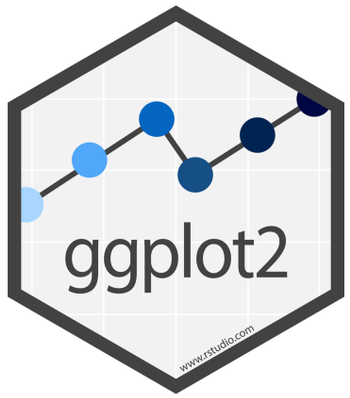
Visualize Data with



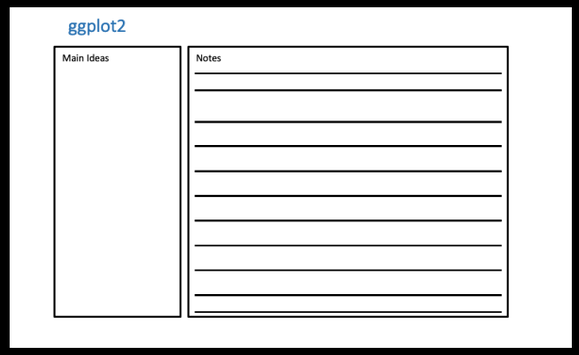
"The simple graph has brought more information to the data analyst’s mind than any other device. "

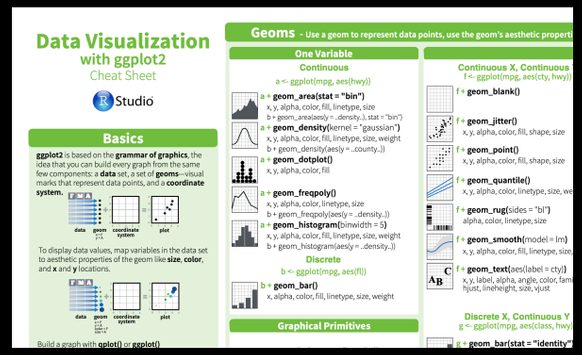
- John Tukey

Some people question whether it's best to teach data visualization first. One reason to start with it is because it has such a high impact on your ability to analyze data.

Exercise: What do you already know?

- Split into Groups of 2
- Ask you partner:
  - *What do you already know about ggplot2?*
  - *How do you currently visualize data?*
- 5 minutes





Notes form & Cheat Sheet

Please take out

There are two documents that I'd like you to take out before we start. The first is the notes form I gave you.

Your Turn

Click on
01-ggplot-exercises.Rmd

Setup

The setup chunk is always run once before anything else

mpg

Fuel economy data for 38 models of car.

```
mpg
```

Run this chunk and look at the resulting data

Quiz

Confer with your group.

What relationship do you expect to see between engine size (displ) and mileage (hwy)?

No peeking ahead!

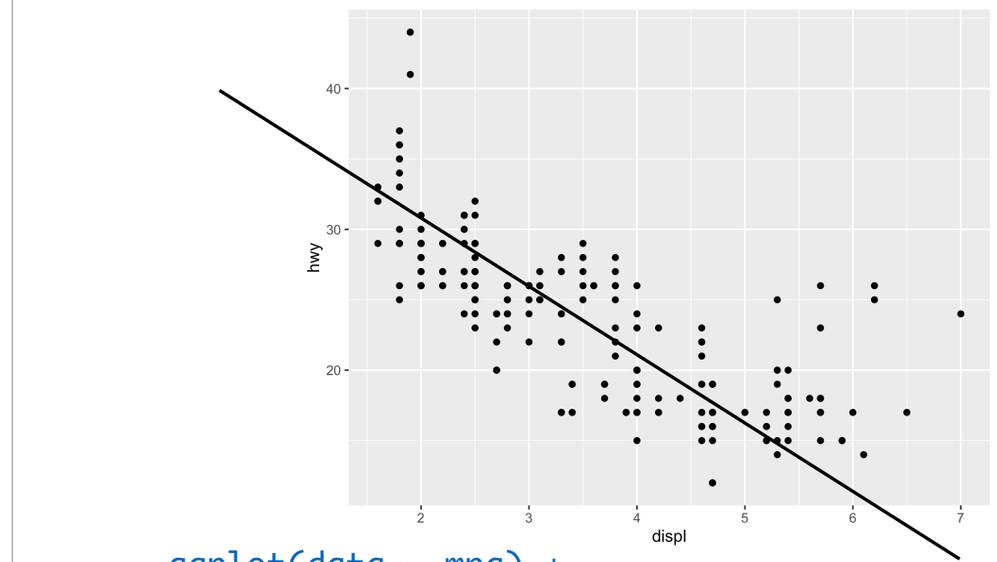
Your Turn 1

Run this code in your notebook to make a graph.

Pay strict attention to spelling, capitalization, and parentheses!

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

You should have a blank place in your notebook to enter this code. Enter it then click the "Run" arrow




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

What we see here is that as engines get larger, their mileage tends to decrease

1. "Initialize" a plot with ggplot()
2. Add layers with geom_ functions

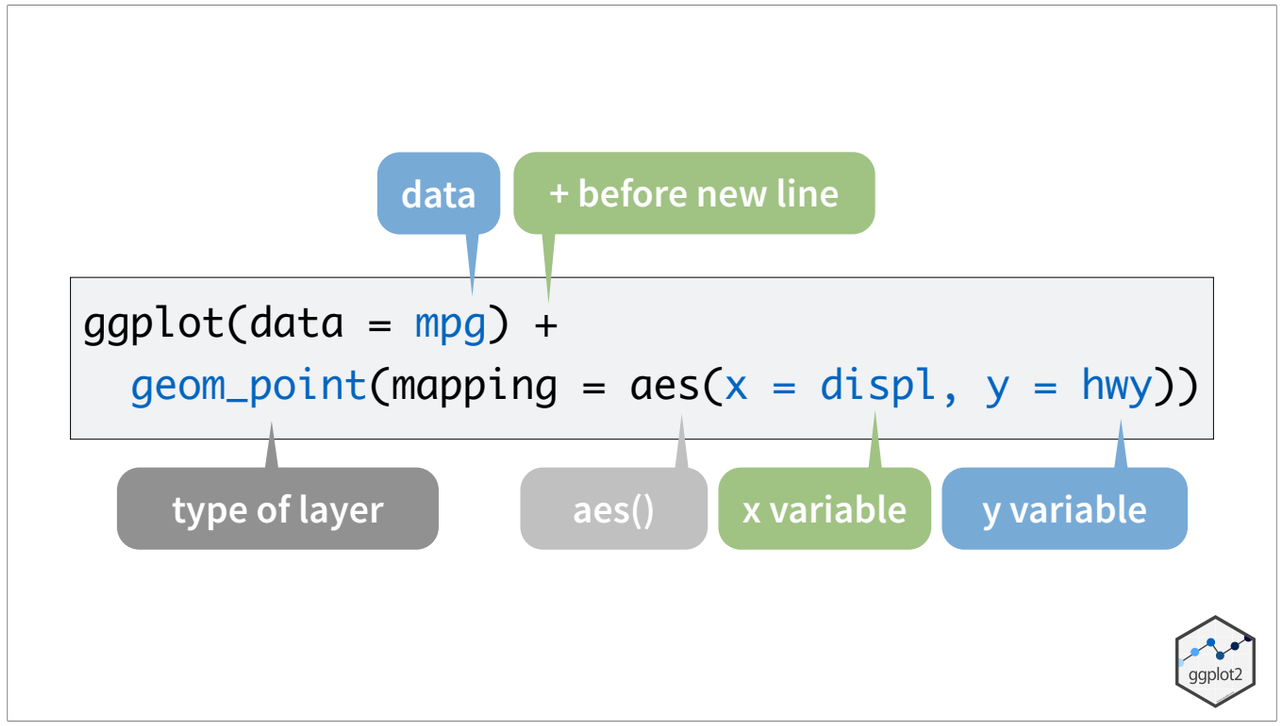
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

ggplot2 has a very unique syntax. The code to create all plots follows the same pattern. First, we initialize ggplot with a dataframe. Then we add "layers" with functions that start with the word "geom_".

Pro tip: Always put the + at the end of a line, Never at the start

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

The #1 bug that beginners make is that they forget to put the + sign at the end of each line!



I mentioned that each layer starts with "geom_". "geom" is short for "geometry". Here we wanted a scatterplot, so we used "geom_point". Geometry is a key concept in ggplot, and we'll be exploring it in detail later.

Another key concept is "aesthetic mappings". Information such as what variables should be on the x- and y- axes are considered aesthetic mappings.

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

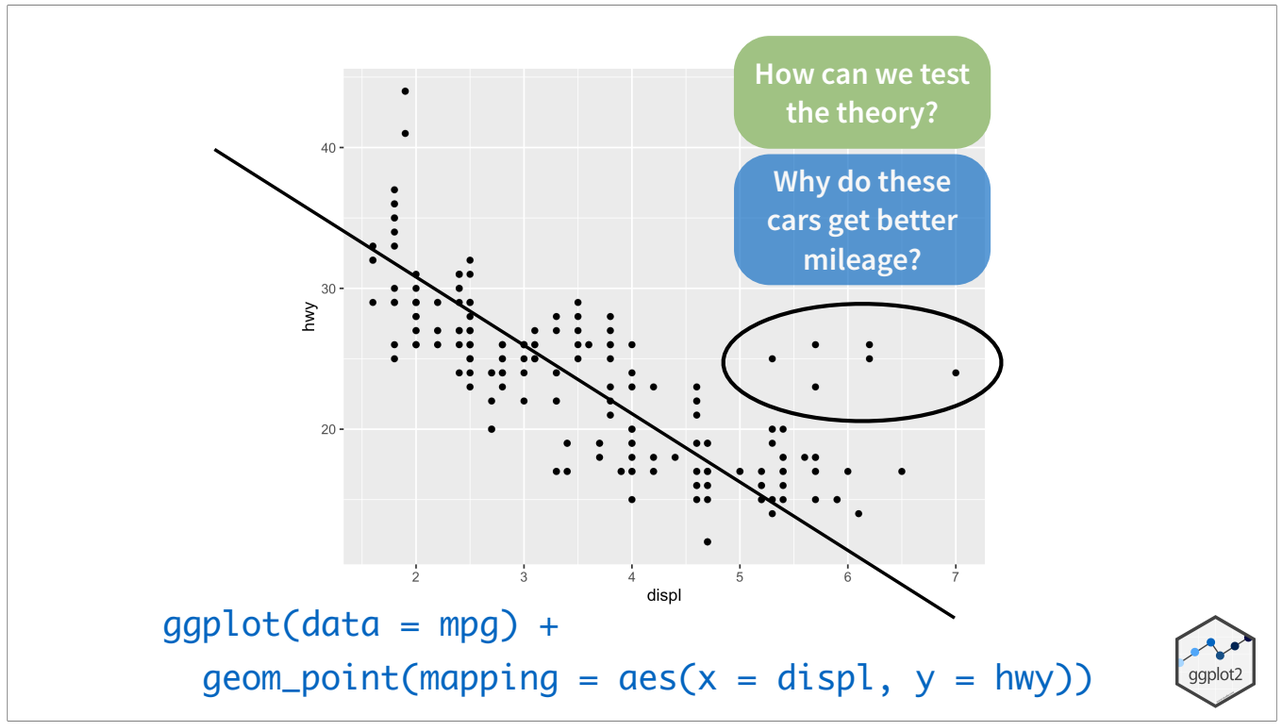
If you had to memorize one thing in this lesson, let it be this: The template for all ggplots. To make any graph, you need 3 things: your data, some geometry (like points, lines, bars) and some aesthetic mappings (such as axis choices, color and shape).

Mappings

Now let's look at aesthetic mappings in more detail.

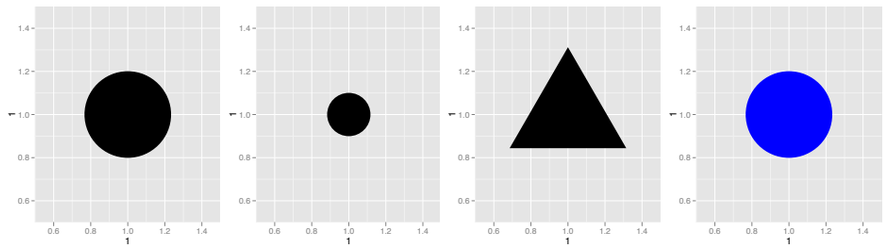
"The greatest value of a picture is when it forces us to notice what we never expected to see."

