Post #1

Armaan Kohli

10/22/2017

Stat 133 Post Number 1

Data Visualization with ggplot

For those that work inside the field of statistics, data visualization is typically the end result of a large data analysis project. We go through the whole process of importing, tidying, and data manipulation so that we have data simple enough that it can be represented by any type of chart. By doing so, we are able to communicate our discoveries to those that may be interested in the topic we are exploring, but don't have a strong grasp in data analysis. In R specifically, an especially useful package, ggplot2, provides us with many of the tools we need for data visualization. We will explore this package and its tools in this post. ggplot2 is particularly useful, according to Hadley Wickham in his paper, because it allows us to get the visualizations that we want without having to "iterate between modeling, transforming, and visualizing."" So that you can also try some of the features I will be talking about in this post, I will use a dataset that R provides for us.

If you have not installed ggplot2, first follow these steps so you can access these datasets as well as utilize ggplot2.

```
#install.packages('ggplot2')
library(ggplot2)
```

Basic Structure of ggplot2

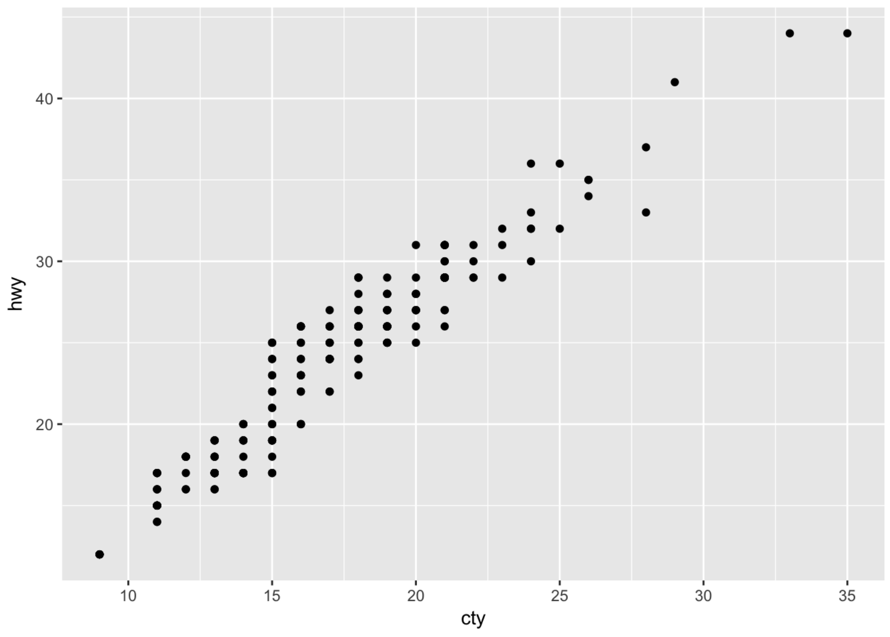
All graphs created by ggplot2 follow a very similar structure. In "R for Data Science," Hadley Wickham and Garrett Grolemund give us a good format to use when we are making our graphs: ggplot(data =) + (mapping = aes()) where we replace the bracketed items with whatever is necessary. Of course, there are more parameters for each and more information we can add, but this will suffice for now.

Let's look at some examples of simple plots we can make with the provided datasets from ggplot2. But to do that, we need to first look at this dataset that we can work with. So in the console, let's look at this dataset. We will call View() on our dataset, so you can look at it in the source pane instead of in your console.

```
#View(mpg)
```

Here is a simple example of a scatterplot that we can make with ggplot2. We will plot the area of a county against the total population.

