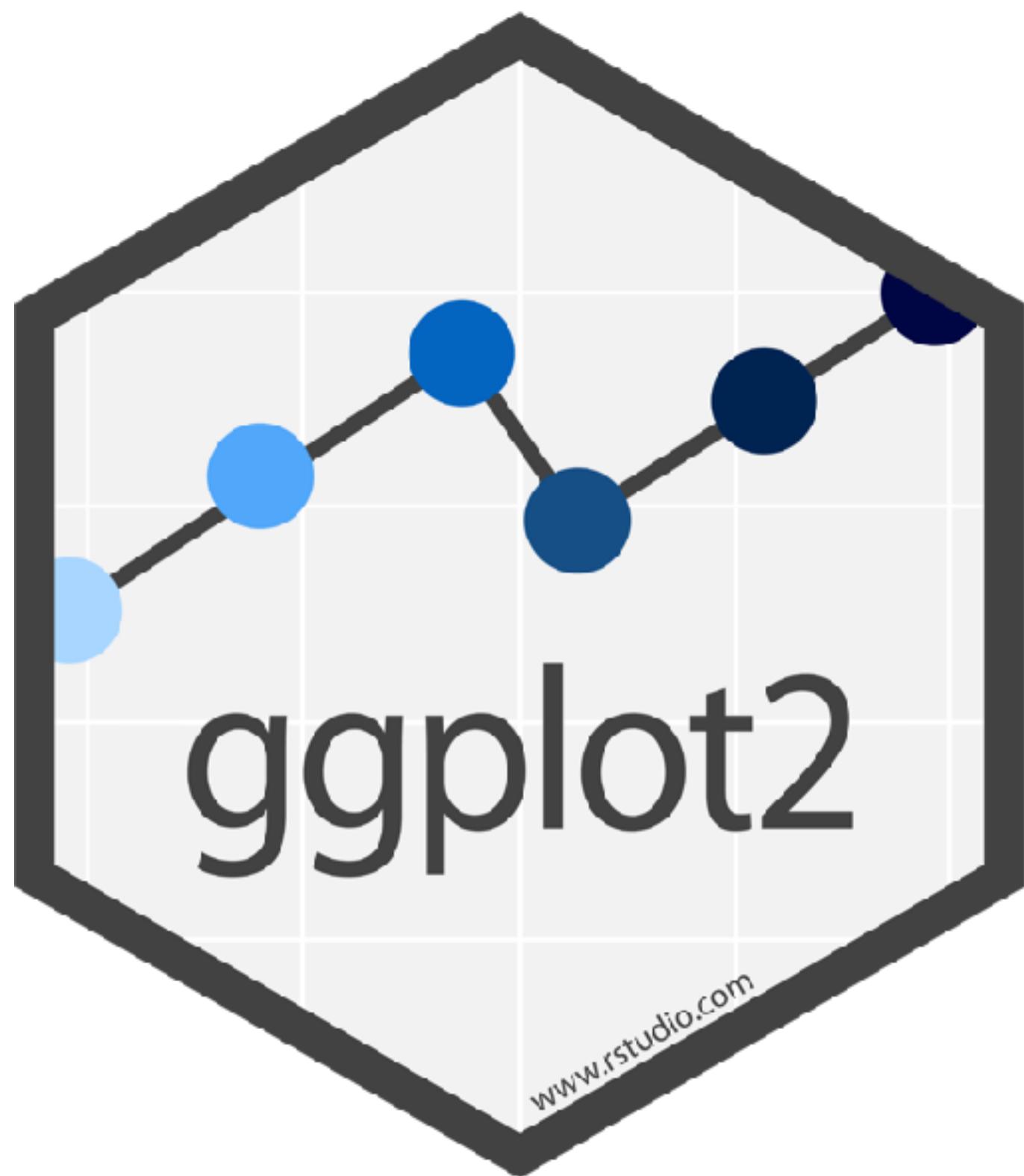
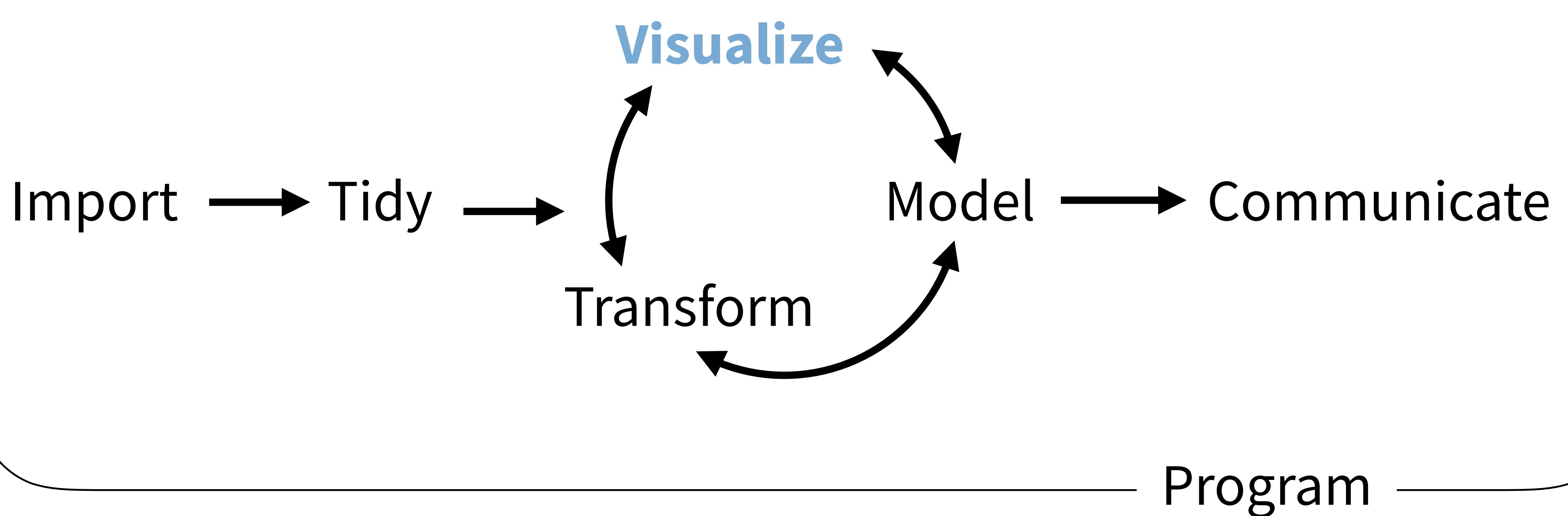


# Visualize Data with



# (Applied) Data Science



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

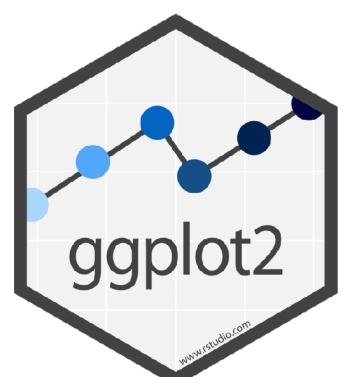
- John Tukey

# mpg

Fuel economy data for 38 models of car.

mpg

manufacturer	displ	year	cyl	trans	drv	cty	h...	fl	class
<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
audi	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	2.8	1999	6	manual(m5)	f	18	26	p	compact
audi	3.1	2008	6	auto(av)	f	18	27	p	compact



# Quiz

Confer with your group.

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

No peeking ahead!

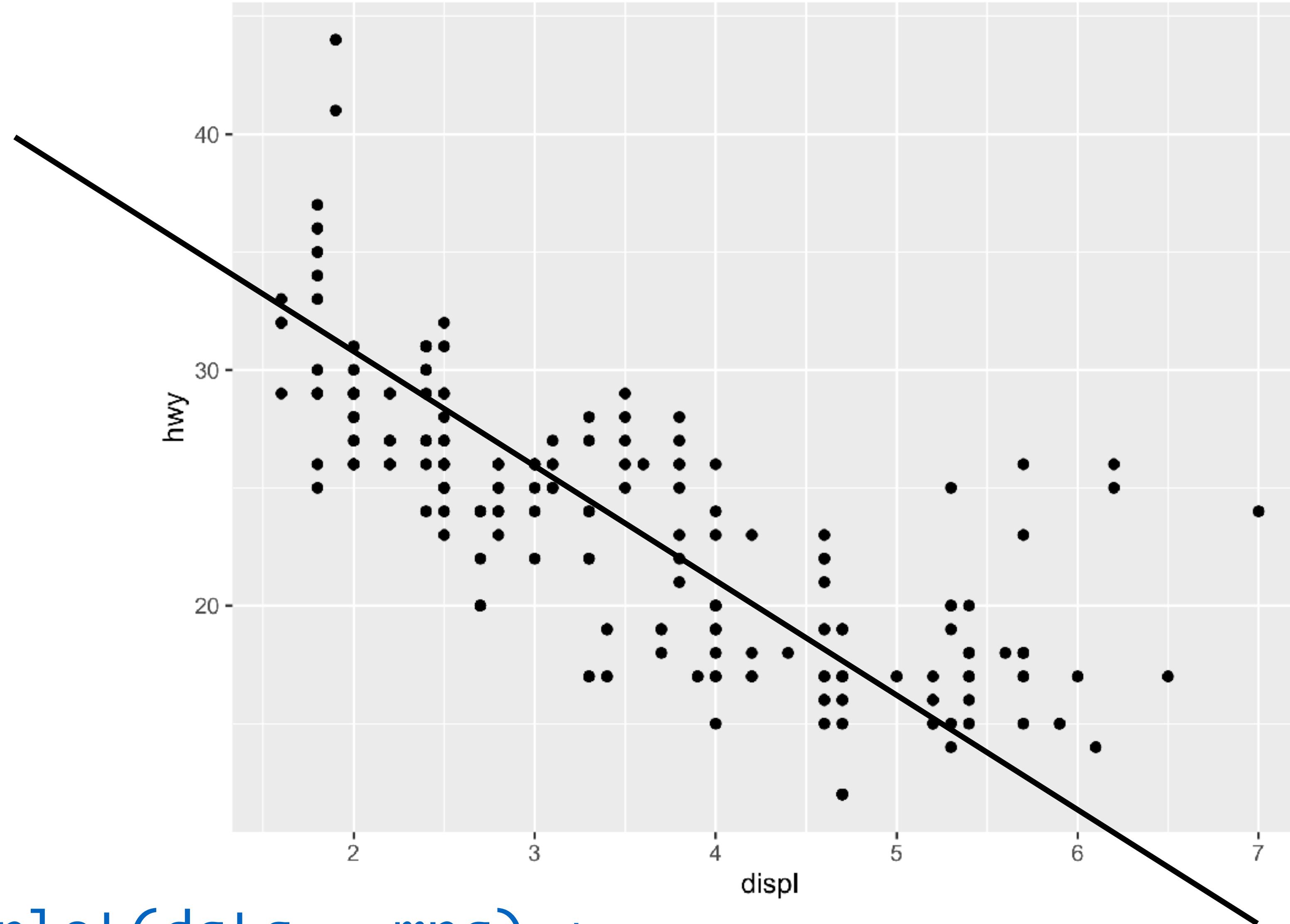


# Your Turn 1

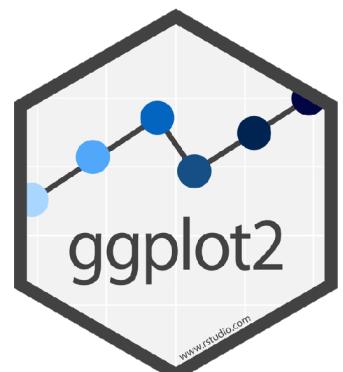
Run this code in **02-Visualize-Exercises.Rmd** to make a graph. Pay strict attention to spelling, capitalization, and parentheses!

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



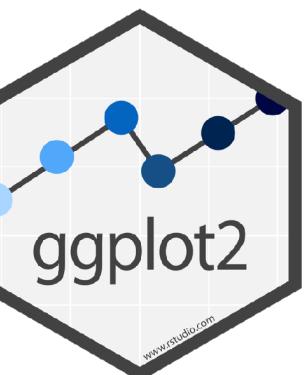


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```



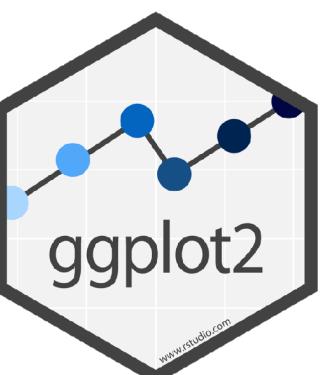
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

```
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

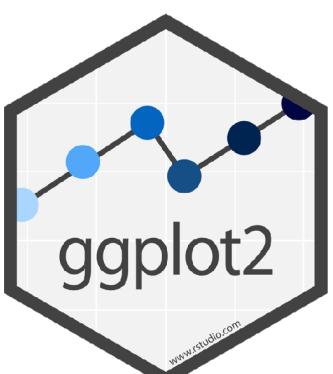
+ before new line

type of layer

aes()

x variable

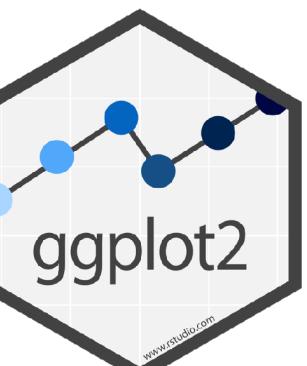
y variable



# A template

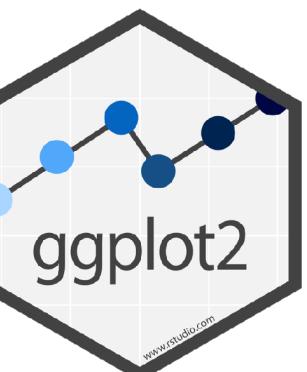
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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



# A template

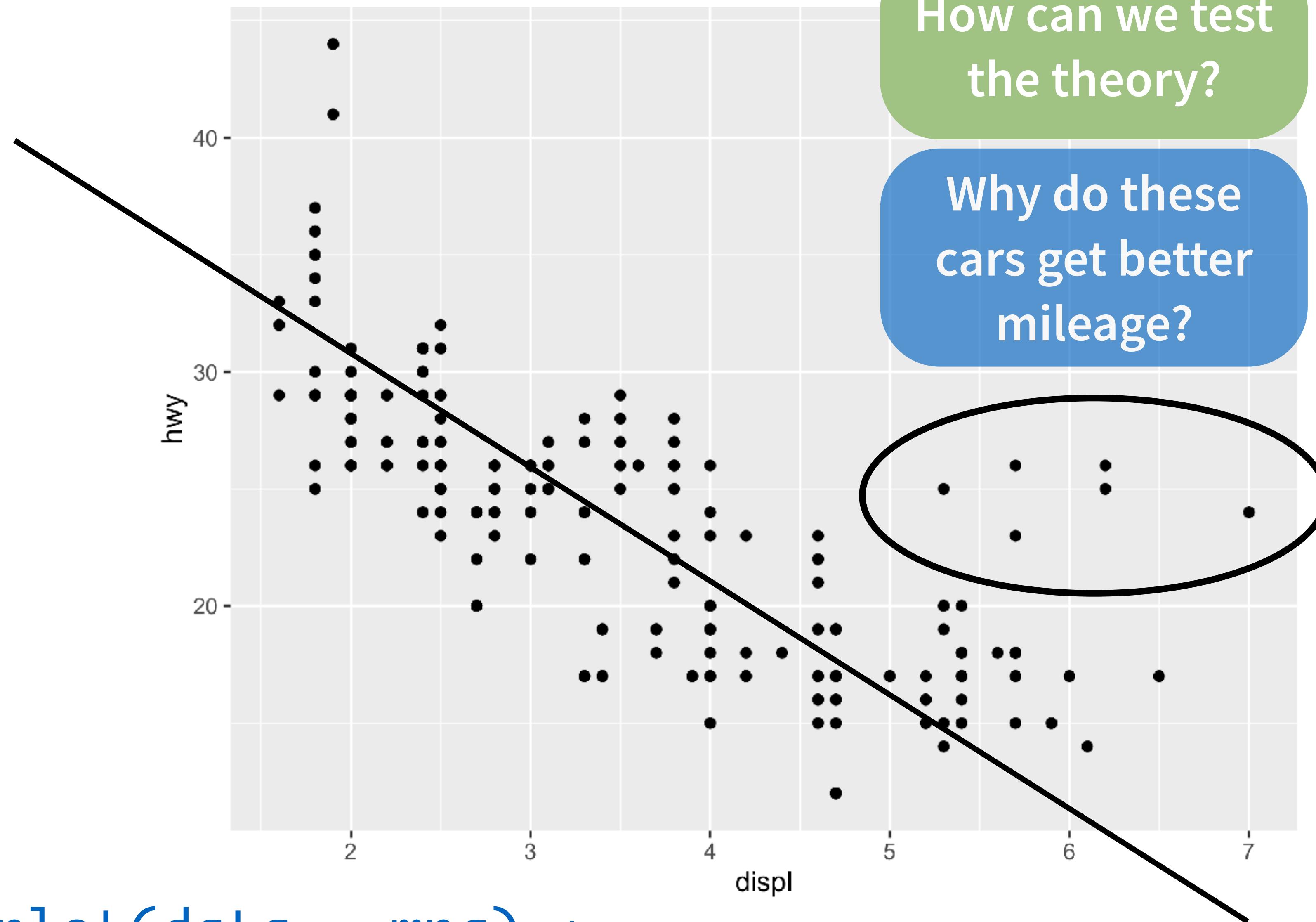
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```



# Mappings

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

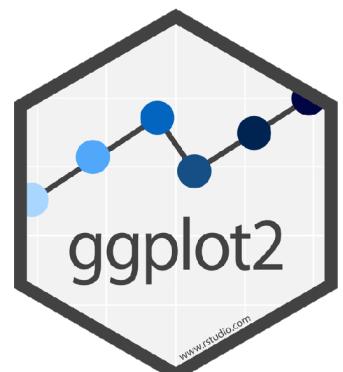
- John Tukey



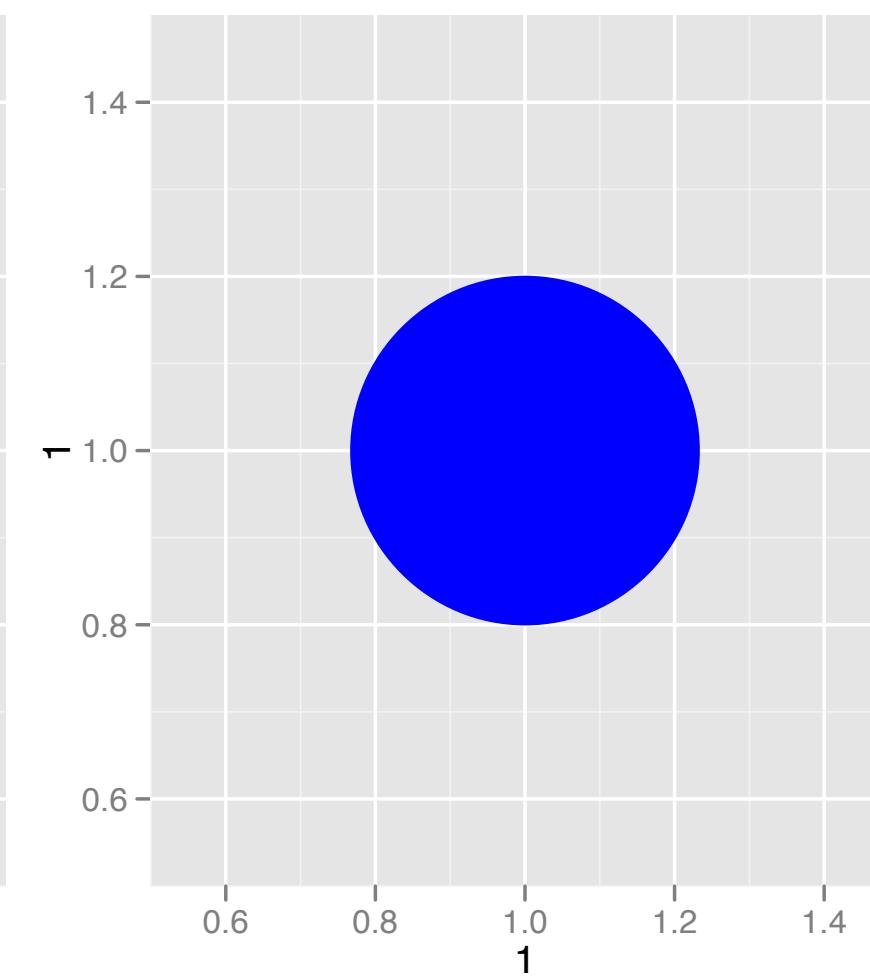
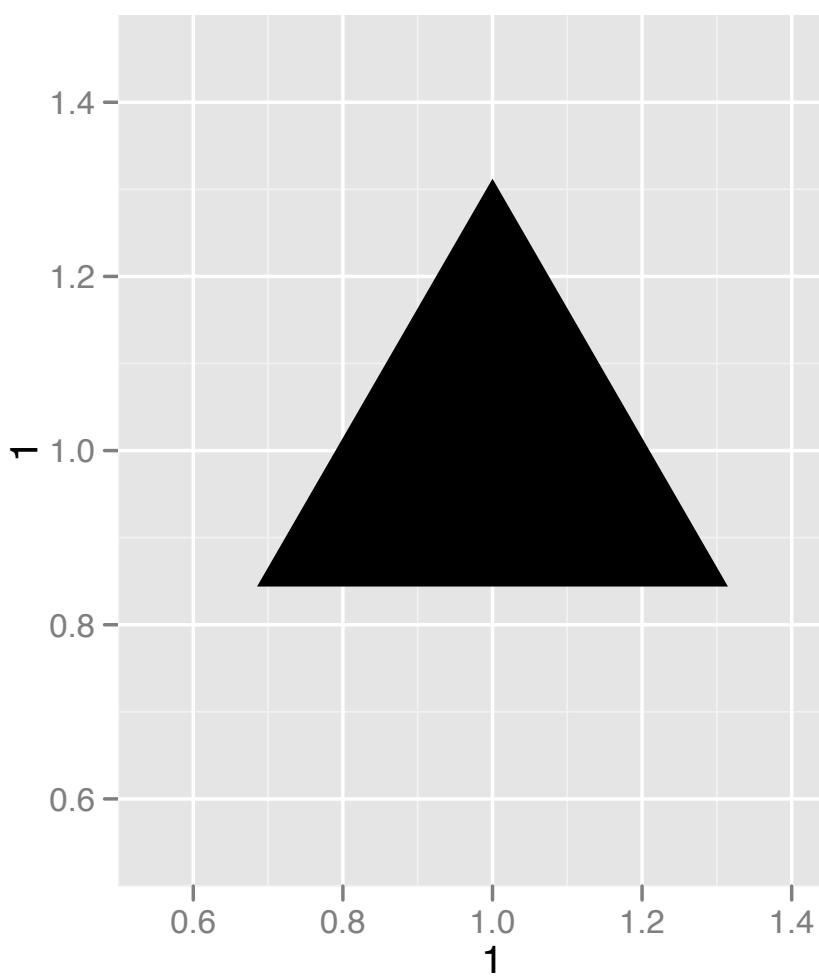
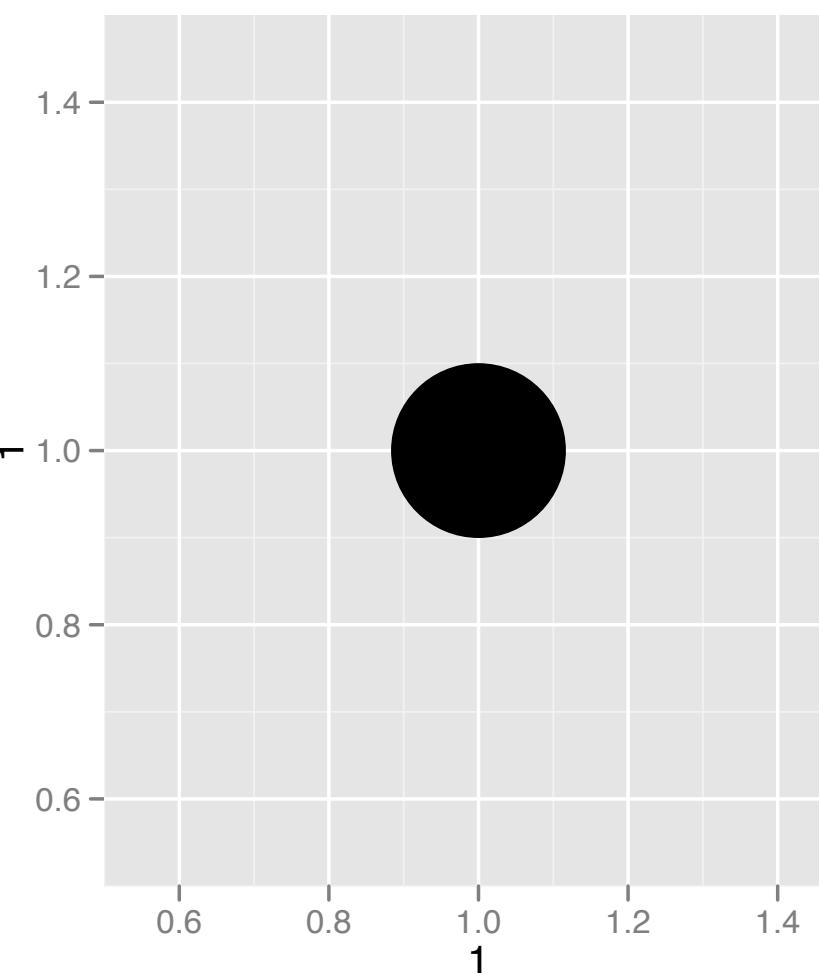
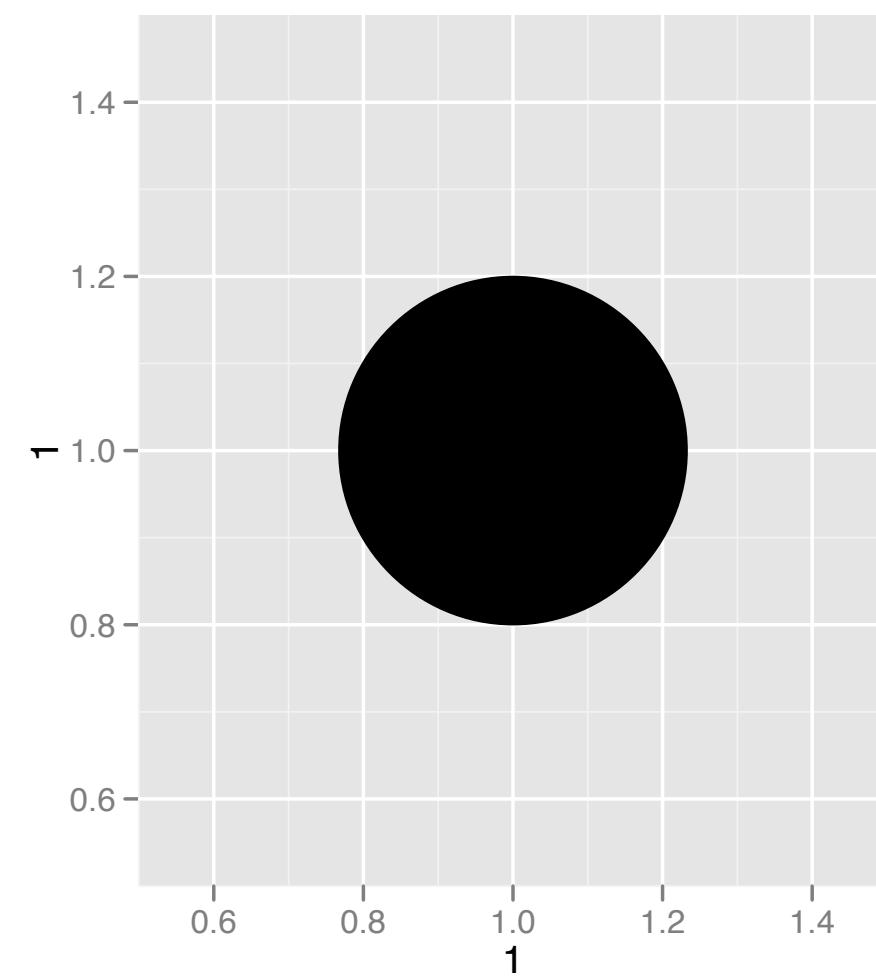
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can we test  
the theory?

Why do these  
cars get better  
mileage?



# Aesthetics

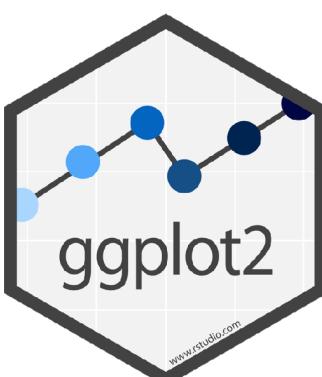


# Aesthetics

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))
```