- John Tukey

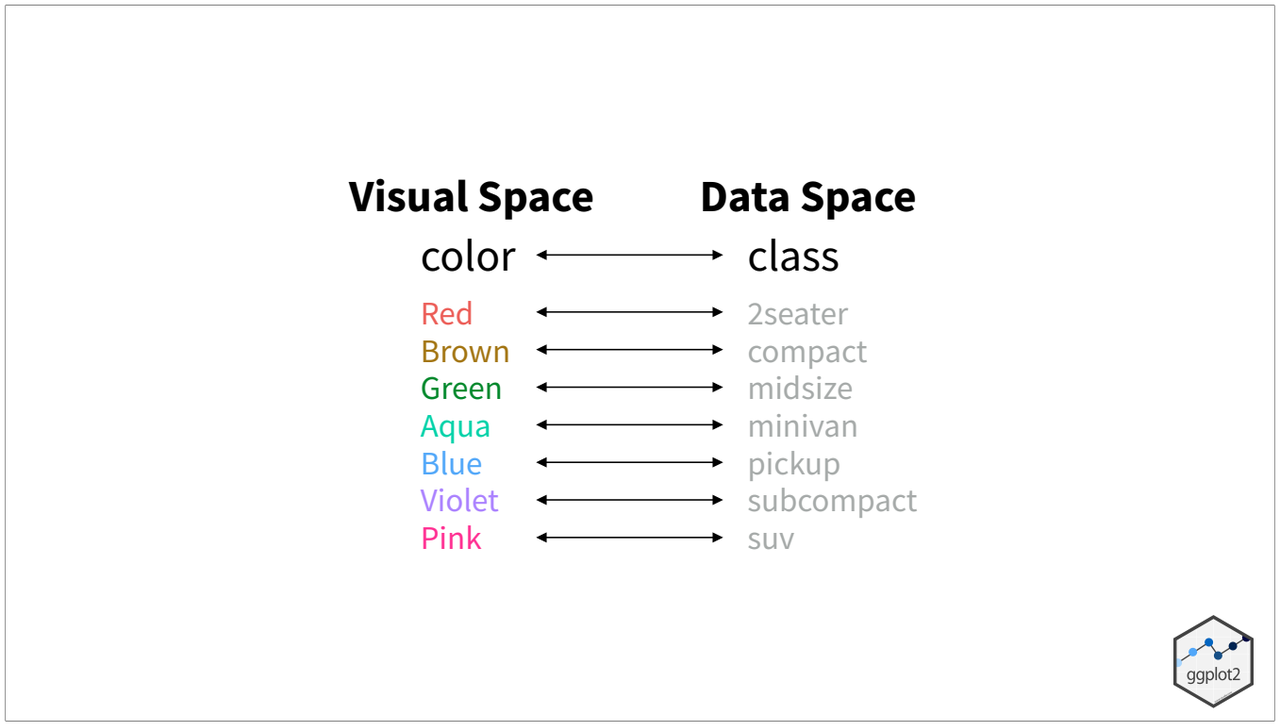


Let's motivate our discussion of aesthetic mappings. You might have noticed that there is a cluster of cars that seem to get better mileage than we expect. Can you take a guess as to why? In this section we're going to use aesthetic mappings to look at patterns in this dataset.

Aesthetics



We already looked at how the x and y axes are a type of aesthetic mapping. Besides that, we can also use options like size, shape and color.



You might recall that the mpg dataset has a column called "class". We can make an aesthetic mapping between that column and the color of the dots.

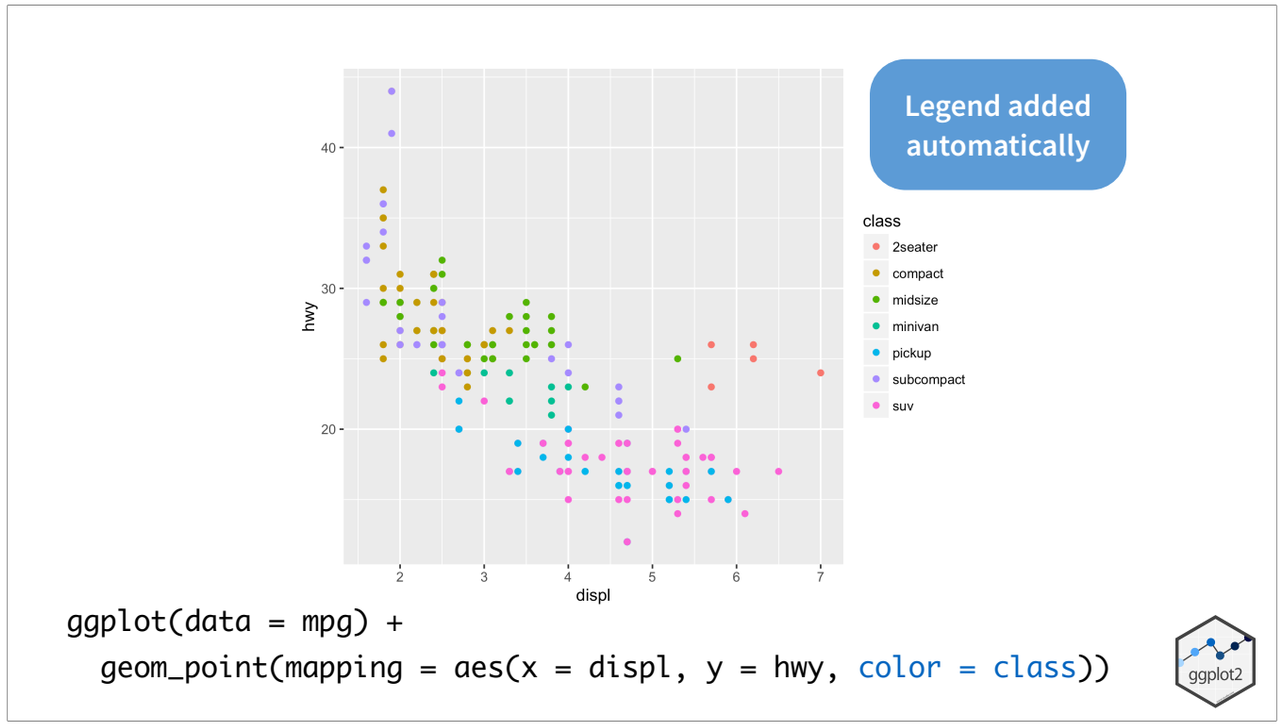
Aesthetics

aesthetic property

Variable to map it to

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, size = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, shape = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, alpha = class))
```

What makes ggplot2 so powerful is that setting aesthetic mappings is so easy. Setting any aesthetic property like color or size is just as easy as setting the x and y variables.




```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

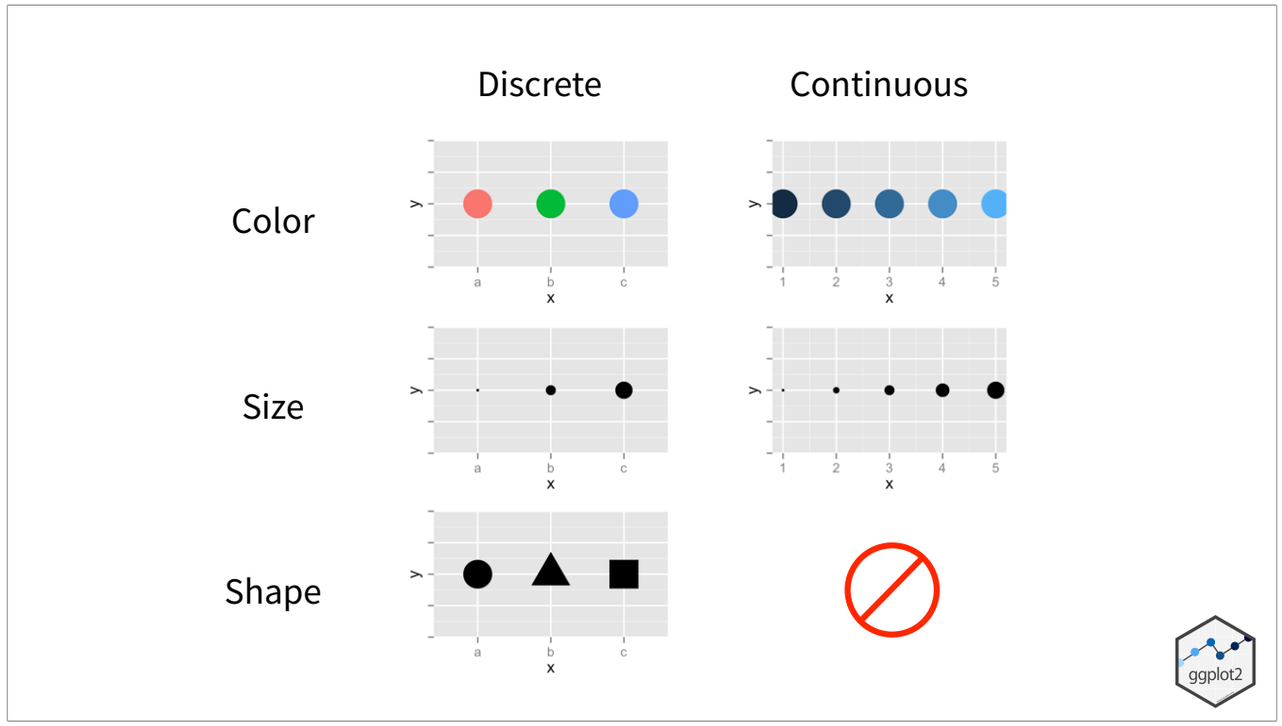
Note that the legend is added automatically.

Your Turn 2

In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

Do different things happen when you map aesthetics to discrete and continuous variables?

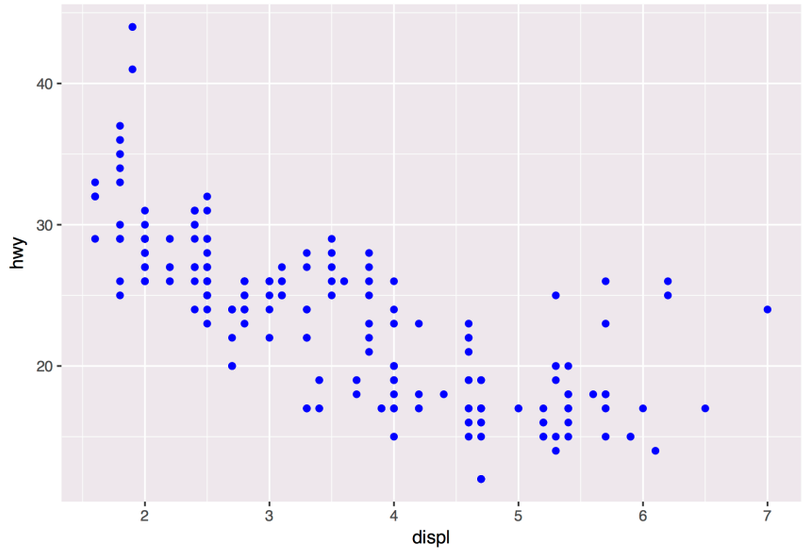
What happens when you use more than one aesthetic?



In R, it's common to classify variables as either Discrete or Continuous. Most aesthetic mappings behave similarly for discrete and continuous variables. For example, color and size behave similarly. Shape is the only variable that simply does not work for continuous variables.

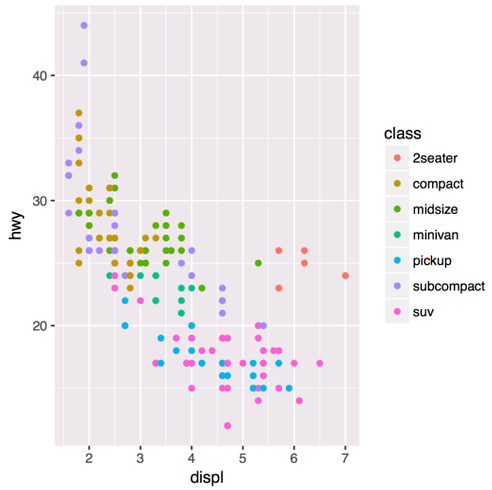
set vs. map

How would you make this plot?



How is this plot different than the one you just made?

The answer is that each and every point has its color set to blue.

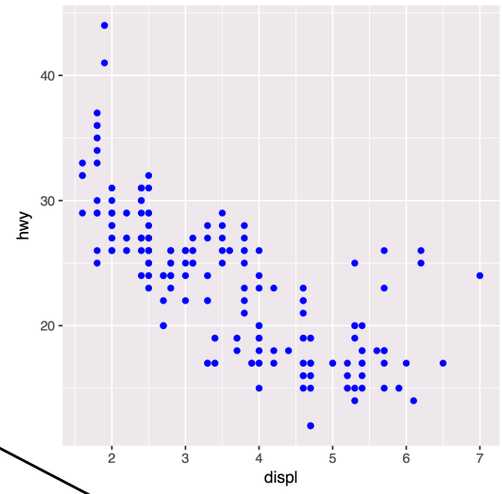


Inside of aes(): maps an aesthetic to a variable

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

As a reminder, in the last exercise you set *color=class* as an aesthetic mapping. That made color vary as a function of class.

