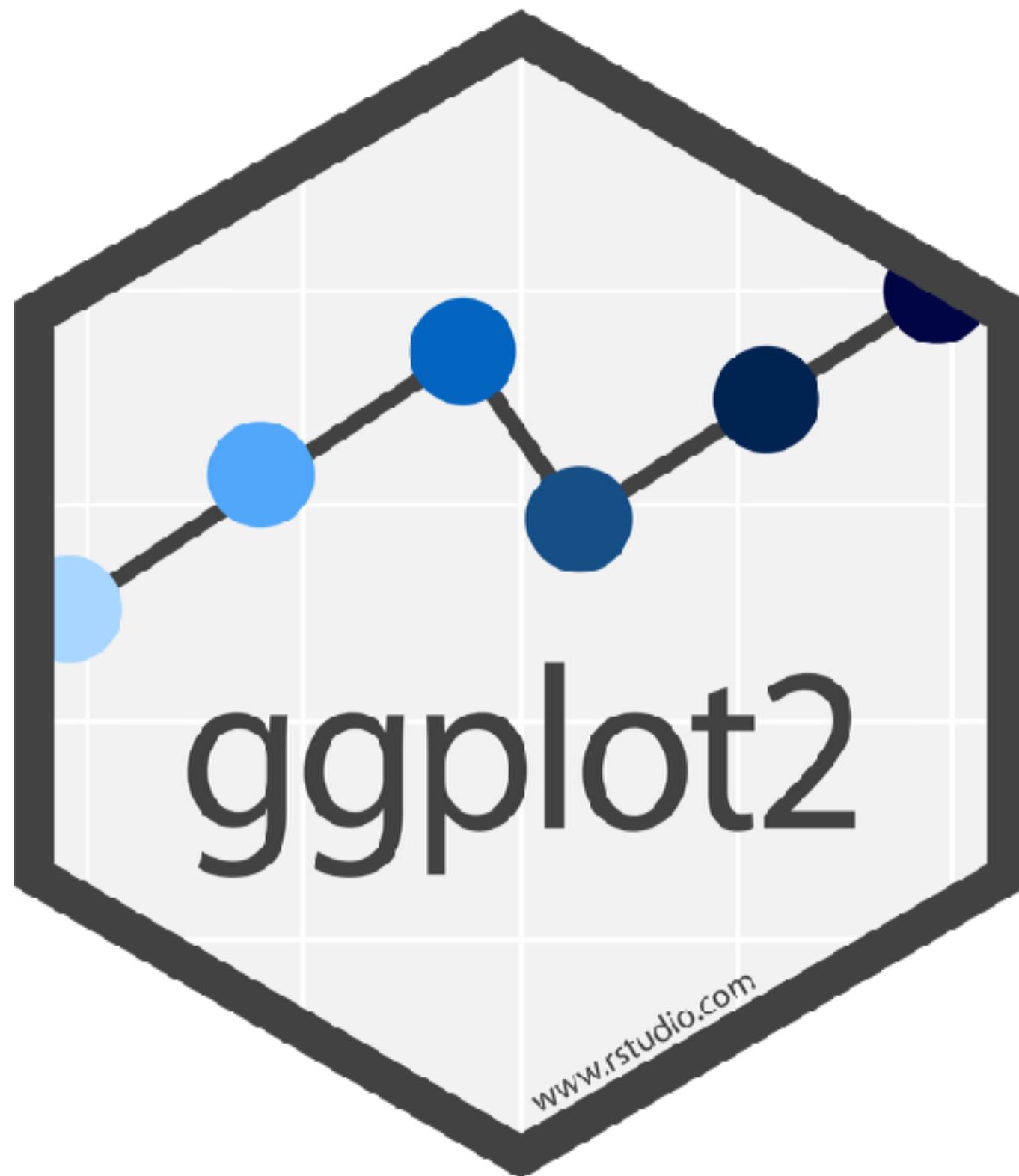


Visualize Data with



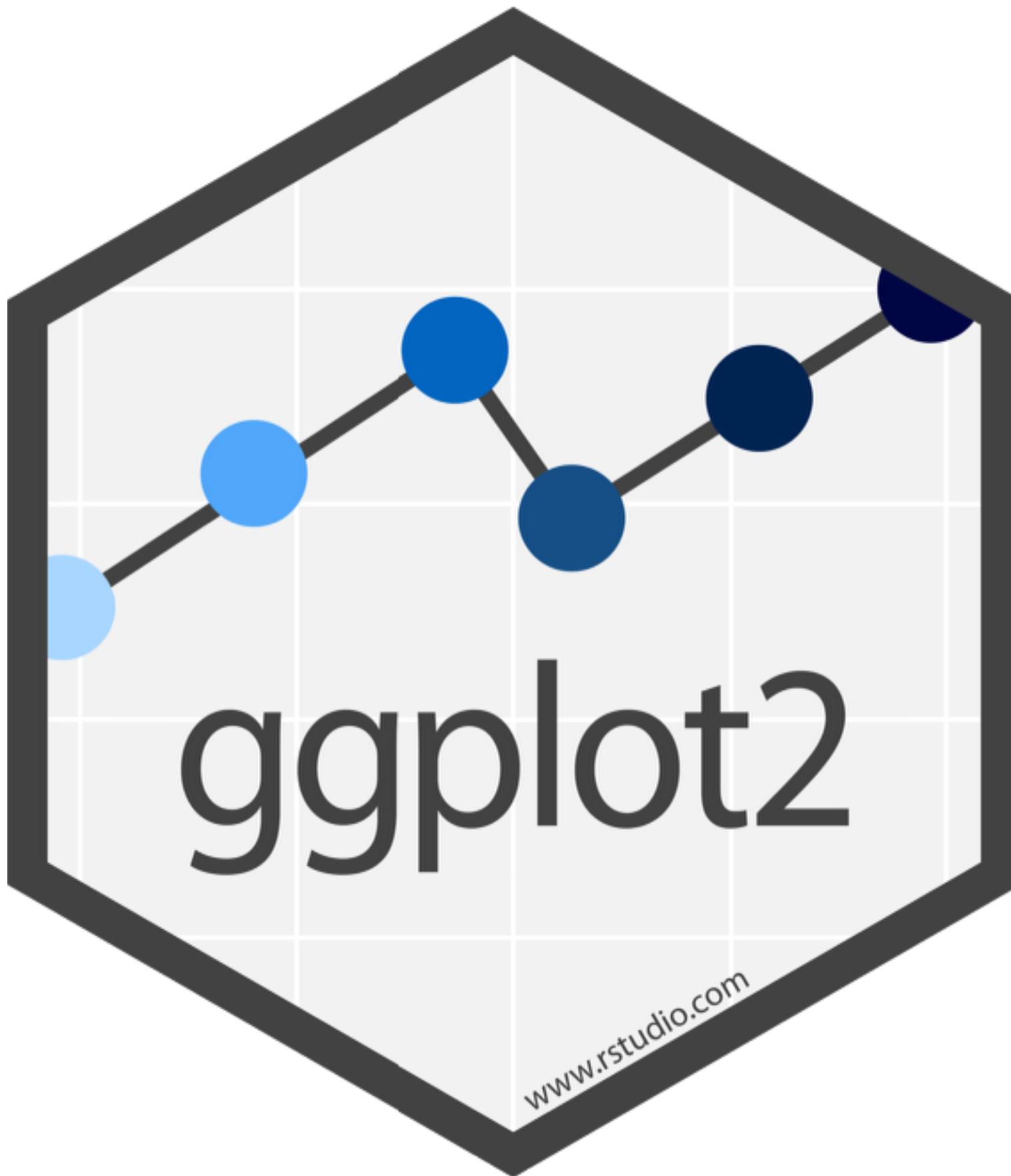
In R4DS

Visualizing Data

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

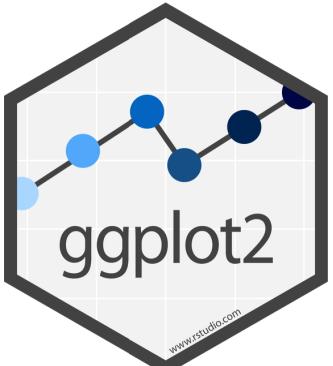
- John Tukey

ggplot2



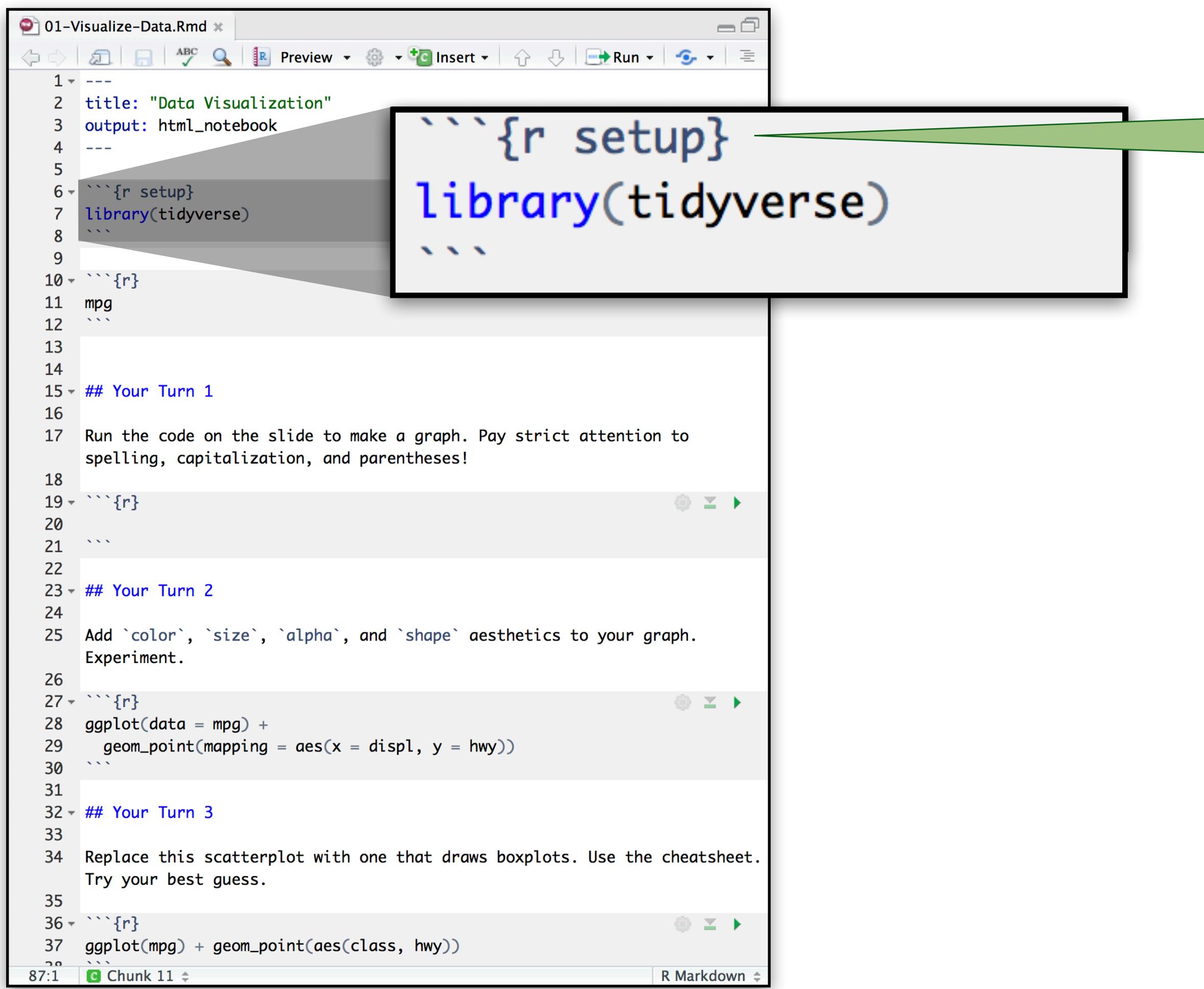
One of the earliest members of the tidyverse.

Complicated plots, come from combining simple components.



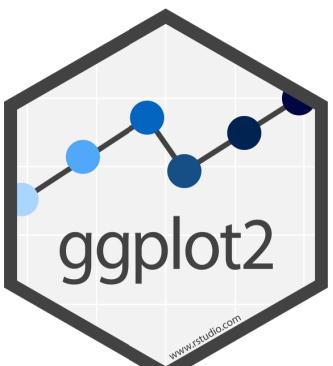
Setup

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
```

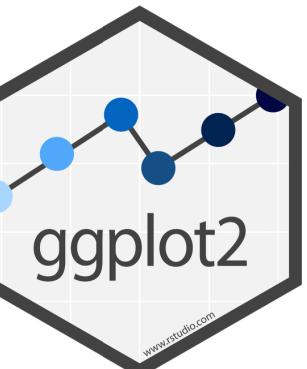


mpg

Fuel economy data for 38 models of car.