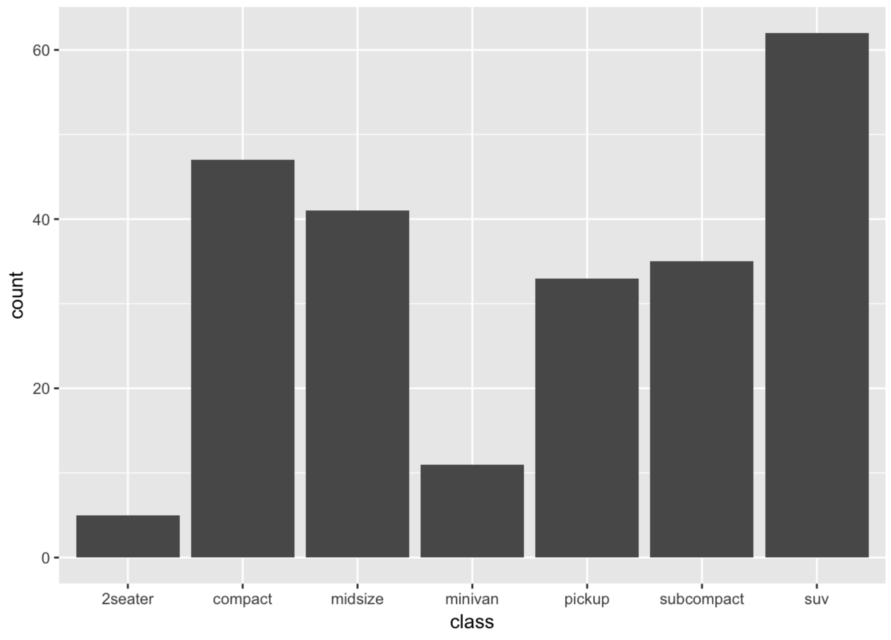
```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy))
```



This scatterplot shows all of the cars city miles per gallon against the highway miles per gallon.

Let's look at another simple chart that we can create!

```
ggplot(data = mpg) + geom_bar(mapping = aes(x = class))
```

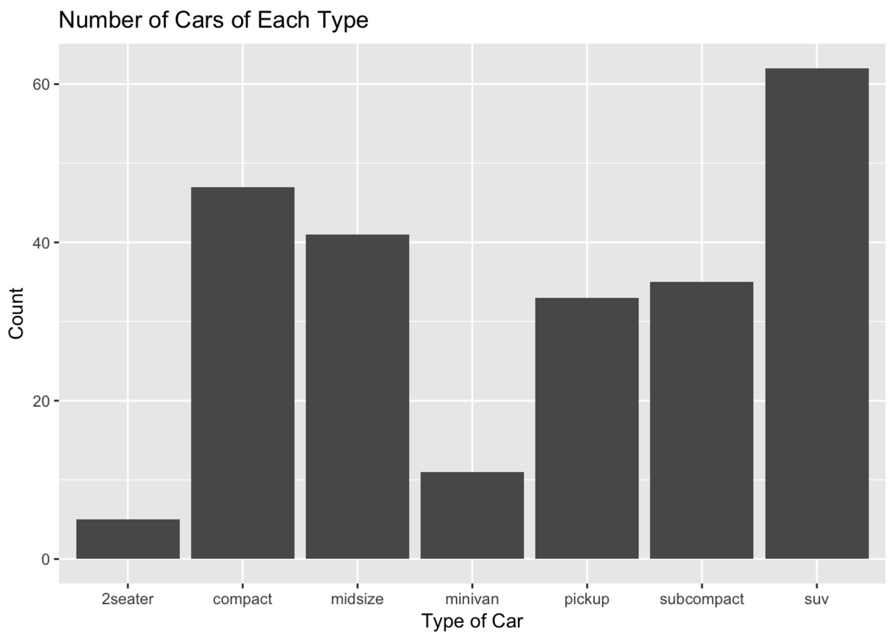


Here, we only use an ‘x’ parameter in our mapping because geom_bar will count the number of times that each entry shows up for us!

Now that we have seen some simple charts, we can add some simple features to our chart to make it look nicer, like the ones that are shown in this video.