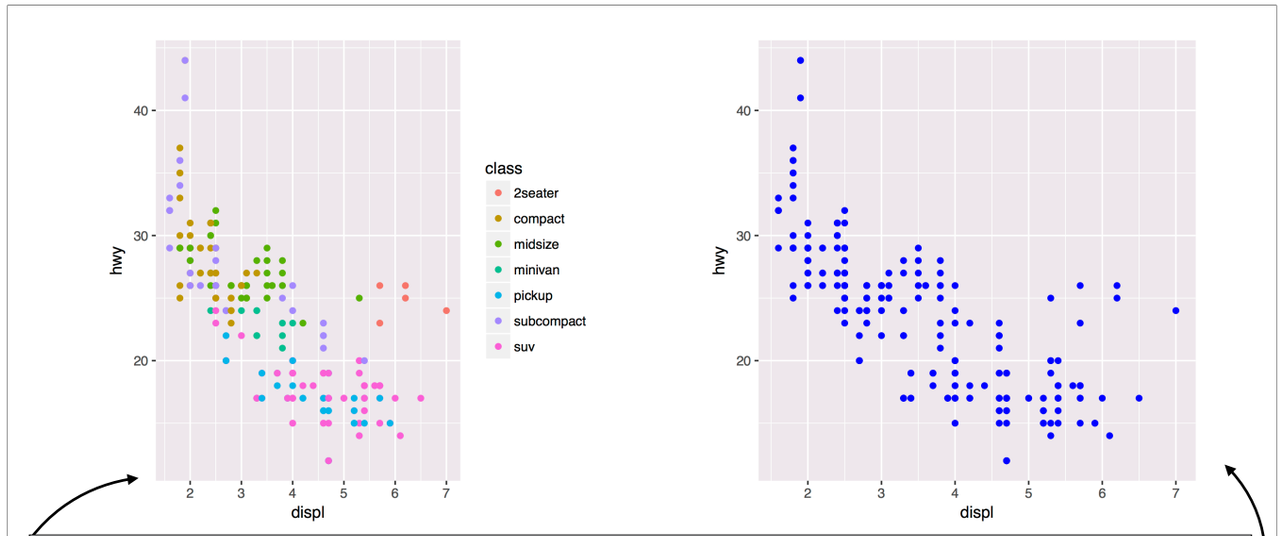
Outside of aes(): sets an aesthetic to a value



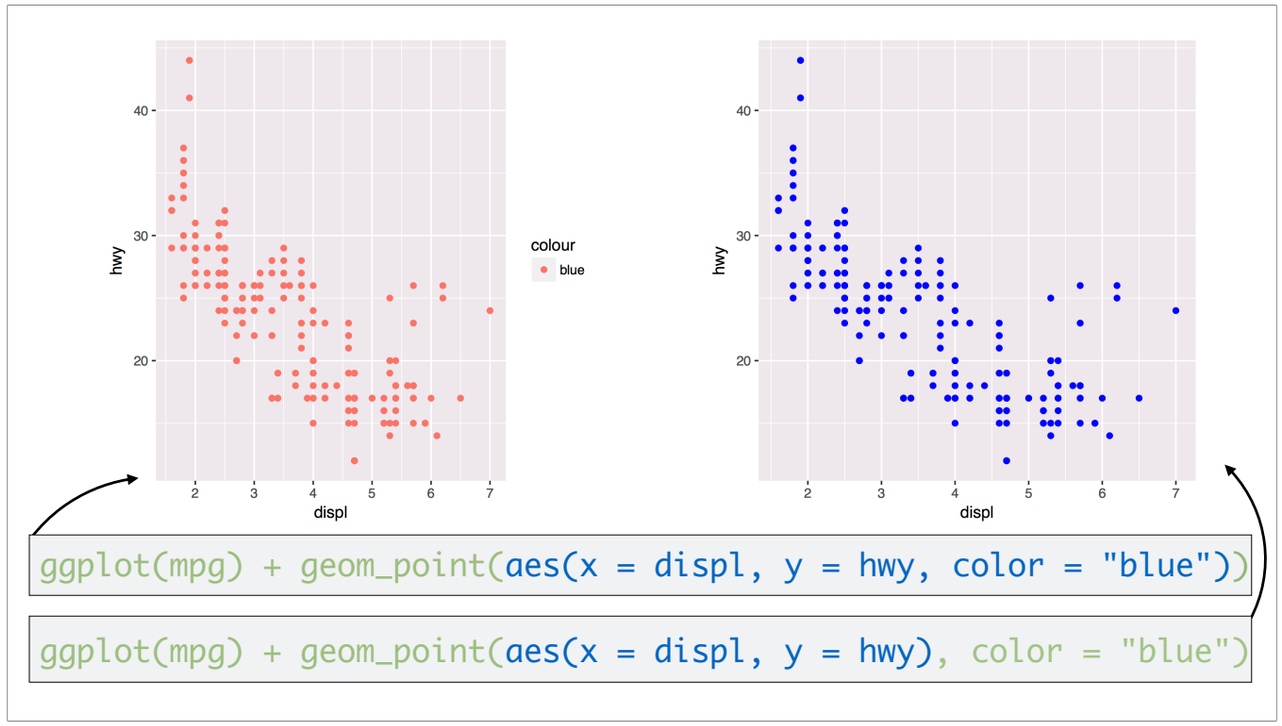

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), color = "blue")
```

If we instead set color="blue" (a constant value) *outside* the "aes" call, then it applies to each element of the geometry. We are no longer **mapping** color to a variable, we are **setting** it for all data points.



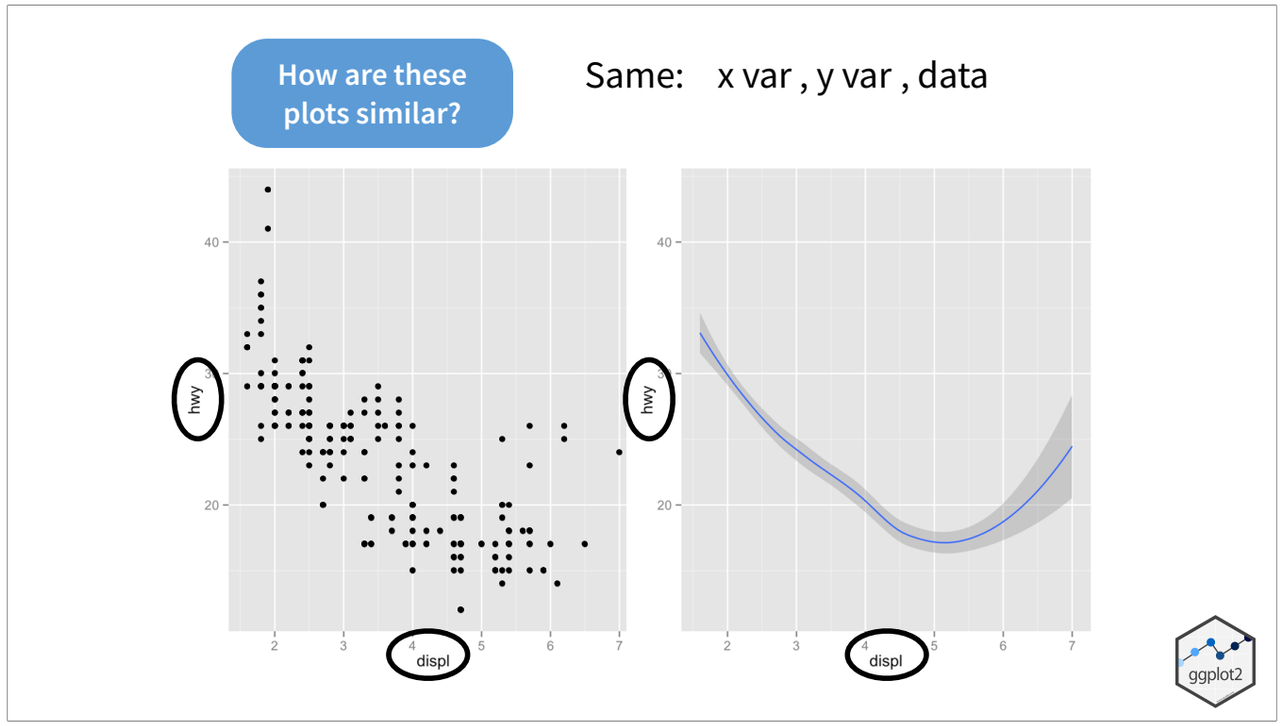
Here we see the two graphs side by side. In the first chart, color is an aesthetic mapping: it varies based on a value in the dataset.
In the second chart, we set color as a constant value (so it's outside the "aes" call). This distinction is very important.



Here we see one of the most common errors that people make with ggplot2. In the first chart we set *color*="*blue*". But because it's in the aesthetic mapping call, ggplot2 assumes that it is a variable. So "blue" becomes a part of the legend.

Geoms

Now let's look at the second major part of the ggplot2 template: geometries.

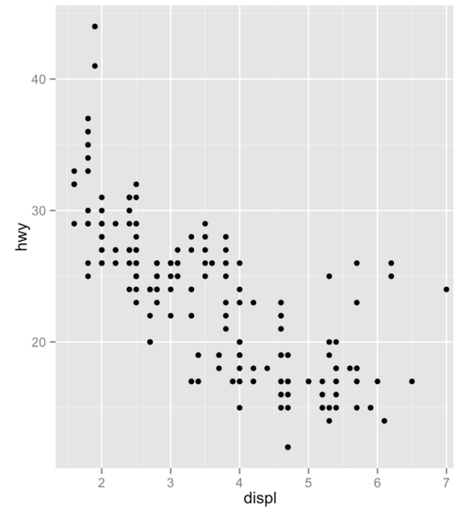


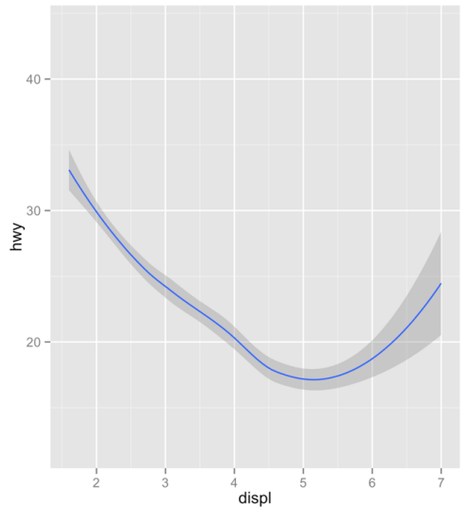
These are two graphs that appear similar but have an important difference. How many similarities can you find?

How are these plots different?

Different: geometric object (geom),

e.g. the visual object used to represent the data





Now that you already pointed out the similarities, try to point out the differences.

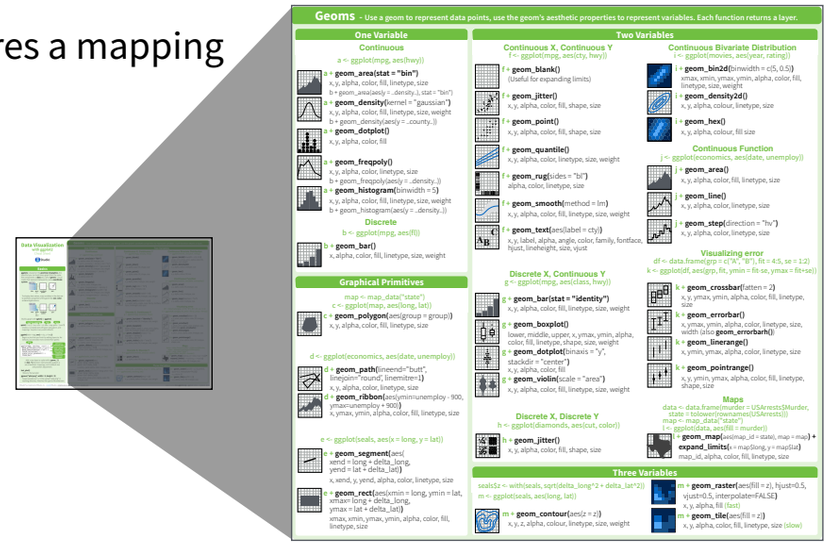
geoms

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

What makes ggplot2 special is that each geometric object has its own name, and it is trivial to change the geometry used.

geom_ functions

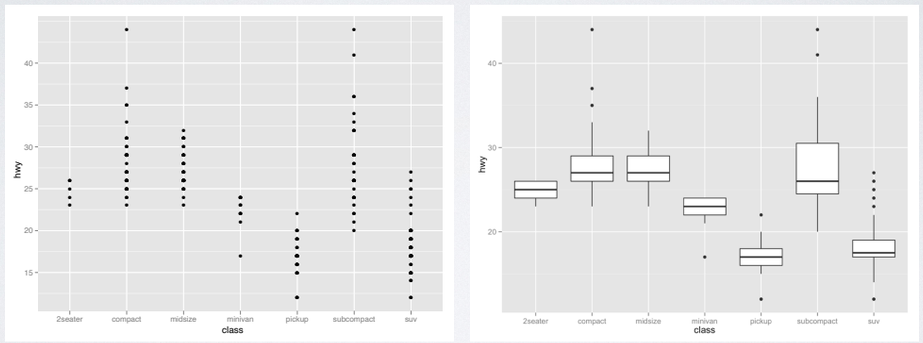
Each requires a mapping argument.



Please take out your ggplot2 "Cheat Sheet". You can see here a list of all the geometric objects that ggplot2 can render. Note that *each of them requires an aesthetic mapping, which is why we learned about mappings first.*

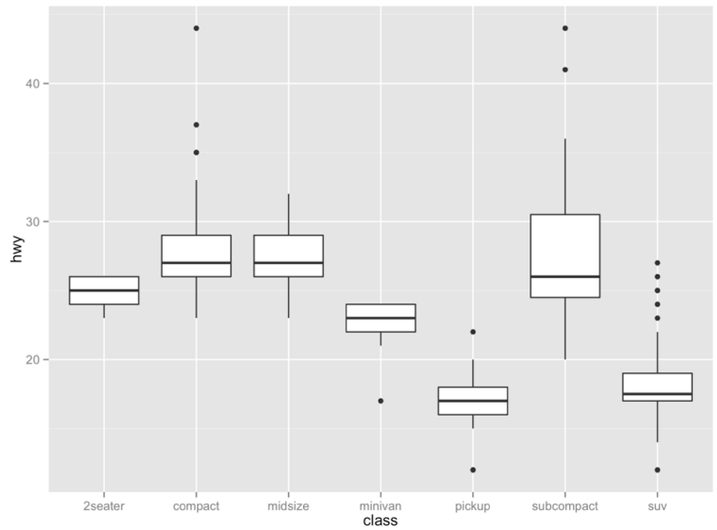
Your Turn 3

With your partner, decide how to replace this scatterplot with one that draws boxplots. Use the cheatsheet. Try your best guess.




