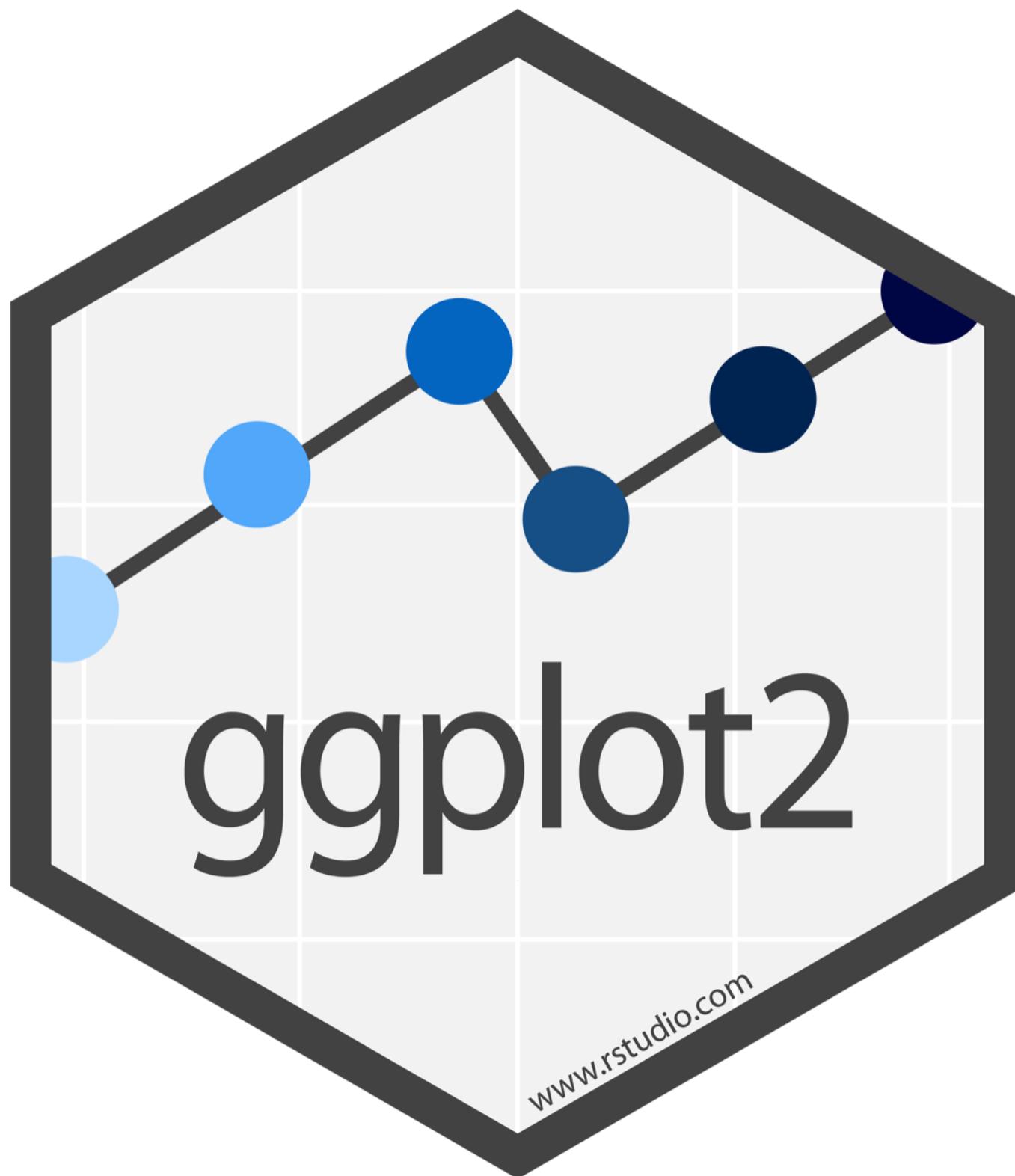


Visualize Data with



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

- John Tukey

"We eat first with our eyes. "

-

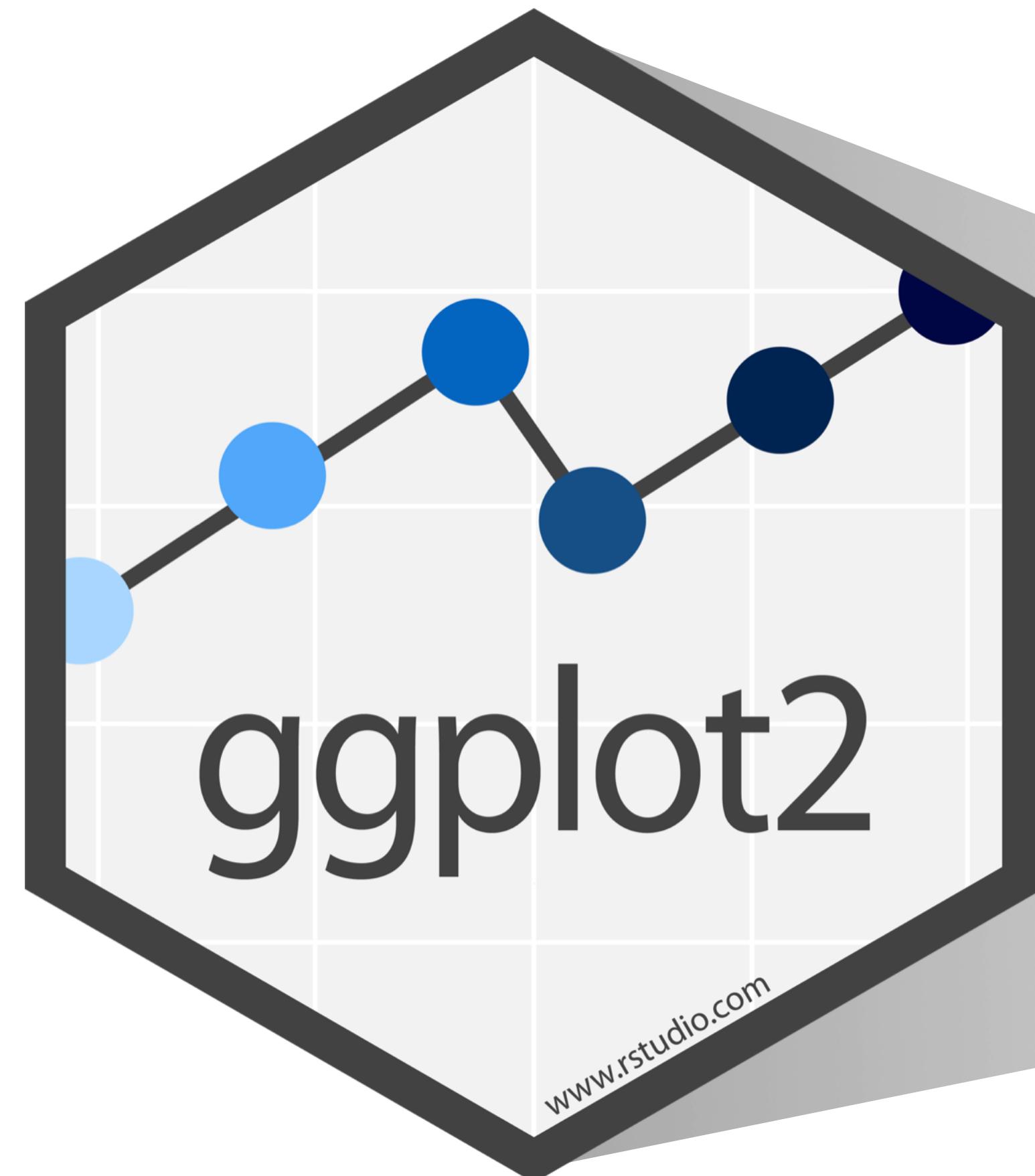
Apicius
1st Century Roman gourmand



AESTHETICS THAT ATTRACT US TO FOOD

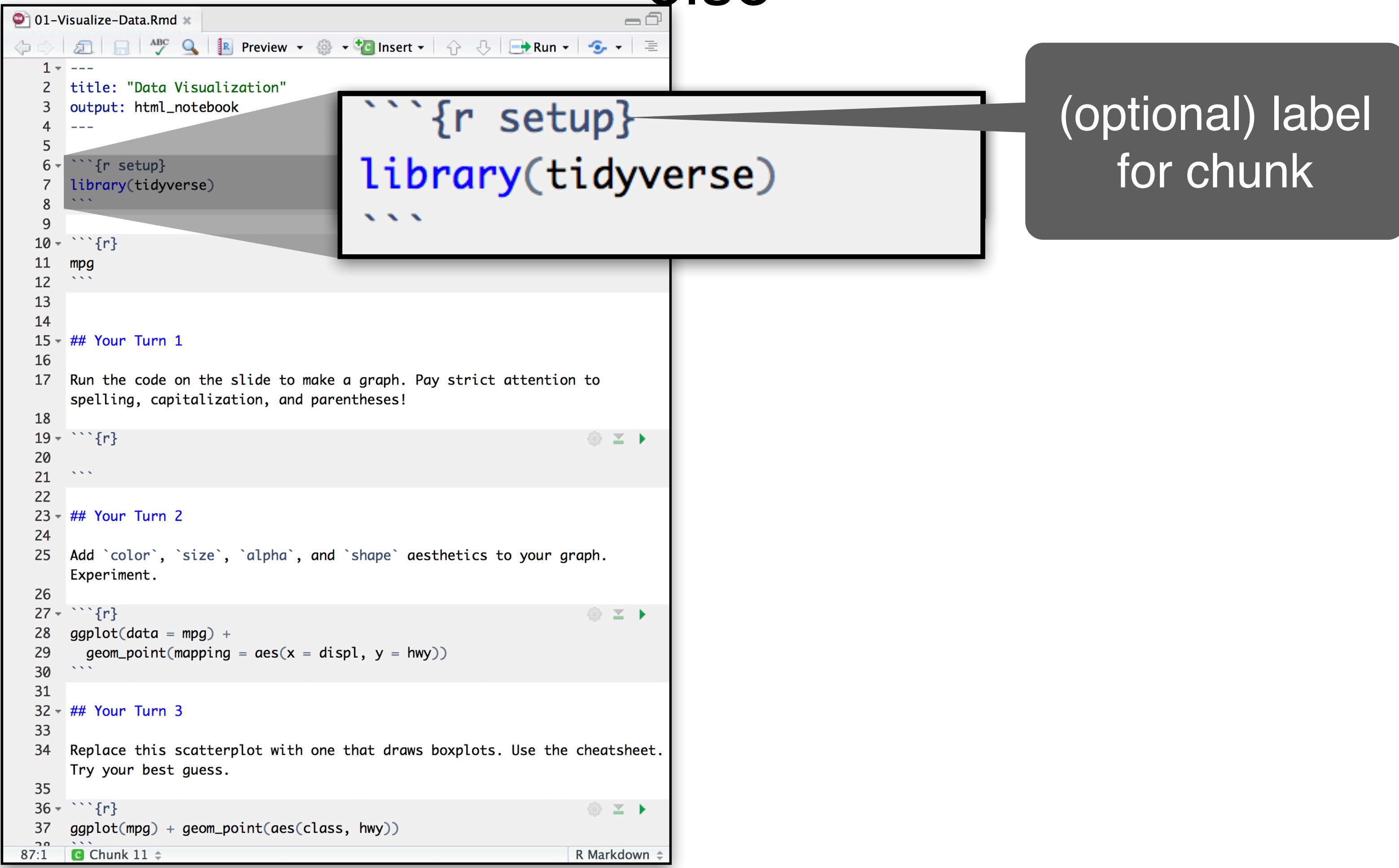
Color
Freshness
Aroma
Taste
Texture





Setup

The setup chunk is always run once before anything else



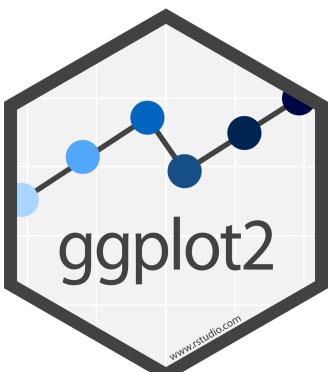
(optional) label for chunk

```
1 ---  
2 title: "Data Visualization"  
3 output: html_notebook  
4 ---  
5  
6 ```{r setup}  
7 library(tidyverse)  
8 ```  
9  
10 ```{r}  
11 mpg  
12  
13  
14  
15 ## Your Turn 1  
16  
17 Run the code on the slide to make a graph. Pay strict attention to  
spelling, capitalization, and parentheses!  
18  
19 ```{r}  
20  
21  
22  
23 ## Your Turn 2  
24  
25 Add `color`, `size`, `alpha`, and `shape` aesthetics to your graph.  
Experiment.  
26  
27 ```{r}  
28 ggplot(data = mpg) +  
29   geom_point(mapping = aes(x = displ, y = hwy))  
30  
31  
32 ## Your Turn 3  
33  
34 Replace this scatterplot with one that draws boxplots. Use the cheatsheet.  
Try your best guess.  
35  
36 ```{r}  
37 ggplot(mpg) + geom_point(aes(class, hwy))  
38  
87:1 | Chunk 11 | R Markdown
```

mpg

Fuel economy data for 38 models of
car

mpg



Quiz

Confer with your neighbours.

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

No peeking ahead!



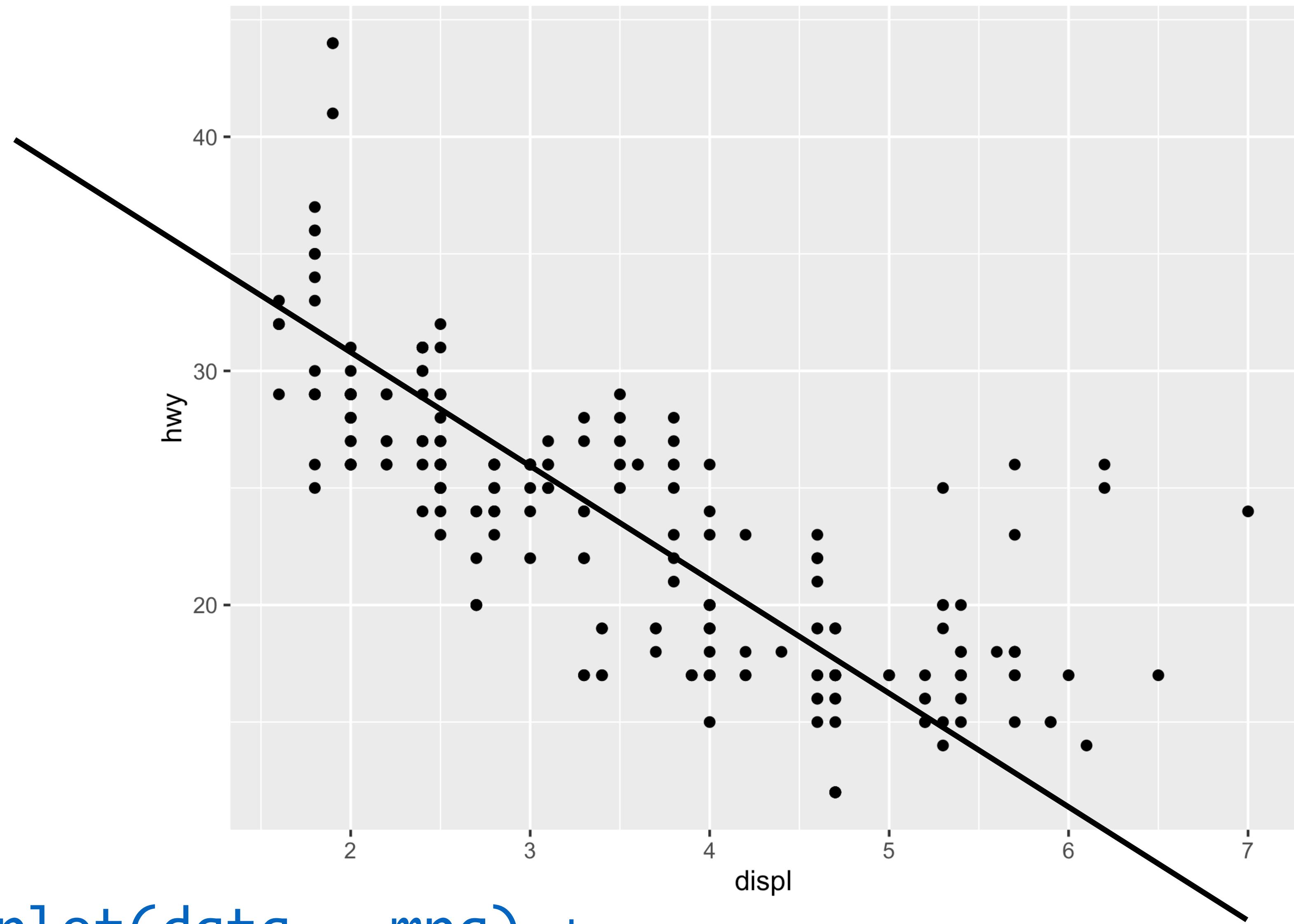
Your Turn 1

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))
```

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

data

+ before new line

type of layer

aes()

x

y variable

A template

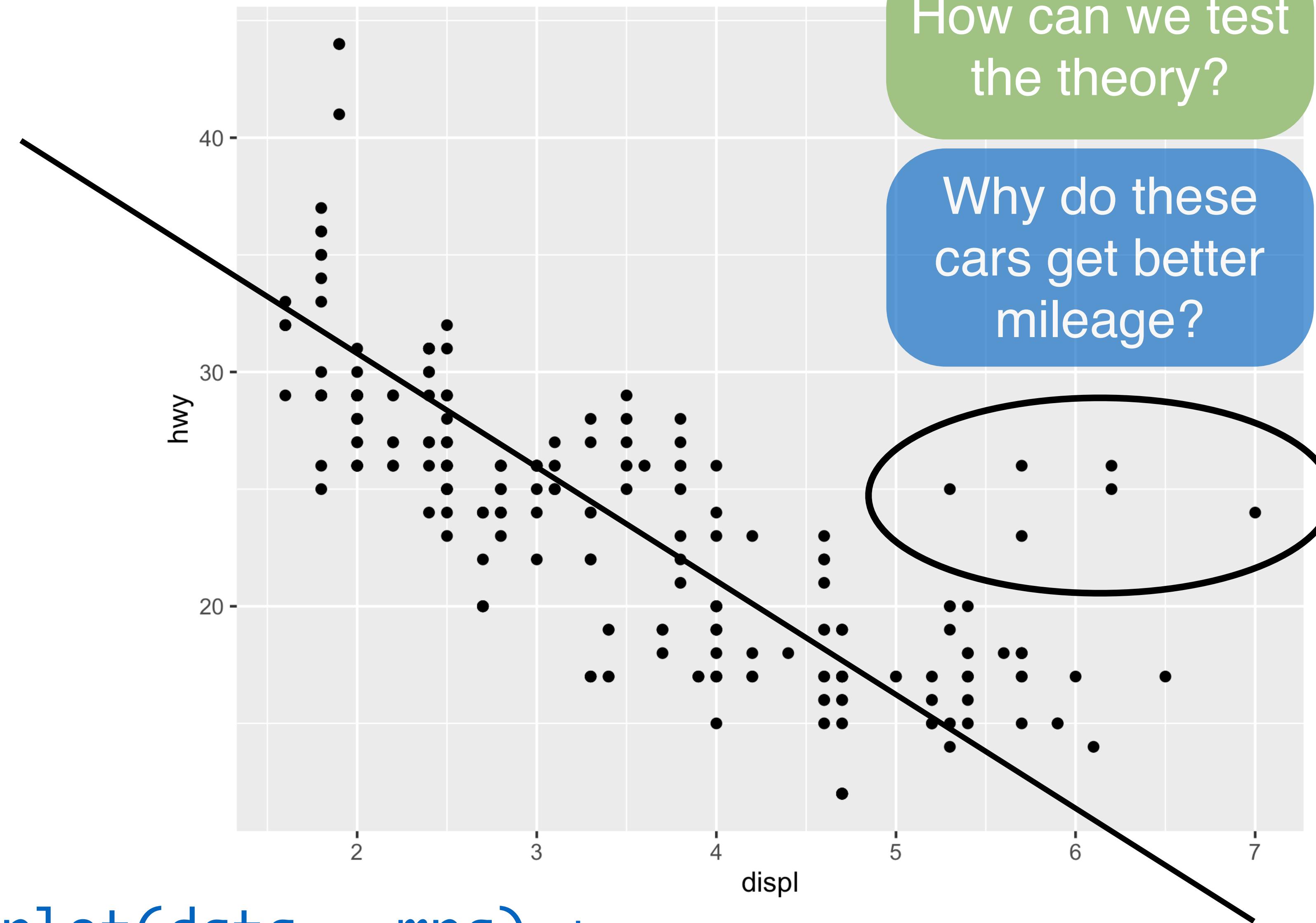
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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

A template

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

Mappings

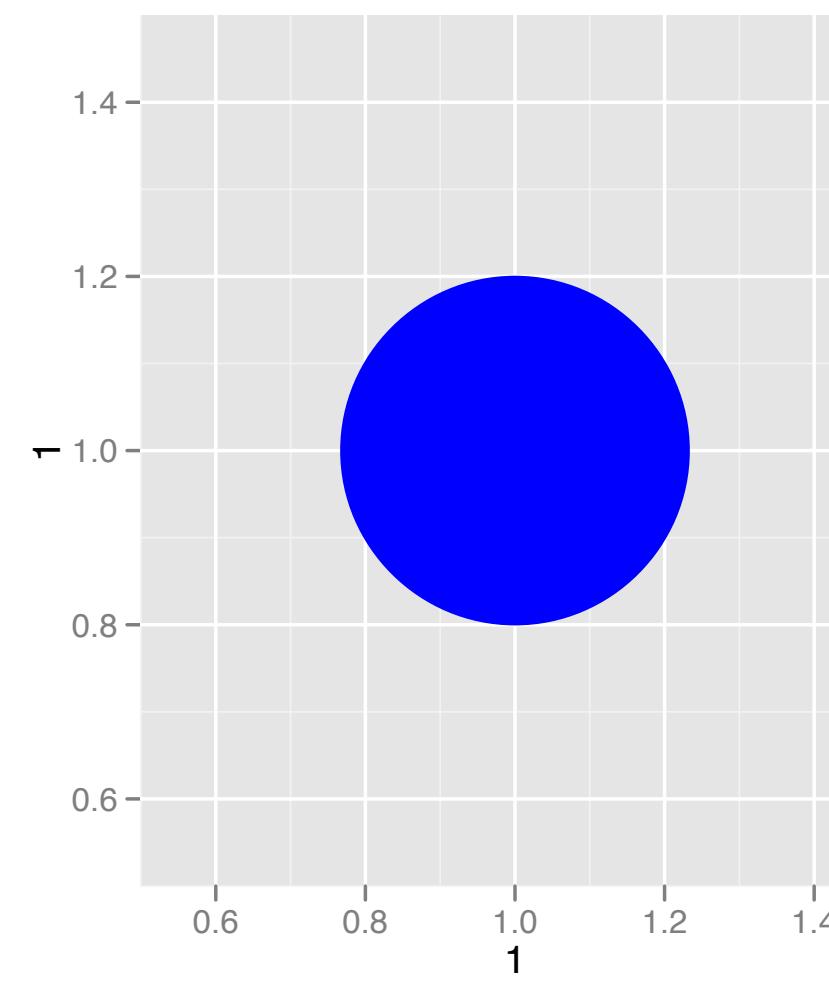
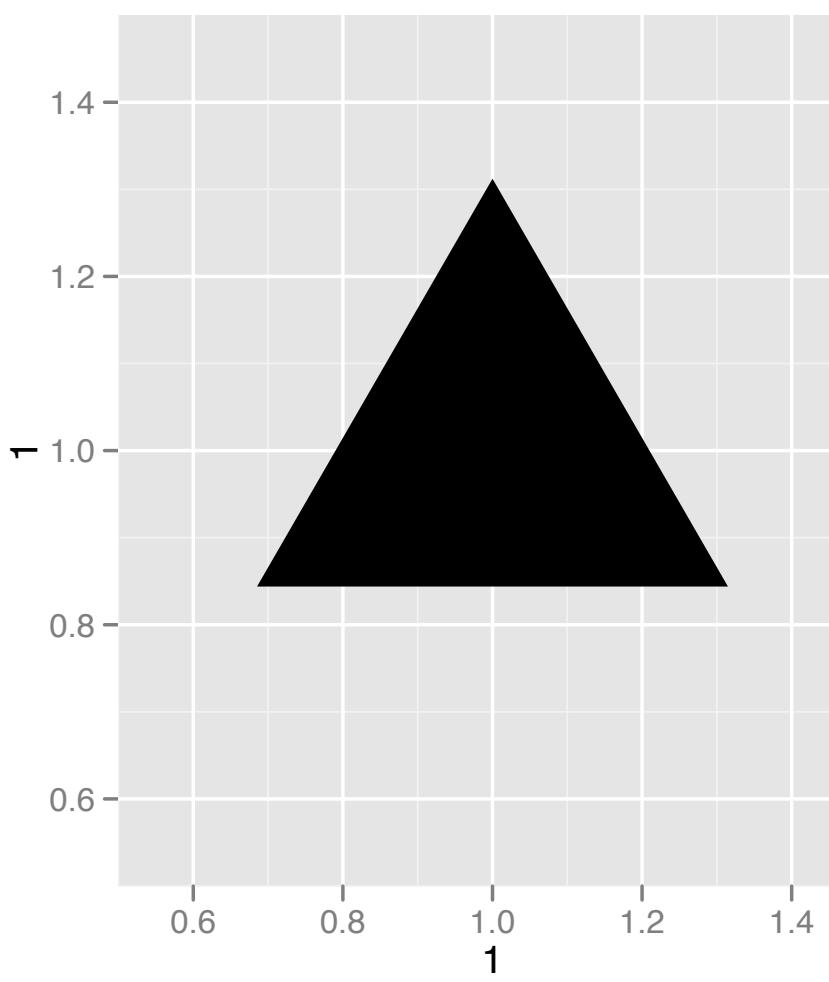
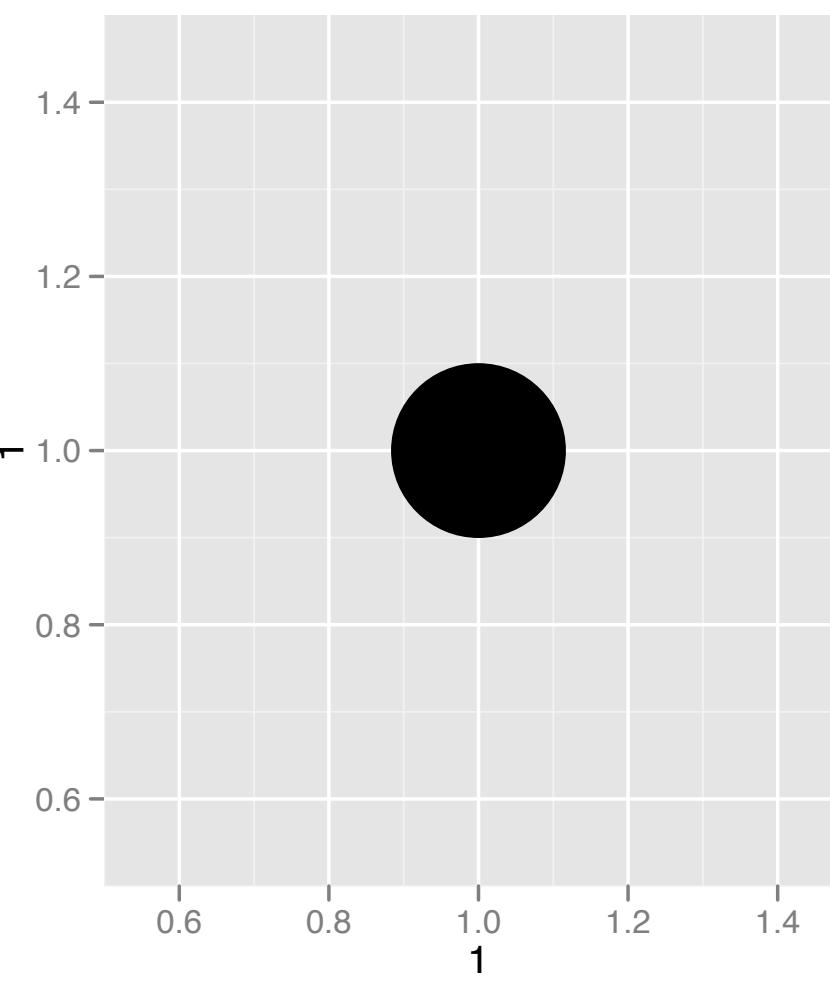
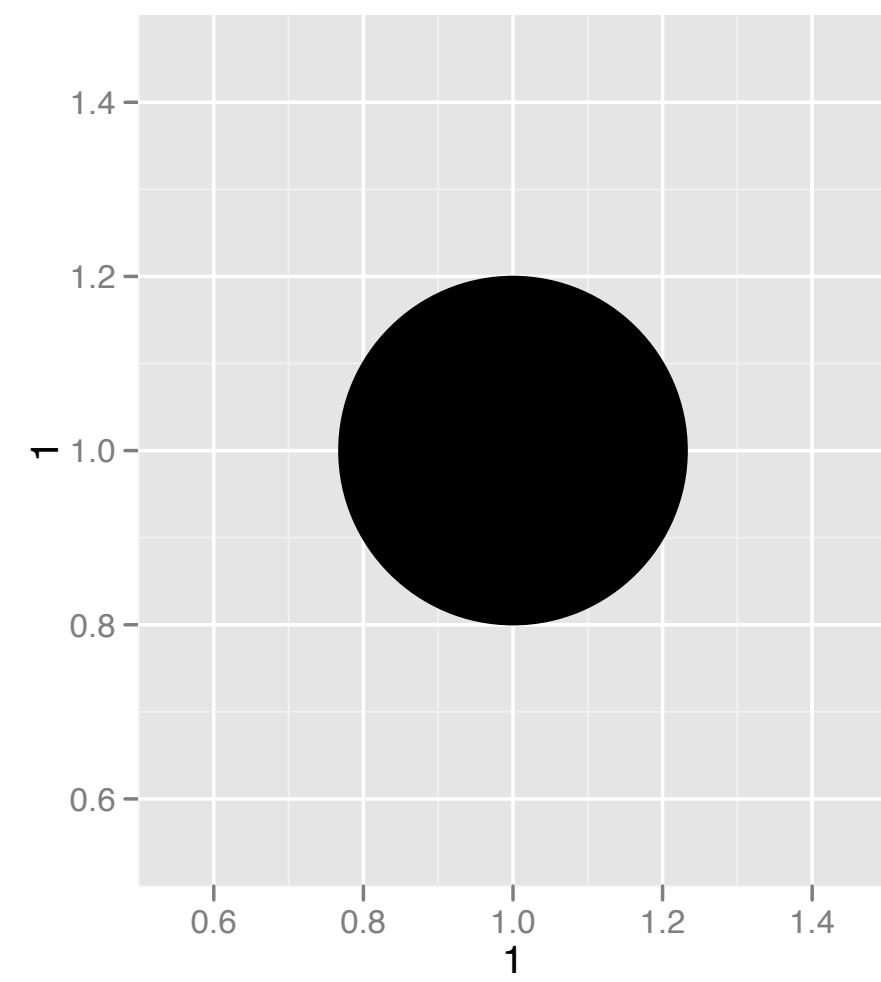


How can we test
the theory?

Why do these
cars get better
mileage?

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

Aesthetics



Visual Space

color

Red

Brown

Green

Aqua

Blue

Violet

Pink

Data Space

class

2seater

compact

midsize

minivan

pickup

subcompact

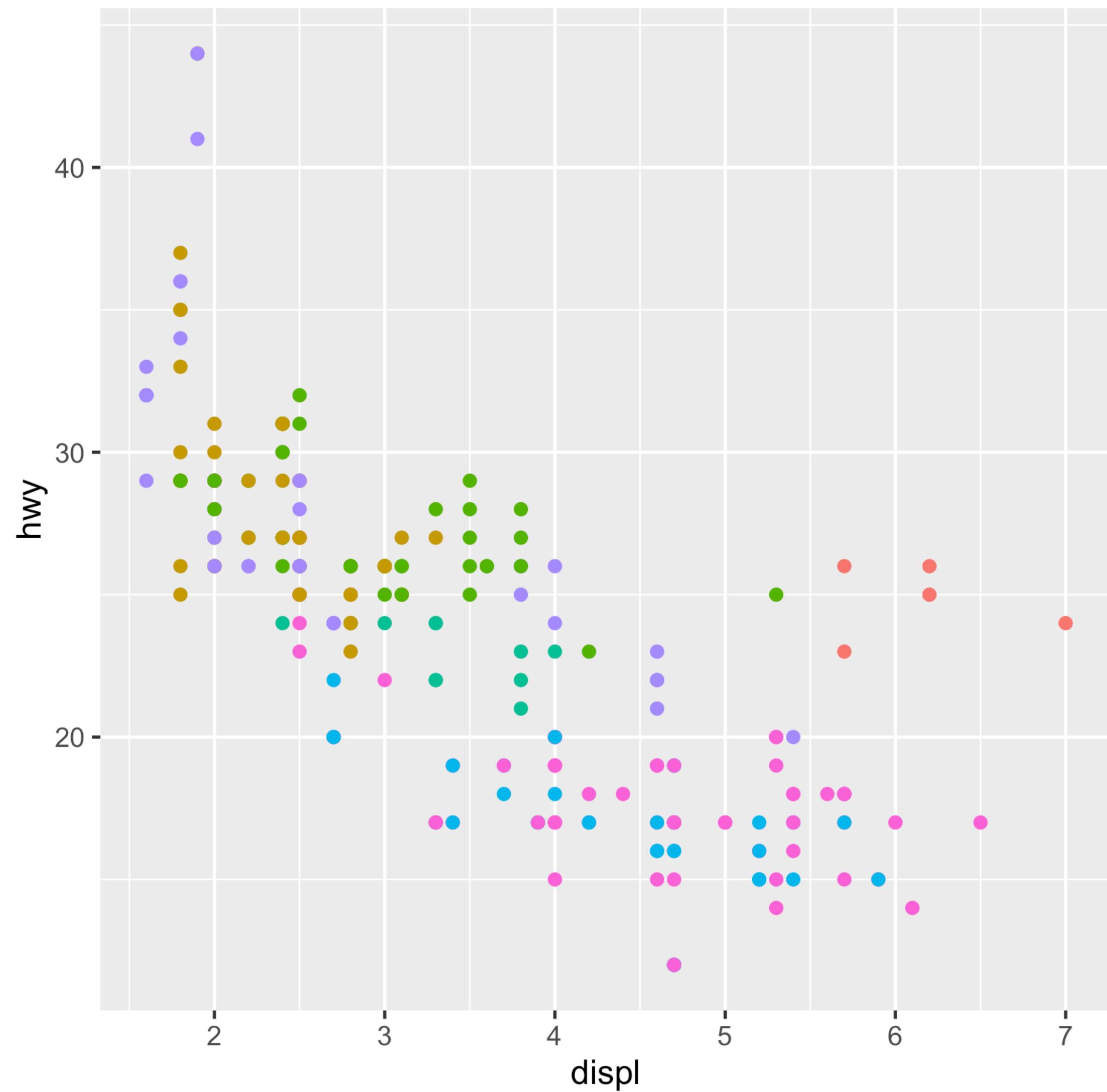
suv

Aesthetics

aesthetic
property

Variable to
map it to

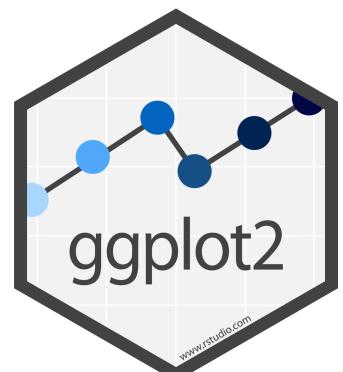
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))  
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, size = class))  
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, shape = class))  
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, alpha = class))
```