To start off we can add some titles to our graph. We don’t have a main title for this class, and maybe “class” on the x-axis is confusing for those that don’t know cars that well. So by adding labs() to our plot, we can get our desired title.

```
ggplot(data = mpg) + geom_bar(mapping = aes(x = class)) + labs(title = 'Number of Cars of Each Type', x = 'Type of
Car', y = 'Count')
```

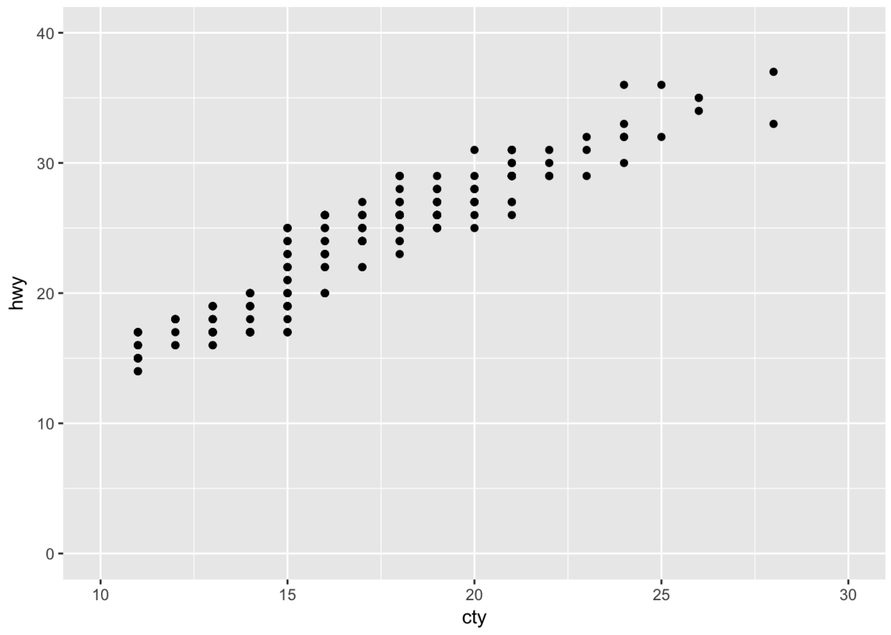


Here, we even changed the y-axis title by capitalizing it so that the titles would look proper.

Something else that we can do with these plots is set limits on the axis. Let’s say that, in the case of our scatterplot, we didn’t want to see the points at the very top, or very right. So, what we can do is set the limits of our axes to ignore those points.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy)) + xlim(c(10, 30)) + ylim(c(0, 40))
```

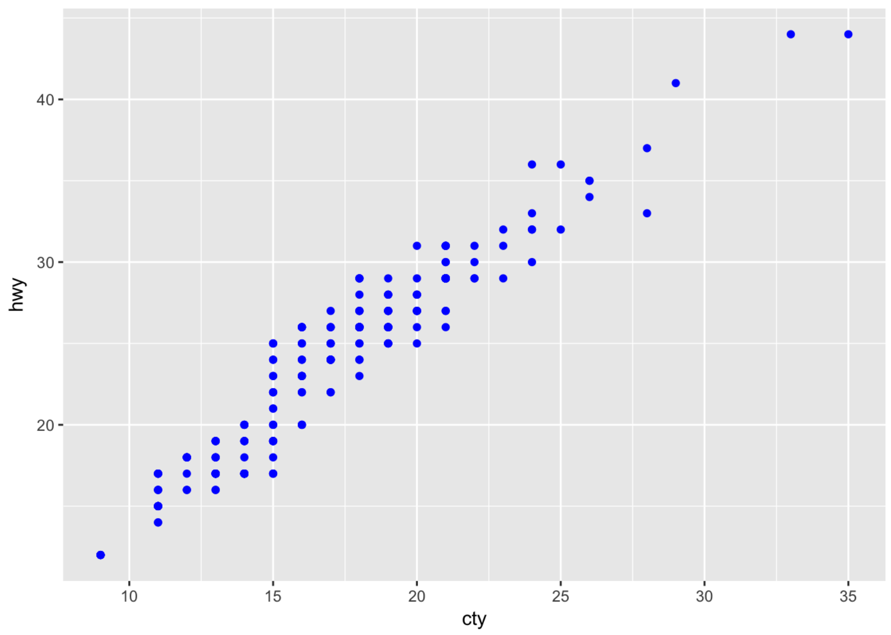
```
## Warning: Removed 8 rows containing missing values (geom_point).
```



Note: You will get a warning that there are missing values. This is how we can set our limits so we can focus on the main points of the dataset.

Another simple feature we can work with for ggplot is that we can change the color of the datapoints. For example, we can do

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy), color = 'blue')
```