```
mpg
```

```
?mpg
```



Quiz

Confer with your neighbours.

What relationship do you expect to see between engine size (displ) and highway fuel efficiency (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!

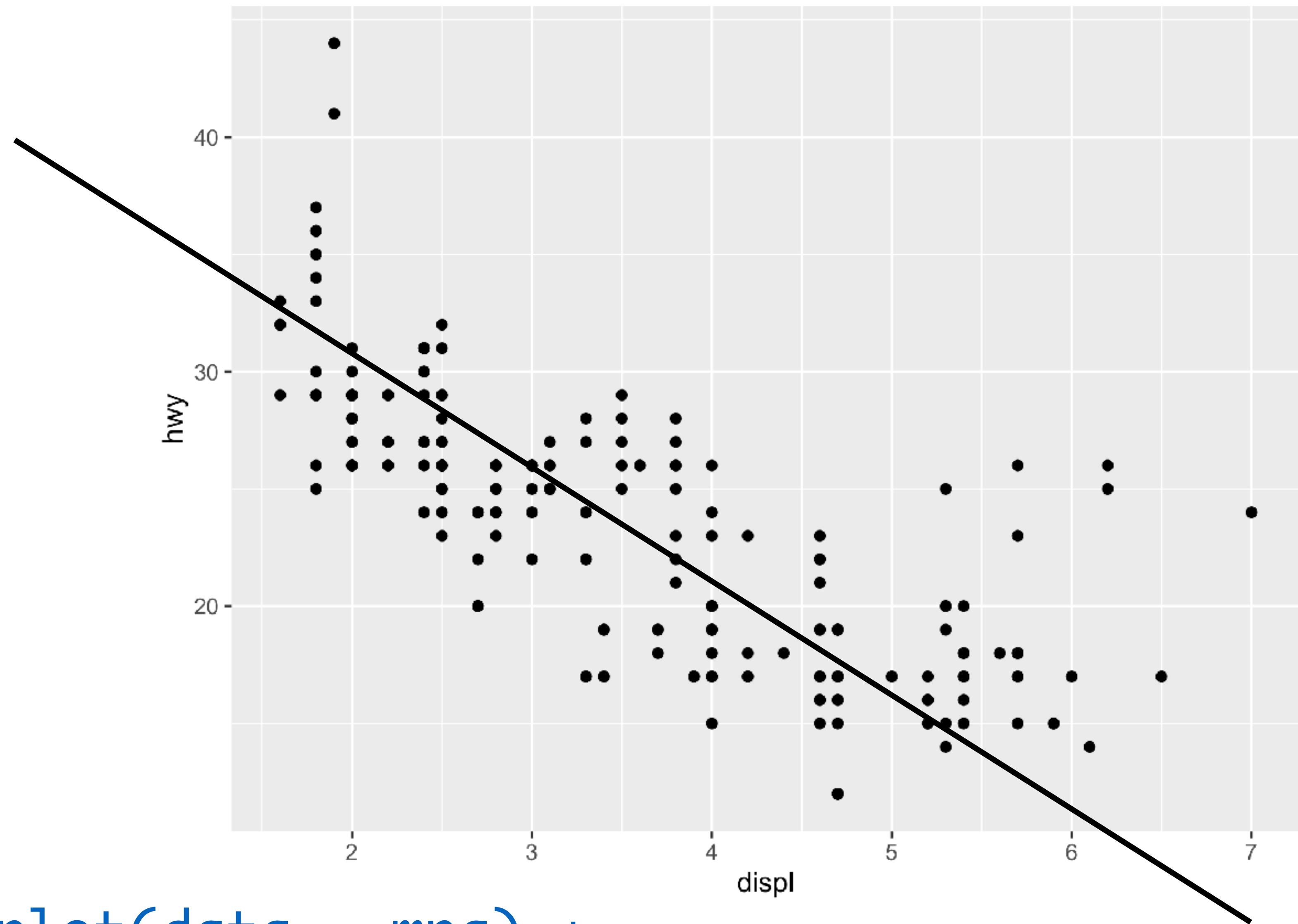
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

I'm working on it

I'm stuck!

I'm done!





```
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

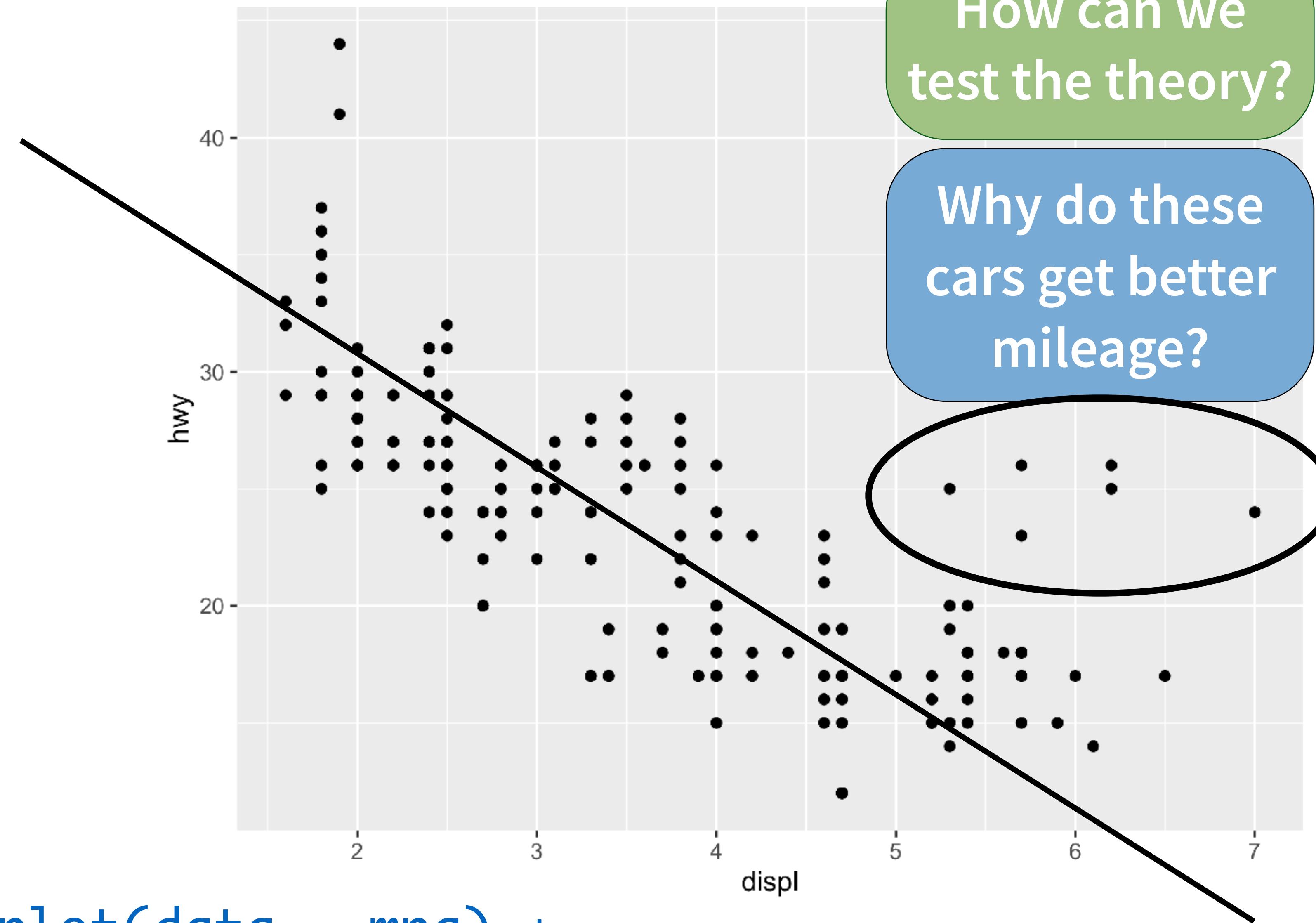
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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

A Template

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

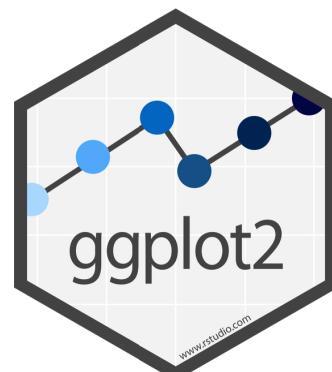
Mappings



How can we
test the theory?