```
ggplot(mpg) + geom_point(aes(class, hwy))
```

Now we're going to do a series of quick exercises that relate to using different geometries. In this first exercise, I'm giving you code to make a scatterplot. And I want you to convert it to a boxplot instead.

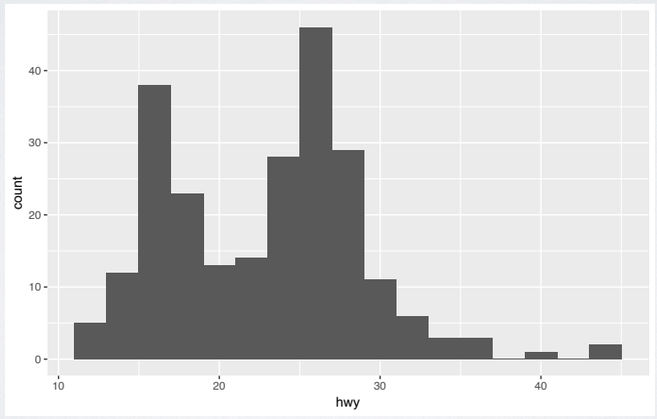



```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

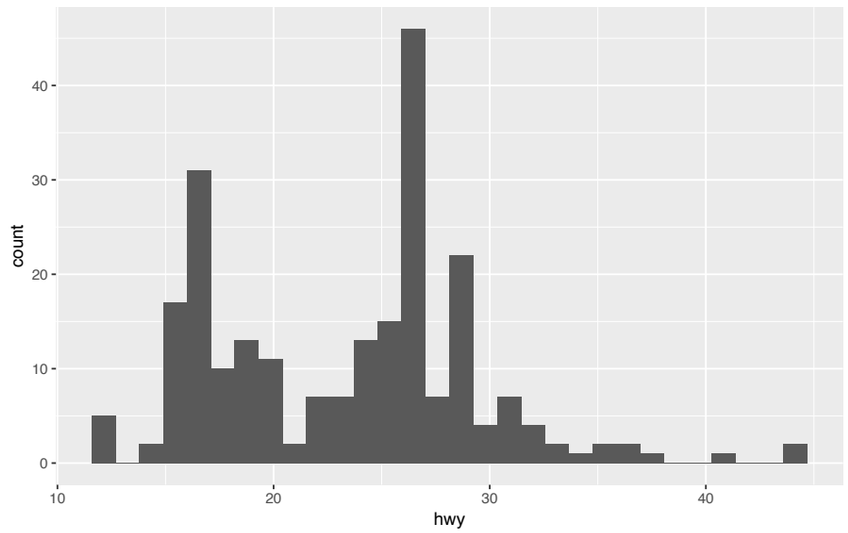
The answer is to use **geom_boxplot** instead of **geom_point**

Your Turn 4

With your partner, make the histogram of **hwy** below. Use the cheatsheet. Hint: do not supply a **y** variable.

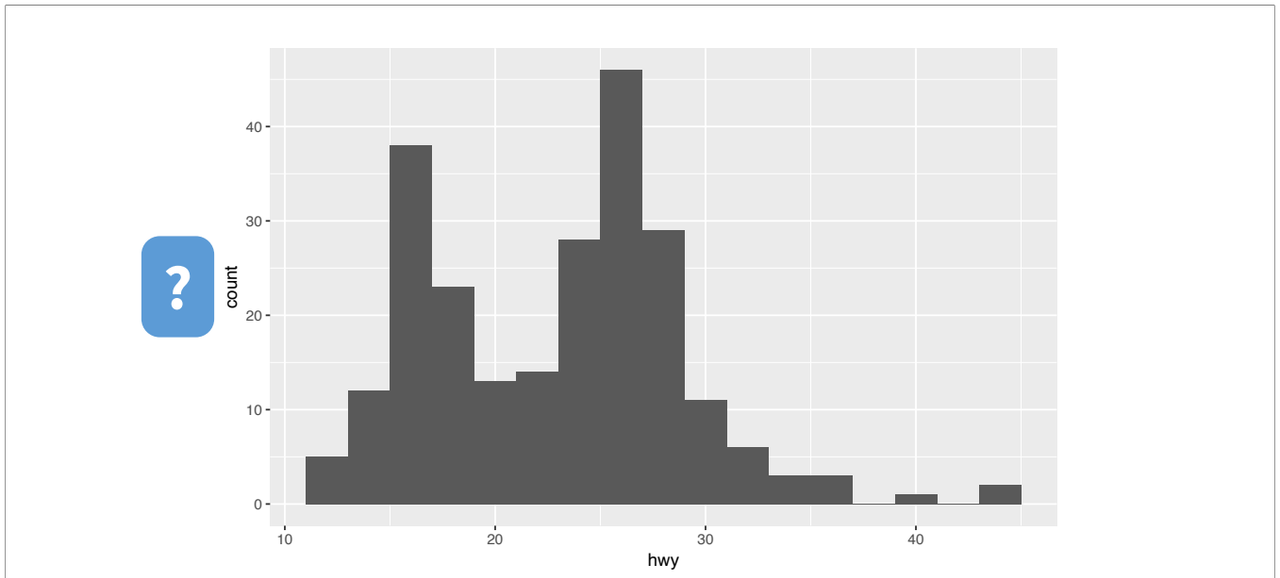


How would you make a histogram of highway mileage (**hwy**)?




```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy))
```

Histograms only take one variable - the x variable. The y value is calculated automatically.

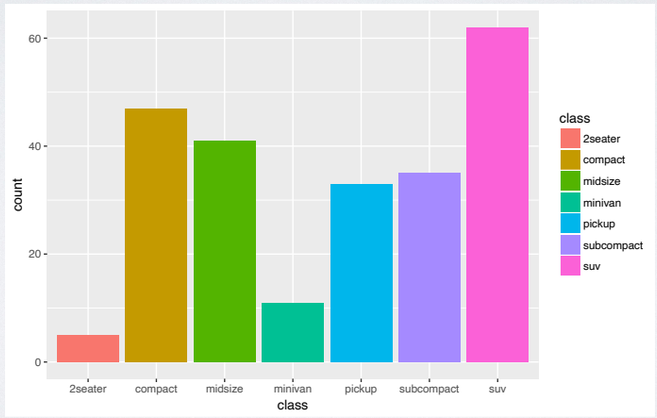



```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy), binwidth = 2)
```

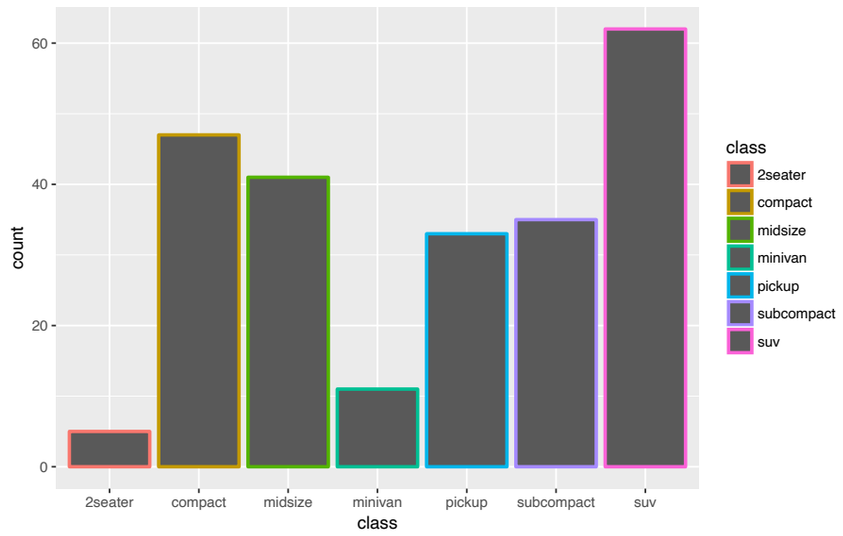
You can also set the size of the bins for the histogram with the **binwidth** parameter.

Your Turn 5

With your partner, make the bar chart of **class** colored by **class** below. Use the cheatsheet. Try your best guess.

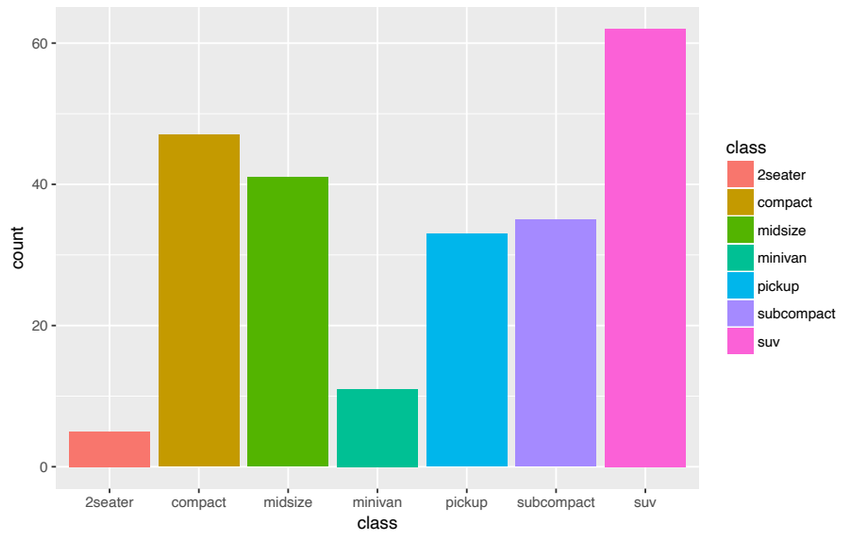


Histograms are used to show counts of quantitative data. Bar charts are used to show counts of categorical data.




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

I'm sure that many of you tried this, and were surprised by the result.




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

The way to get the correct result is to use a new aesthetic mapping: **fill**.

The goal here is to show you that there are lots of different aesthetic mappings. And sometimes they are not so obvious. Even experts at ggplot2 need to sometimes look at the documentation!

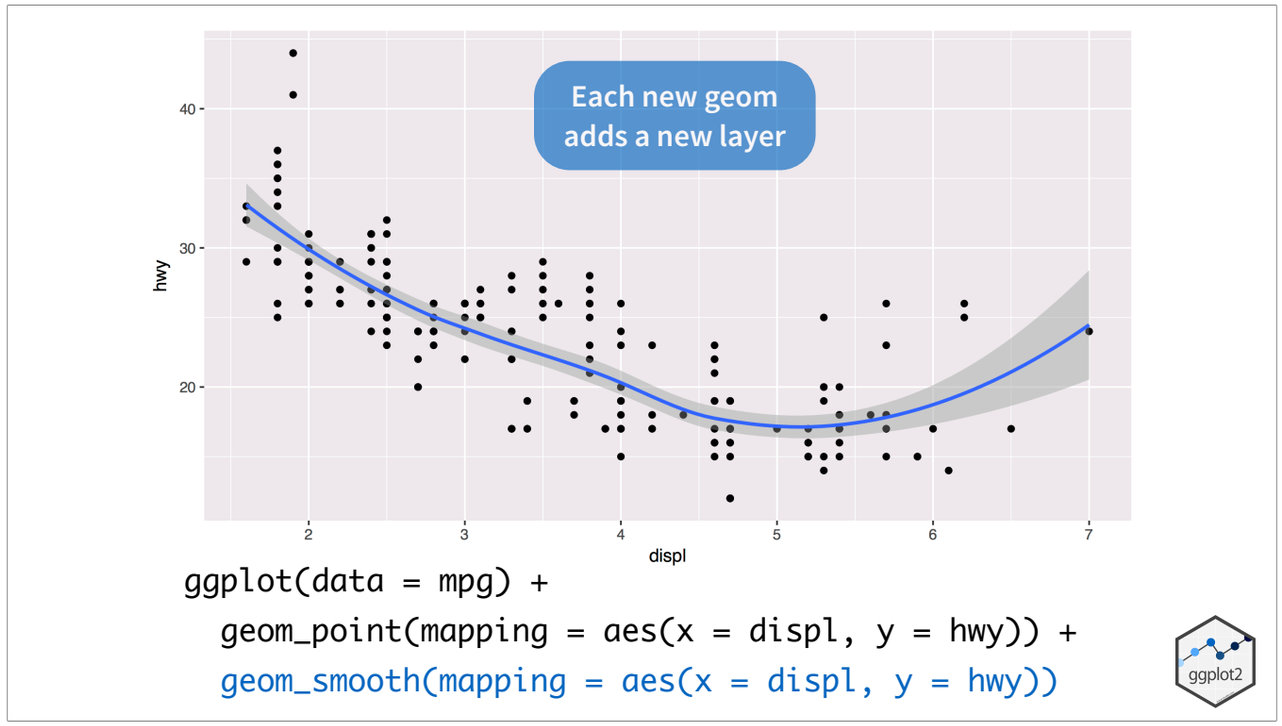
Your Turn 6

With your partner, predict what this code will do.