Now, we can greate simple graphs and add some simple features to them. To learn more about the other types of graphs that we can create, you can take a look at the ggplot2 cheatsheet online!

Adding Some Variation

Although there are very few functions that actually create three dimensional graphs, ggplot also provides many ways for us to add a third "dimension" to our graph. It can do this with the color, shape, and size mapping as well as faceting.

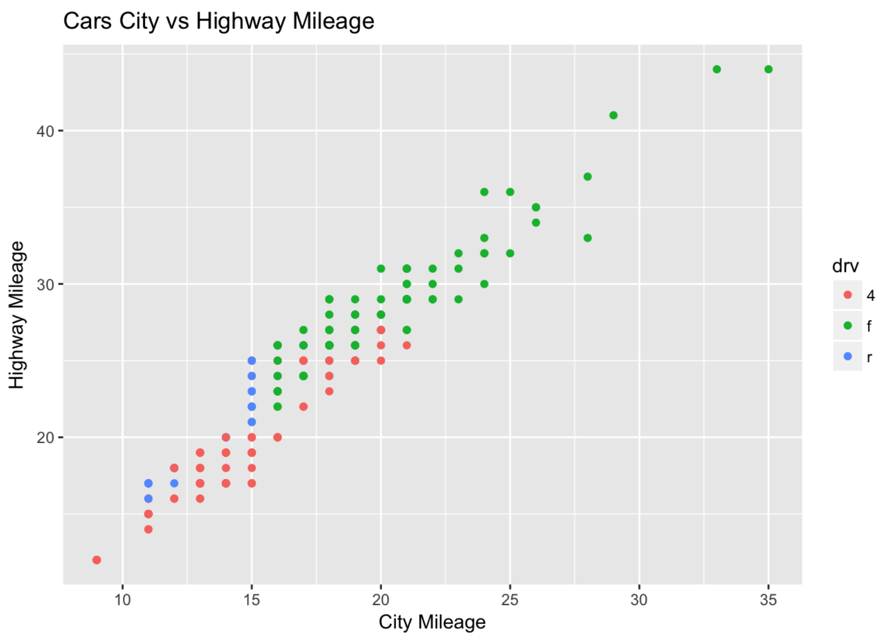
Mappings

Mappings are a powerful tool for making our graphs more understandable. They can mae our graphs look nicer as well as differentiate the different groups of our data. However, when using this, we must always remember that we make visualizations "not to create pretty pictures, but to better understand our data" (Wickham),