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

Legend added
automatically

class
2seater
compact
midsize
minivan
pickup
subcompact
suv



Your Turn 2

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

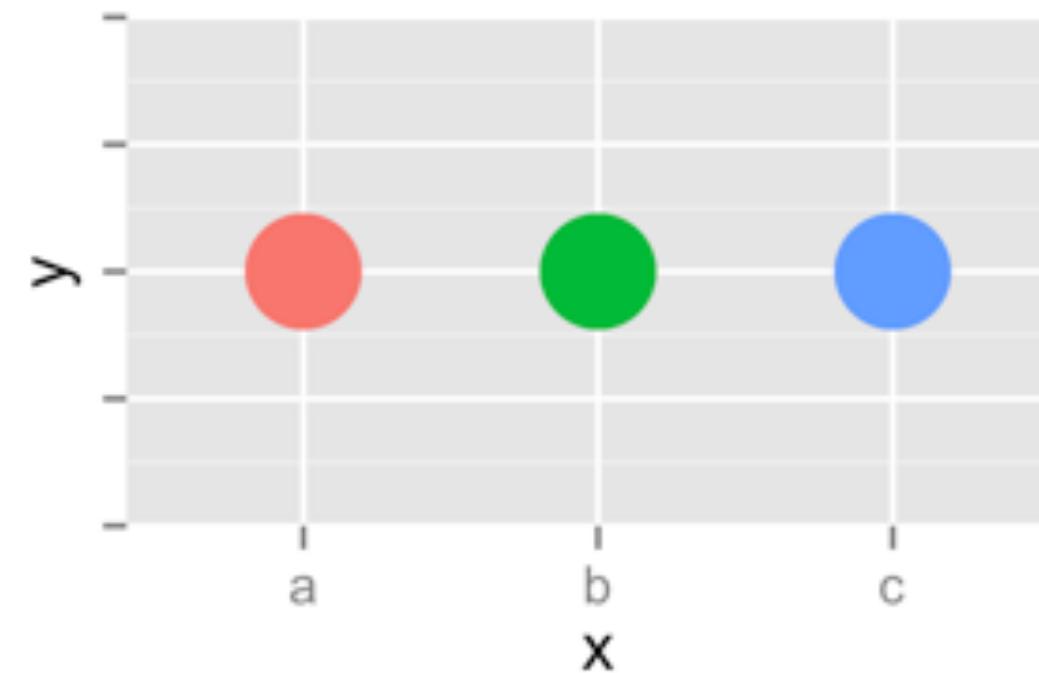
Do different things happen when you map aesthetics to discrete and continuous variables?

What happens when you use more than one aesthetic?



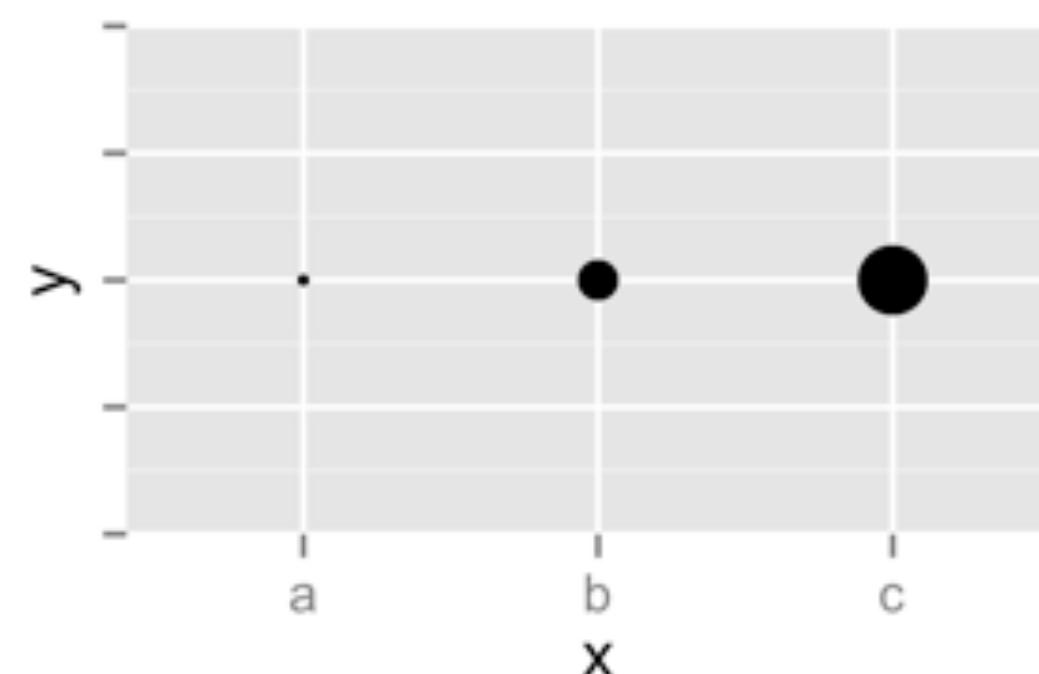
Color

Discrete

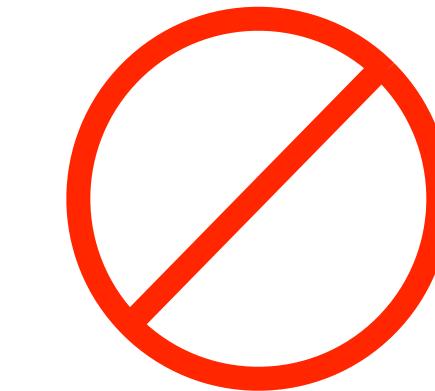
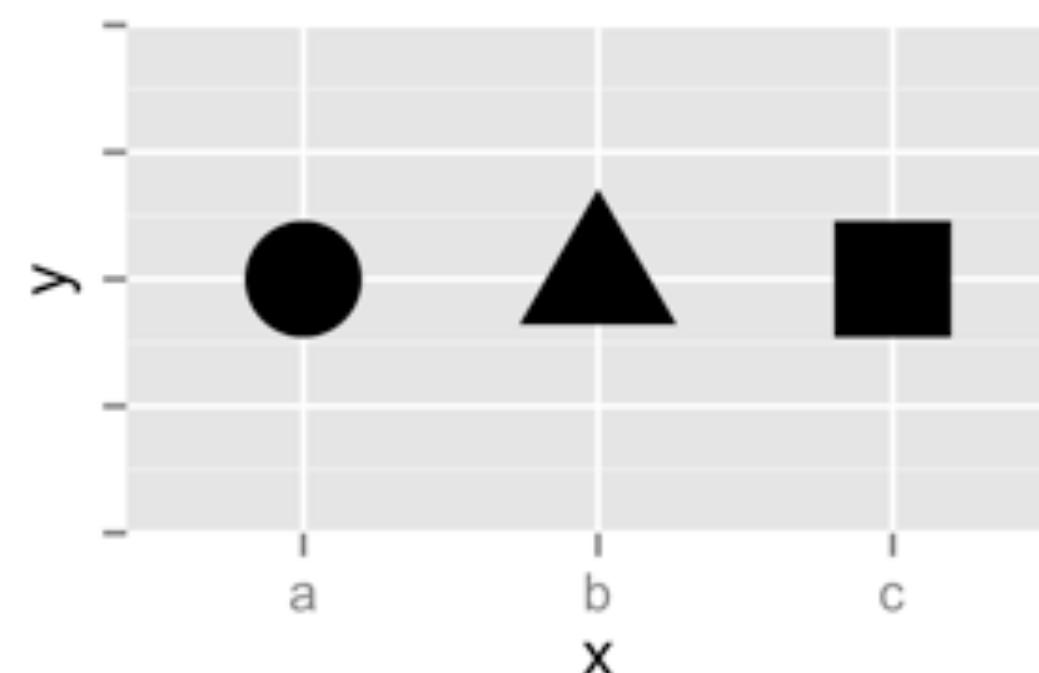


Size

Continuous

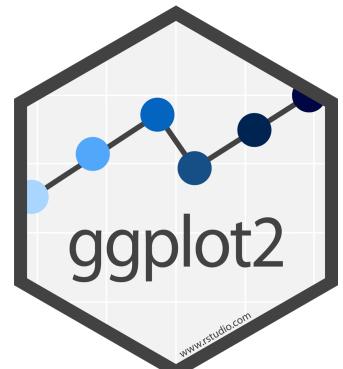
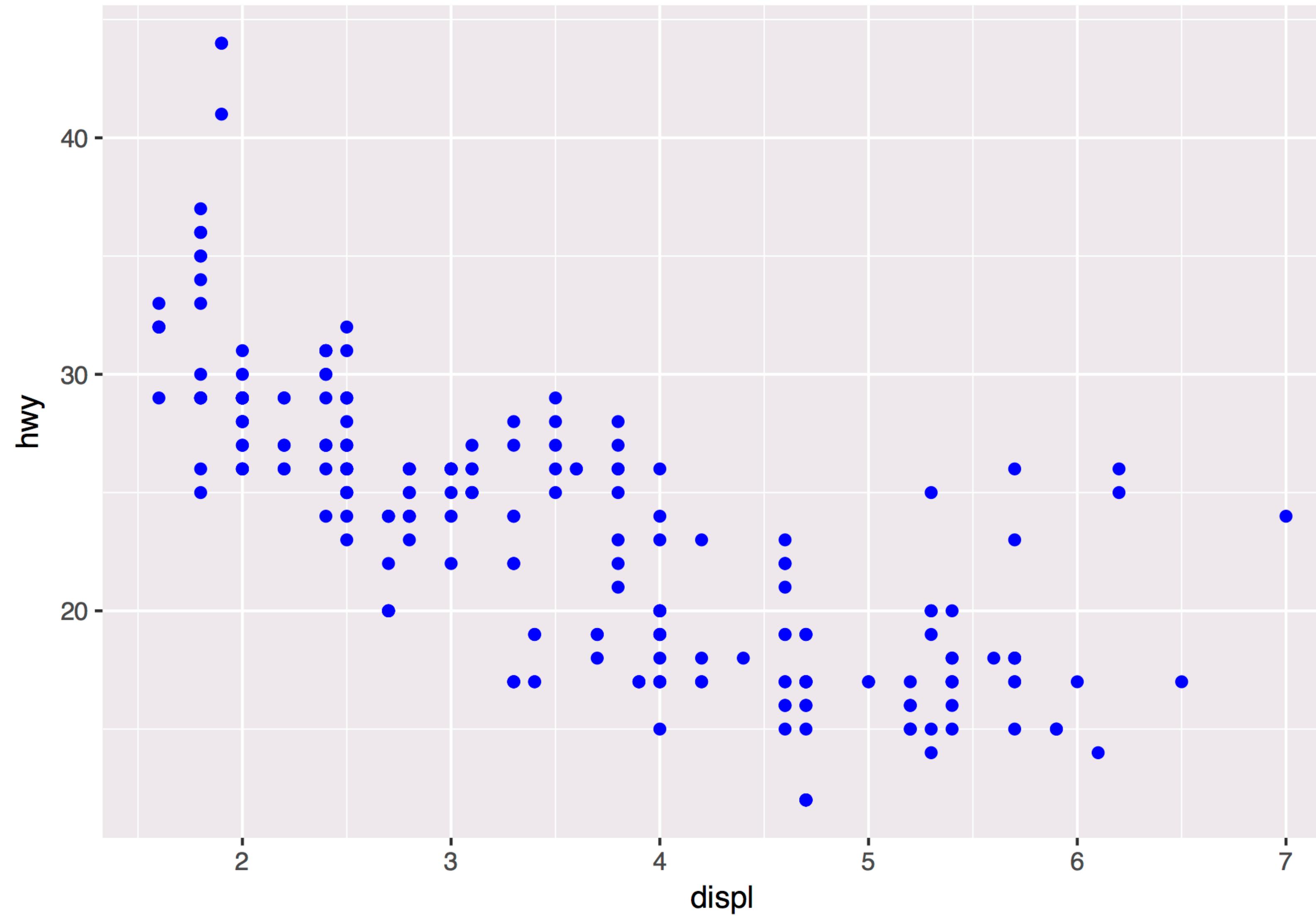


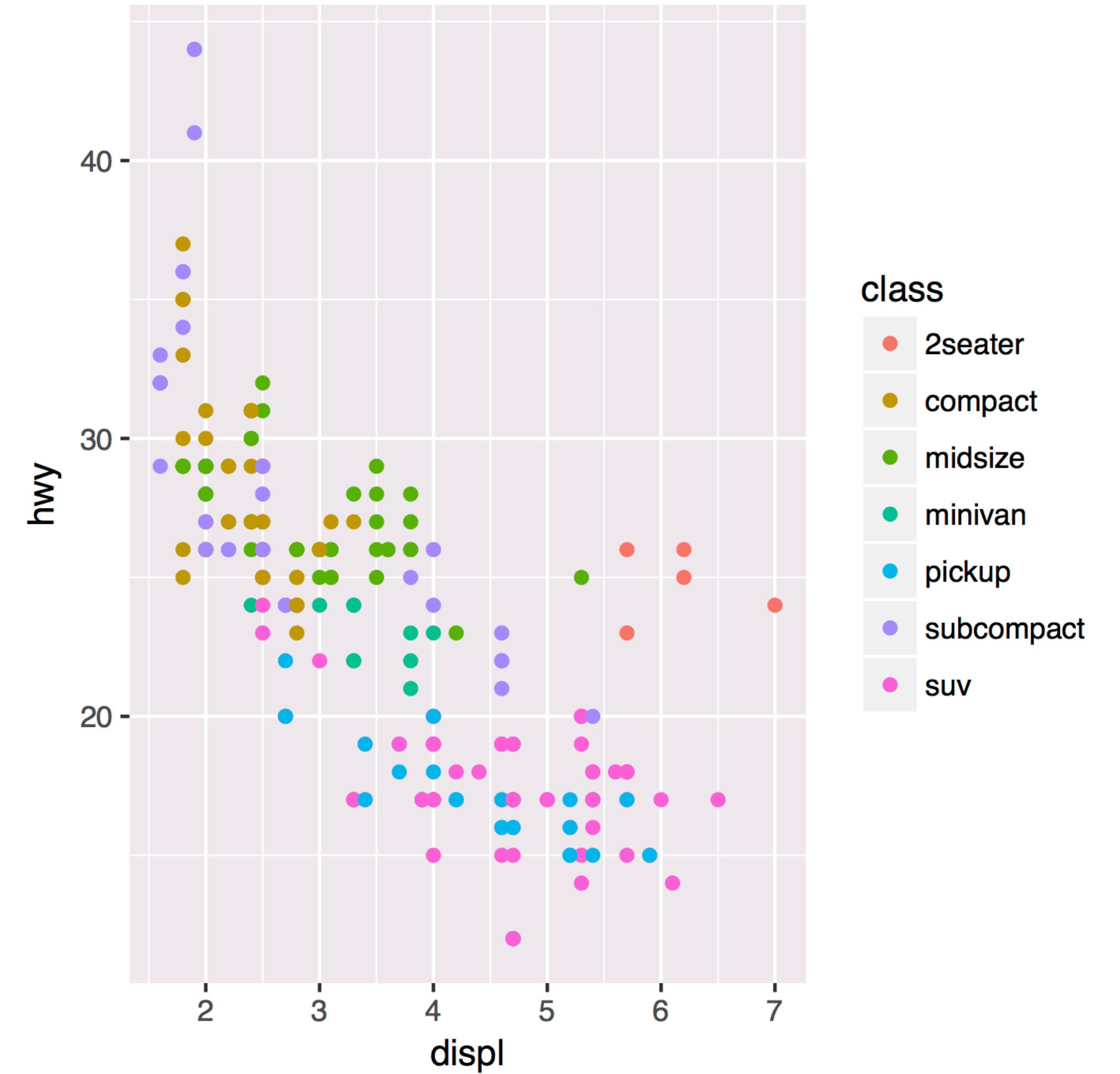
Shape



set vs. map

How would you make this plot?