# 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?



## Visual Space

color

Red

Brown

Green

Aqua

Blue

Violet

Pink

## Data Space

class

2seater

compact

midsize

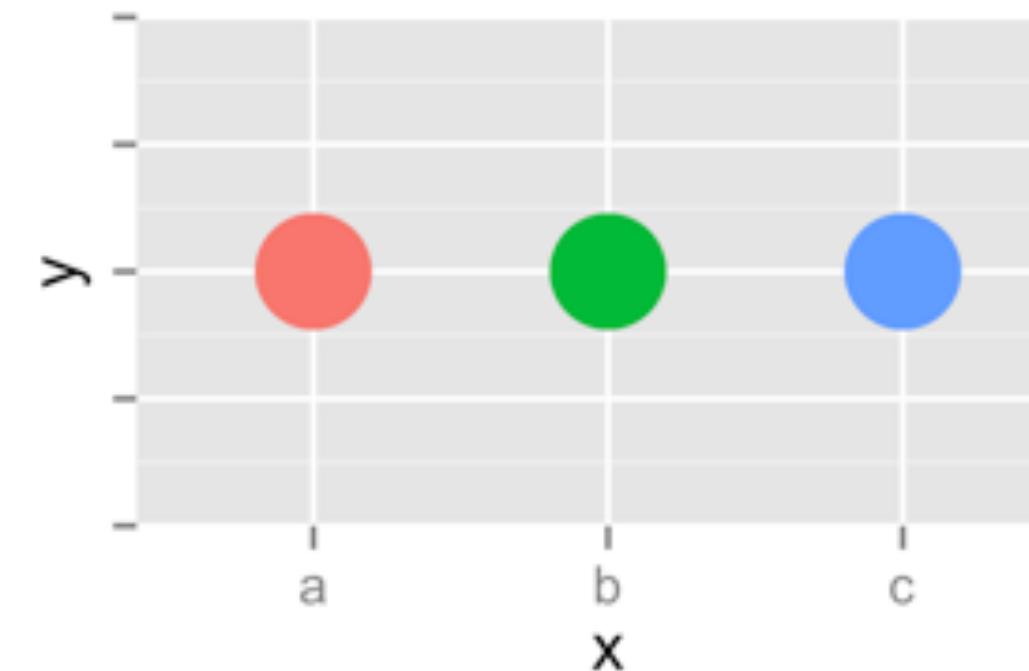
minivan

pickup

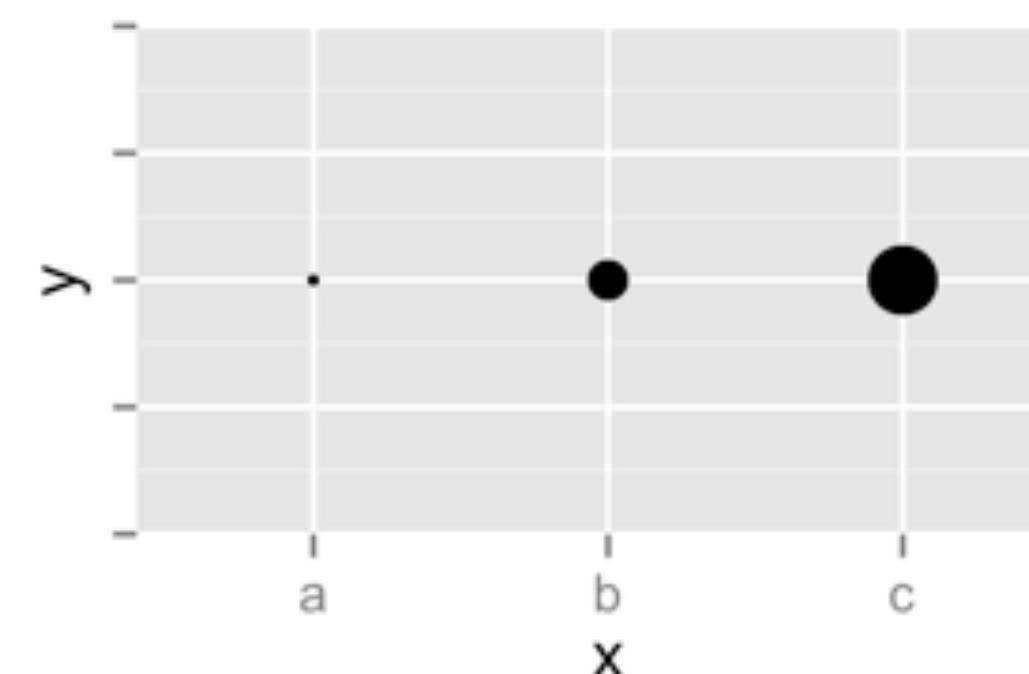
subcompact

suv

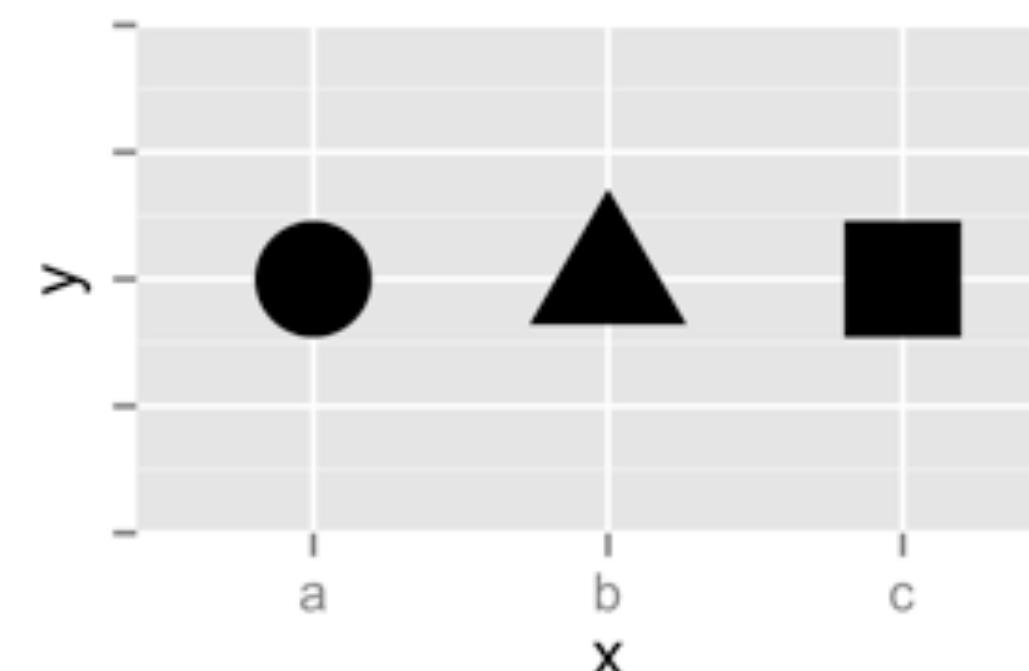
Color



Size

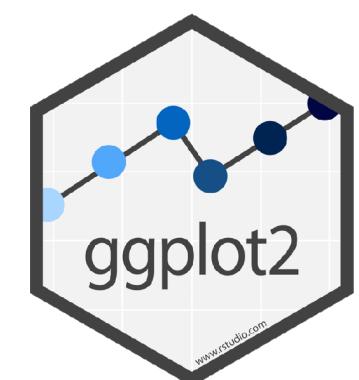
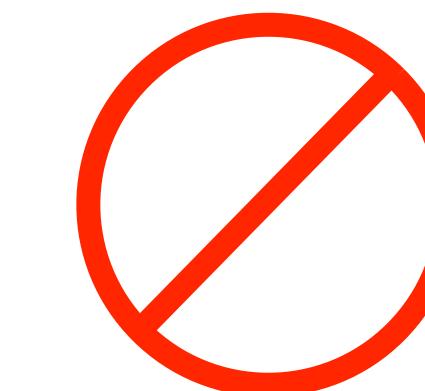
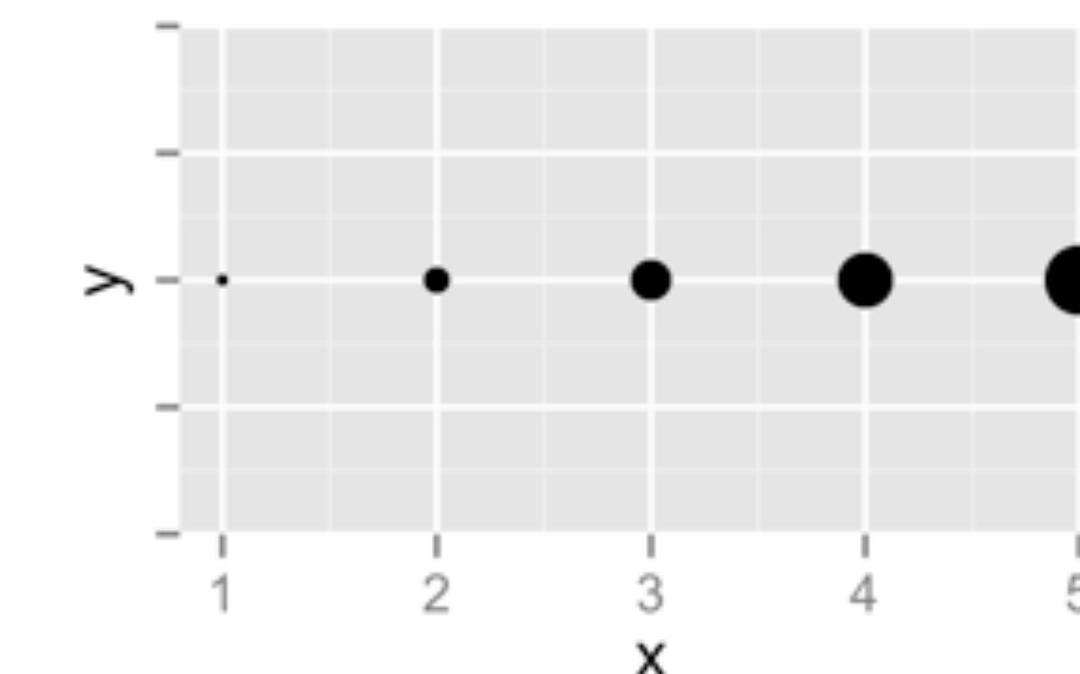
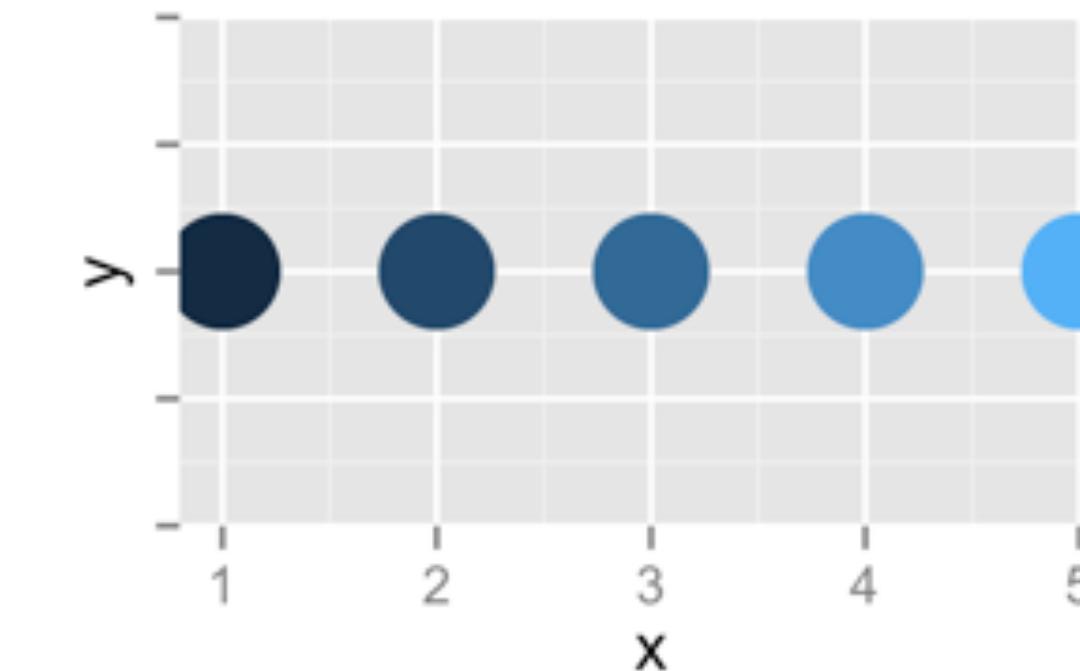


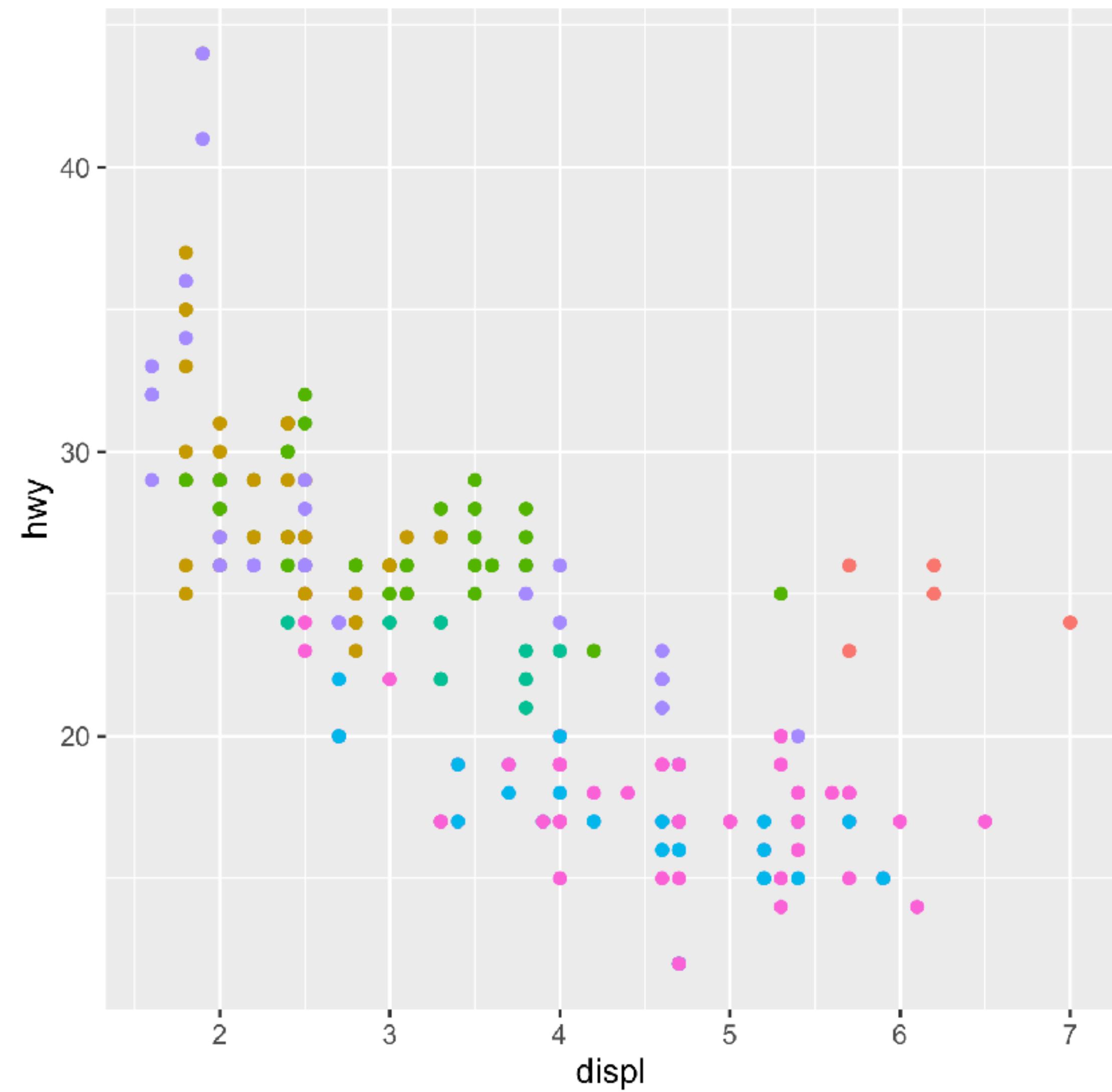
Shape



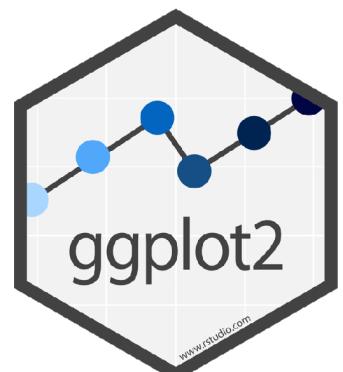
20

Continuous



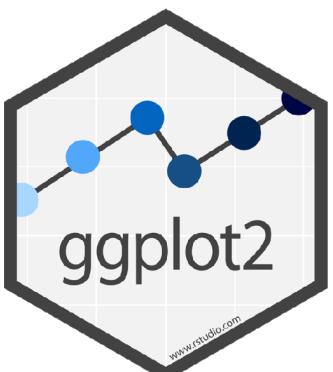


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



# ERROR!

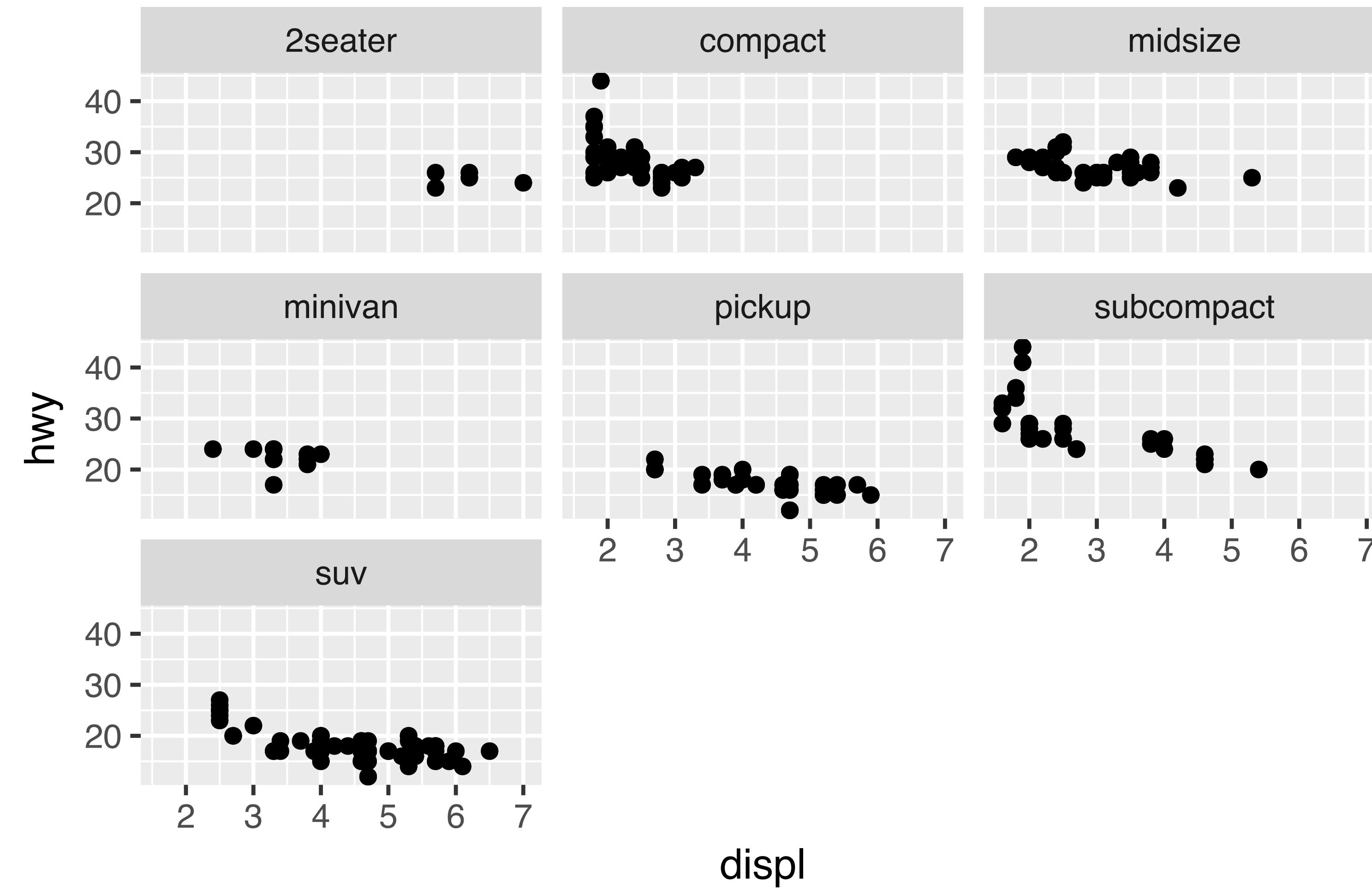
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy), color = class)
```



# Facets

# Facets

Subplots that display subsets of the data.



# Help me

What do `facet_grid` and `facet_wrap` do?

```
q <- ggplot(mpg) + geom_point(aes(x = displ, y = hwy))  
q + facet_grid(. ~ cyl)  
q + facet_grid(drv ~ .)  
q + facet_grid(drv ~ cyl)  
q + facet_wrap(~ class)
```

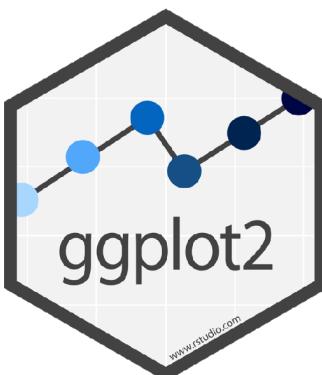
# summary

`facet_grid()` - 2D grid, rows ~ cols, . for no split  
`facet_wrap()` - 1D ribbon wrapped into 2D

# A ggplot2 template

Make any plot by filling in the parameters of this template

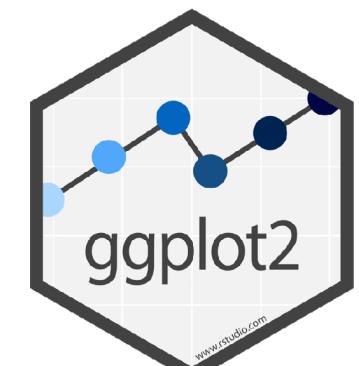
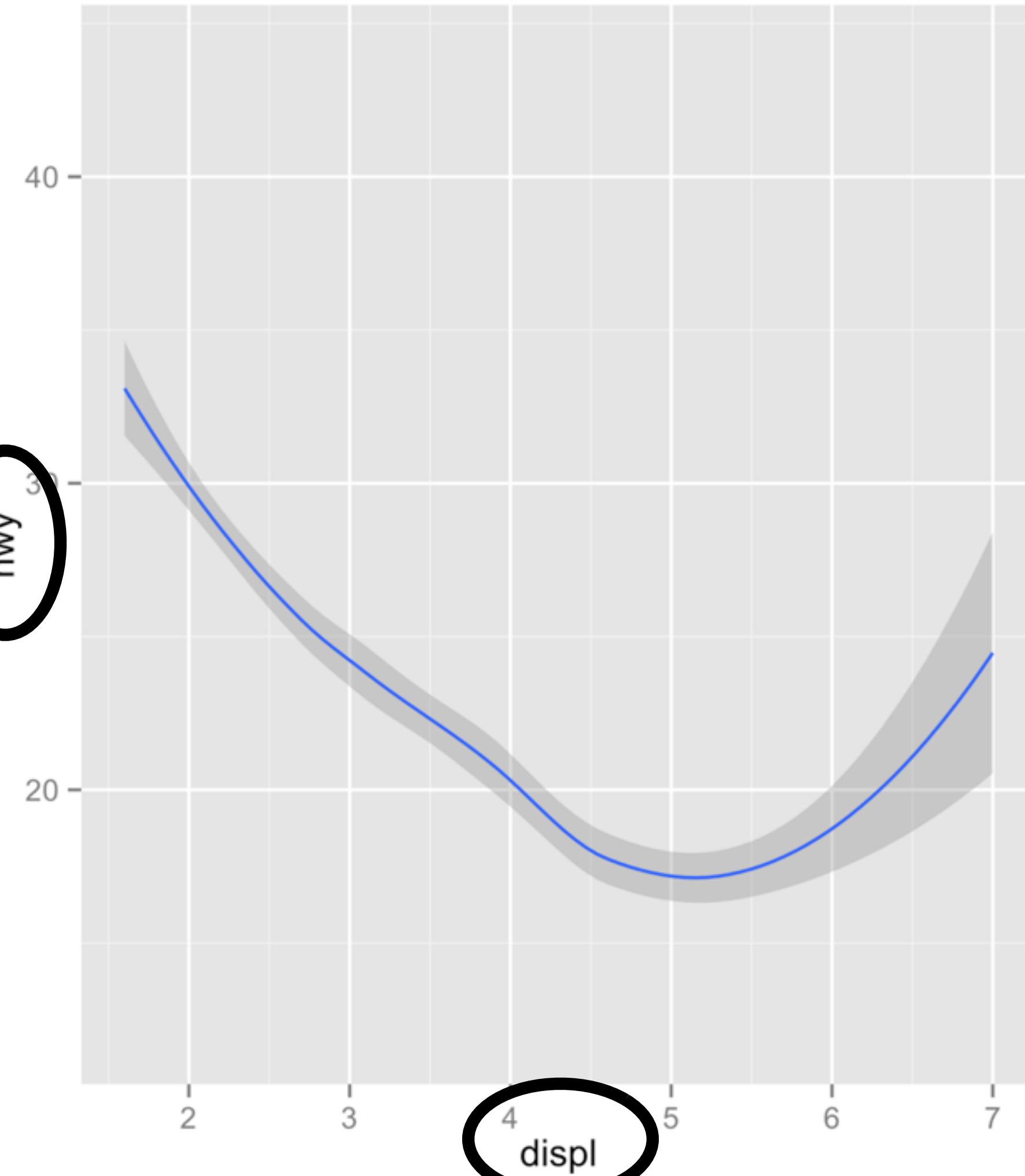
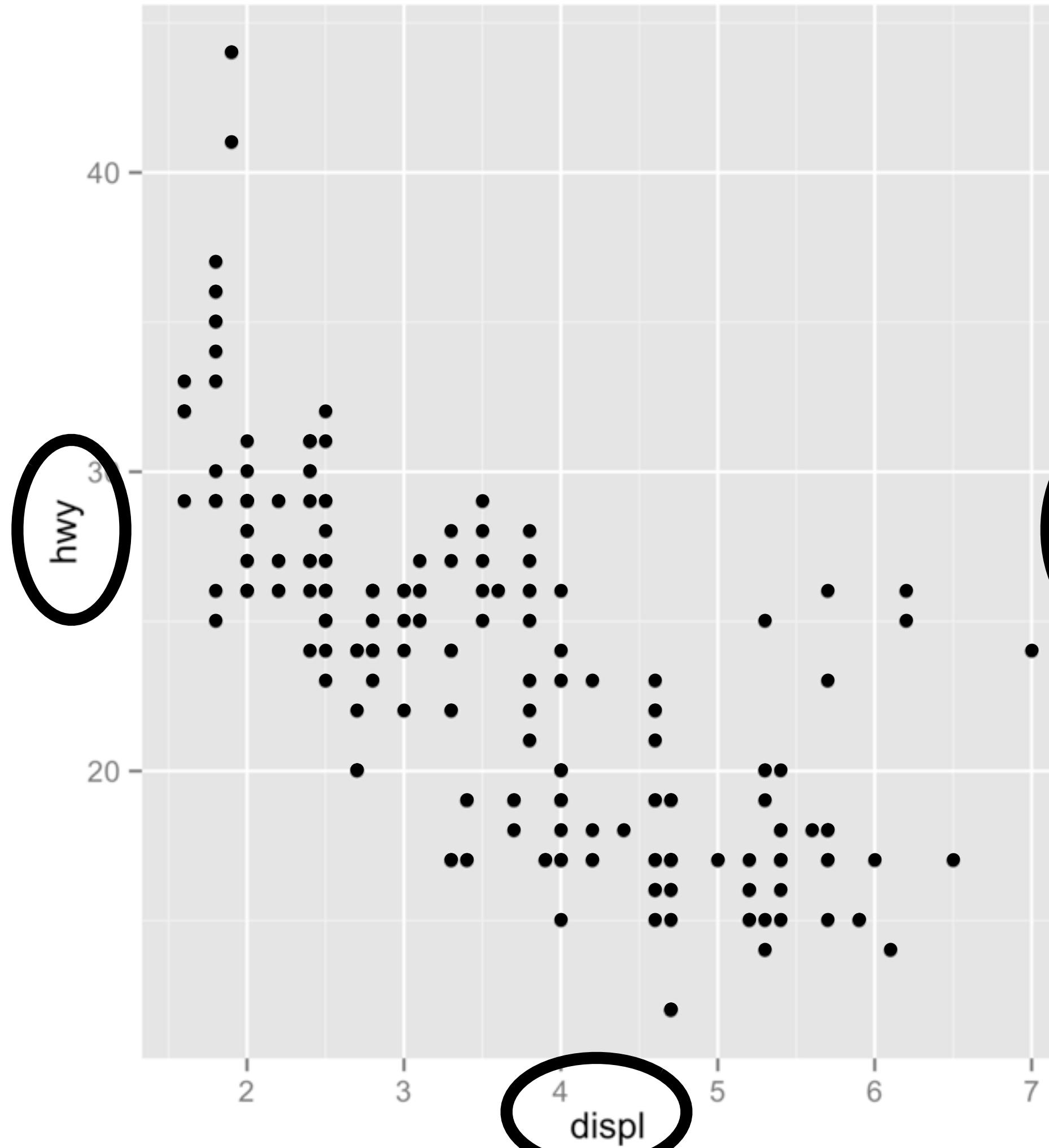
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>)) +  
<FACET_FUNCTION>
```



