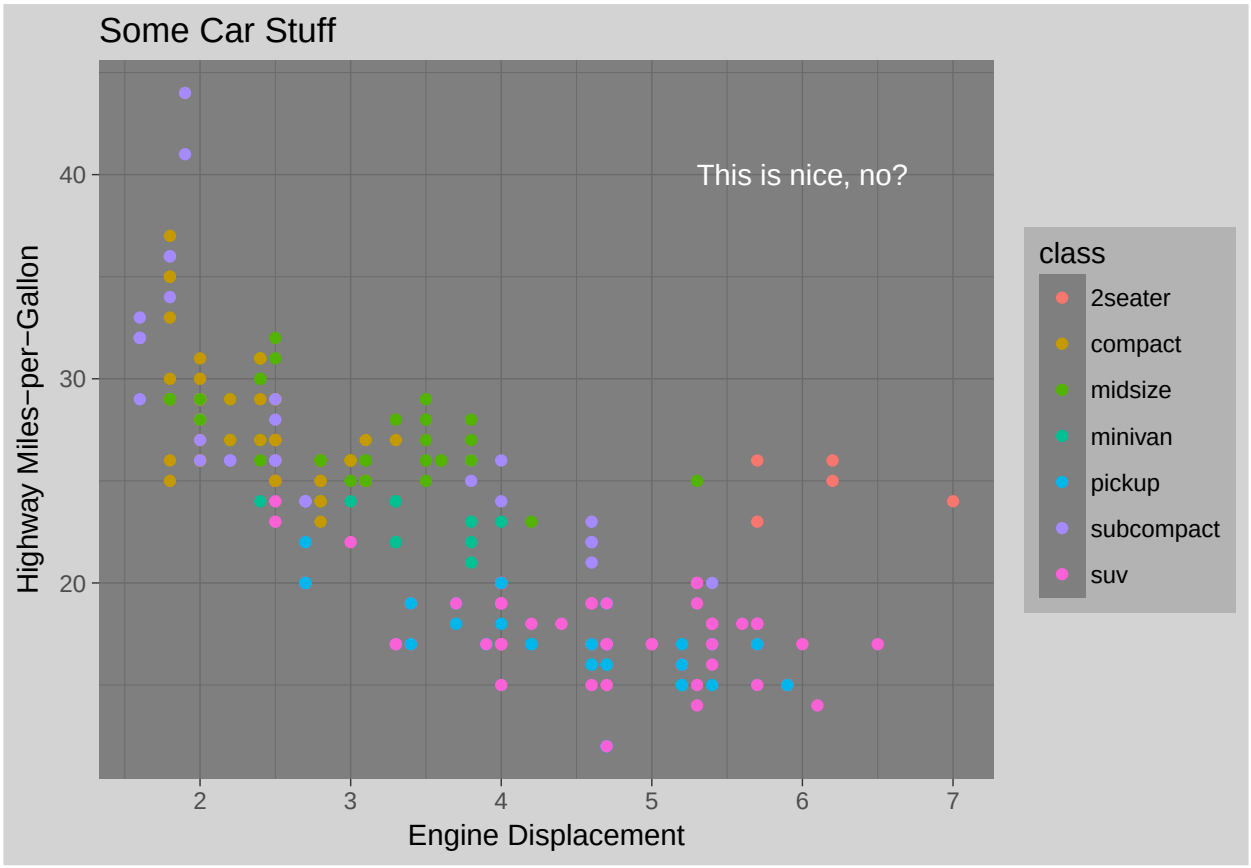



```
baseinfo + geom_point() + geom_smooth(method = "lm")  #or, for example, to plot both the points and a l
```



We add things like titles and text and axis labels to the plot in the same way.