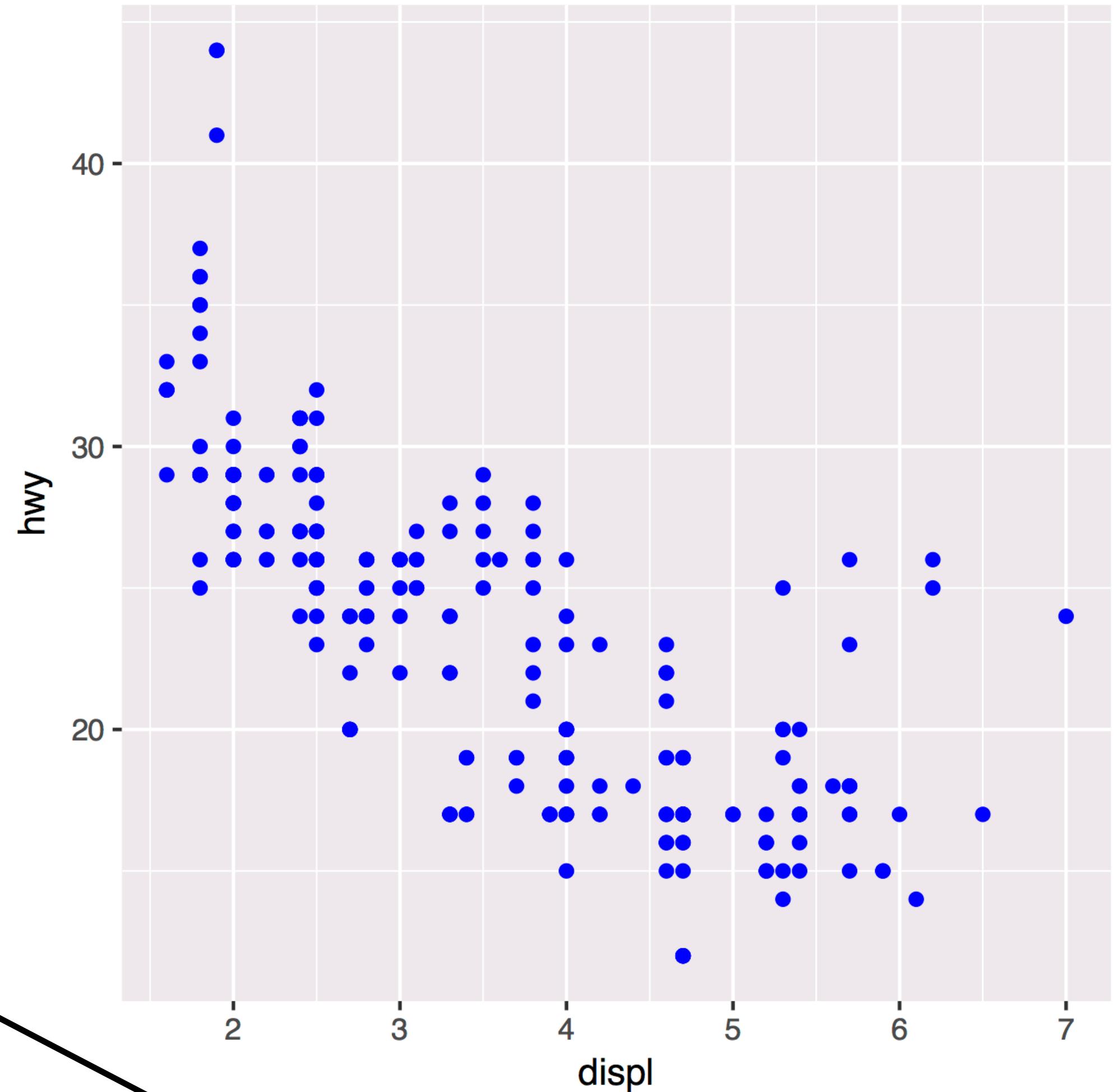




Inside of aes(): maps an aesthetic to a variable

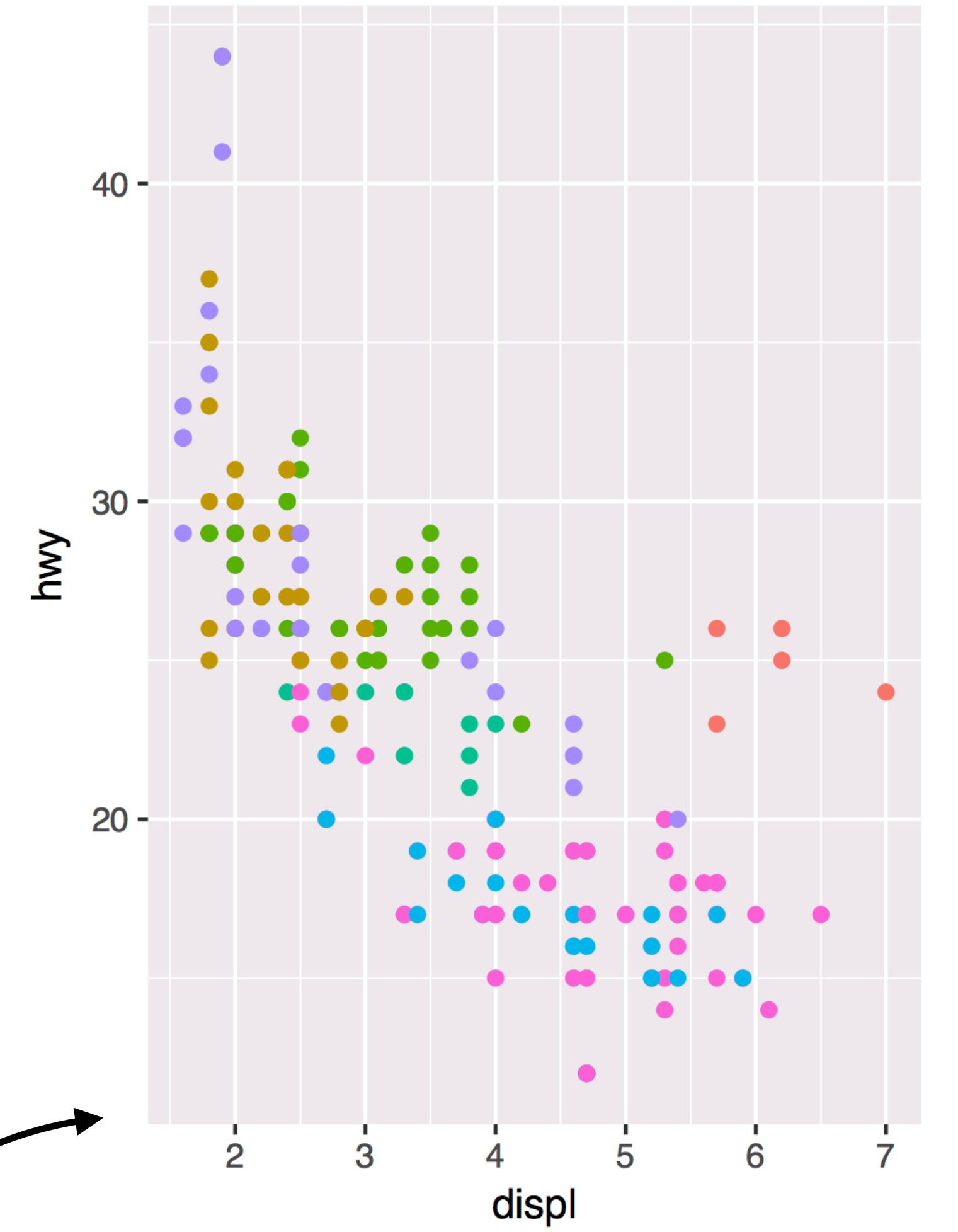
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



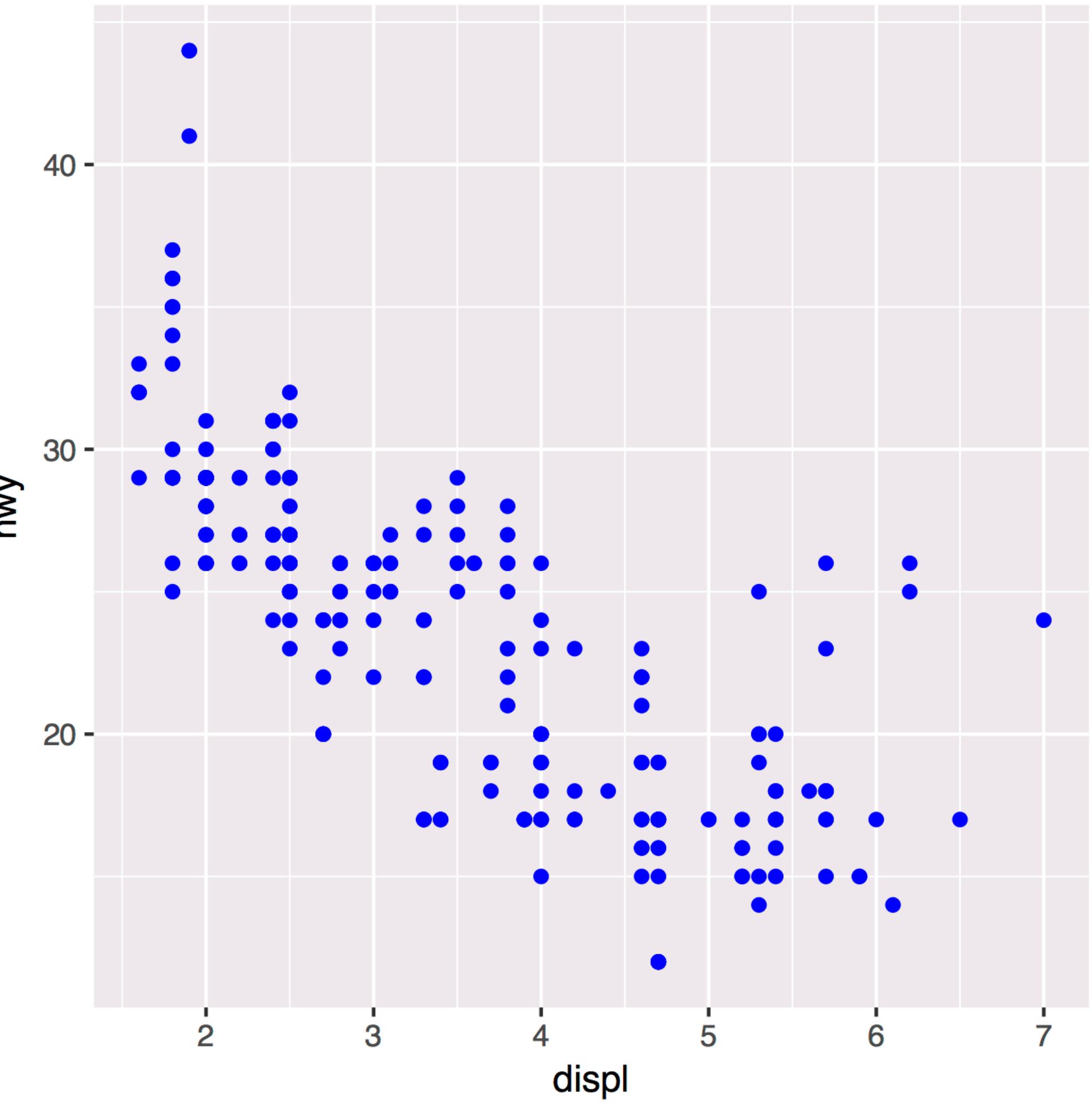
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



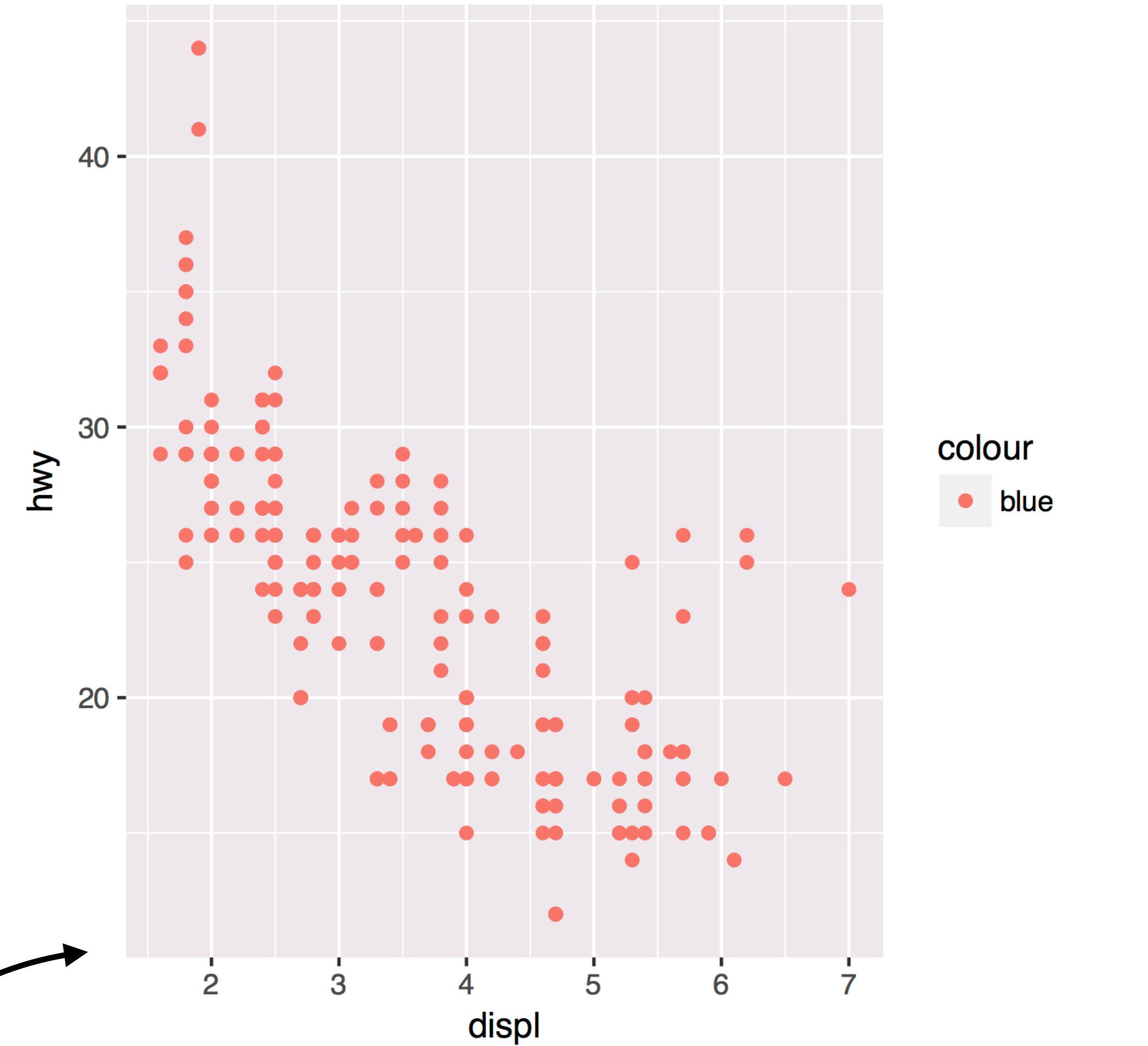
class

- 2seater
- compact
- midsize
- minivan
- pickup
- subcompact
- SUV

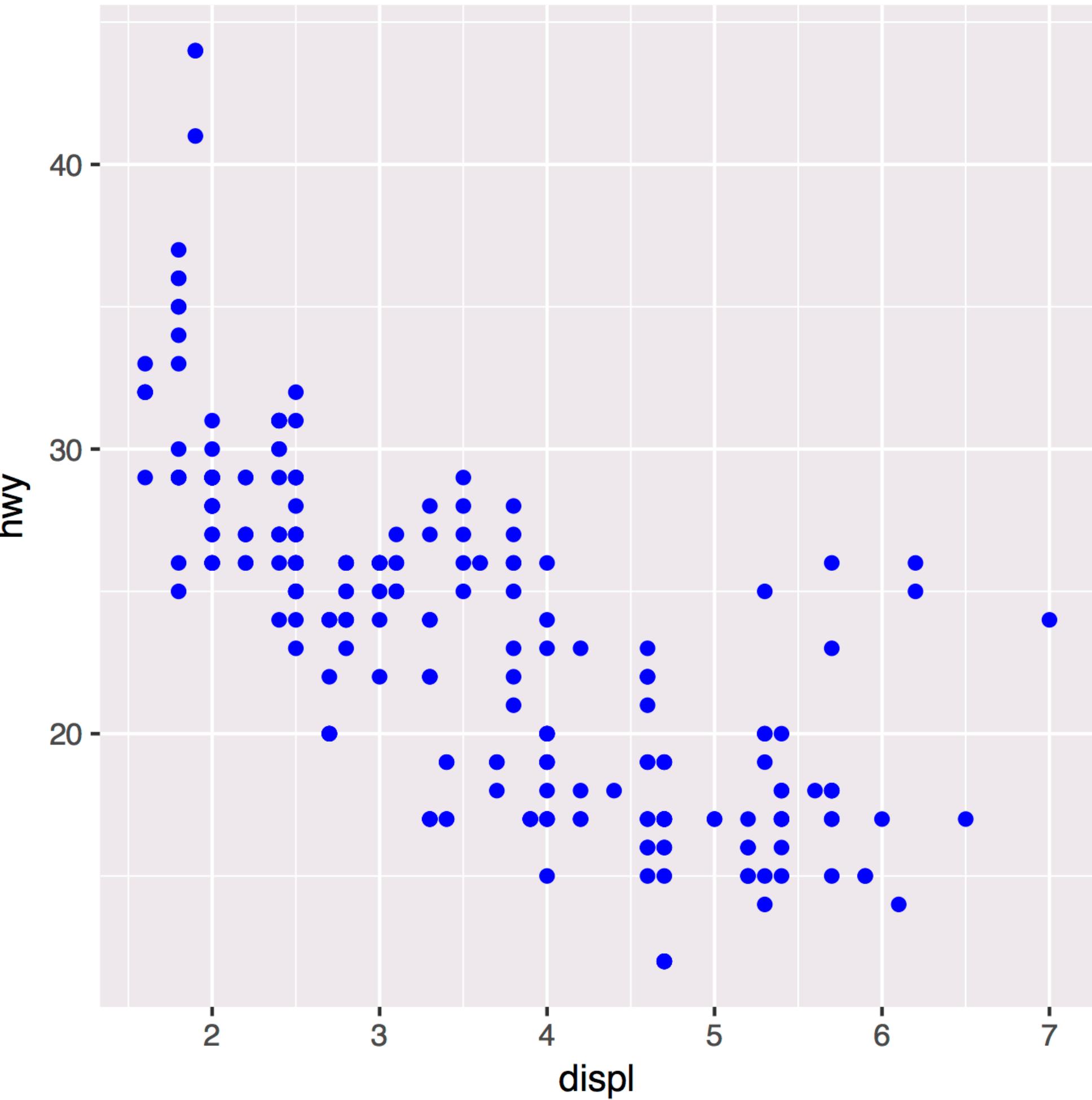


```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



colour
red



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

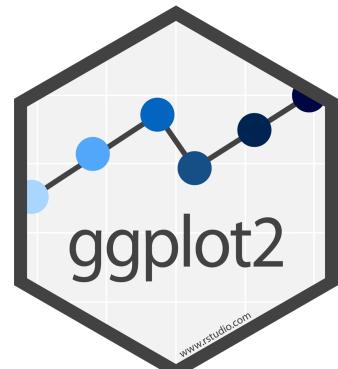
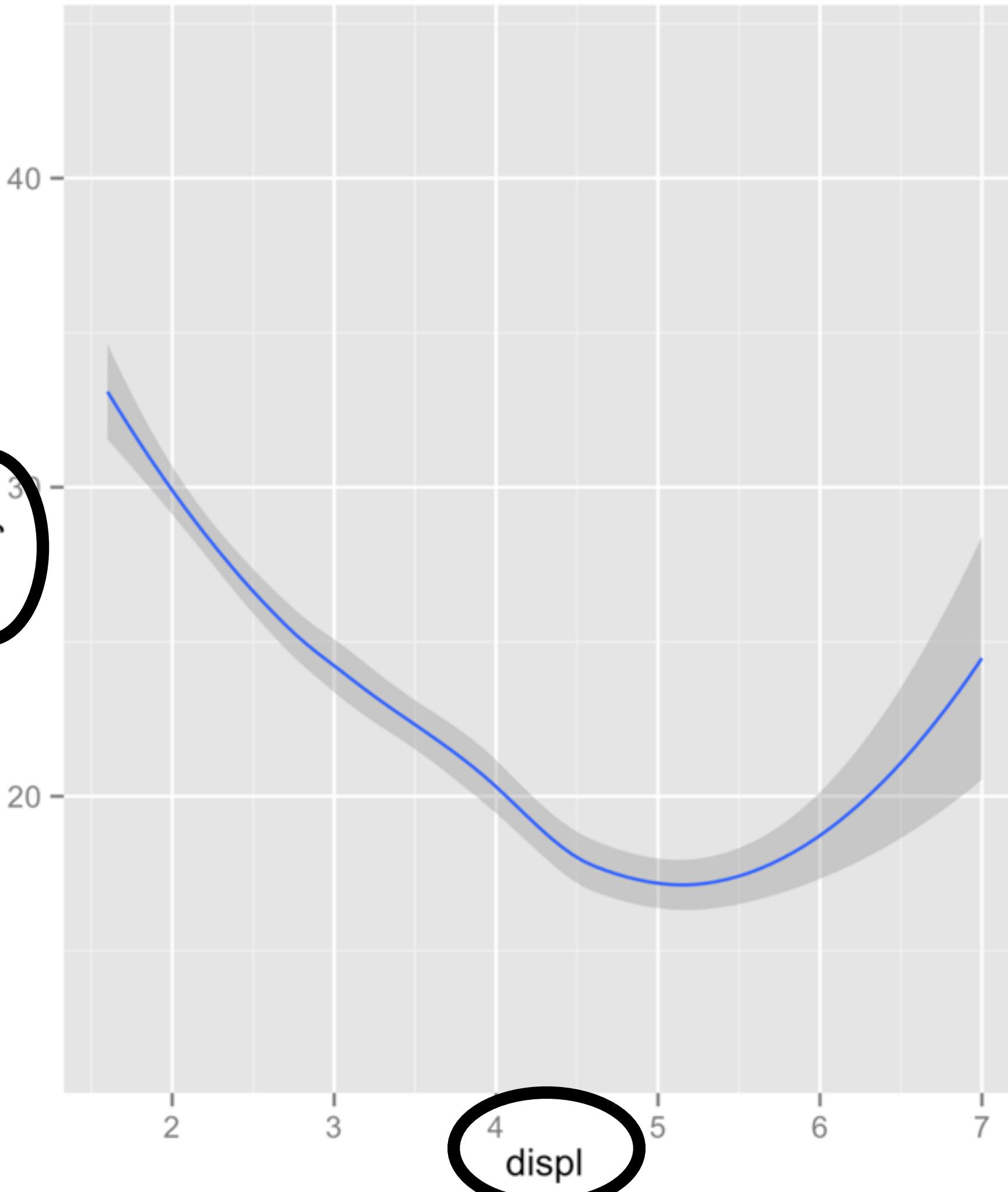
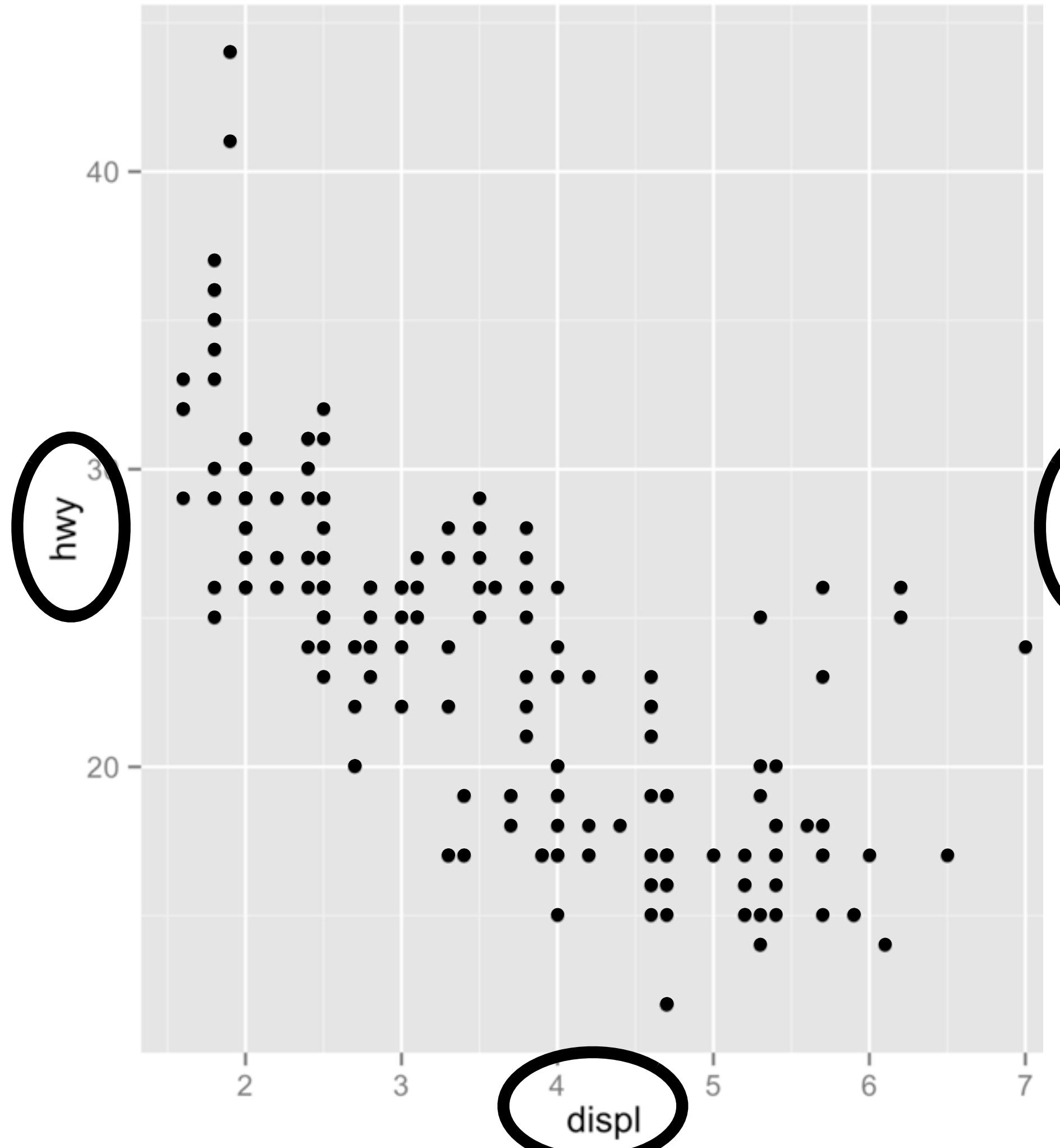
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), color = "blue")
```

Geoms



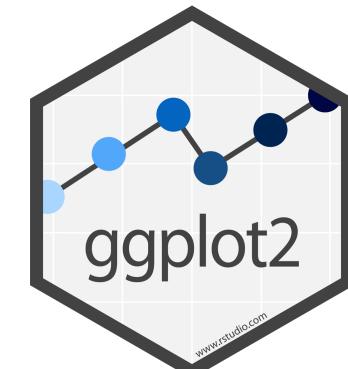
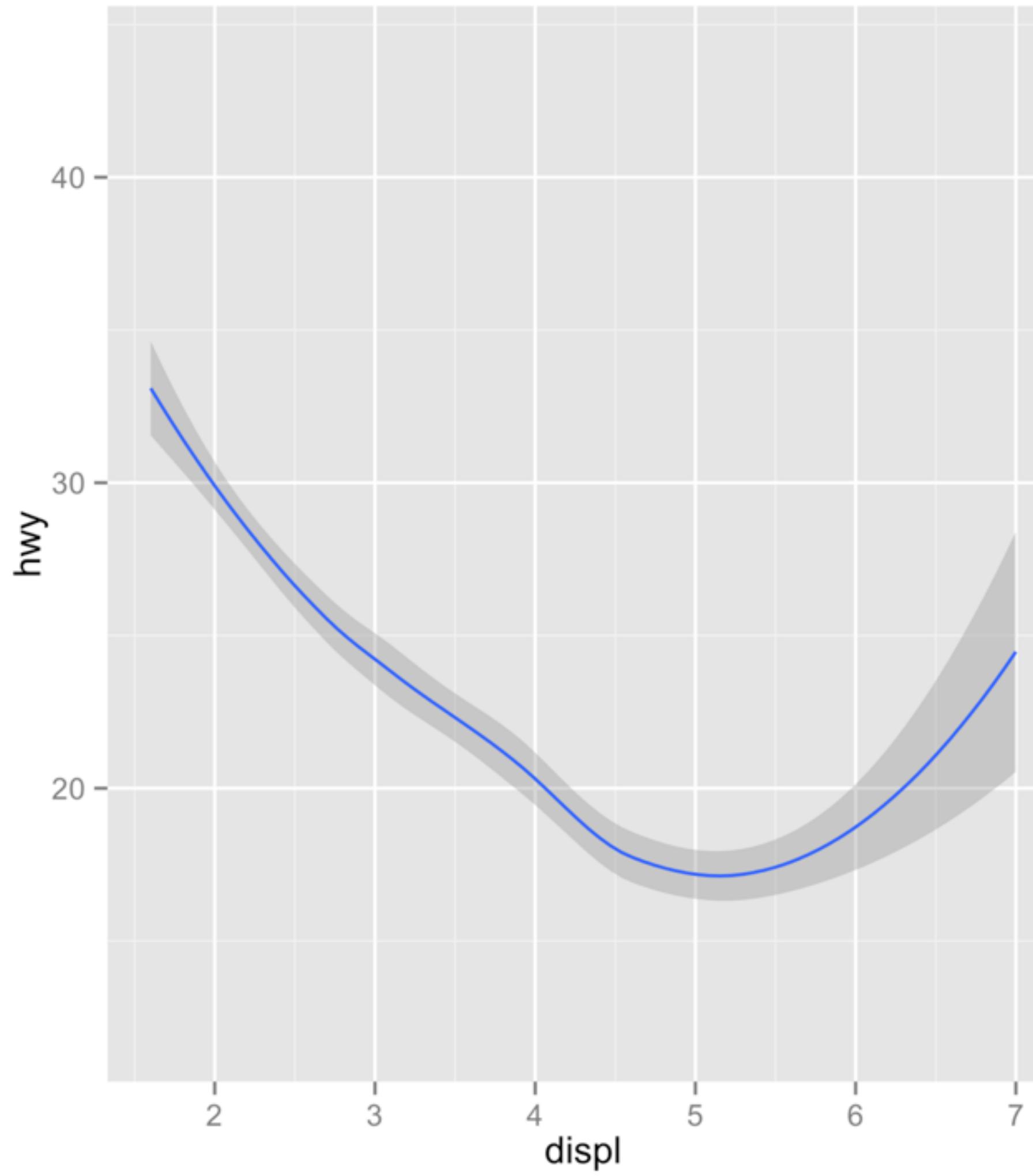
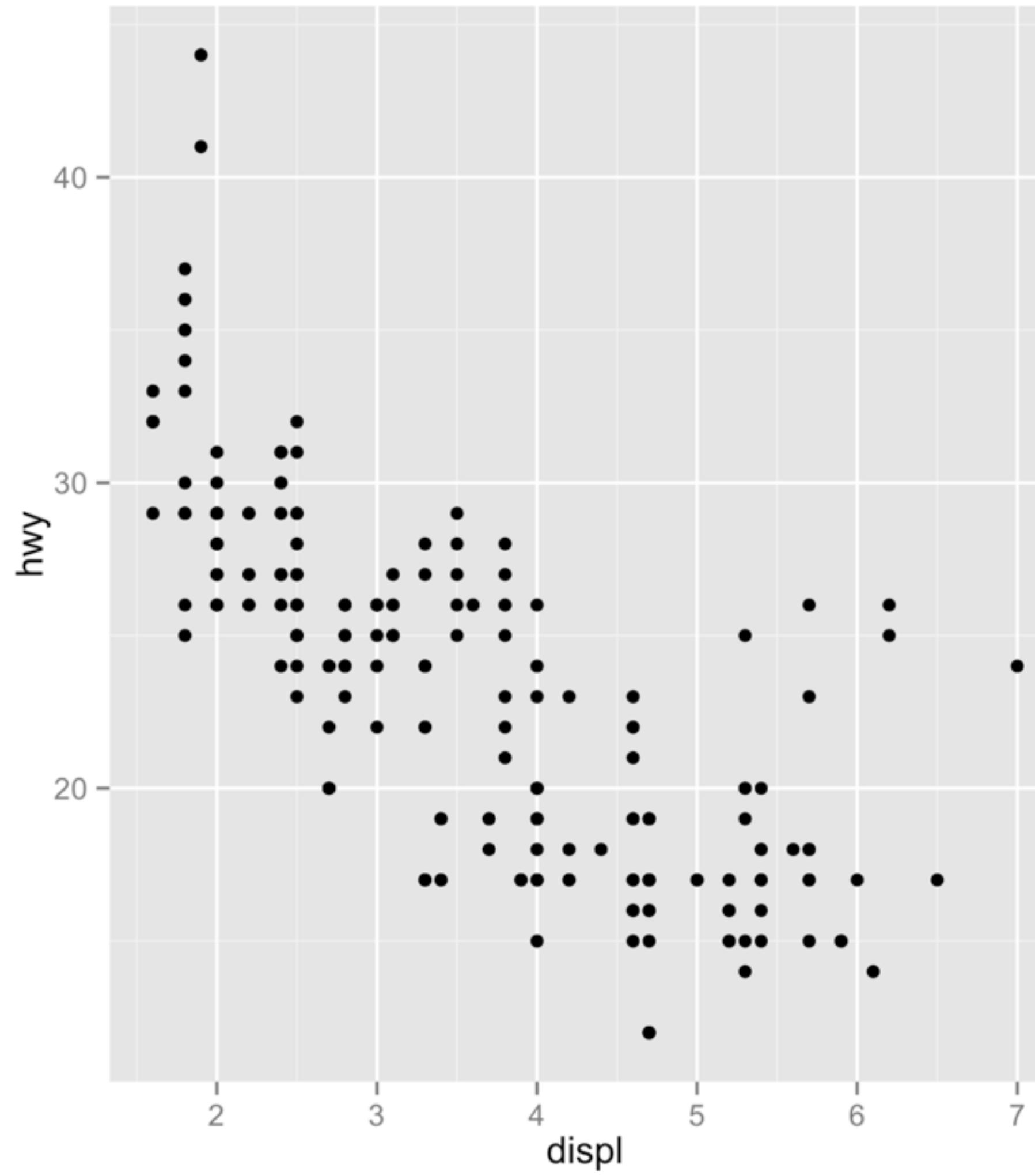
How are these plots similar?

