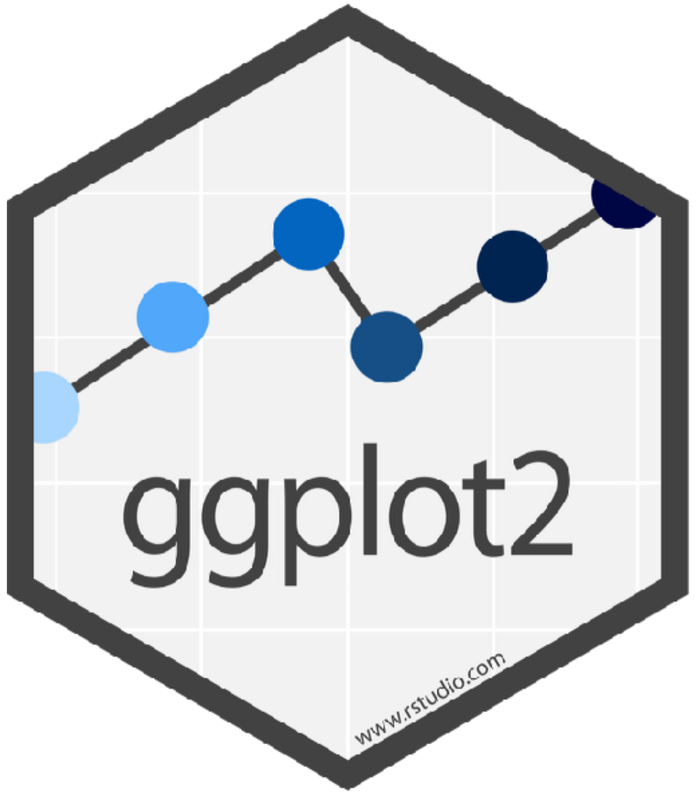
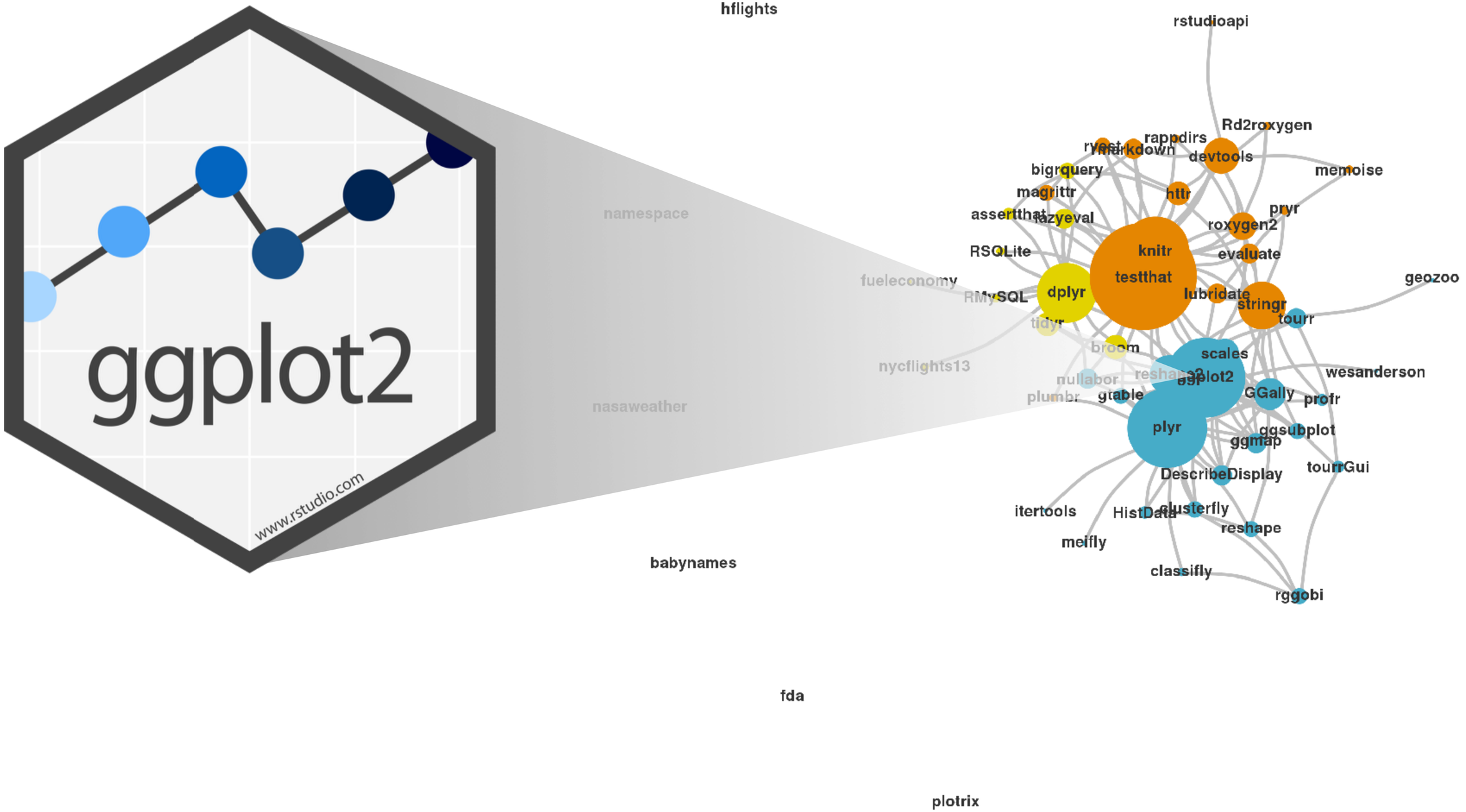
Visualize Data with



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

- John Tukey



 Display by Adolfo Álvarez

Setup

The setup chunk is always run once before anything else

mpg

Fuel economy data for 38 models of car.

```
mpg
```

Quiz

Confer with your group.

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

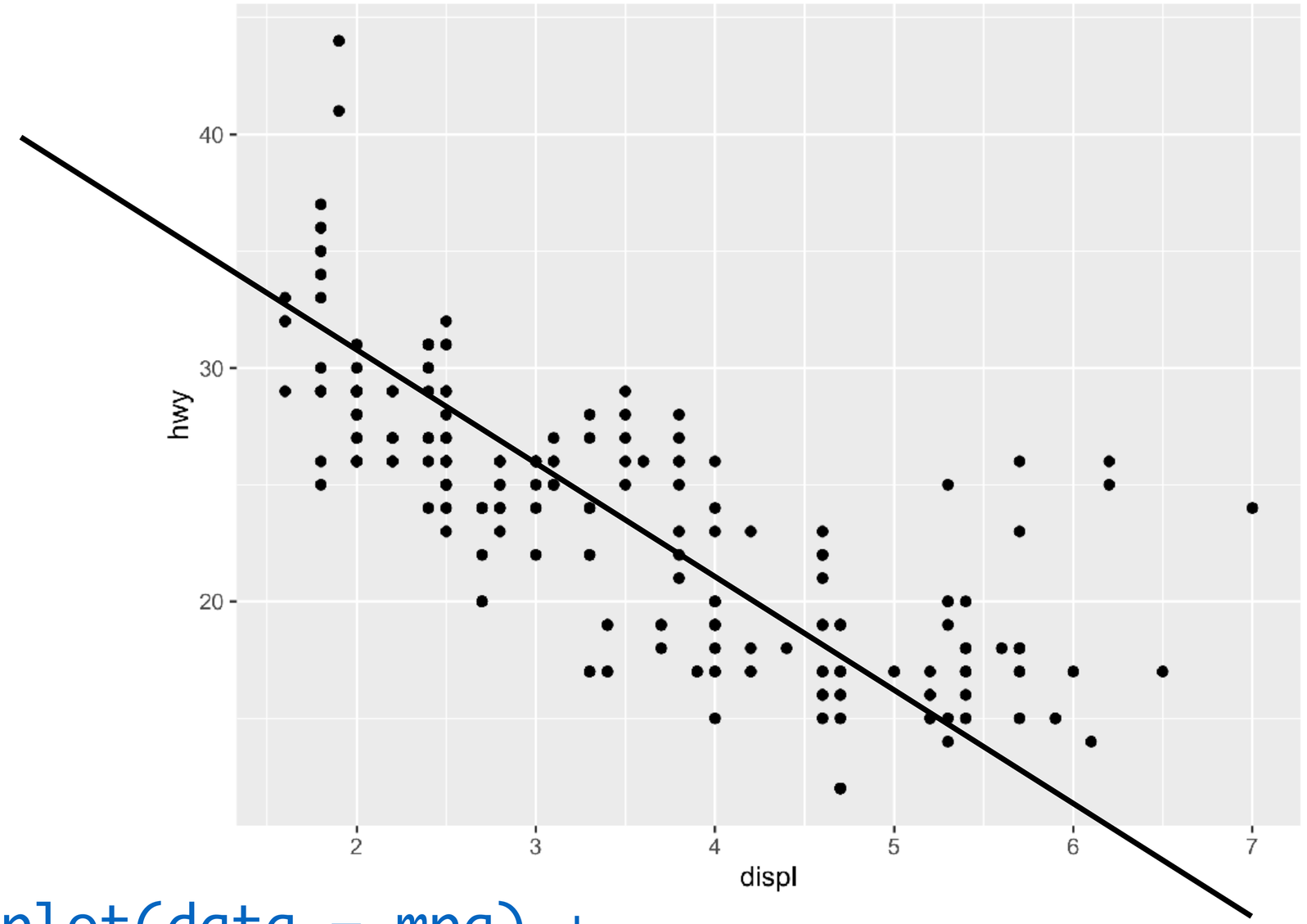
No peeking ahead!

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))
```




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

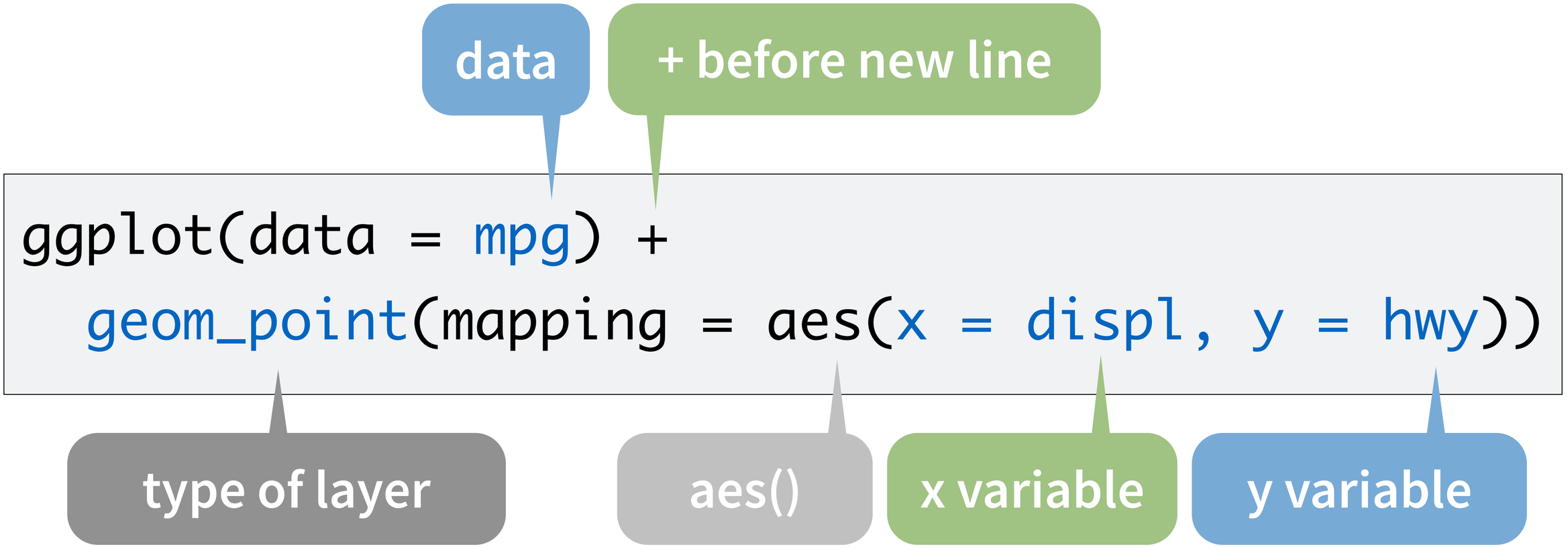
1. "Initialize" a plot with ggplot()

2. Add layers with geom_ functions

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

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))
```



A template

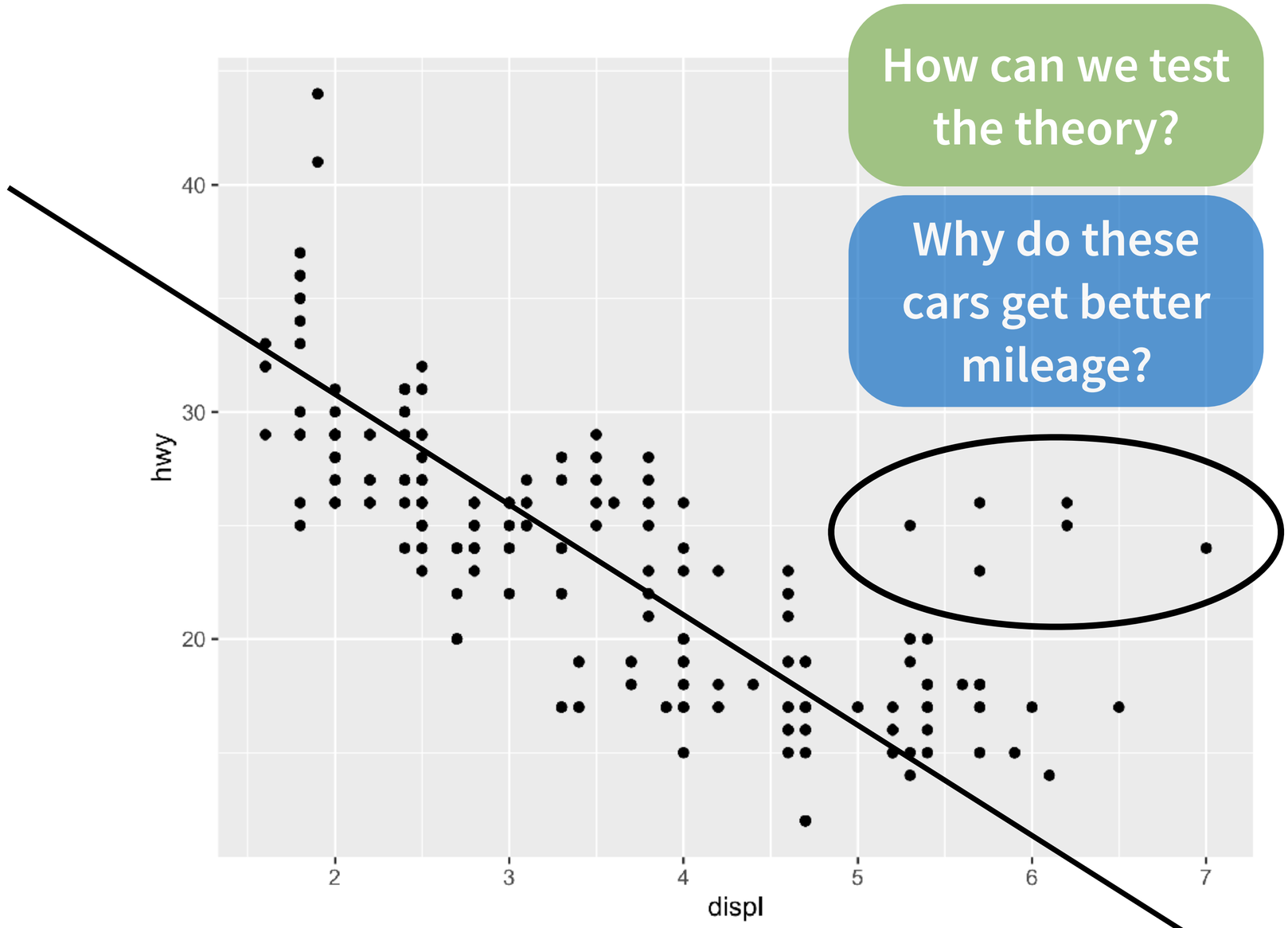
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

```
  geom_point(mapping = aes(x = displ, y = hwy))
```

A template

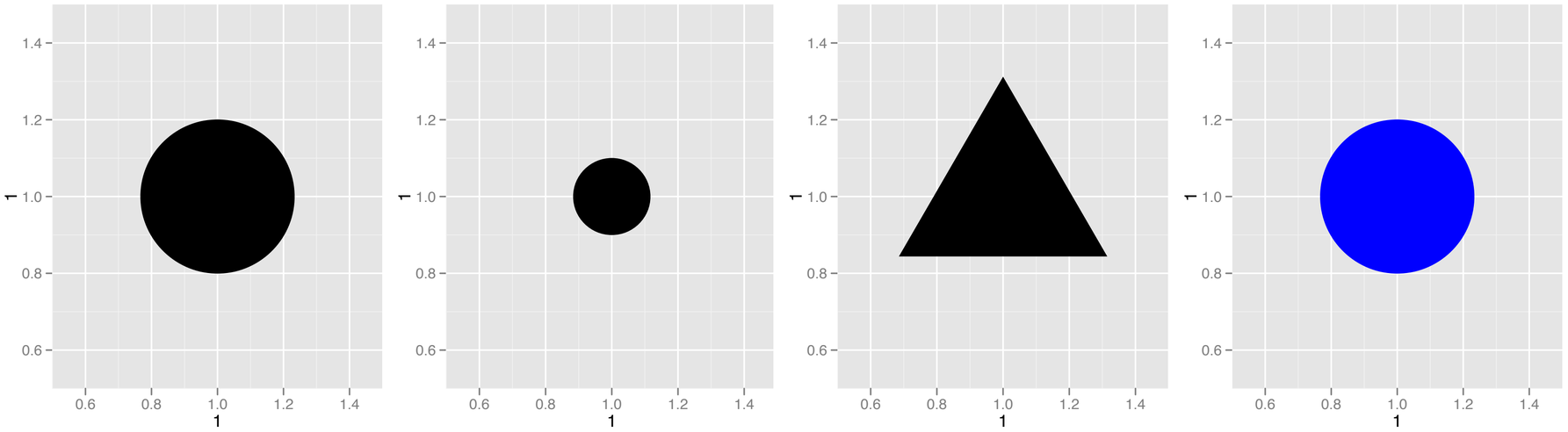
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