Then run it.

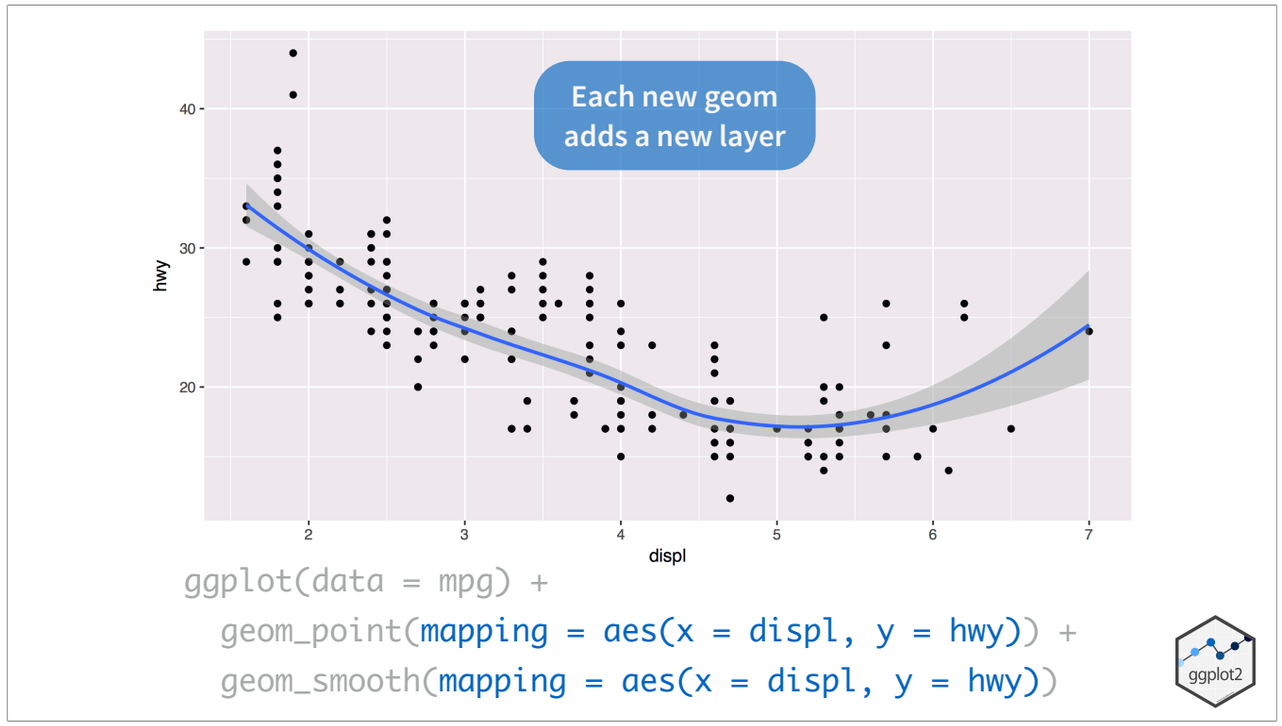
```
ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  geom_smooth(aes(displ, hwy))
```

This is a reverse of the type of exercises we've been doing until now. Now I want you to look at code, and guess what it will do.




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

This is the first time we've seen a single chart composed of multiple geometries! And as you can imagine, a single chart can contain any number of layers.

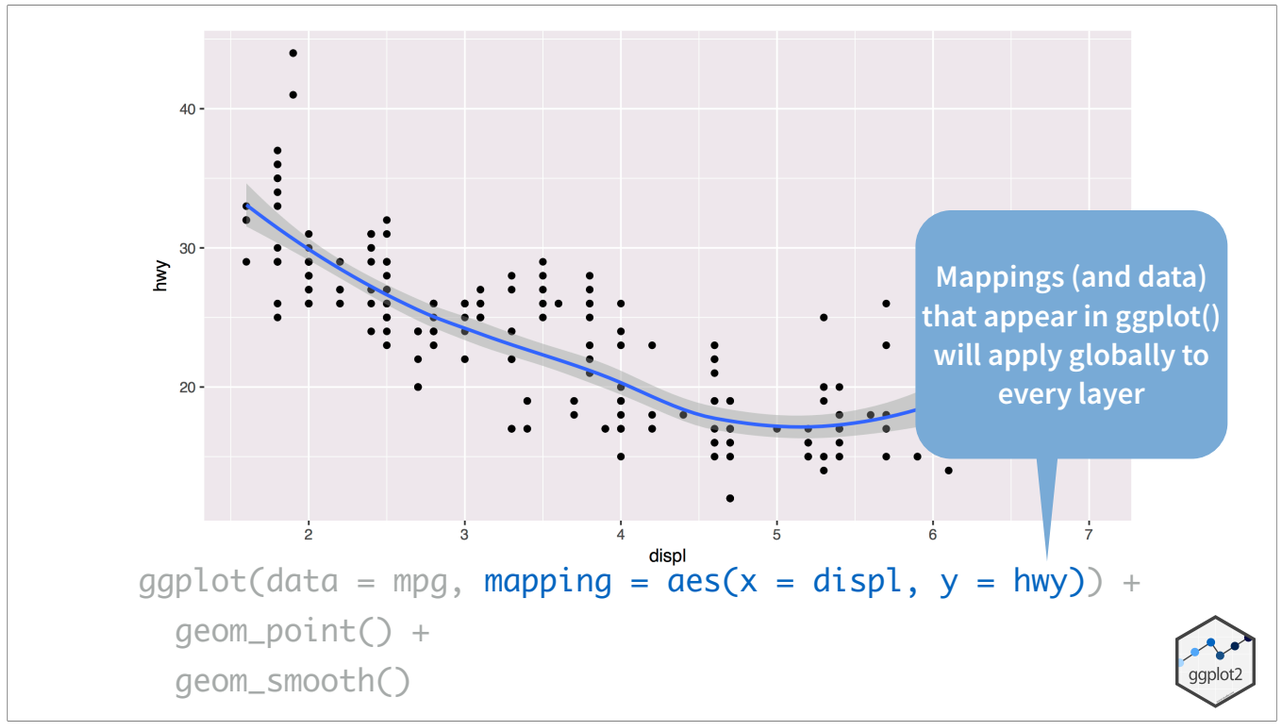



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

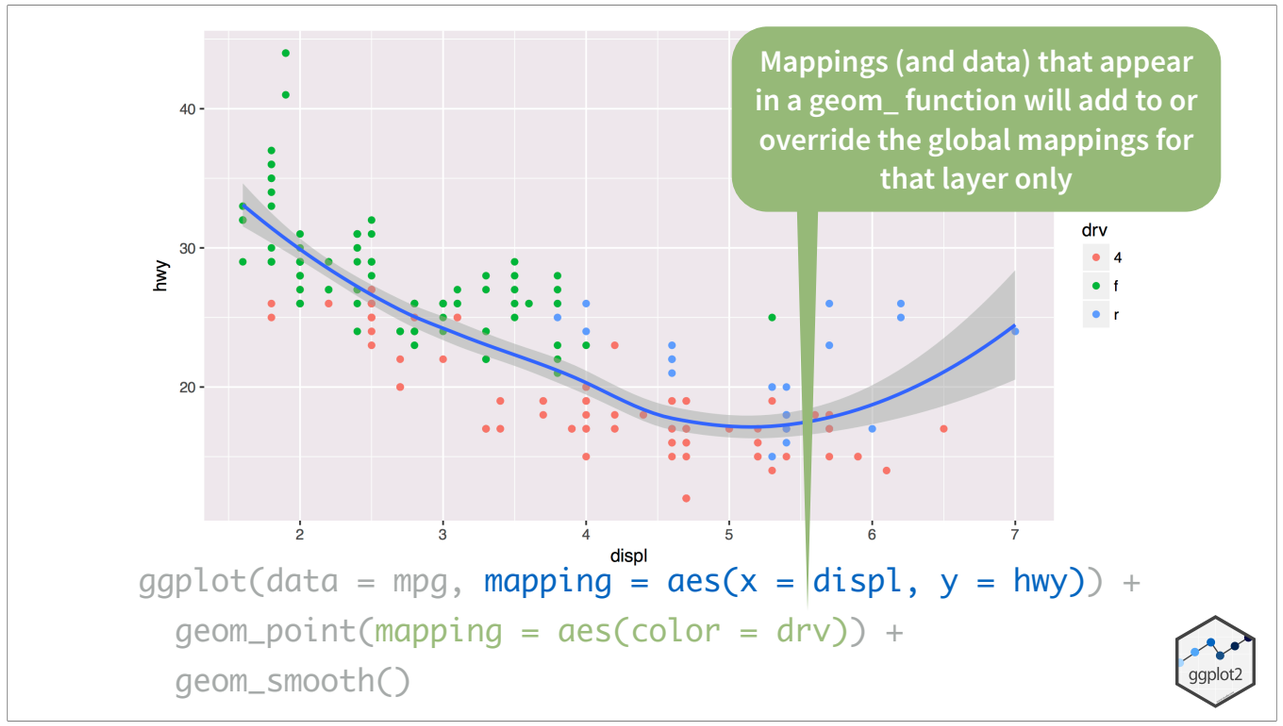
One thing I want you to focus on here is that there seems to be some code duplication: the aesthetic mappings for both layers are identical. In the next section, we'll learn how to address this.

global vs. local

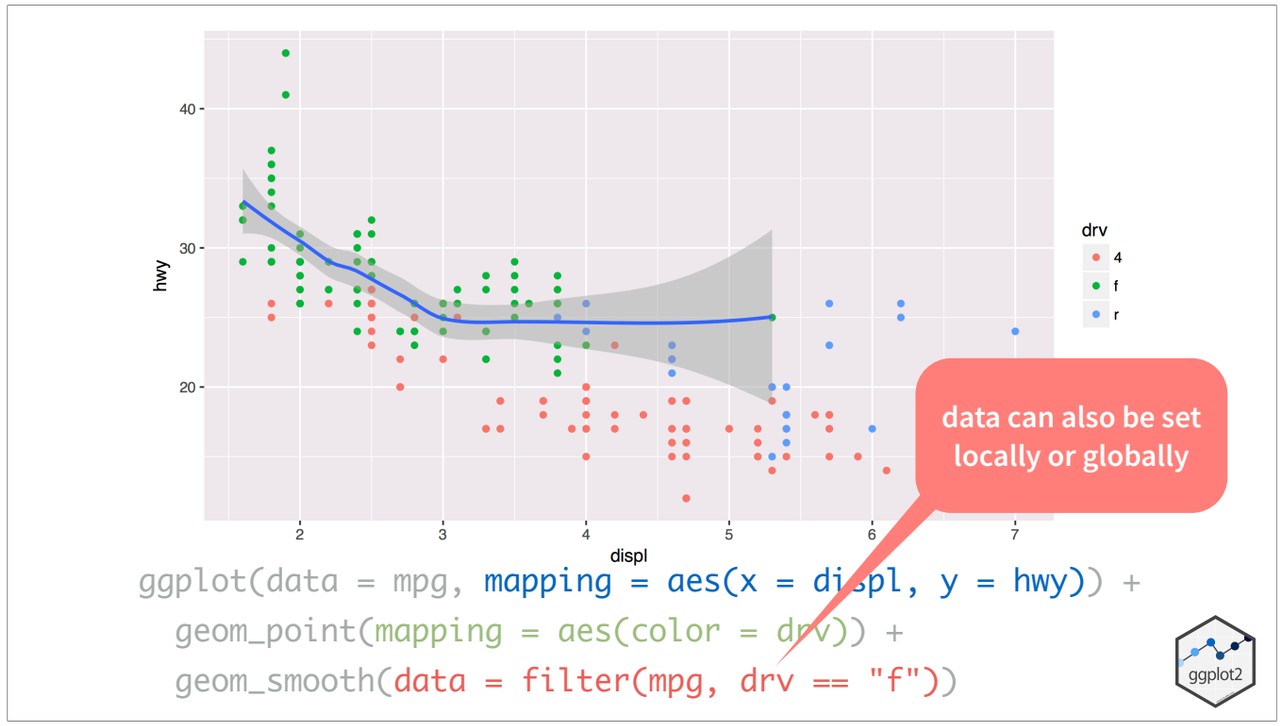
The way that we avoid this duplication is by learning how to set aesthetic mappings globally (as opposed to locally).