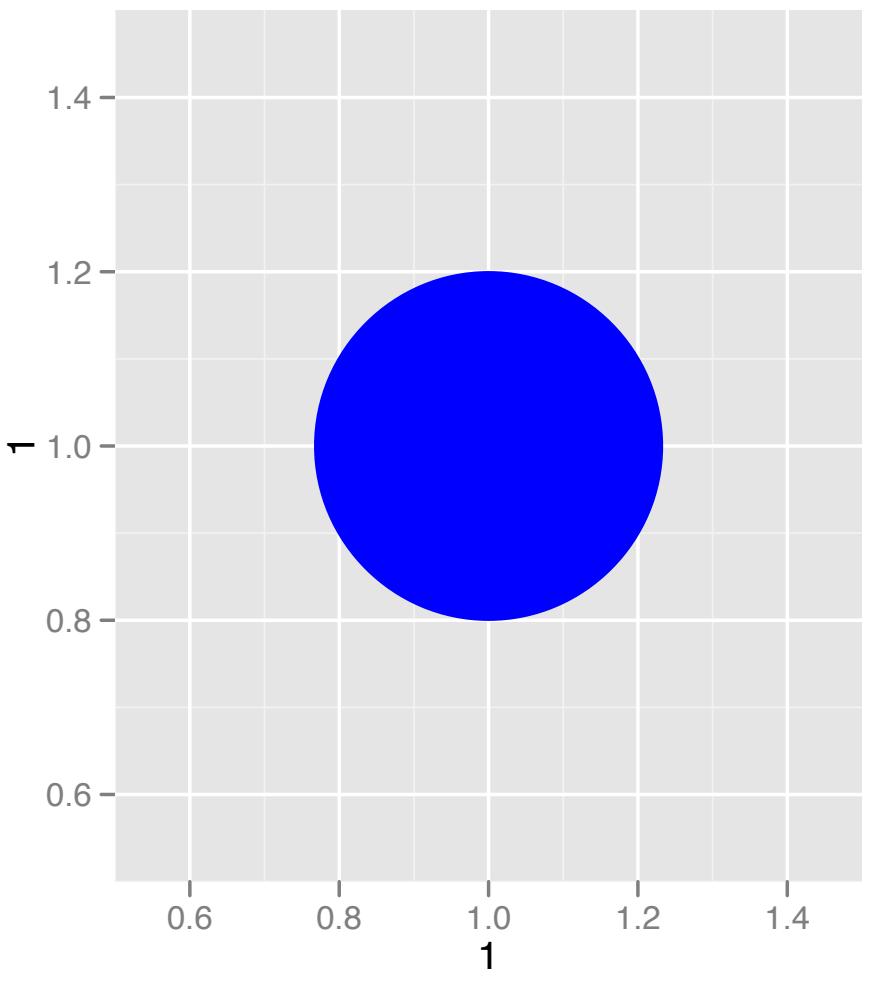
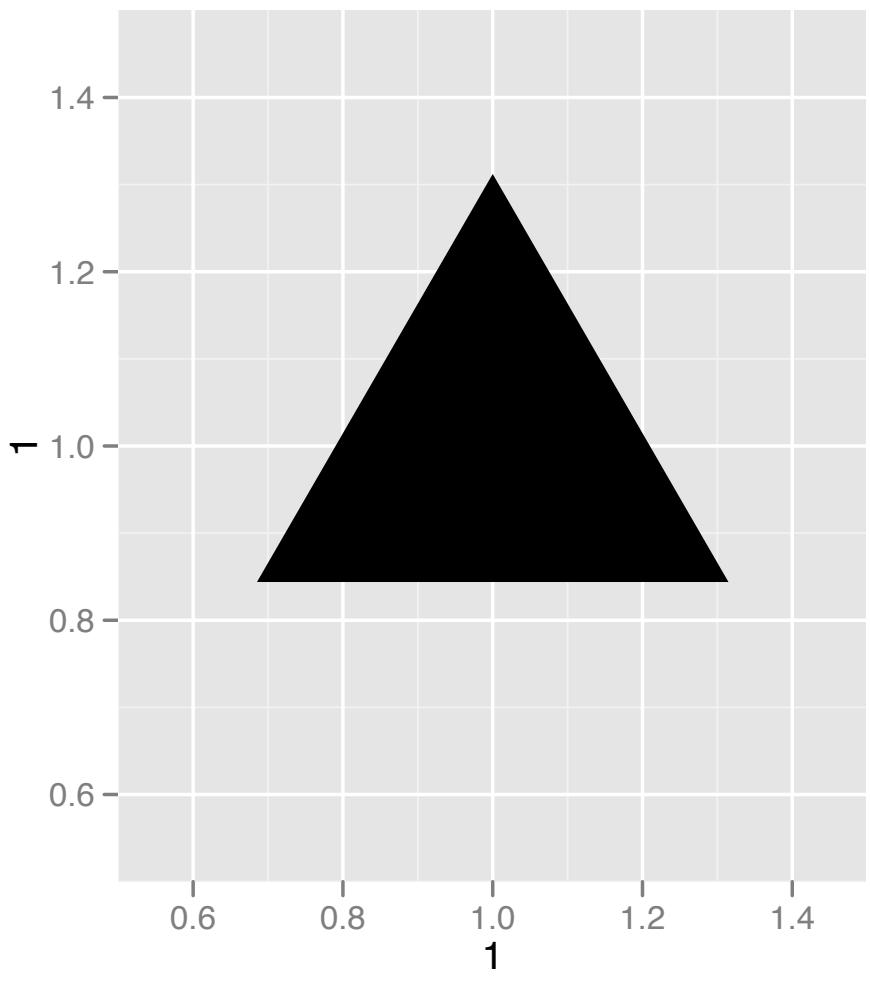
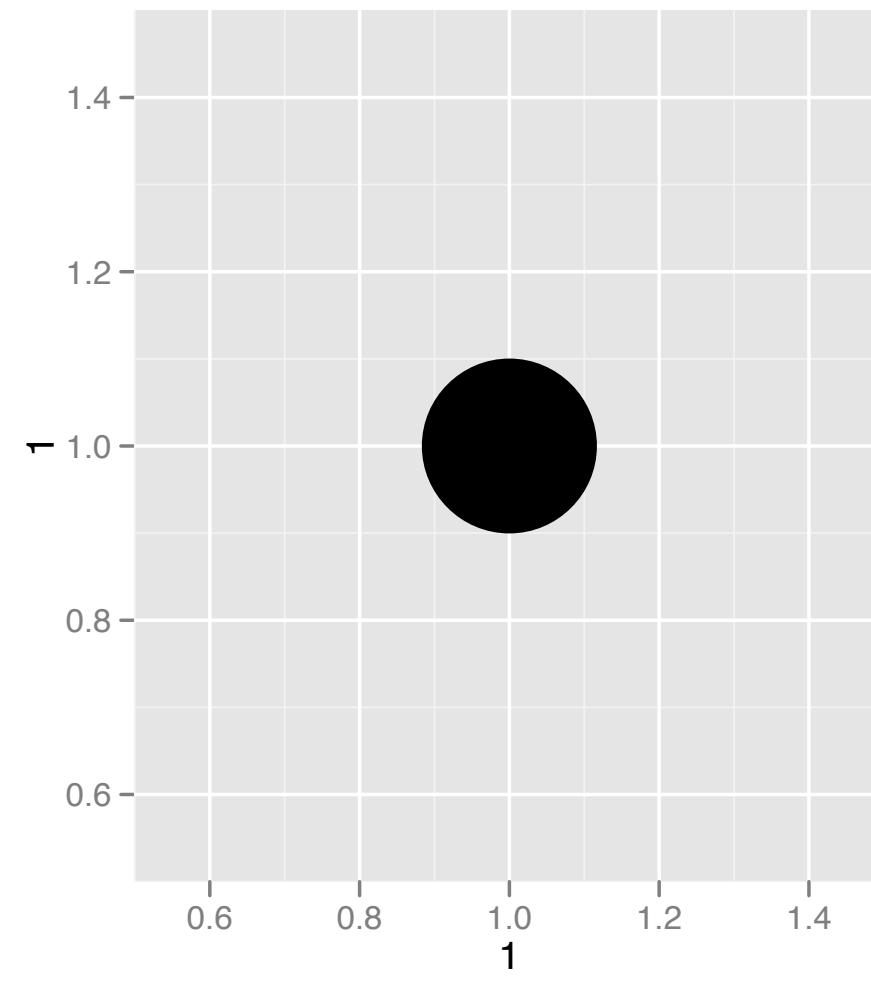
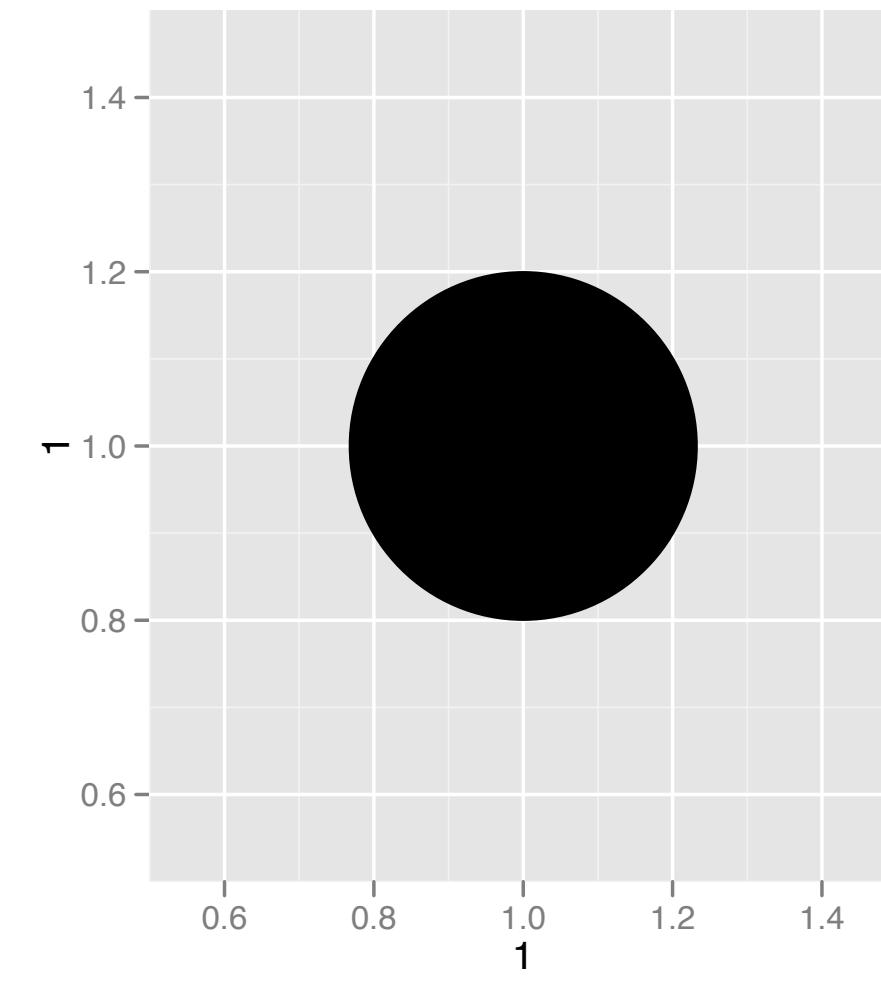
Why do these
cars get better
mileage?