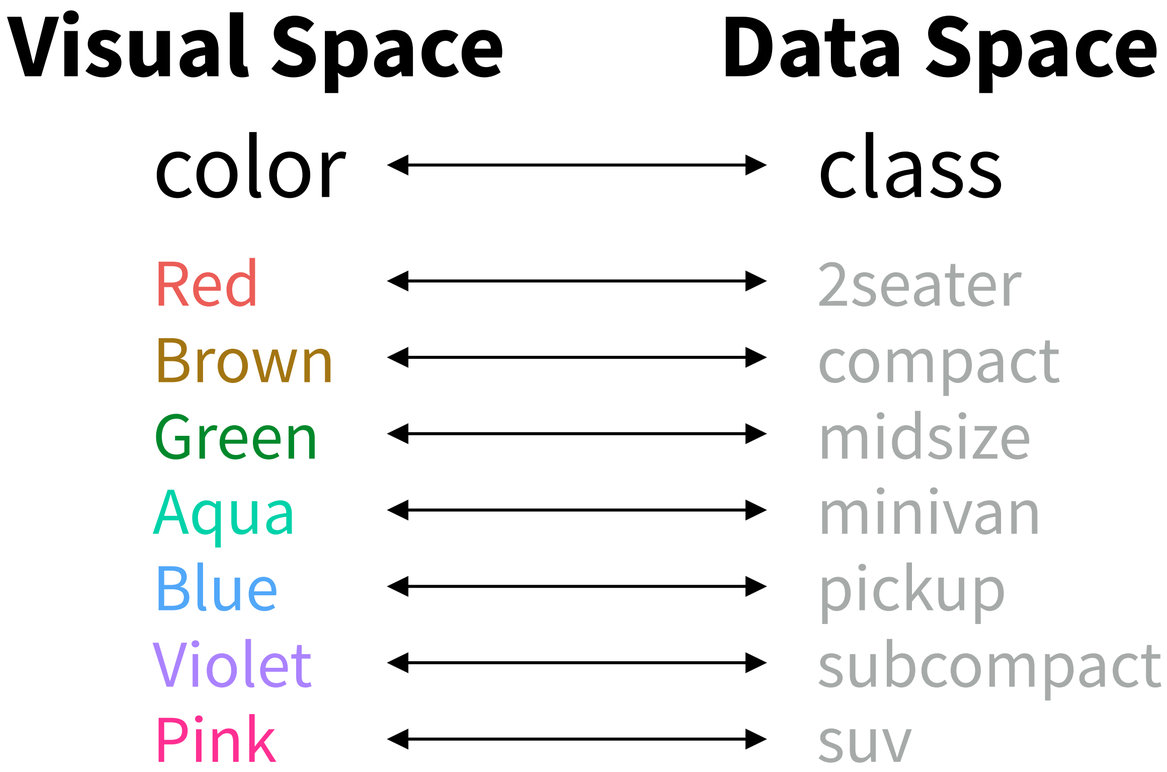

Mappings




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

Aesthetics



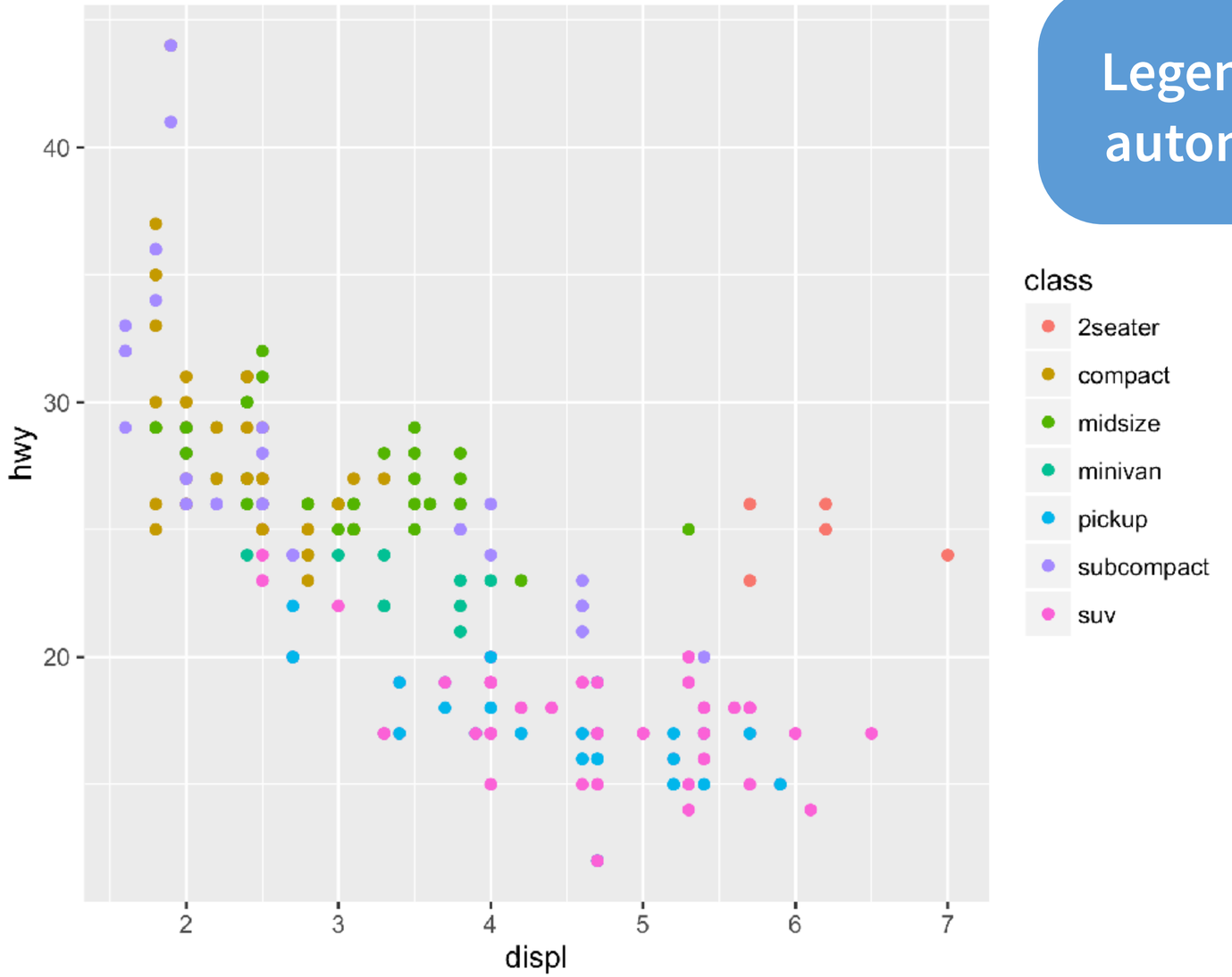


Aesthetics

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))
```



Legend added automatically

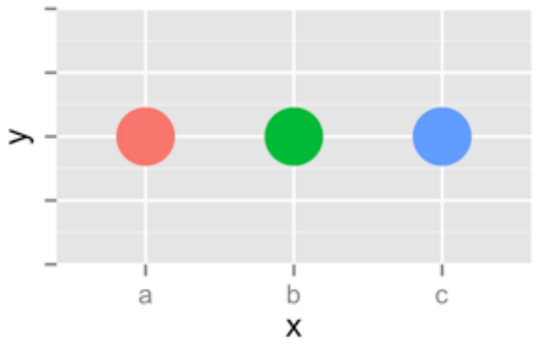
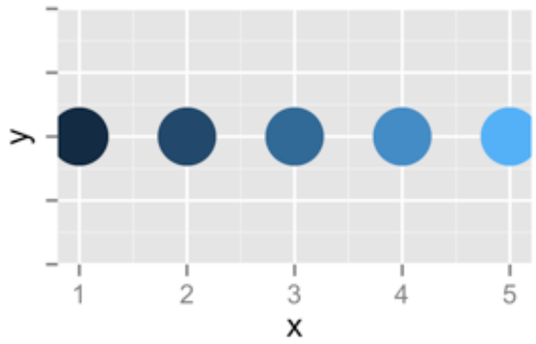
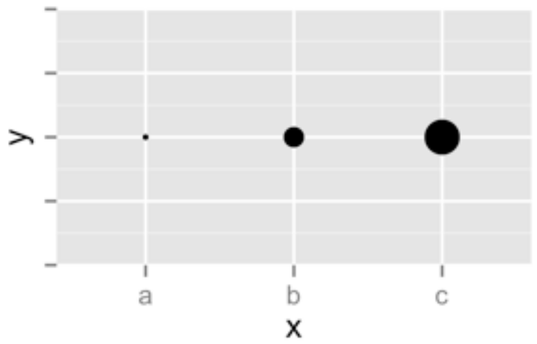
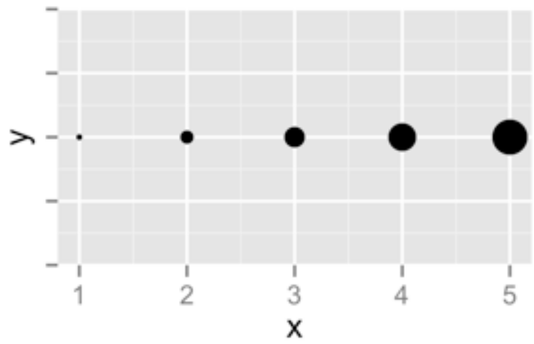
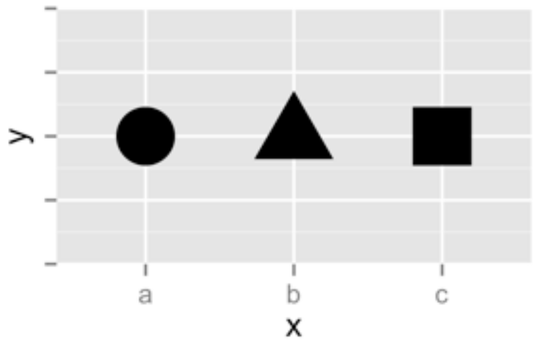
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

Your Turn 2

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

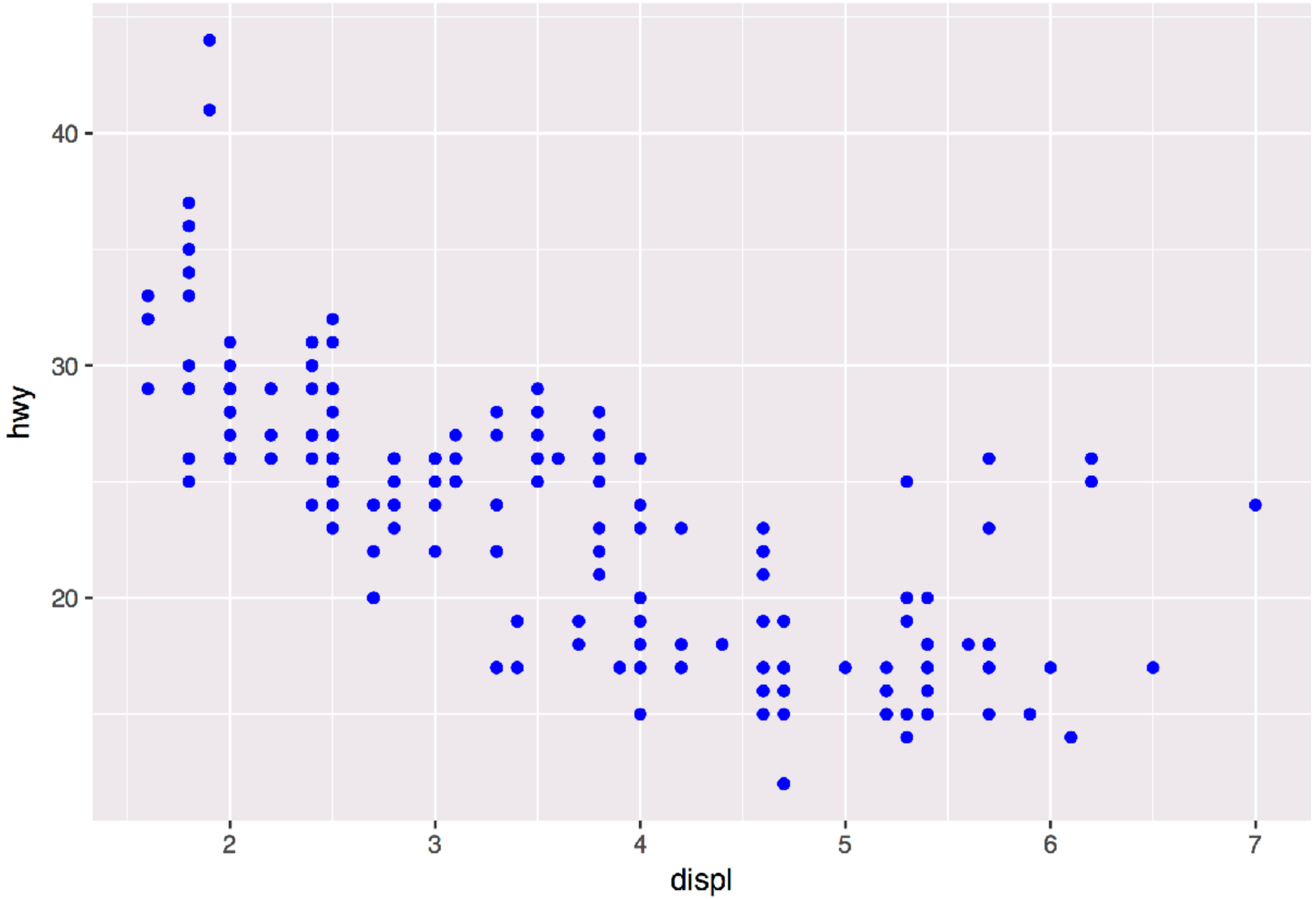
Do different things happen when you map aesthetics to discrete and continuous variables?

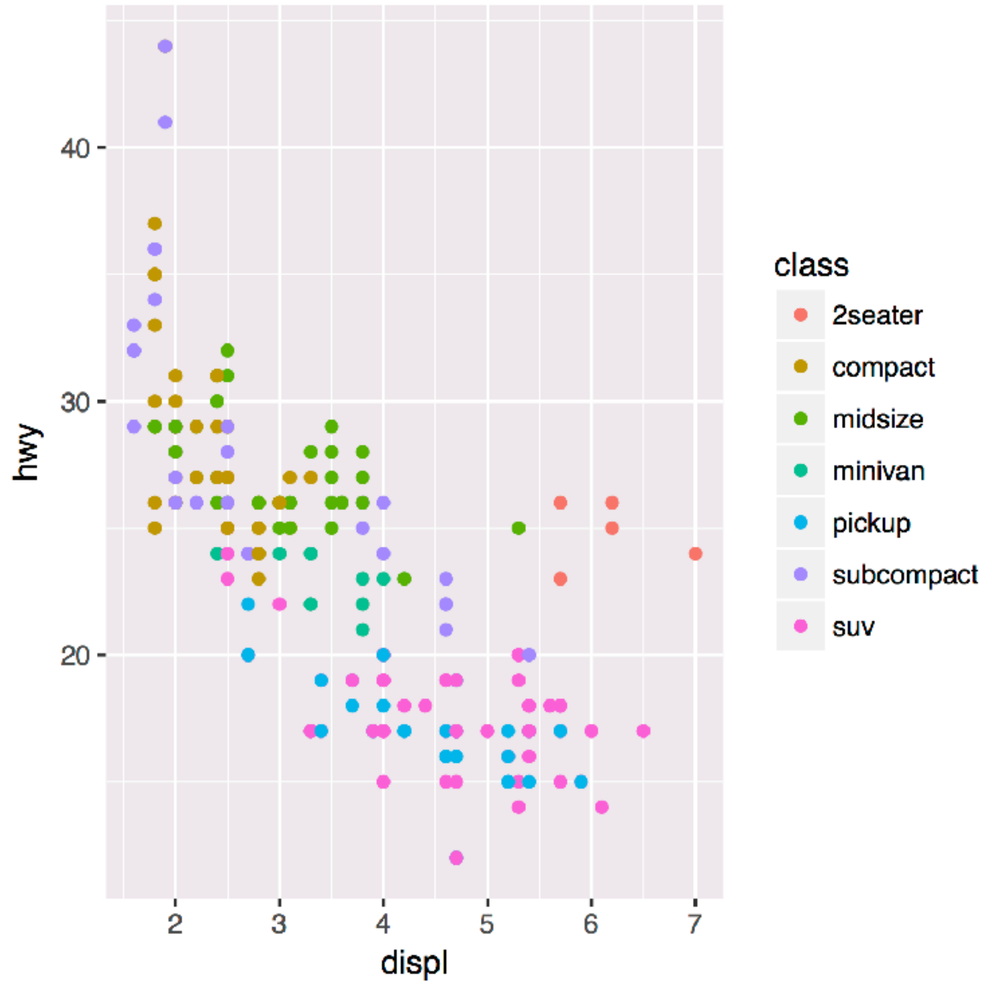
What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

set vs. map

How would you make this plot?

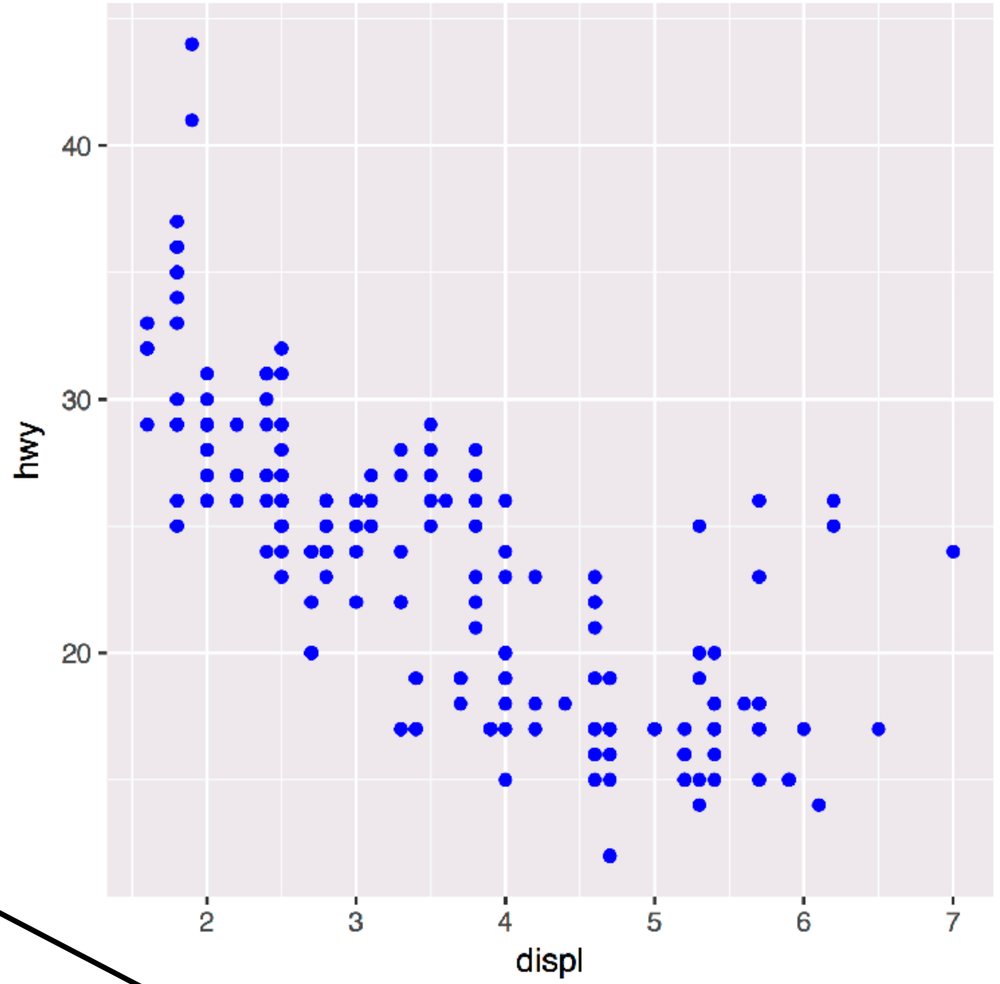




Inside of aes(): maps an aesthetic to a variable

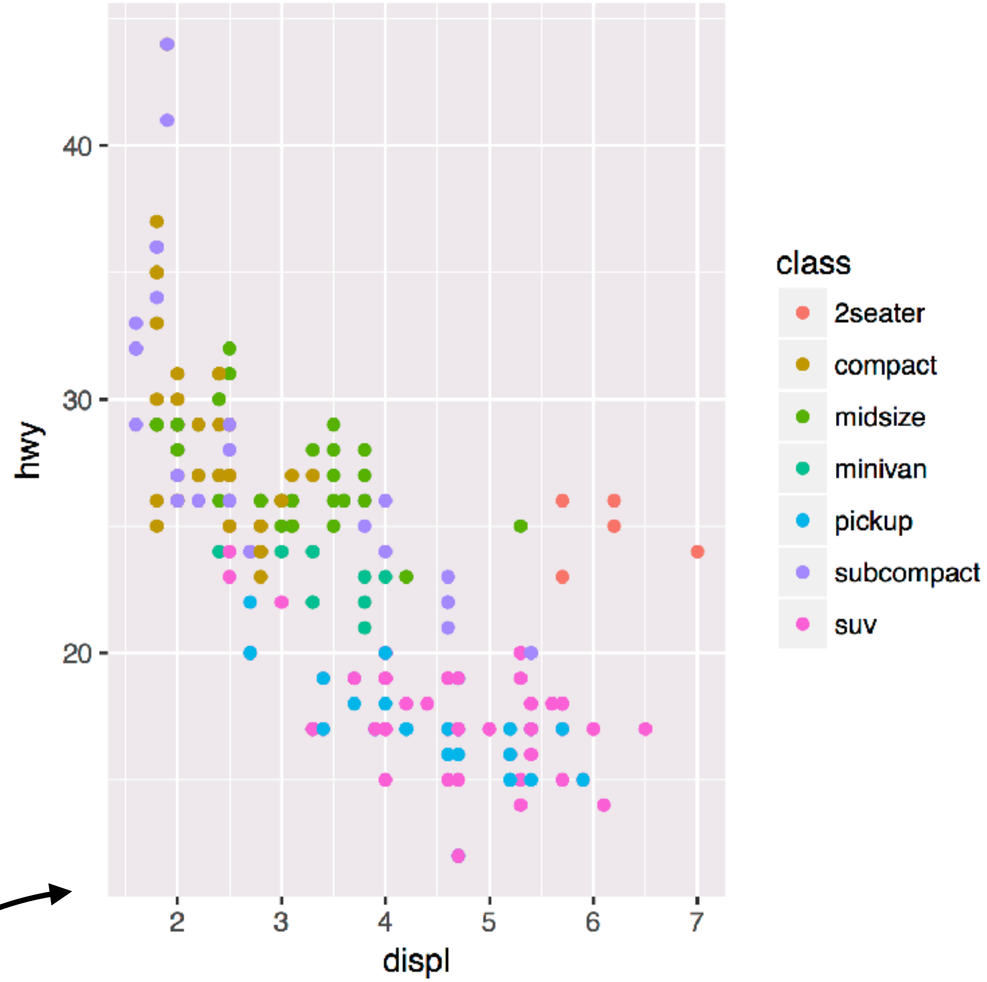
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

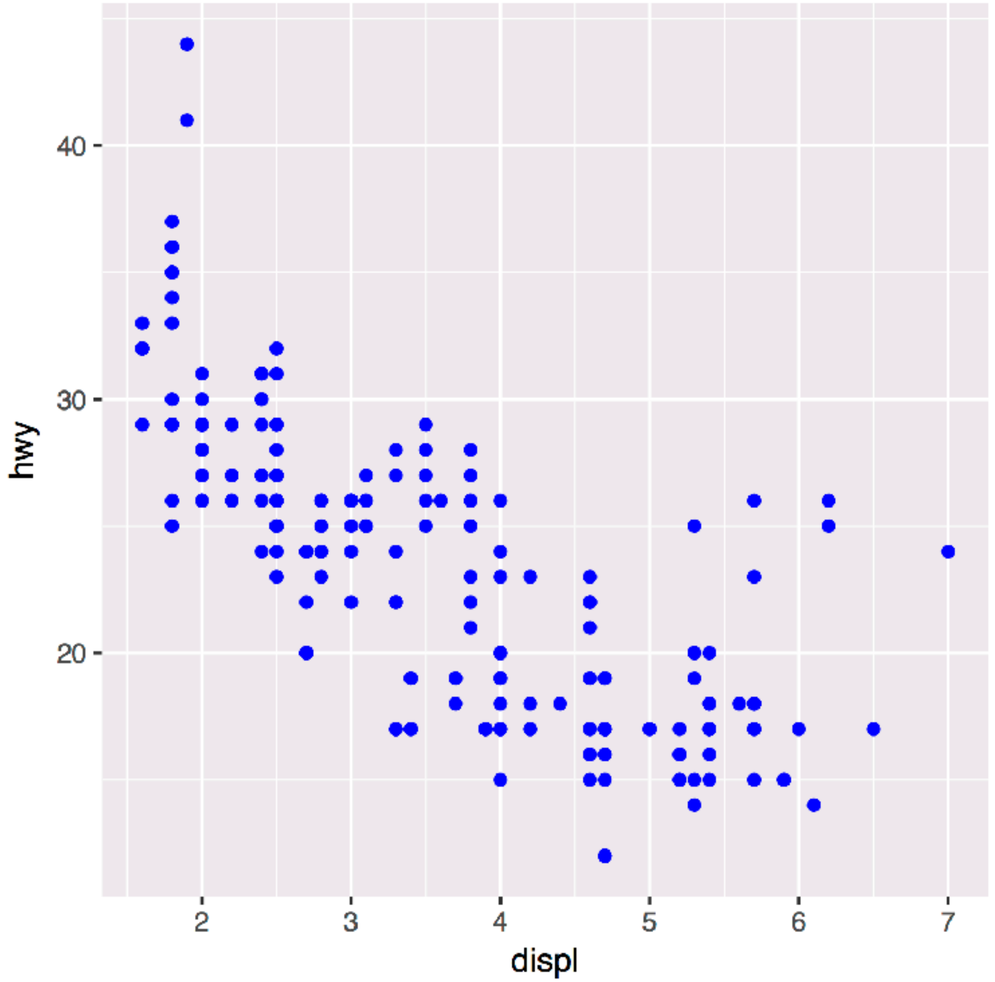
Outside of aes(): sets an aesthetic to a value




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

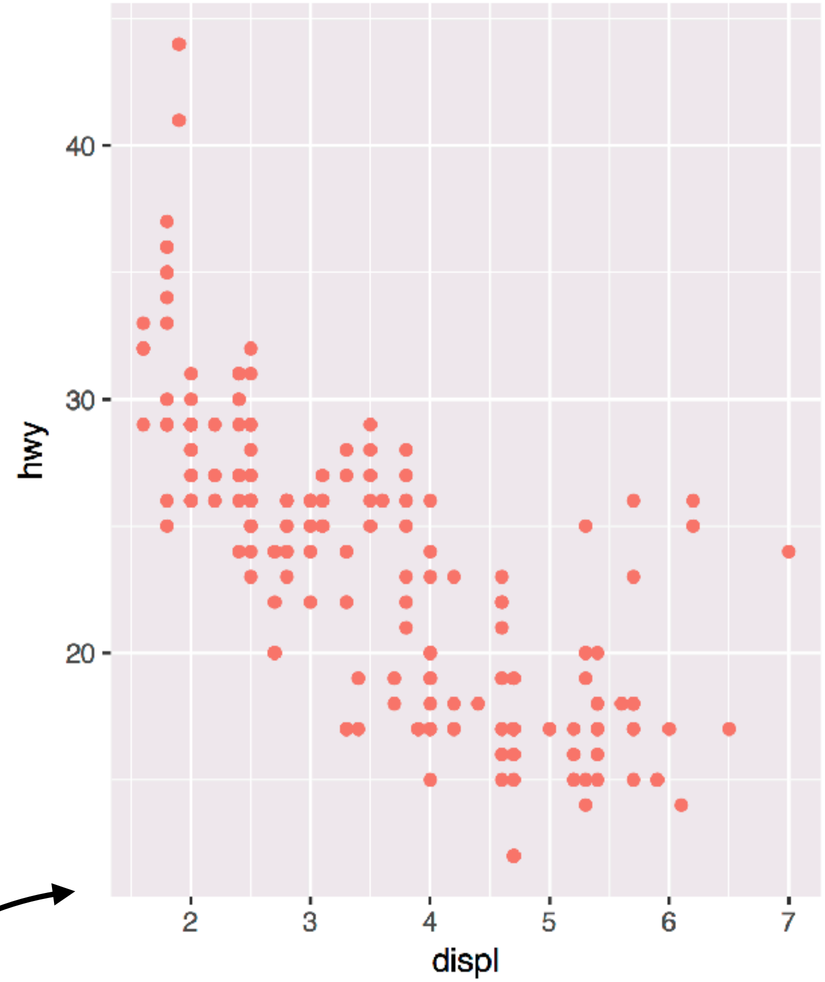
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), color = "blue")
```