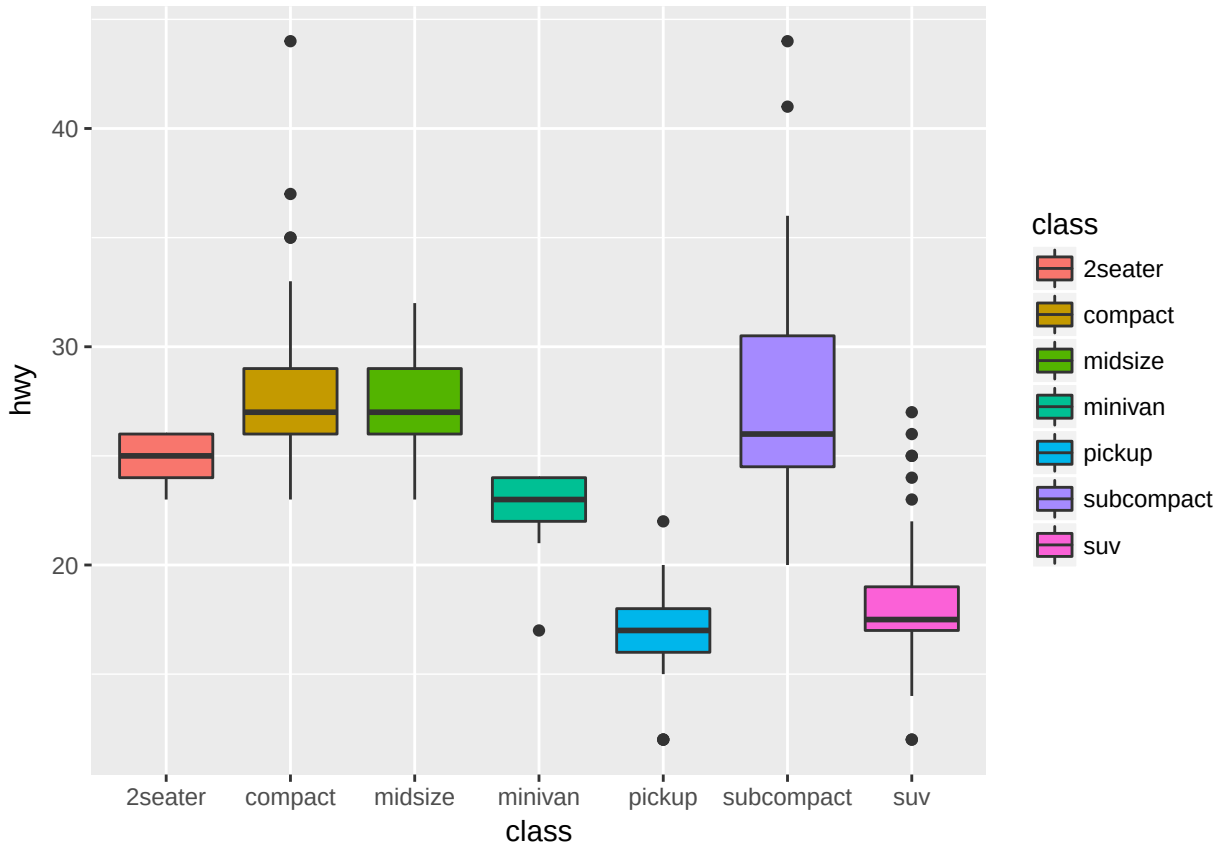
```
baseinfo + geom_point() +
  ggtitle("Some Car Stuff") + labs(x = "Engine Displacement", y = "Highway Miles-per-Gallon") +
  annotate("text", x = 6, y = 40, label ="This is nice, no?", color = "white") +