Color Mapping

Instead of just changing all of the points on one plot to a certain color, we can change the colors based on another categorical feature. Going back to our original scatterplot of city mileage against highway mileage, we can get more insight as to how manufacturers or what type of drive cars have affects their mileage.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy, color = drv)) + labs(title = 'Cars City vs Highway
Mileage', x = 'City Mileage', y = 'Highway Mileage')
```

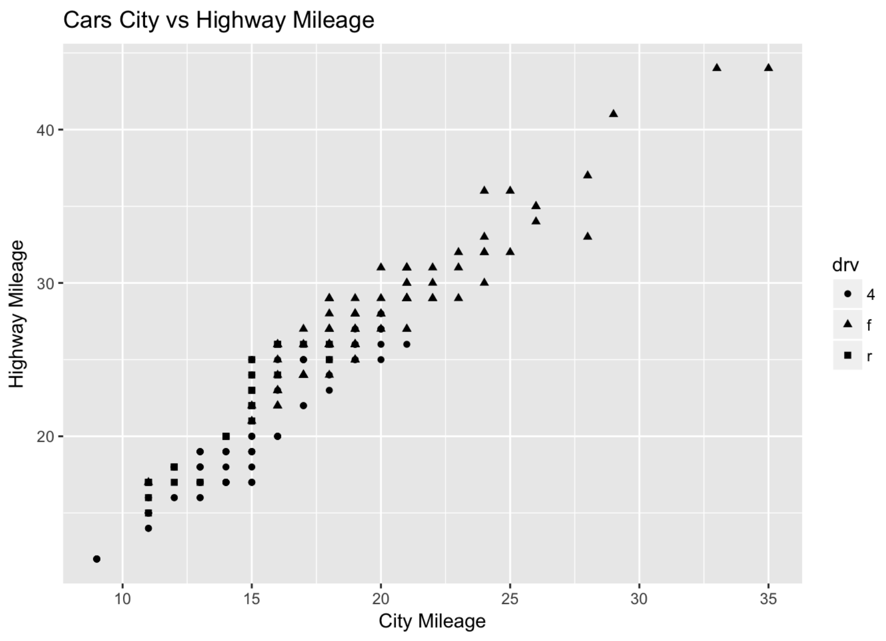


A few things to note from this graph. Previously, we set the color parameter of the geom_point function, which made all the points on the graph the same color. However, here we put the color parameter in the mappings, because we wanted the color to be dependent on another variable. If we specified the color inside the mapping to a specific color, it would be meaningless. Another thing we can note from this graph is that cars that have 4-wheel drive and rear-wheel drive tend to have poor mileage whereas cars with front wheel drive have better mileage. \

Shape

Another way we can add a third "dimension" is by making each point a different shape dependent on a different variable.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy, shape = drv)) + labs(title = 'Cars City vs Highway
Mileage', x = 'City Mileage', y = 'Highway Mileage')
```

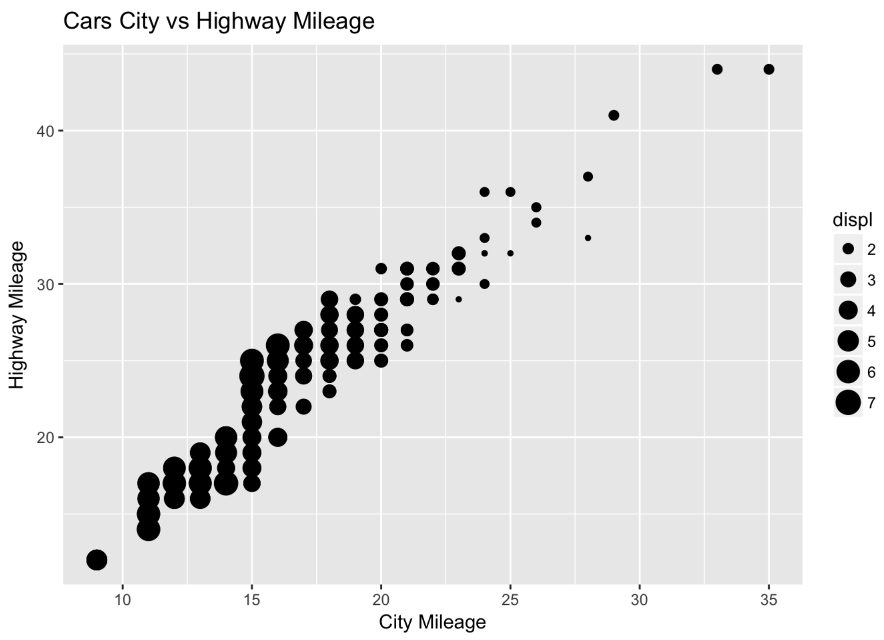


This one doesn't look as easy to read as the other one, so you may want to use color to separate the types of drive. However, there will be other times where it may be best to use shape instead. It's up to you, the user to decide!