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



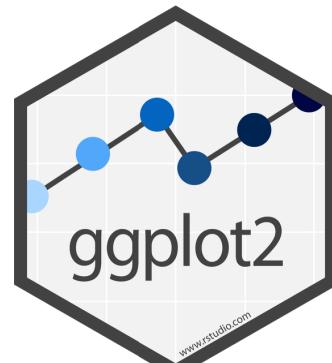
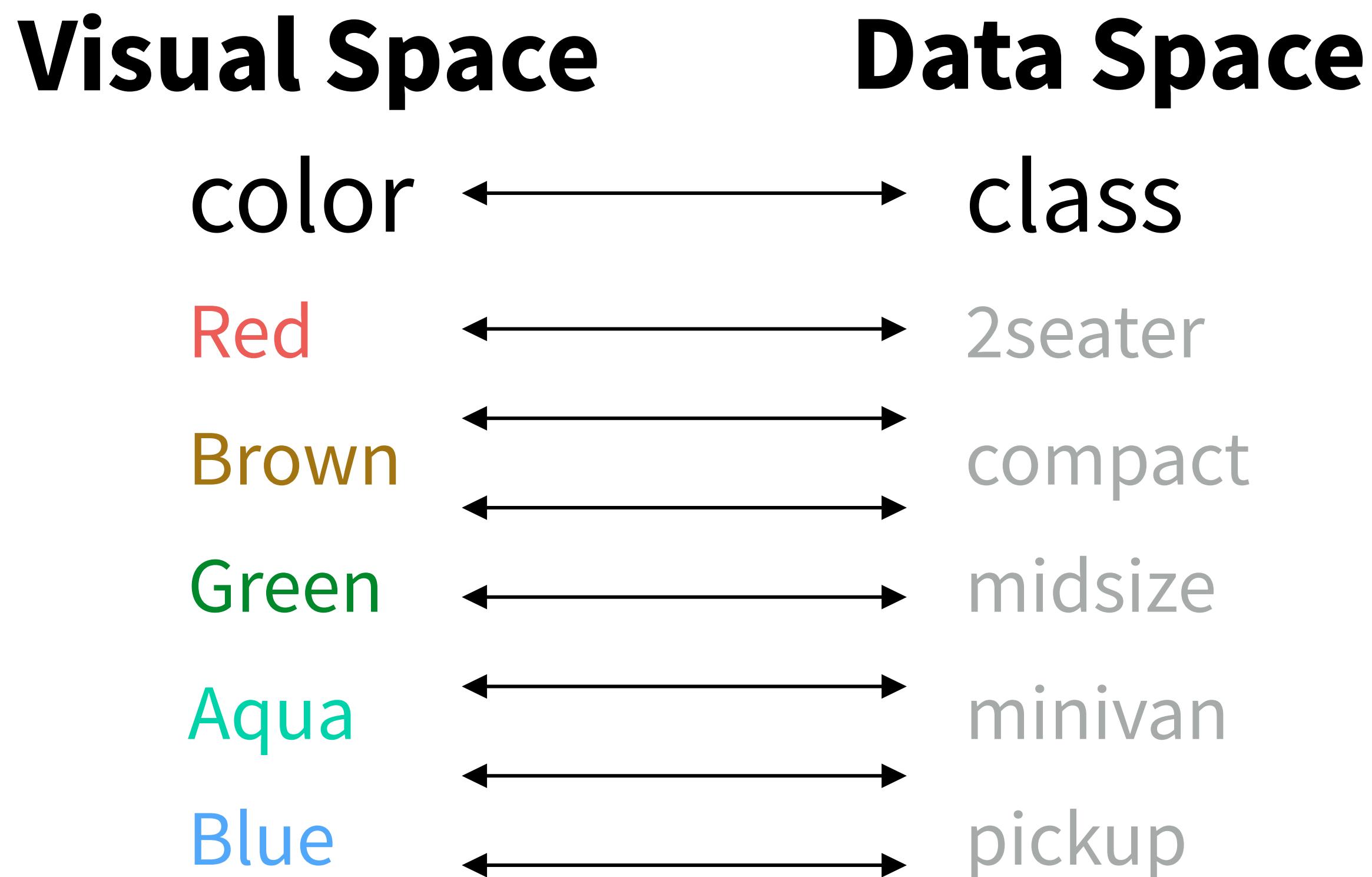
Aesthetics