  theme_dark() + #one of several pre-set themes; you can change everything about the plot, such as the
  theme(plot.background = element_rect(fill = "lightgrey"),
        legend.background = element_rect(fill = "grey70"))
```



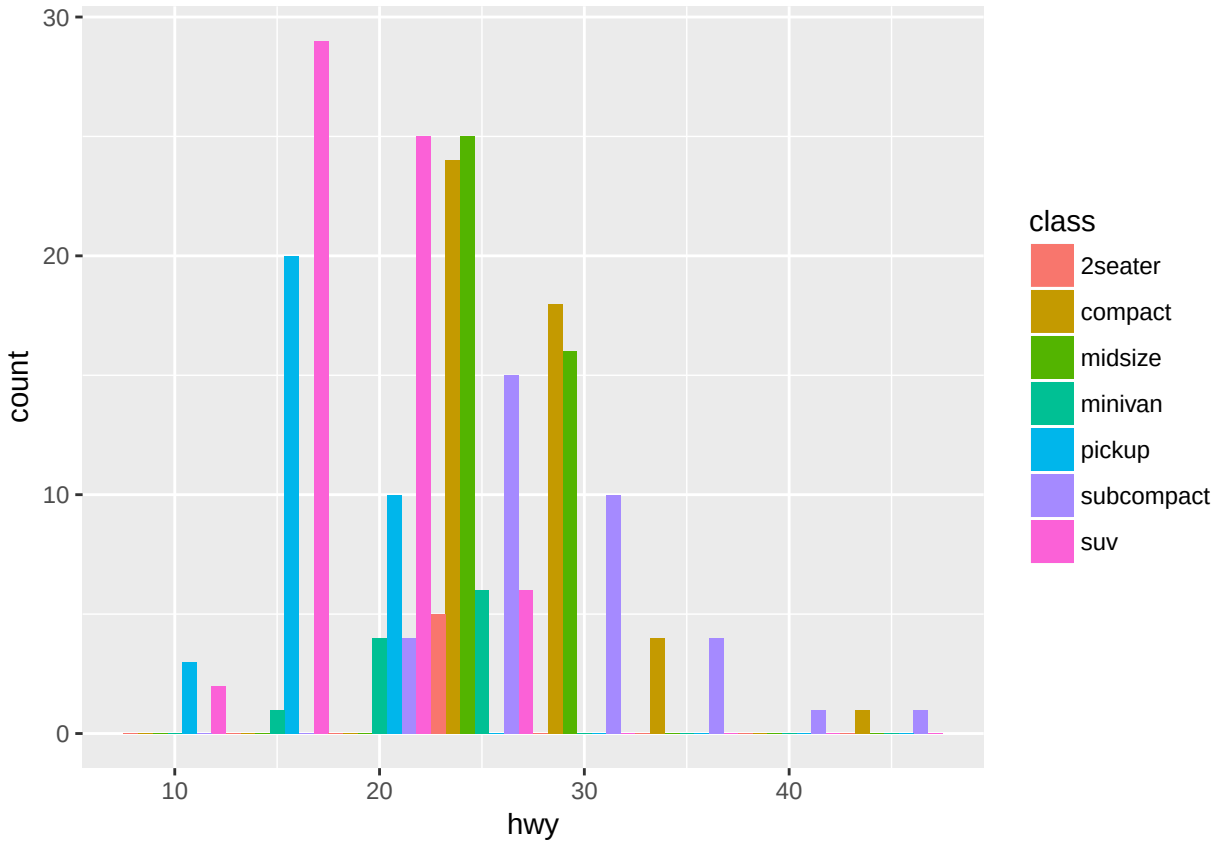
So you can see the appeal. Once you get the