Size

Finally, let's say that we want to add a continuous varaiable to this graph to provide some more variation. If we wanted to see if how many liters the engine can take had anything to do with the mileage, we could just add the size mapping.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy, size = displ)) + labs(title = 'Cars City vs Highwa
y Mileage', x = 'City Mileage', y = 'Highway Mileage')
```



Other Mappings

Thse aren't the only types of aesthetics that you can use inside your mappings. To look at more of the mappings that ggplot offers, you can check out this site!

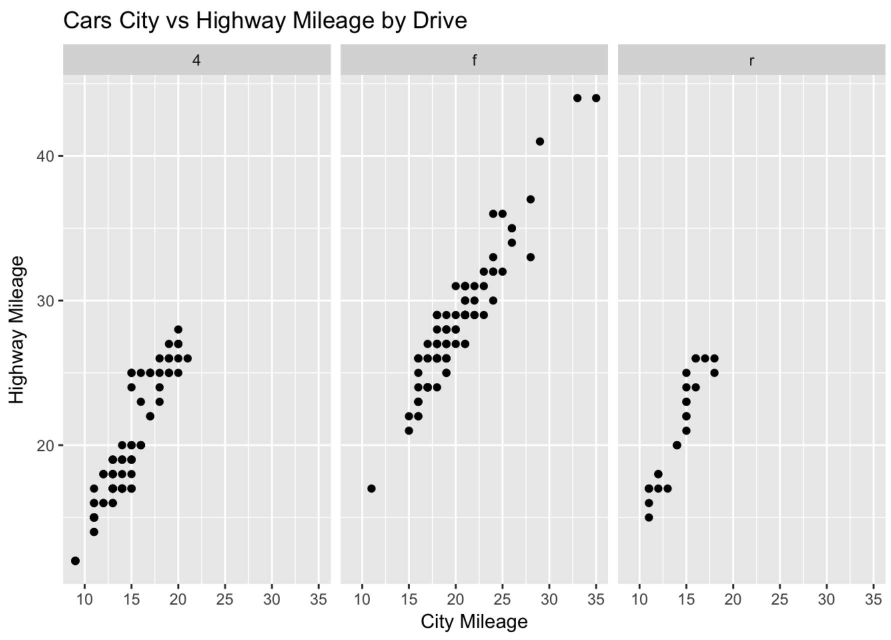
Faceting

The mappings aren't the only way to provide some more variation to our graphs. We also have faceting with ggplot2. Faceting creates multiple plots based on the categorical variable that you want to separate by. There are two ways in which we can facet: facet_wrap() and facet_grid().

facet_wrap()

Sticking with earlier examples, let's say we wanted to create the plots based on the type of drive the car takes. Here, a plot will be created for each of the types of drive and the points that are in each plot will only be cars that have that type of drive.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy)) + facet_wrap(~ drv) + labs(title = 'Cars City vs
Highway Mileage by Drive', x = 'City Mileage', y = 'Highway Mileage')
```