Same: x var, y var, data



How are these plots different?

Different: geometric object (geom),
e.g. the visual object used to represent the data

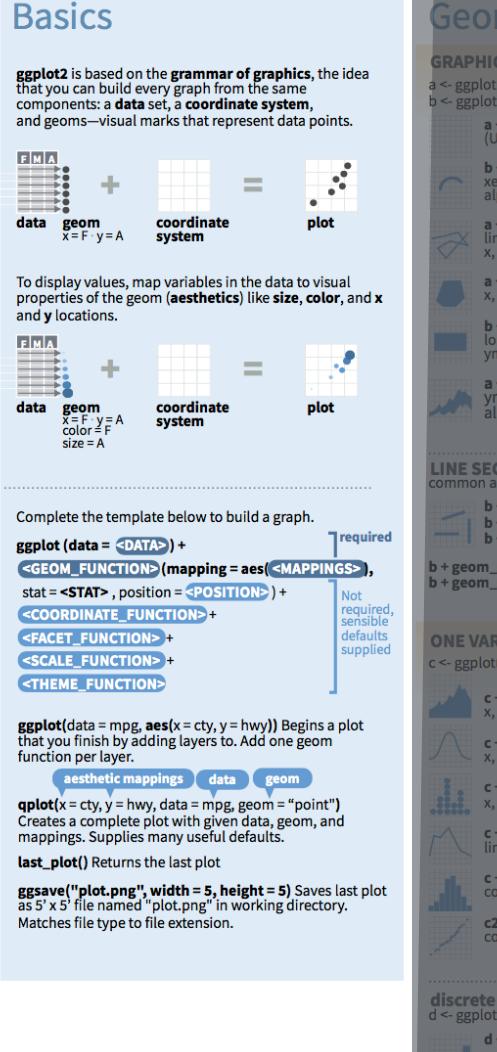


geoms

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

Each requires a mapping argument.

Data Visualization with ggplot2 :: CHEAT SHEET



R Studio

CC by RStudio

geom_ functions

Geoms Use a geom function to represent data points, use the geom's aesthetic properties to represent variables.
Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

- a + geom_blank()**
(Useful for expanding limits)
- b + geom_curve(aes(yend = lat + 1, xend = long + 1, curvature = z))** - x, yend, y, end, alpha, angle, color, curvature, linetype, size
- a + geom_path(lineend = "butt", linejoin = "round", linemetre = 1)** - x, y, alpha, color, group, linetype, size
- a + geom_polygon(aes(group = group))** - x, y, alpha, color, fill, group, linetype, size
- b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

- common aesthetics: x, y, alpha, color, linetype, size
- b + geom_abline(aes(intercept = 0, slope = 1))**
 - b + geom_hline(aes(intercept = lat))**
 - b + geom_vline(aes(xintercept = long))**
 - b + geom_segment(aes(yend = lat + 1, xend = long + 1))**
 - b + geom_spoke(aes(angle = 1:1155, radius = 1))**

ONE VARIABLE continuous

- ```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```
- c + geom\_area(stat = "bin")** - x, y, alpha, color, fill, group, linetype, size
  - c + geom\_boxplot()** - x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, size
  - f + geom\_dotplot(binaxis = "y", stackdir = "center")** - x, y, alpha, color, fill, group
  - f + geom\_density(kernel = "gaussian")** - x, y, alpha, color, fill, group, linetype, size
  - f + geom\_dotplot()** - x, y, alpha, color, fill
  - c + geom\_freqpoly()** - x, y, alpha, color, group, linetype, size
  - c + geom\_histogram(binwidth = 5)** - x, y, alpha, color, fill, linetype, size, weight
  - c2 + geom\_qq(aes(sample = hwy))** - x, y, alpha, color, fill, linetype, size, weight

### discrete

- ```
d <- ggplot(mpg, aes(fl))
d + geom_bar()
```
- d + geom_bar()** - x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

- ```
e <- ggplot(mpg, aes(cty, hwy))
```
- e + geom\_label(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
  - e + geom\_litter(height = 2, width = 2)** - x, y, alpha, color, fill, shape, size
  - e + geom\_point()** - x, y, alpha, color, fill, shape, size, stroke
  - e + geom\_quantile()** - x, y, alpha, color, group, linetype, size, weight
  - e + geom\_rug(sides = "bl")** - x, y, alpha, color, linetype, size
  - e + geom\_smooth(method = lm)** - x, y, alpha, color, fill, group, linetype, size, weight
  - e + geom\_text(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

#### discrete x , continuous y

- ```
f <- ggplot(mpg, aes(class, hwy))
```
- f + geom_col()** - x, y, alpha, color, fill, group, linetype, size
 - f + geom_boxplot()** - x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

- f + geom_dotplot(binaxis = "center")** - x, y, alpha, color, fill, group
- f + geom_errorbar()** - x, y, max, min, alpha, color, fill, group, linetype, size, width (also **geom_errorbarh()**)
- j + geom_linerange()** - x, y, ymin, alpha, color, group, linetype, size
- j + geom_pointrange()** - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

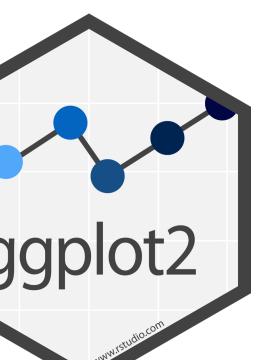
discrete x , discrete y

- ```
g <- ggplot(diamonds, aes(cut, color))
```
- g + geom\_count()** - x, y, alpha, color, fill, shape, size, stroke

### THREE VARIABLES

- ```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
```
- ```
l <- ggplot(seals, aes(long, lat))
```
- l + geom\_contour(aes(z = z))** - x, y, z, alpha, colour, group, linetype, size, weight
  - l + geom\_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)** - x, y, alpha, fill
  - l + geom\_tile(aes(fill = z))** - x, y, alpha, color, fill, linetype, size, width

ggplot2



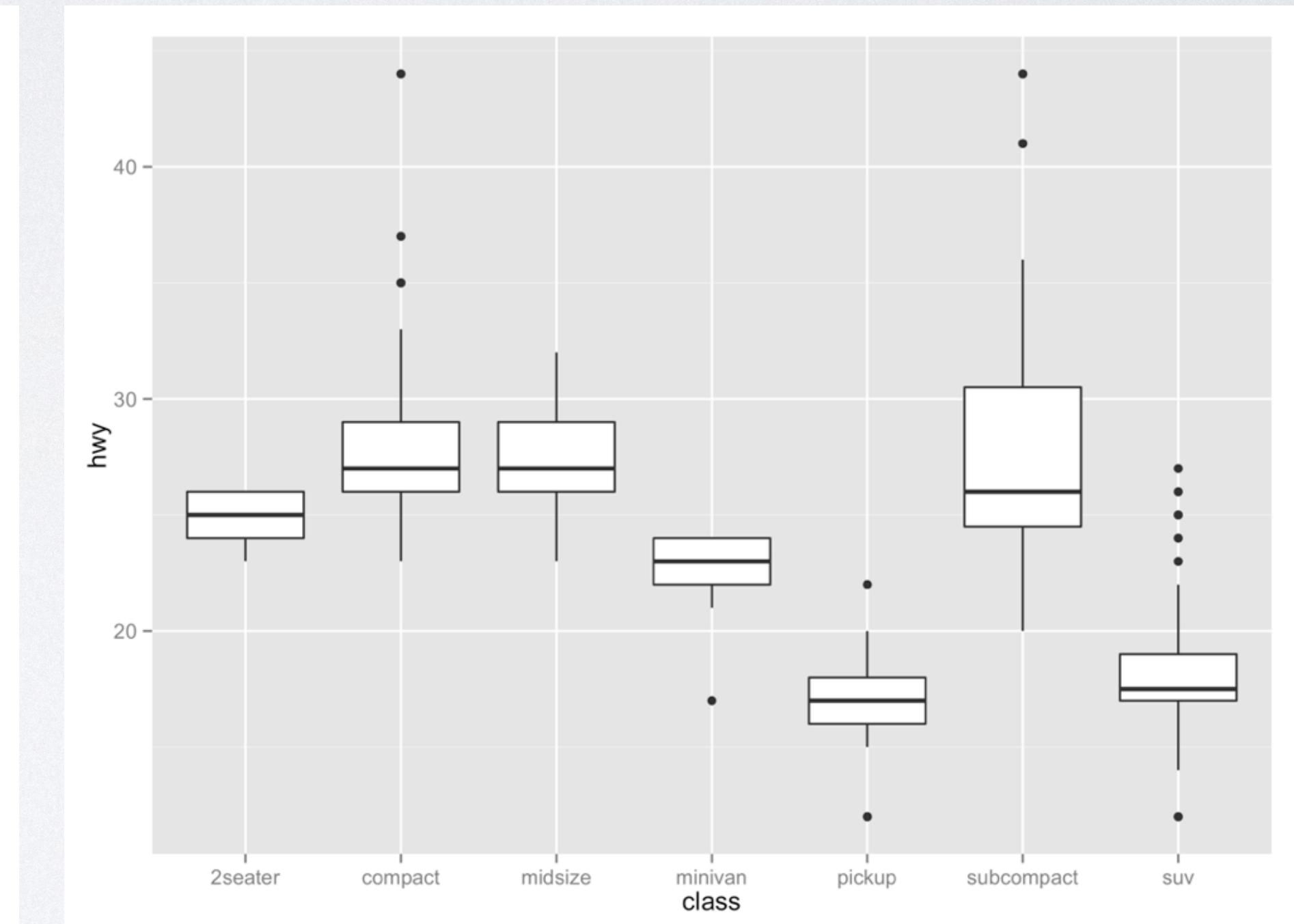
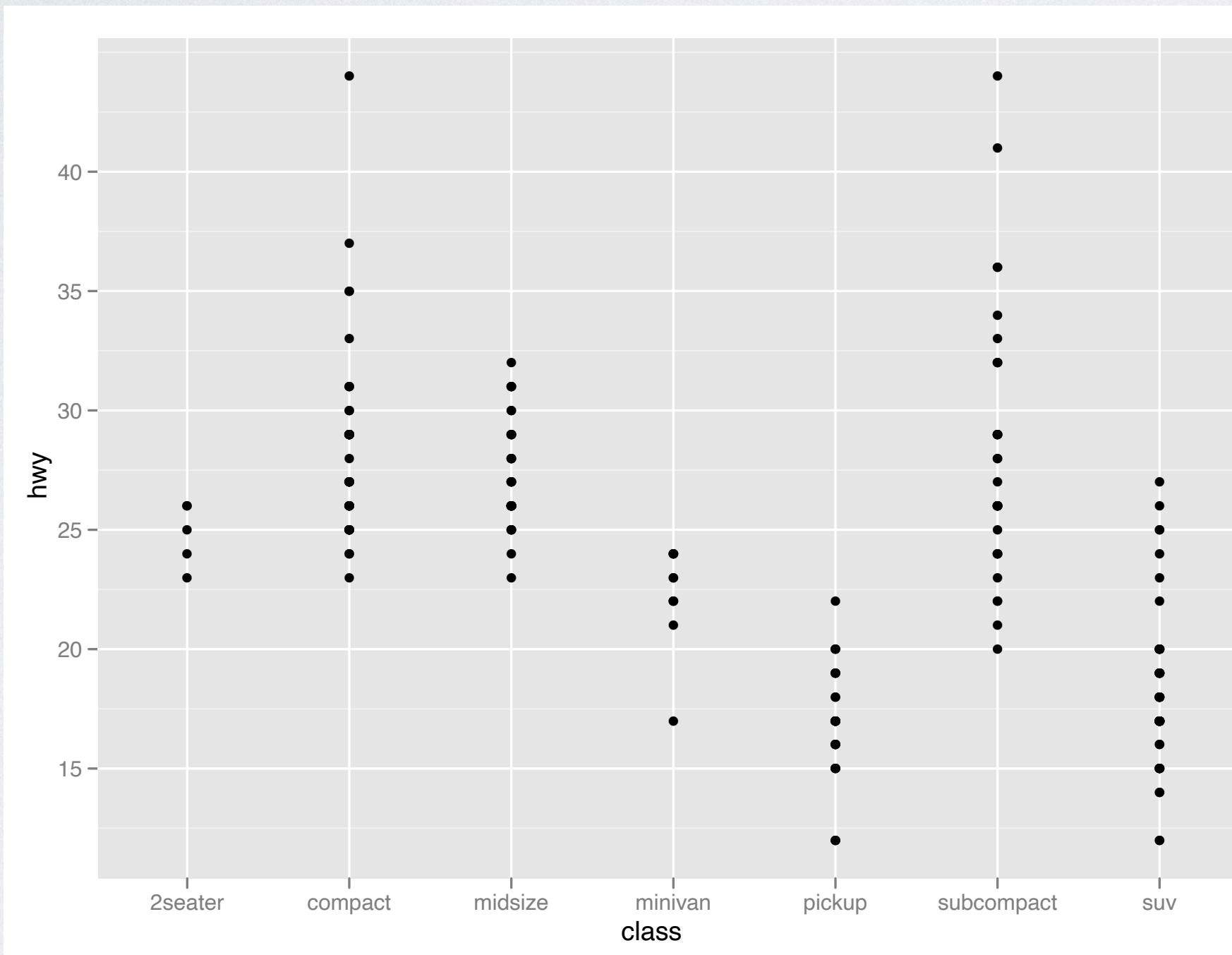
# Your Turn

Pair up.



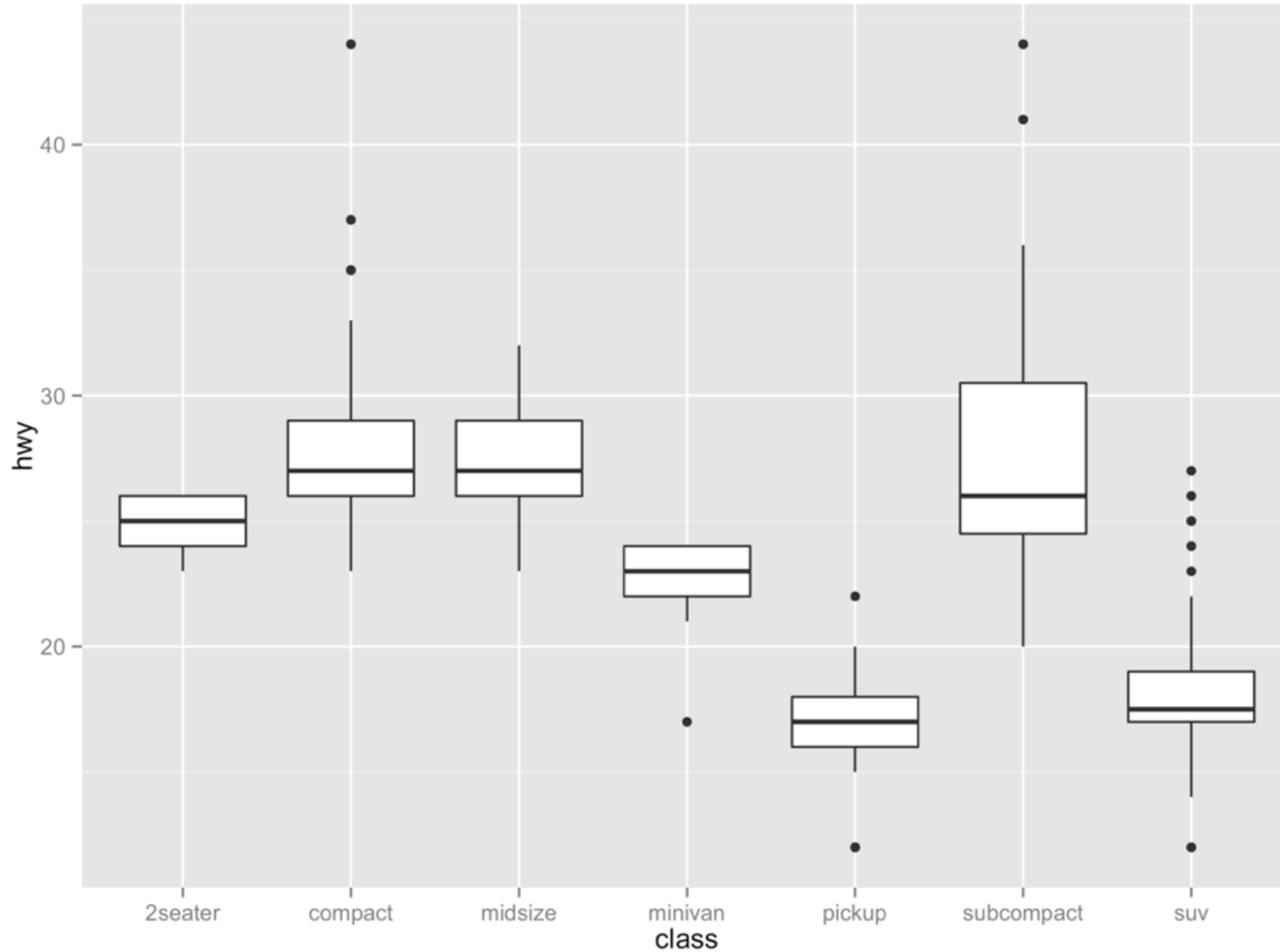
# Your Turn 3

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

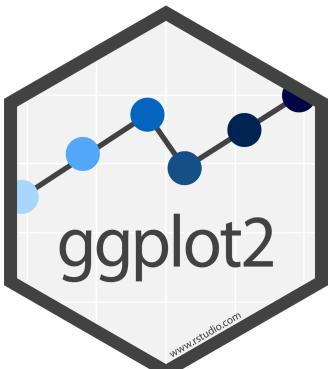


```
ggplot(mpg) + geom_point(aes(class, hwy))
```

02 : 00

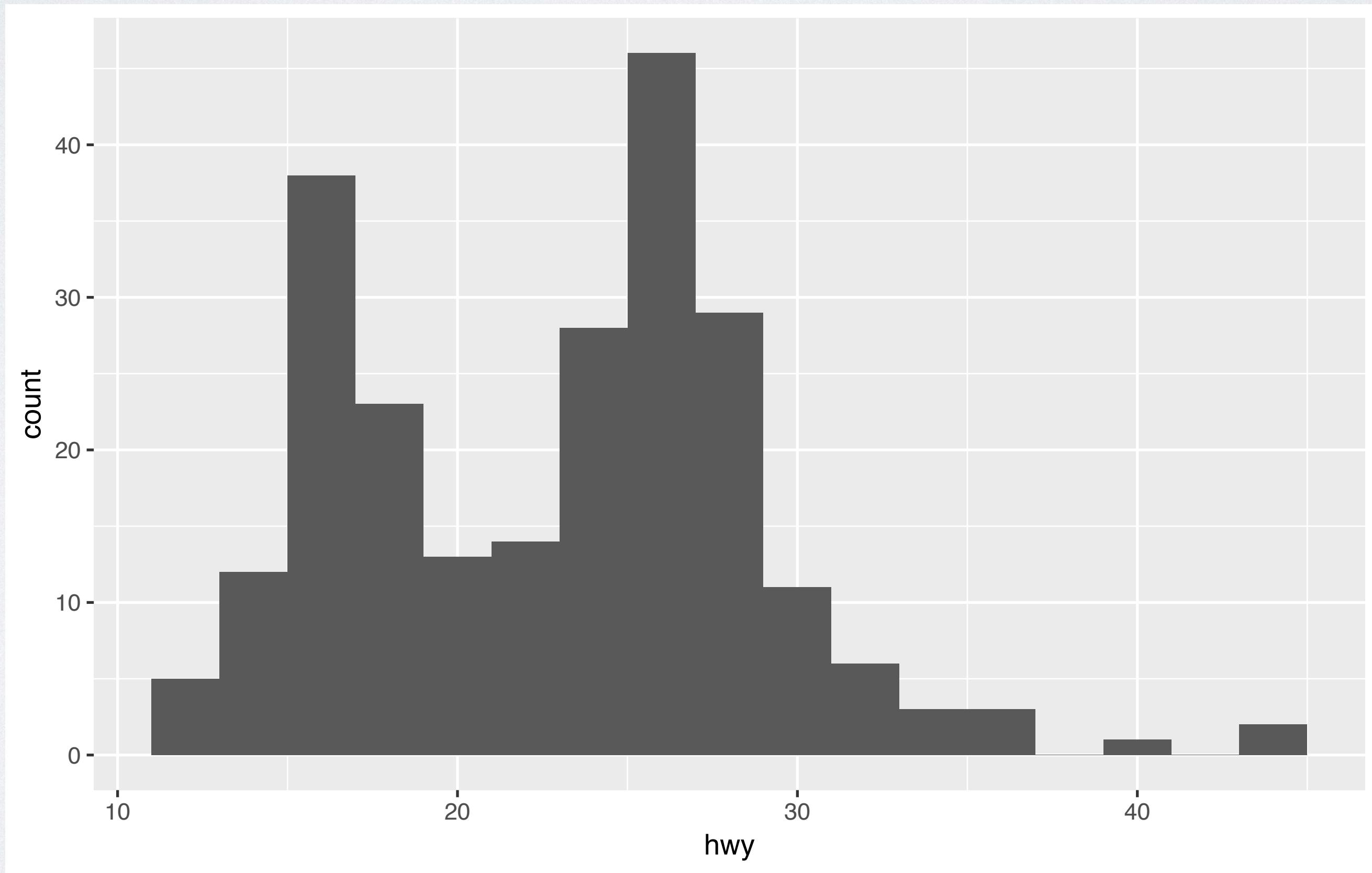


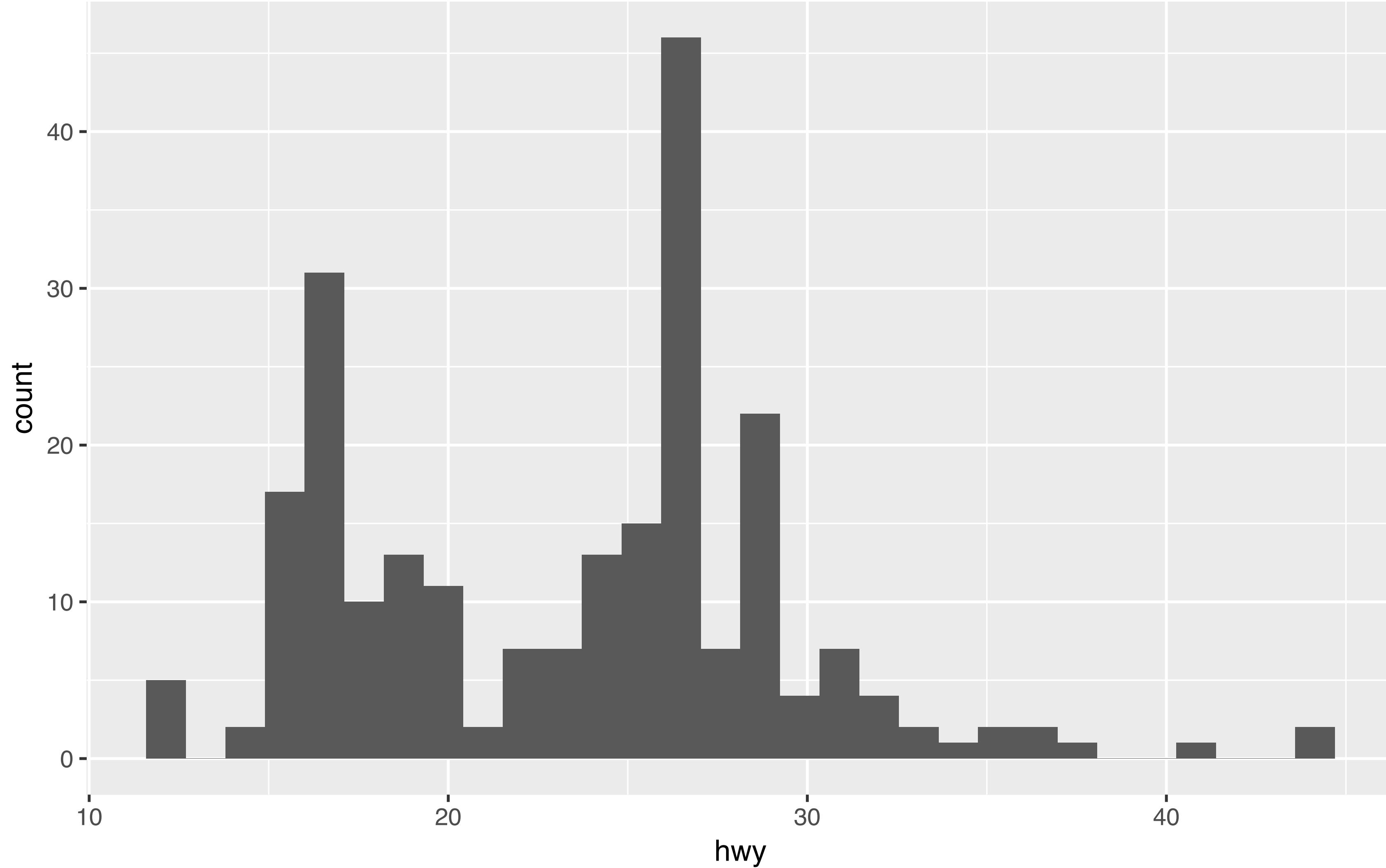
```
ggplot(data = mpg) +
 geom_boxplot(mapping = aes(x = class, y = hwy))
```



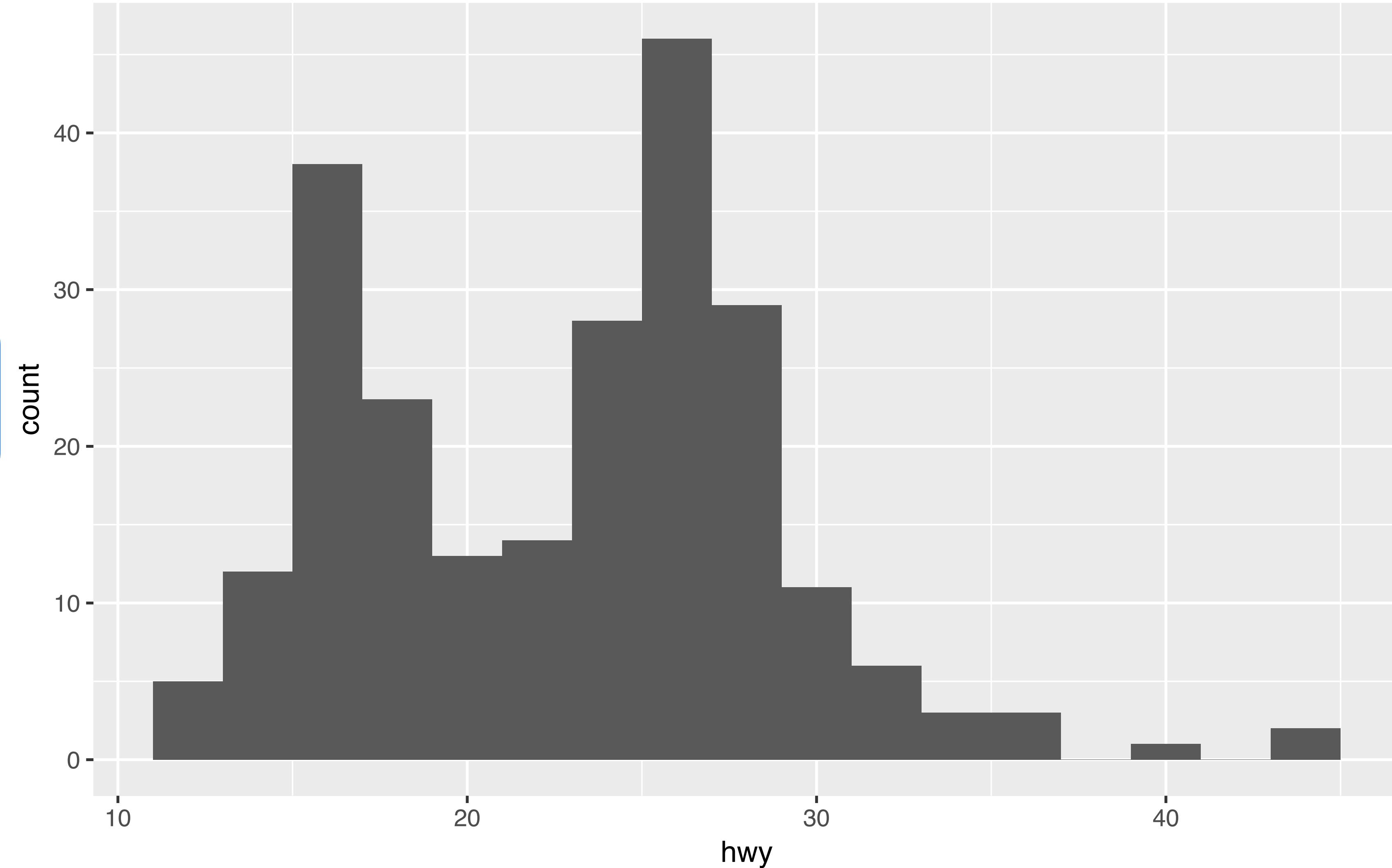
# Your Turn 4

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

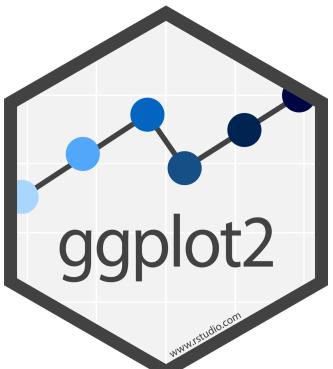