Visual properties of a geometric object



How do the appearance of these points vary?

Mappings describe how aesthetics should relate to variables in the data.



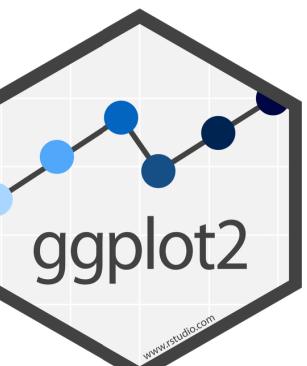
Aesthetics

aesthetic
property

Variable to
map it to

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

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



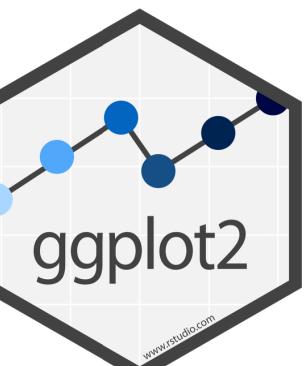
Aesthetics

aesthetic
property

Variable to
map it to

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

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



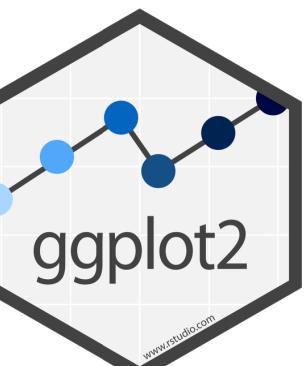
Aesthetics

aesthetic
property

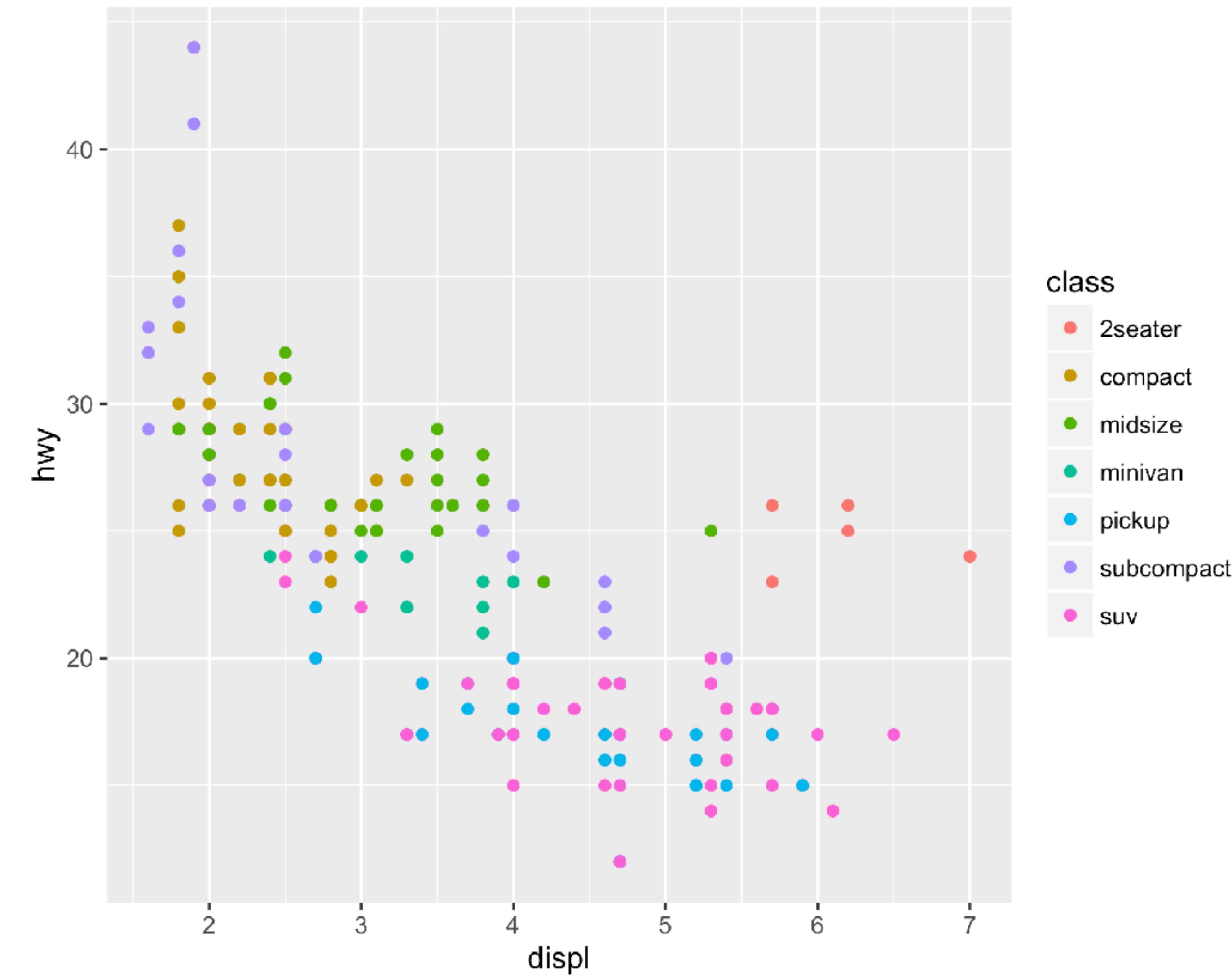
Variable to
map it to

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

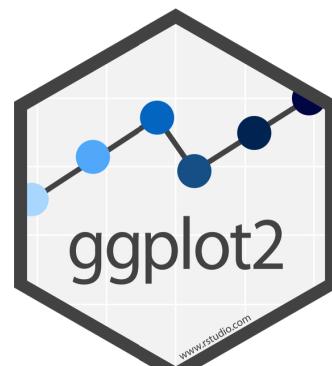
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```