hang of it, you can make all kinds of very nice plots.

Some further examples:

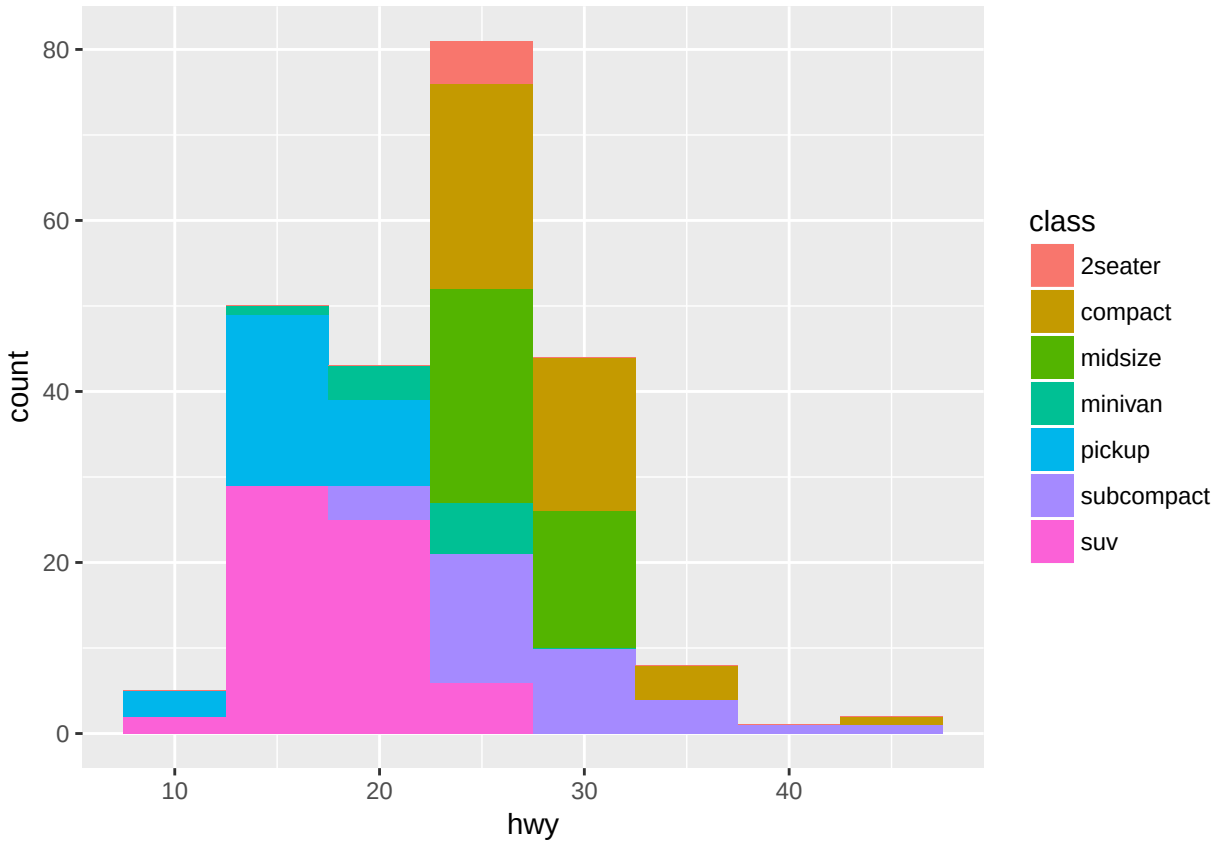
```
ggplot(mpg, aes(class, hwy, fill = class)) + geom_boxplot()  #Boxplot
```



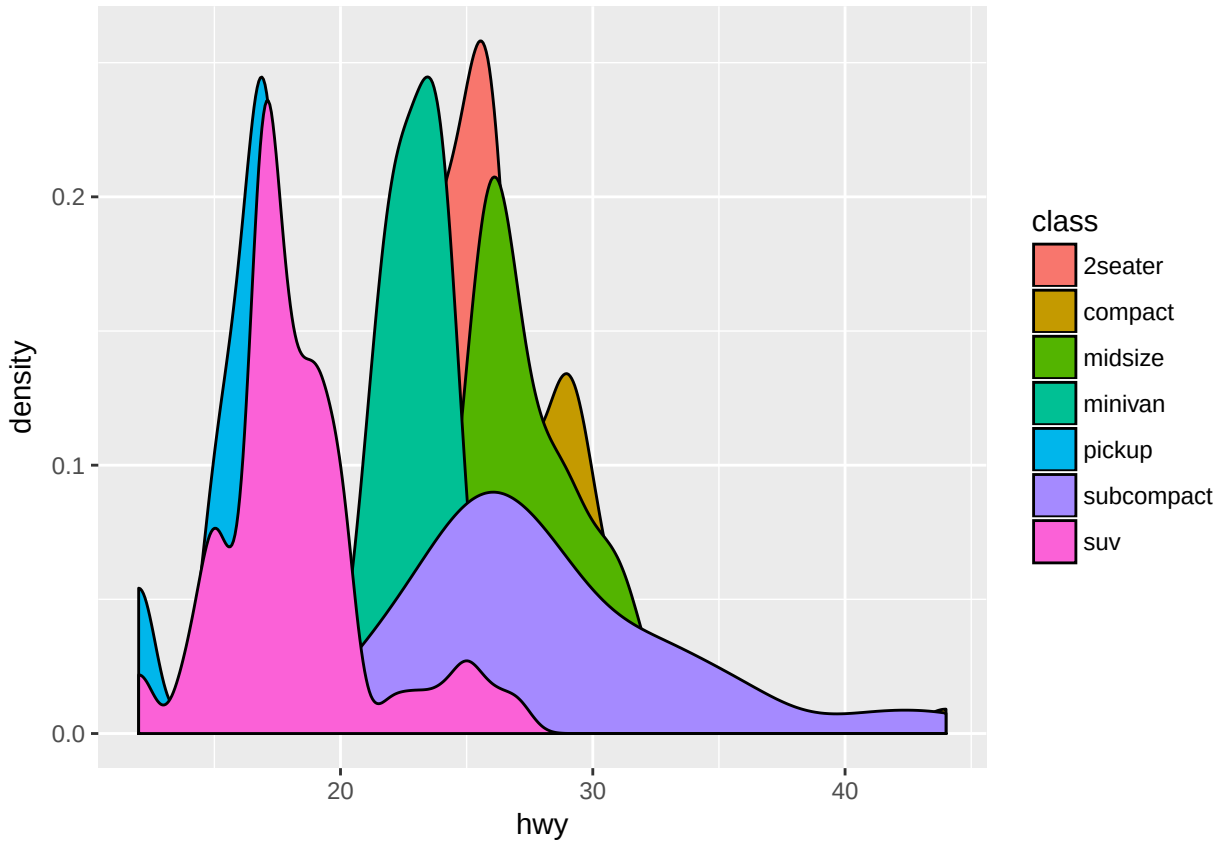

```