```
ggplot(data = mpg) +
 geom_histogram(mapping = aes(x = hwy))
```



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

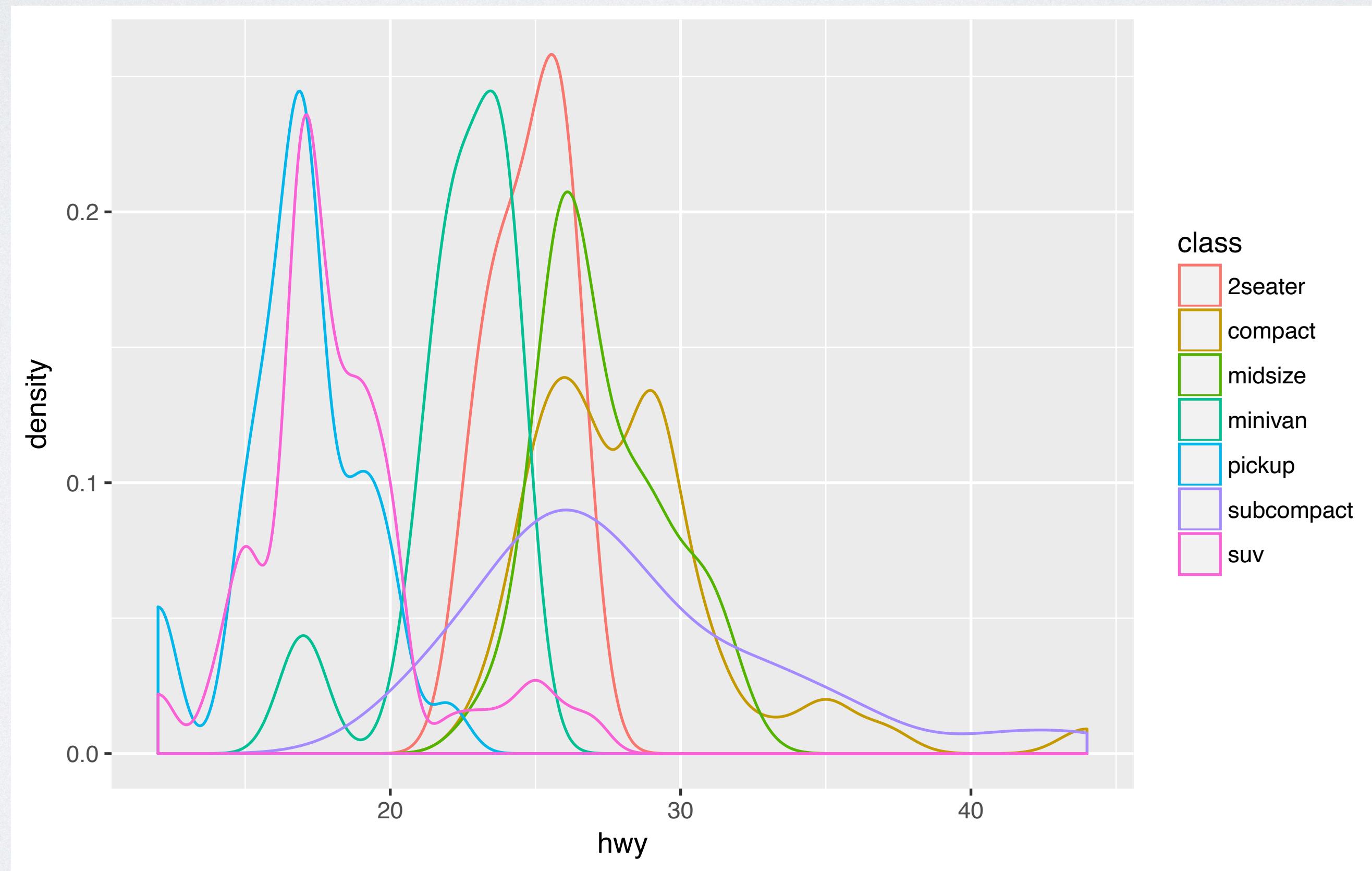


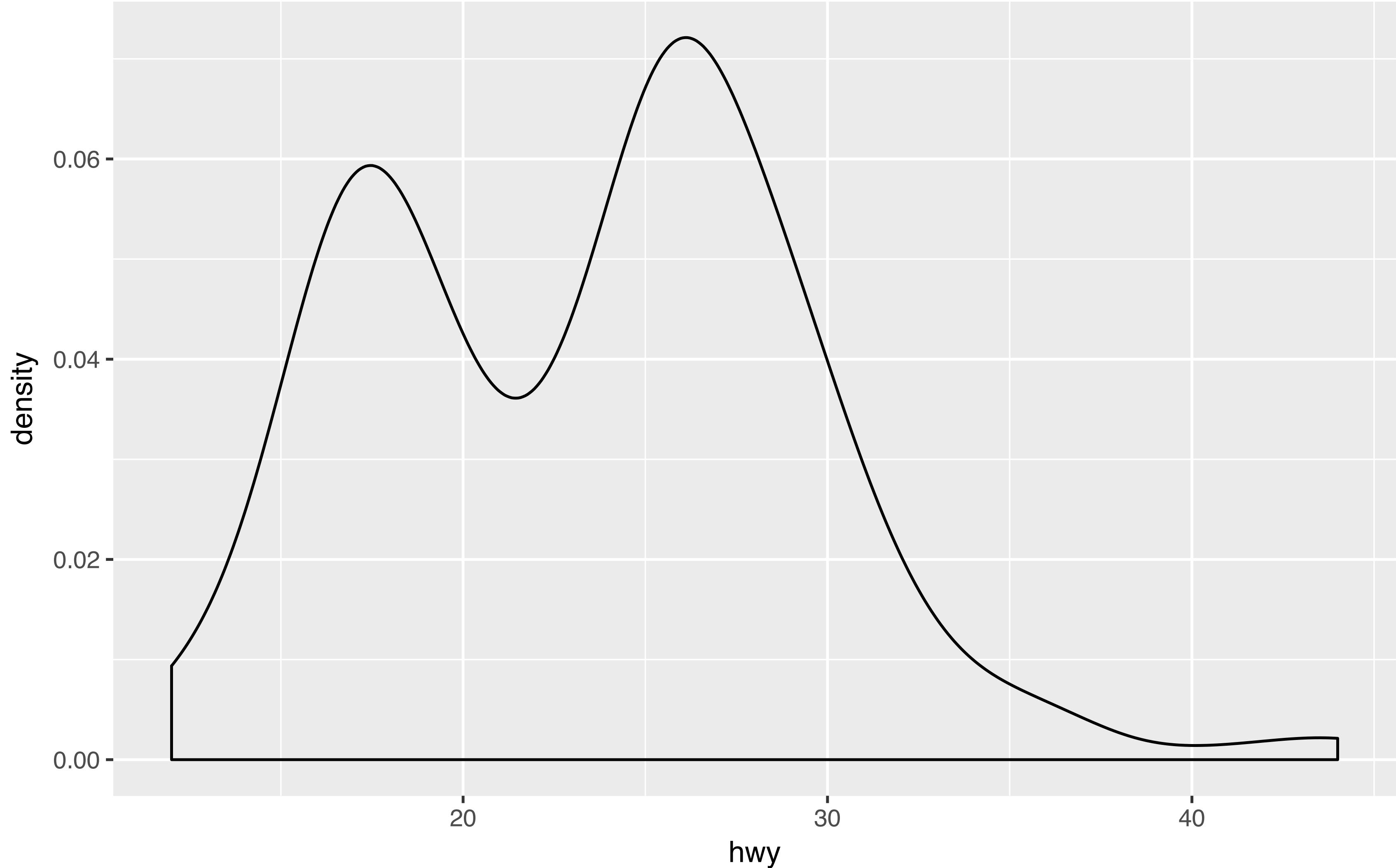
# Part 2



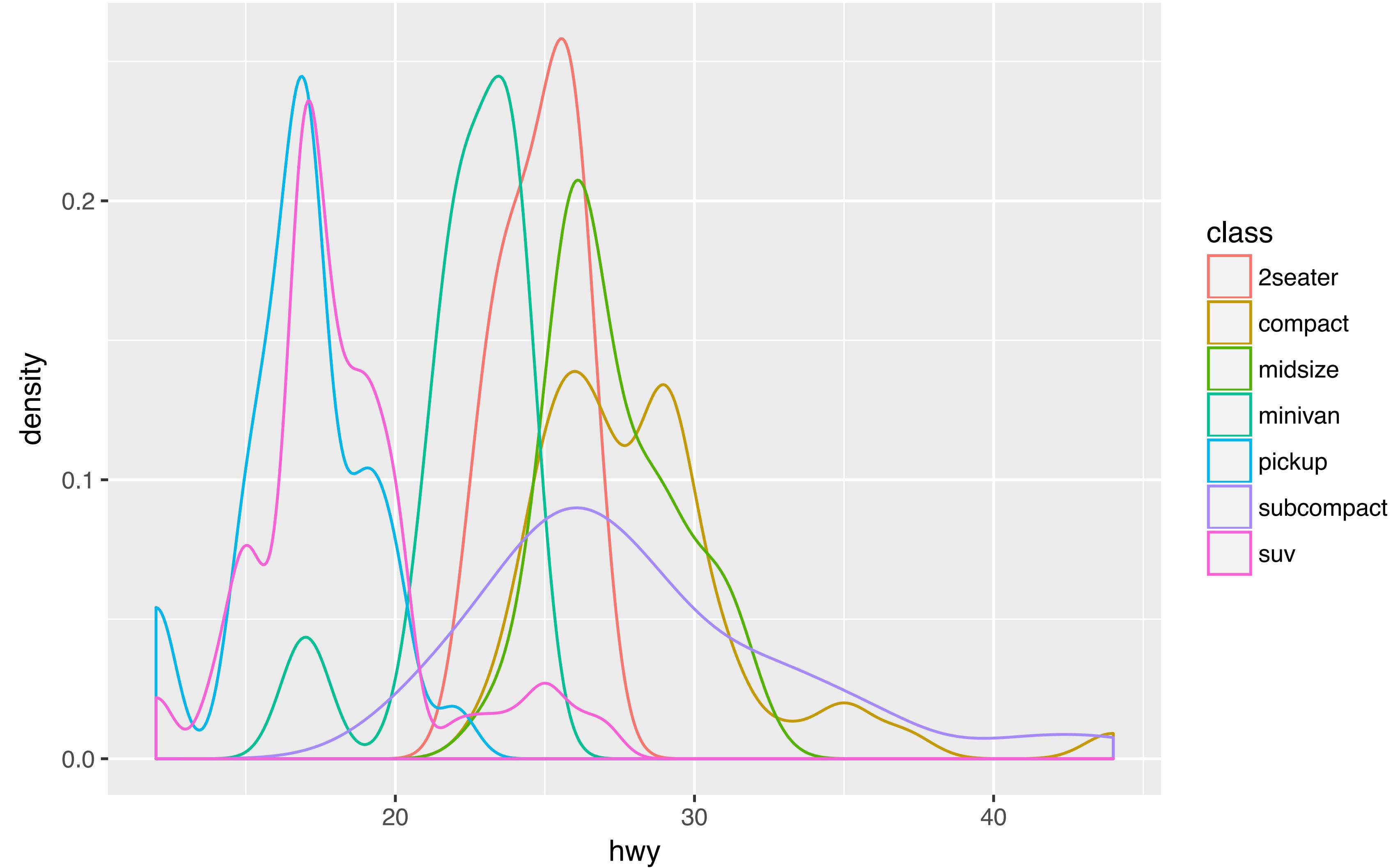
# Your Turn 5

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

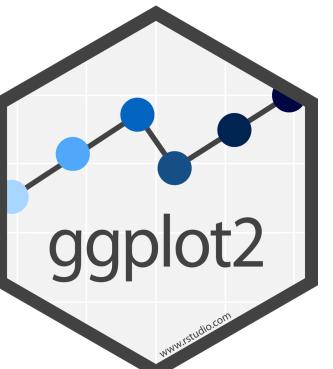


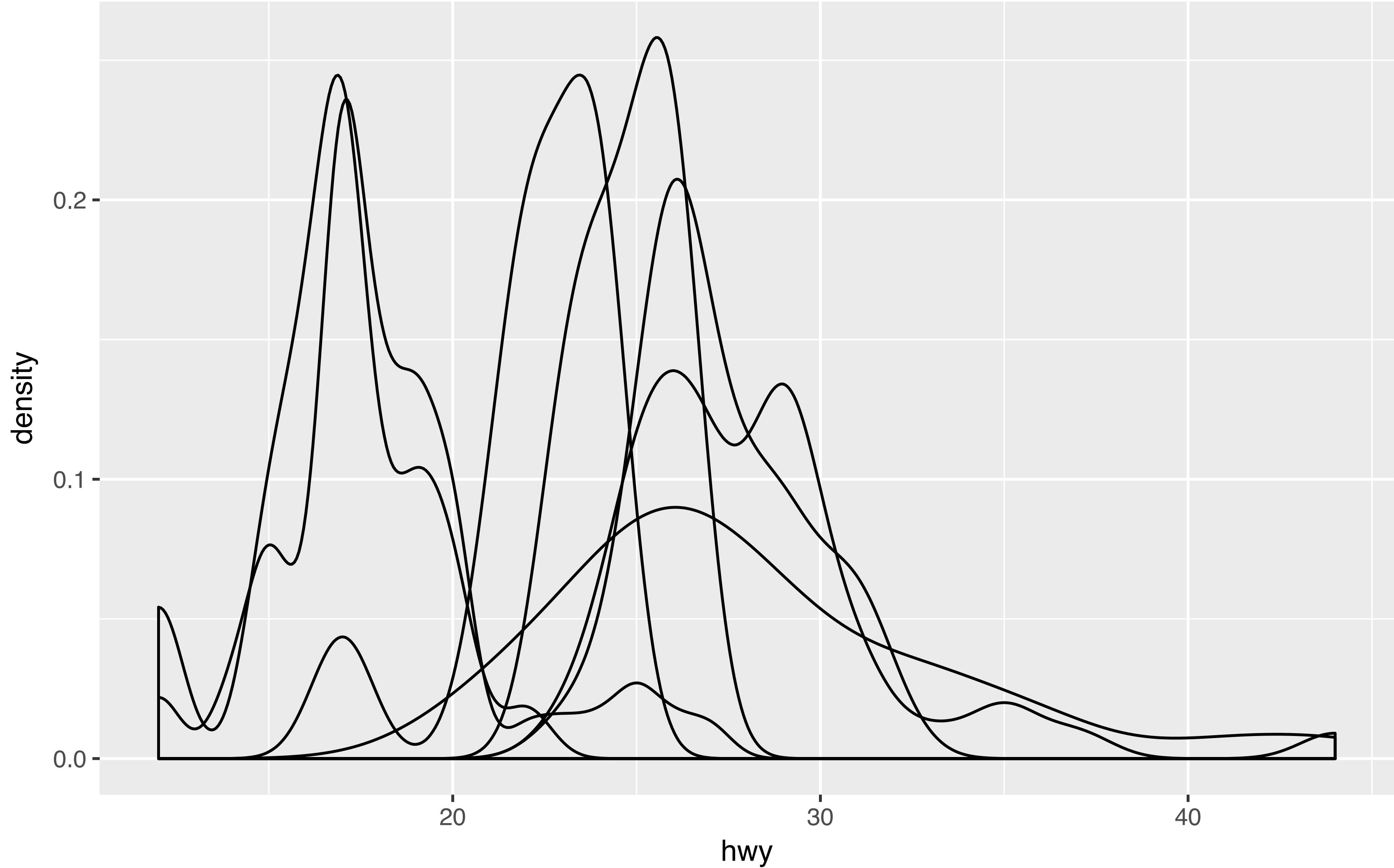


```
ggplot(data = mpg) +
 geom_density(mapping = aes(x = hwy))
```

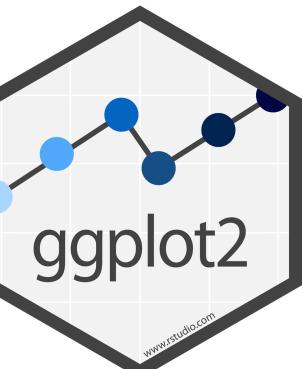