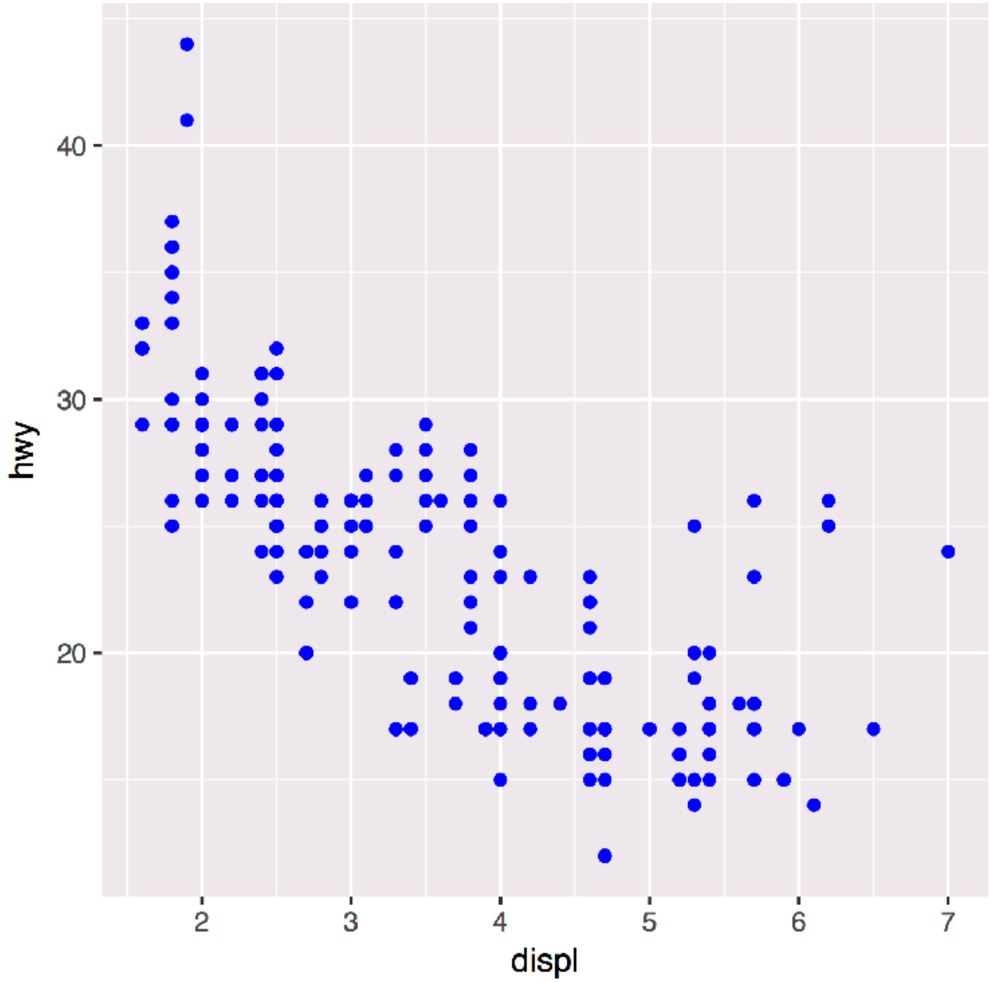





```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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





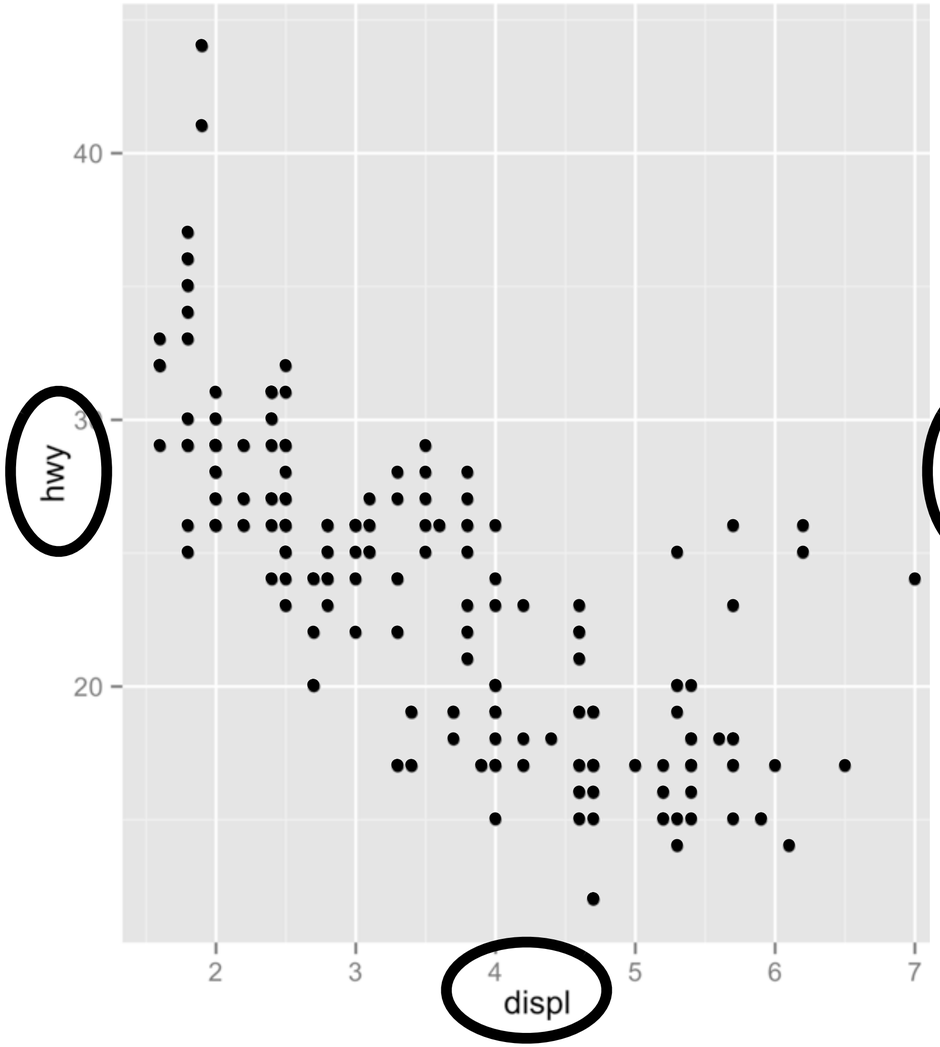

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = "blue"))
```

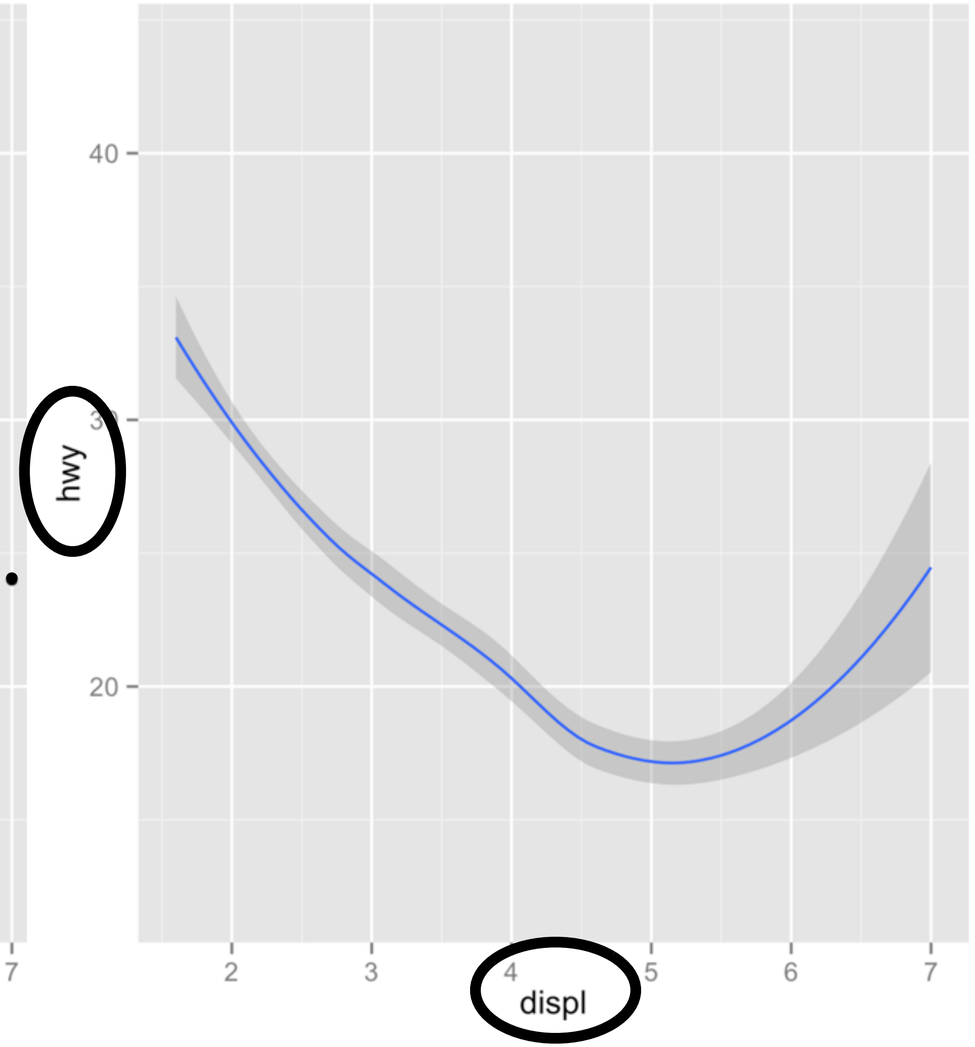
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), color = "blue")
```

Geoms

How are these plots similar?

Same: x var , y var , data



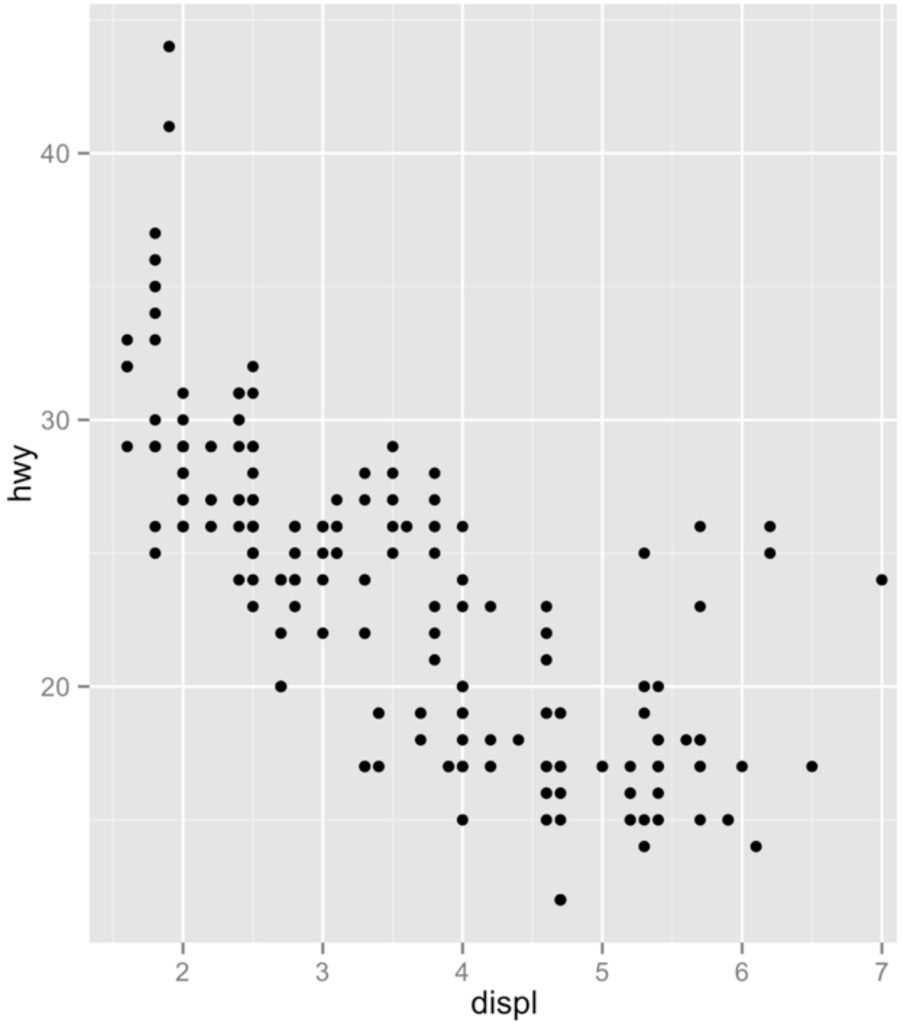


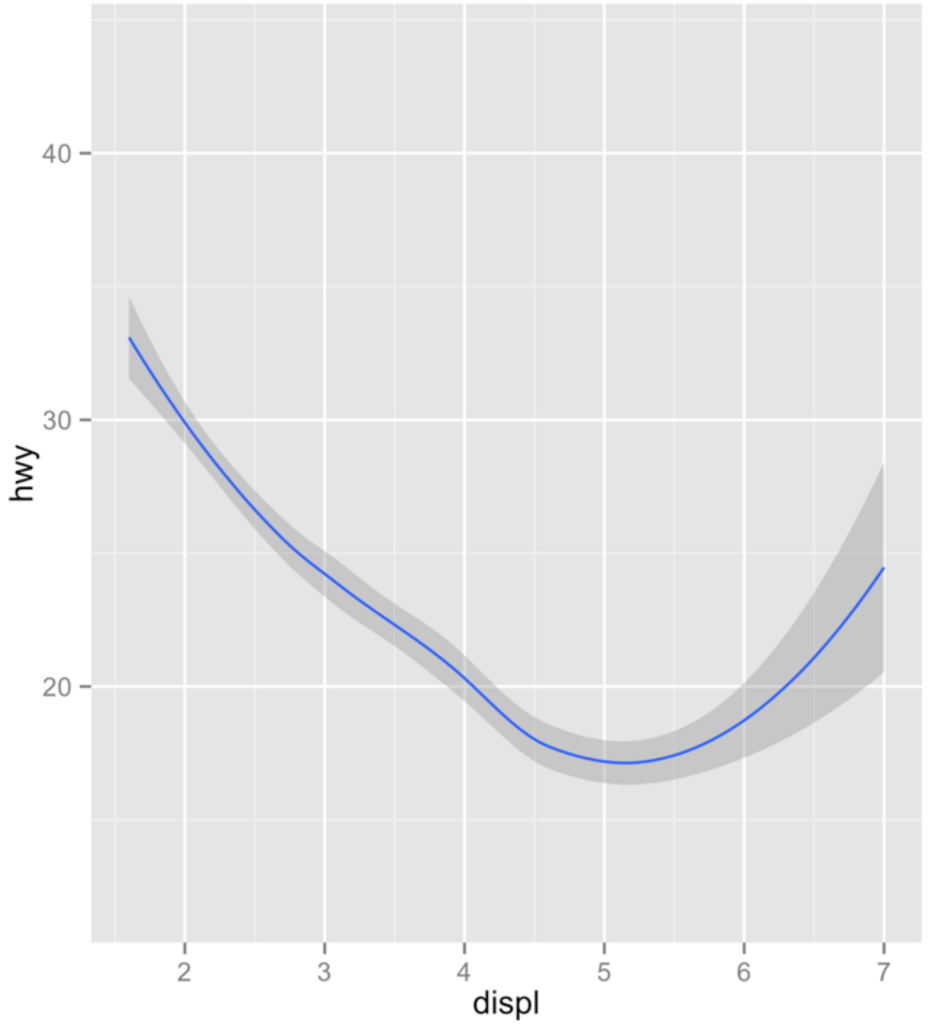
ggplot2

How are these plots different?

Different: geometric object (geom),

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



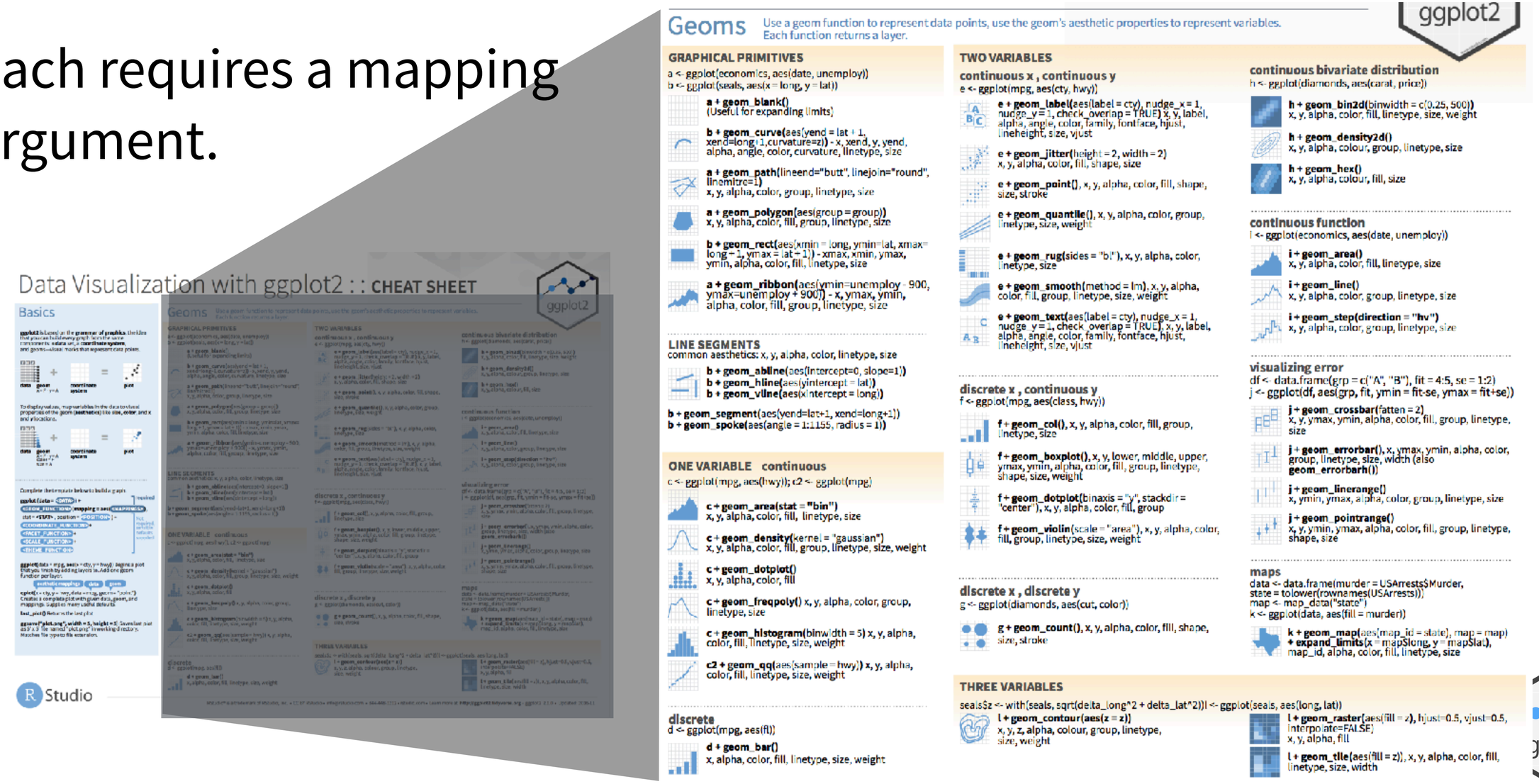


geoms

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

geom_ functions

Each requires a mapping argument.

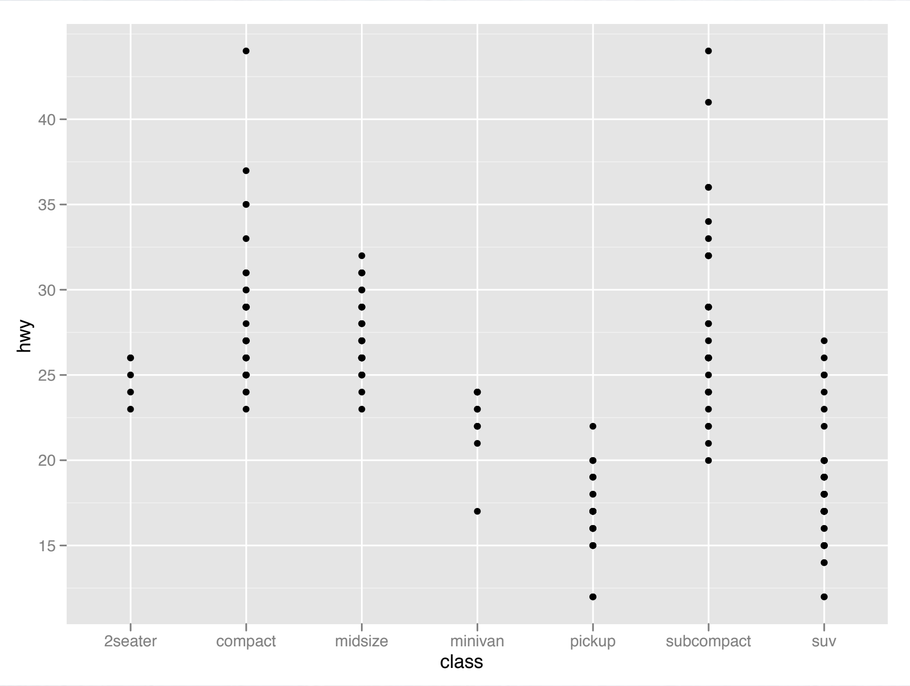


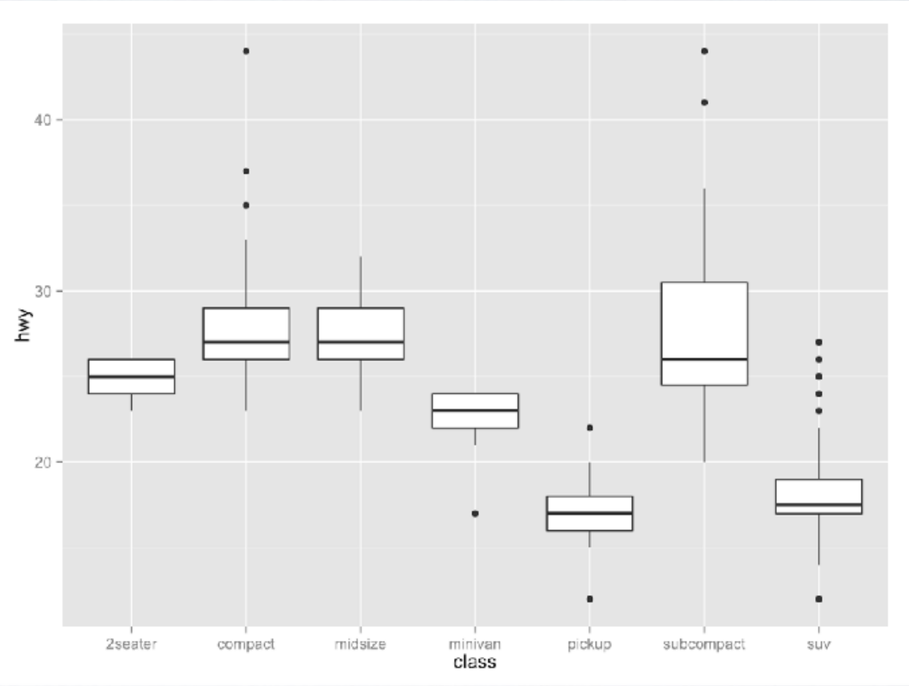
Your Turn

Pair up.

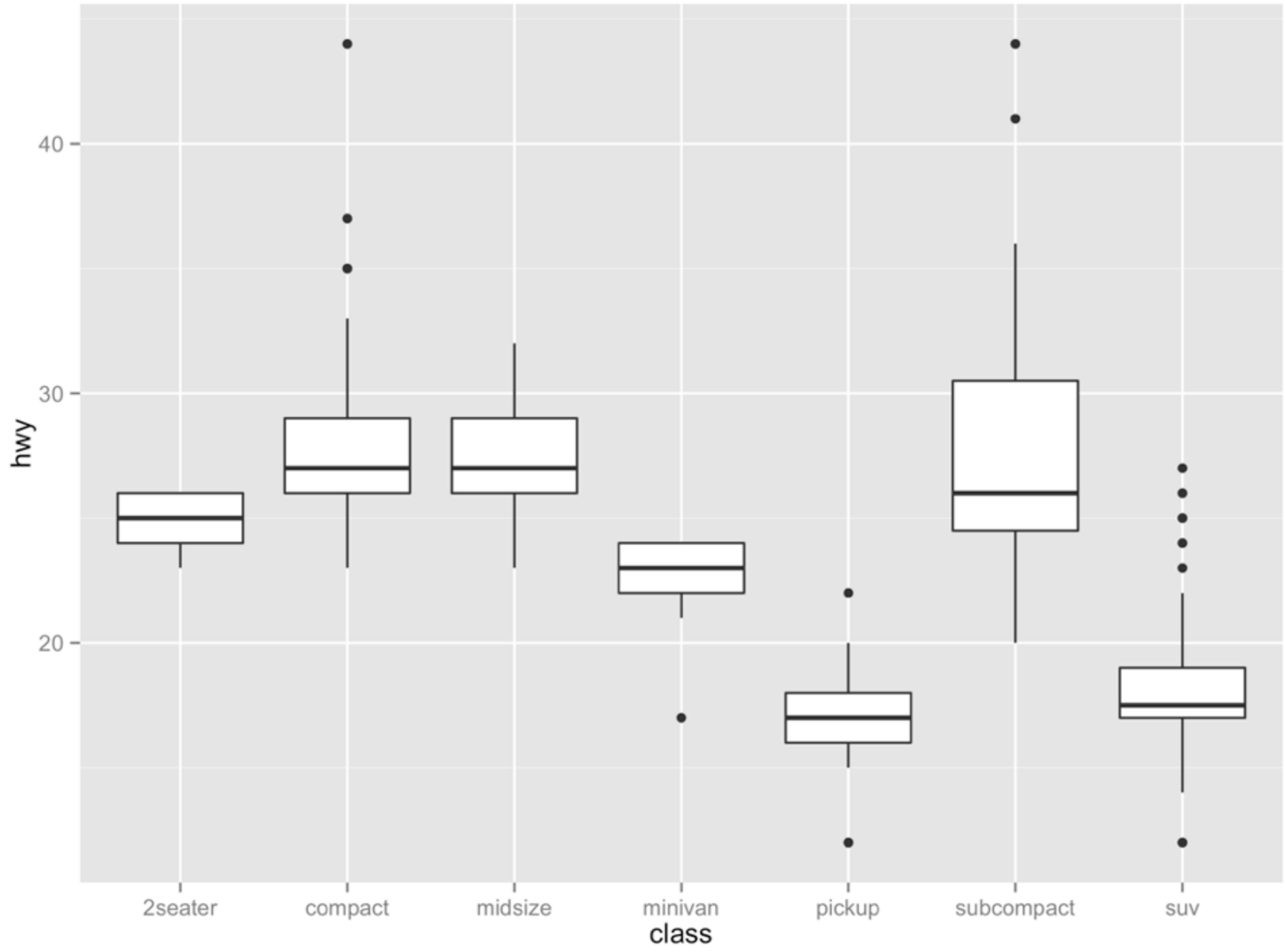
Your Turn 3

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





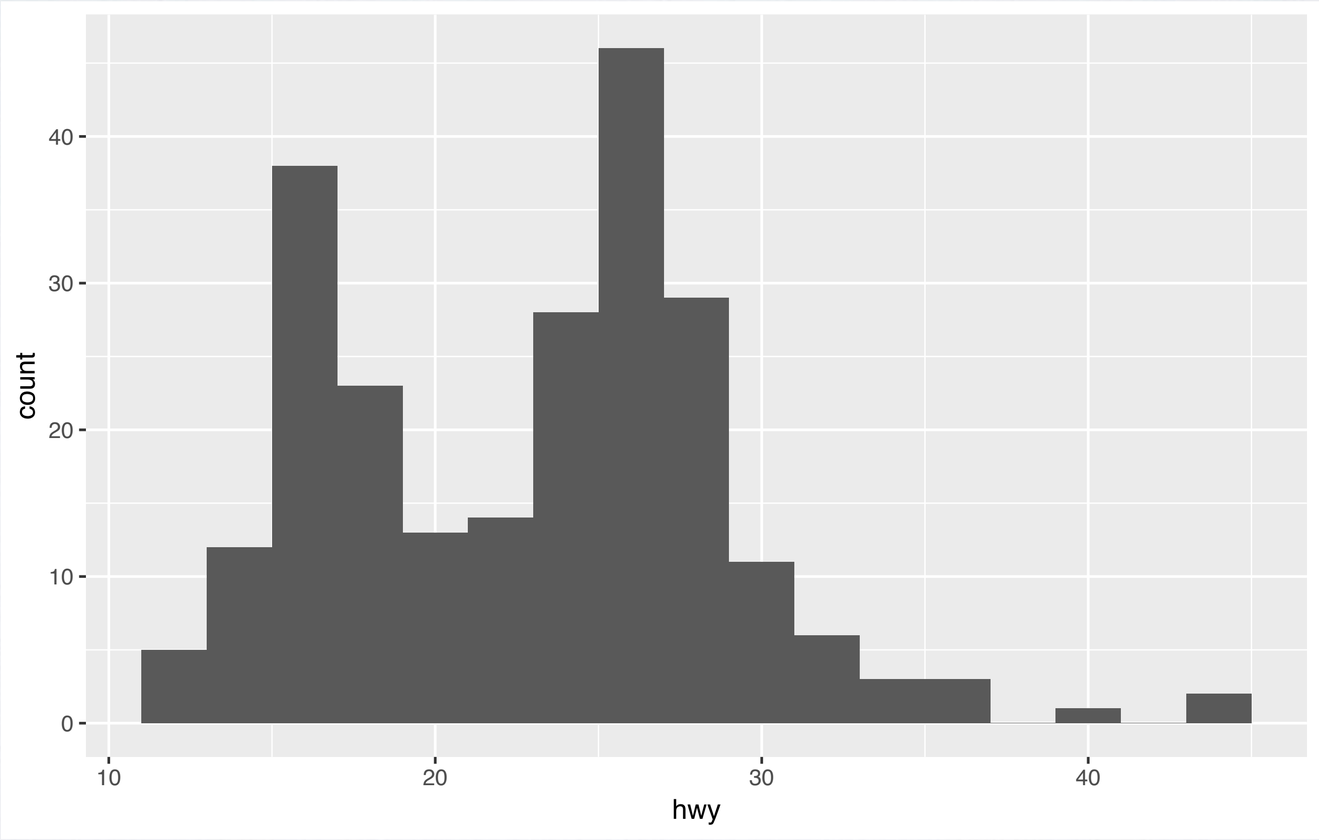