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



Legend
added
automatically



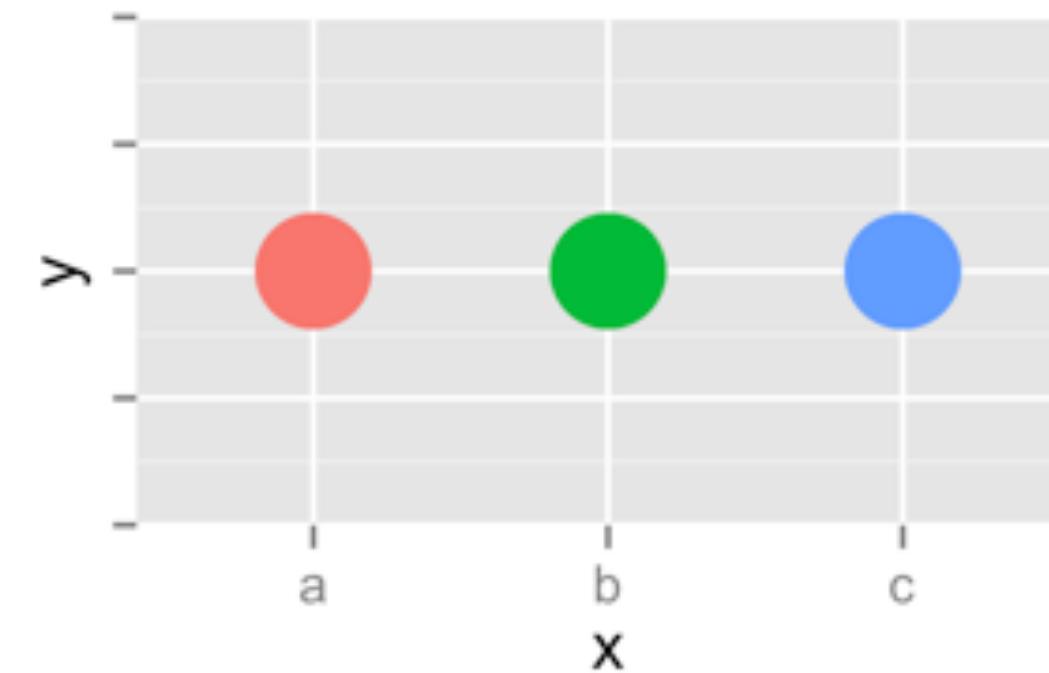
Your Turn 2

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

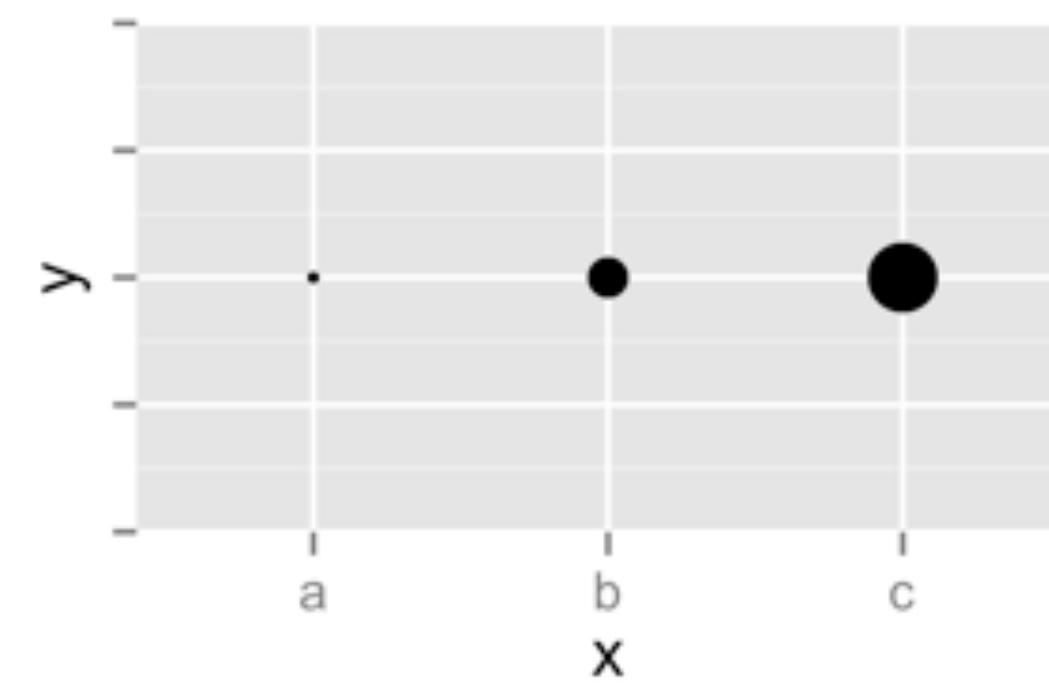
- Do different things happen when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?