```
ggplot(data = mpg) +
 geom_density(mapping = aes(x = hwy, color = class))
```



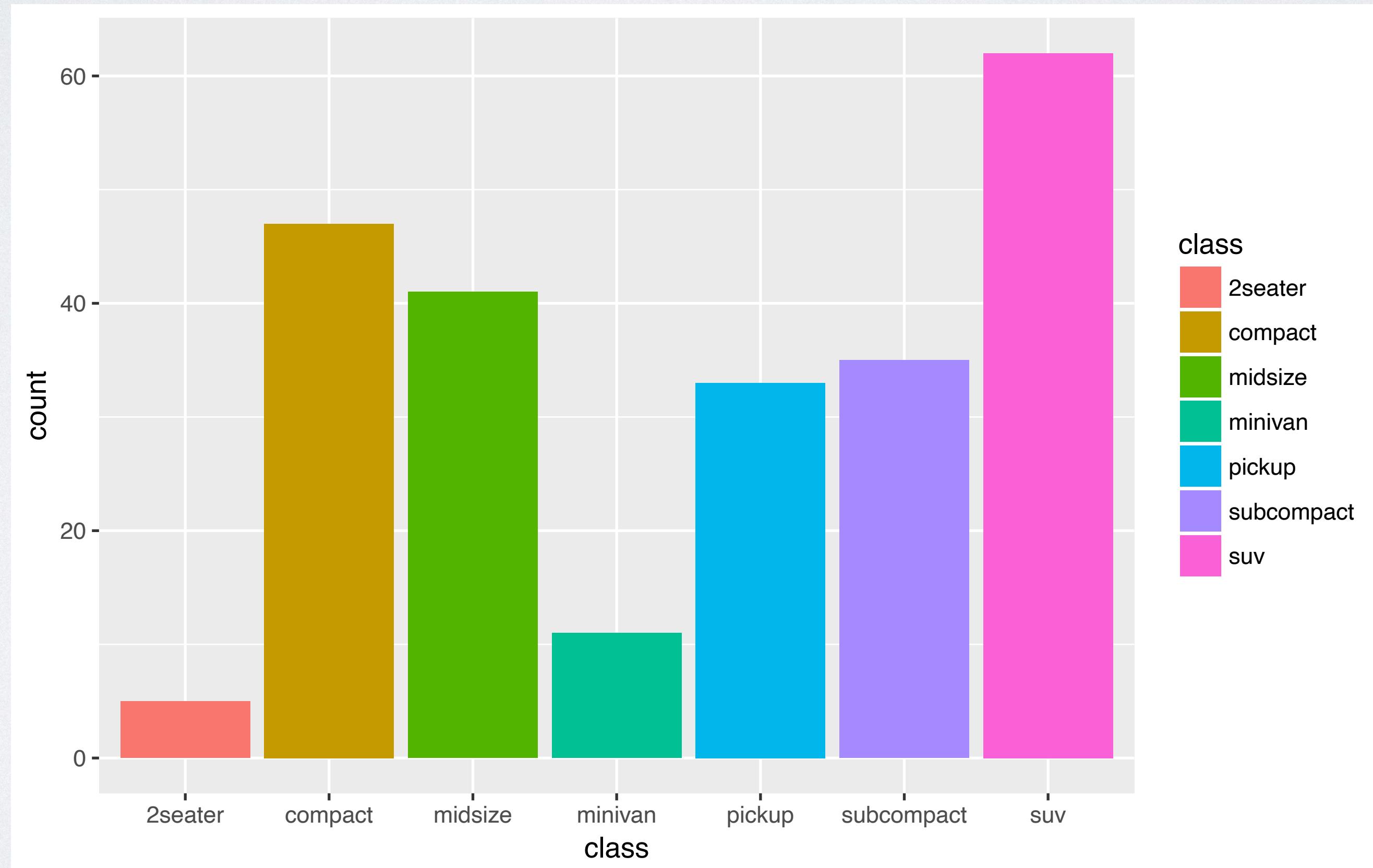


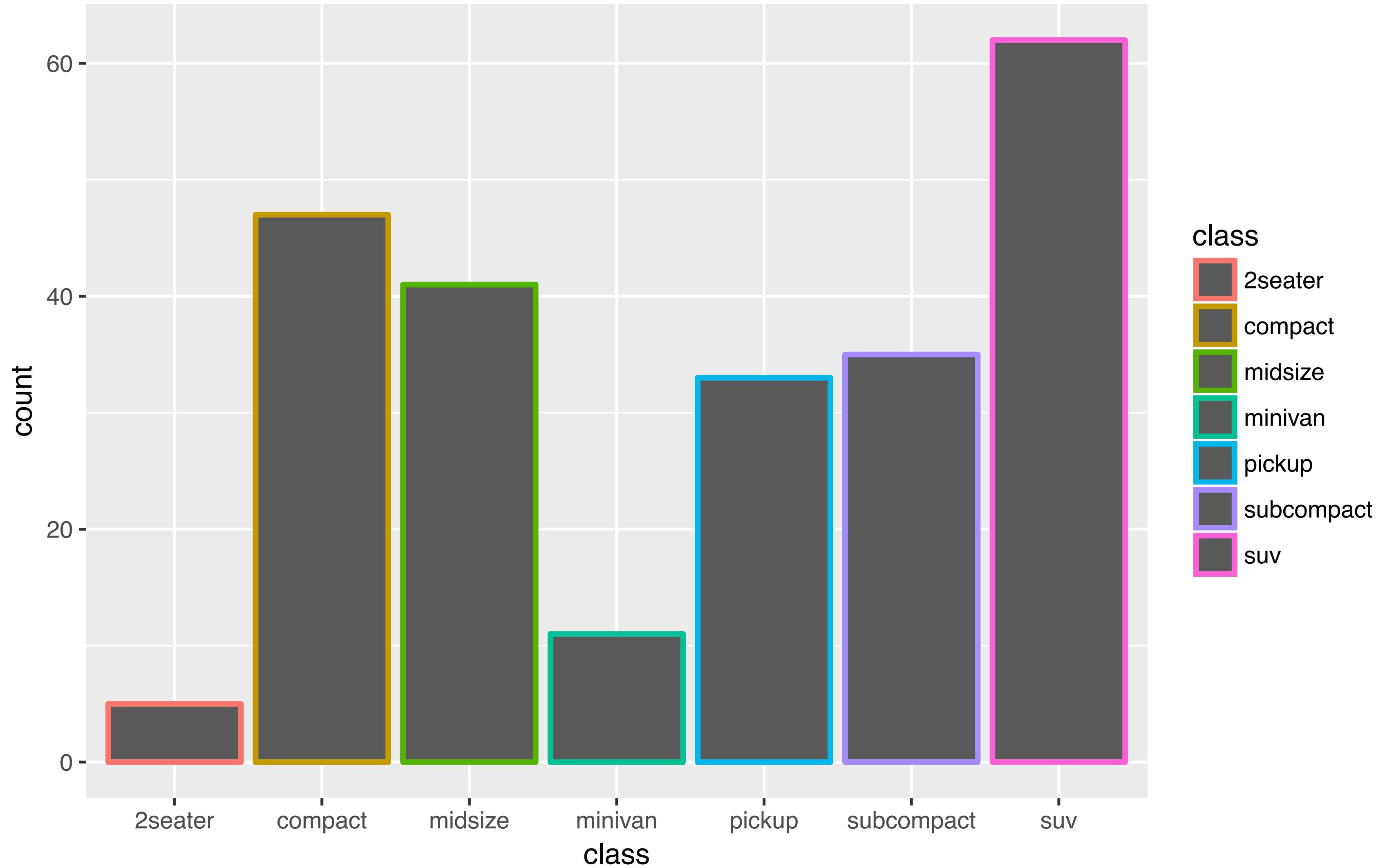
```
ggplot(data = mpg) +
 geom_density(mapping = aes(x = hwy, group = class))
```



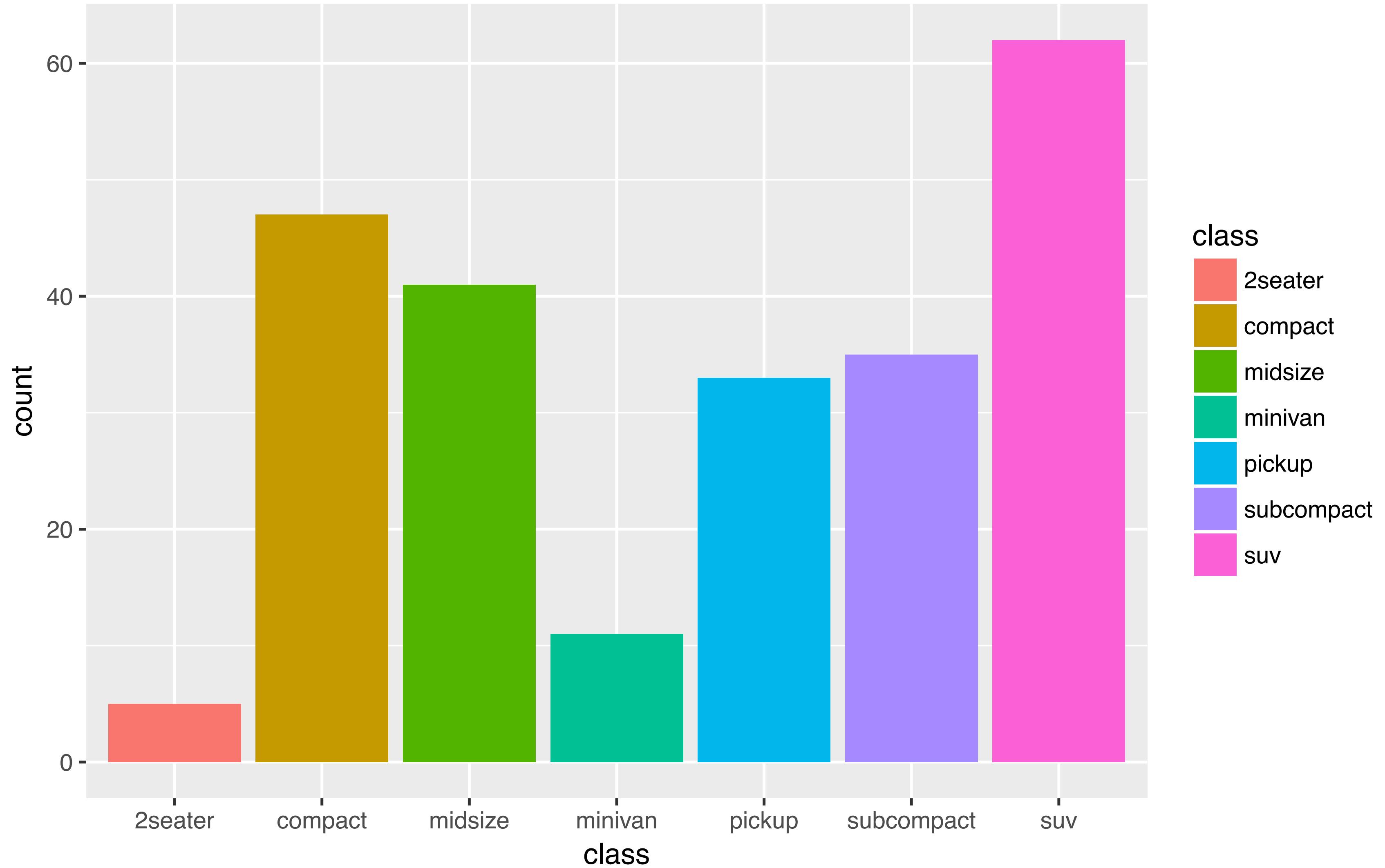
# Your Turn 6

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



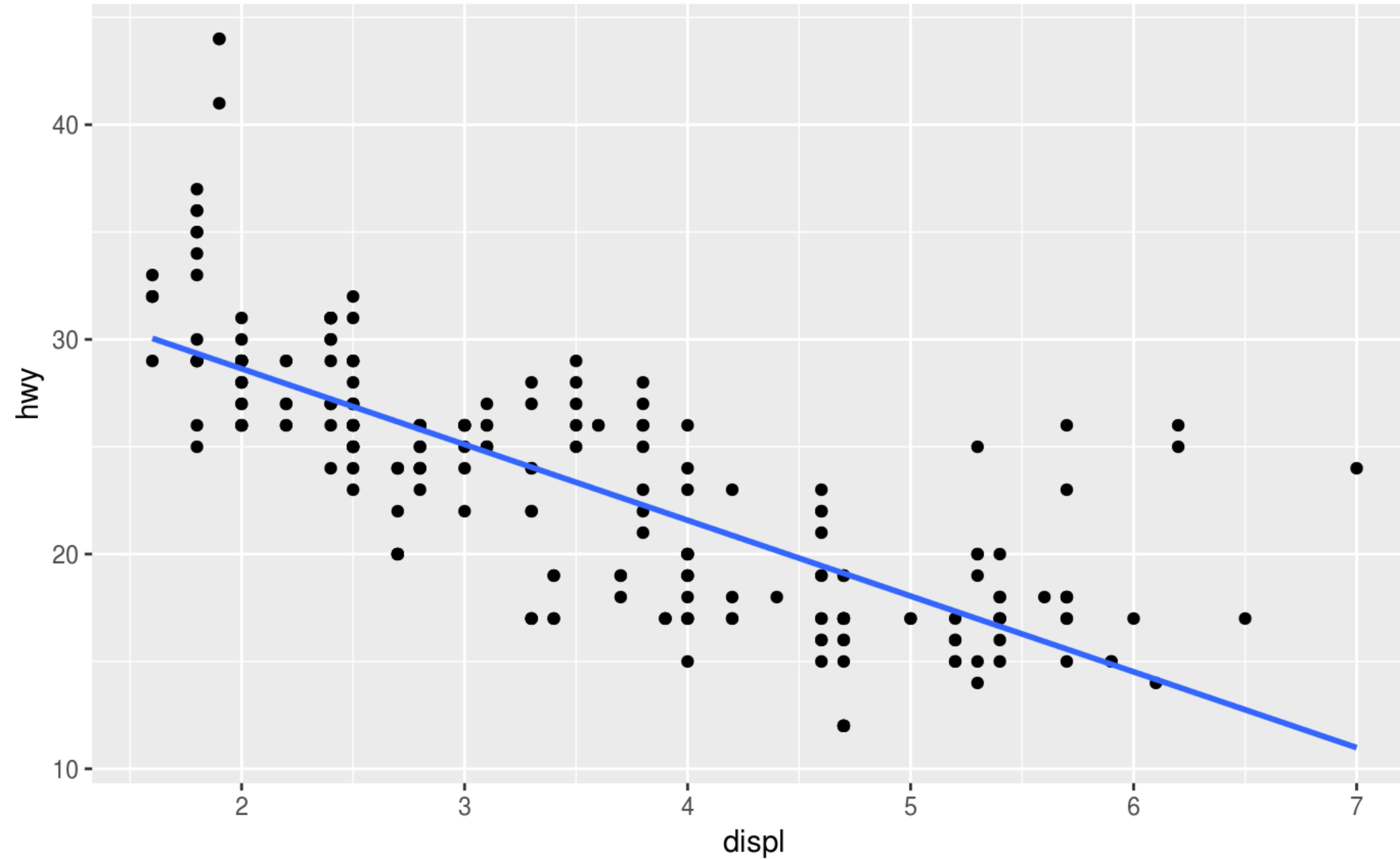


```
ggplot(data = mpg) +
 geom_bar(mapping = aes(x = class, color = class))
```



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

# BUT HOW DO WE MAKE A COMPLEX PRESENTATION?



# Your Turn 7

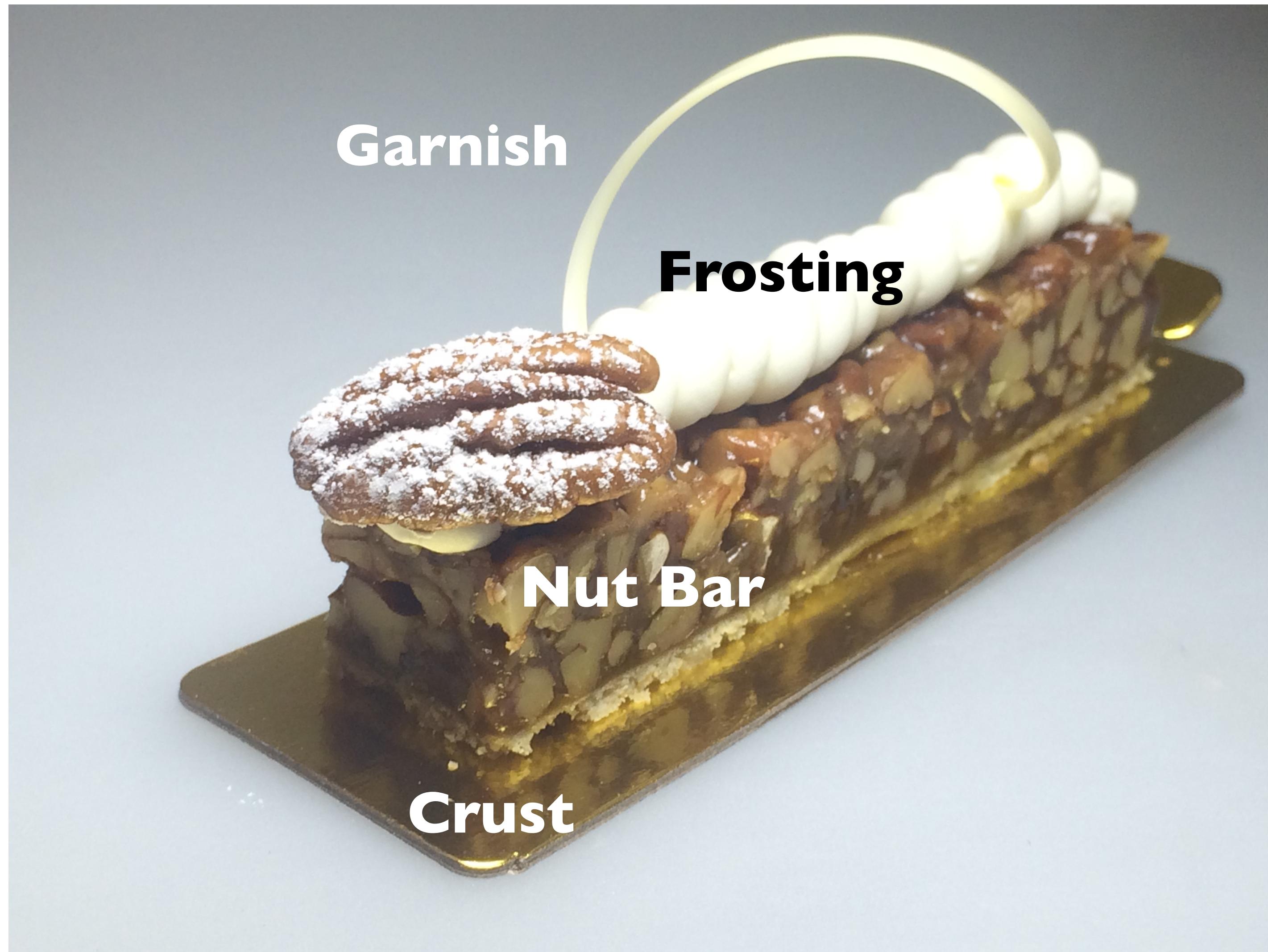
With your partner, predict what this code will do.

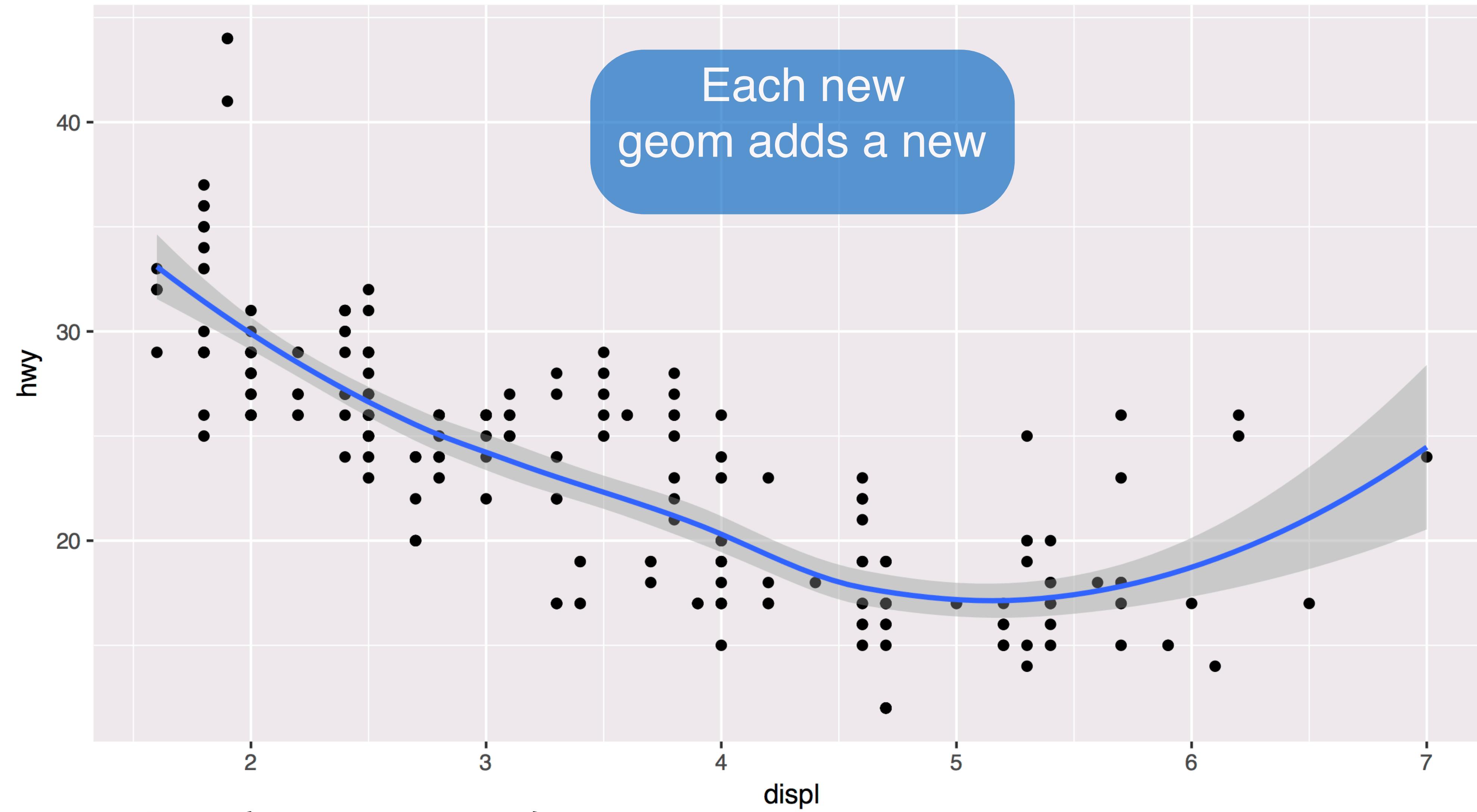
Then run it.

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

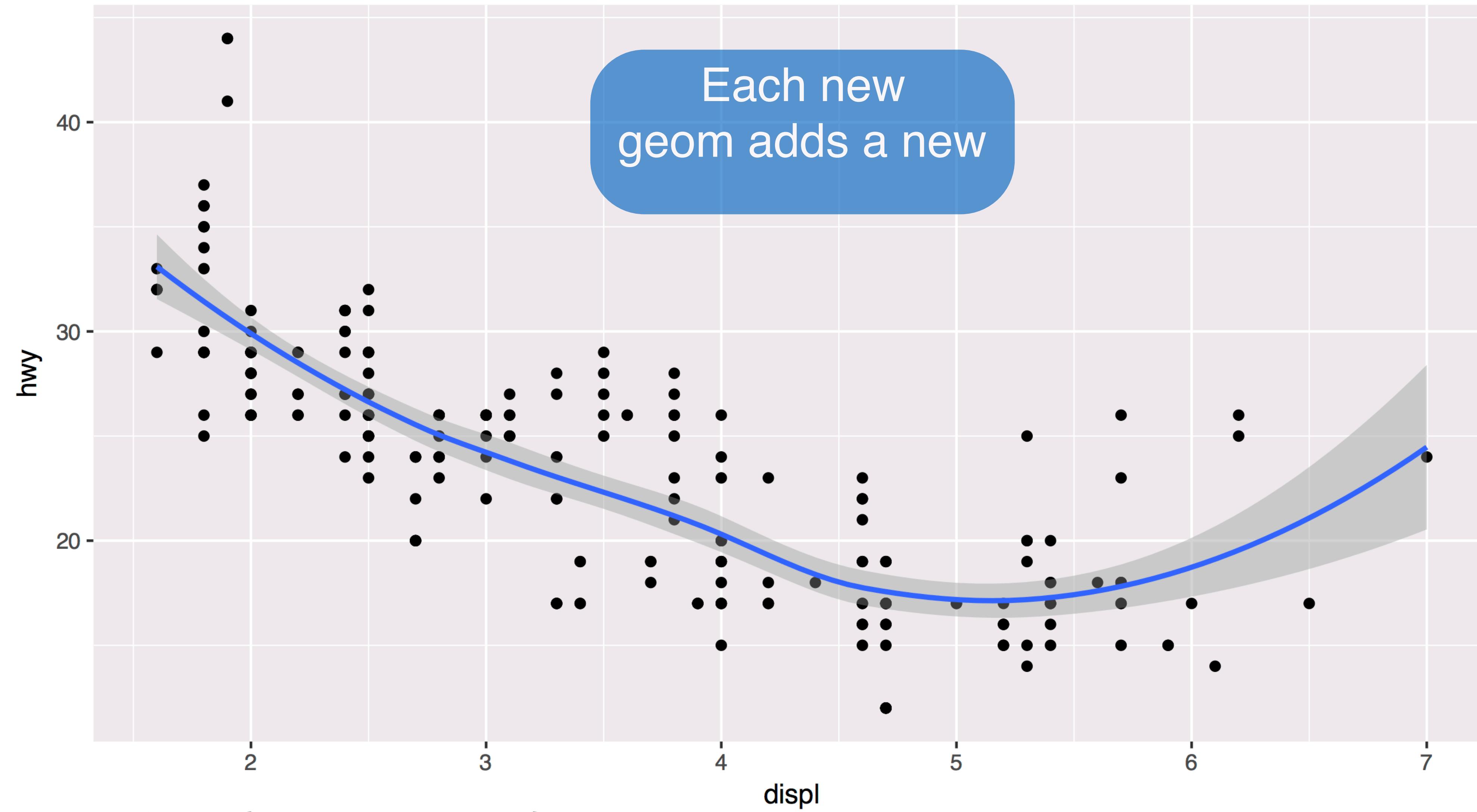


# THE SAME WAY YOU MAKE THIS: IN LAYERS





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



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

# YOU CAN CHOOSE FROM MANY GEOMS



**ggplot2** part of the [tidyverse](#)  
3.1.0.9000

A layer combines data, aesthetic mapping, a geom (geometric object), a stat (statistical transformation), and a position adjustment. Typically, you will create layers using a `geom_` function, overriding the default position and stat if needed.

 `geom_abline()` `geom_hline()`  
`geom_vline()`

Reference lines: horizontal, vertical, and diagonal

 `geom_bar()` `geom_col()`  
`stat_count()`

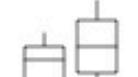
Bar charts

 `geom_bin2d()` `stat_bin_2d()`

Heatmap of 2d bin counts

 `geom_blank()`

Draw nothing

 `geom_boxplot()` `stat_boxplot()`

A box and whiskers plot (in the style of Tukey)

 `geom_contour()` `stat_contour()`

2d contours of a 3d surface

 `geom_count()` `stat_sum()`

Count overlapping points

 `geom_density()` `stat_density()`

Smoothed density estimates

# LEARN MORE IN THE RSTUDIO.CLOUD

## PRIMERS

Learn Guide What's New Primers DataCamp Courses Cheat Sheets

Carl Howe

### R Studio Primers

Learn data science basics with the interactive tutorials below.

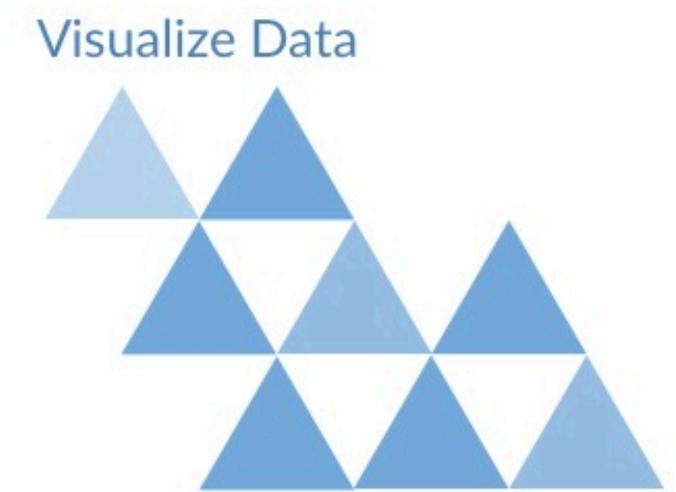
#### The Basics



Start here to learn the skills that you will rely on in every analysis (and every primer that follows): how to inspect, visualize, subset, and transform your data, as well as how to run code.



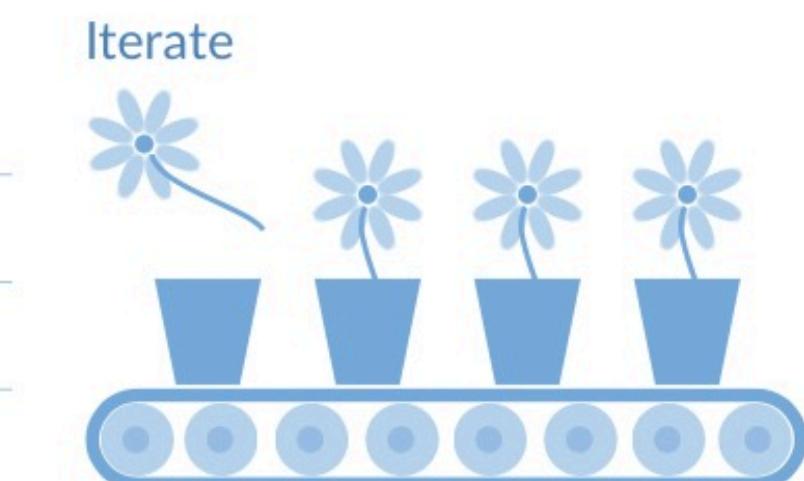
Learn the most important data handling skills in R: how to extract values from a table, subset tables, calculate summary statistics, and derive new variables.



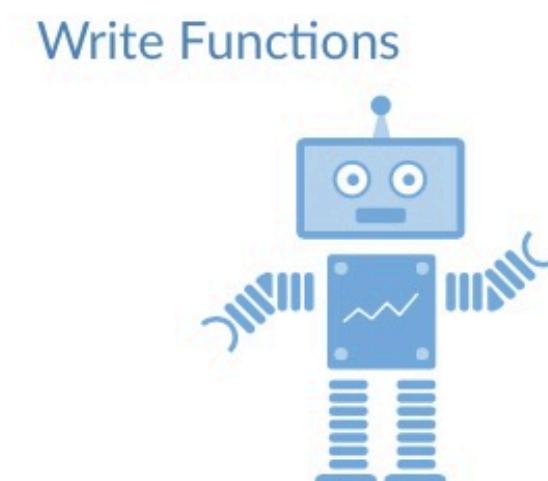
Learn how to use ggplot2 to make any type of plot with your data. Then learn the best ways to visualize patterns within values and relationships between variables.



Unlock the tidyverse by learning how to make and use tidy data, the data format designed for R.



Master a core programming paradigm with the purrr package: for each \_\_\_ do \_\_\_.



Functions are the key to programming in R. This primer will teach you how to write and use your own reusable functions.

# LEARN MORE IN THE RSTUDIO.CLOUD PRIMERS

The screenshot shows a navigation bar with tabs: Learn, Guide, What's New, Primers (which is highlighted in blue), and Documentation. Below the navigation bar, a secondary menu titled "The Basics" is displayed. The main content area features a large title "Data Visualization Basics" and a "Welcome" section. A list of topics follows: Welcome, A code template, Aesthetic mappings, Geometric objects, and The ggplot2 package. The "Welcome" topic is currently selected, indicated by a gray background.

☰ Learn      Guide      What's New      Primers      Documentation

☰ The Basics

## Data Visualization Basics

### Welcome

Data visualization is a process of creating visual representations of data to facilitate better understanding. It can help you visualize data, your analysis, and your results. One of the most common frustrations of learning data visualization is that there is no one right way to do it. This tutorial will teach you how to create effective visualizations for visualizing data. The skills you learn here will quickly become useful in your work.

This tutorial will teach you how to create effective visualizations for visualizing data. The skills you learn here will quickly become useful in your work.

- 1. How to create effective visualizations
- 2. How to add variables to your visualizations
- 3. How to select the right type of visualization for your data

These skills will quickly become useful in your work.

[Data primer.](#)

## Data Visualization Basics

Welcome

A code template

Aesthetic mappings

Geometric objects

The ggplot2 package

Start Over

### Welcome

Data visualization is a process of creating visual representations of data to facilitate better understanding. It can help you visualize data, your analysis, and your results. One of the most common frustrations of learning data visualization is that there is no one right way to do it.

This tutorial will teach you how to create effective visualizations for visualizing data. The skills you learn here will quickly become useful in your work.

1. How to create effective visualizations
2. How to add variables to your visualizations
3. How to select the right type of visualization for your data

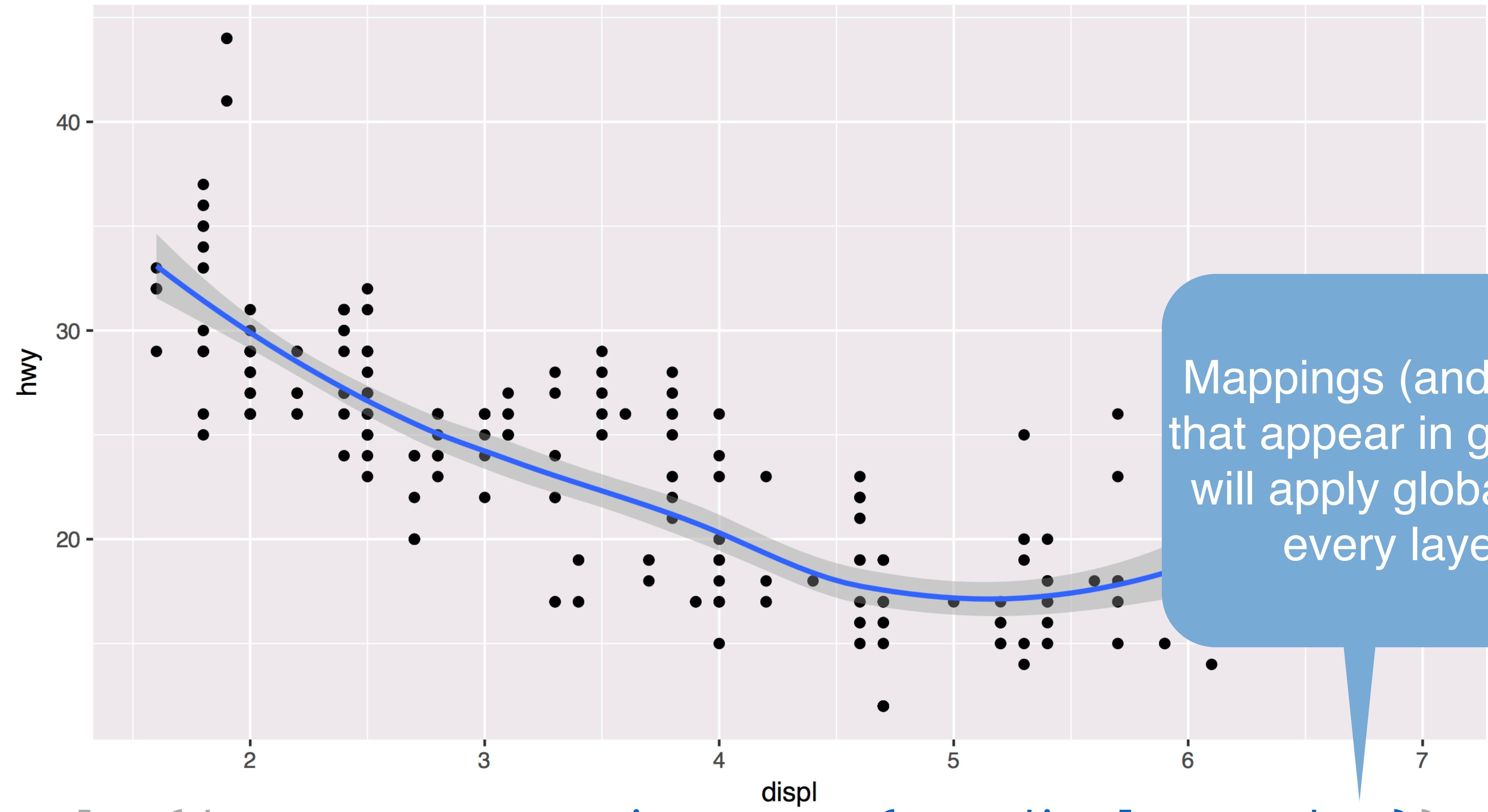
These skills will quickly become useful in your work.

[Data primer.](#)

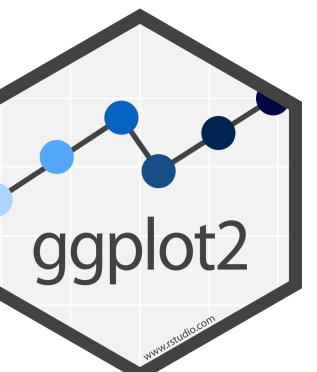
Acknowledgements

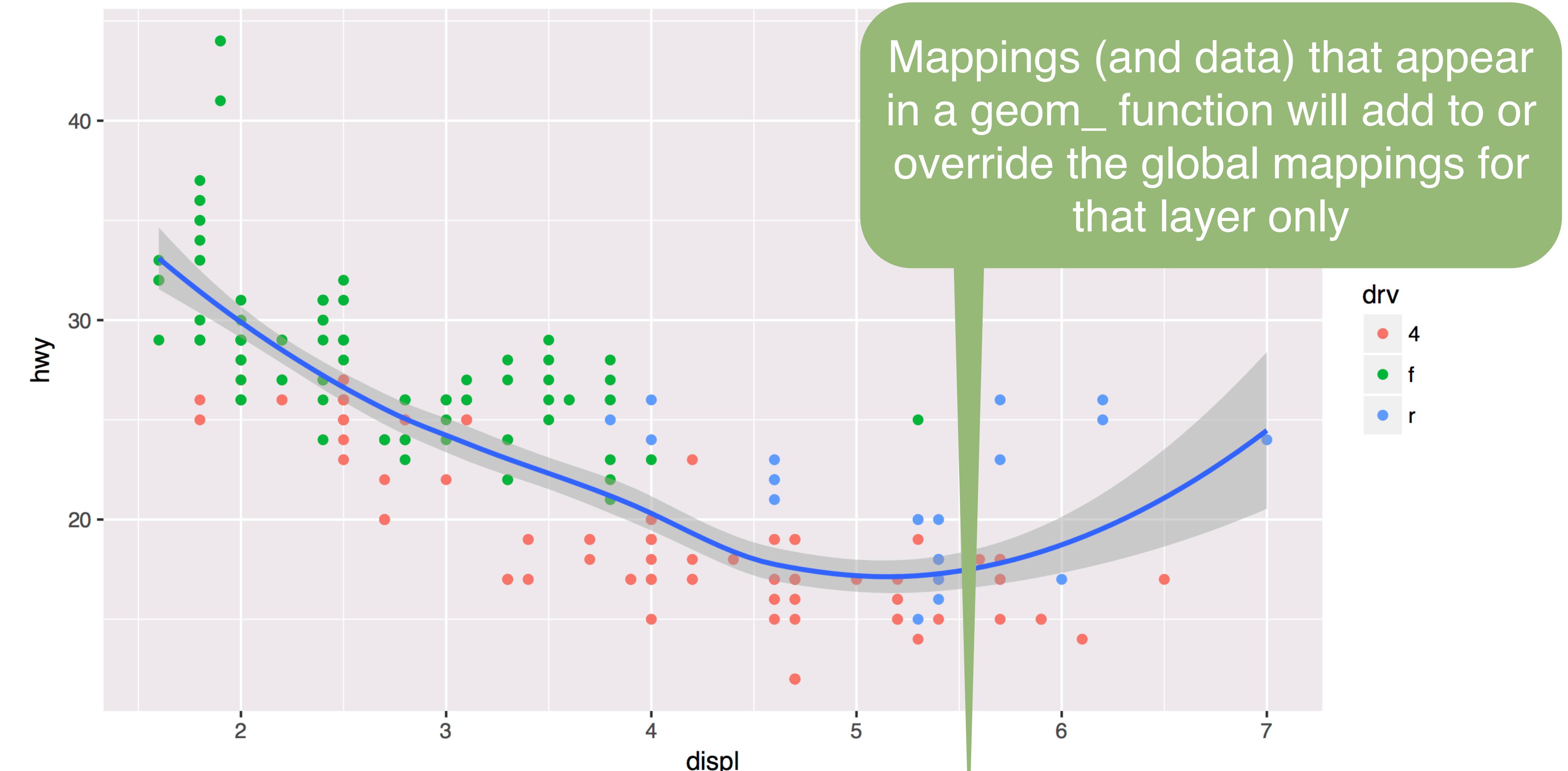


global vs. local

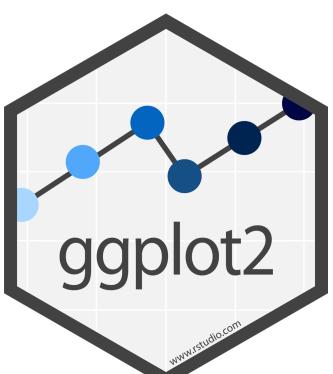


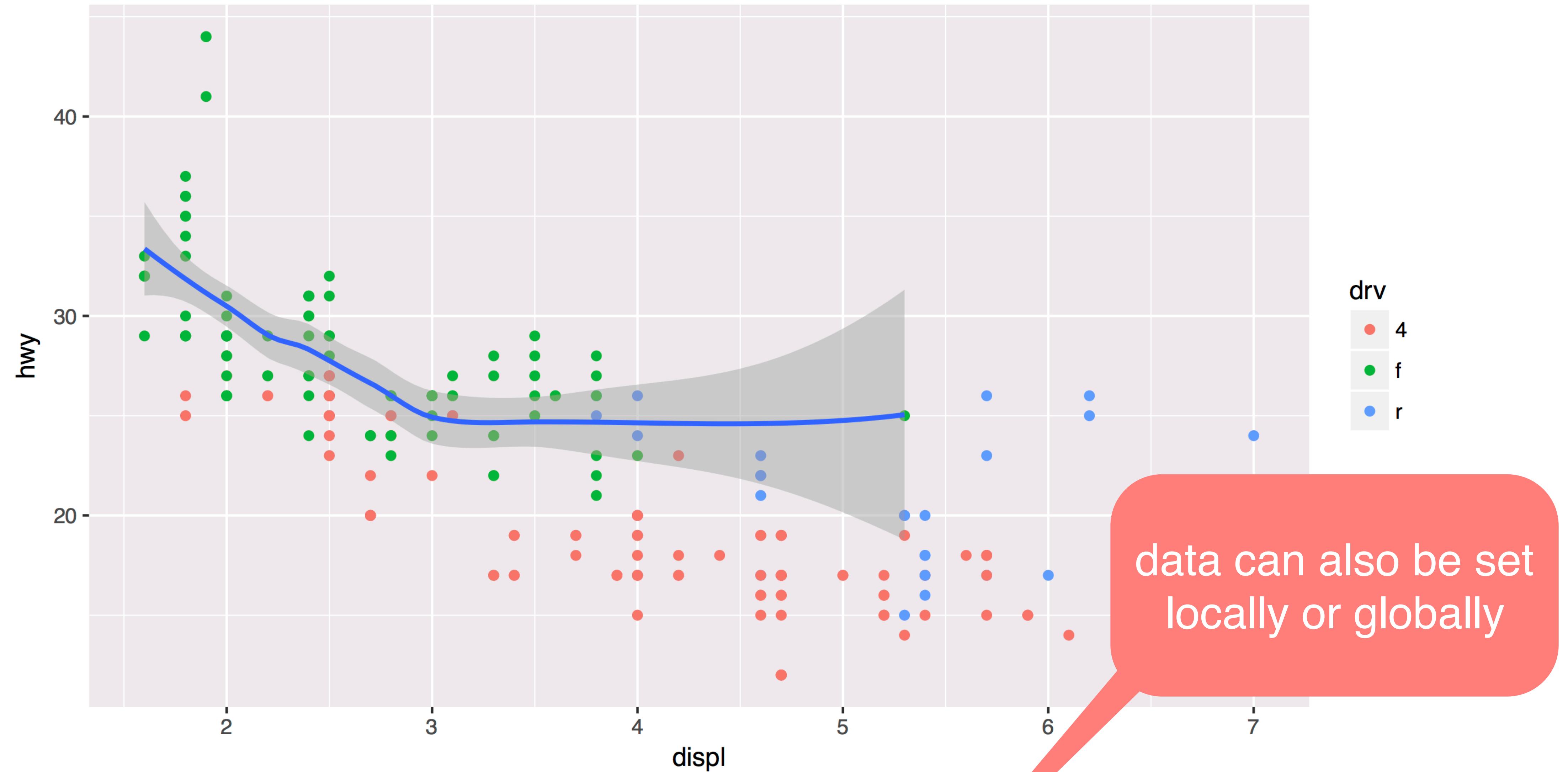
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
 geom_point() +
 geom_smooth()
```