# Geoms

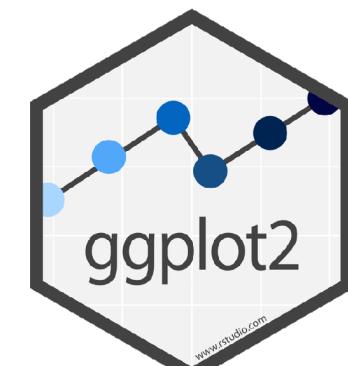
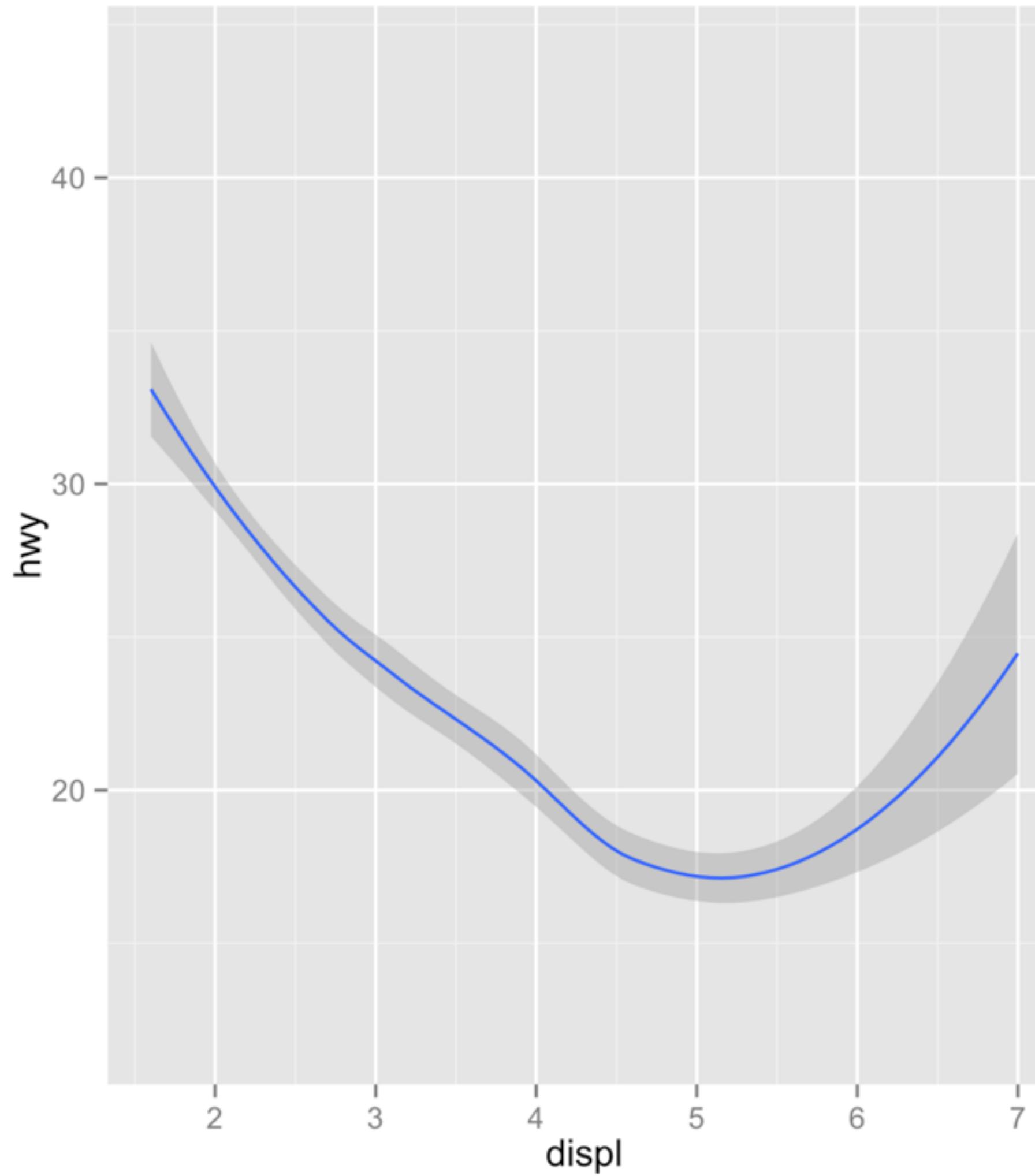
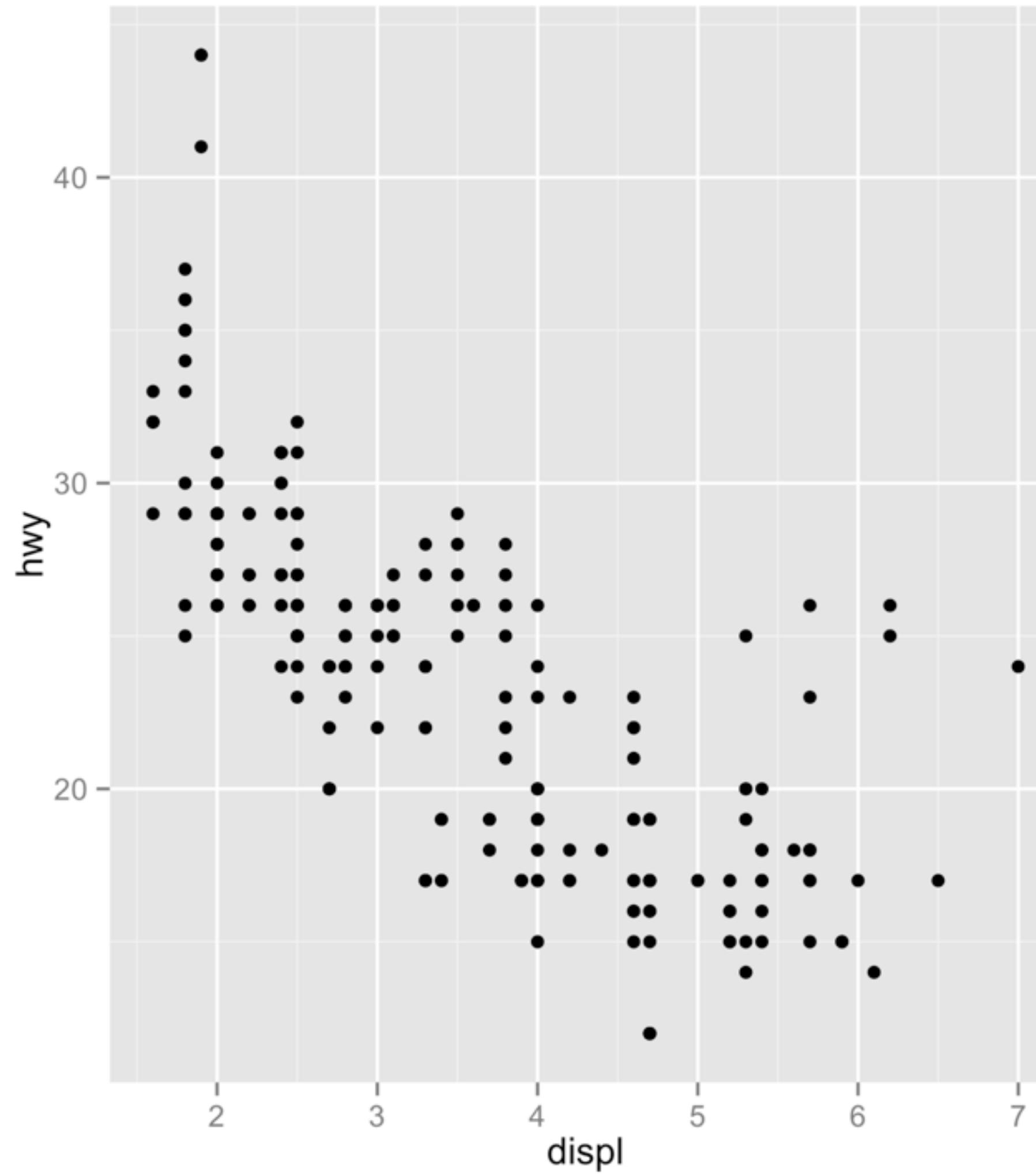
How are these plots similar?

Same: x var , y var , data



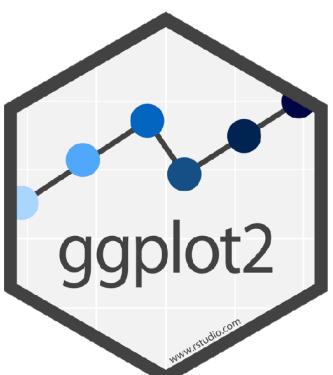
How are these plots different?

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



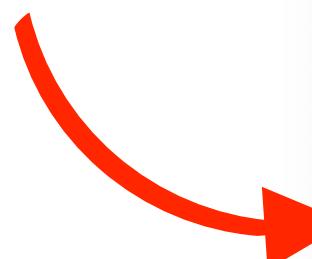
# geoms

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



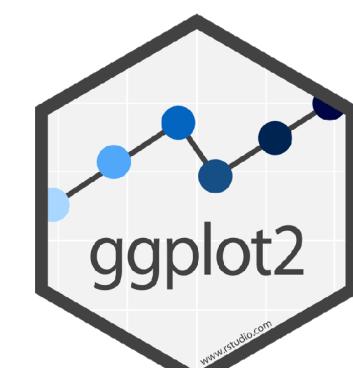
# [rstudio.cloud/learn/cheat-sheets](https://rstudio.cloud/learn/cheat-sheets)

**CLICK  
CHEAT SHEETS  
IN THE SIDEBAR**



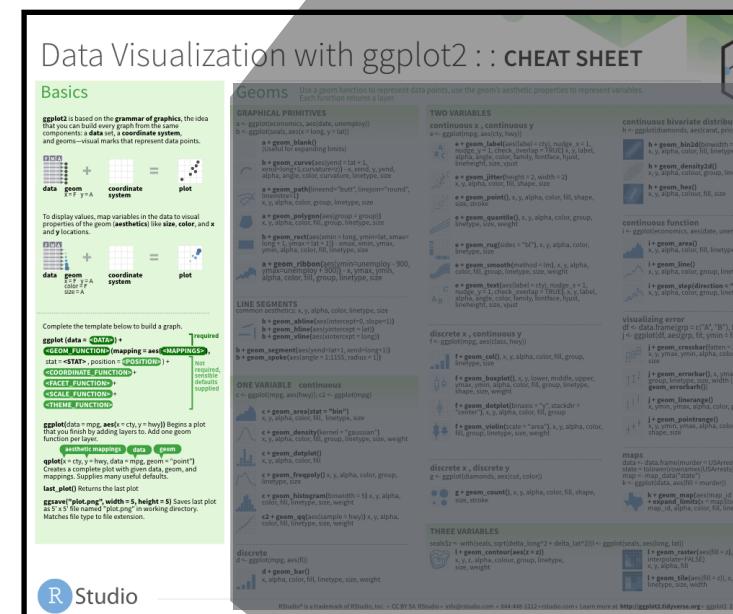
The screenshot shows the RStudio Cloud interface with the 'Learn' tab selected in the top navigation bar. On the left, a sidebar menu is open, with the 'Cheat Sheets' option highlighted in blue. The main content area displays four cheat sheets: 'Deep Learning with Keras', 'Work with Strings', 'Dates and Times', and 'Apply Functions'. Each card includes a preview of the cheat sheet, a 'Download' button, and an 'Updated December 2017' note.

- Deep Learning with Keras**  
Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. Keras supports both convolution based networks and recurrent networks (as well as combinations of the two), runs seamlessly on both CPU and GPU devices, and is capable of running on top of multiple back-ends including TensorFlow, CNTK, and Theano.  
Updated December 2017
- Work with Strings**  
The stringr package provides an easy to use toolkit for working with strings, i.e. character data, in R. This cheatsheet guides you through stringr's functions for manipulating strings. The back
- Dates and Times**  
Lubridate makes it easier to work with dates and times in R. This lubridate cheatsheet covers how to round dates, work with time zones, extract elements of a date or time, parse dates into R and more. The back of the cheatsheet describes lubridate's three timespan classes: periods, durations, and intervals; and explains how to do math with date-times.  
Updated December 2017
- Apply Functions**  
The purrr package makes it easy to work with lists and functions. This cheatsheet will remind you how to manipulate lists with purrr as well as how to apply functions iteratively to each element of a list



# geom\_ functions

Each requires a mapping argument.



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

**ggplot2**

**GRAPHICAL PRIMITIVES**

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

**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\_jitter(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

**continuous bivariate distribution**

```
h <- ggplot(diamonds, aes(carat, price))
```

- h + geom\_bin2d(binwidth = c(0.25, 500))** x, y, alpha, color, fill, linetype, size, weight
- h + geom\_density2d()** x, y, alpha, colour, group, linetype, size
- h + geom\_hex()** x, y, alpha, colour, fill, size

**continuous function**

```
i <- ggplot(economics, aes(date, unemploy))
```

- 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**

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

- j + geom\_crossbar(fatten = 2)** x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- j + geom\_errorbar()**, x, y, ymax, ymin, alpha, color, group, linetype, size, width (also **geom\_errorbarh()**)
- j + geom\_linerange()** x, ymin, ymax, alpha, color, group, linetype, size
- j + geom\_pointrange()** x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

**maps**

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
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)); c2 <- ggplot(mpg)
```

- c + geom\_area(stat = "bin")** 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 + geom\_qq(aes(sample = hwy))** x, y, alpha, color, fill, linetype, size, weight

**discrete x , discrete y**

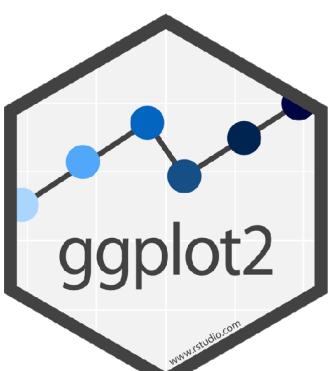
```
g <- ggplot(diamonds, aes(cut, color))
```

- 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



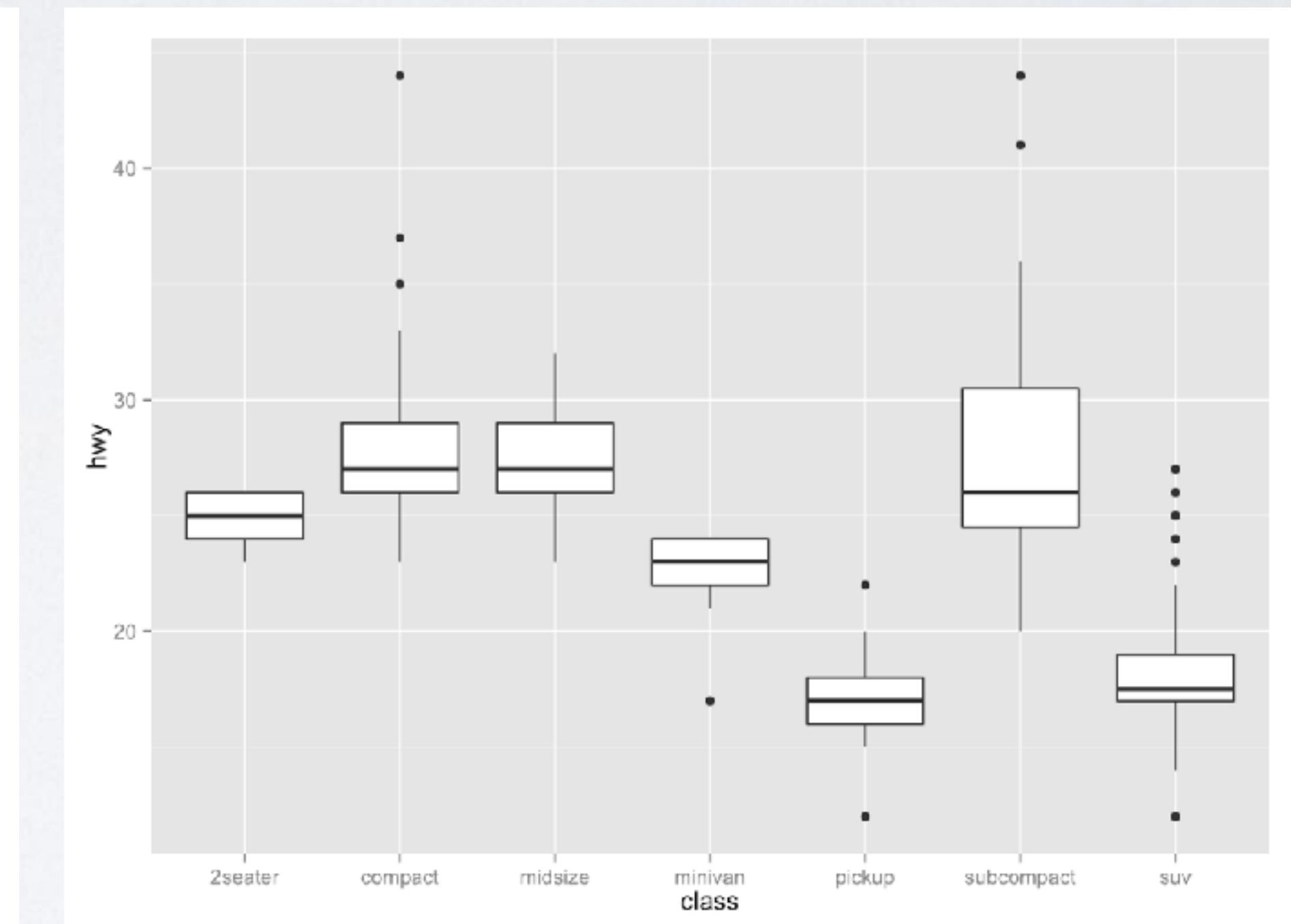
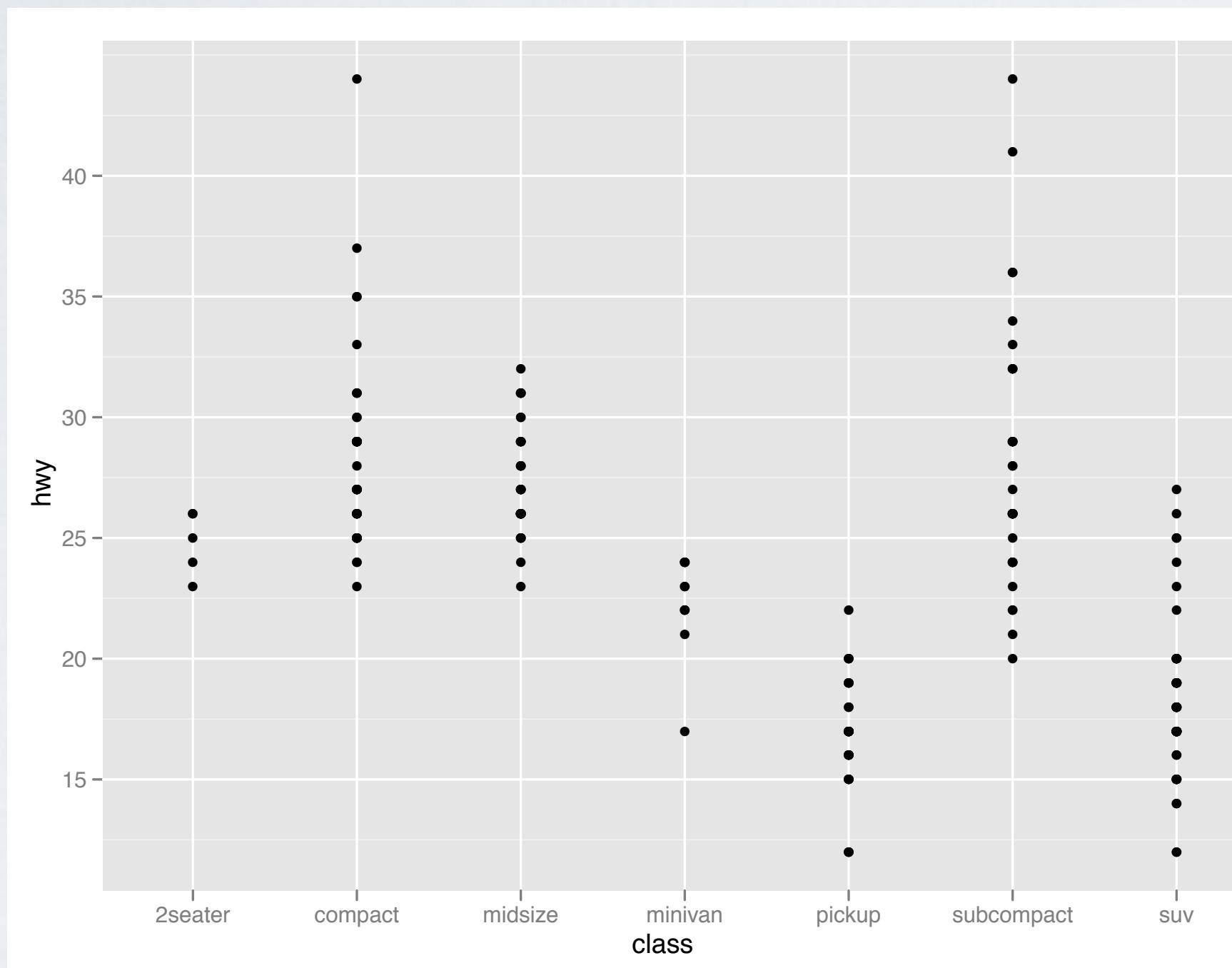
# Your Turn

Pair up.



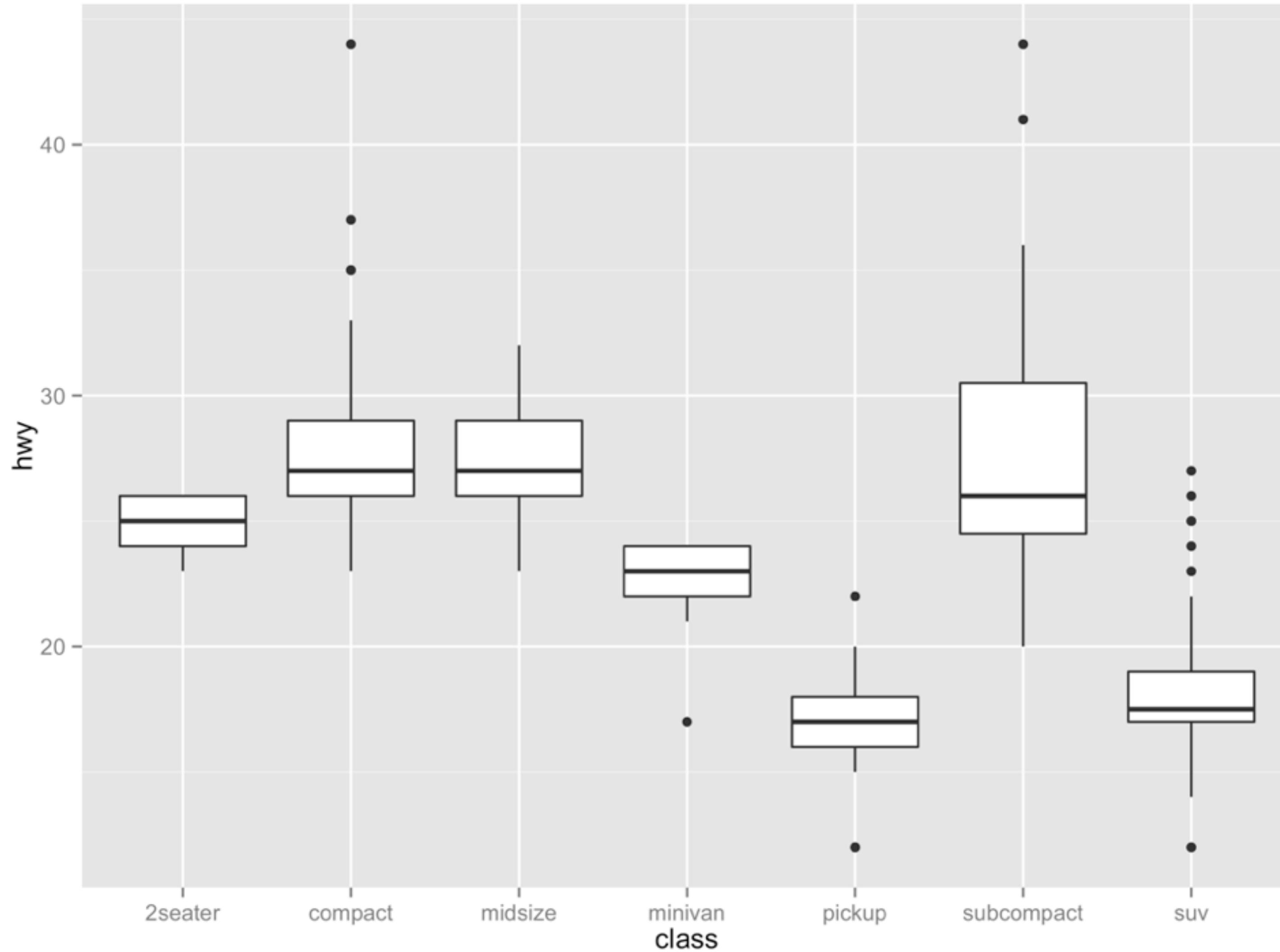
# Your Turn 3

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

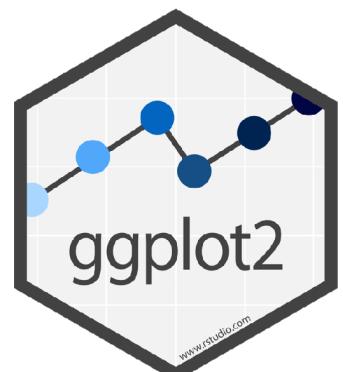


```
ggplot(mpg) + geom_point(aes(class, hwy))
```

02 : 00

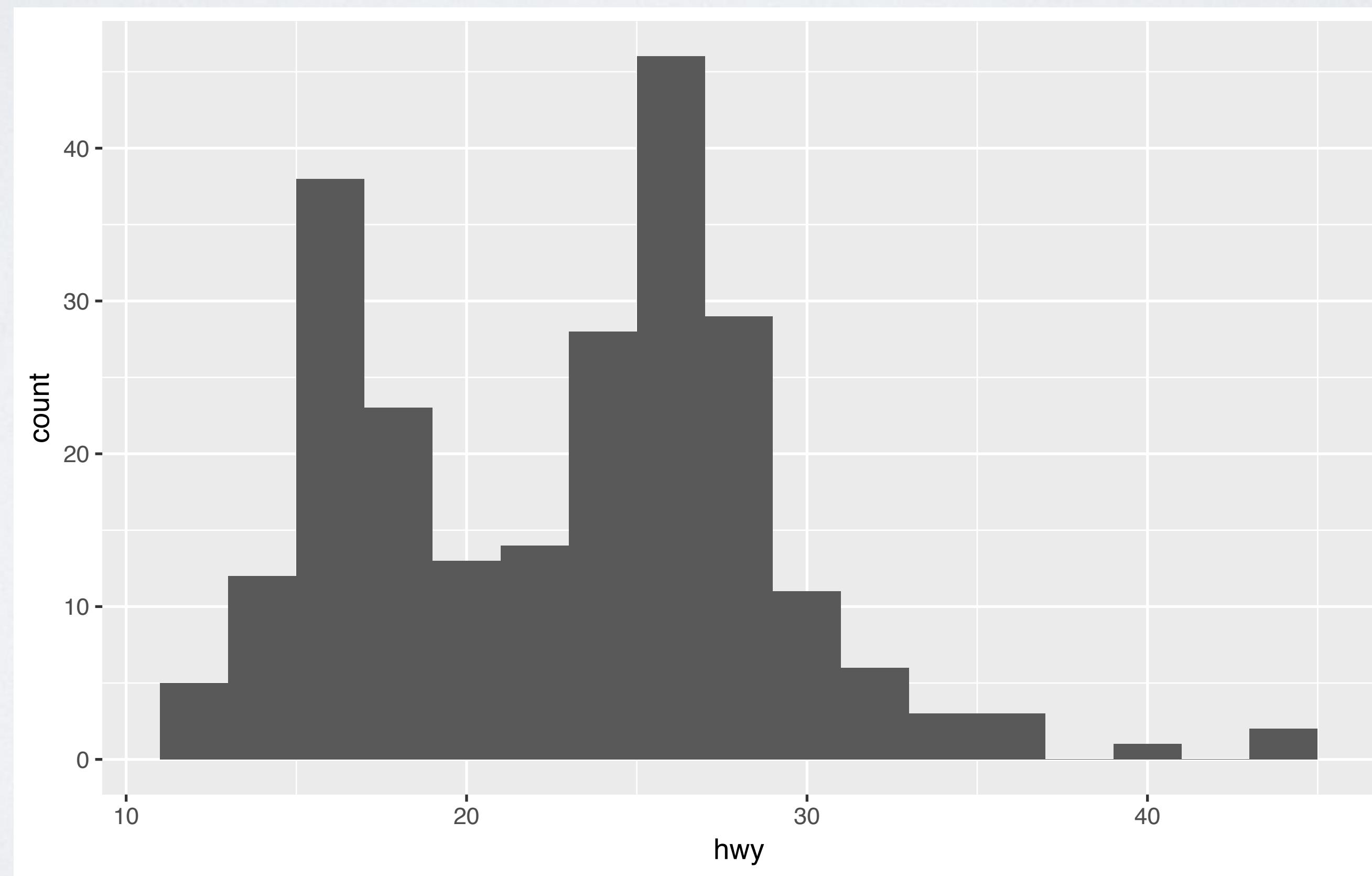


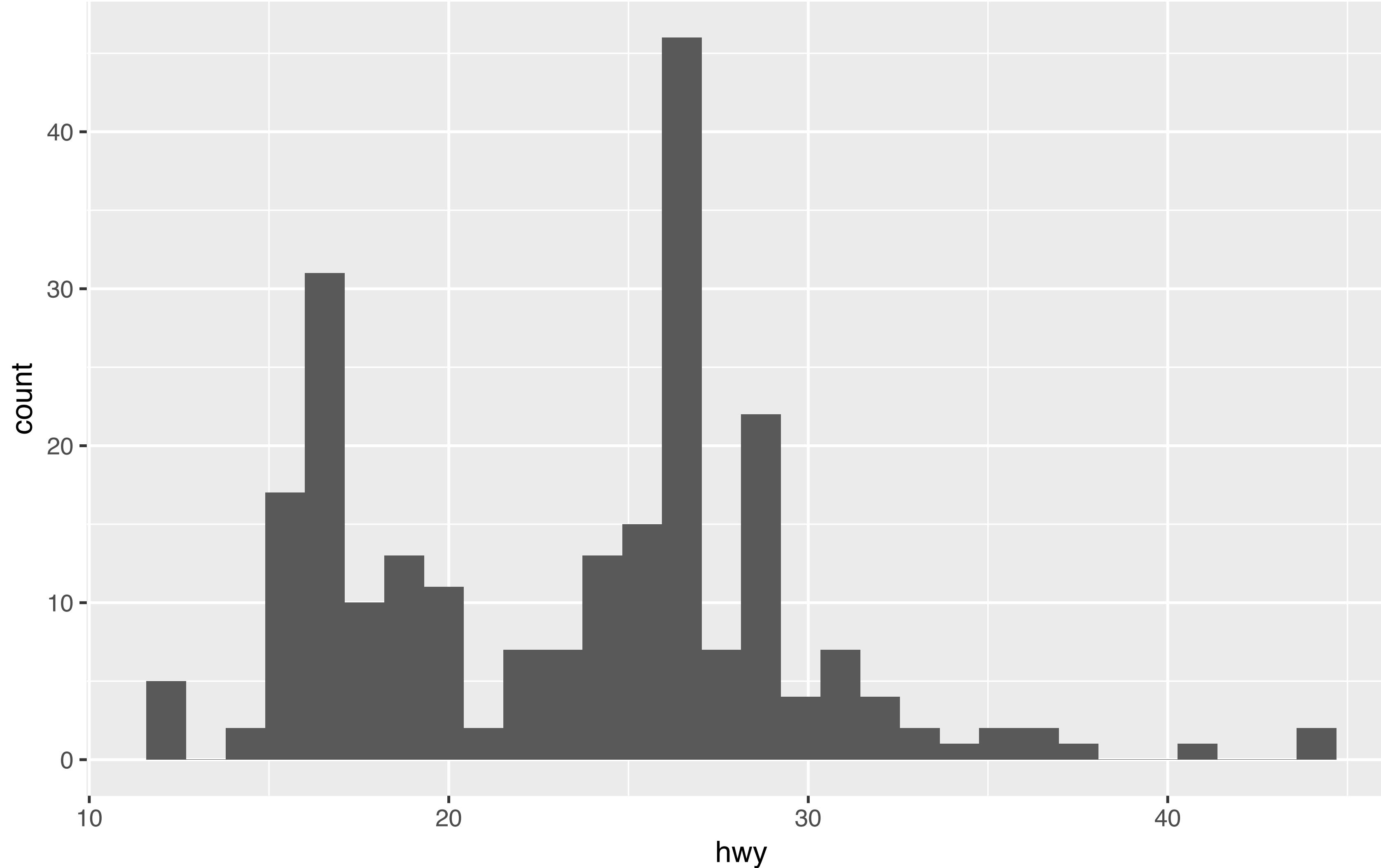
```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = class, y = hwy))
```



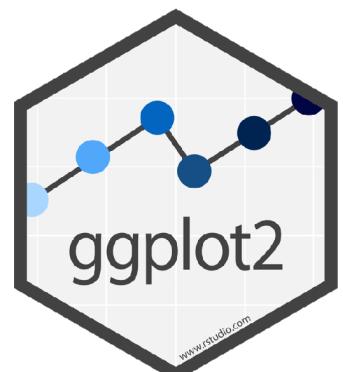
# Your Turn 4

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



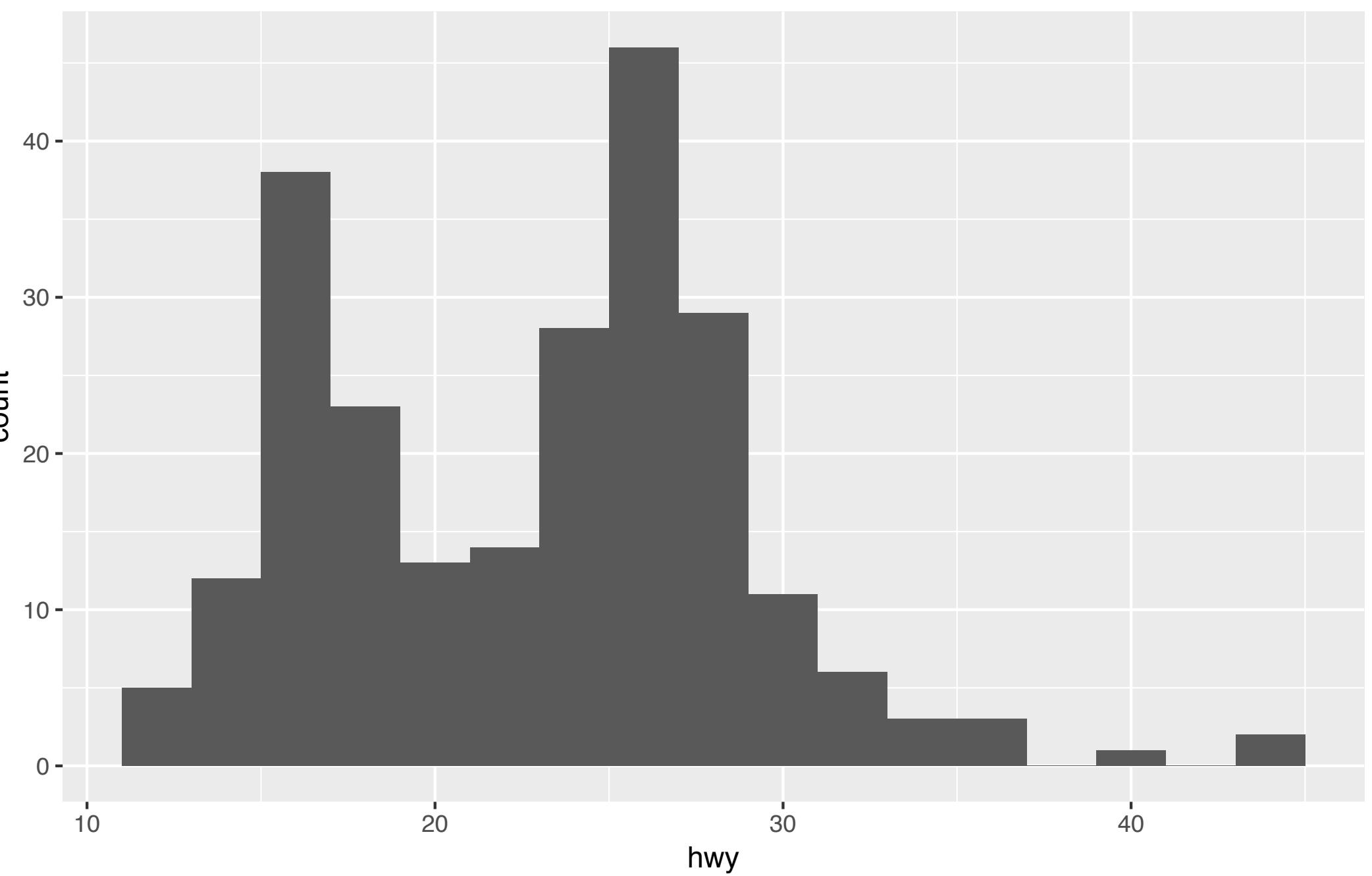
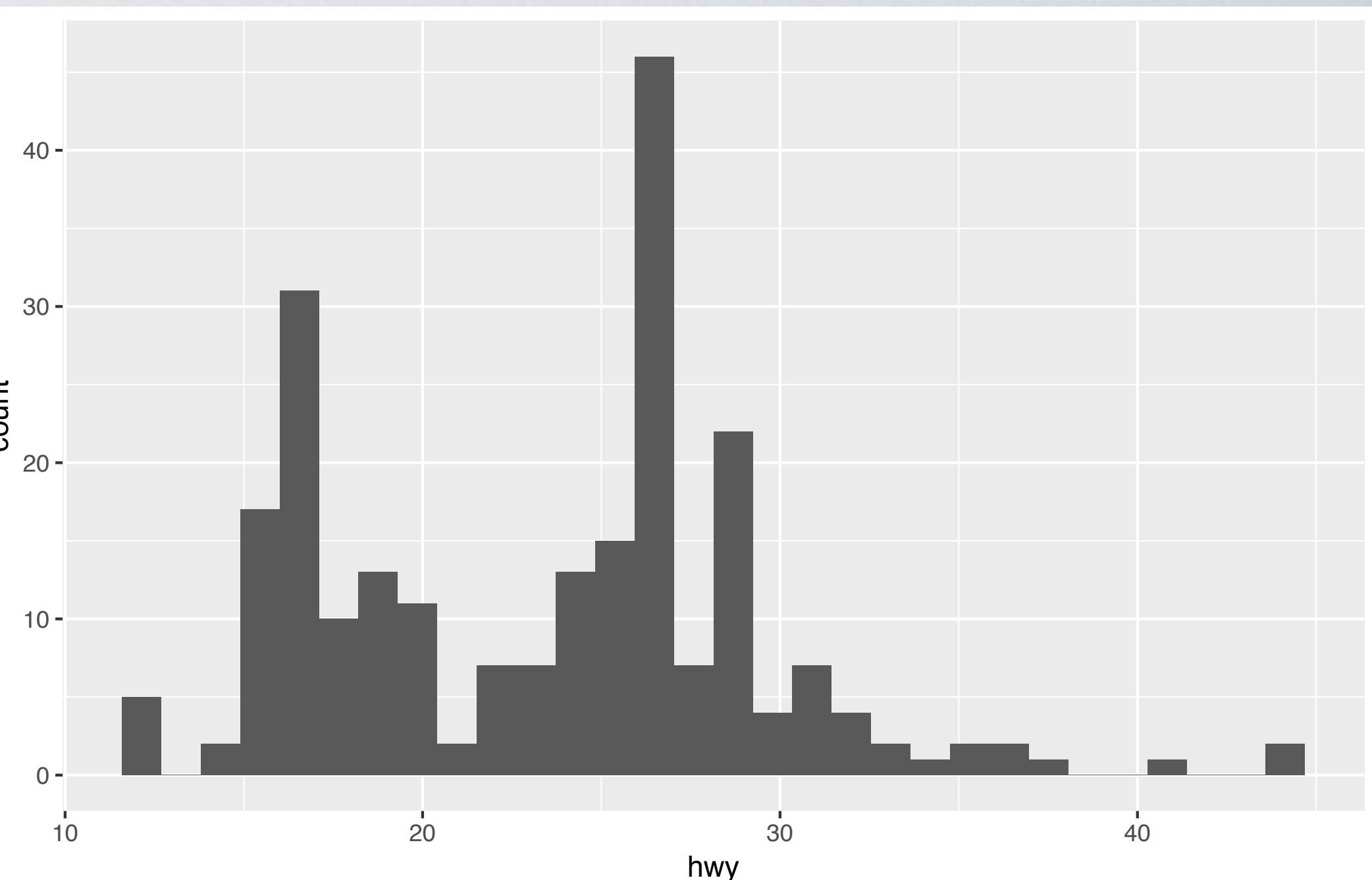