ggplot(mpg, aes(hwy, fill = class)) + geom_histogram(binwidth = 5, position = "dodge")  #Side-by-side h
```




```
ggplot(mpg, aes(hwy, fill = class)) + geom_histogram(binwidth = 5)  #stacked histogram
```




```
ggplot(mpg, aes(hwy, fill = class)) + geom_density()  #density plot
```

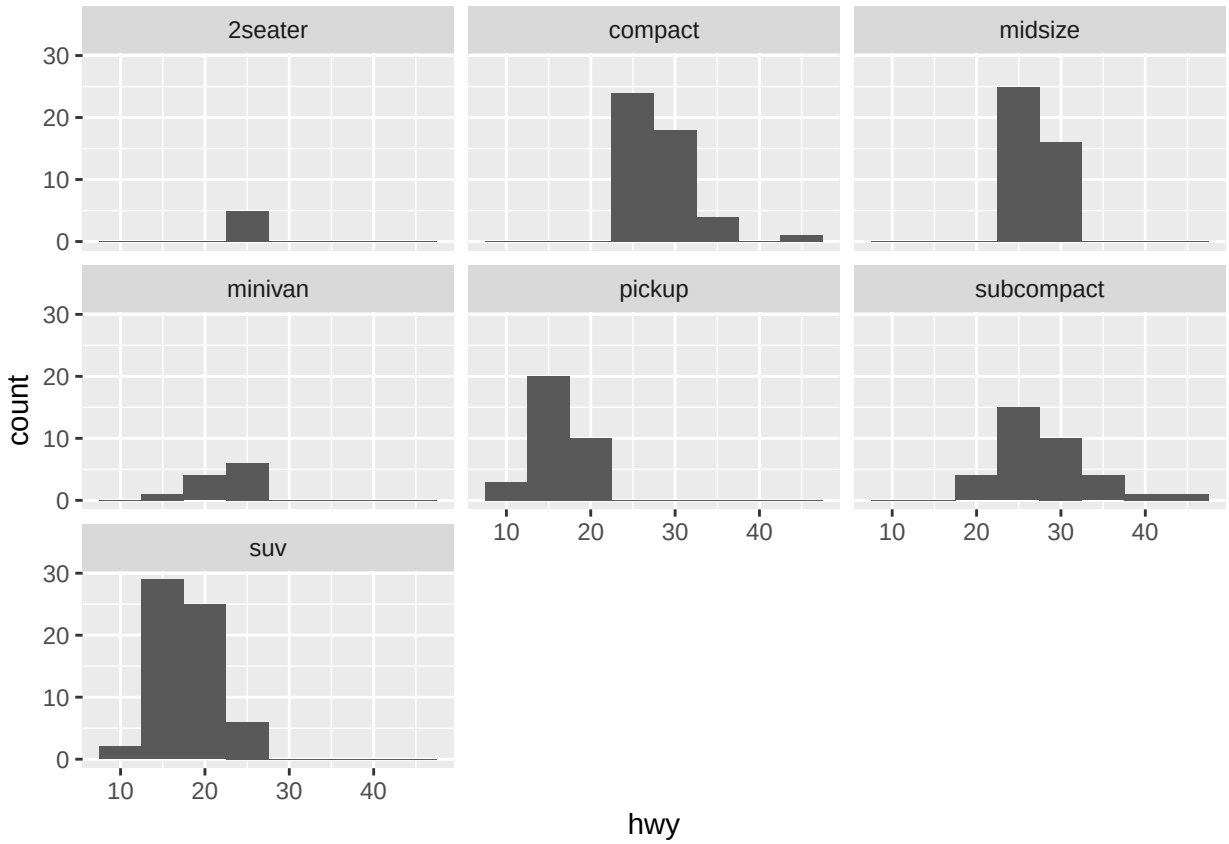