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





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

# Saving graphs

# Your Turn 8

What does this command return?

`getwd()`

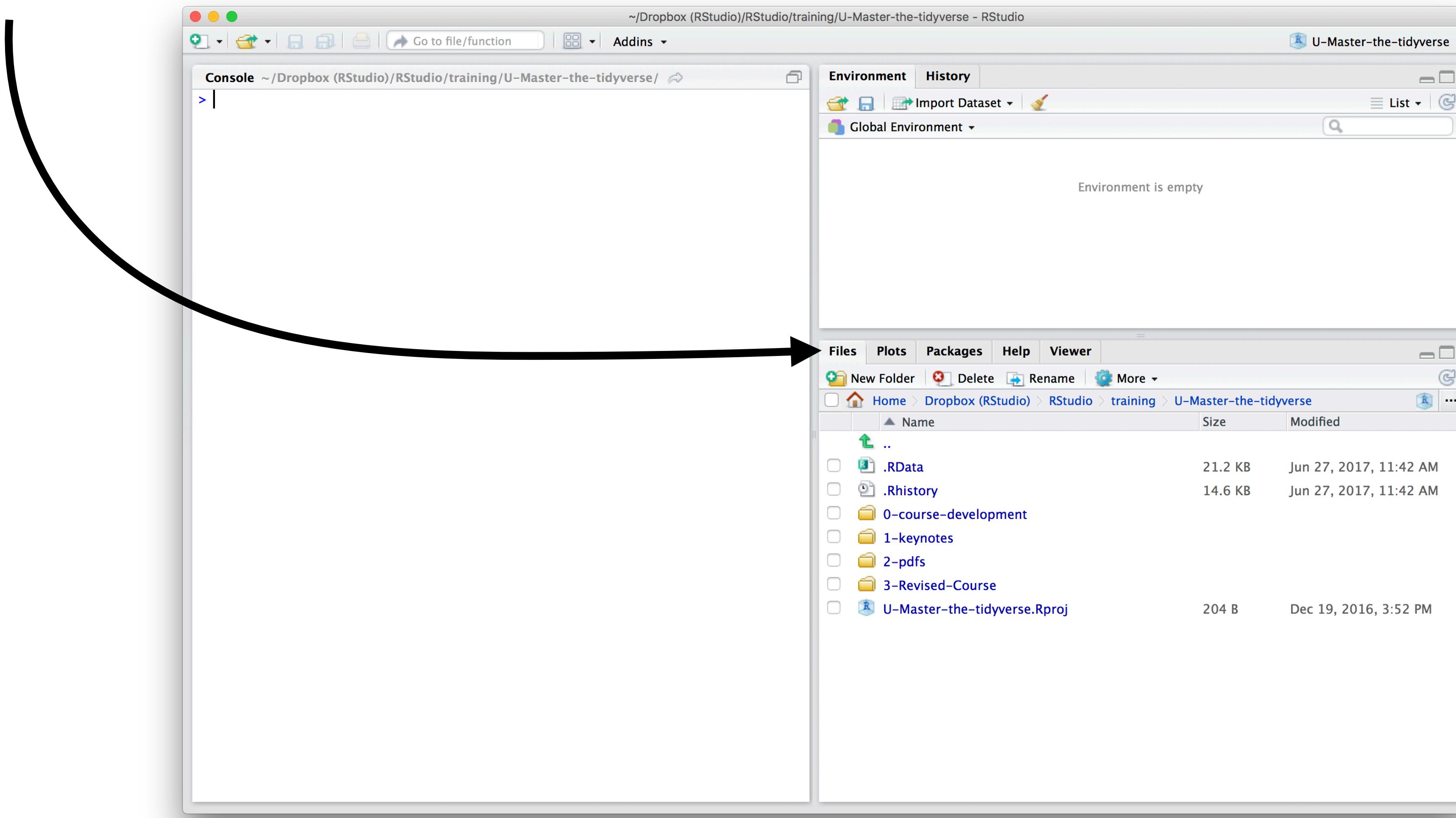


# Working directory

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

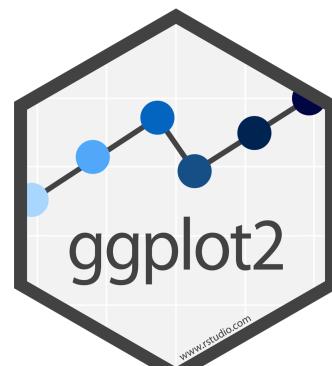
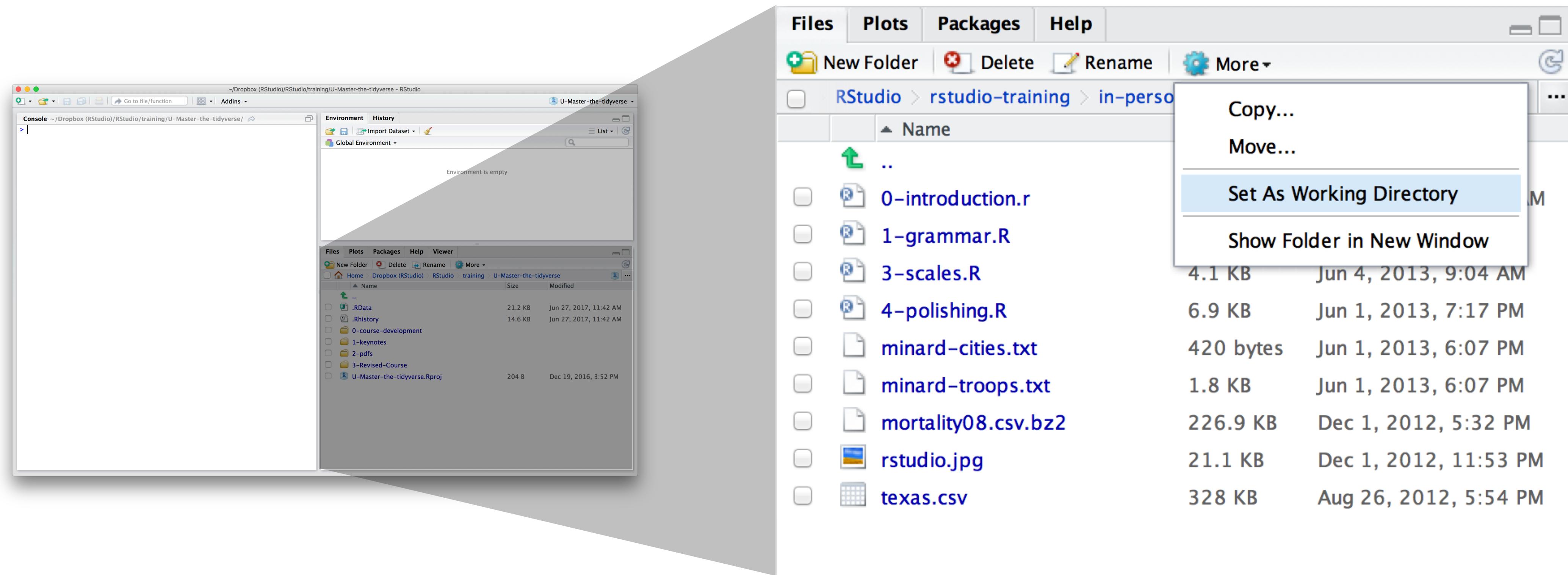
- This folder is known as your "**working directory**"
- When you save files, R will save them here
- When you load files, R will look for them here

# 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

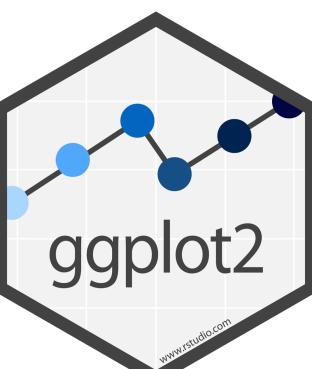
`ggsave()` saves the last plot.

Uses size on screen:

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

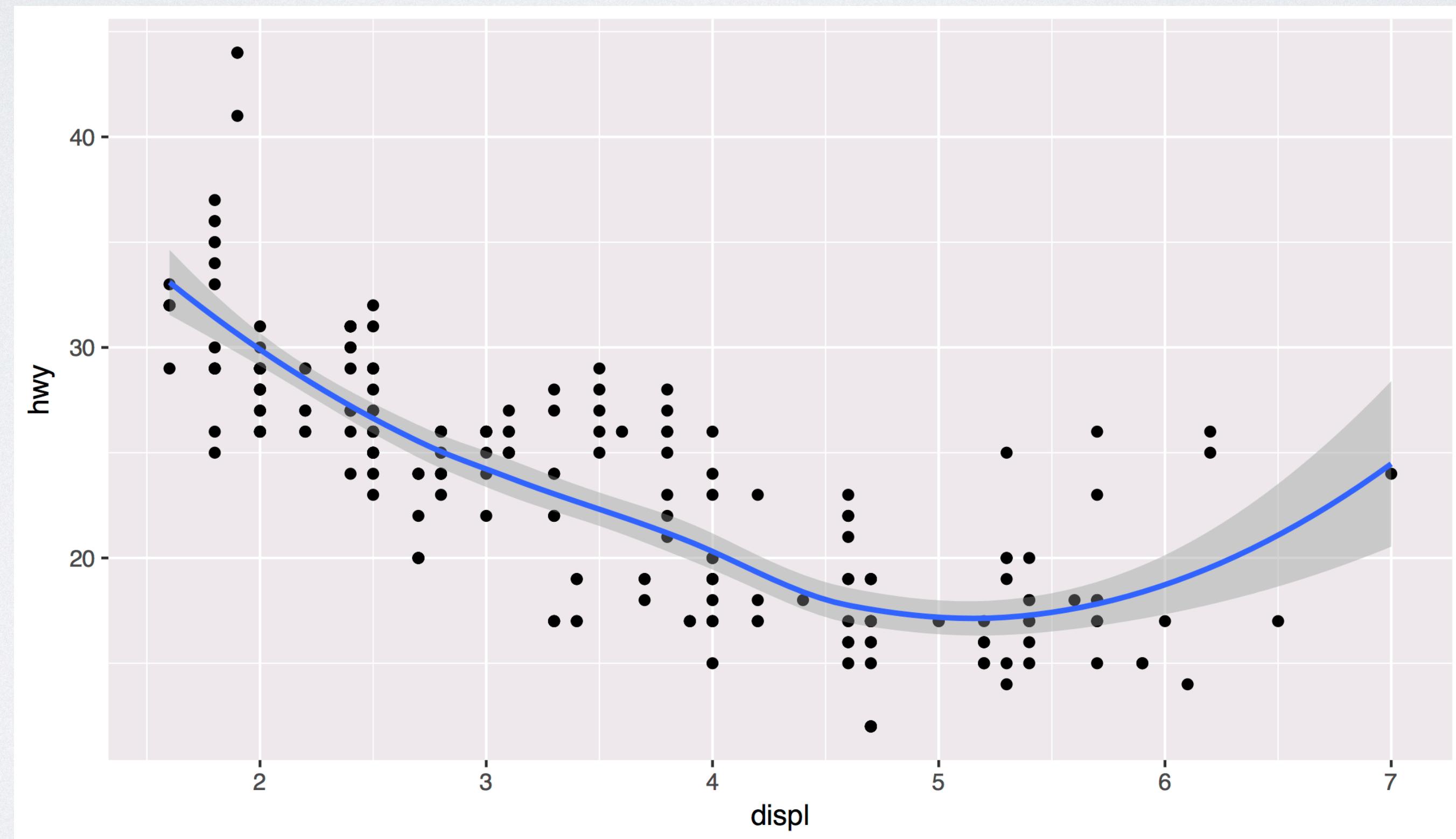
Specify size in inches

```
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).

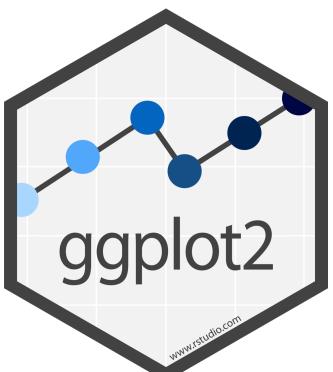


# Grammar of Graphics

# To make a graph

[template]

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



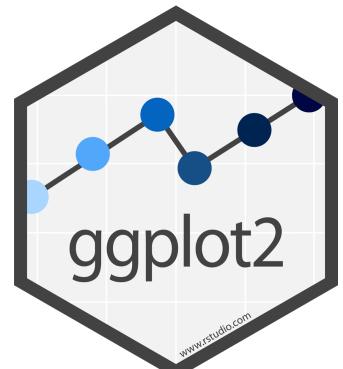
# To make a graph

| 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  |

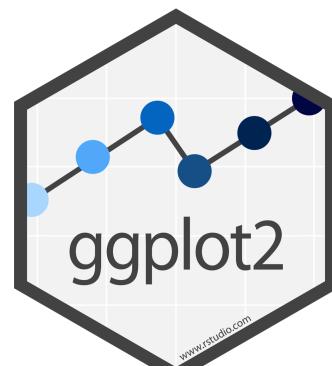
data

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

mappings

| 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

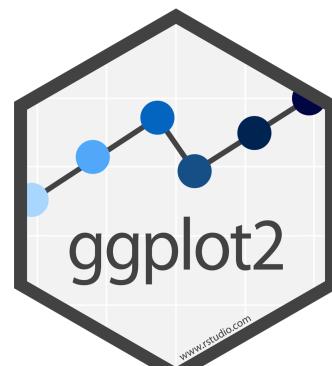
geom

1. Pick a data set

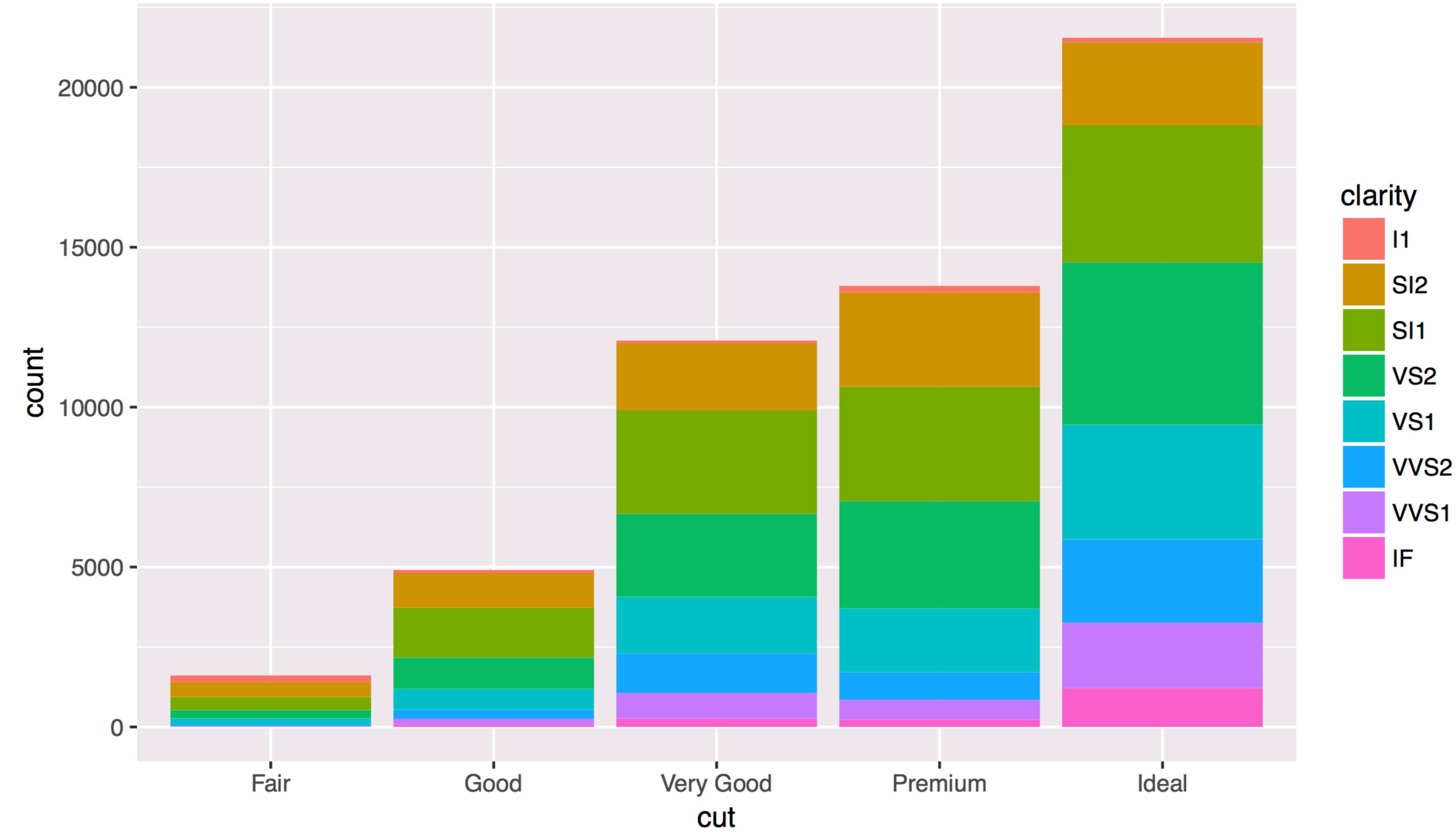
```
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a geom  
to display cases