```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy))
```



# Quiz

## What is the difference?



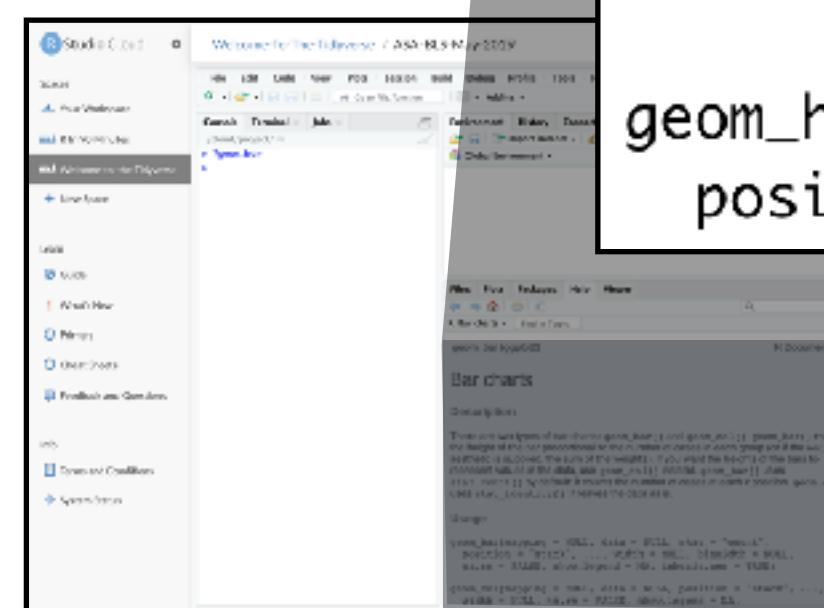
# "Help" pages

To open the documentation  
for a function, type

```
?geom_histogram
```

?

function name (no  
parentheses)



geom\_freqpoly {ggplot2}

R Documentation

## Histograms and frequency polygons

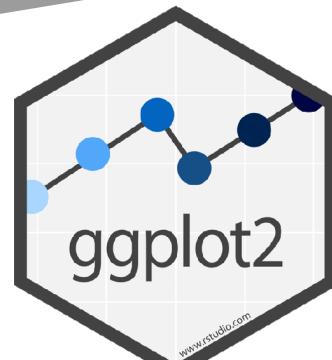
### Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

### Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",  
  position = "identity", ..., na.rm = FALSE, show.legend = NA,  
  inherit.aes = TRUE)
```

```
geom_histogram(mapping = NULL, data = NULL, stat = "bin",  
  position = "stack", ..., binwidth = NULL, bins = NULL,
```



# Tips

- **scan** page for relevant info
- **ignore** things that don't make sense
- **try out** the examples

geom\_freqpoly {ggplot2} R Documentation

## Histograms and frequency polygons

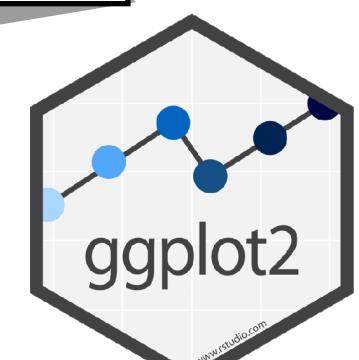
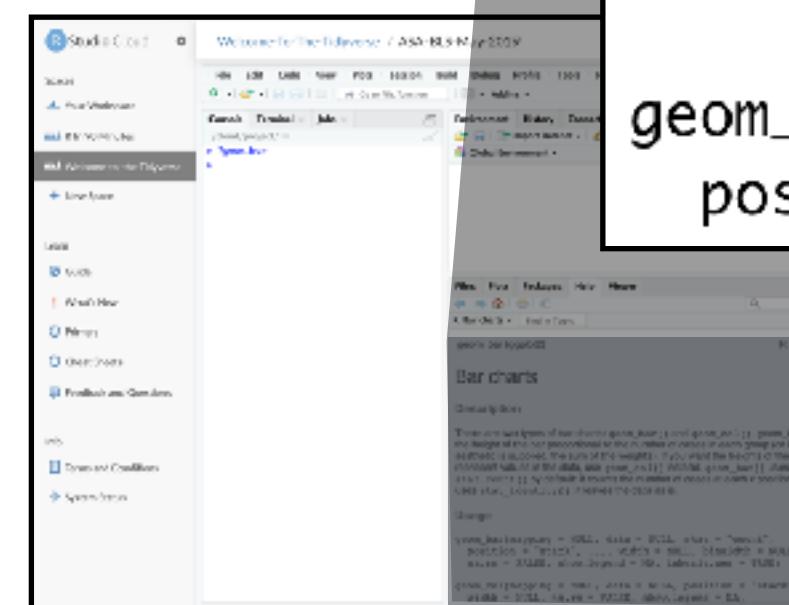
### Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

### Usage

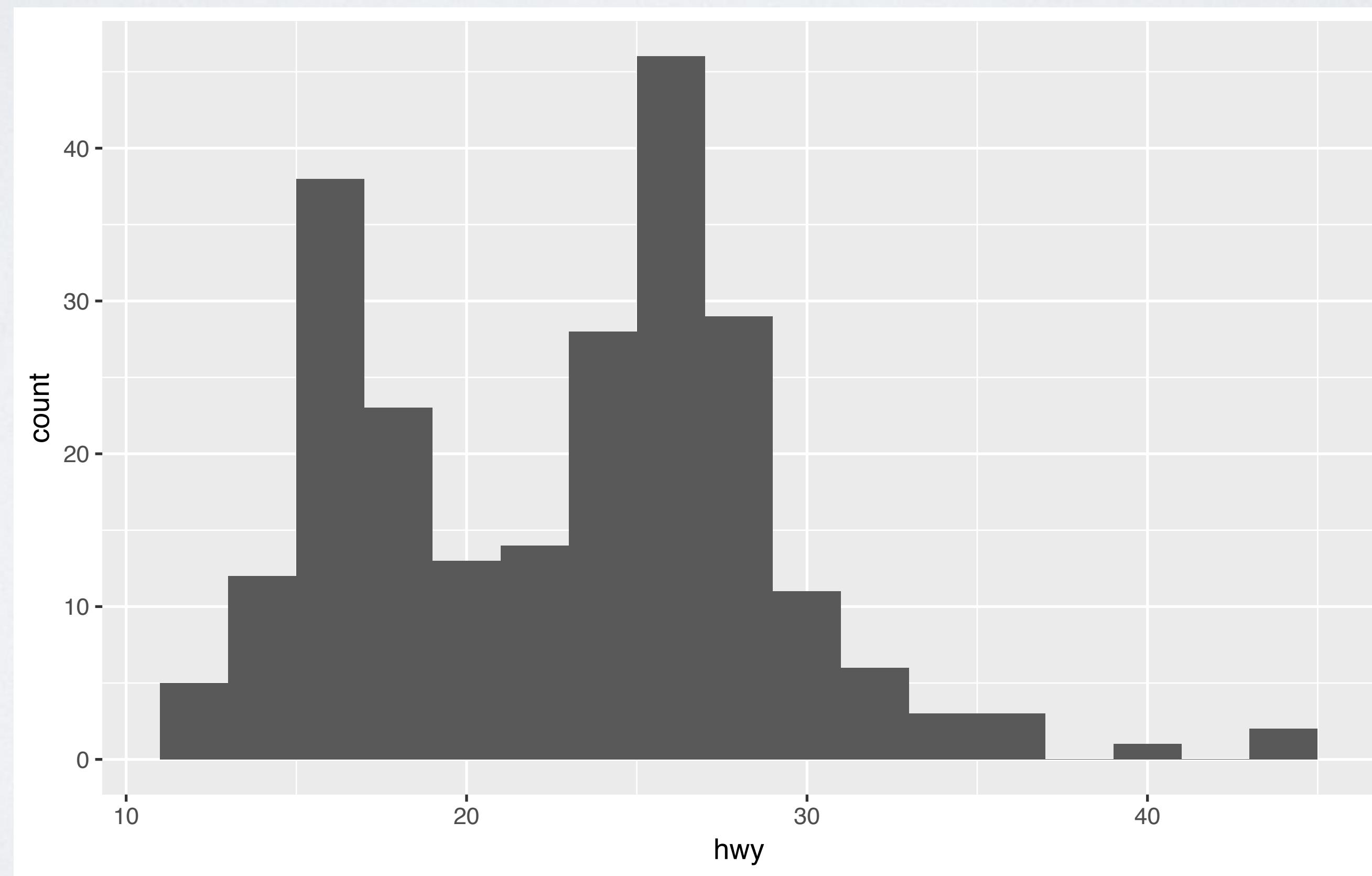
```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)
```

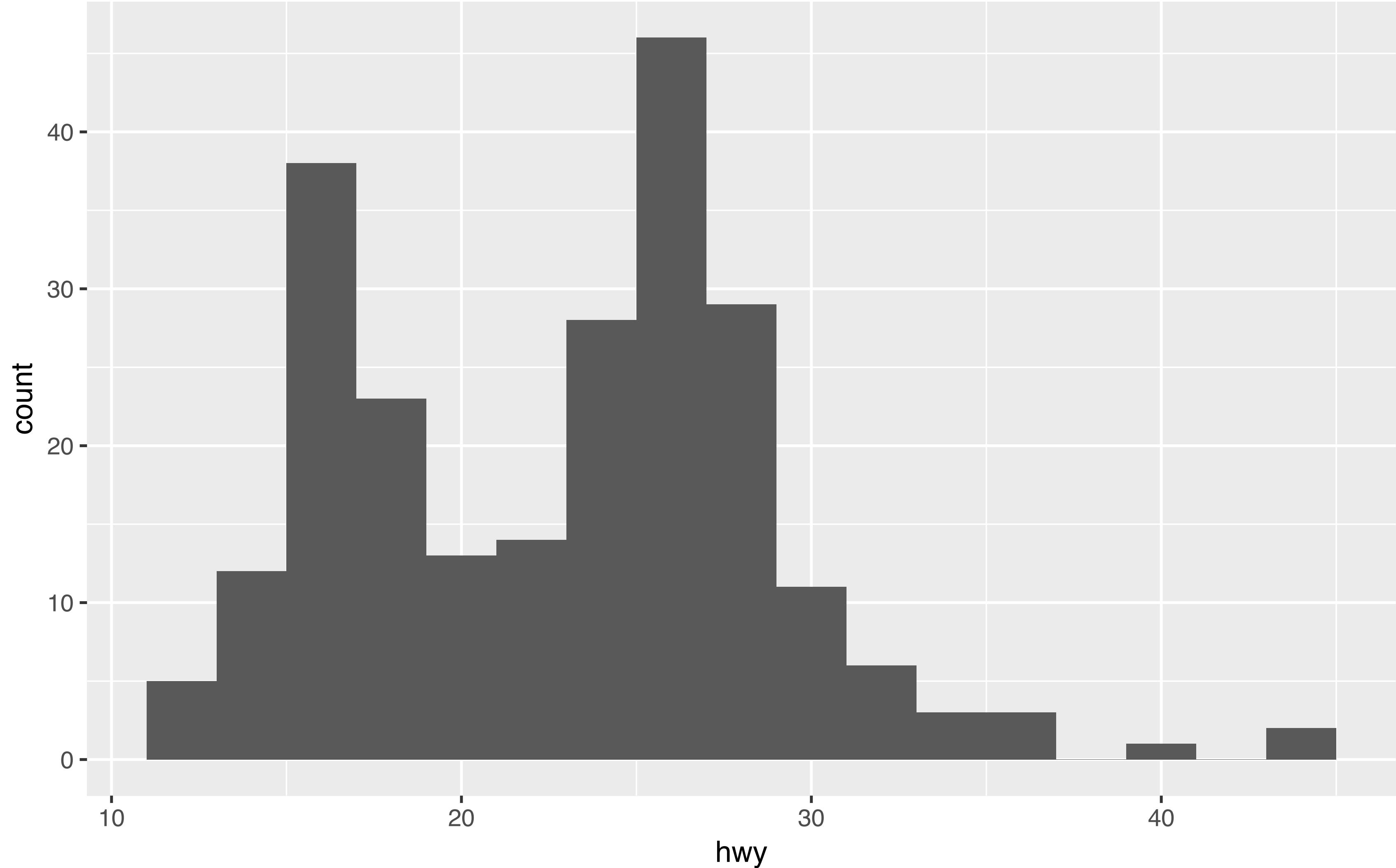
```
geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```



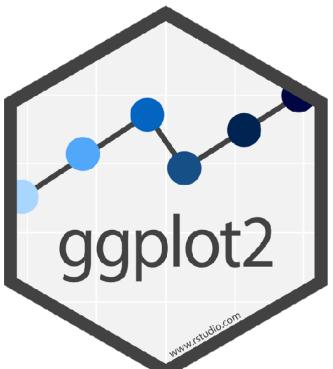
# Your Turn 5

Use the help page for `geom_histogram`  
to make the bins 2 mpg wide.





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



# ggplot2.tidyverse.org

The screenshot shows a web browser window with the title "Create Elegant Data Visualisation" and the user "Garrett". The address bar contains "ggplot2.tidyverse.org". The page itself is the ggplot2 homepage, featuring the ggplot2 logo and the text "part of the tidyverse". It includes sections for "Usage", "Links", "License", and "Developers". A code snippet is shown in a box:

```
library(ggplot2)

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

A scatter plot is displayed below the code, showing the relationship between engine displacement (displ) and fuel economy (hwy) for different car classes. The legend indicates that red dots represent "2seater" vehicles.

**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()`).

**Links**

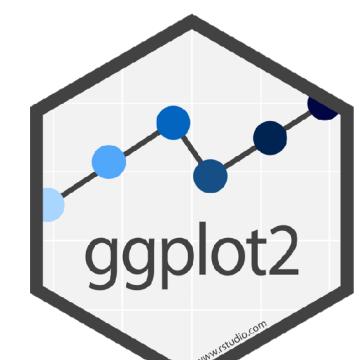
- Download from CRAN at <https://cran.r-project.org/package=ggplot2>
- Browse source code at <https://github.com/tidyverse/ggplot2>
- Report a bug at <https://github.com/tidyverse/ggplot2/issues>
- Learn more at <http://r4ds.had.co.nz/data-visualisation.html>

**License**

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

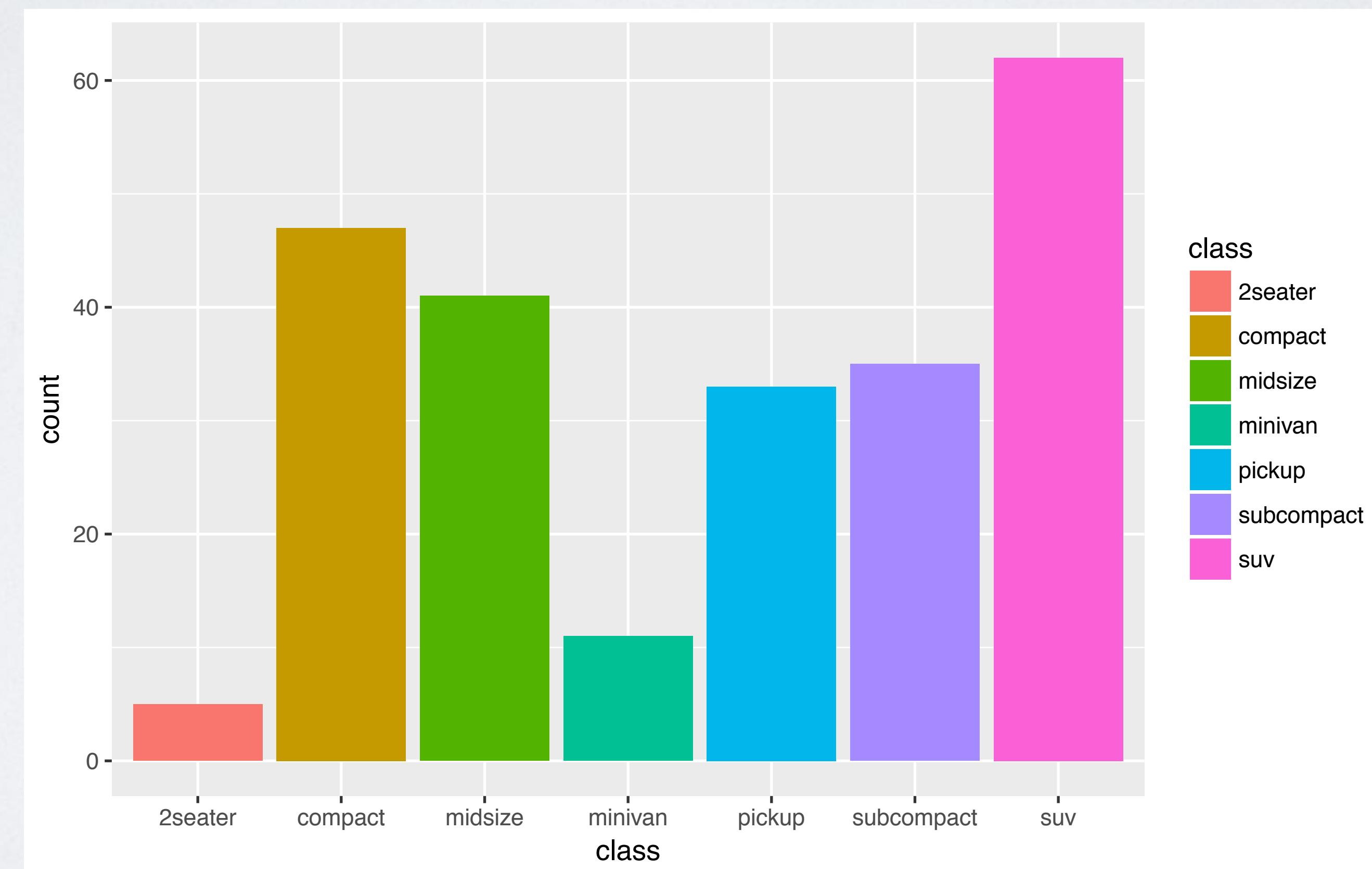
**Developers**

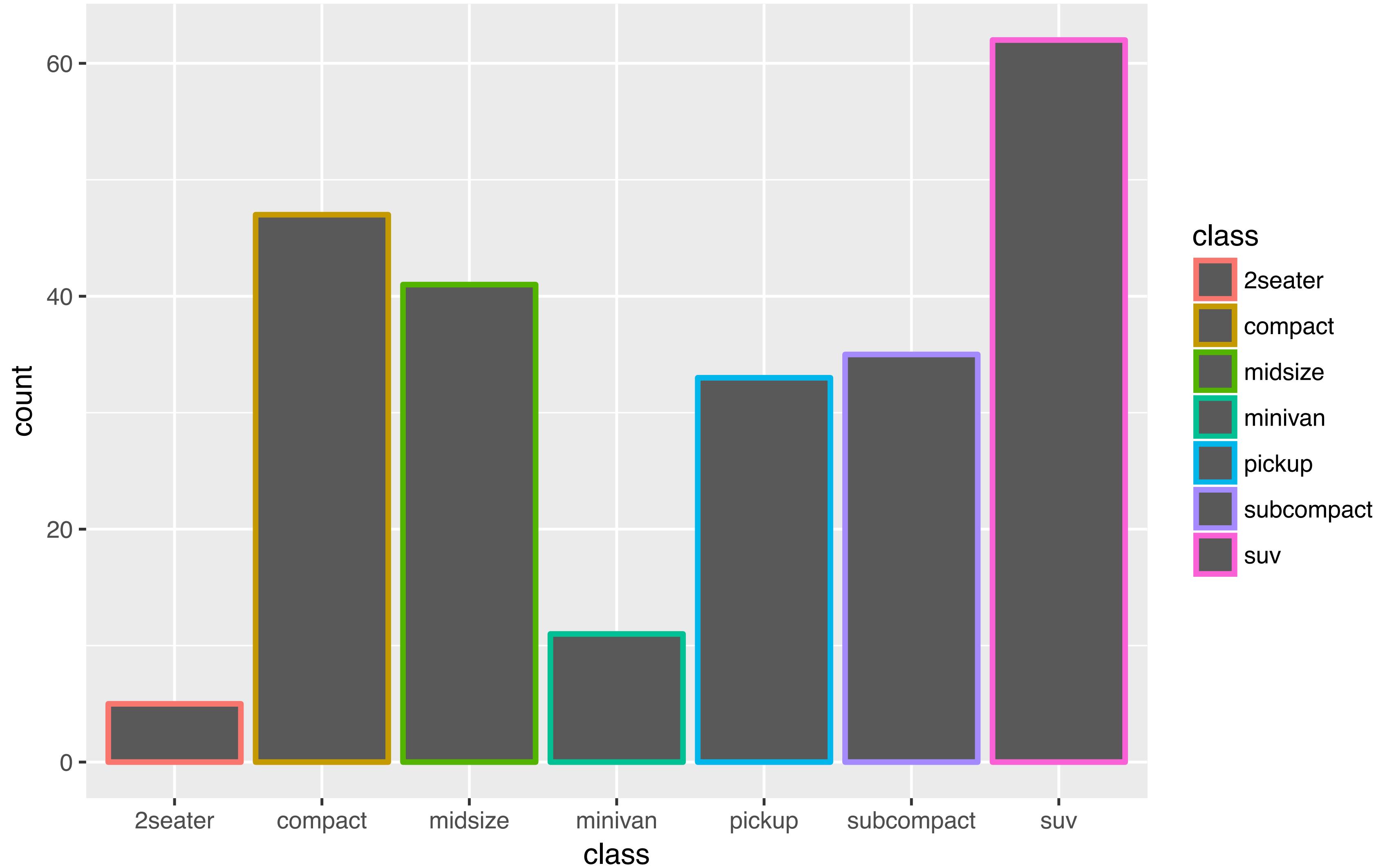
Hadley Wickham  
Author maintainer



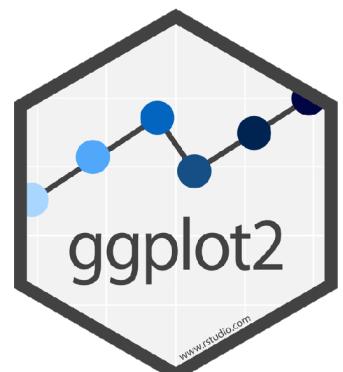
# Your Turn 6

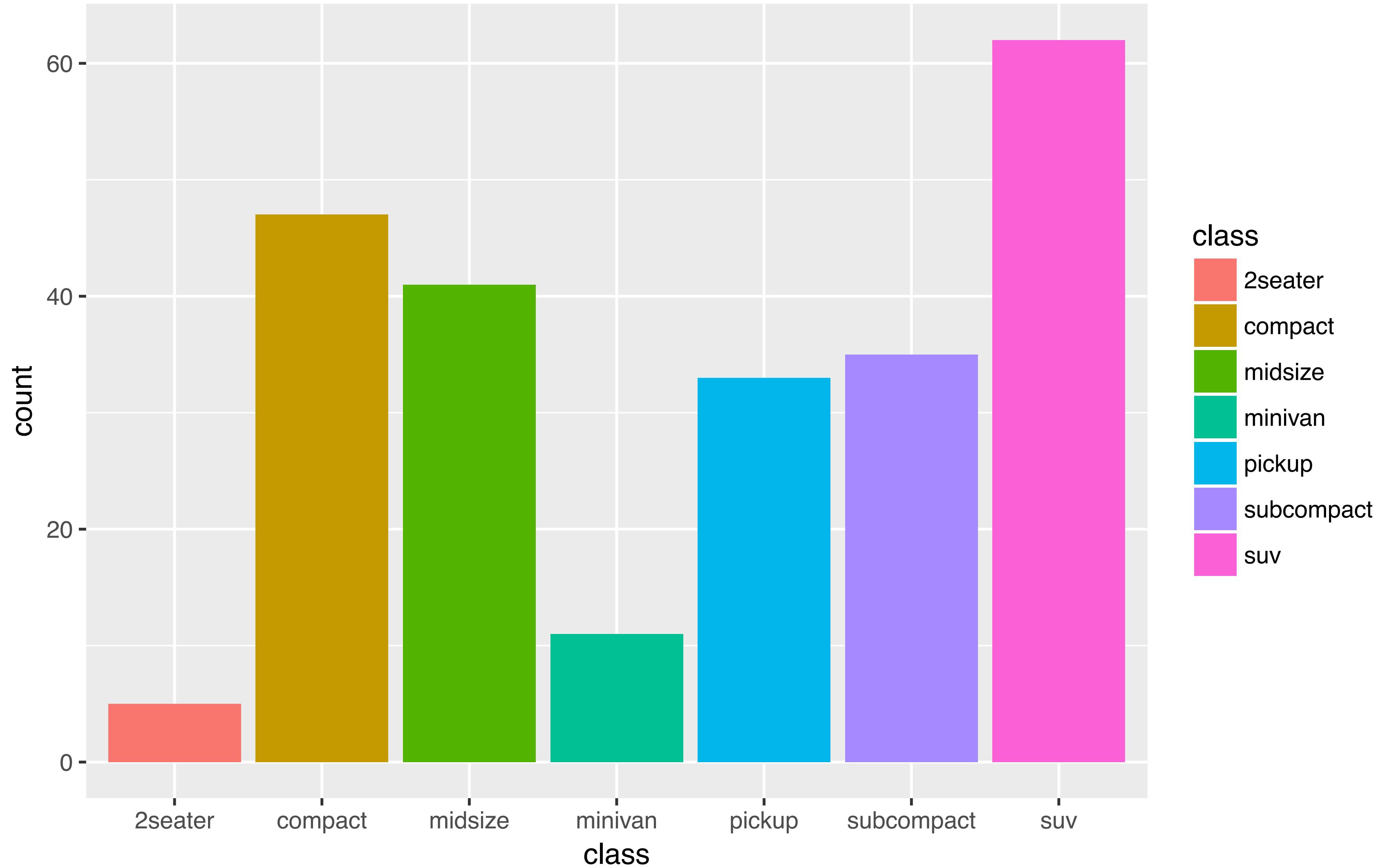
With a partner, make the bar chart of **class** below. Use the cheatsheet. Hint: do not supply a **y** variable.