When you set an aesthetic mapping inside the initial "ggplot" call, the mapping will apply to each layer in the entire plot.



But this inheritance isn't permanent. You can selectively override these variables in each layer



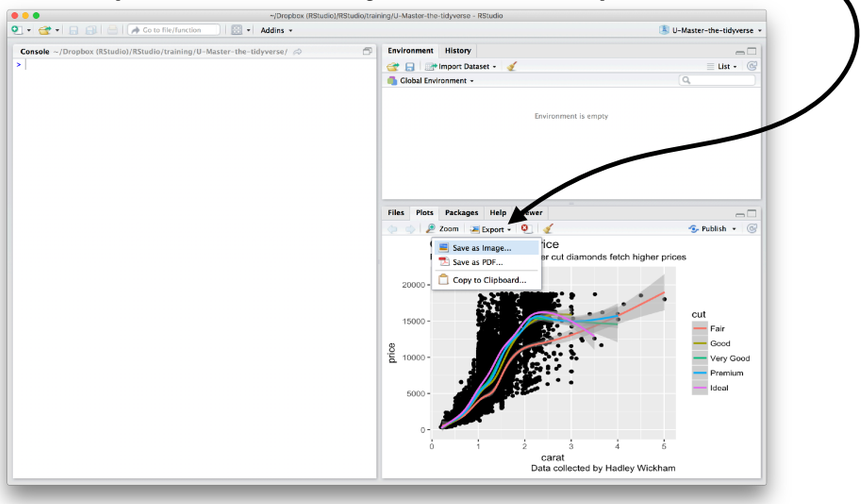
In fact, you can even go further. Here we're selectively setting a dataset for the final layer. The smooth line here is only for cars with front wheel drive.

Saving graphs

So far we've learned how to create graphs inside R Notebooks. But what if you want to save a chart and email it to a friend?

Manually saving plots

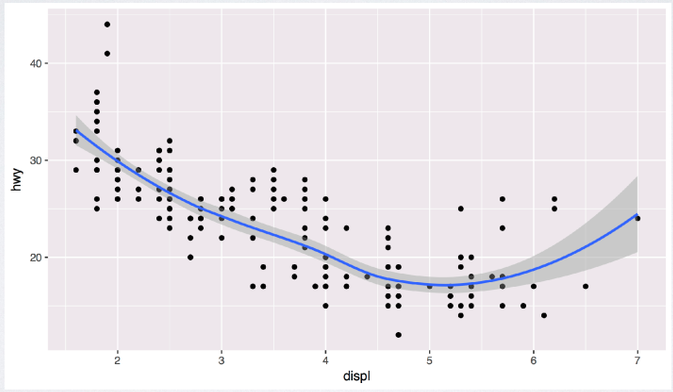
Save plots manually with the export menu



The solution is to run the code that creates the graph at the console. It will cause the graph to appear in this lower-right-hand pane. That pane has an "Export" button that lets you save it.

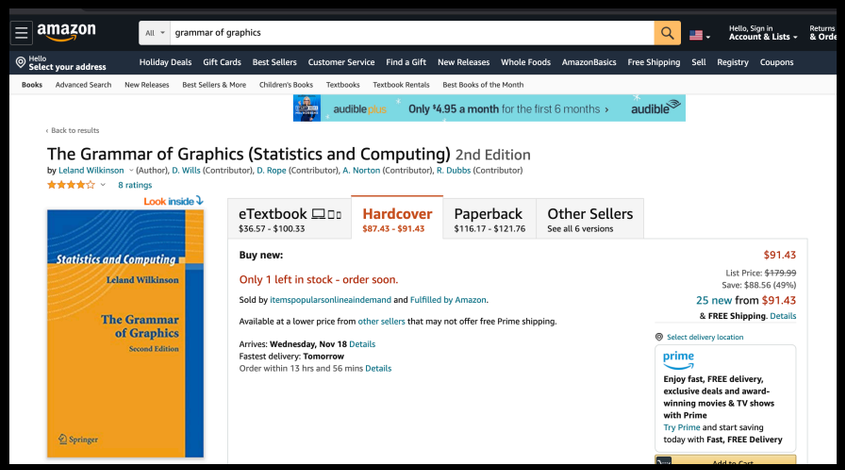
Your Turn 7

Run your code from the last exercise in the Console. Click "Export" in the window that it appears in, and then save the graph!



Grammar of Graphics

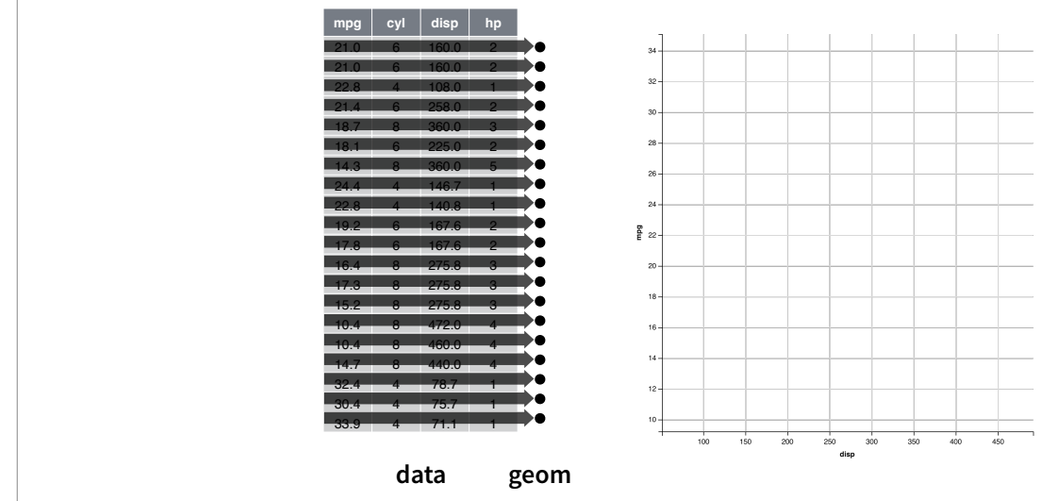
You've now developed an intuition about the grammar of graphics. Let's go a step further, and talk about it explicitly.



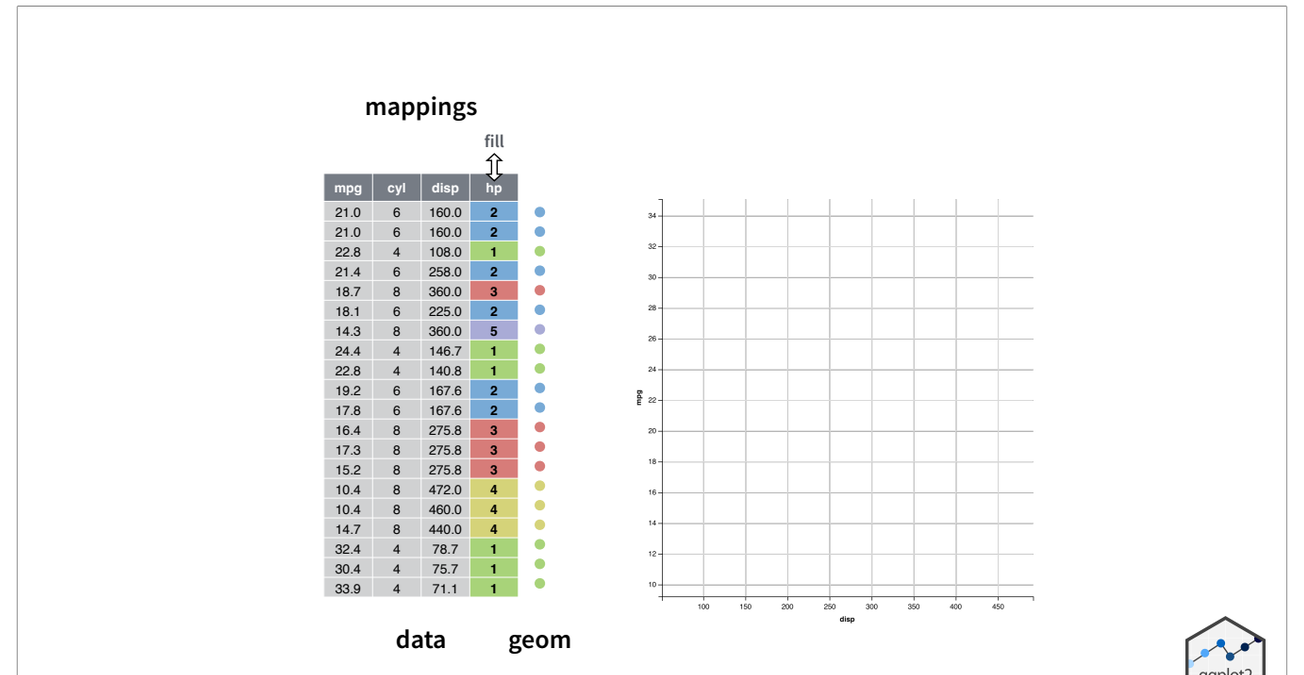
The Grammar of Graphics

Leland Wilkinson

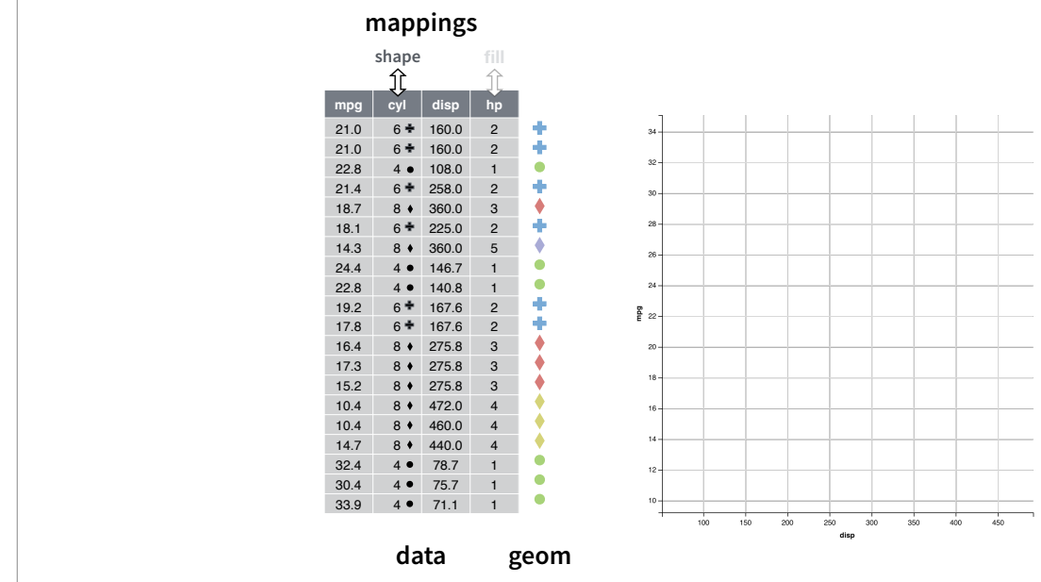
Many people think that Hadley invented the Grammar of Graphics. But this is not the case. In fact, you will often hear Hadley refer to ggplot2 as "An implementation of the Grammar of Graphics". The idea actually came from Leland Wilkinson. I believe he first published this idea in 1996, and it was an attempt to describe all possible statistical graphics. Wilkinson died in 2021.



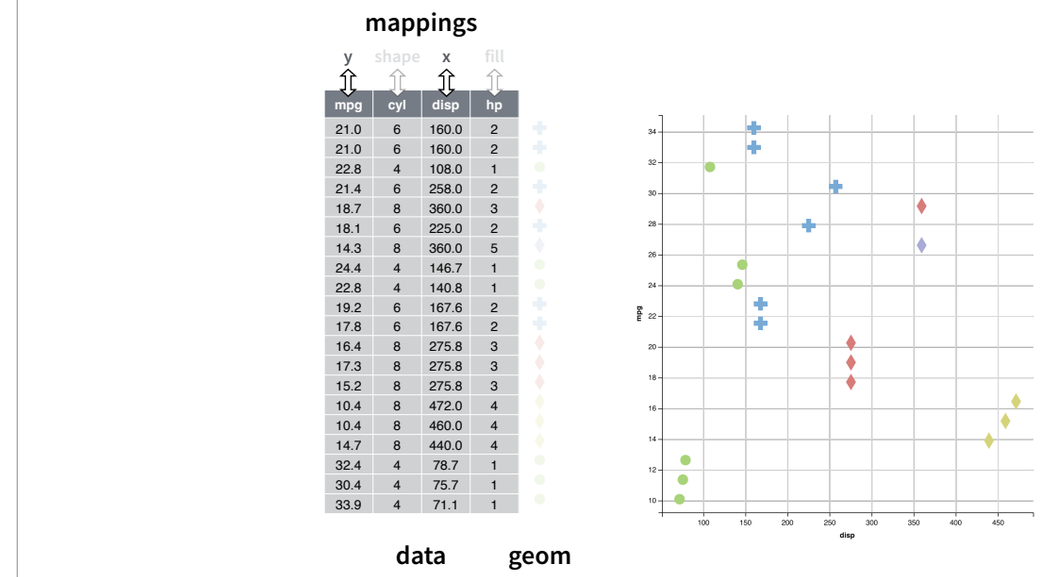
Each data point is represented by some geometric object. Here each point became a circle.