```
ggplot(mpg) + geom_point(aes(class, hwy))
```

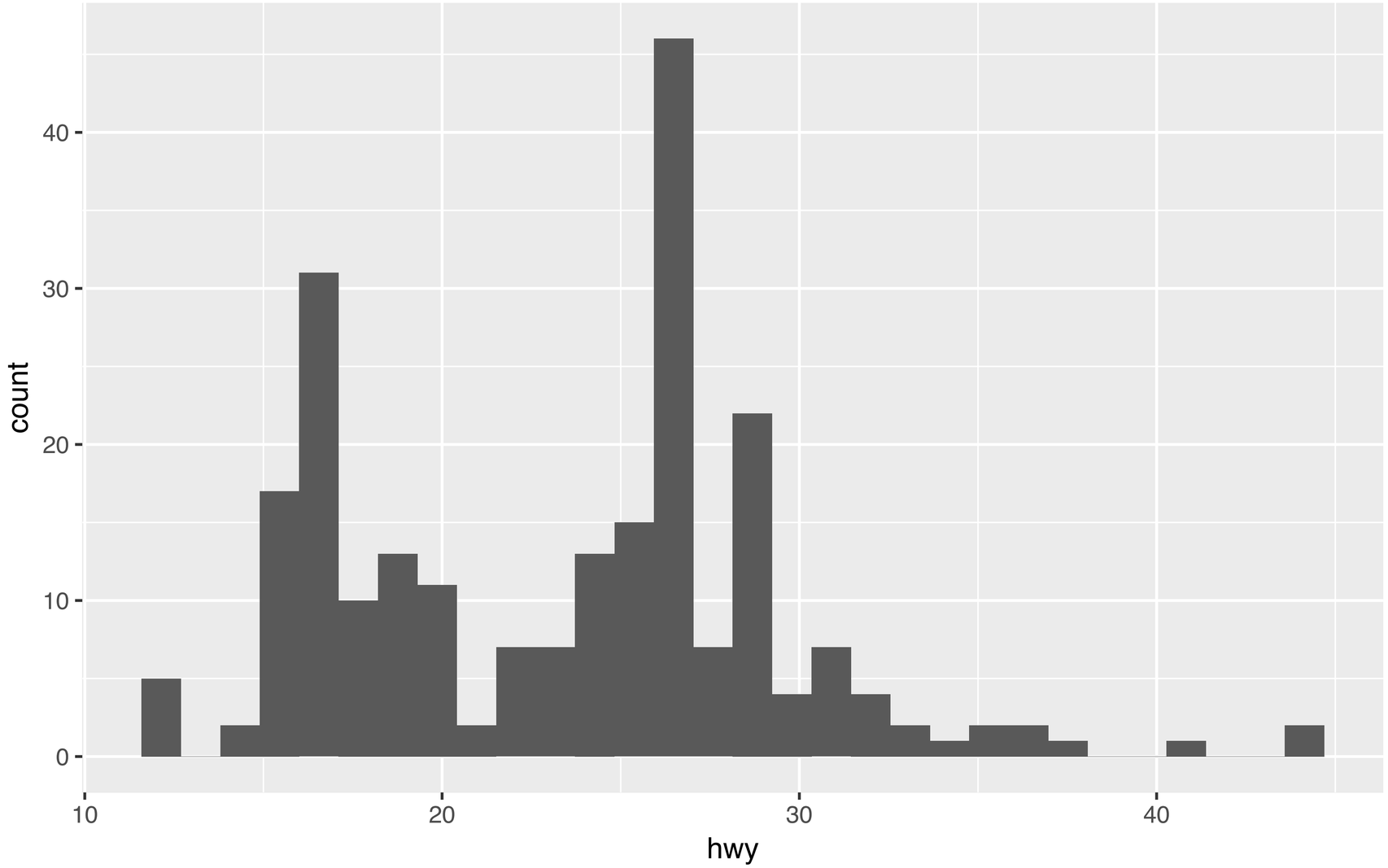



```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

Your Turn 4

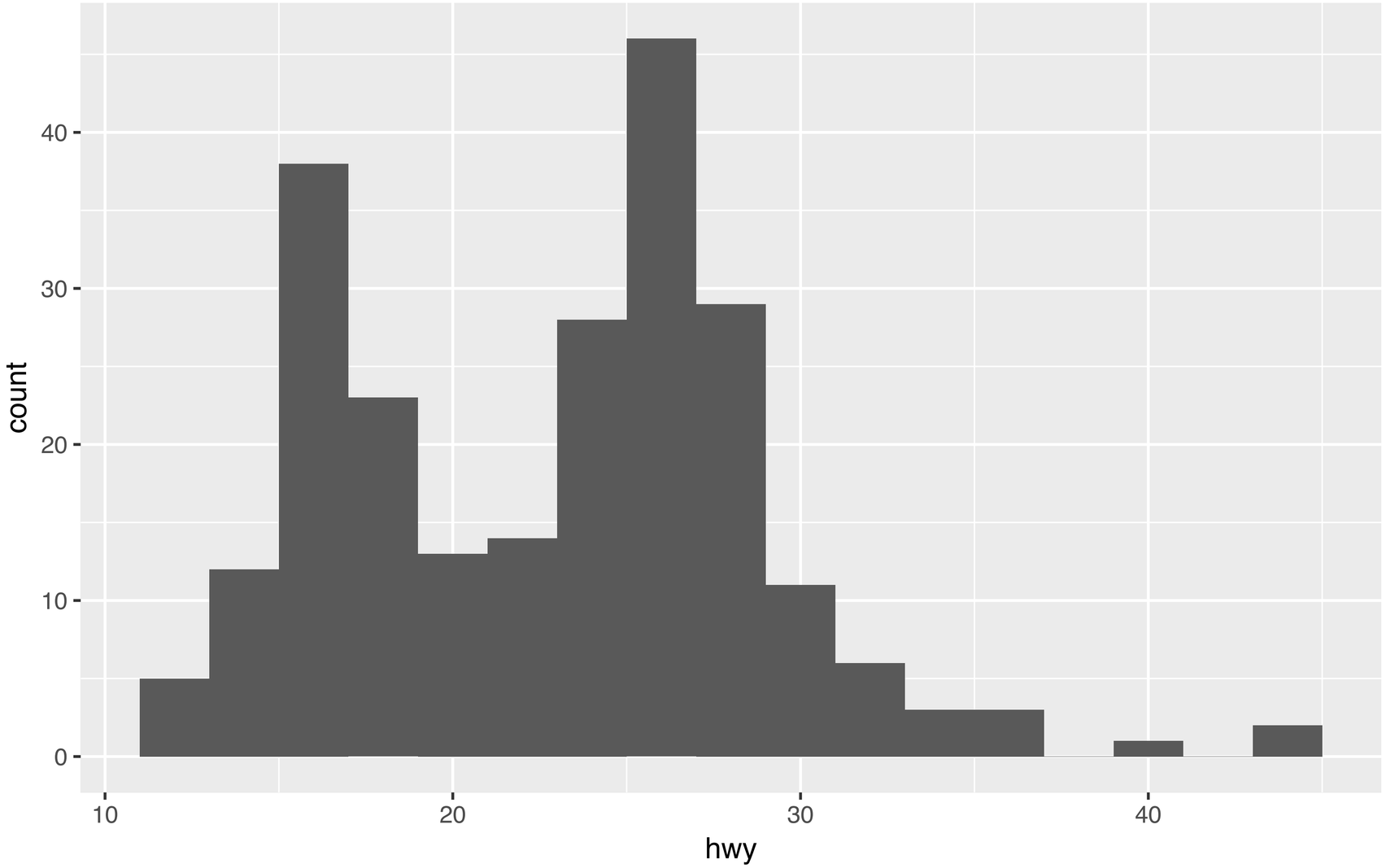
With your partner, make the histogram of **hwy** below. Use the cheatsheet. Hint: do not supply a **y** variable.






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

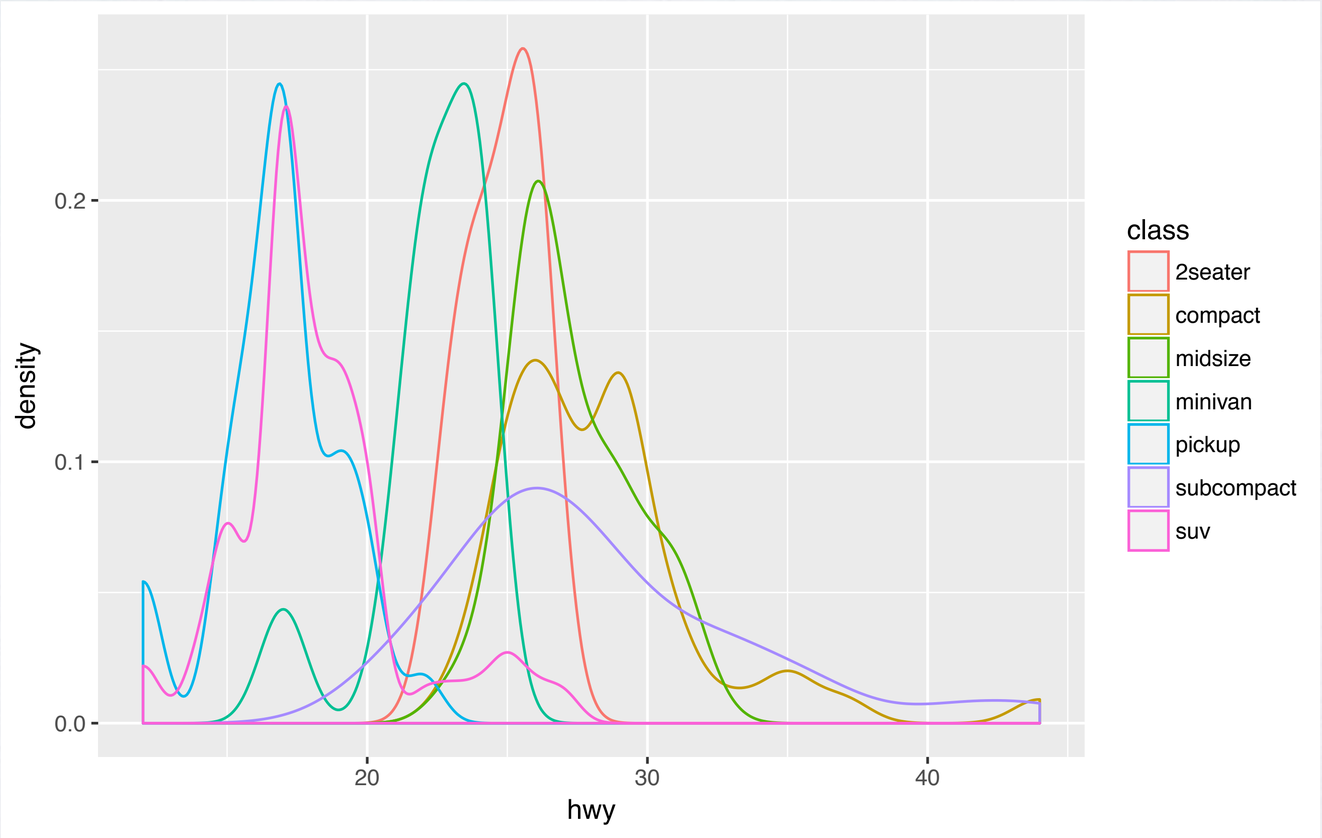
?

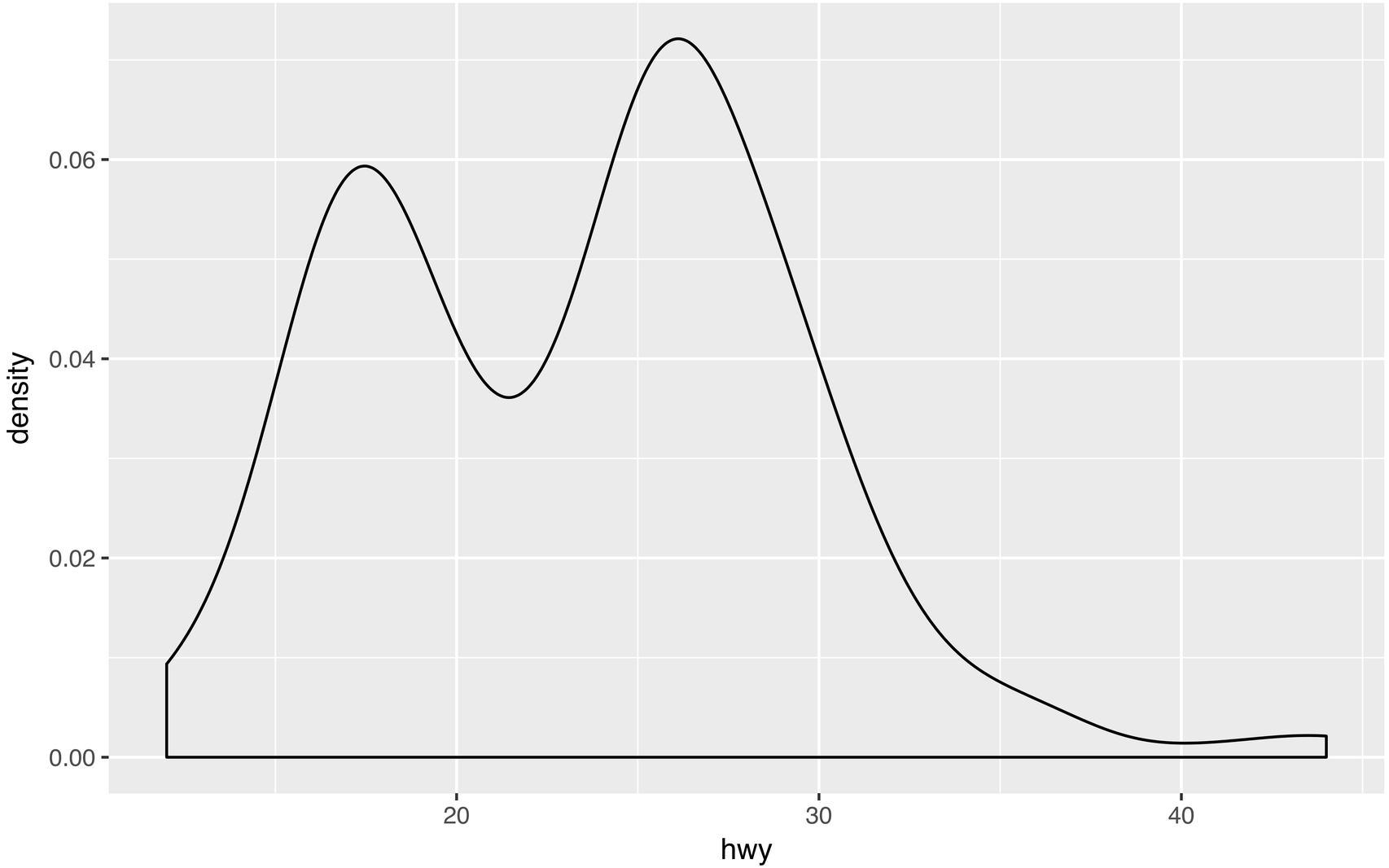



```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy), binwidth = 2)
```

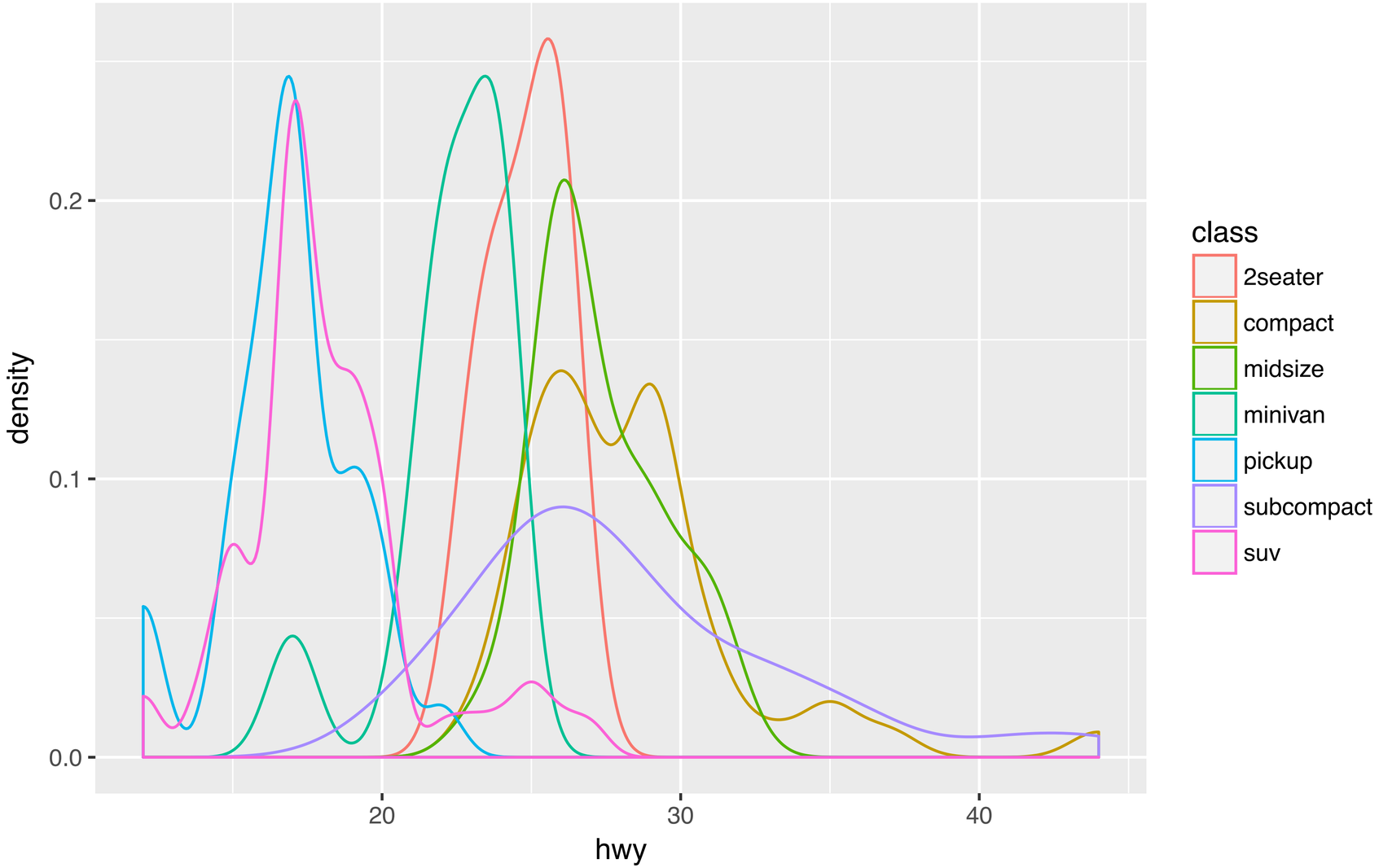
Your Turn 5

With your partner, make the density plot of **hwy** colored by **class** below. Use the cheatsheet. Try your best guess.