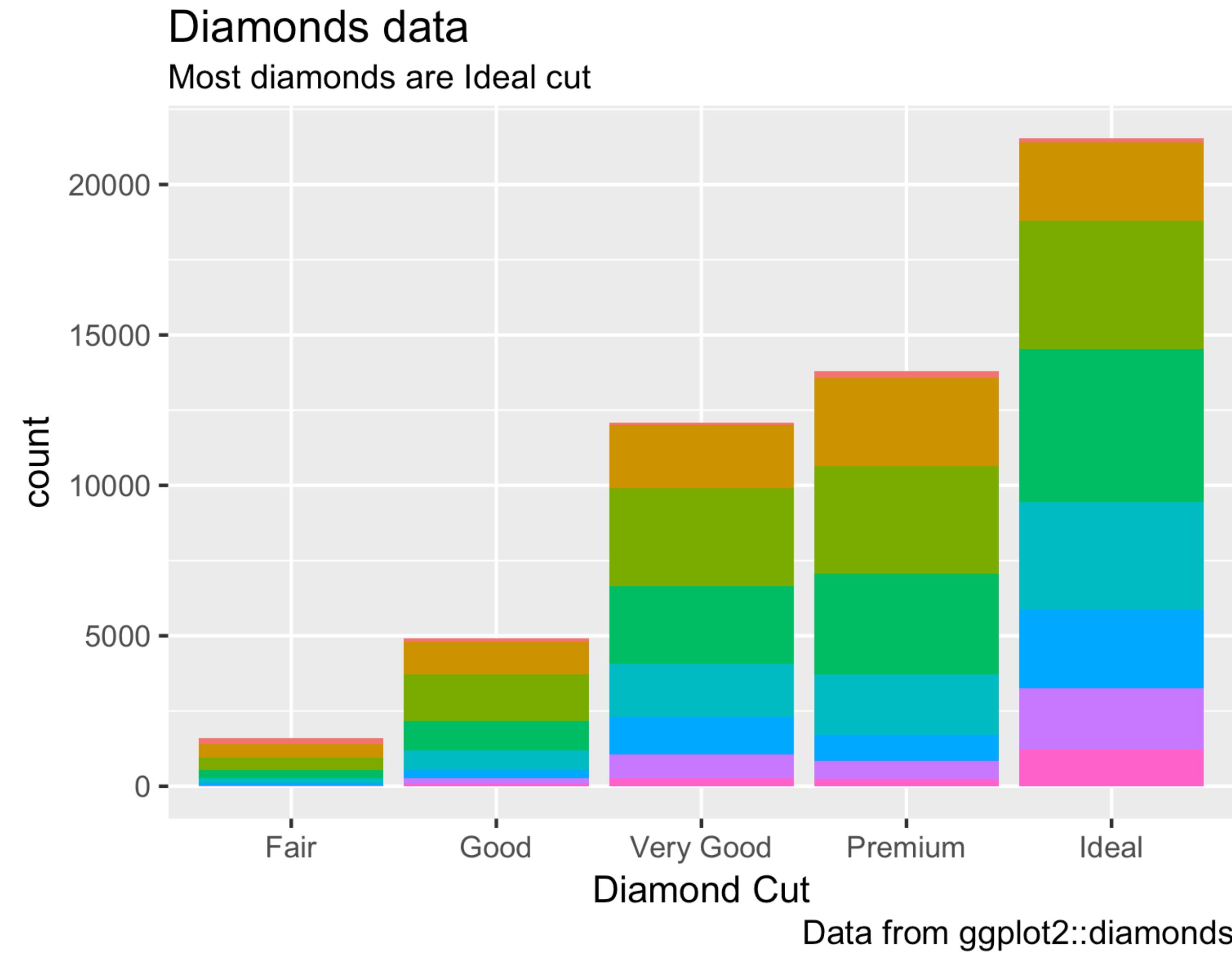
3. Map aesthetic  
properties to  
variables



What else?

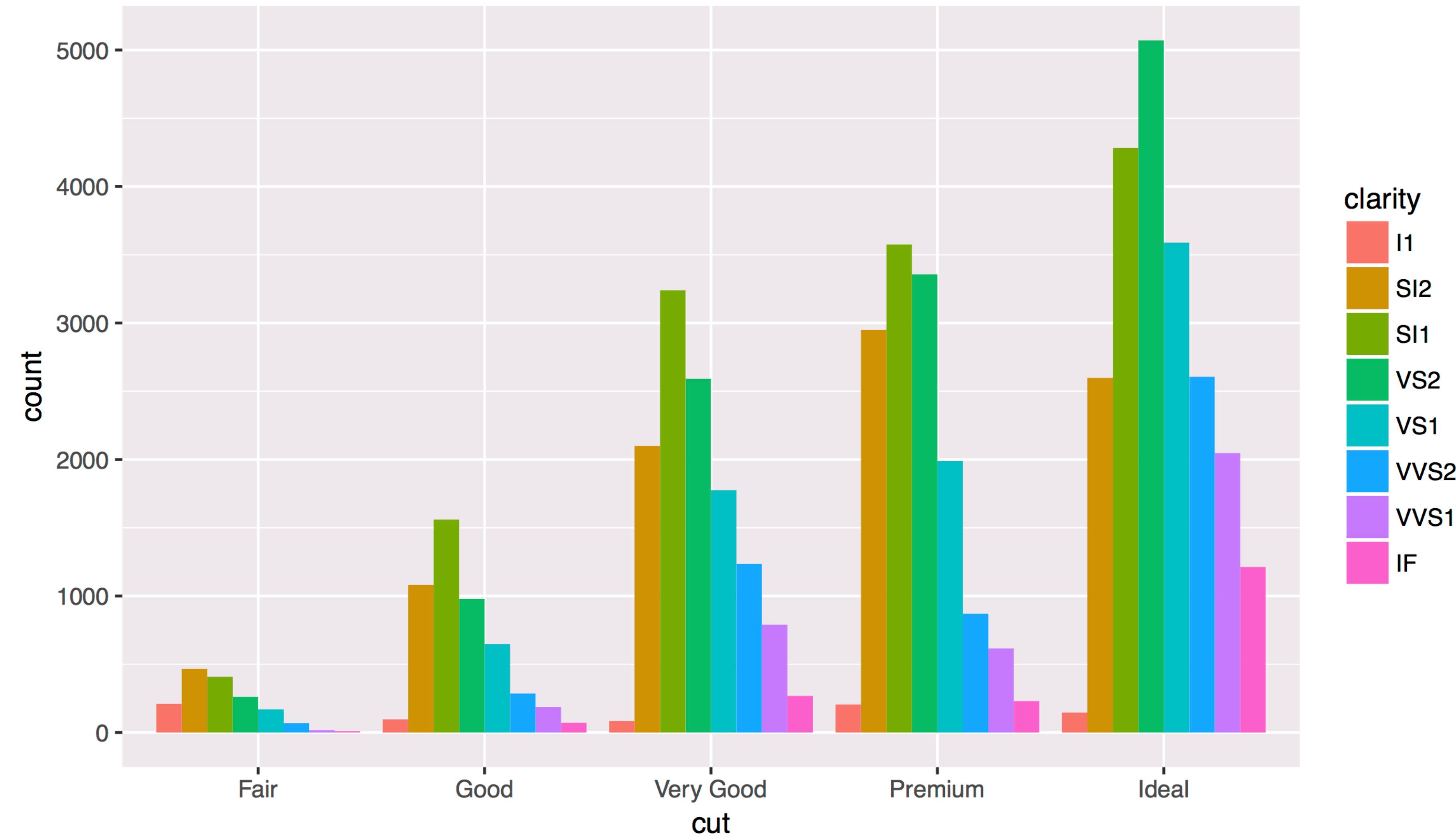


# Titles and captions



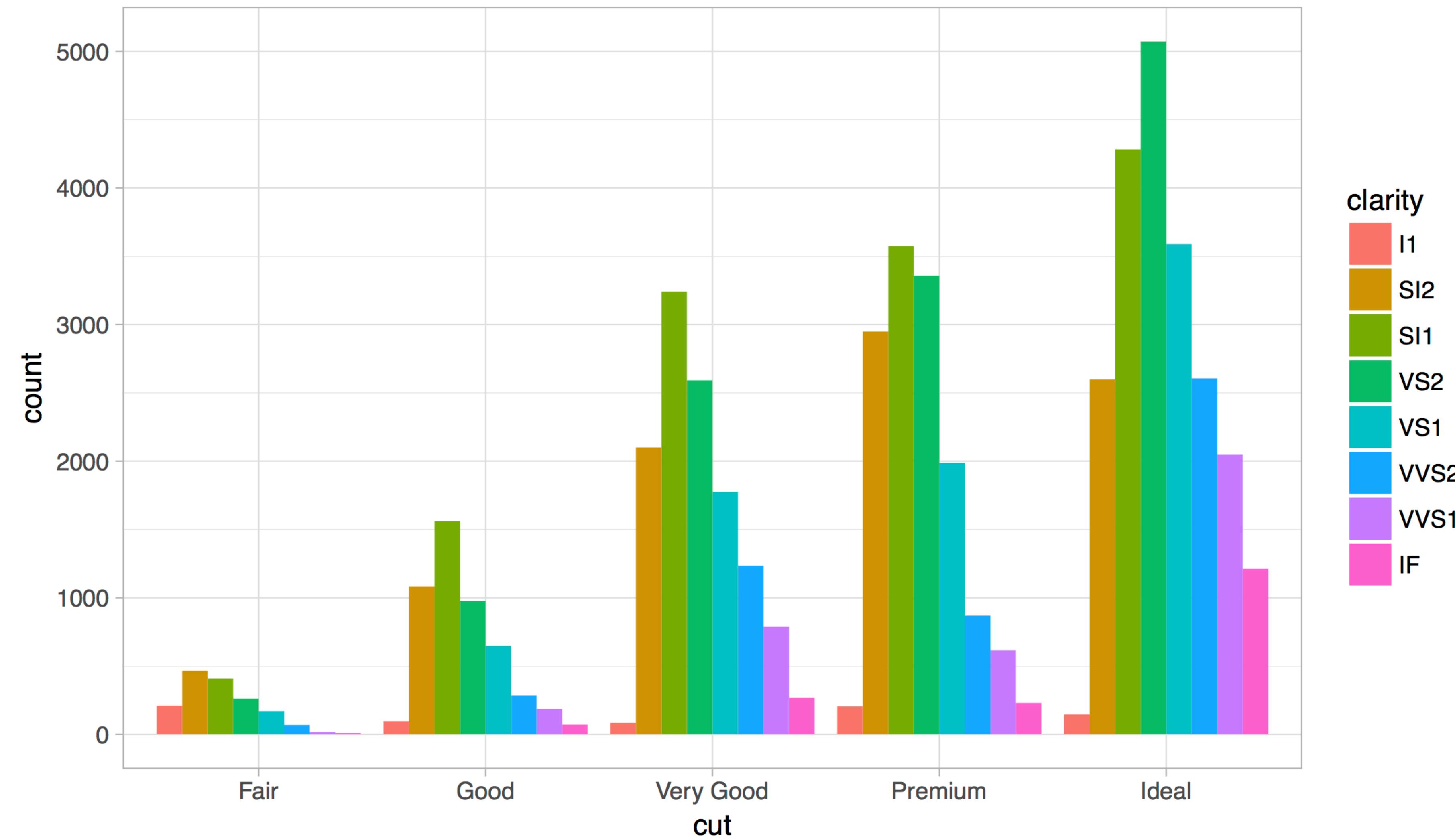
# Position Adjustments

## How overlapping objects are arranged



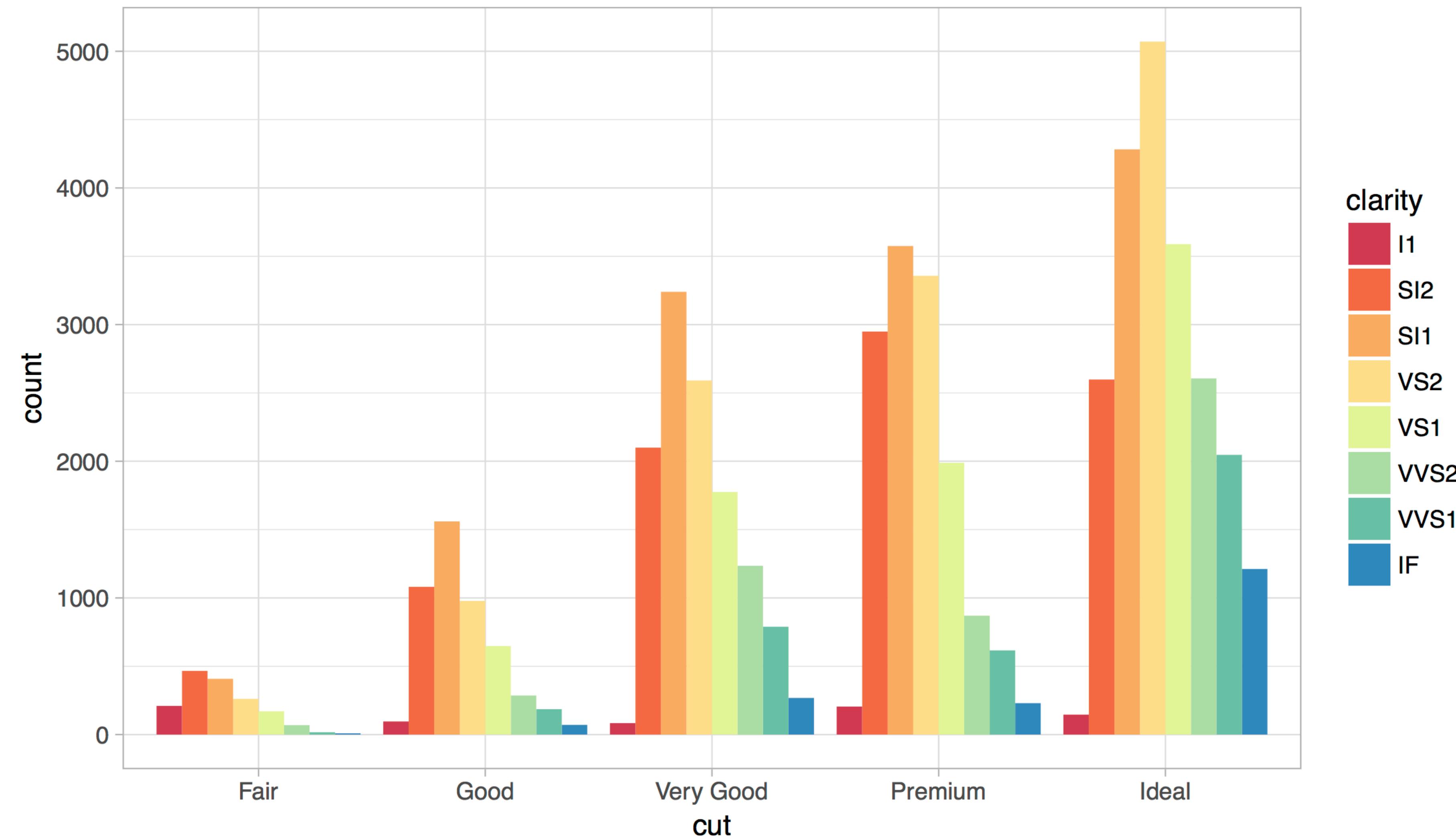
# Themes

## Visual appearance of non-data elements



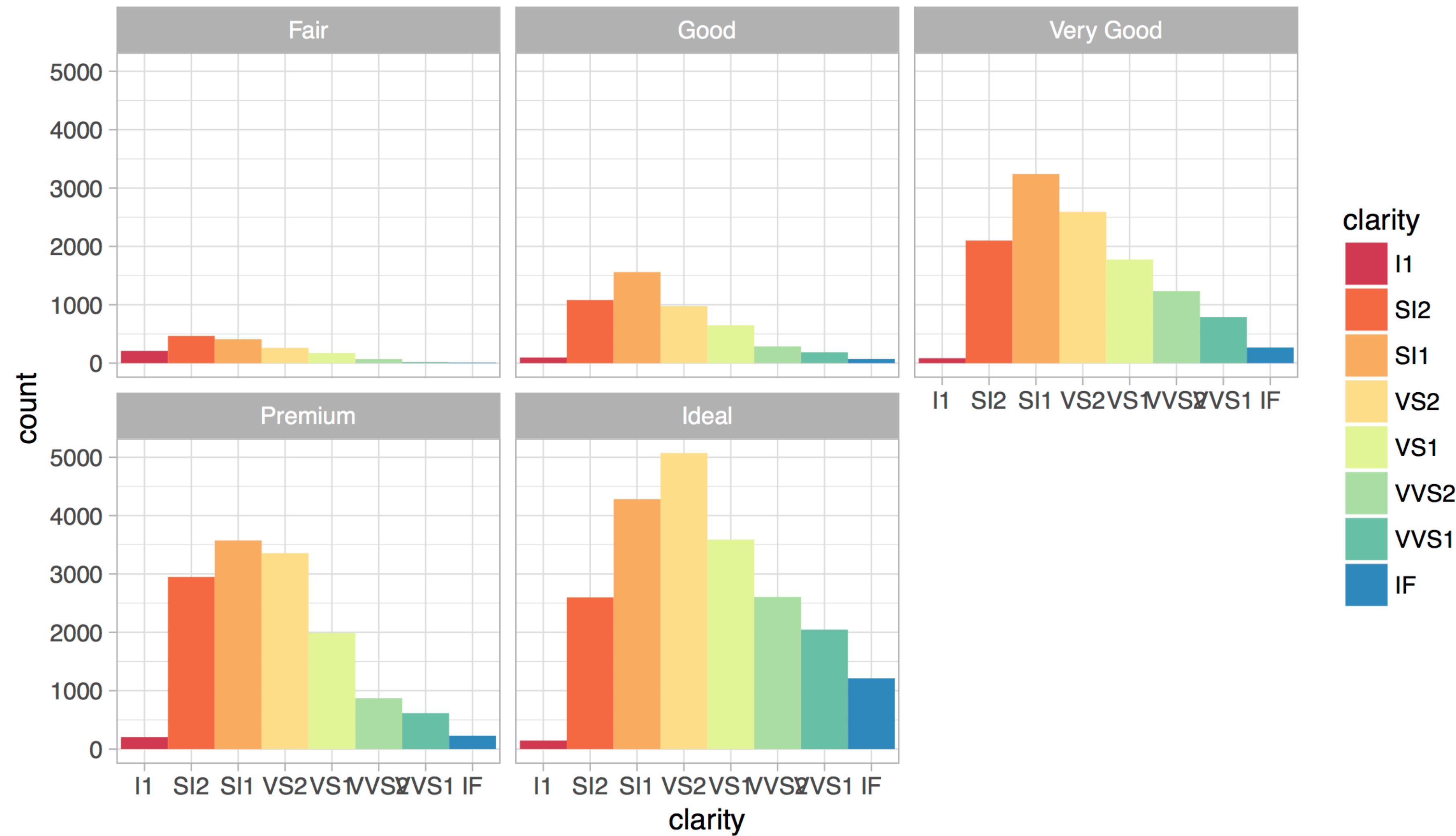
# Scales

Customize color scales, other mappings

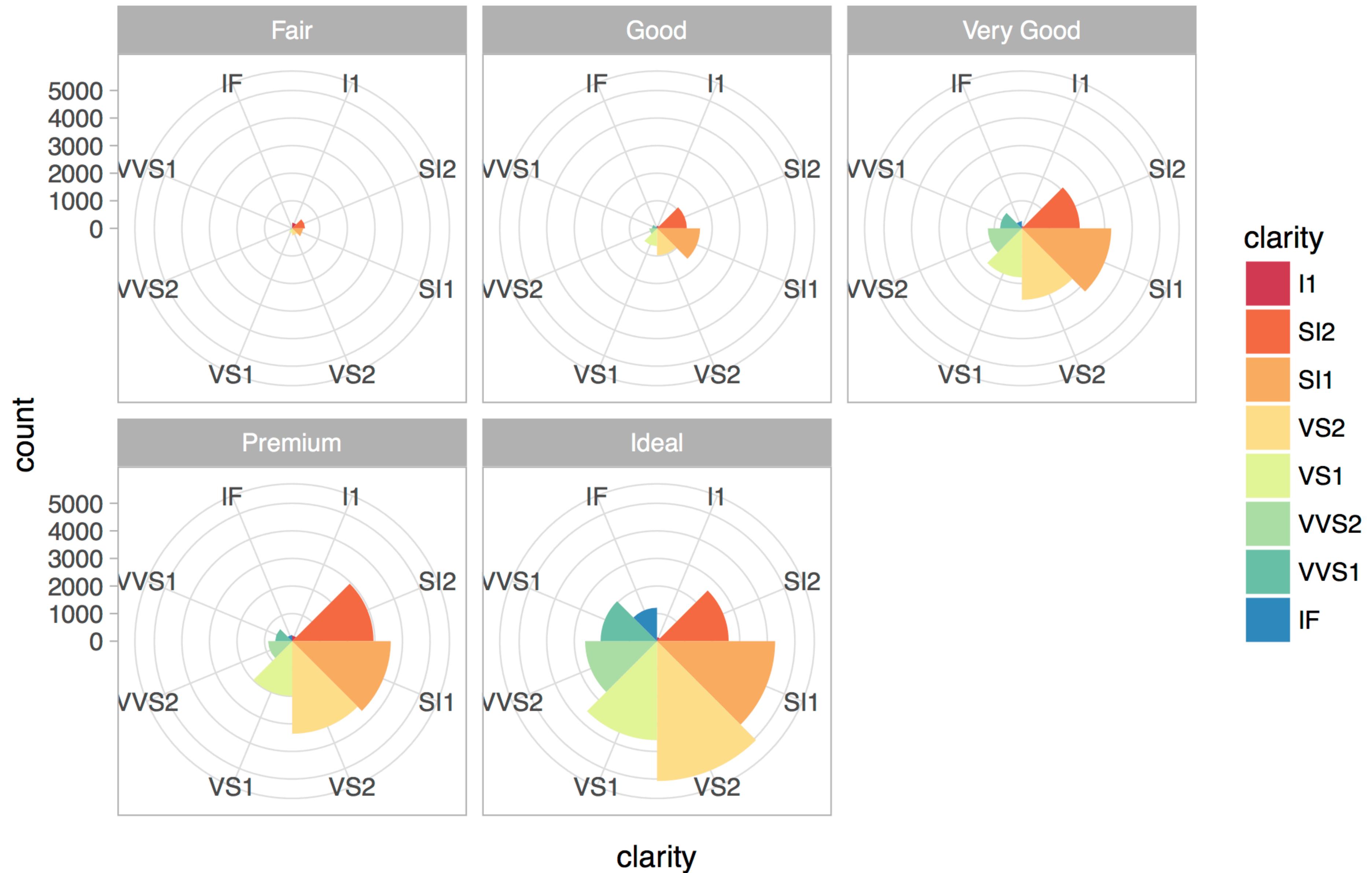


# Facets

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

Visualization with ggplot2 :: CHEAT SHEET

**Geoms** Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

**GRAPHICAL PRIMITIVES**

- a + `geom_blank()` (Used for expanding limits)
- b + `geom_curve(aes(yend = lat + 1, nudge_y = 1, check_overlap = TRUE))` x, y, label, alpha, angle, color, curvature, linetype, size
- a + `geom_path(lineend = "butt", linejoin = "round", linemetre = 1)` x, y, alpha, color, group, linetype, size
- a + `geom_polygon(aes(group = group))` x, y, alpha, color, fill, group, linetype, size
- b + `geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))` x, y, alpha, color, fill, group, linetype, size
- a + `geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))` x, y, alpha, color, fill, group, linetype, size

**TWO VARIABLES**

- continuous x , continuous y**
- continuous bivariate distribution**

**continuous function**

- i + `geom_area()` x, y, alpha, color, fill, linetype, size
- i + `geom_line()` x, y, alpha, color, group, linetype, size
- i + `geom_step(direction = "hv")` x, y, alpha, color, group, linetype, size

**visualizing error**

- f + `geom_crossbar(fatten = 2)` x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- j + `geom_errorbar()` x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- j + `geom_linerange()` x, y, ymin, ymax, alpha, color, group, linetype, size
- j + `geom_pointrange()` x, y, ymin, ymax, alpha, color, fill, group, linetype, size

**maps**

- data <- data.frame(murder = USArests\$Murder, state = tolower(rownames(USArests)))
- map <- map\_data("state")
- k <- ggplot(data, aes(fill = murder))
- k + `geom_map(aes(map_id = state), map = map) + expand_limits(x = map$long, y = map$lat, map_id, alpha, color, fill, linetype, size)`

**ONE VARIABLE**

- continuous**
- c <- `ggplot(mpg, aes(hwy))` Begins a plotting layers to. Add one geom
- c + `geom_boxplot()` x, y, alpha, color, fill, linetype, size
- c + `geom_density(kernel = "gaussian")` x, y, alpha, color, fill, group, linetype, size, weight
- c + `geom_dotplot()` x, y, alpha, color, fill
- c + `geom_freqpoly()` x, y, alpha, color, group, linetype, size
- c + `geom_histogram(binwidth = 5)` x, y, alpha, color, fill, linetype, size, weight
- c2 <- `ggplot(aes(sample = hwy))` x, y, alpha, color, fill, linetype, size, weight

**discrete**

- d <- `ggplot(mpg, aes(f1))`
- d + `geom_bar()` x, alpha, color, fill, linetype, size, weight

**THREE VARIABLES**

- seals2 <- `with(seals, sqrt(delta_long^2 + delta_lat^2))` l <- `ggplot(seals, aes(long, lat))`
- l + `geom_raster(aes(z = z))` x, y, z, alpha, colour, group, linetype, size, weight
- l + `geom_tile(aes(fill = z))` x, y, alpha, color, fill, linetype, size, width



RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at <http://ggplot2.tidyverse.org> • ggplot2 2.1.0 • Updated: 2016-11

# ggplot2.tidyverse.org

The screenshot shows a web browser window displaying the ggplot2.tidyverse.org website. The title bar reads "Create Elegant Data Visualisati x" and "Garrett". The address bar shows the URL "ggplot2.tidyverse.org". The page itself has a header with the ggplot2 logo and "part of the tidyverse". It features a main content area with a section titled "Usage" containing text and R code examples, and a "Links" sidebar with various links to download, browse source code, report bugs, and learn more.

## Usage

It's hard to succinctly describe how ggplot2 works because it embodies a deep philosophy of visualisation. However, in most cases you start with `ggplot()`, supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_flip()`).

```
library(ggplot2)

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

## Links

- Download from CRAN at [https://cran.r-project.org/  
package=ggplot2](https://cran.r-project.org/package=ggplot2)
- Browse source code at [https://github.com/tidyverse/  
ggplot2](https://github.com/tidyverse/ggplot2)
- Report a bug at [https://github.com/tidyverse/  
ggplot2/issues](https://github.com/tidyverse/ggplot2/issues)
- Learn more at [http://r4ds.had.co.nz/data-  
visualisation.html](http://r4ds.had.co.nz/data-<br/>visualisation.html)

## License

[GPL-2](#) | file [LICENSE](#)

## Developers

Hadley Wickham  
Author, maintainer



CC by RStudio