We can do aesthetic mappings as well. So we can take an attribute (here horsepower), and use it to change the color of the geom.

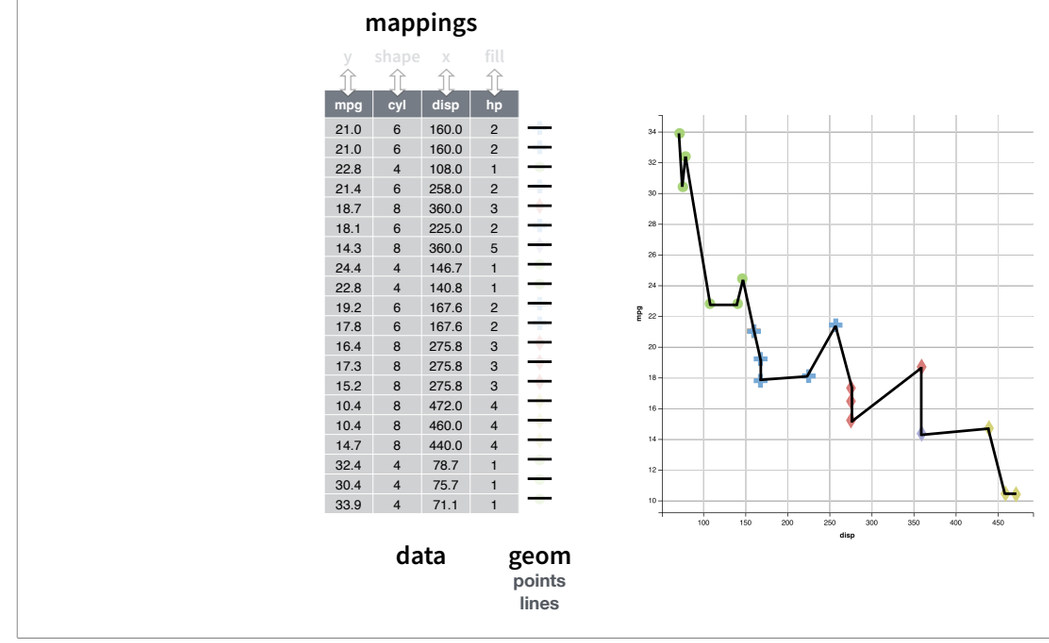


We can keep doing this. For example, we can use the number of cylinders to change the shape of the geom.

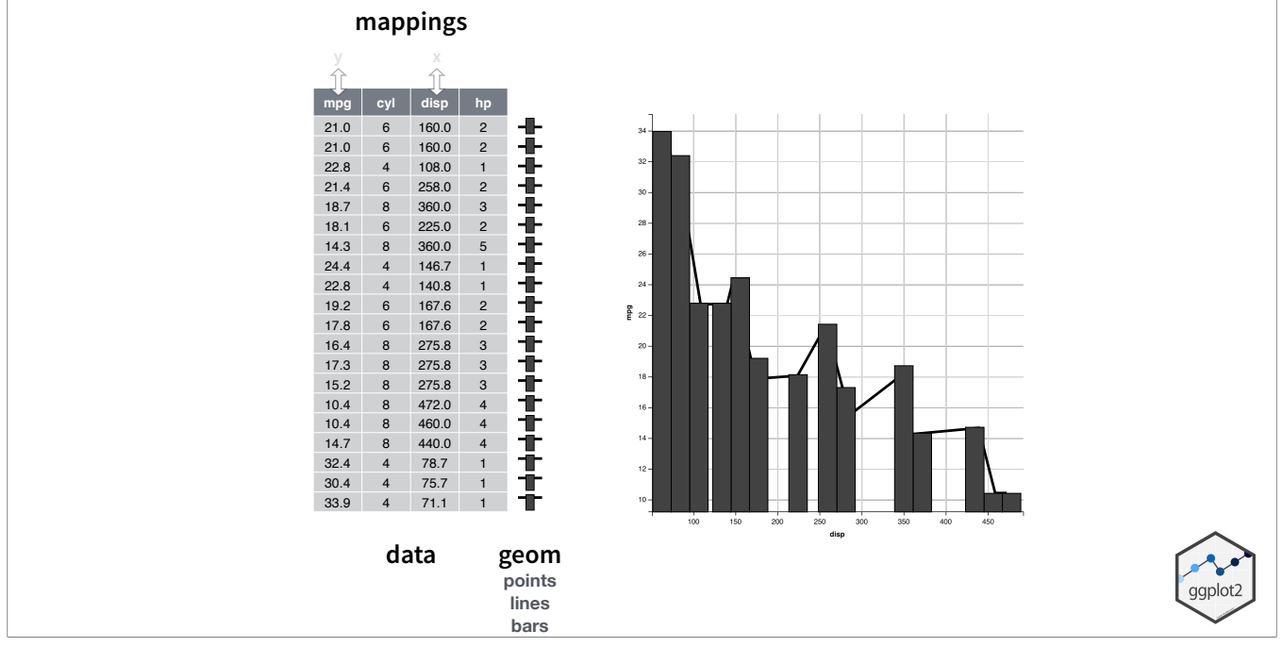


Generally speaking, though, things only get interesting when we map two properties to x and y.

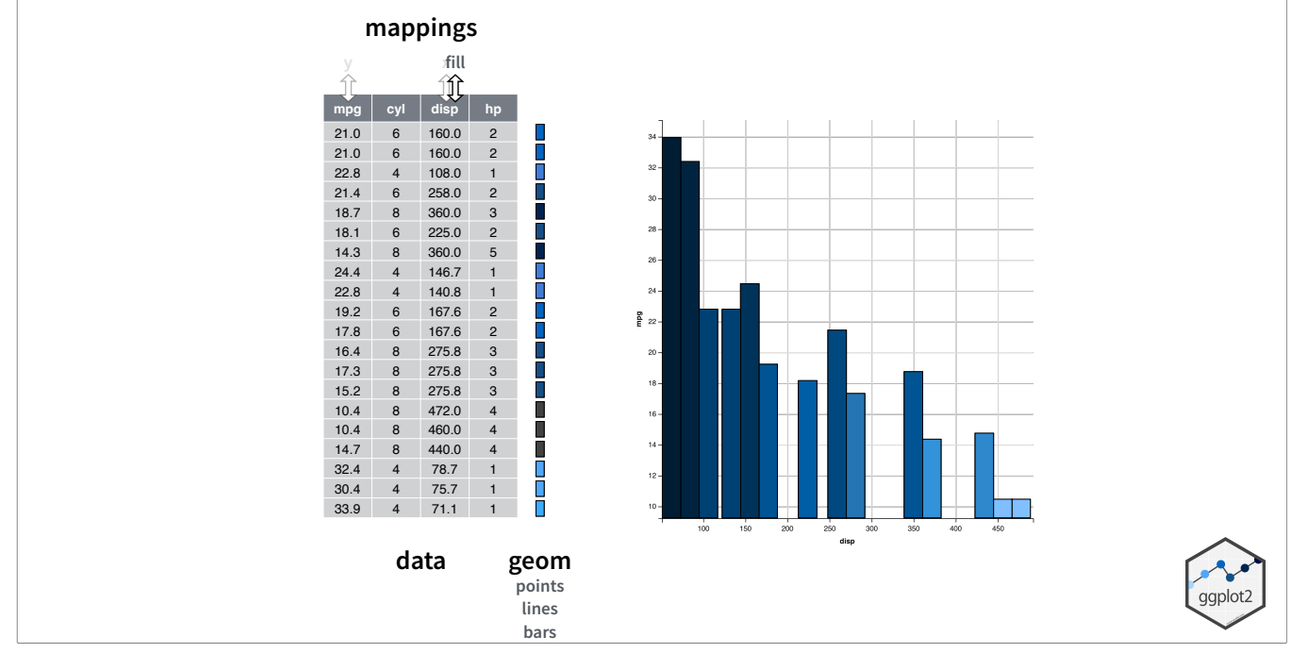
And I want to point out that this is an example of the power of tidy data. Every row is a case, every column is a variable, and every cell contains data.



What makes ggplot special is how easy it is to add and experiment with geometric layers. So here we decided to keep our colored shapes, but also add in lines.



And we can also add in bars



And even color the bars as well

To make a graph

[template]

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

I mentioned this earlier, but the key for you to remember is that it all of ggplot2 comes down to this single template.

To make a graph

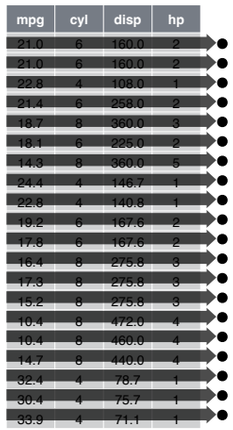
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data

1. Pick a **data** set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

To make a graph



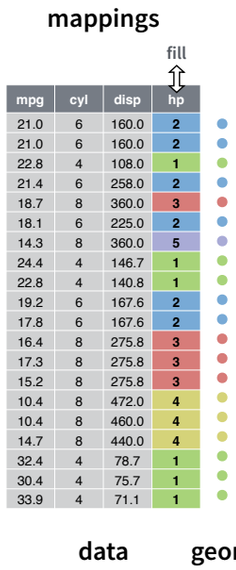
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

1. Pick a **data** set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom**
   to display cases

To make a graph



1. Pick a **data** set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

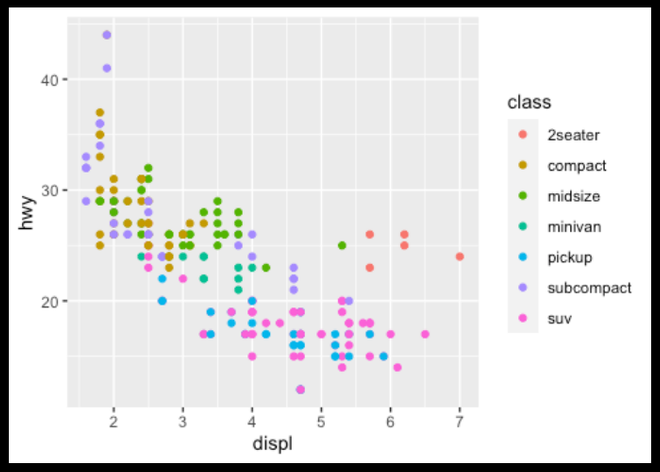
2. Choose a **geom** to display cases

3. **Map** aesthetic properties to variables

What else?

Faceting

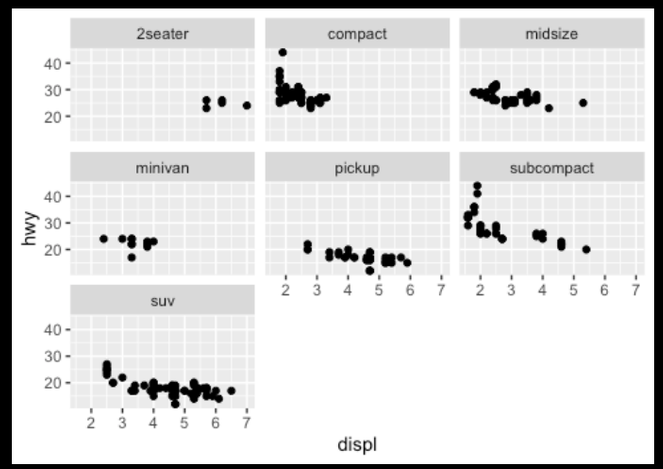
(AKA "One graph for each sub-group")




```
ggplot(mpg) +
    geom_point(aes(displ, hwy, color=class))
```

This is a graph that we've seen a lot today. It's a scatterplot that shows Highway Mileage by Engine Displacement. It goes further, and colors the dots by the class of car.