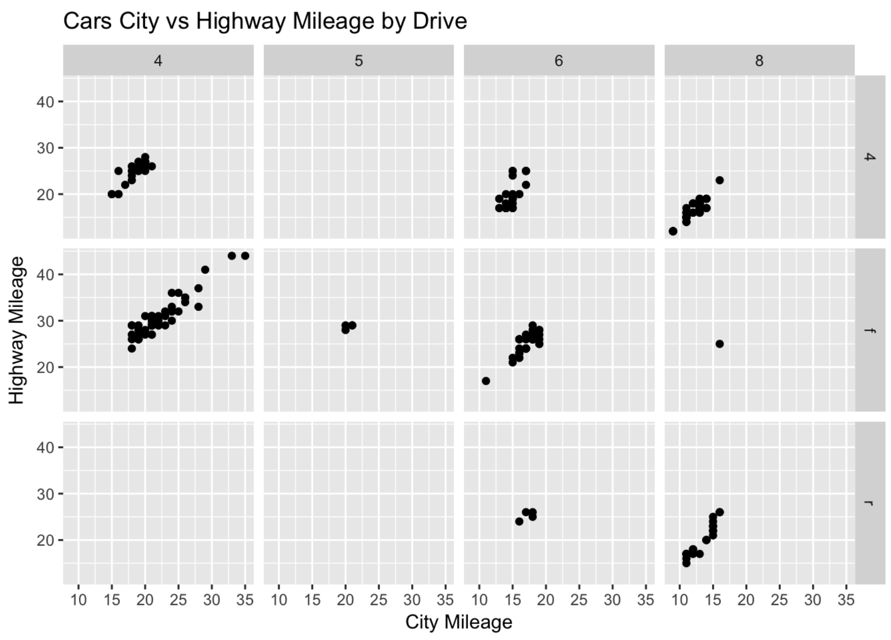


Just to reiterate what was explained earlier, we have three plots because there are three separate types of drive. This may give us more insight into what types of drive than color does. However, it is more ineffective than mappings when there are a large number of types of drive.

facet_grid()

Finally, let's say that we wanted to add two dimensions to our graph. This can be accomplished by using facet_grid(). With this, we can add two categorical variables to create a grid of plots. With this function, we will get as many plots as the number of types of the first categorical variable we are using times the number of types of the second categorical variable. In this example, we will use the drv and cyl, which is the type of engine, variables. So, each plot will have all the points that share a given combination of drv and cyl, as explain in this video. Empty plots, here, coordinate to no cars having that combination of drv and cyl.

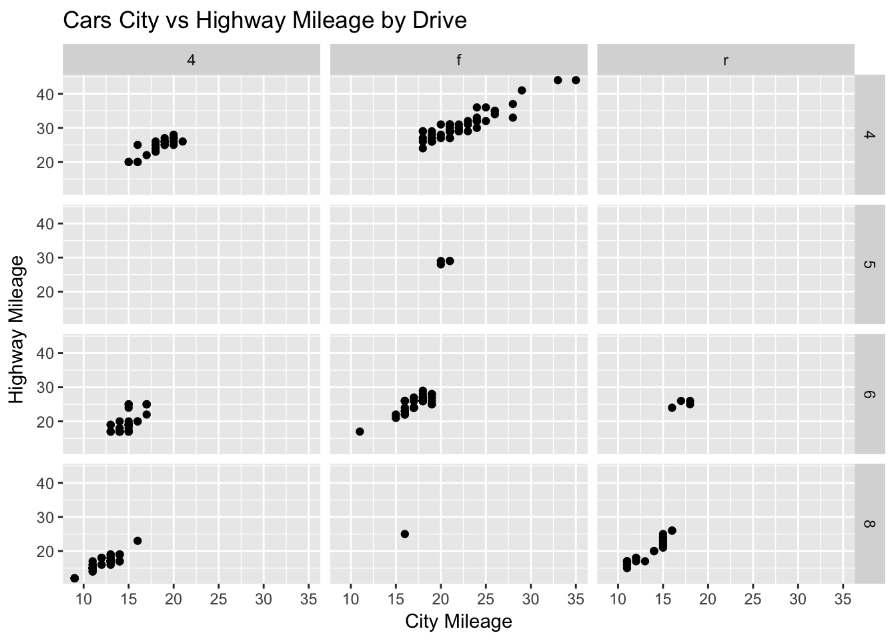
```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy)) + facet_grid(drv ~ cyl) + labs(title = 'Cars City
vs Highway Mileage by Drive', x = 'City Mileage', y = 'Highway Mileage')
```



In this example there are 4 types of cyl and 3 types of drv, so there are 12 total plots on our grid.

And if we switch the order of drv and cyl, we almost “transpose” the grid of our plots.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy)) + facet_grid(cyl ~ drv) + labs(title = 'Cars City
vs Highway Mileage by Drive', x = 'City Mileage', y = 'Highway Mileage')
```



Conclusion

With ggplot, there are many functions we can use and parameters to edit in order to display the data that you want to. By learning more about ggplot, you can vastly increase your data visualization skills. Of course, I didn’t talk about all of these in this post because there are so many! It’s up to you to learn about the different ways in which we can use this brilliant package.