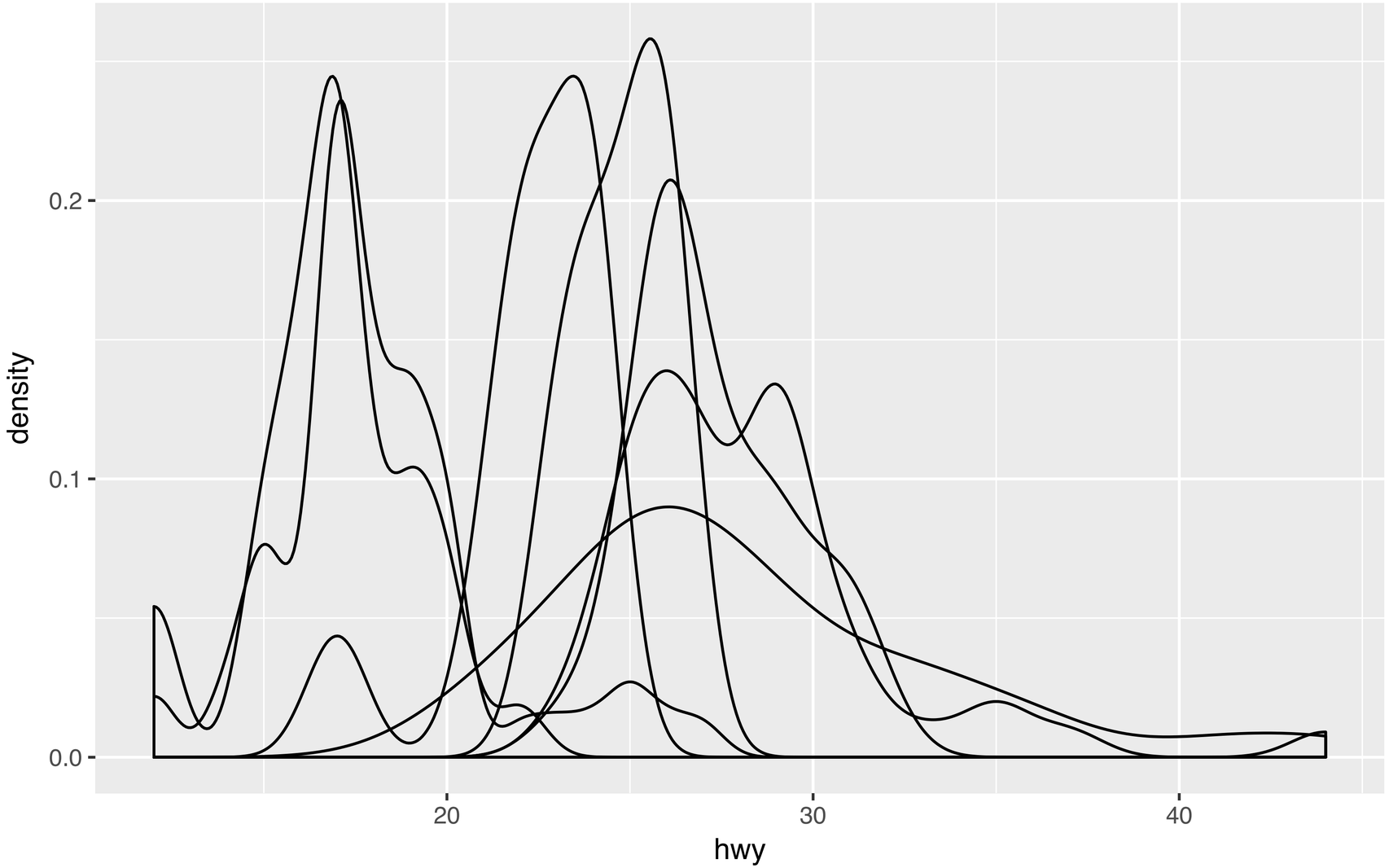





```
ggplot(data = mpg) +
  geom_density(mapping = aes(x = hwy))
```



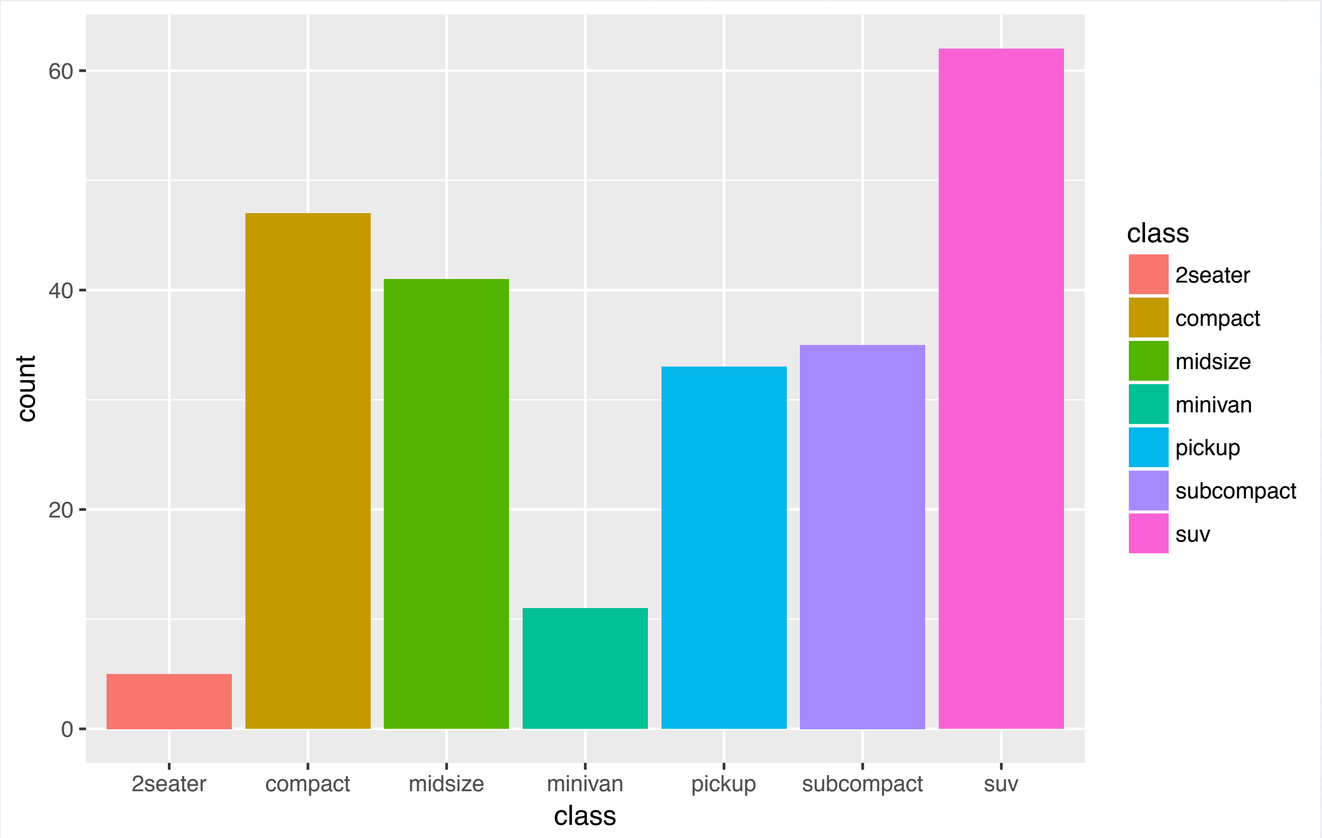

```
ggplot(data = mpg) +
  geom_density(mapping = aes(x = hwy, color = class))
```

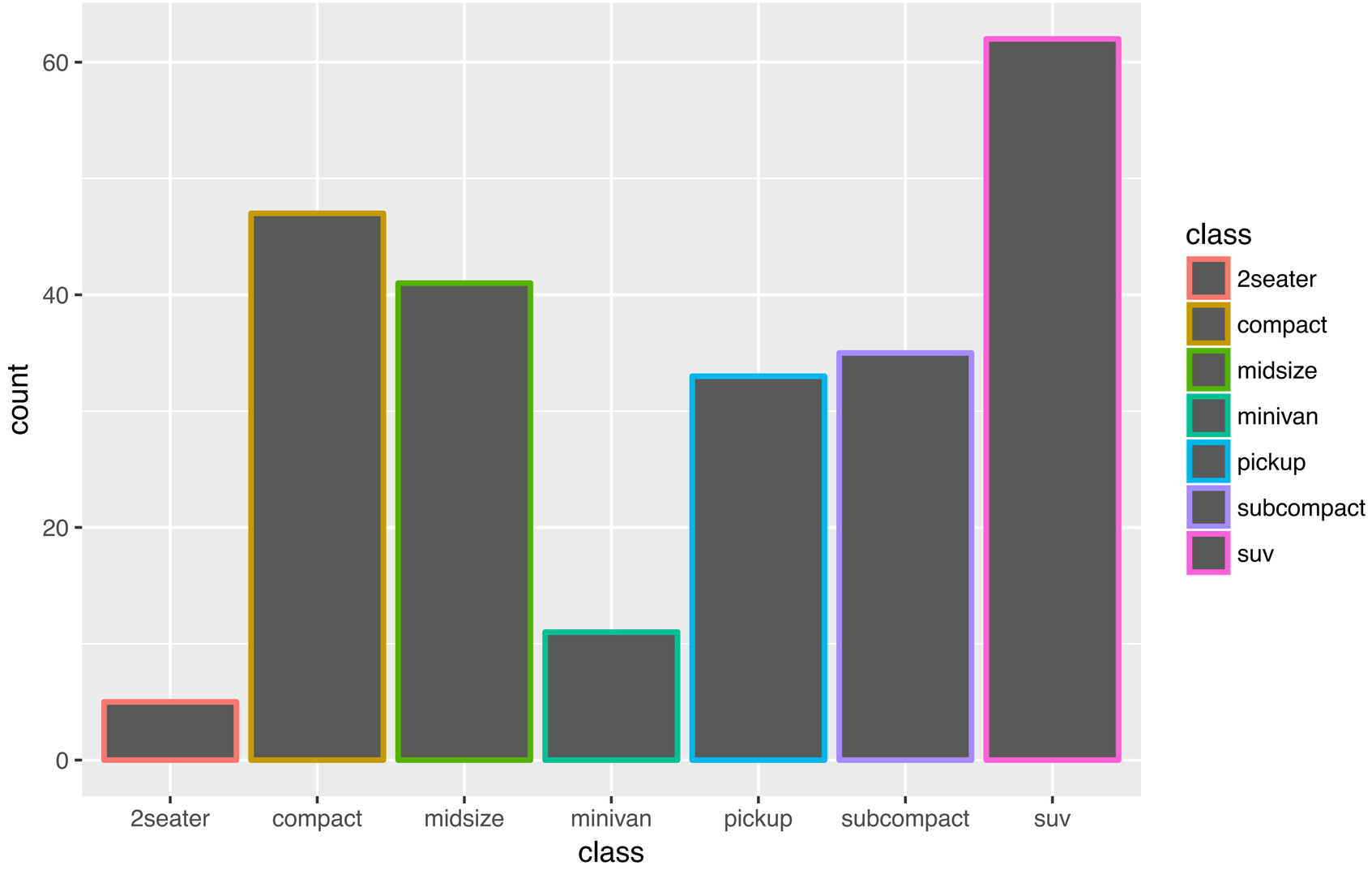



```
ggplot(data = mpg) +
  geom_density(mapping = aes(x = hwy, group = class))
```

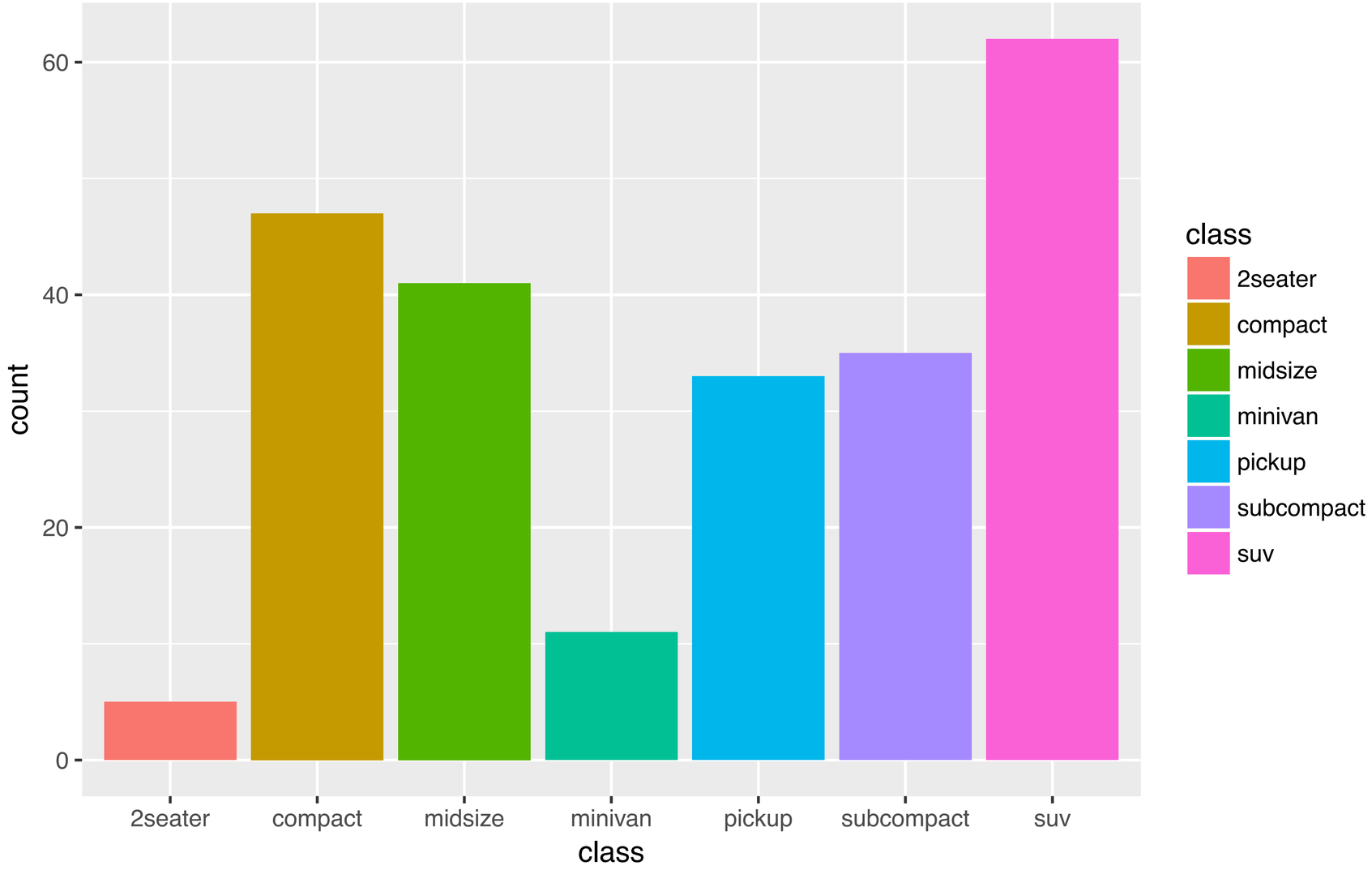
Your Turn 6

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






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



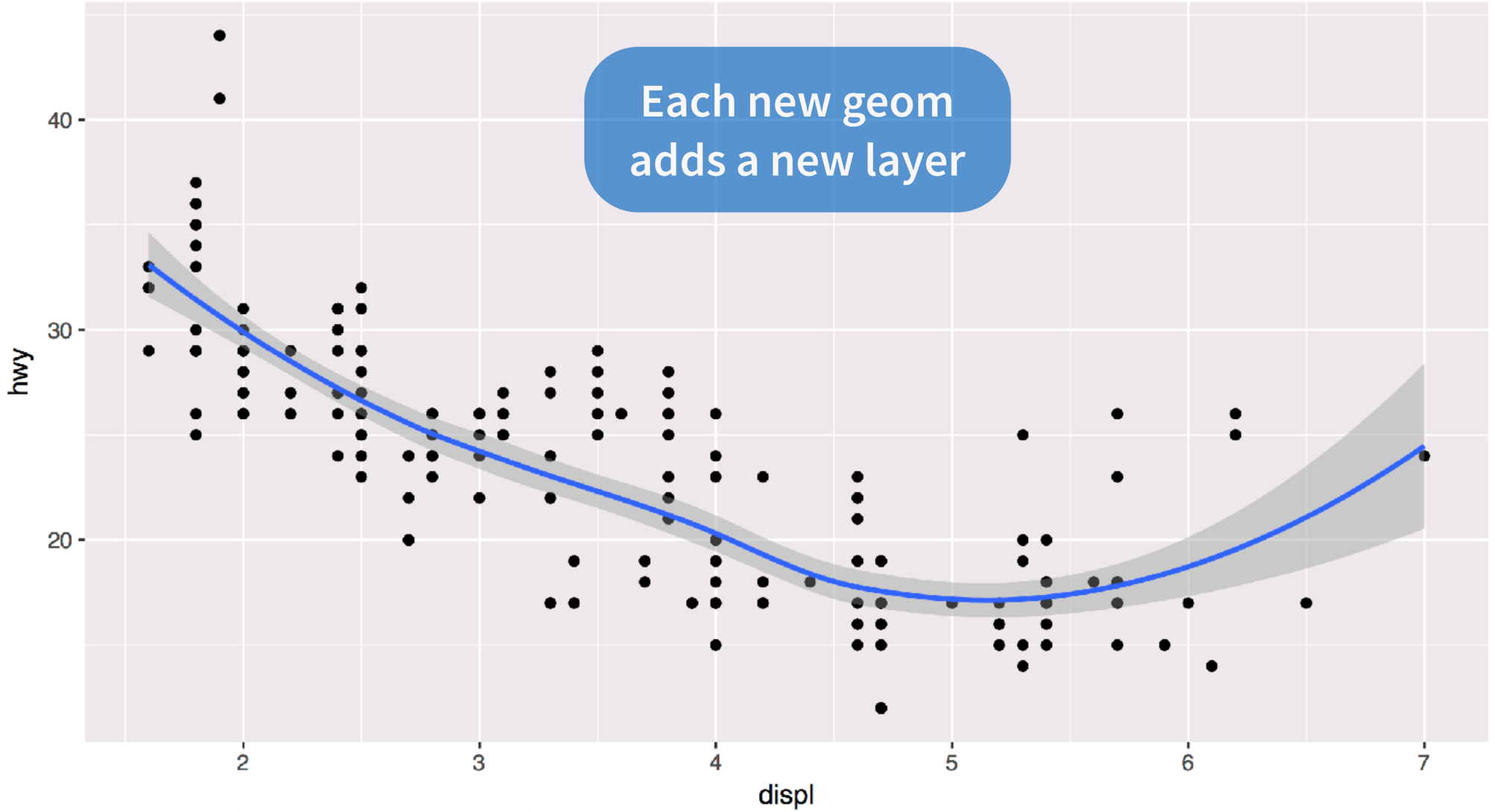

```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x = class, fill = class))
```

Your Turn 7

With your partner, predict what this code will do.