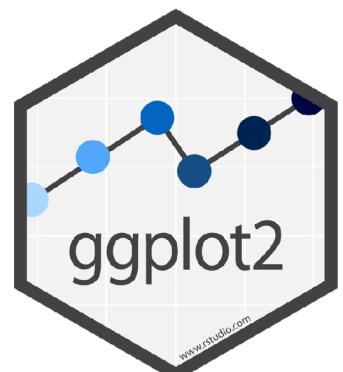


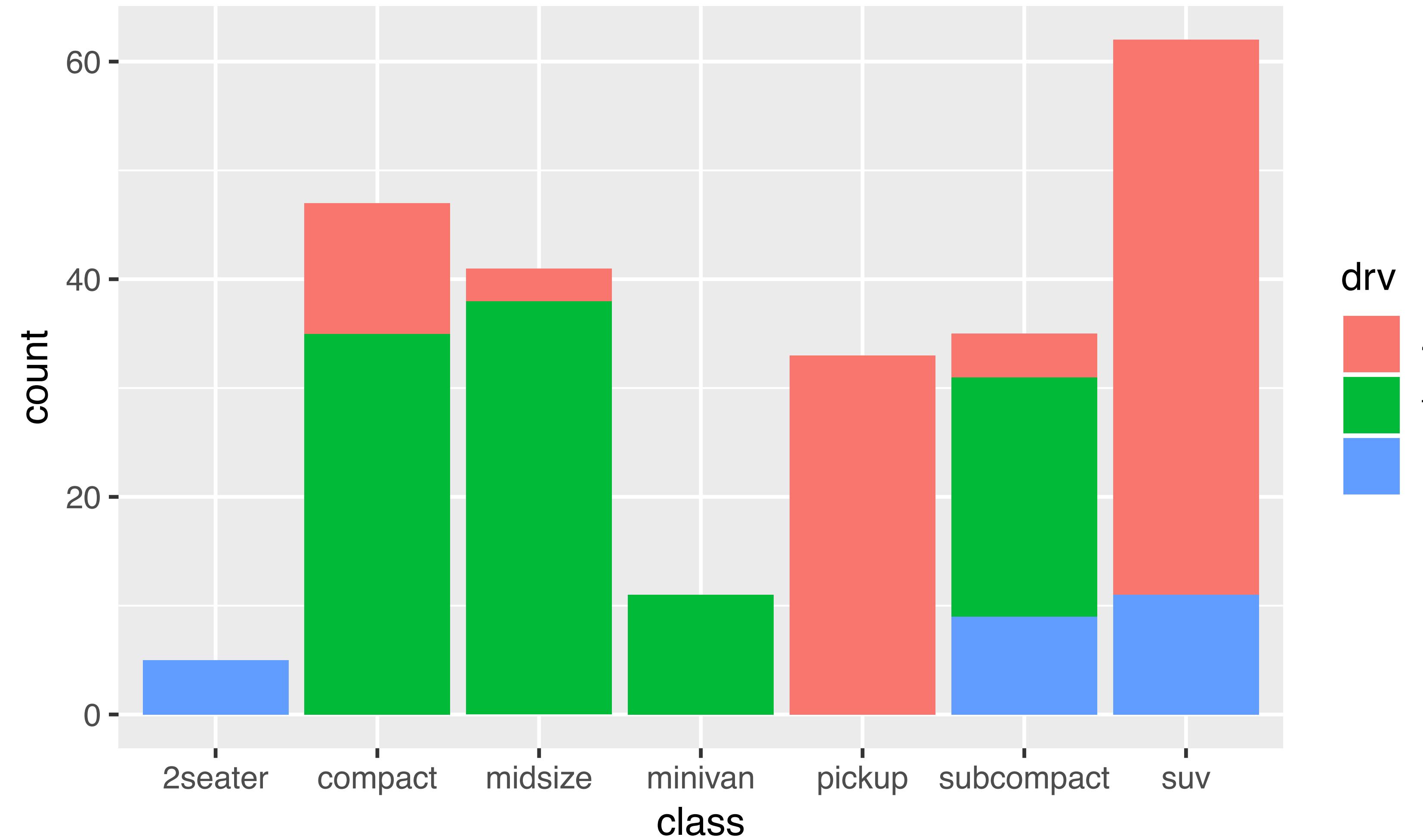
```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = class, color = class))
```



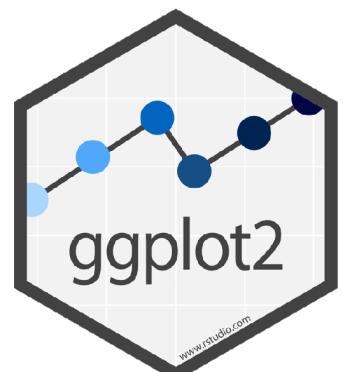


```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = class, fill = class))
```





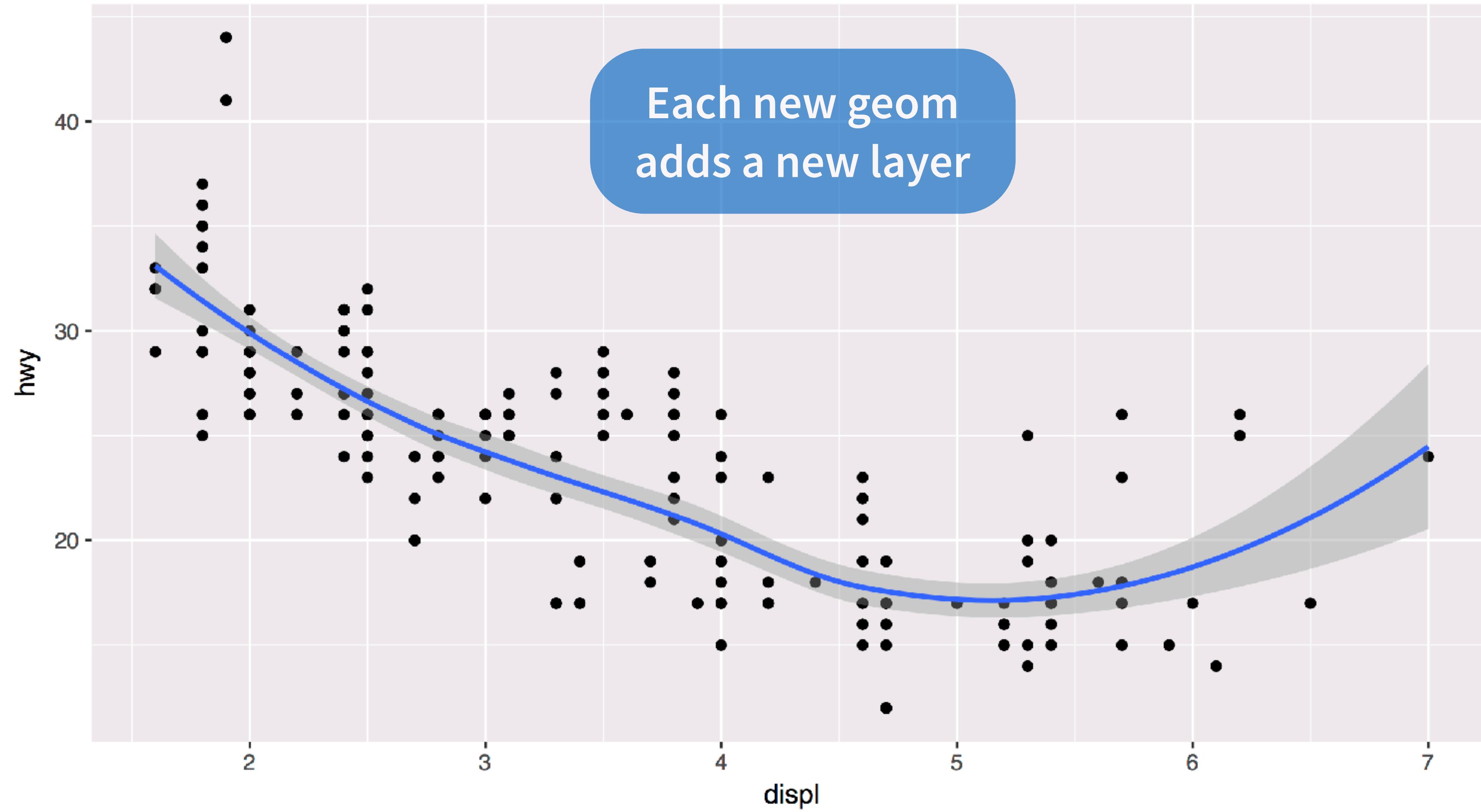
```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = class, fill = drv))
```



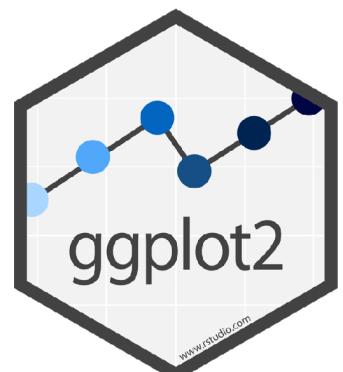
# Quiz

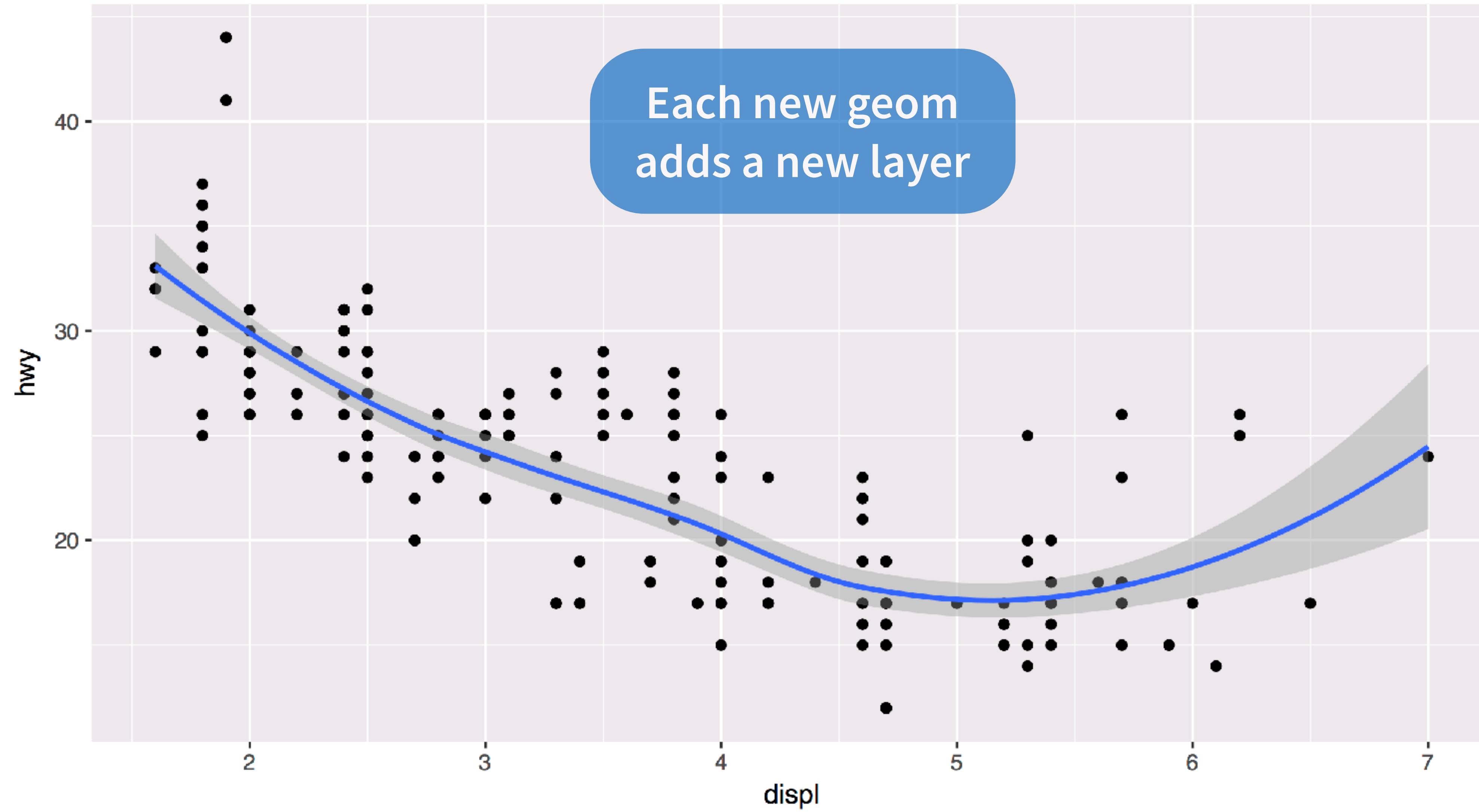
What will this code do?

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

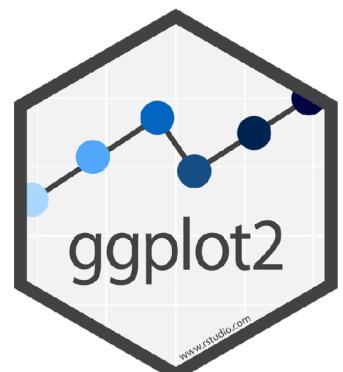


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



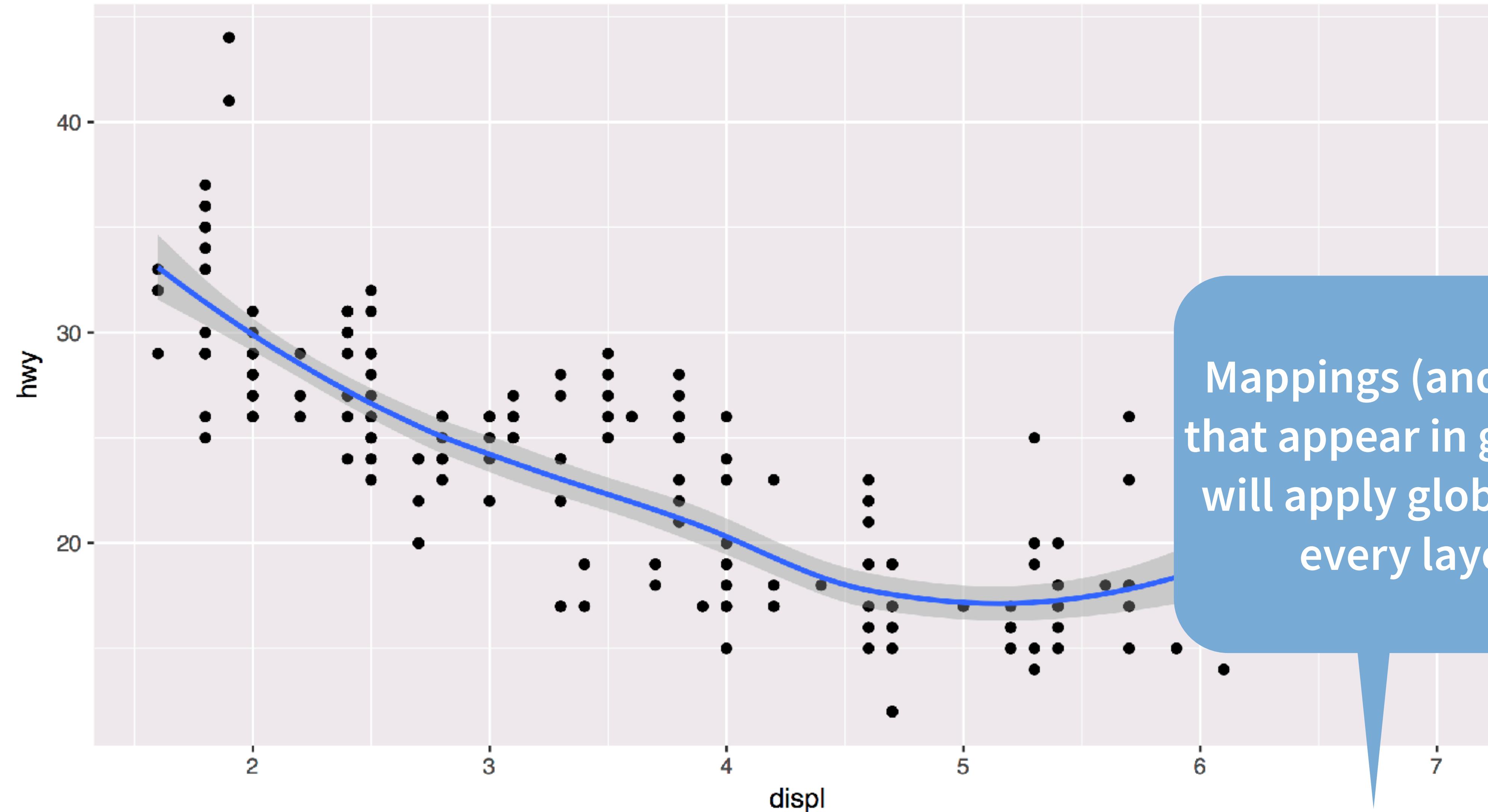


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

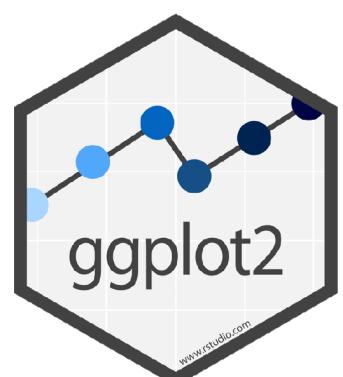


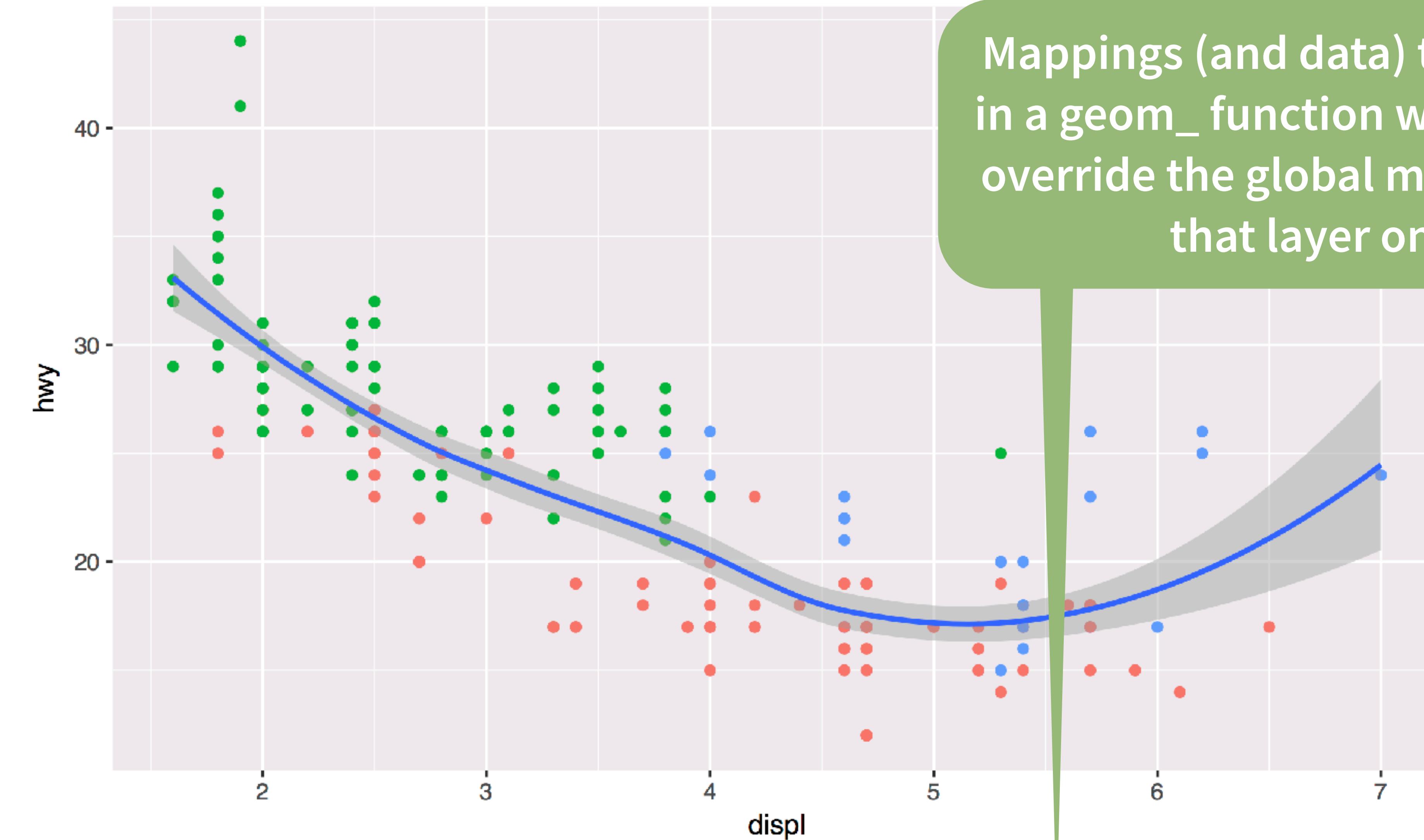
# global vs. local

R

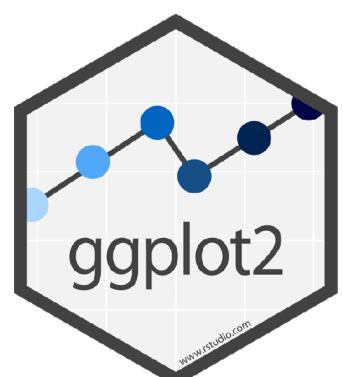


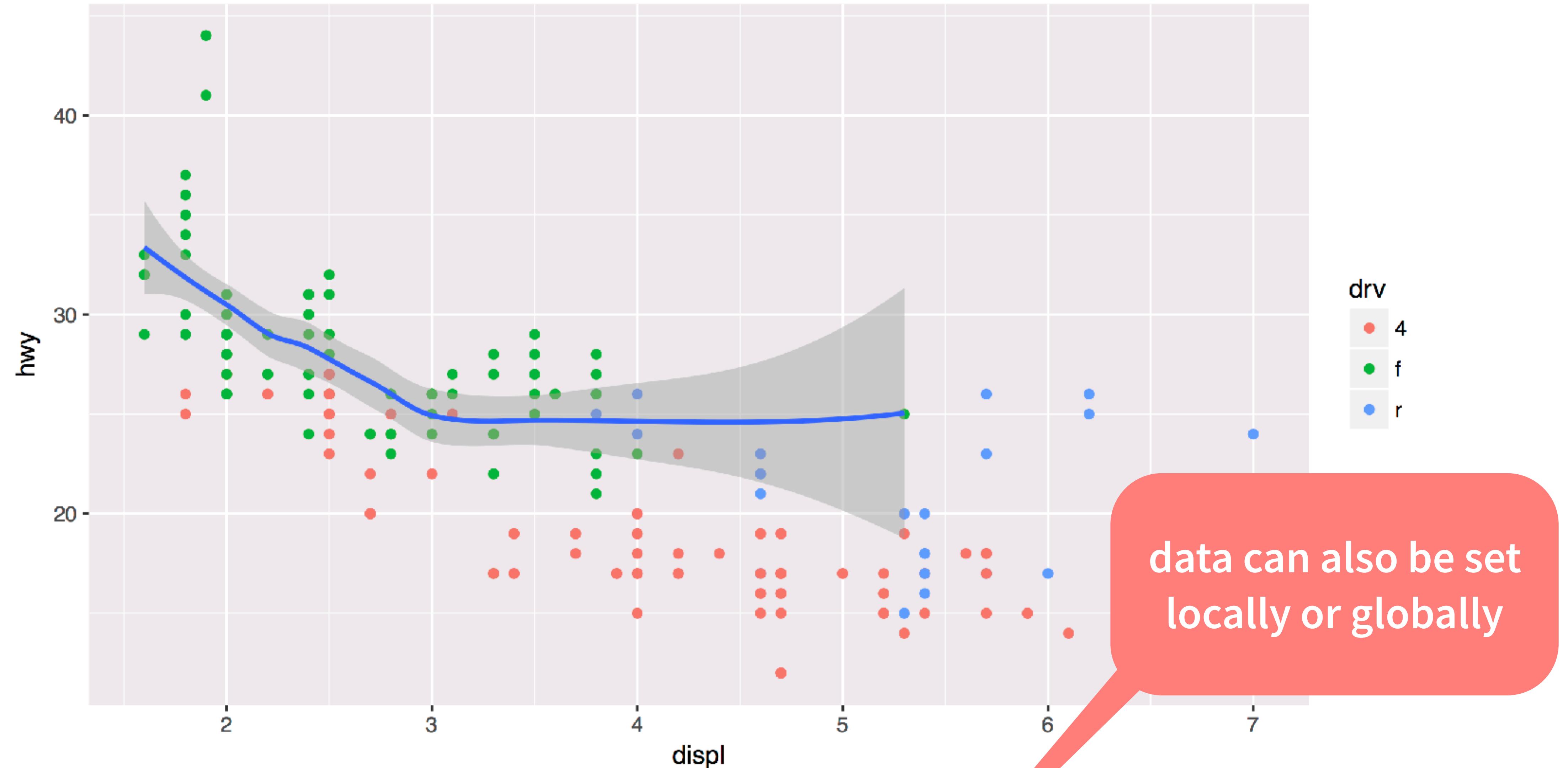
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth()
```



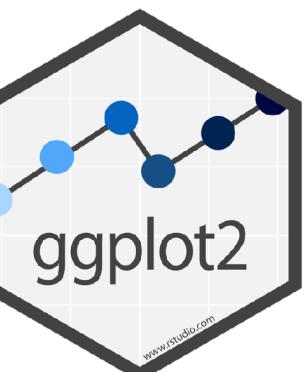


```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth()
```





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



# Quiz

What is different about this plot? Run the code!

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

```
library(plotly)  
ggplotly(p)
```

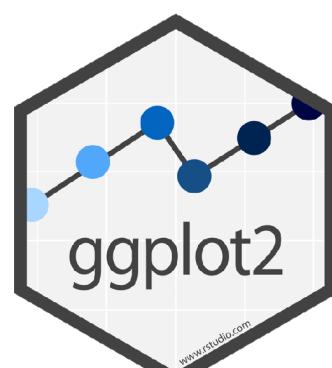
# interactivity



# Plotly

Tools for making interactive plots. [plot.ly/ggplot2/](https://plot.ly/ggplot2/)

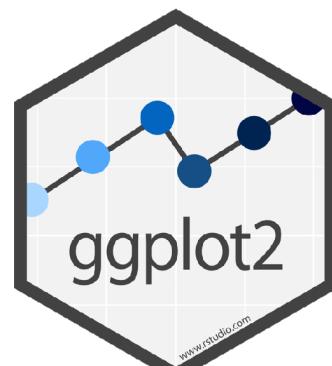
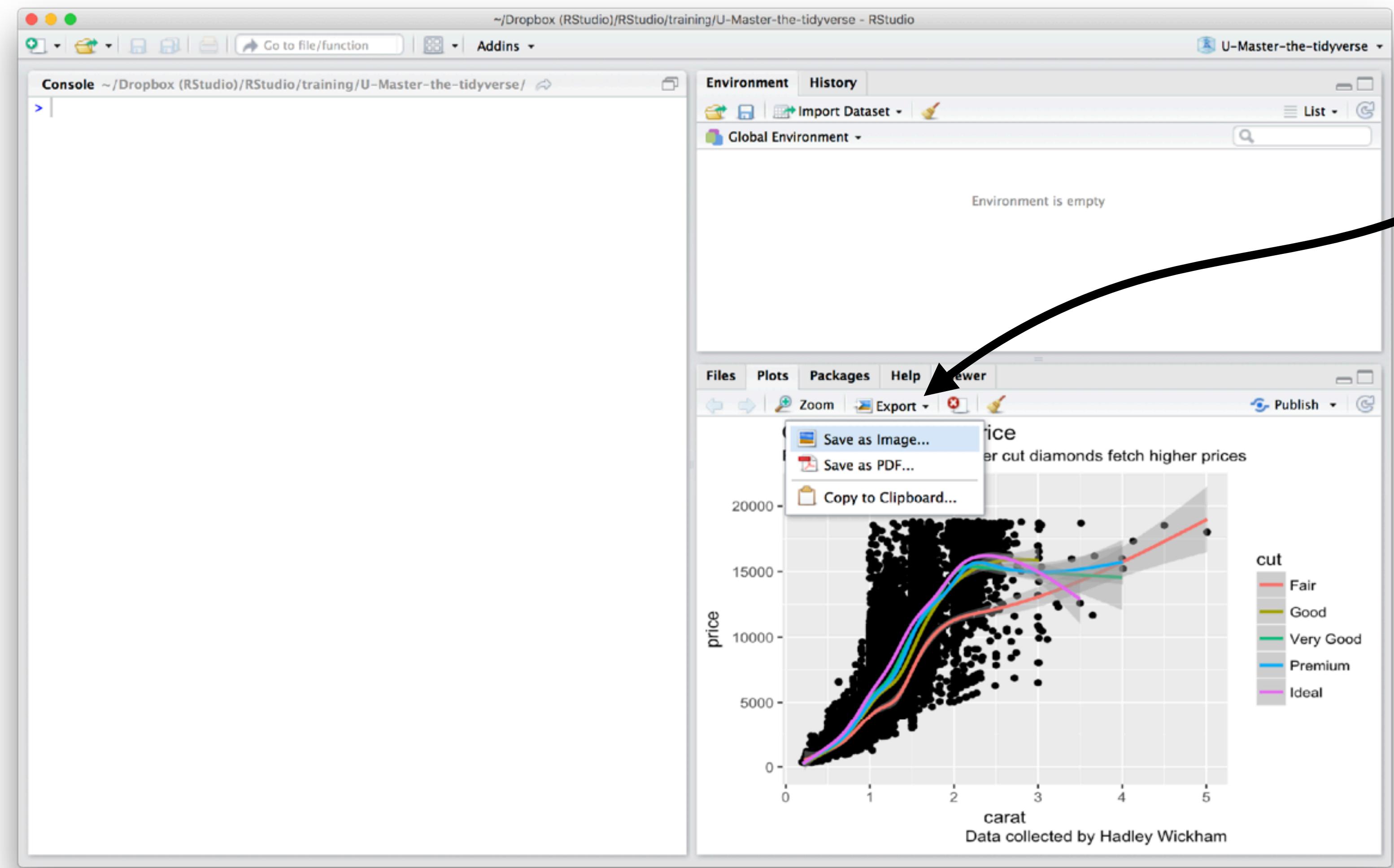
The screenshot shows a web browser window displaying the 'ggplot2 Graphing Library | Plotly' page at [plot.ly/ggplot2/](https://plot.ly/ggplot2/). The page has a dark blue header with the Plotly logo and 'Graphing Libraries'. A 'DEMO DASH' button is visible in the top right. The main content area features a large heading 'Plotly ggplot2 Library' next to a hexagonal icon containing a line plot. Below this, a paragraph explains the library's purpose: 'Plotly for ggplot2 is an interactive, browser-based charting library built on Plotly's open source javascript graphing library, plotly.js. It works entirely locally, through the HTML widgets framework.' On the left, a sidebar menu lists categories: 'Quick Start' (Getting Started, User Guide), 'Examples' (Basic, Statistical, Animations, Layout Options), 'Community' (GitHub), and 'Search' (with a search bar). At the bottom, there is a 'Basic Charts' section with a small icon.



# Saving graphs

# Manually saving plots

Save plots manually with the export menu



# Saving plots

**ggsave()** saves the last plot.

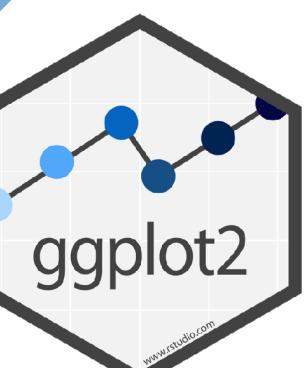
Uses size on screen:

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

Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height = 4)
```

But where will  
it save it?