Facets

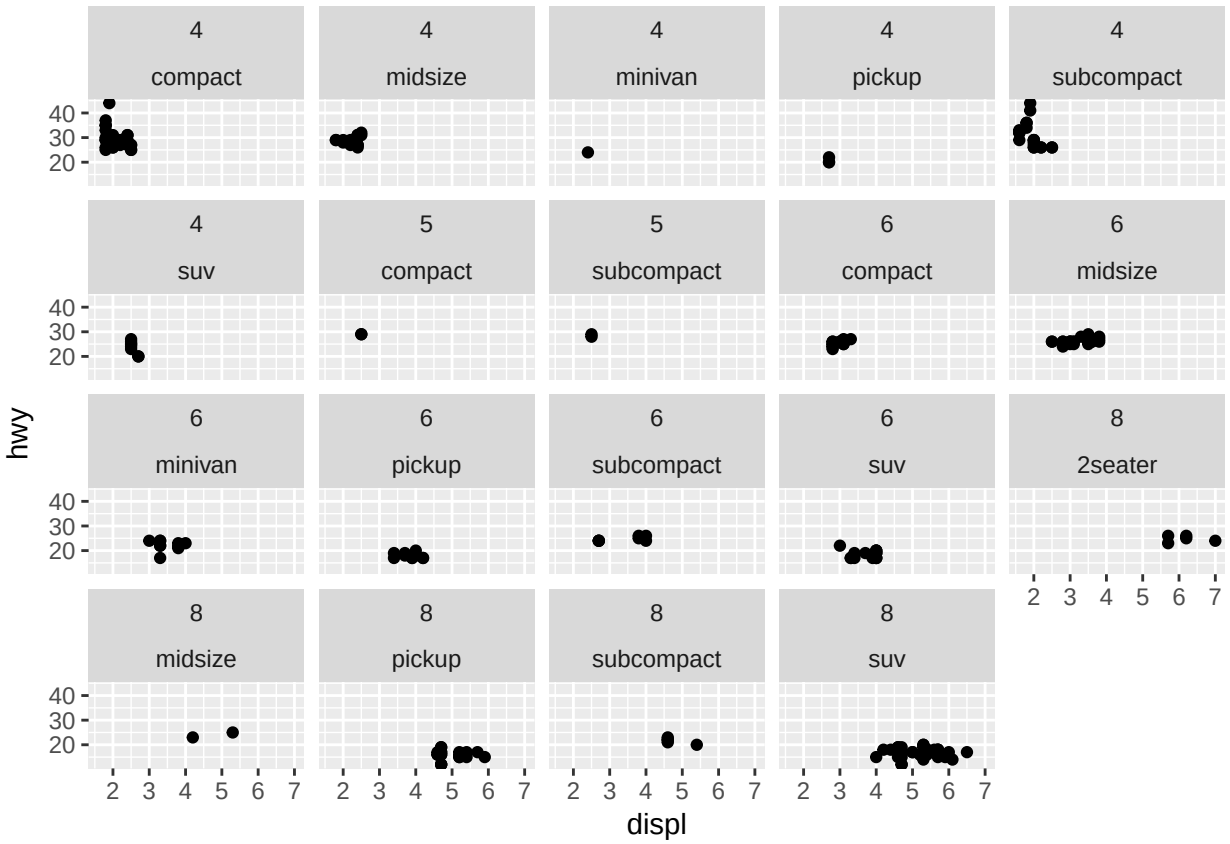
And perhaps the most useful thing about **ggplot**... you can create subplots for different subsets of your data. Which is very often something you want to do. Those subplots are, for some reason, called *facets*.