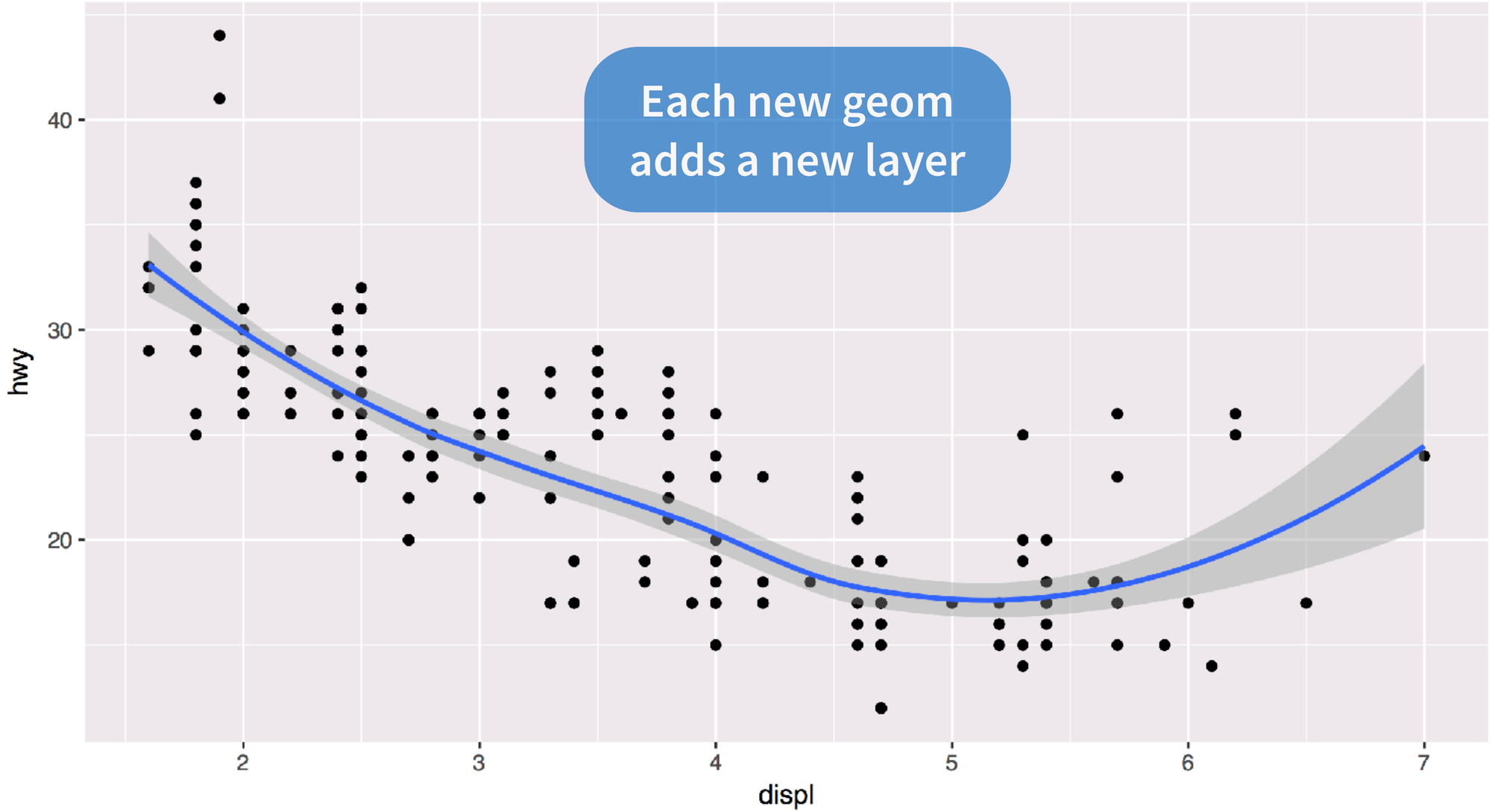
Then run it.

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




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

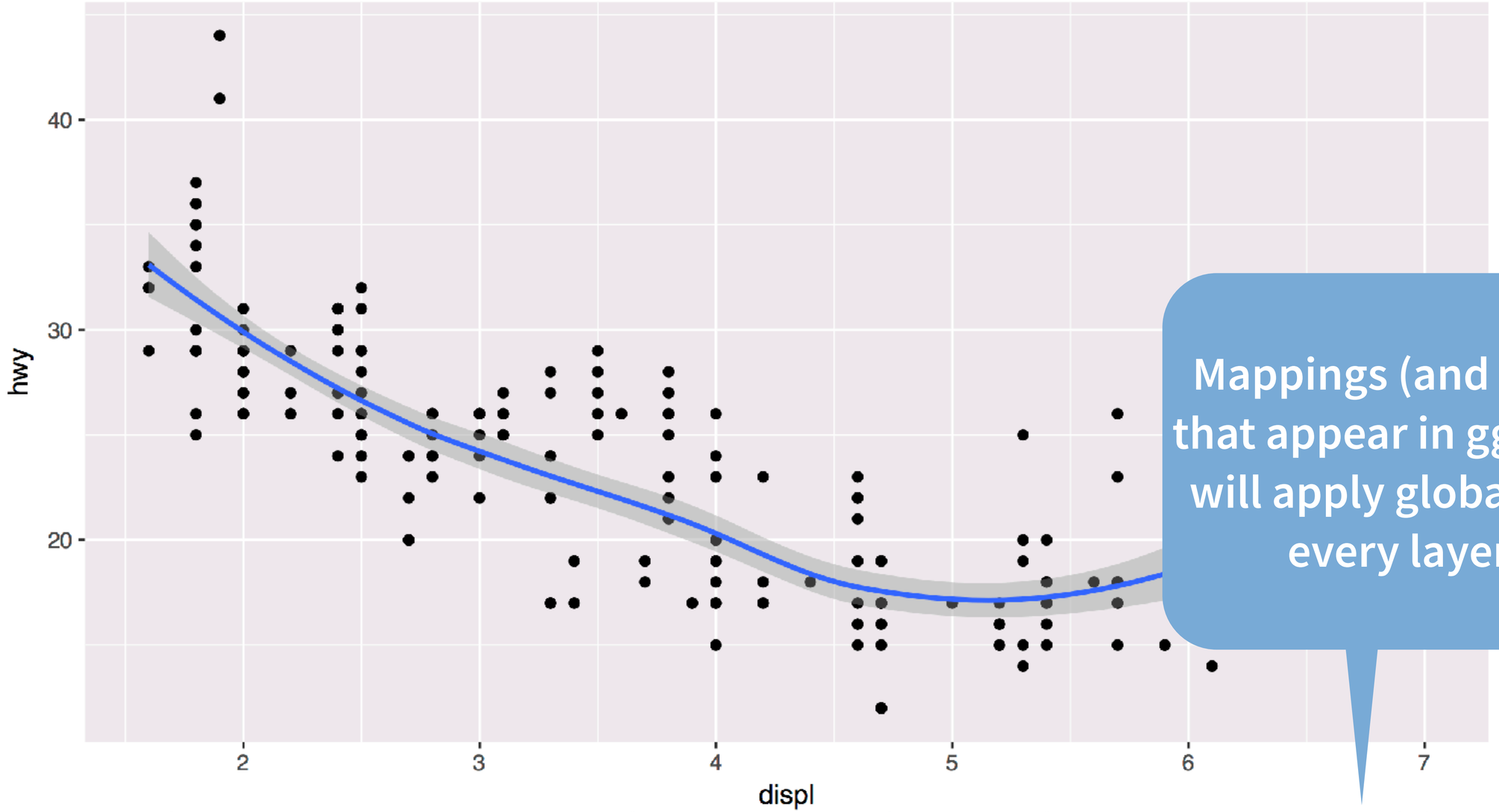

ggplot2




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


ggplot2

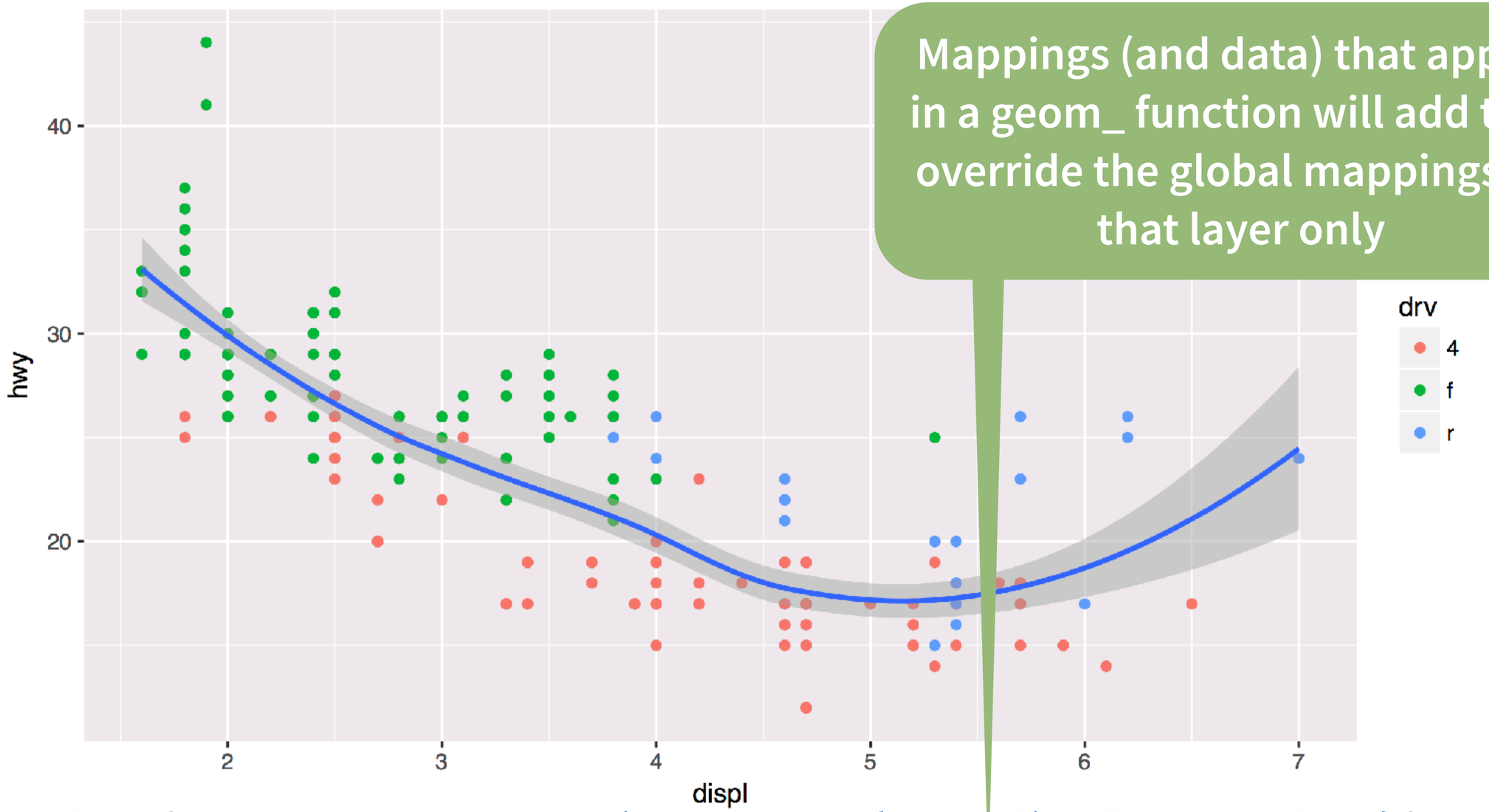
global vs. local



Mappings (and data) that appear in ggplot() will apply globally to every layer

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

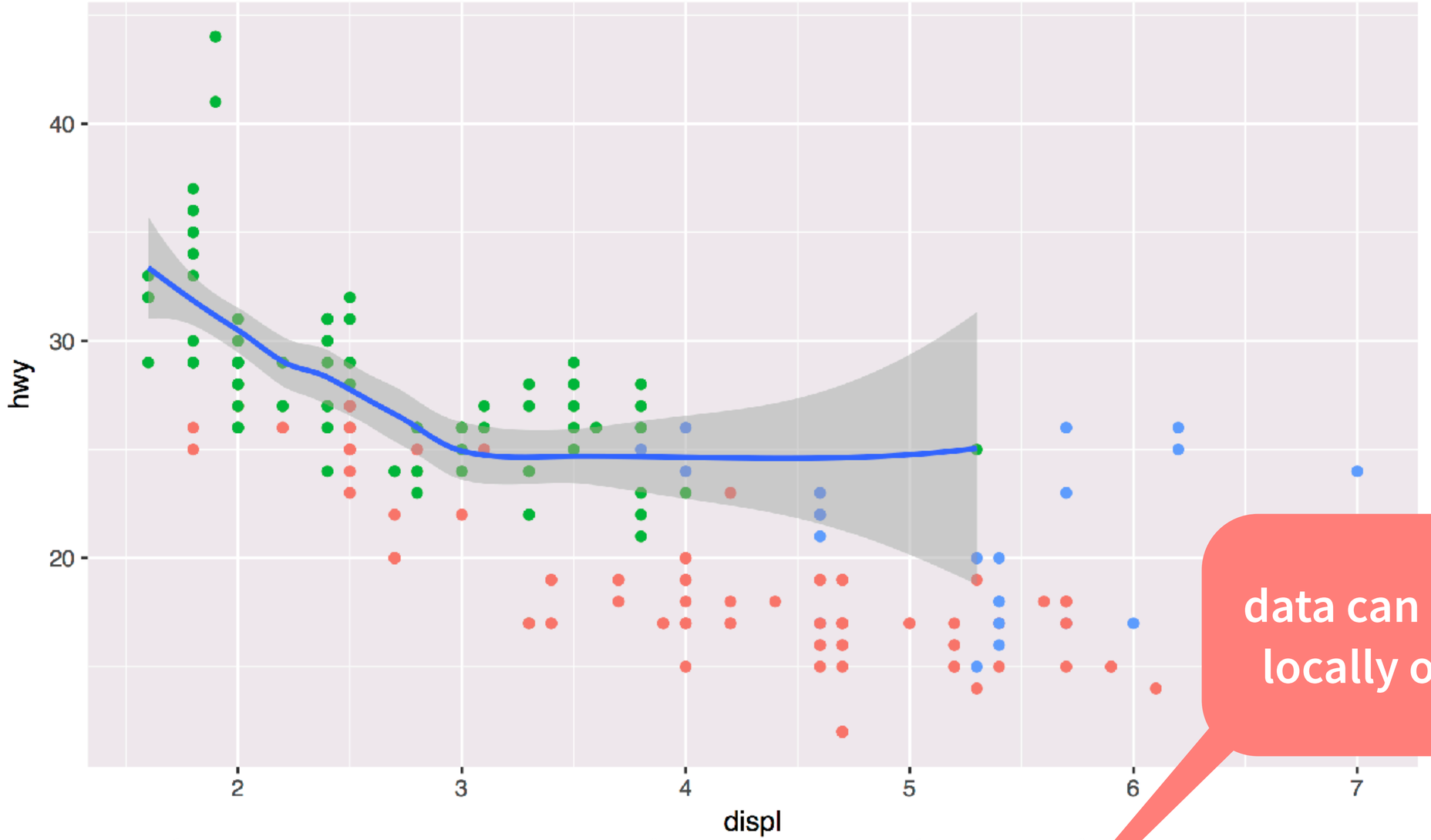

ggplot2



Mappings (and data) that appear in a geom_ function will add to or override the global mappings for that layer only