# Your Turn 7

What does this command return?

getwd()



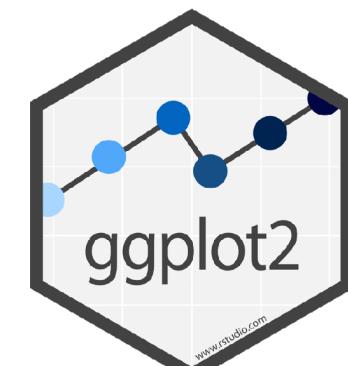
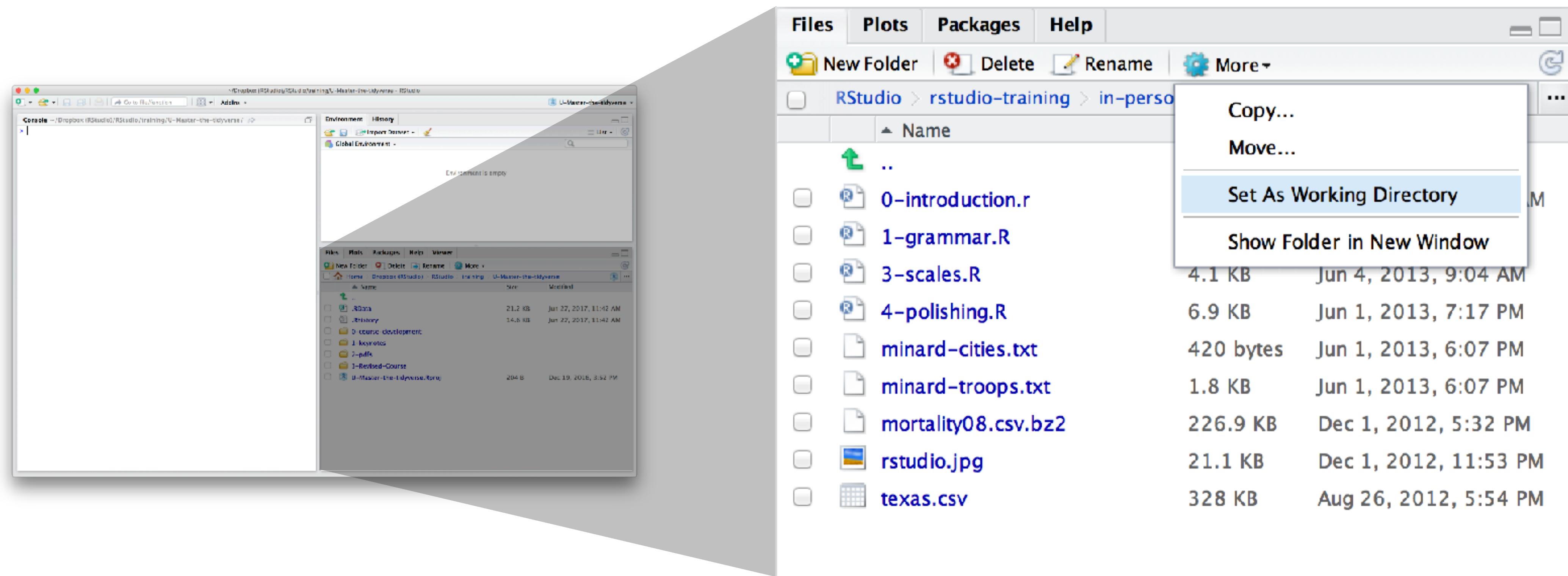
# Working directory

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

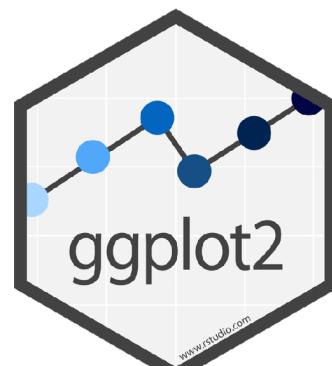
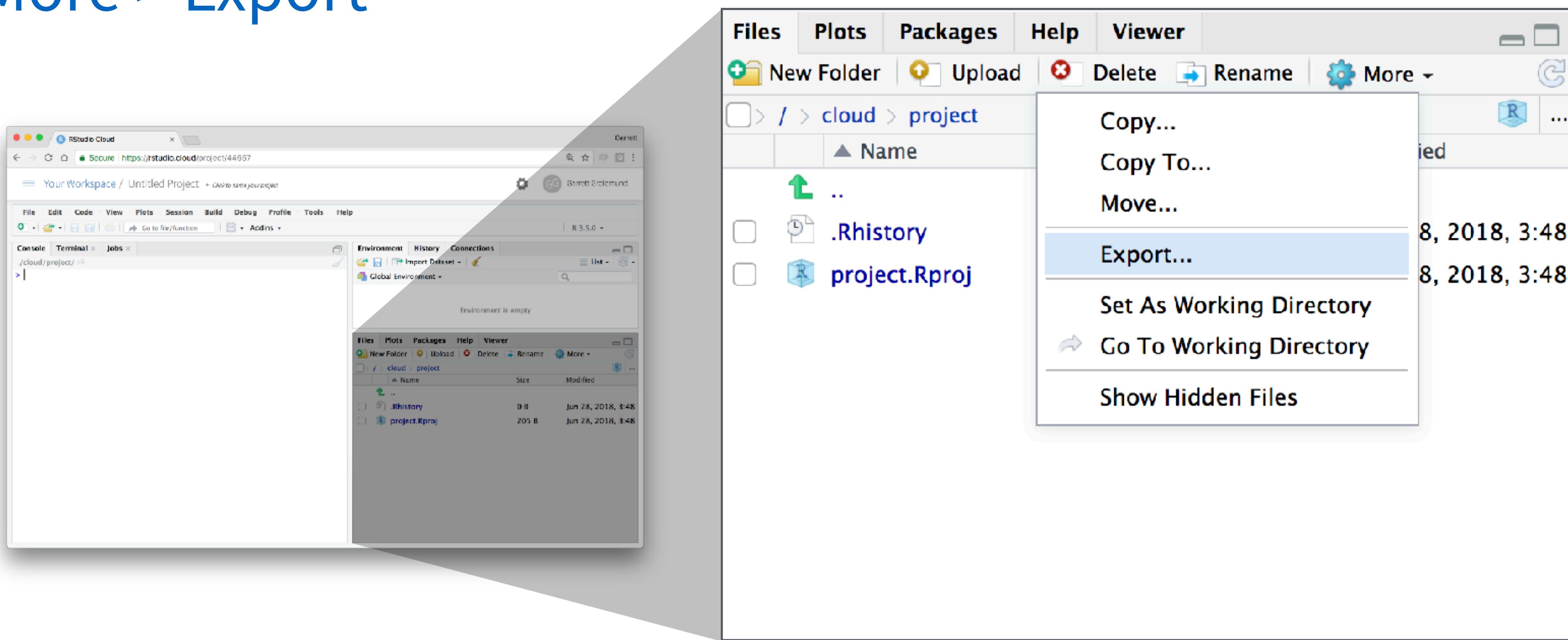
# Changing the Working directory

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



# Download files

In the files pane, check next to the file(s) to download  
More > Export



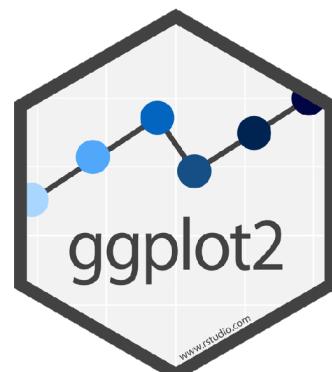
htmlwidgets::  
Save with  
htmlwidgets::saveWidget()

# Plotly

for making interactive plots. [plot.ly/ggplot2/](http://plot.ly/ggplot2/)

The screenshot shows a web browser window displaying the "ggplot2 Graphing Library | Plotly" page at [plot.ly/ggplot2/](http://plot.ly/ggplot2/). The page has a dark blue header with the Plotly logo and a "DEMO DASH" button. A navigation menu on the left includes "Help", "Open Source Graphing Libraries", "Ggplot2", "Fork on GitHub", and links for "Getting Started", "User Guide", "Examples", "Basic", "Statistical", "Animations", "Layout Options", "Community", and "GitHub". The main content area features a large hexagonal logo for "ggplot2" with a line chart and bar chart. The text "Plotly ggplot2 Library" is displayed, followed by a description: "Plotly for ggplot2 is an interactive, browser-based charting library built on Plotly's open source javascript graphing library, plotly.js. It works entirely locally, through the HTML widgets framework." Below this is a "Search" section with a search bar and a "Basic Charts" section.

ggplot2

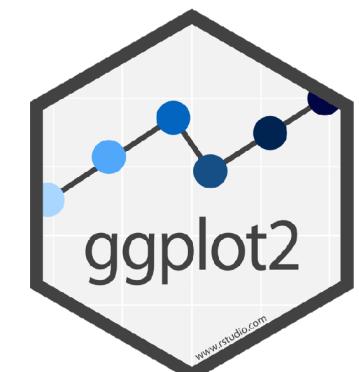
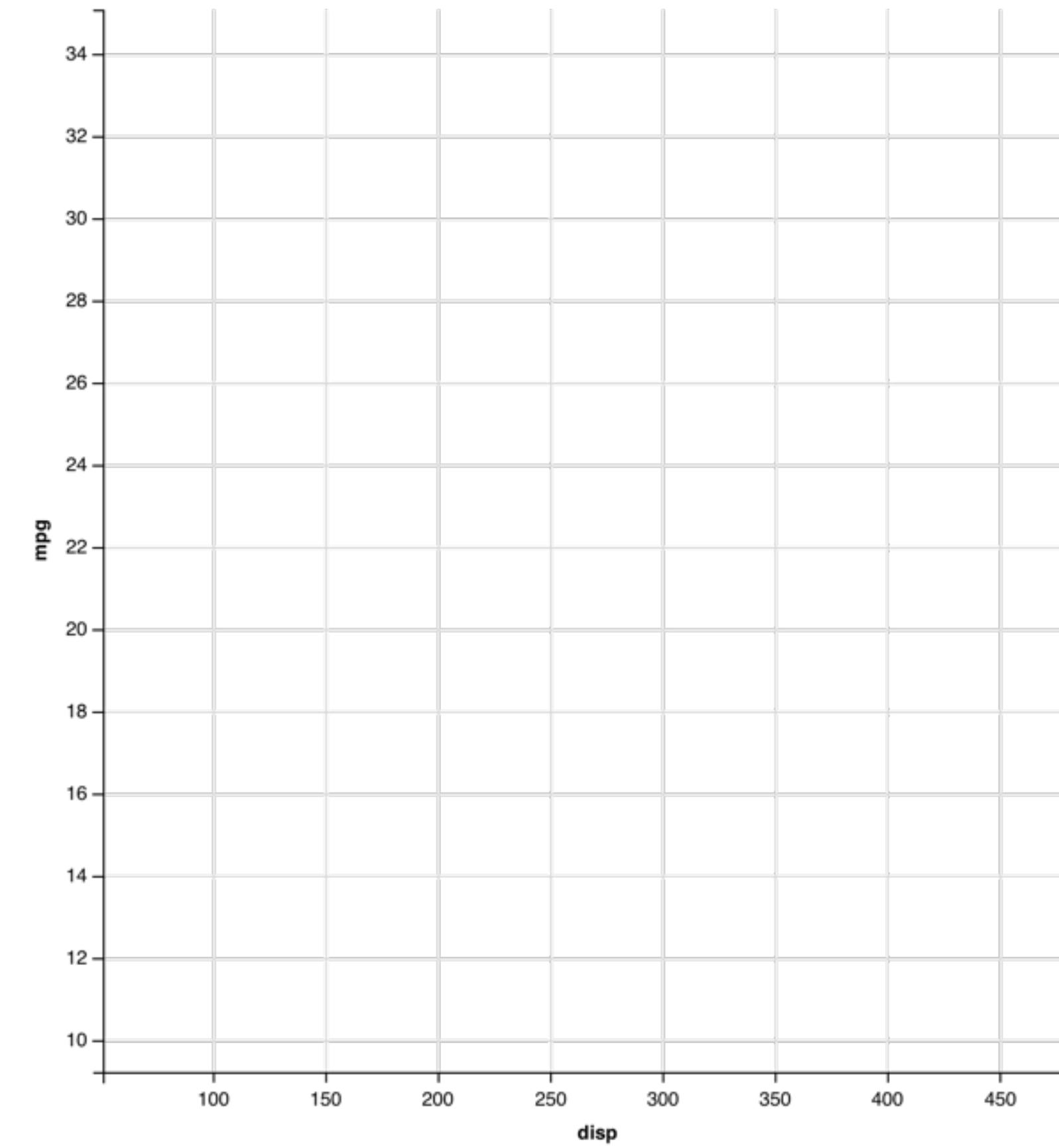


# Grammar of Graphics

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



# mappings

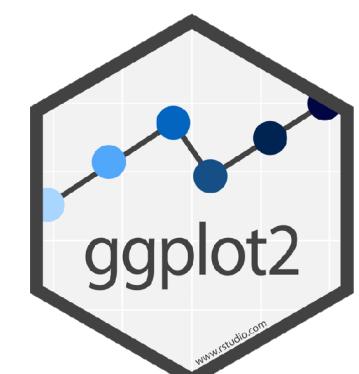
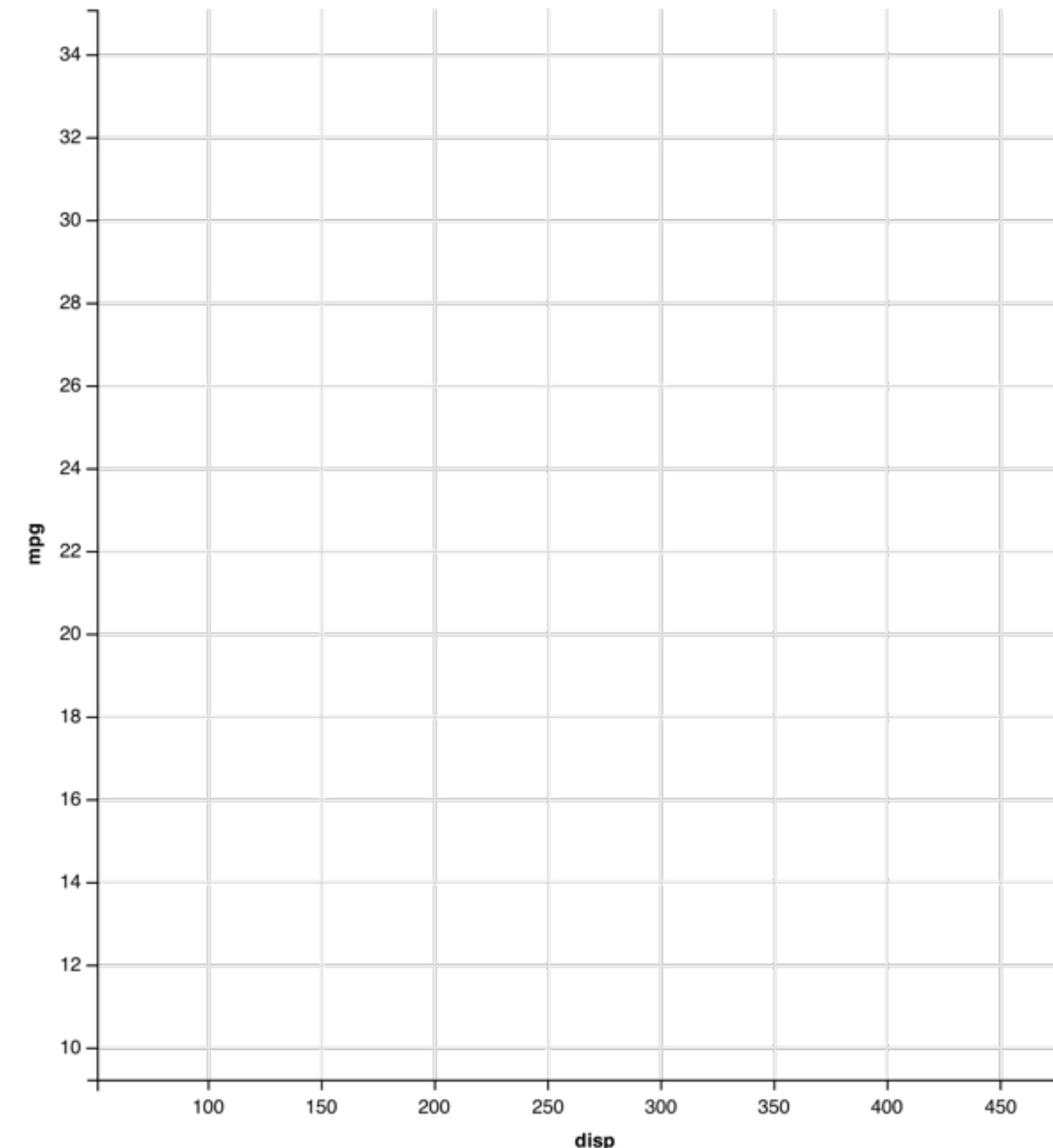
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

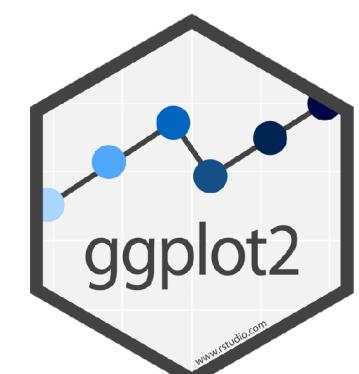
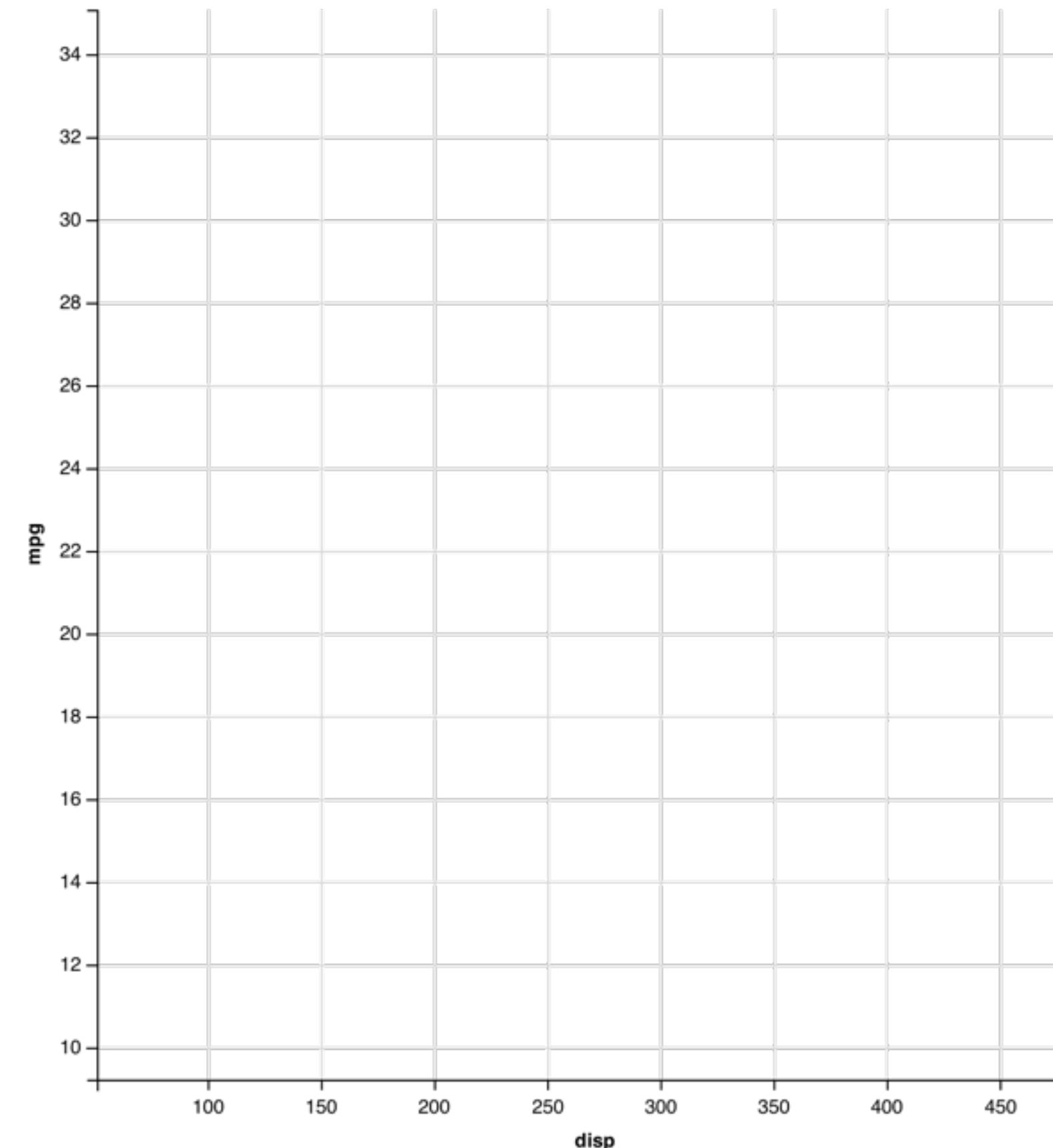


# mappings

shape		fill	
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

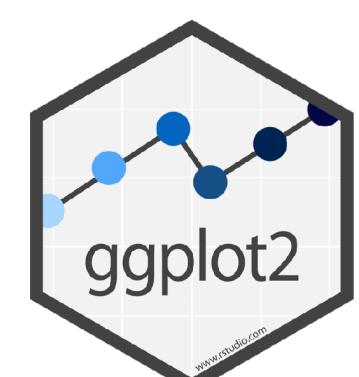
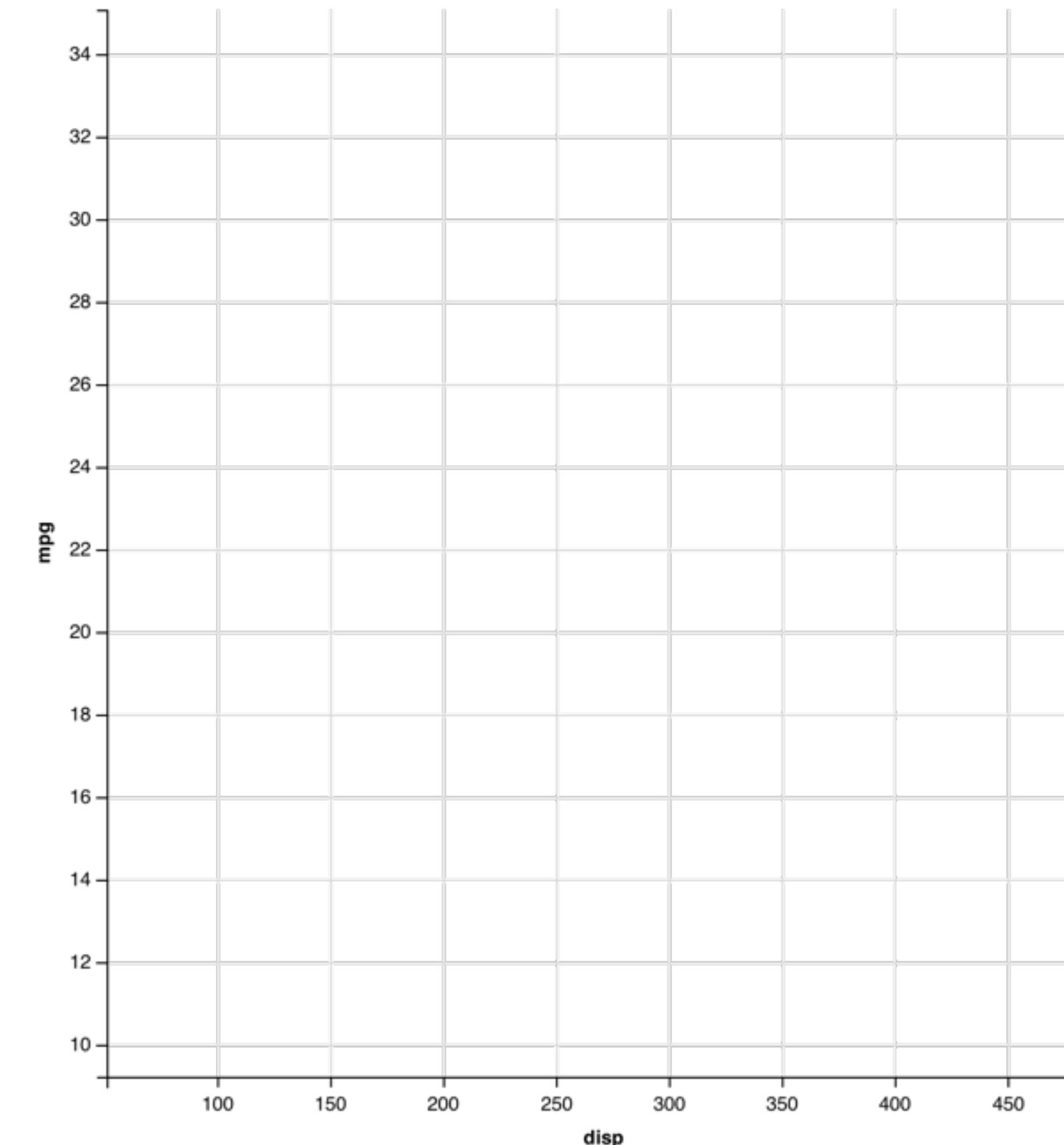