Note how convenient this is: you don't have to split your data into subsets, because **ggplot** does it for you.

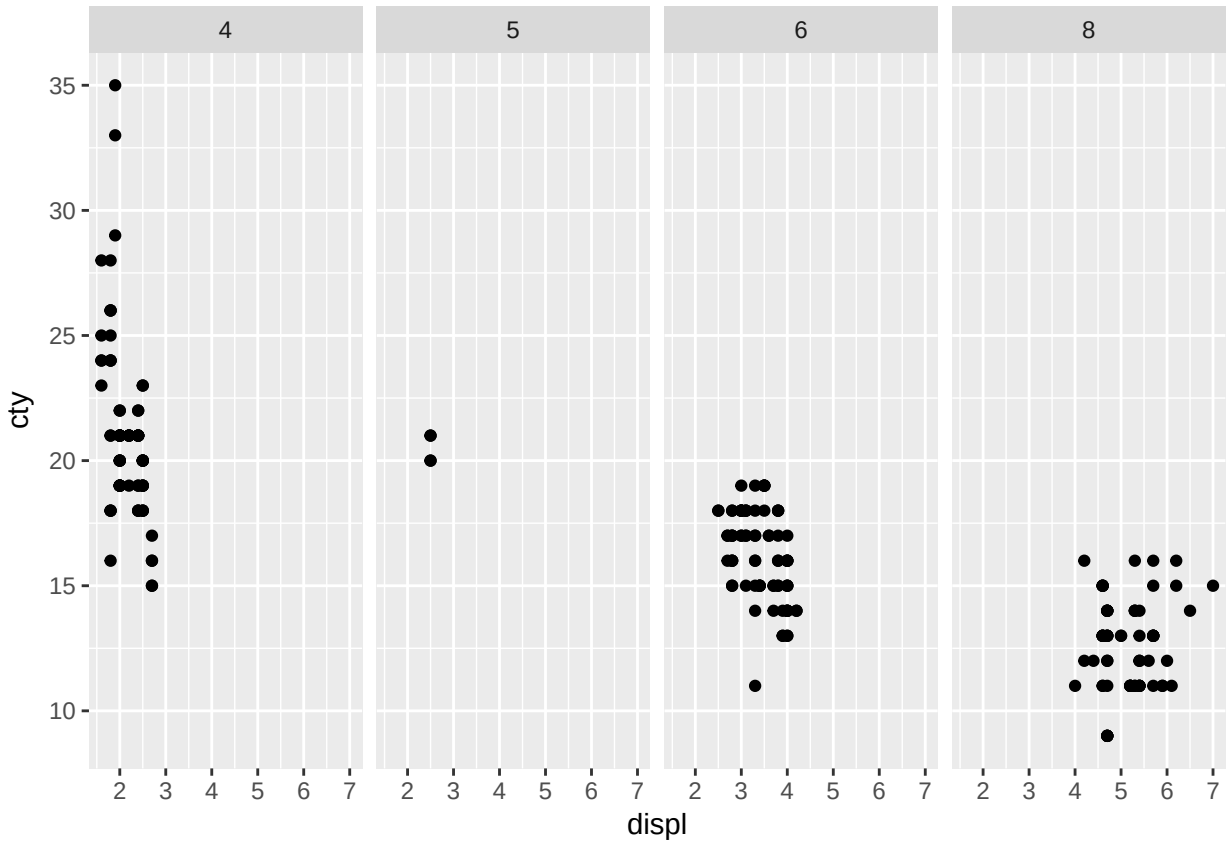
```
ggplot(mpg, aes(hwy)) + geom_histogram(binwidth = 5) + facet_wrap(~class)
```




```
# and you can even facet_wrap by multiple variables:

ggplot(mpg, aes(displ, hwy)) + geom_point() + facet_wrap(~cyl + class)
```




```
# facet_grid is a similar idea
ggplot(mpg, aes(displ, cty)) + geom_point() + facet_grid(. ~ cyl)
```


4
5
6
8
35
30
25
20
15
10
cty
2
3
4
5
6
7
displ