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


ggplot2



data can also be set locally or globally

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth(data = filter(mpg, drv == "f"))
```


ggplot2

Saving graphs

Your Turn 8

What does this command return?

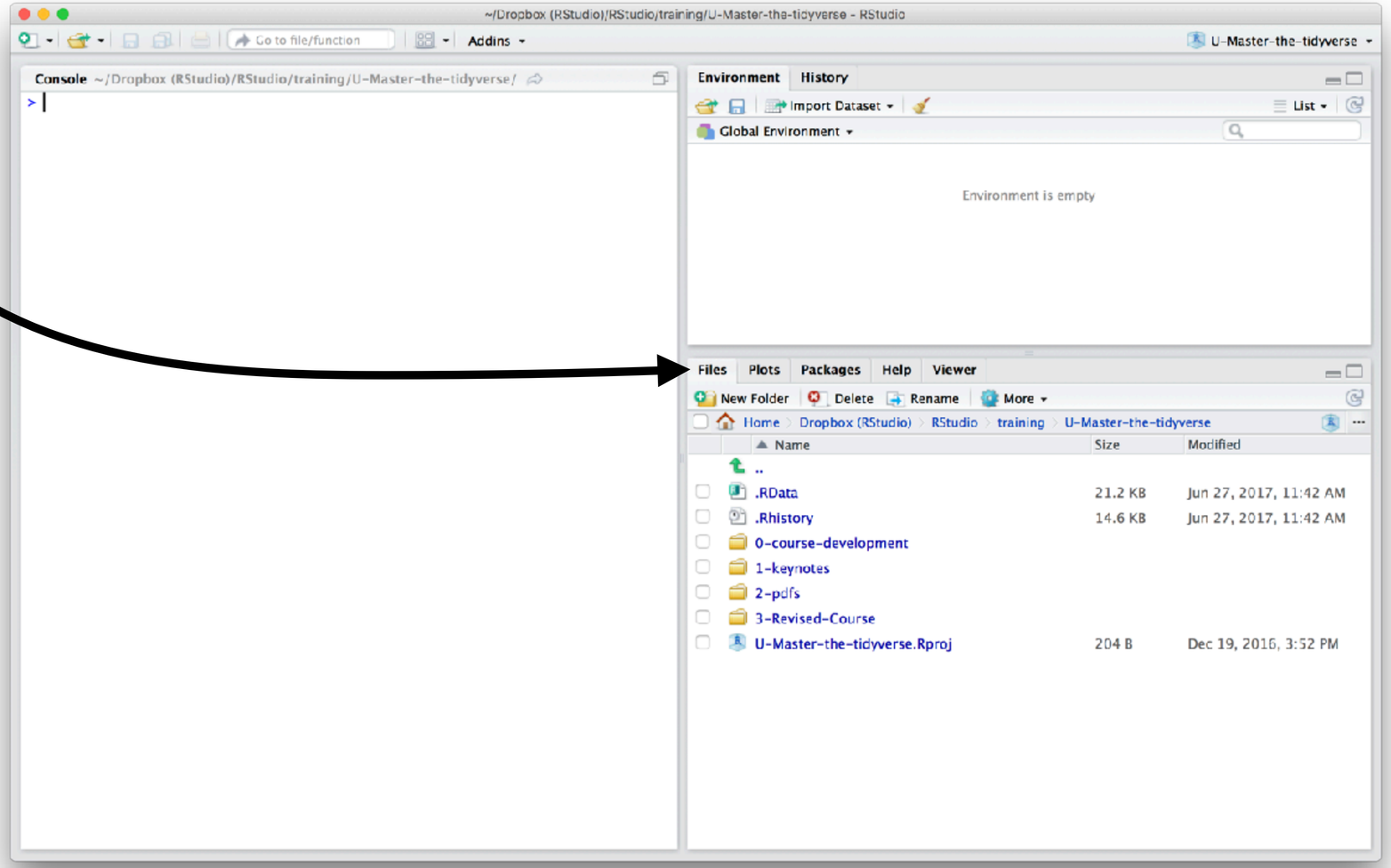
```
getwd()
```

Working directory

R associates itself with a folder (i.e. directory) on your computer.

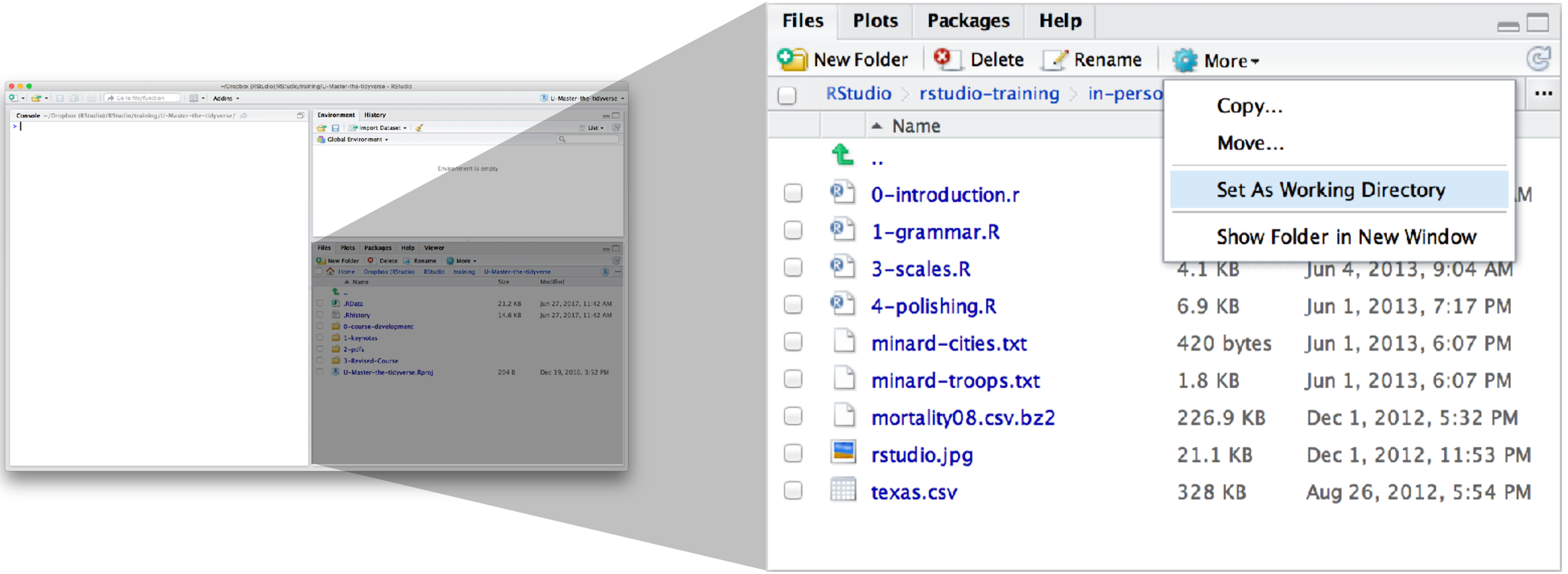
- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here

The files pane of the IDE displays the contents of your working directory



Changing the Working directory

Navigate in the files pane to a new directory. Click More > Set As Working Directory



Saving plots

ggsave() saves the last plot.

Uses size on screen:

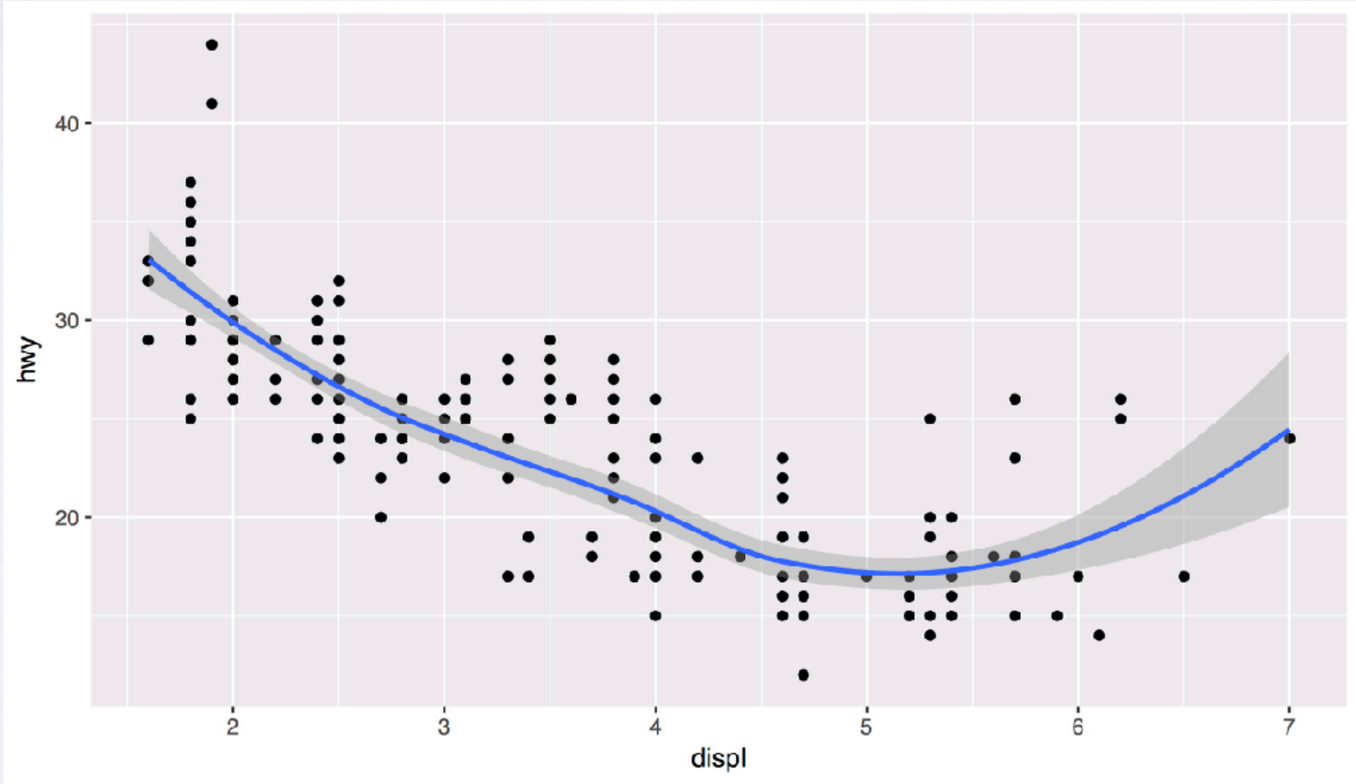
```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height = 6)
```

Your Turn 9

Save your last plot and then locate it in your files pane.
(You may have to refresh the files list).



Grammar of Graphics

To make a graph

[template]

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

To make a graph

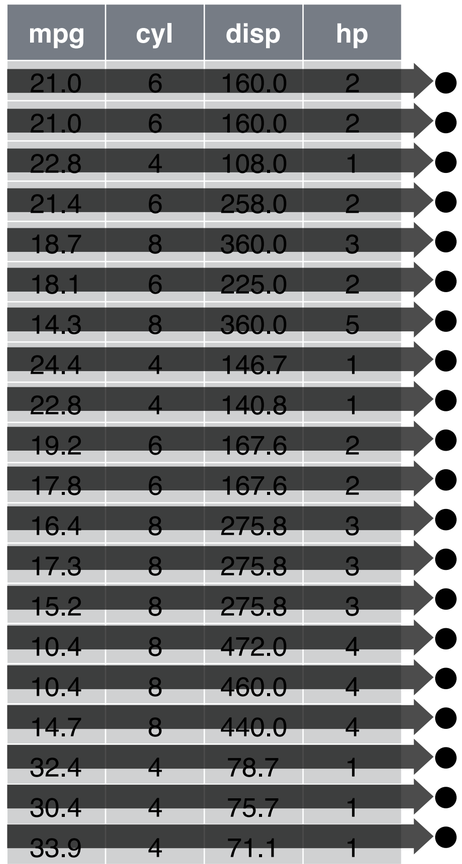
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

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

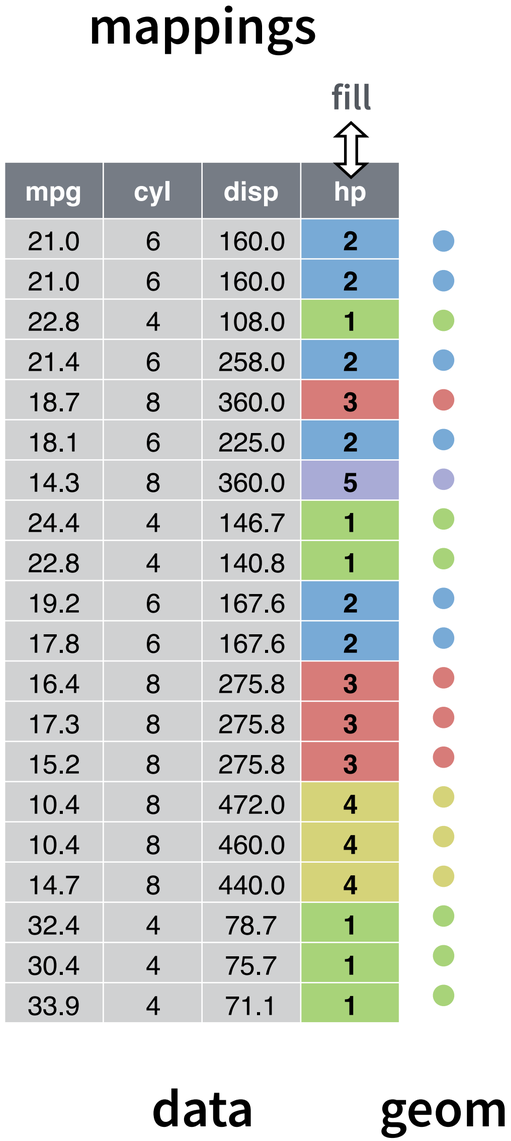
data **geom**

1. Pick a **data** set

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

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

To make a graph



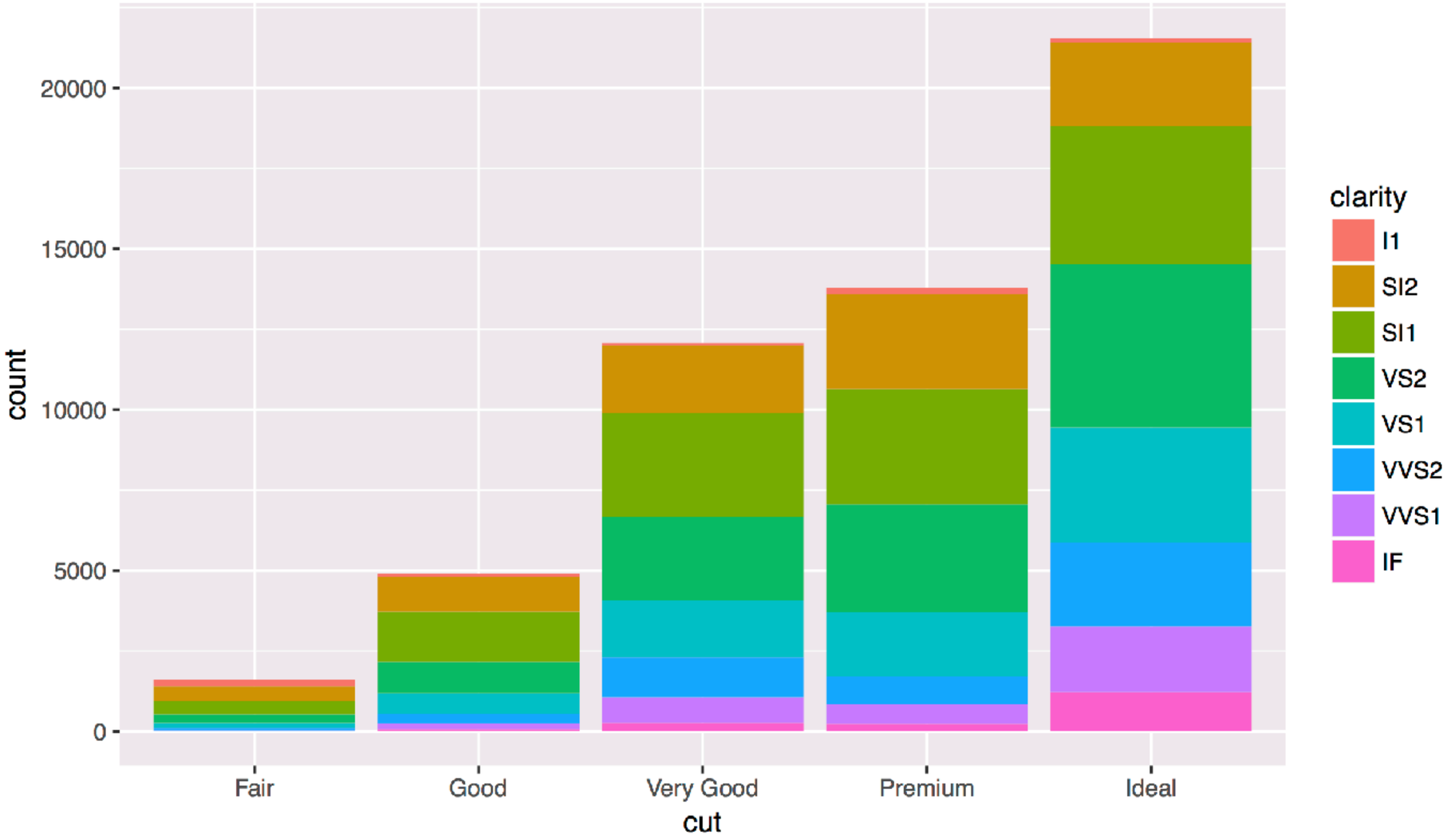
1. Pick a **data** set