# mappings

	shape	x	fill
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

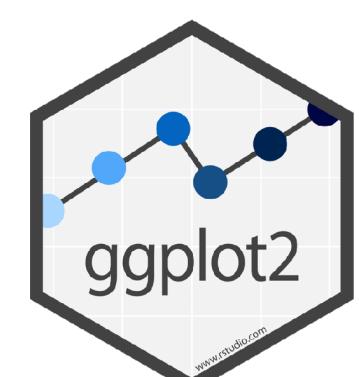
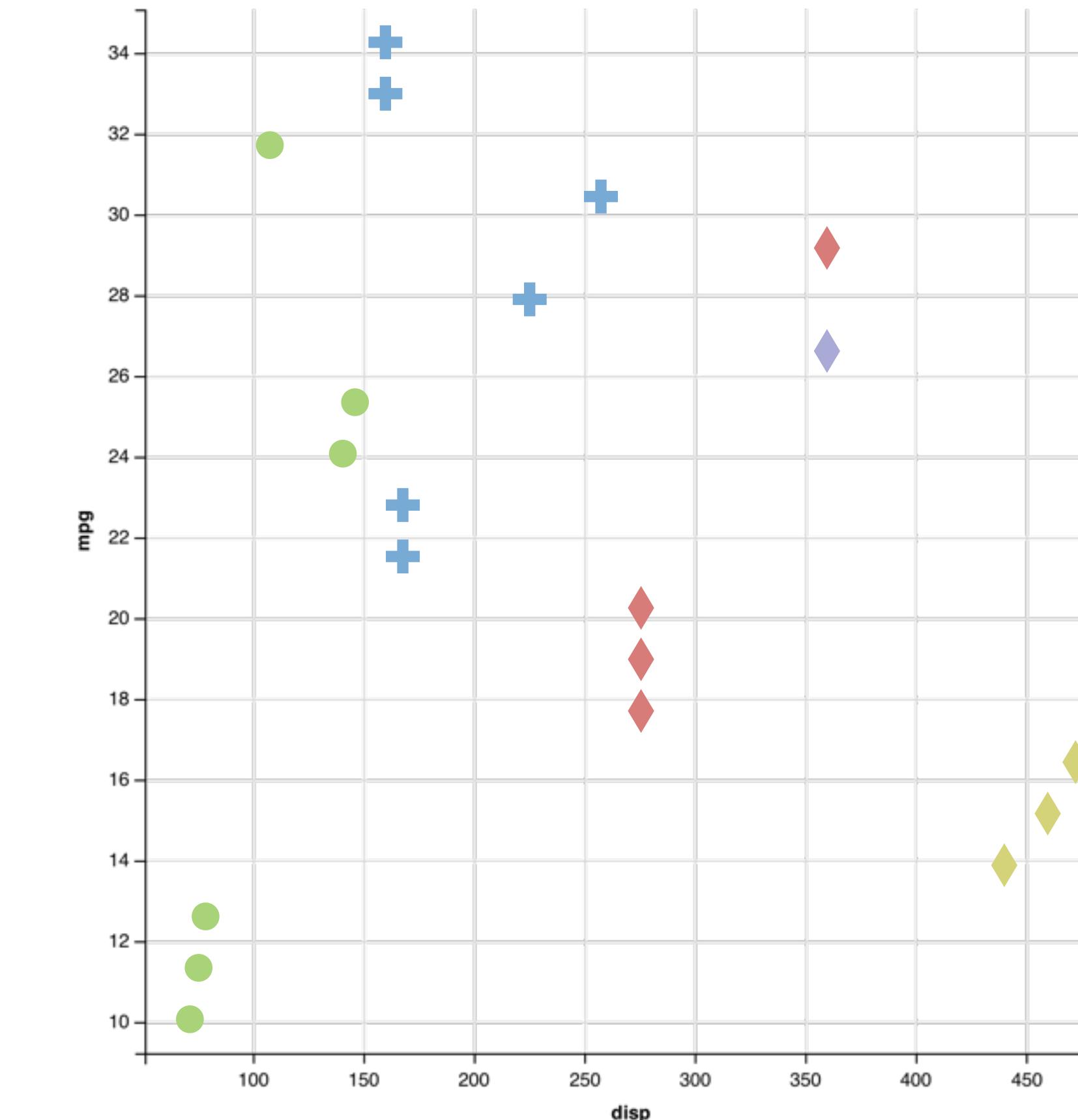


# mappings

	y ↔	shape ↔	x ↔	fill ↔
	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

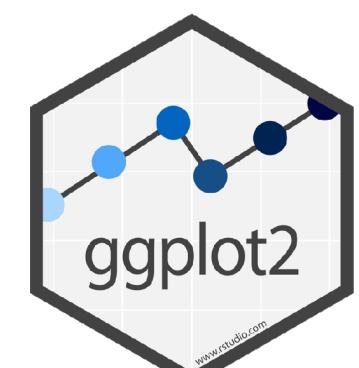
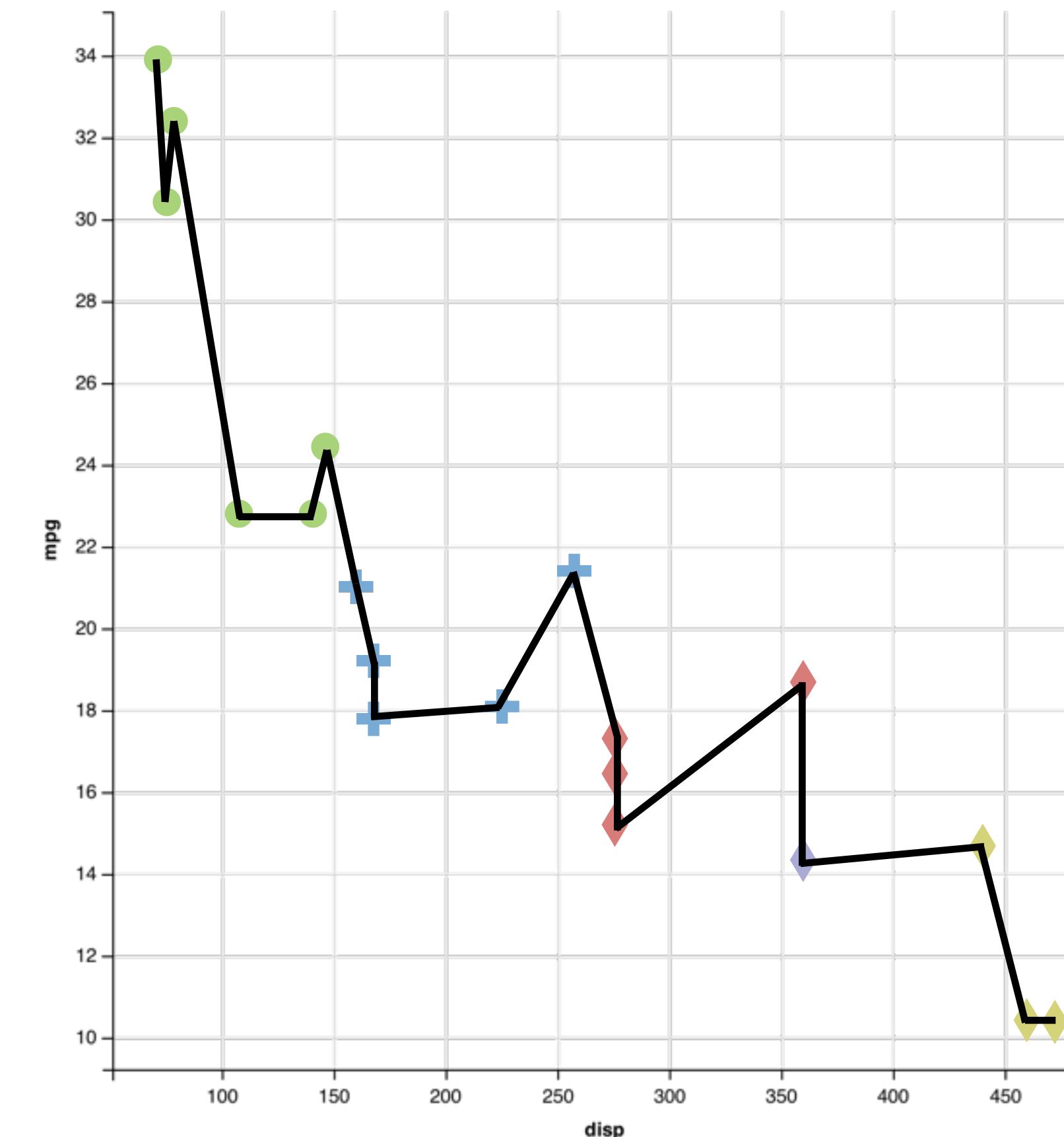


# mappings

	y ↑	shape ↑	x ↓	fill ↓
	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  
points  
lines

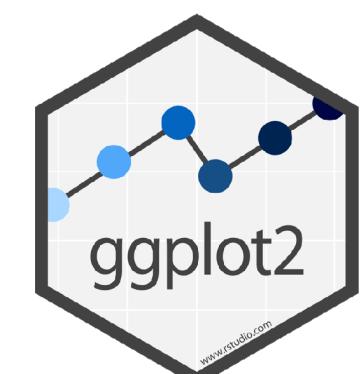
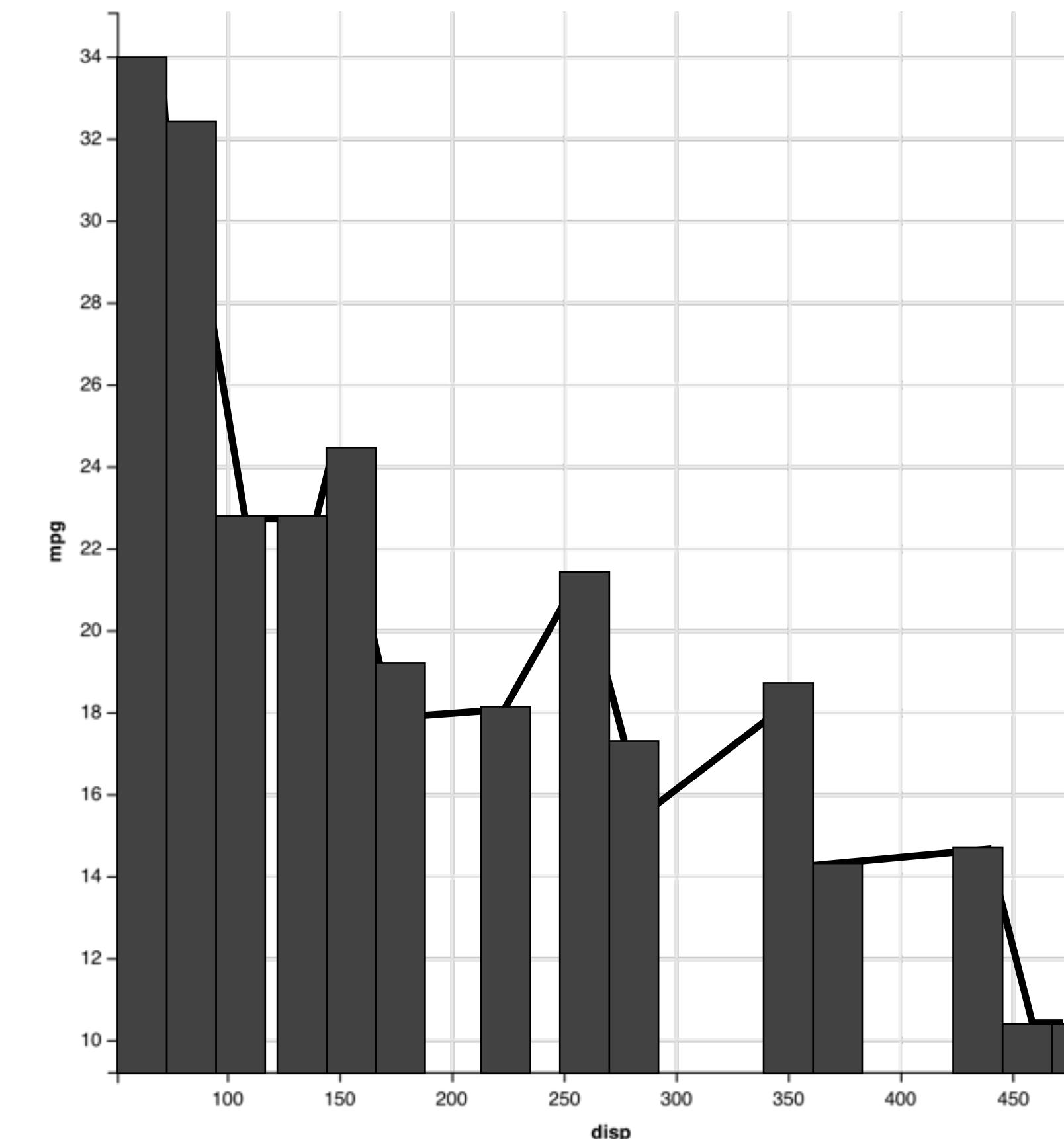


# mappings

	y	x
mpg	↑	↓
cyl		
21.0	6	160.0
21.0	6	160.0
22.8	4	108.0
21.4	6	258.0
18.7	8	360.0
18.1	6	225.0
14.3	8	360.0
24.4	4	146.7
22.8	4	140.8
19.2	6	167.6
17.8	6	167.6
16.4	8	275.8
17.3	8	275.8
15.2	8	275.8
10.4	8	472.0
10.4	8	460.0
14.7	8	440.0
32.4	4	78.7
30.4	4	75.7
33.9	4	71.1

data

geom  
points  
lines  
bars

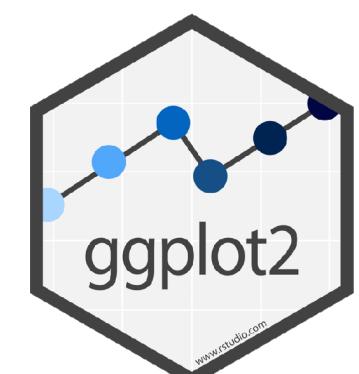
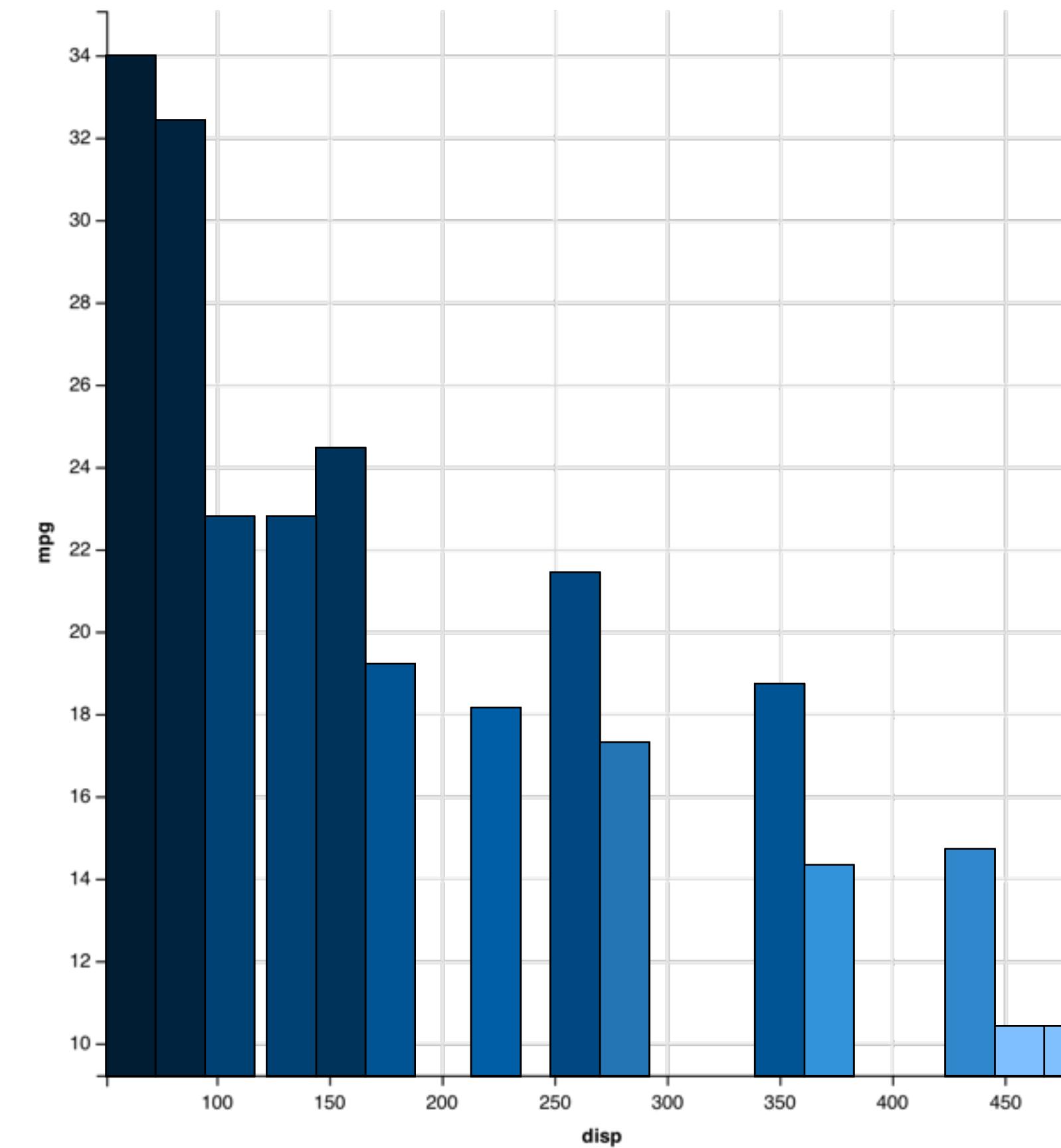


# mappings

	y		fill
	mpg	cyl	disp
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

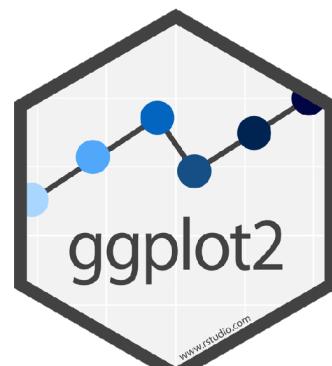
geom  
points  
lines  
bars



# To make a graph

[template]

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



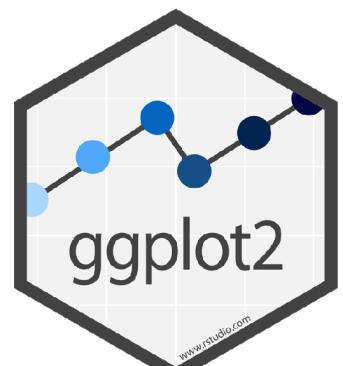
# To make a graph

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	●

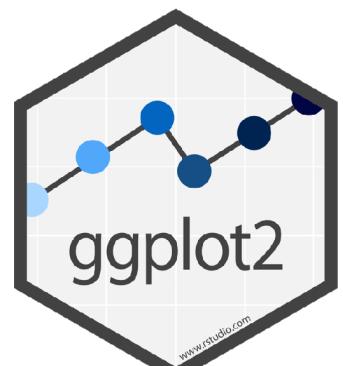
data

geom

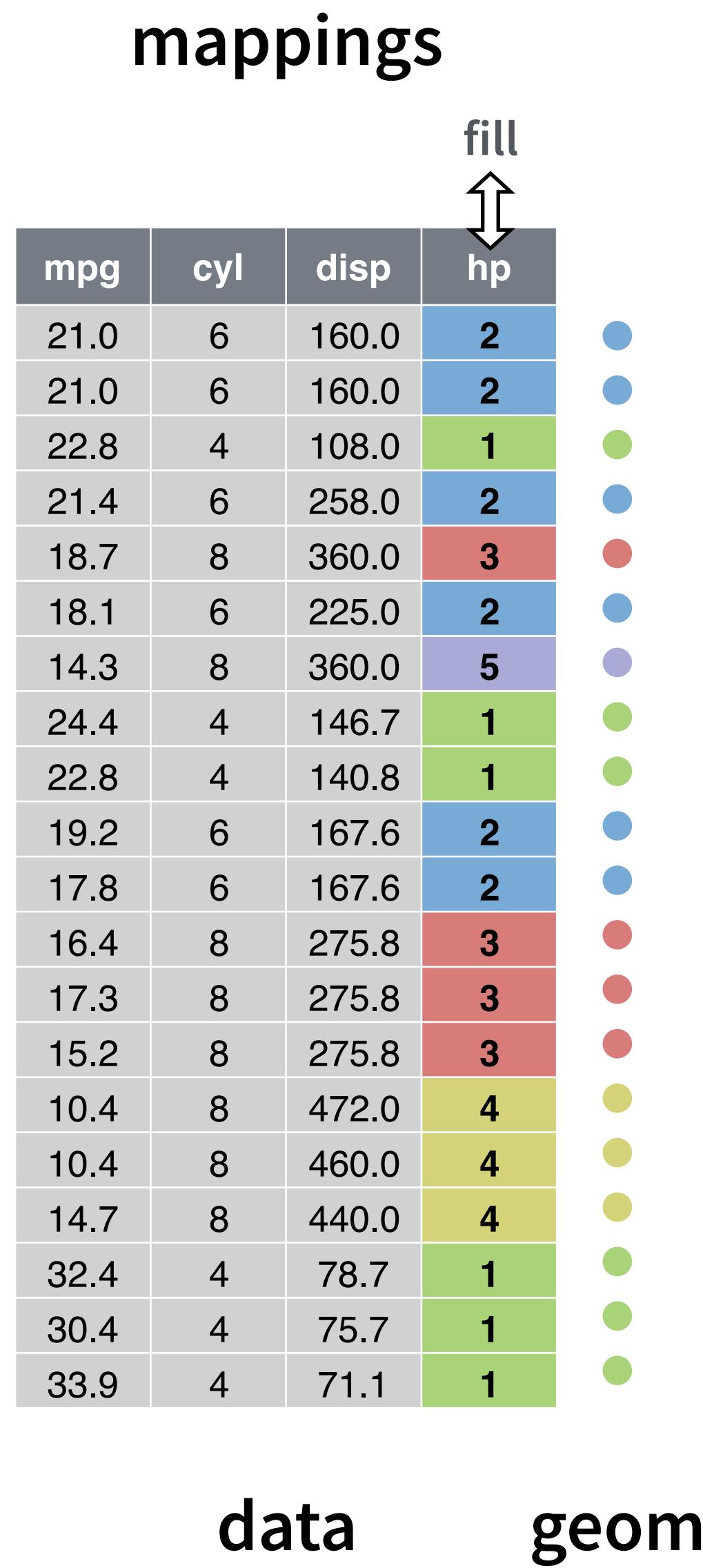
1. Pick a **data** set

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

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



# To make a graph

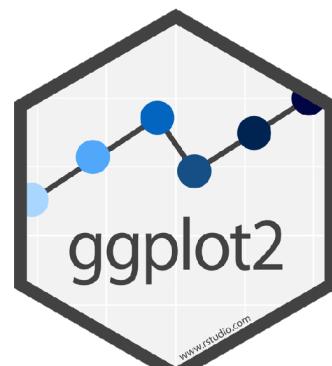


1. Pick a **data** set

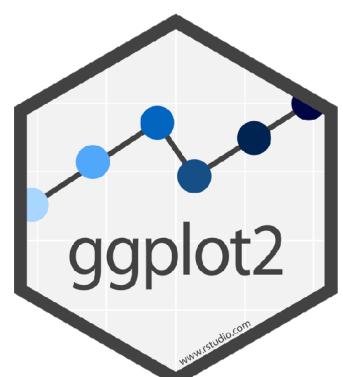
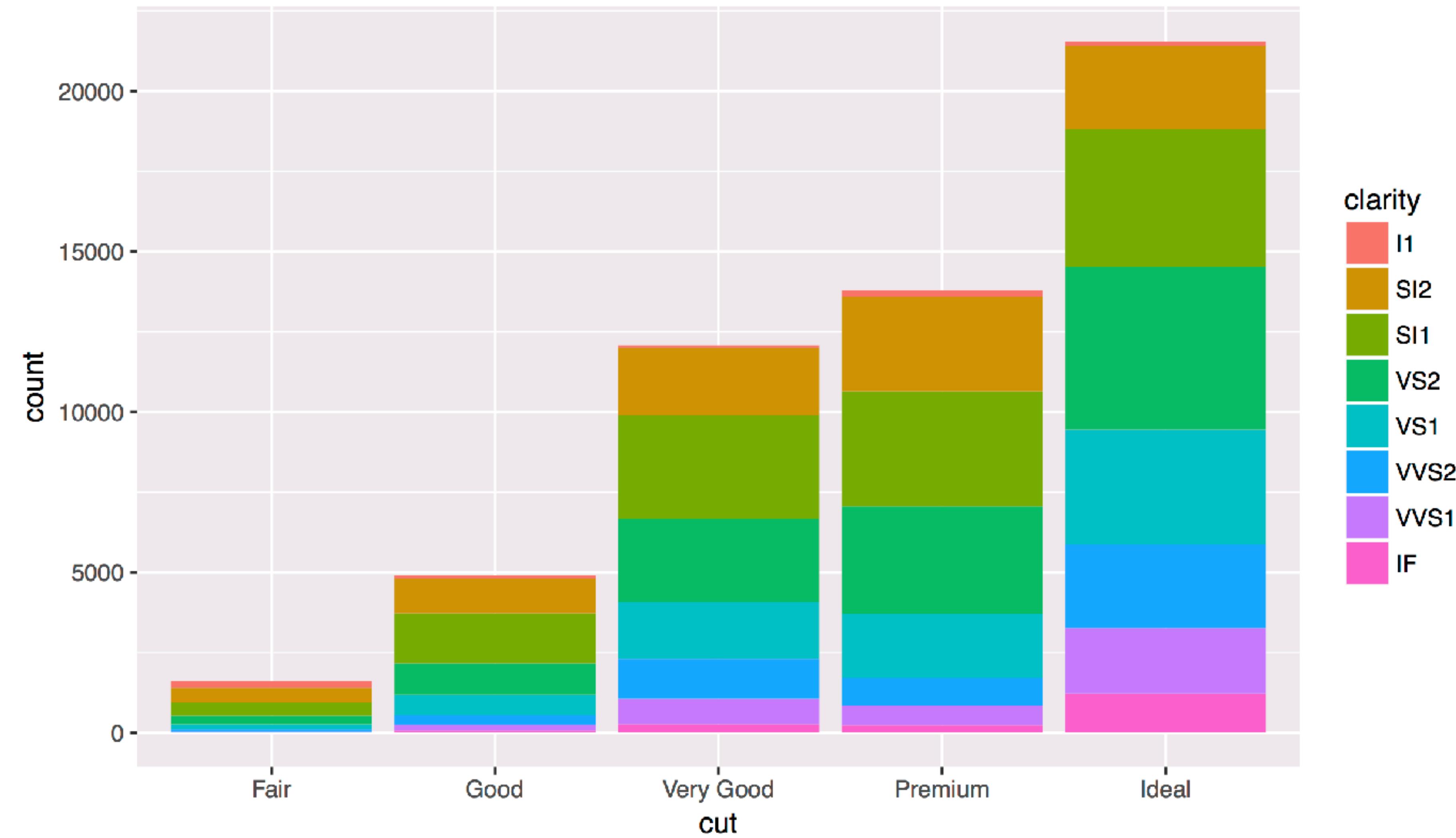
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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

3. **Map** aesthetic  
properties to  
variables

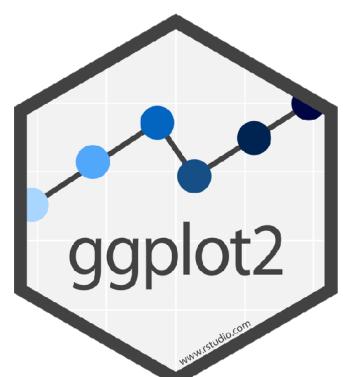
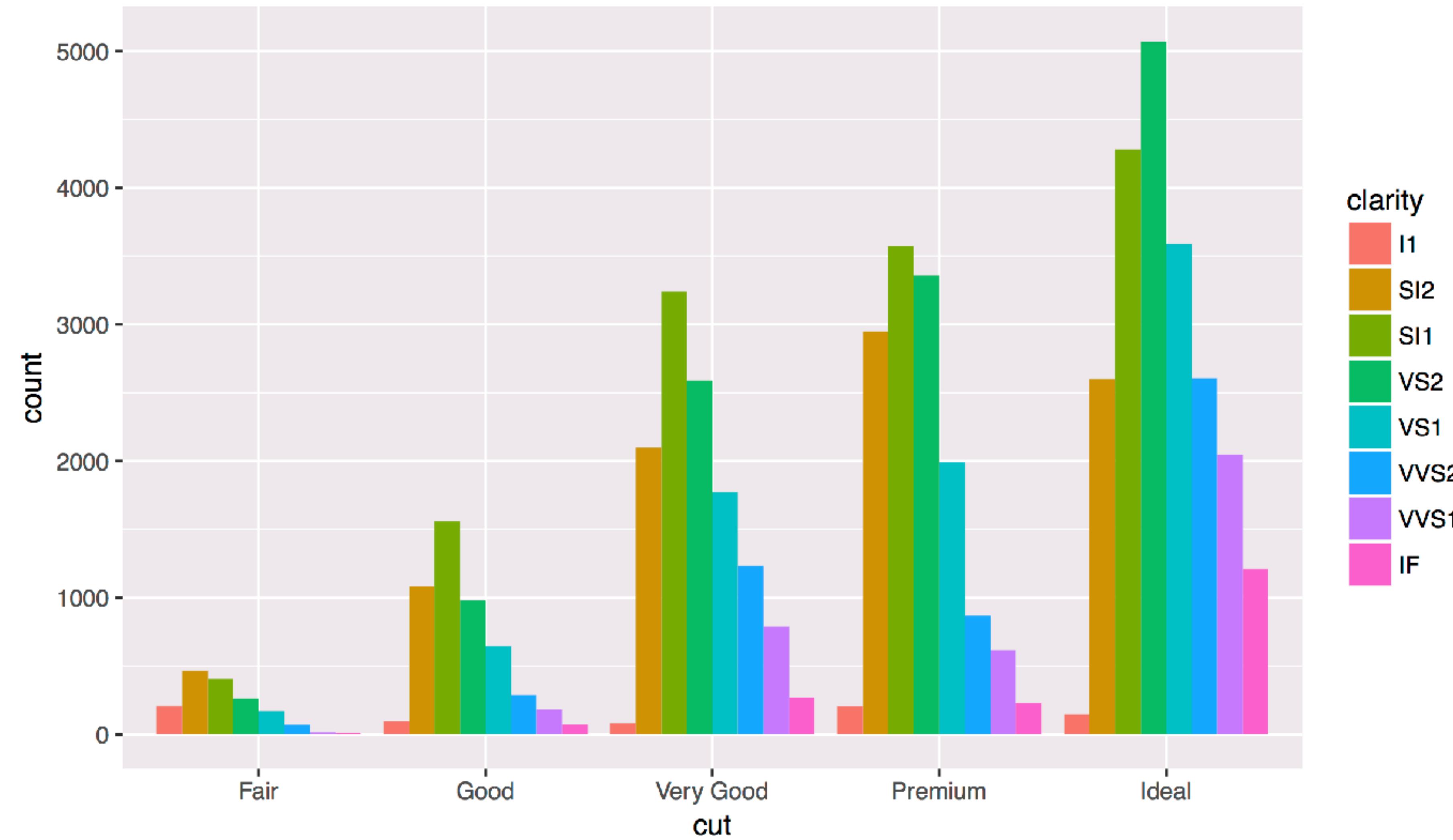