To be honest (and this is perhaps because I'm color blind), I have trouble reading this graph. It would be easier for me if ggplot created one scatterplot for each class

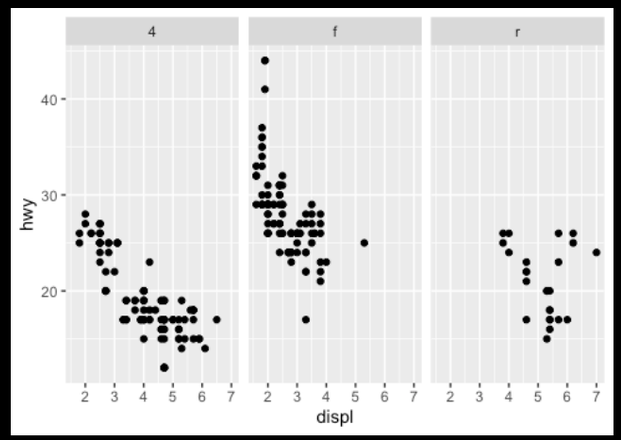



```
ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  facet_wrap(~class)
```

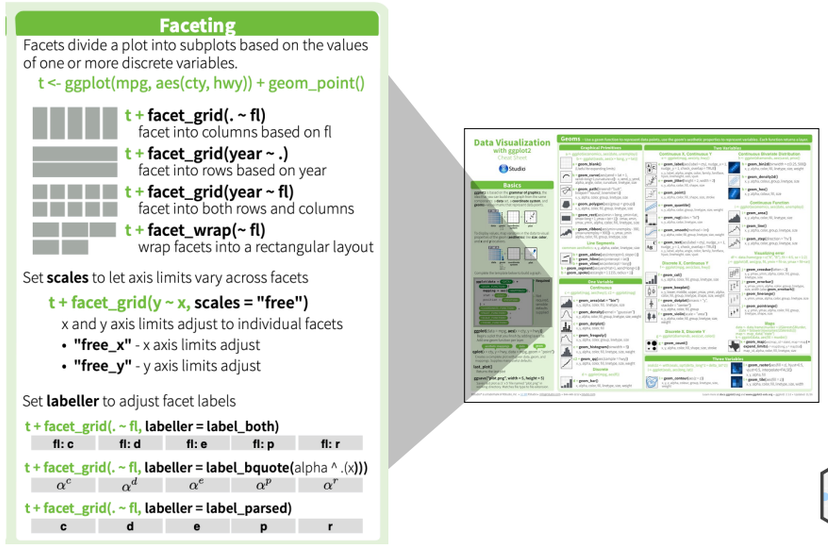
In ggplot2, "facet-ing" is the process of dividing a plot into sub-plots based on a variable. Here the variable is class (which we type as ~class). And the faceting function we use is facet_wrap. Note that the x- and y-axis are the same for all facets.

Your Turn 8

- Use **facet_wrap** to create this graph
- For mpg, it shows displ vs. hwy.
- It is faceted by drv
- 5 Minutes



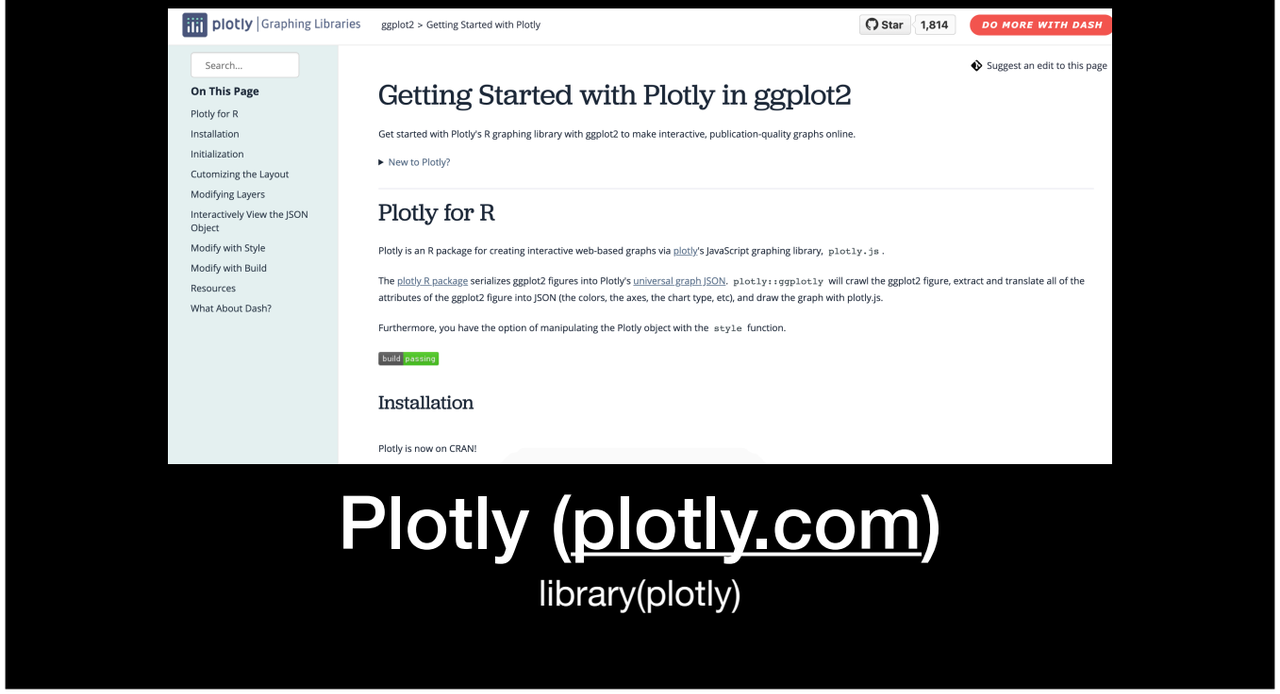
More on Faceting



The cheat sheet I gave you has more information on faceting. While there are more options, the key idea is the same: you are making multiple plots from a single dataset.

Interactive Graphics

ggplot2 is probably the most popular package in all of R for making graphics. It's main limitation, however, is that it does not make interactive graphics. To do that, we need to look elsewhere.



Plotly (plotly.com)

library(plotly)

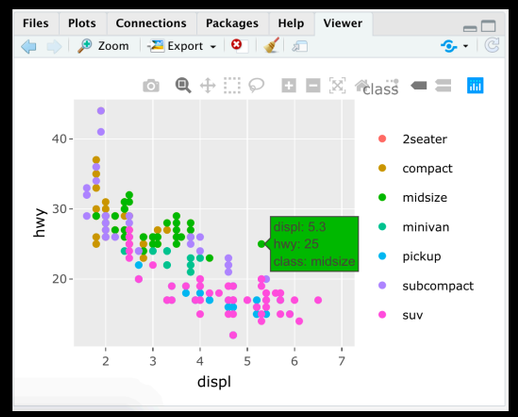
To get started with interactive graphics in R, I recommend using the library plotly. Plotly is a company that, among other things, makes a graphics library for R.

?ggplotly

```
library(plotly)

chart = ggplot(mpg) +
  geom_point(aes(displ, hwy, color=class))

ggplotly(chart)
```

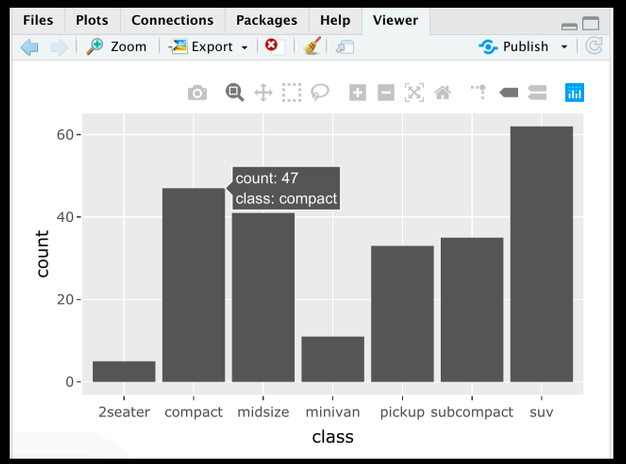


What makes plotly special is that they invented a way to make any chart created with ggplot2 interactive. This is an example. It's the same chart we've been looking at today, but when I roll over a point, I get additional information.

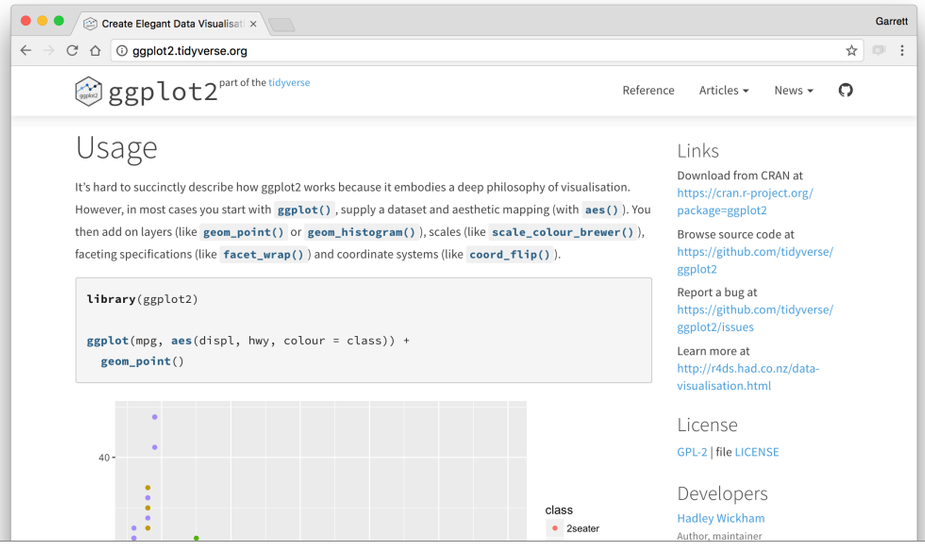
To create this chart, just create it like normal. But instead of printing it, save the results in a variable. And then pass the variable to the function **ggplotly** in the **plotly** library. Note that plotly is not part of the tidyverse - you need to install and load it separately.

Your Turn 10

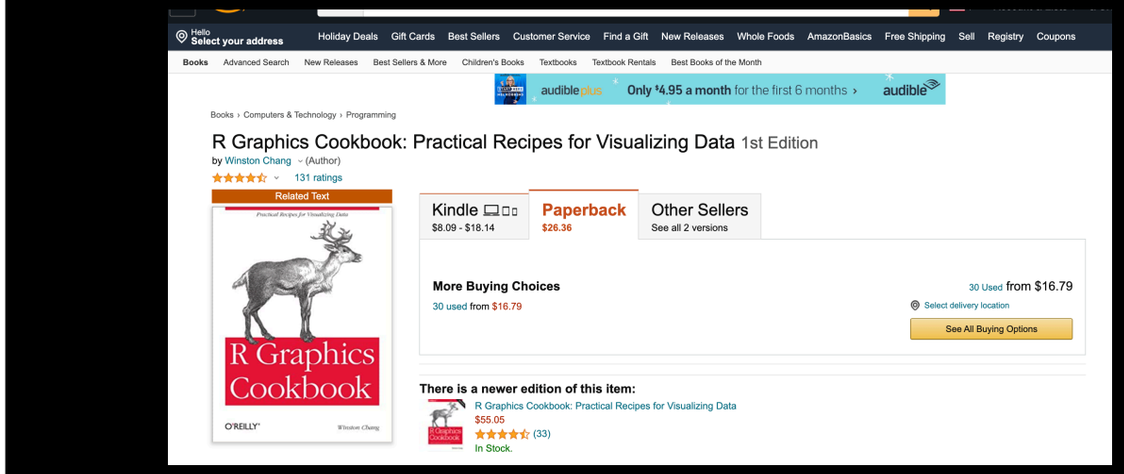
- I gave you code to create the static version of this chart
- Make it interactive by passing the ggplot2 object to **?ggplotly**
- You will need to load the **plotly** library.
- 5 Minutes



ggplot2.tidyverse.org



You've now had a lot of experience with the basic elements of ggplot2 - aesthetics and geometry objects. As you've probably guessed, there is a lot more to learn. A great place to get started is the official ggplot2 website on tidyverse.org: ggplot2.tidyverse.org.



R Graphics Cookbook

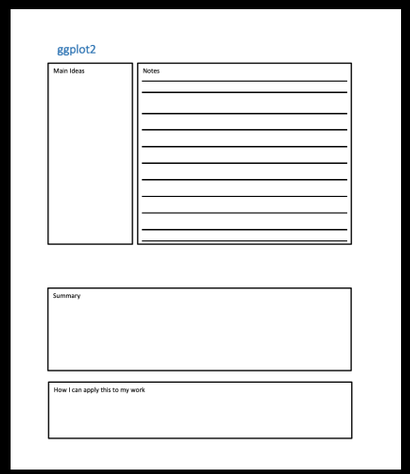
Winston Chang

ggplot was my first introduction to R, and my mapping packages use it extensively. I found this book, the R Graphics Cookbook by Winston Chang, to be very helpful. I believe that Winston was one of Hadley's PhD students, and now works at RStudio.

Closing Exercise

- Fill out the "Summary" section
- Fill out the "How I can apply this to my work"
- Share with your neighbor!

03:00


ggplot2

Main Ideas

Notes

Summary

How I can apply this to my work

Any Questions?