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

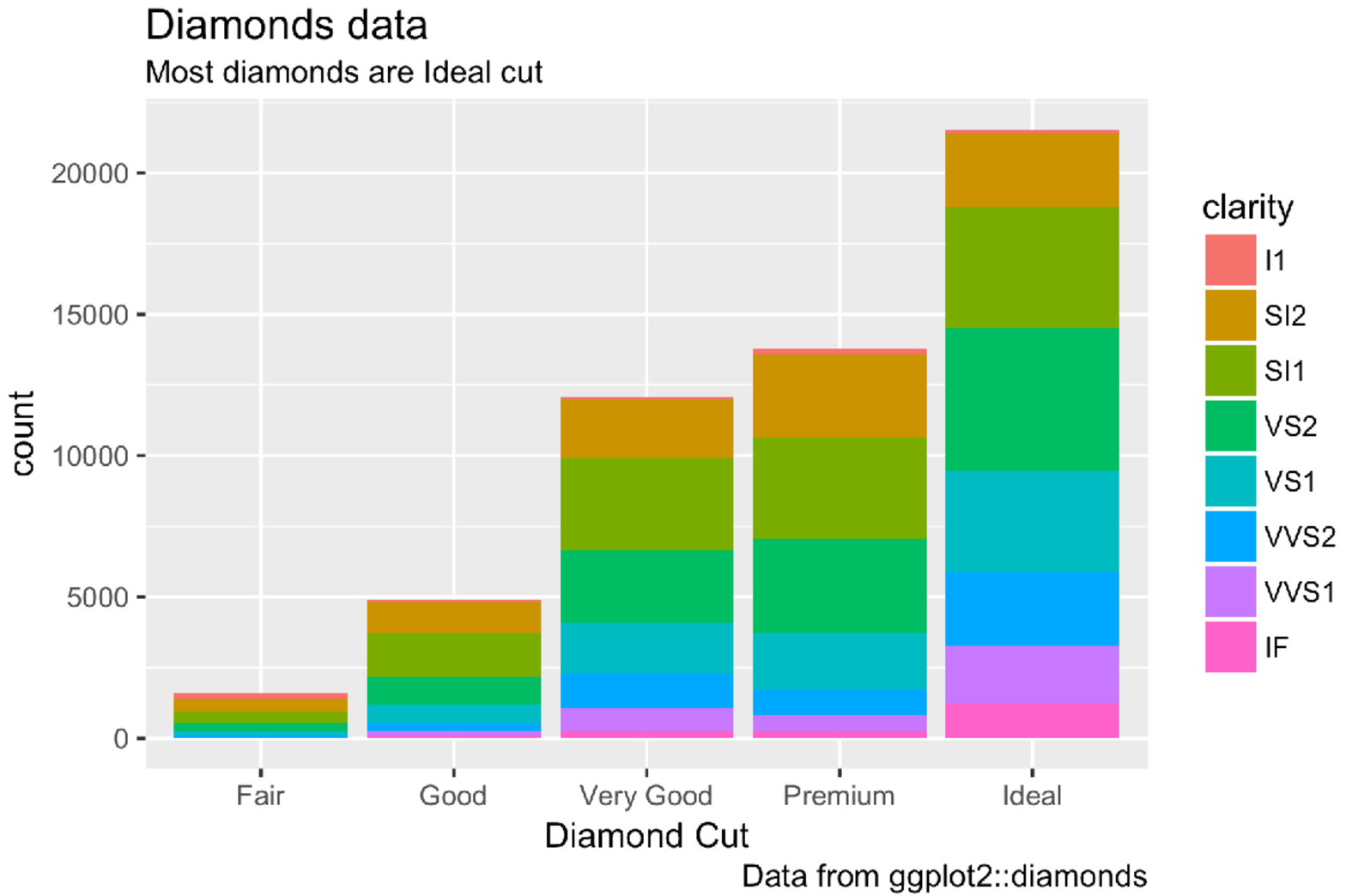
2. Choose a **geom** to display cases

3. **Map** aesthetic properties to variables

What else?

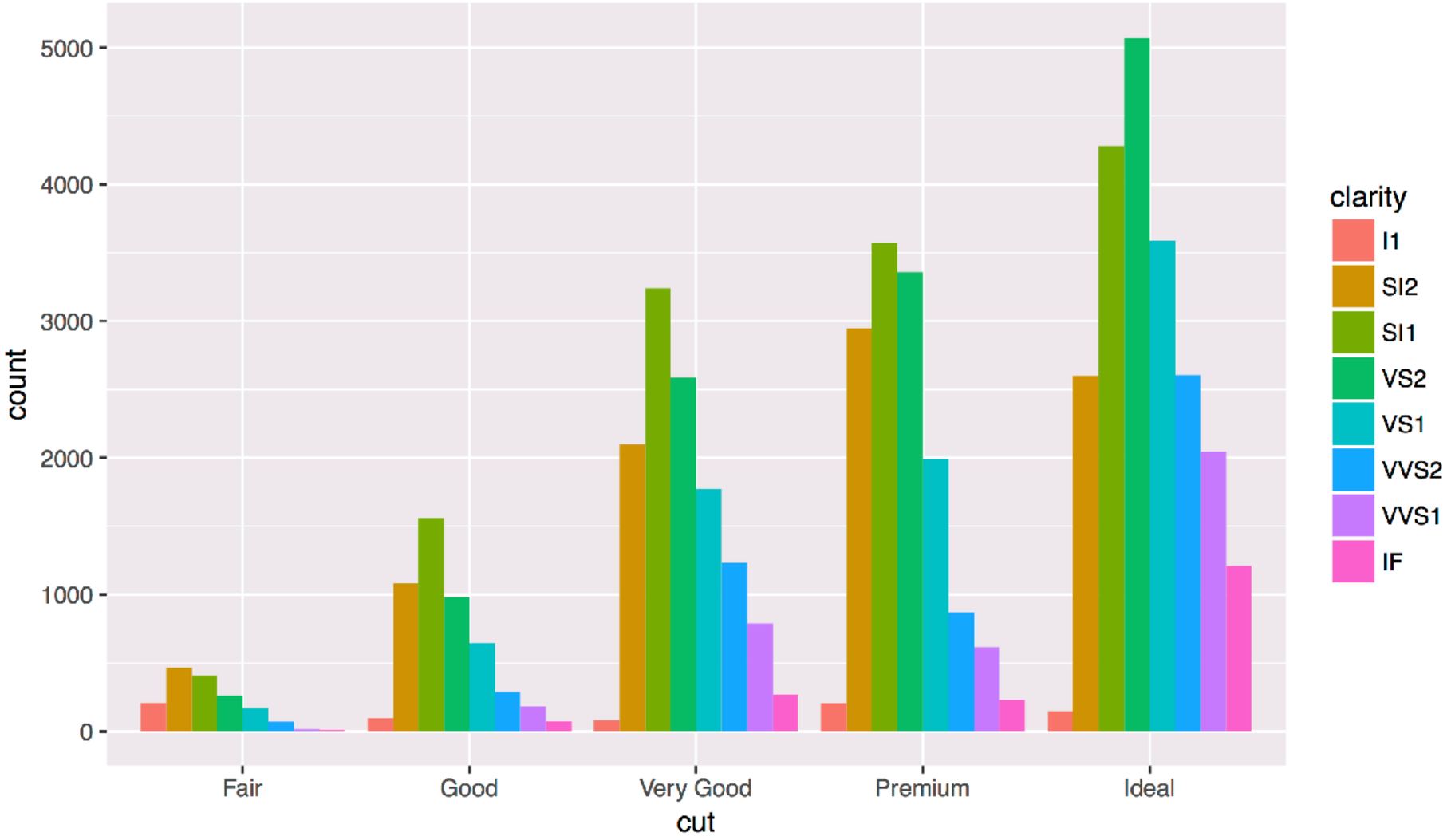


Titles and captions



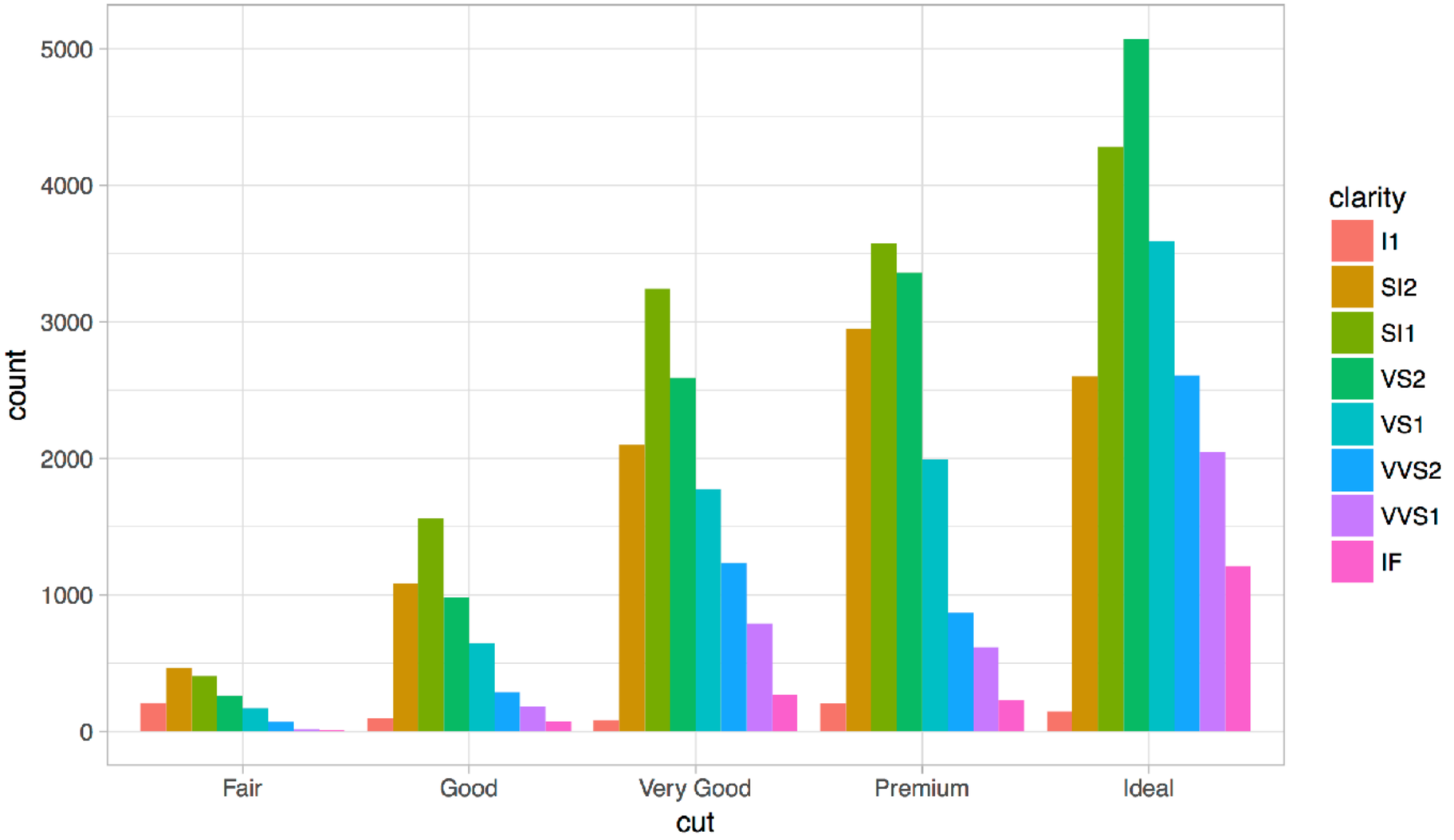
Position Adjustments

How overlapping objects are arranged



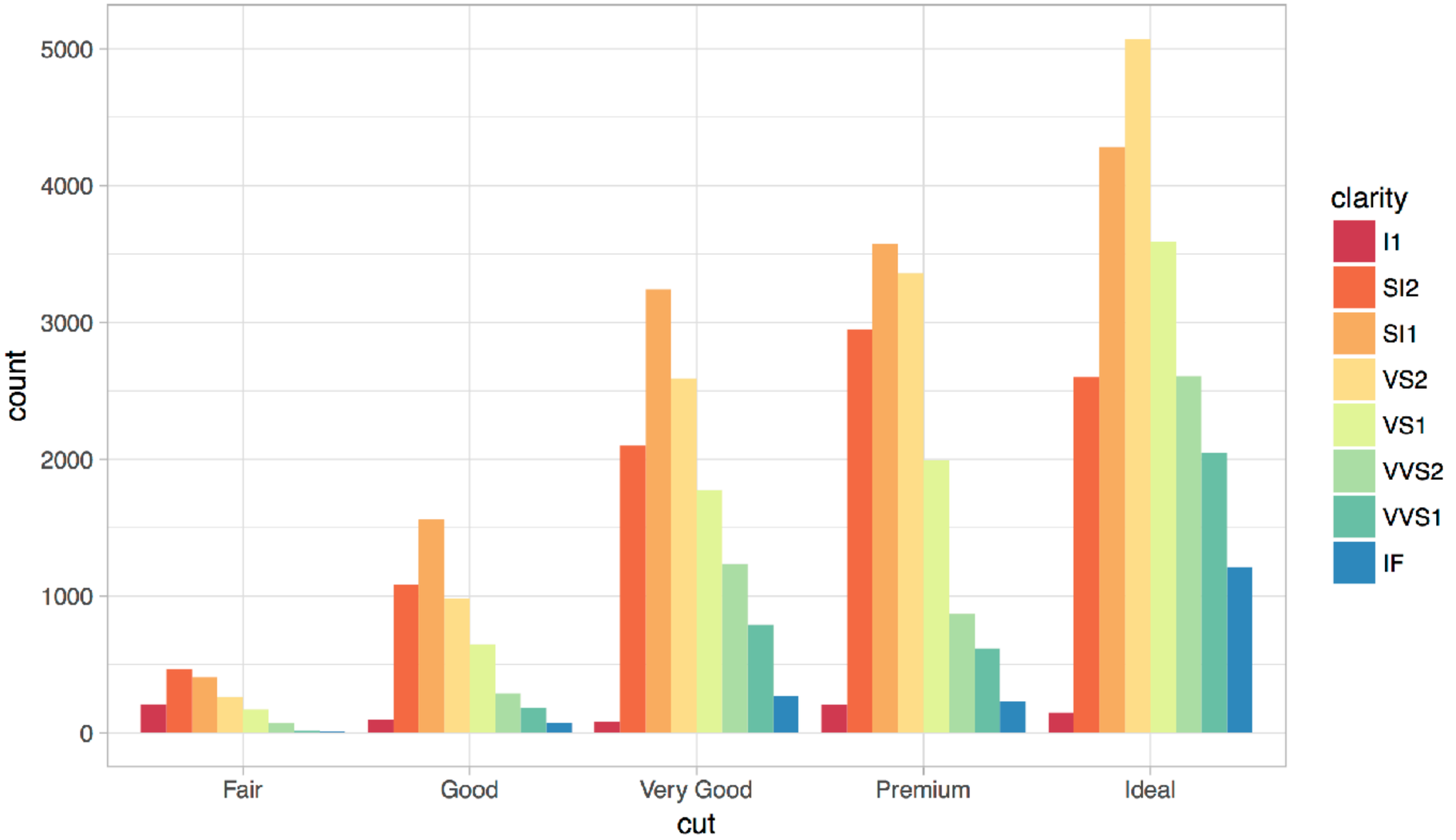
Themes

Visual appearance of non-data elements



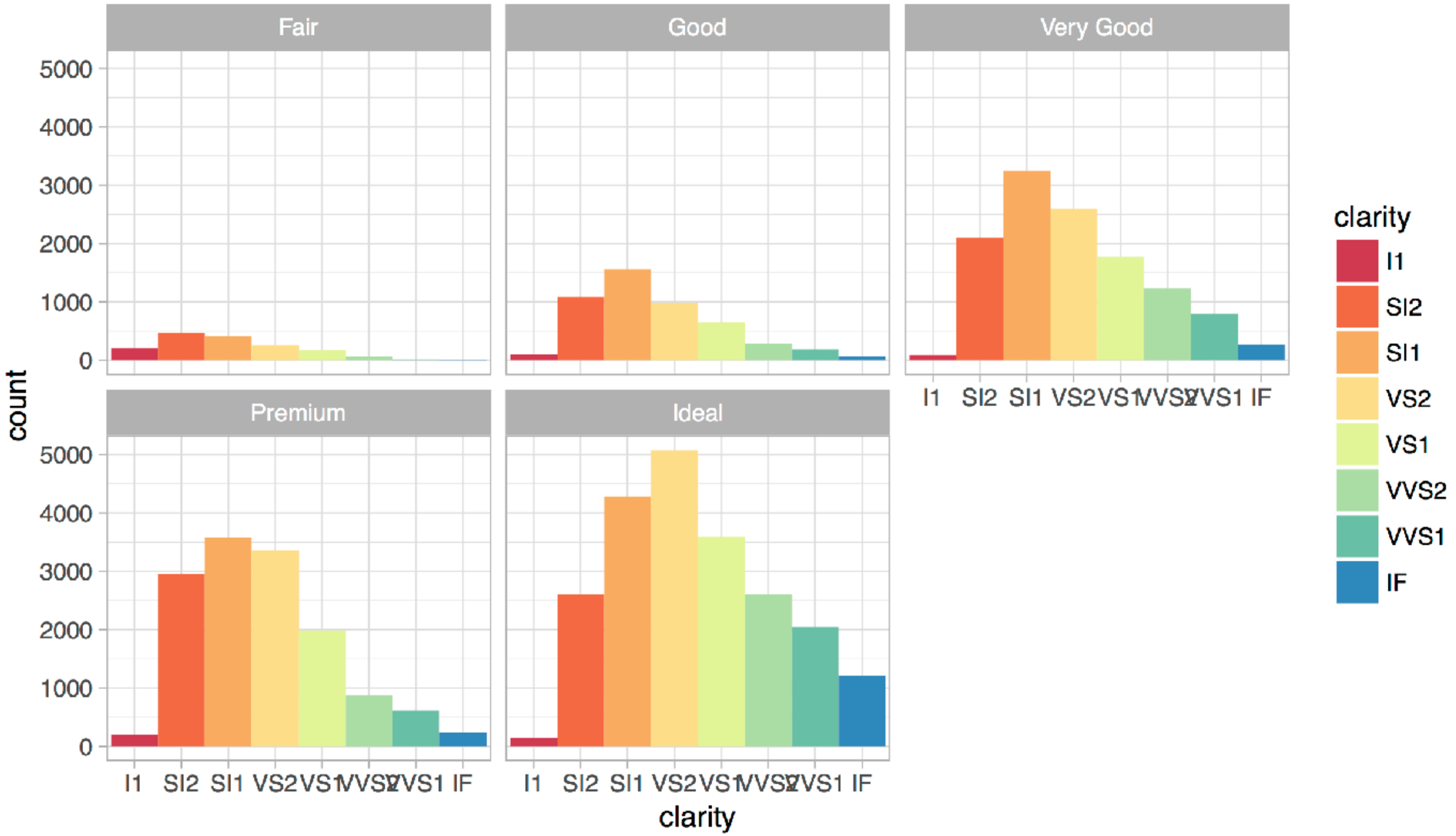
Scales

Customize color scales, other mappings

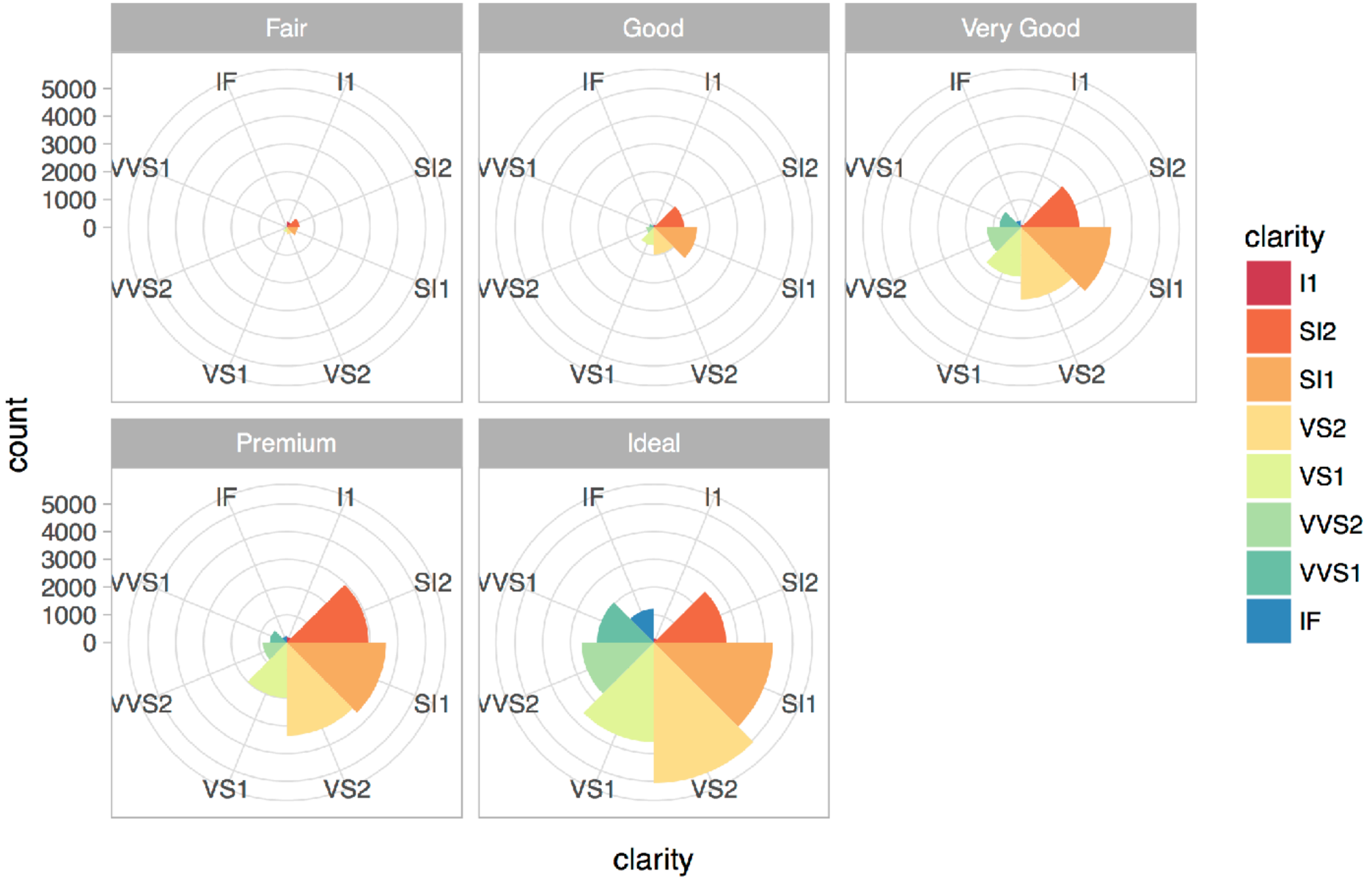


Facets

Subplots that display subsets of the data.



Coordinate systems



A ggplot2 template

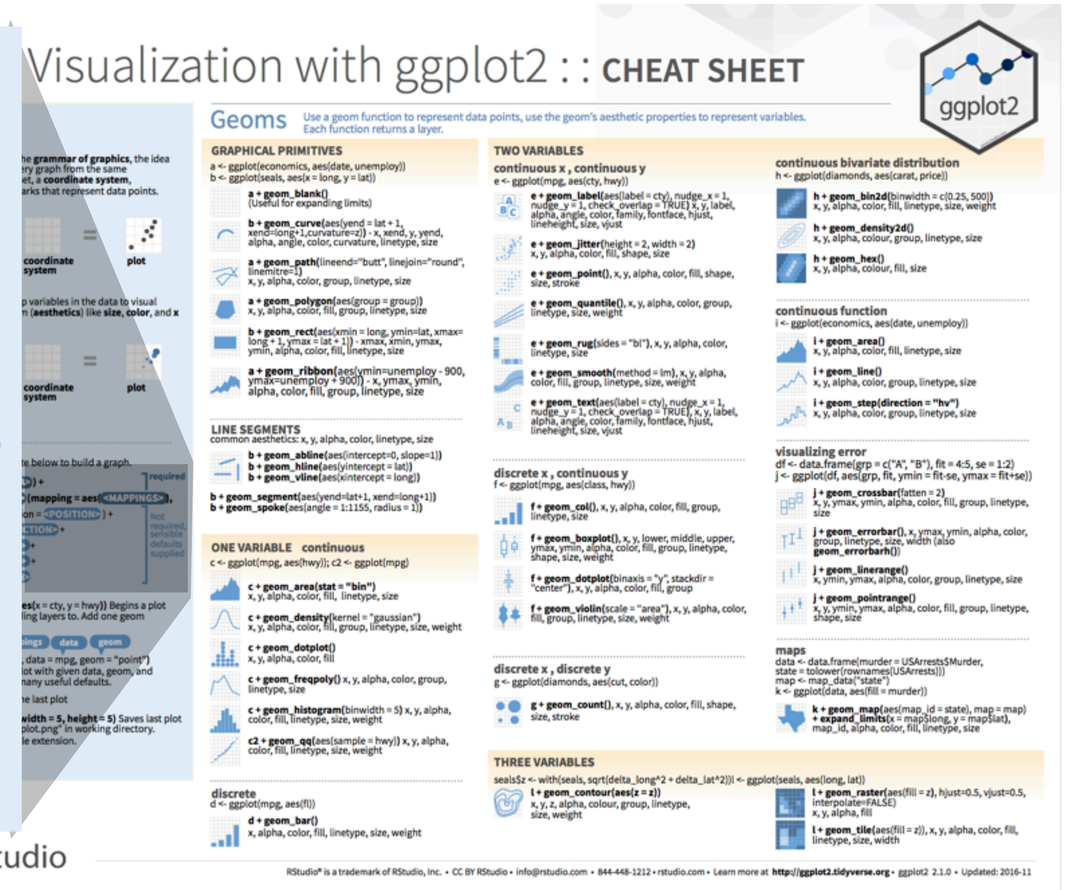
Make any plot by filling in the parameters of this template

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

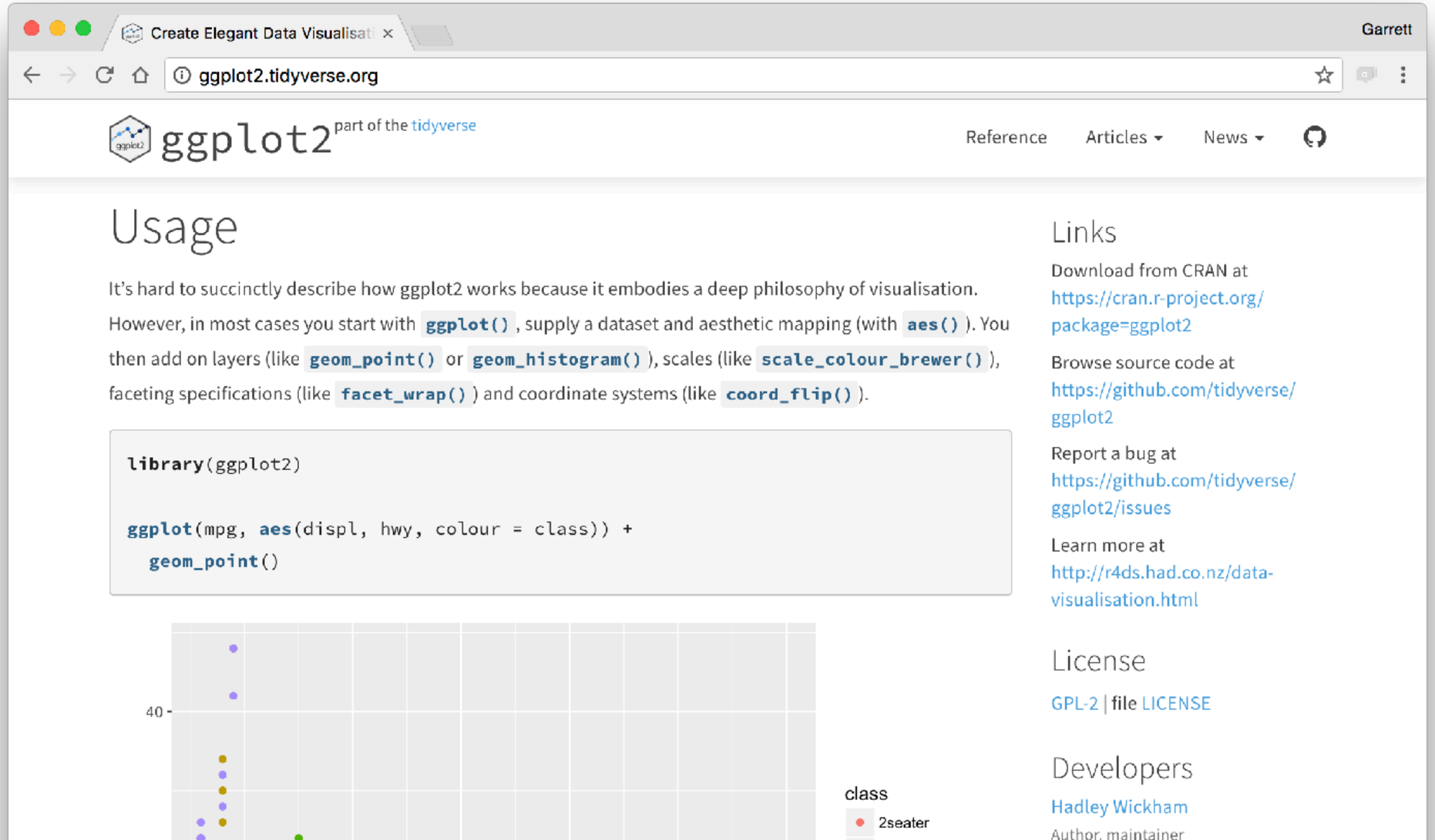
required

Not required, sensible defaults supplied



[Adapted from 'Master the tidyverse' CC by RStudio]

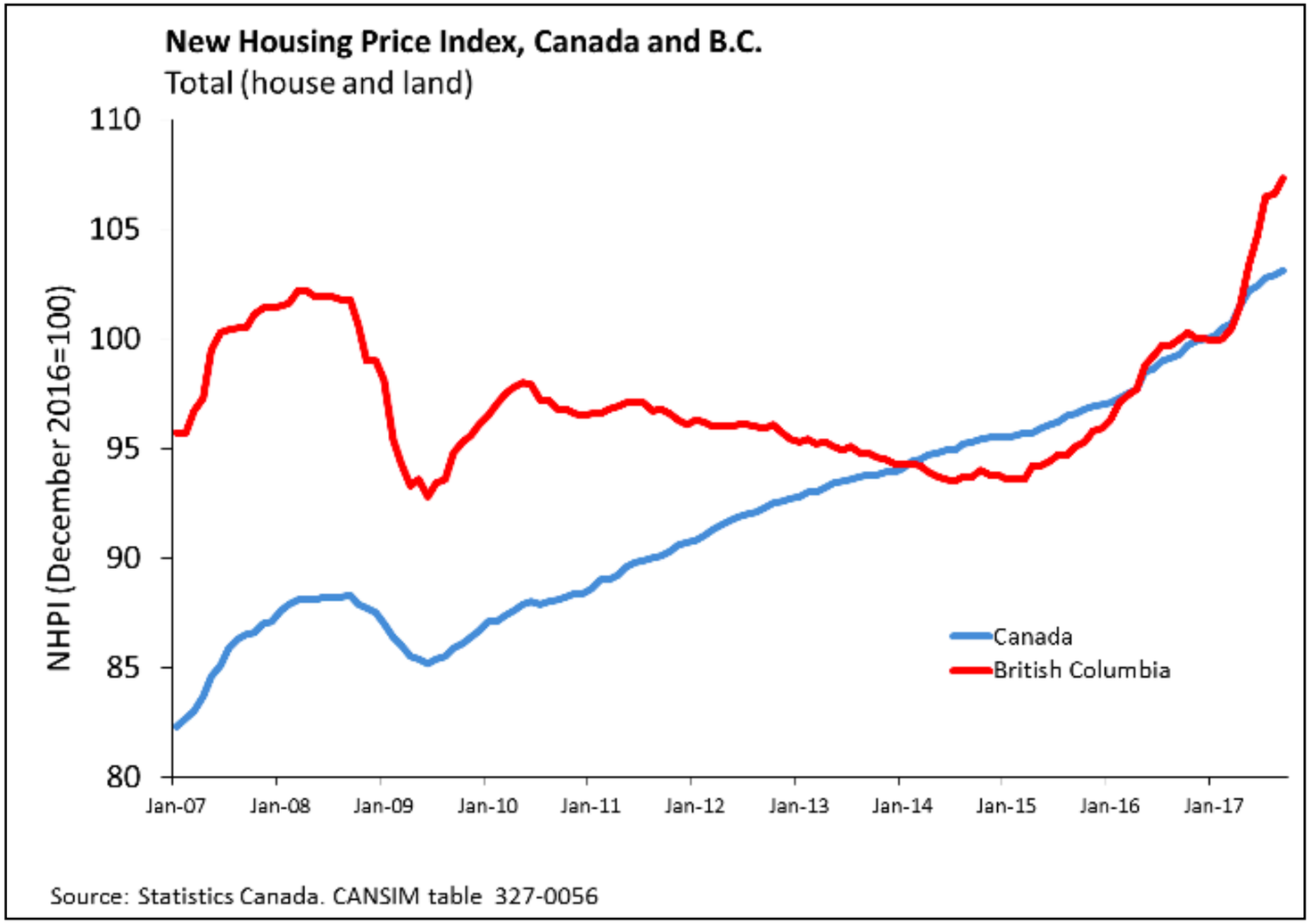
ggplot2.tidyverse.org



Project

New Housing Price Index

https://www2.gov.bc.ca/gov/content/data/statistics/infoline/infoline-2017/17-146-price-new-housing



Your Turn

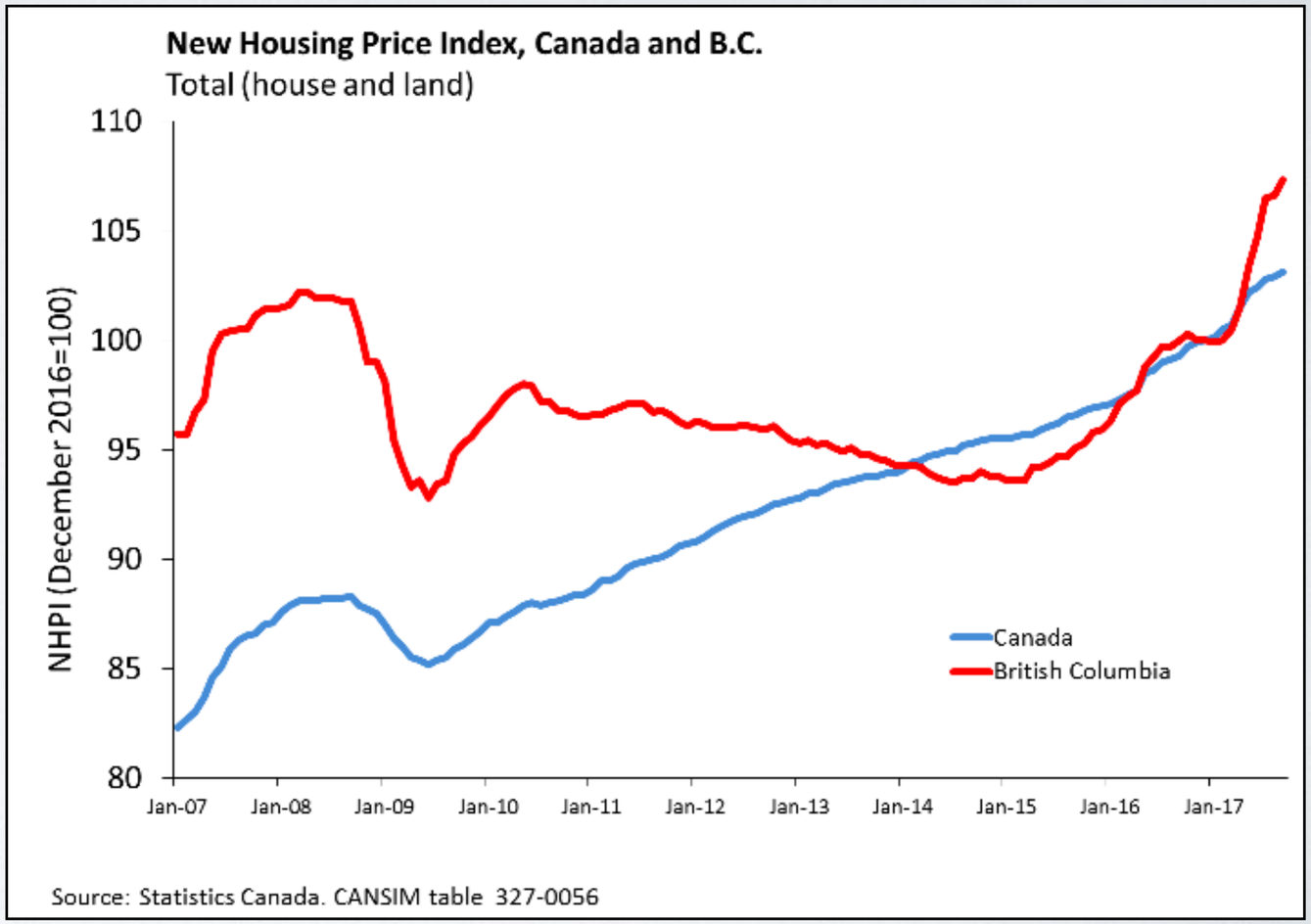
Save your open files.

Navigate into the project_housing directory in the Files pane.

Click on project-housing.Rproj

Your Turn

1. Open 01-visualize.Rmd
2. Take a look at `housing`
3. Create a visualization that imitates this one:



Your Turn

Switch back to the r_intro_bc_stats project.

Open **04-Transform-Data.Rmd**.

Visualize Data with