# what else?



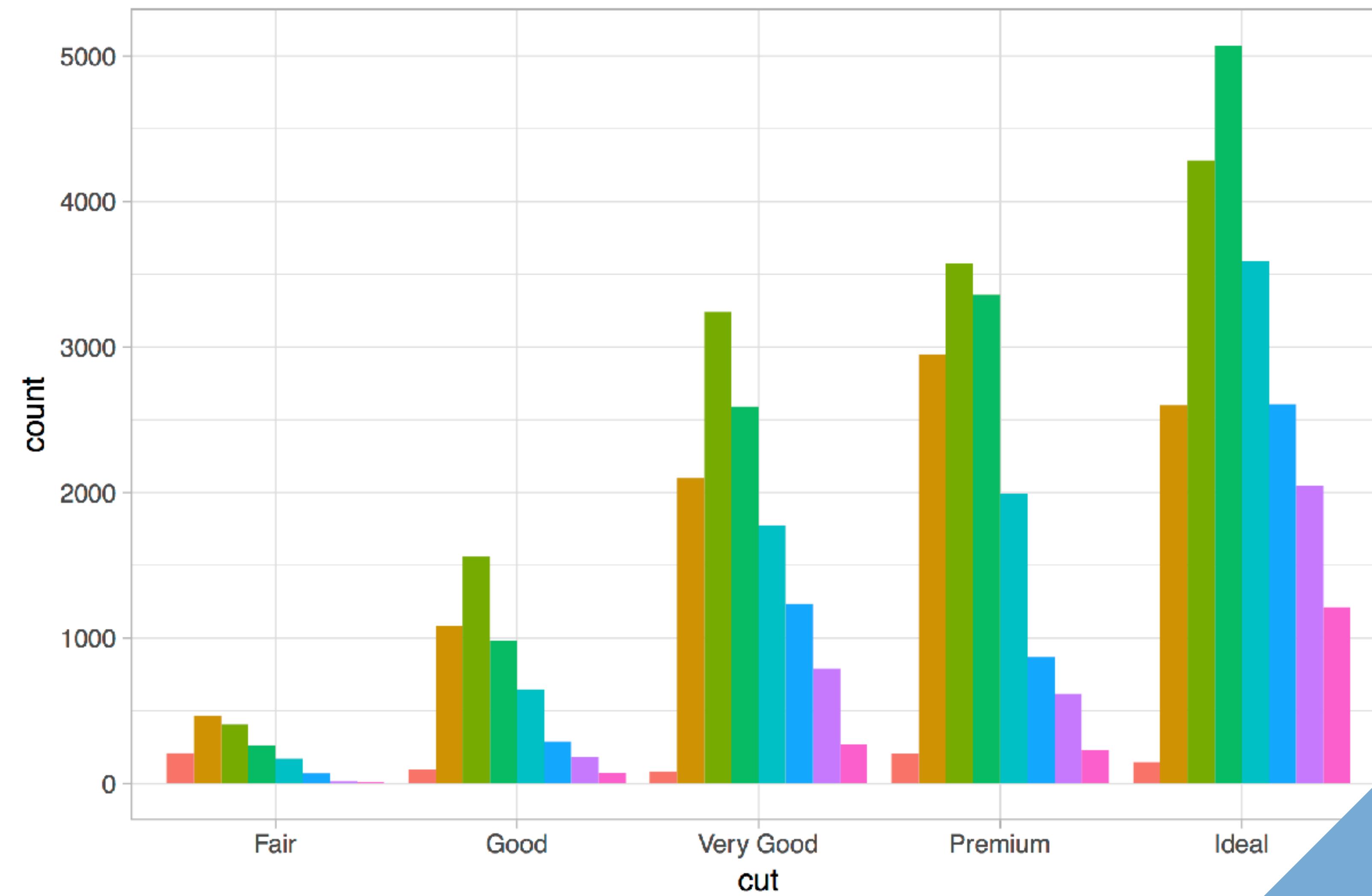
# Position Adjustments

How overlapping objects are arranged



# Themes

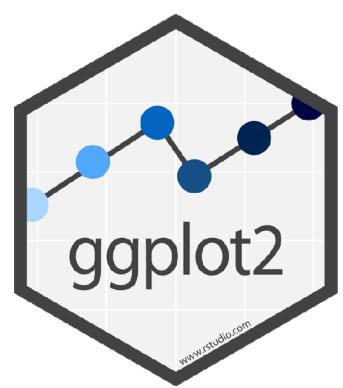
Visual appearance of non-data elements



clarity

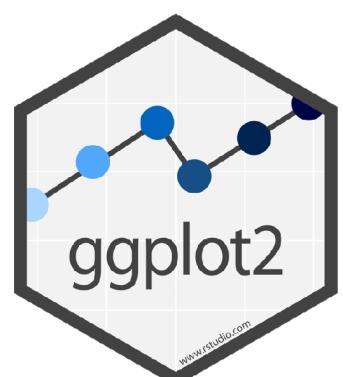
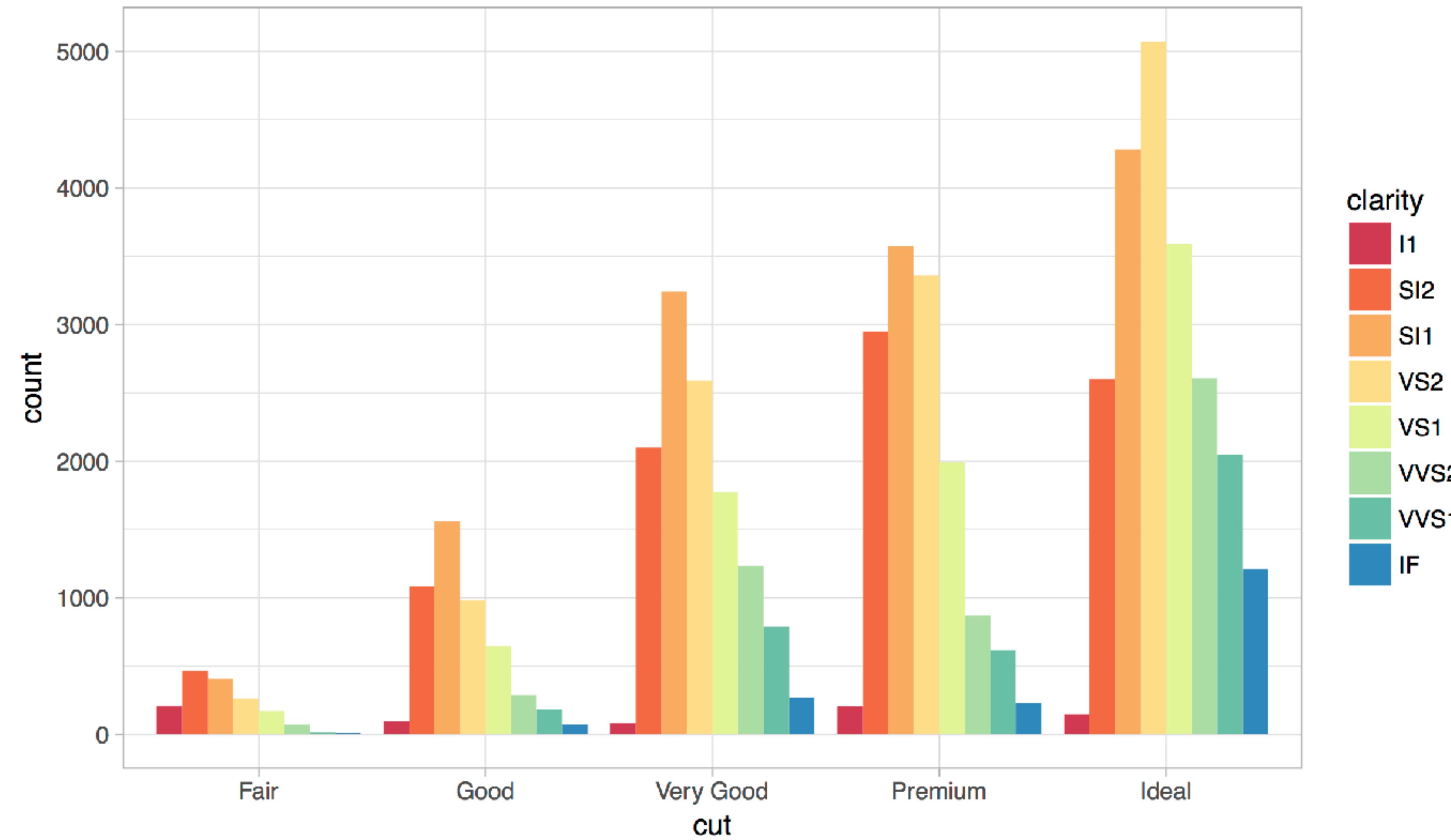
- I1
- SI2
- SI1
- VS2
- VS1
- VVS2
- VVS1
- IF

x theme\_bw()



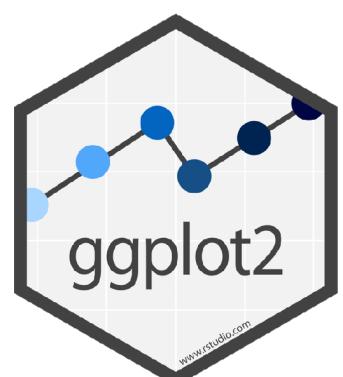
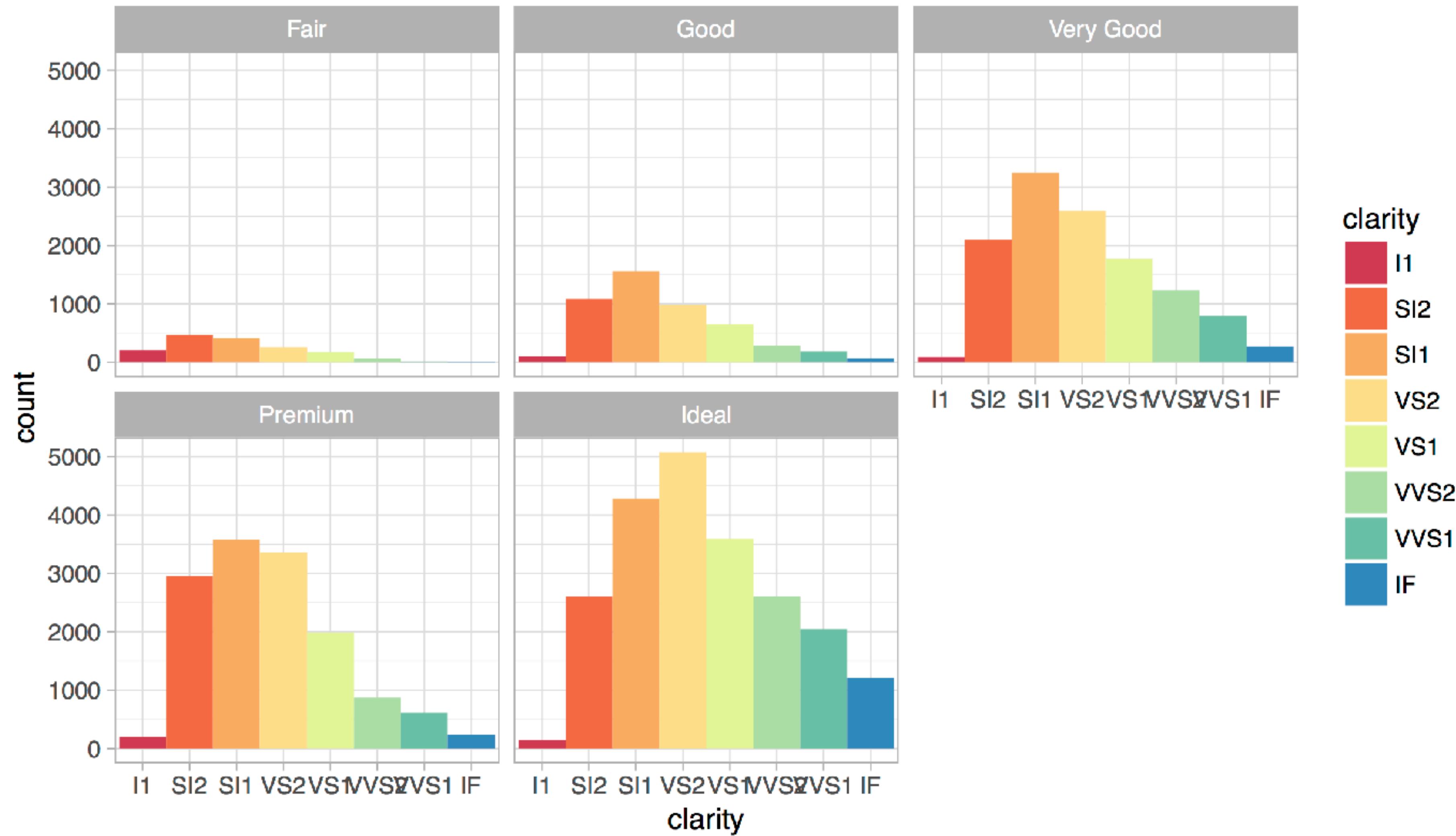
# Scales

Customize color scales, other mappings

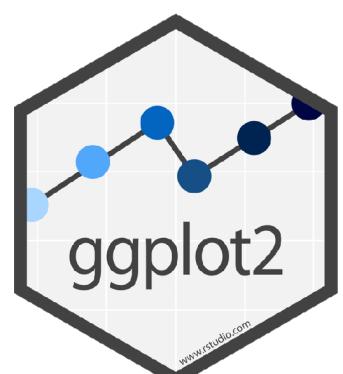
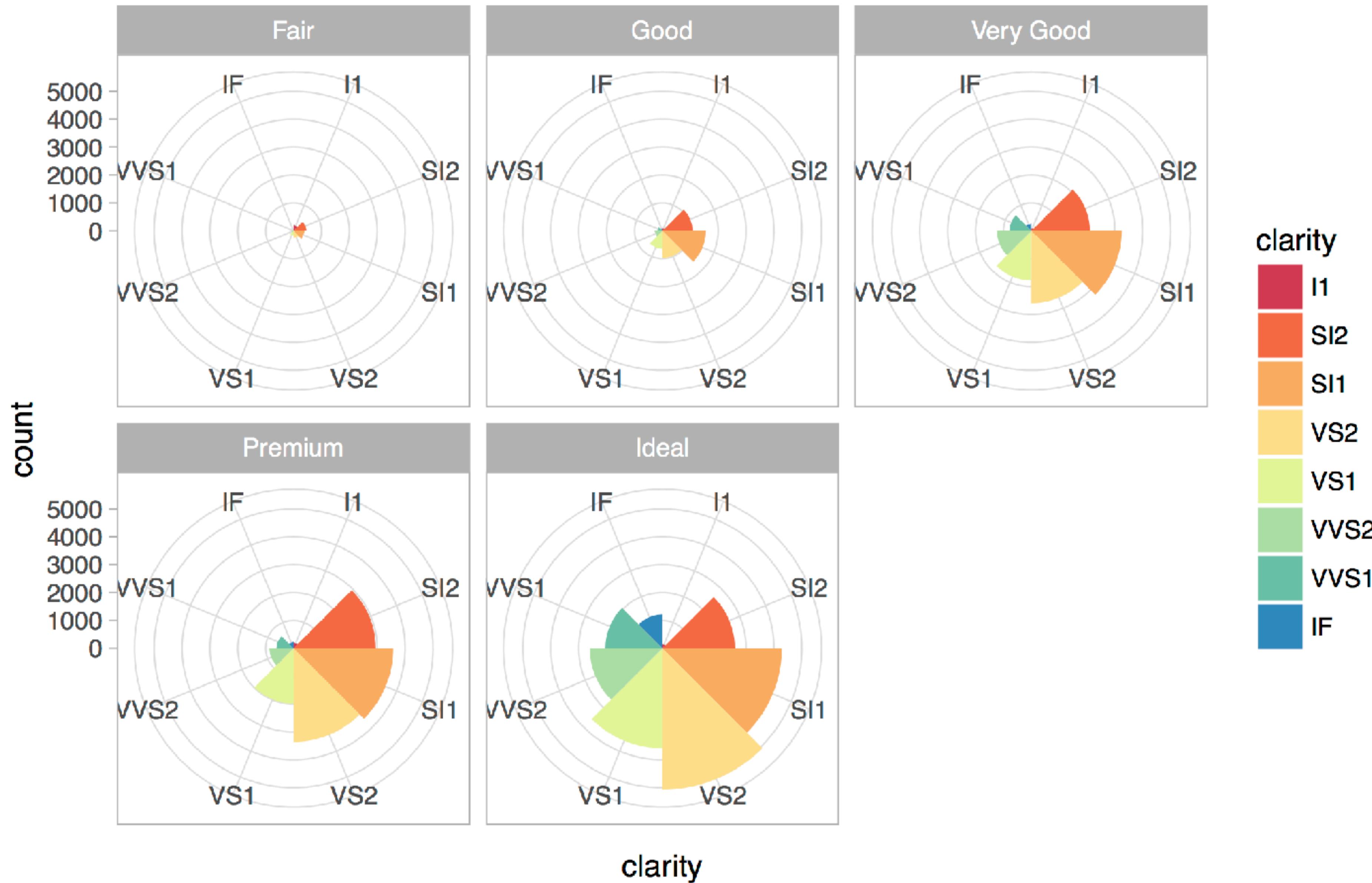


# Facets

Subplots that display subsets of the data.



# Coordinate systems



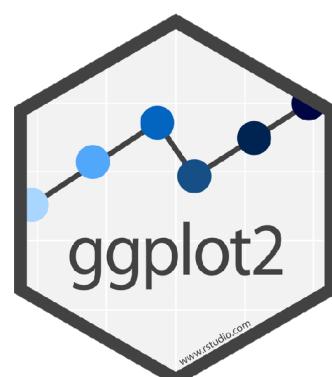
# Titles and captions

## Diamonds data

The data set is skewed towards ideal cut diamonds



Data by Hadley Wickham



# A ggplot2 template

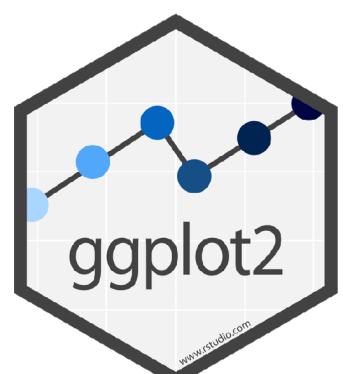
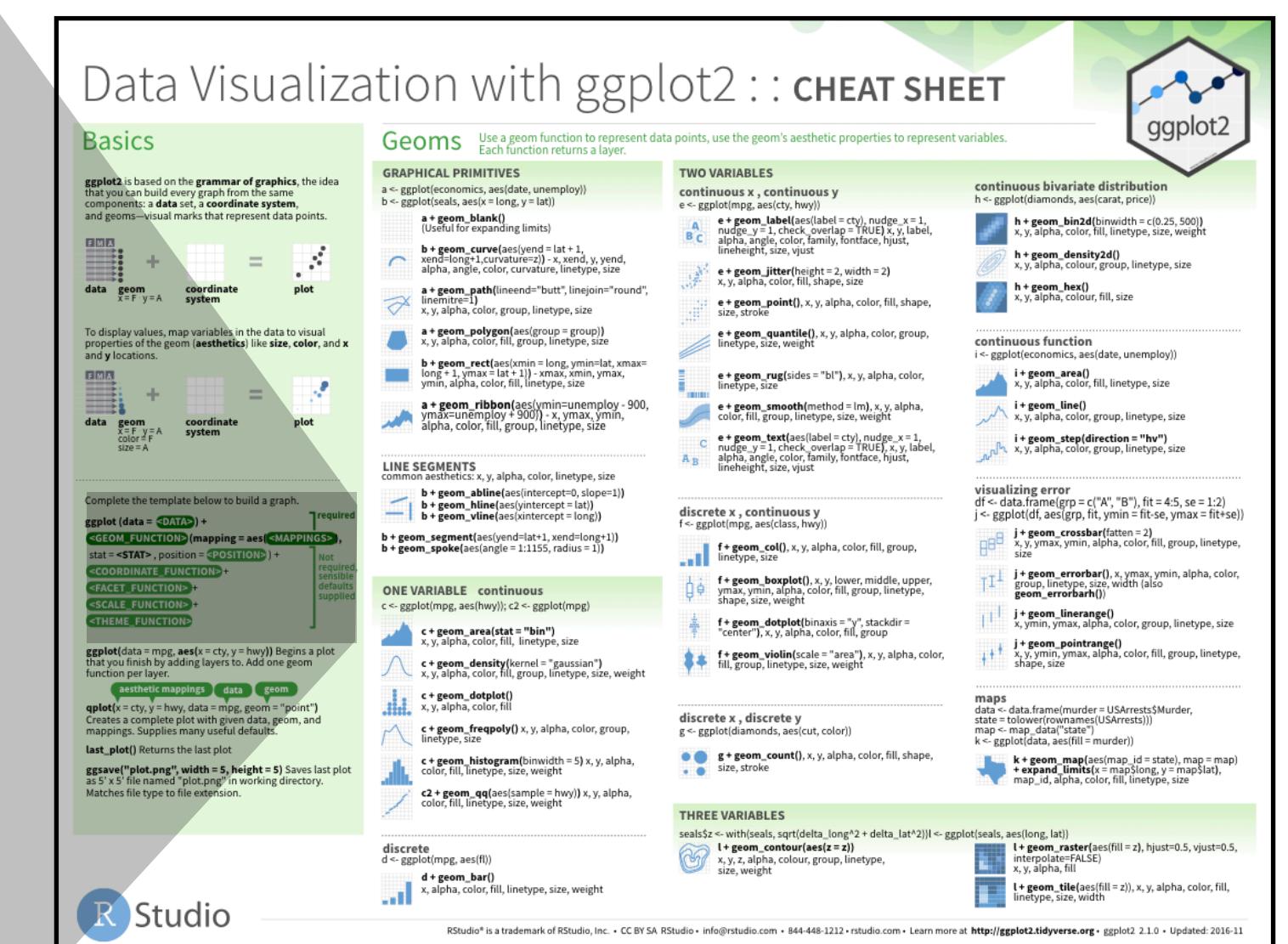
Make any plot by filling in the parameters of this template

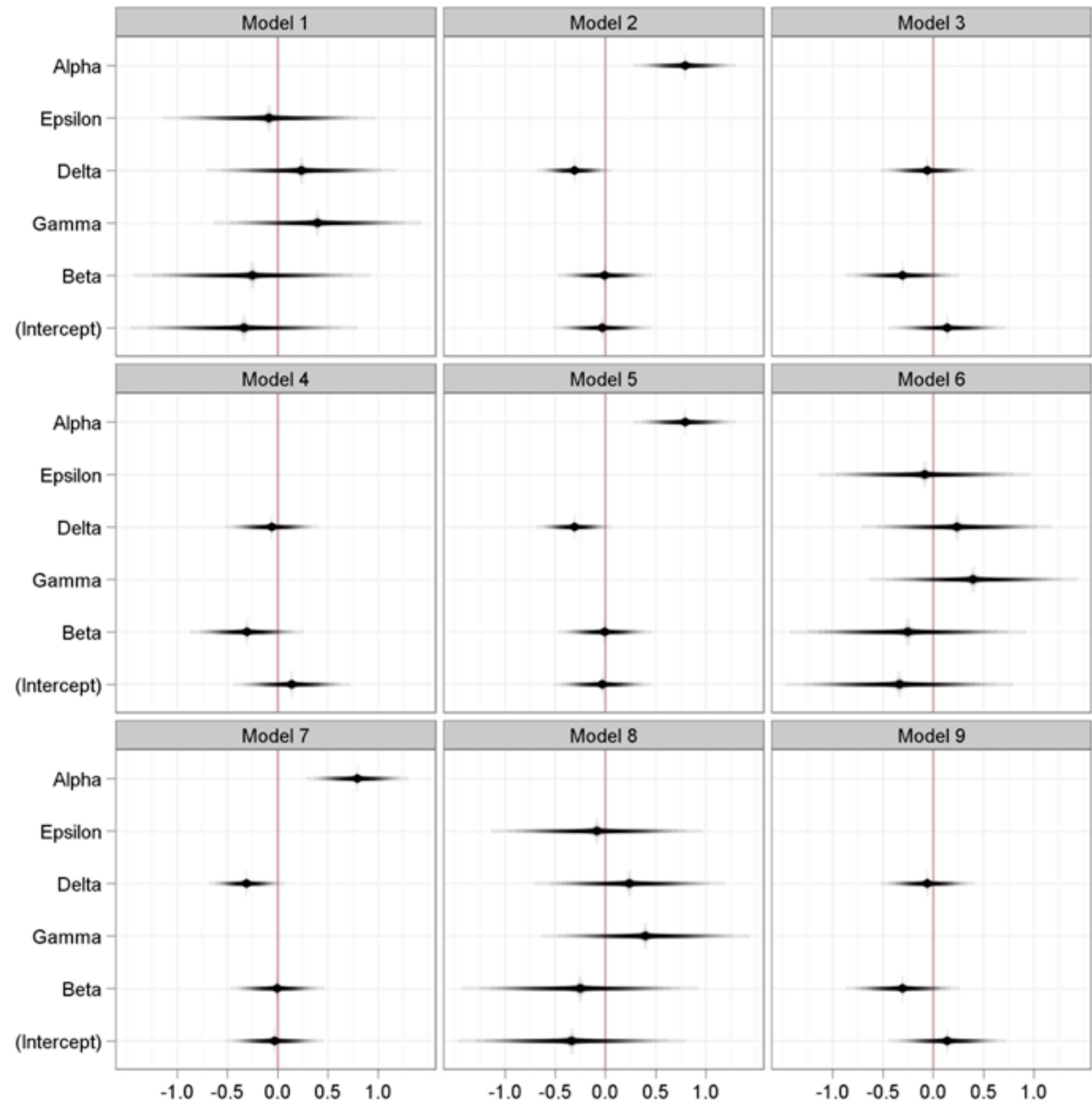
Complete the template below to build a graph.

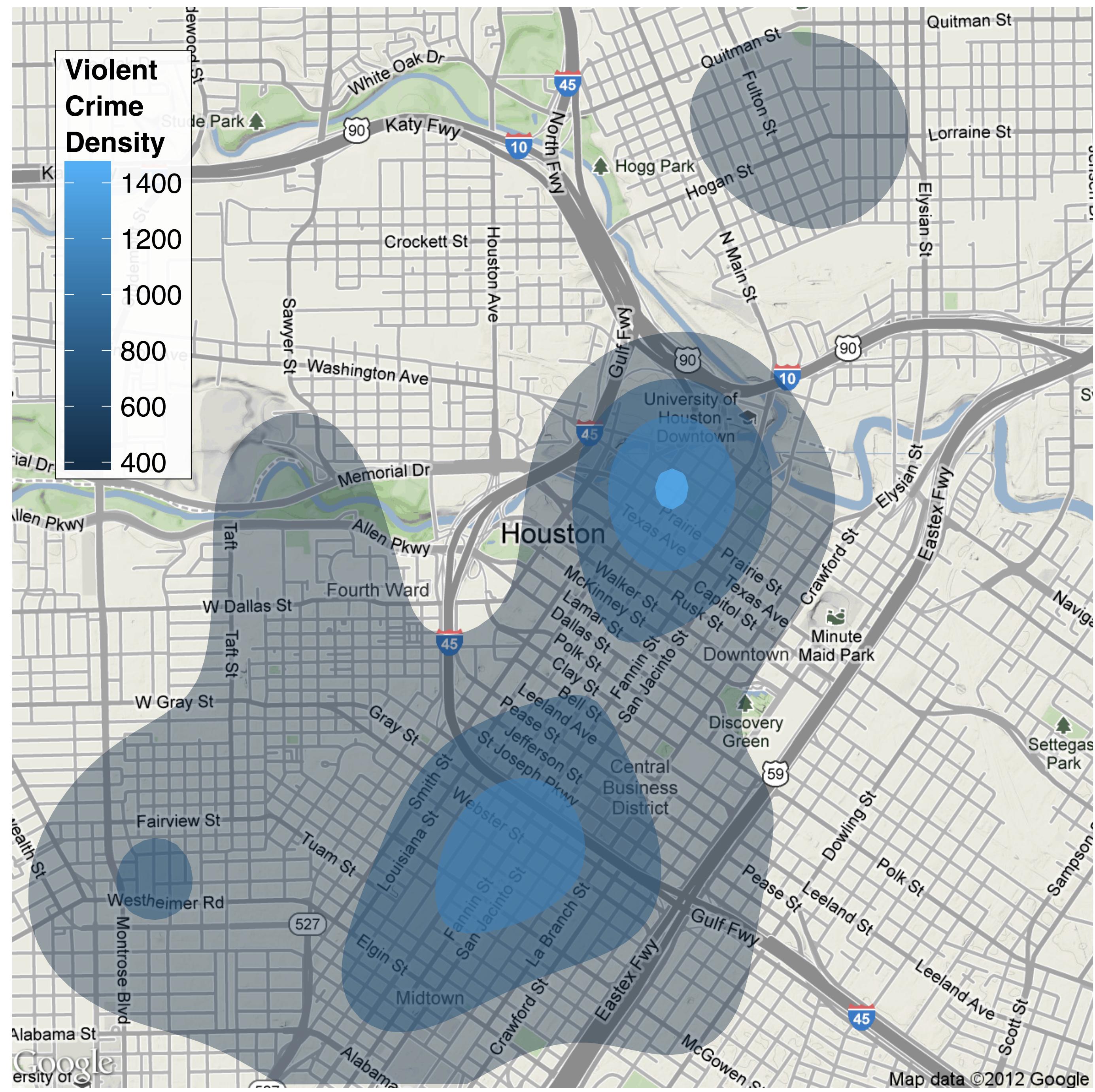
**ggplot (data = <DATA>) +**  
**<GEOM\_FUNCTION>(mapping = aes(<MAPPINGS>),**  
**stat = <STAT>, position = <POSITION>) +**  
**<COORDINATE\_FUNCTION> +**  
**<FACET\_FUNCTION> +**  
**<SCALE\_FUNCTION> +**  
**<THEME\_FUNCTION>**

required

Not required,  
sensible  
defaults  
supplied







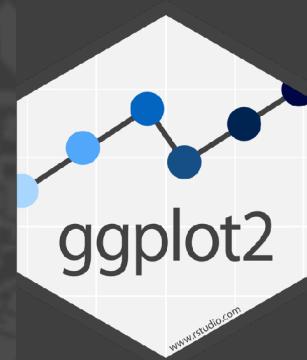
# London Cycle Hire Journeys

Thicker, yellower lines mean more journeys



Data: 3.2 Million Journeys (from TfL)  
Routing: Ollie O'Brien (@oobr) + OpenStreetMap cc-by-sa  
Buildings: OS OpenData Crown Copyright 2011  
Map: James Cheshire (@spatialanalysis)

James Cheshire, <http://bit.ly/xqHhAs>



# r4ds.had.co.nz

COMPLETE  
AND FREE  
ONLINE

The Ultimate Resource for All of Today

R for Data Science

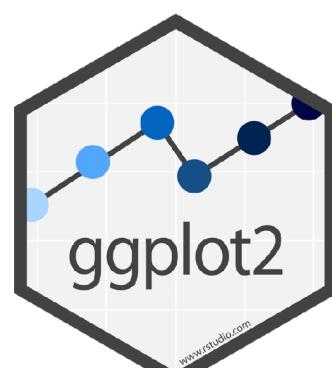
Garrett Grolemund

Hadley Wickham

## Welcome

This is the website for “R for Data Science”. This book will teach you how to do data science with R: You’ll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you’ll learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You’ll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You’ll also learn how to

O'REILLY



# Your Turn

Navigate up to the **03-Transform** folder.

**Open `03-Transform-Exercises.Rmd`**

# Visualize Data with