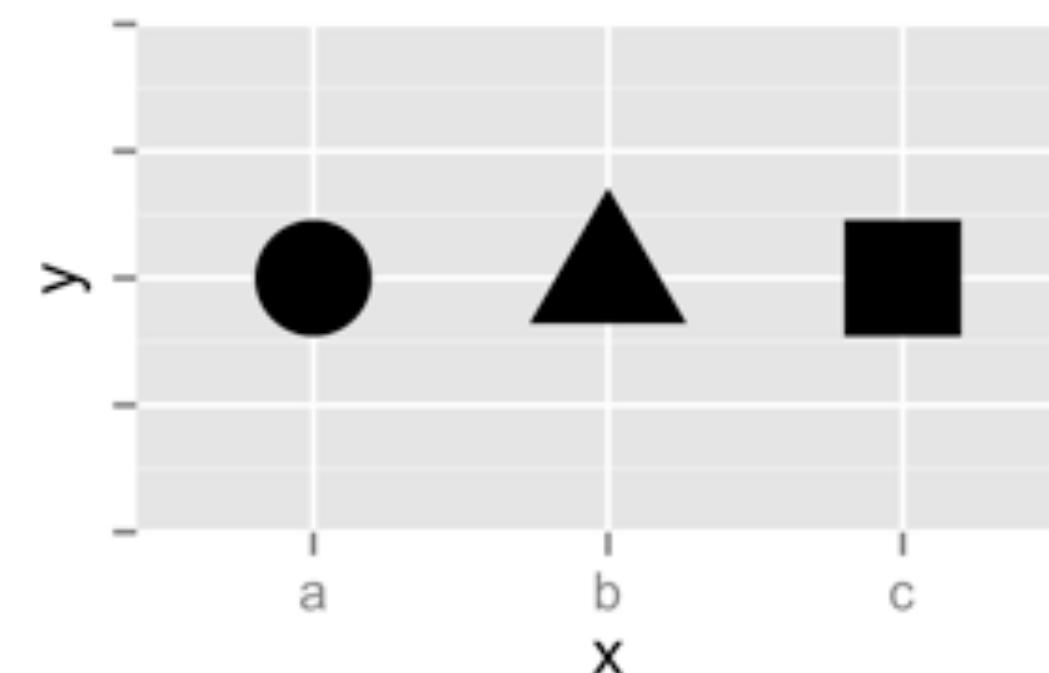
Color



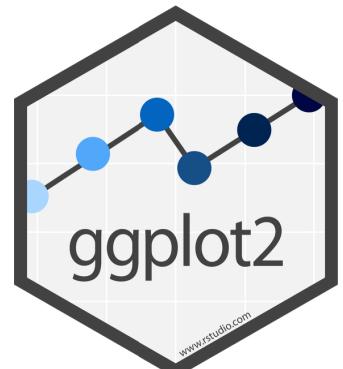
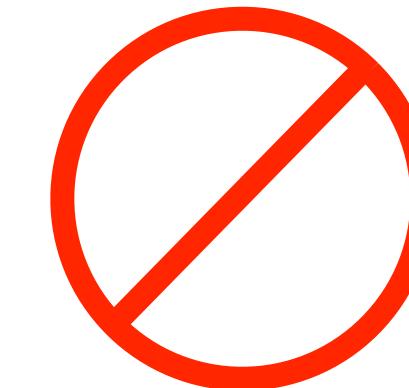
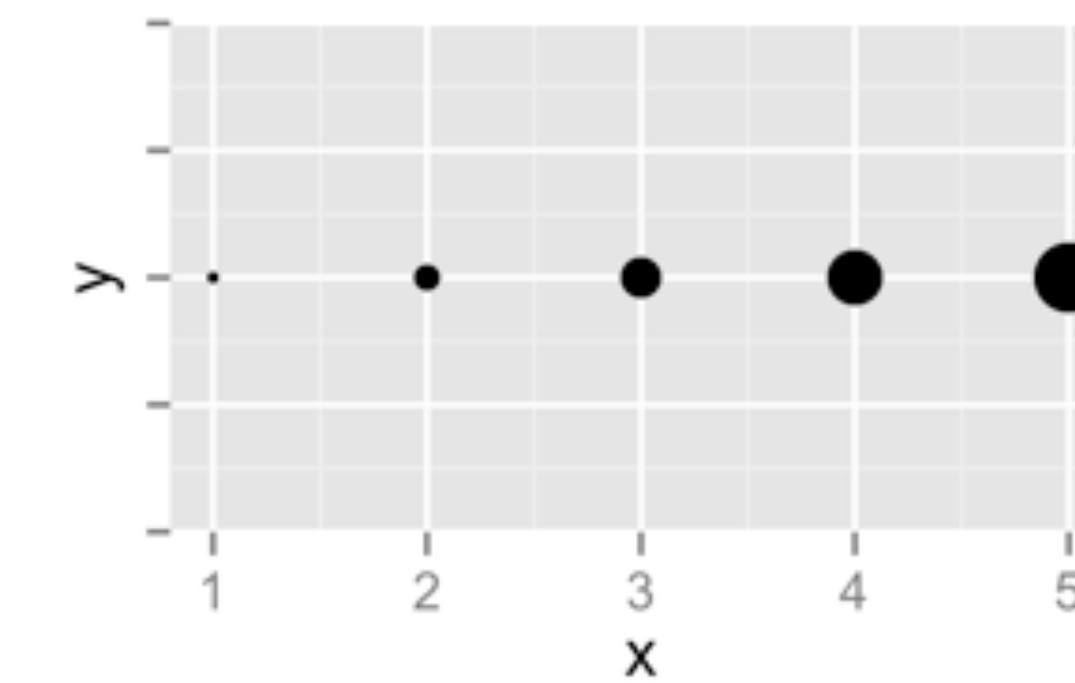
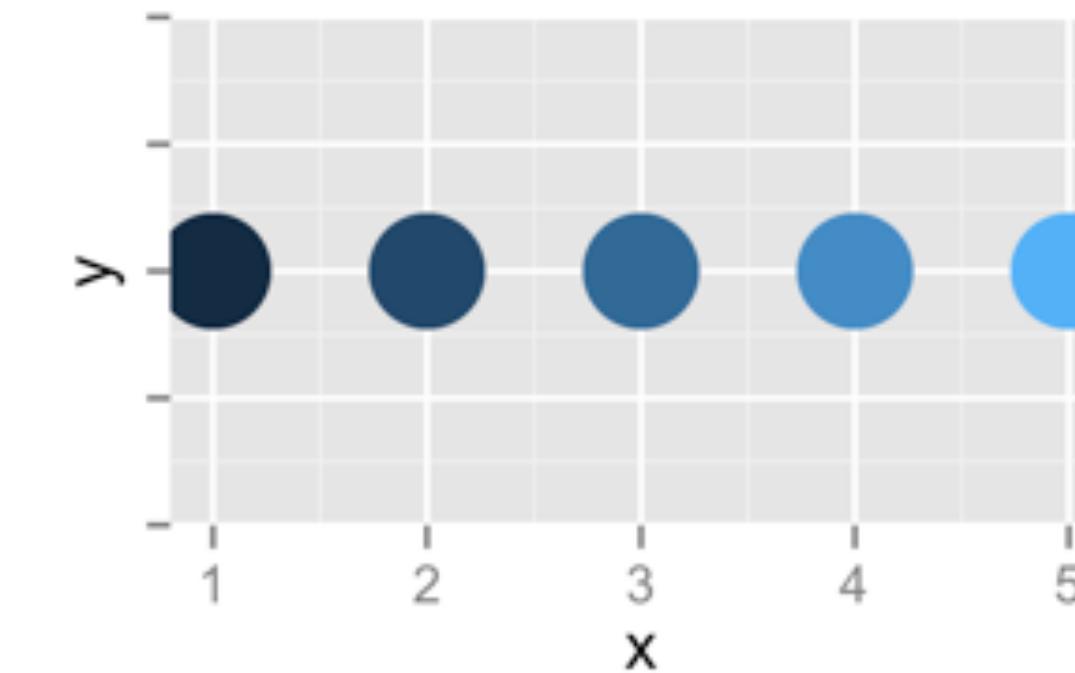
Size



Shape

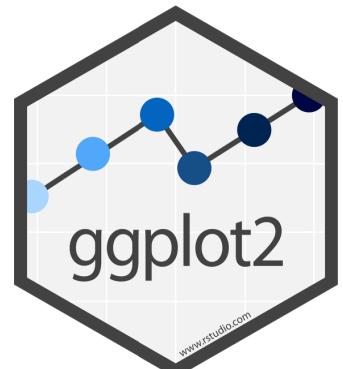
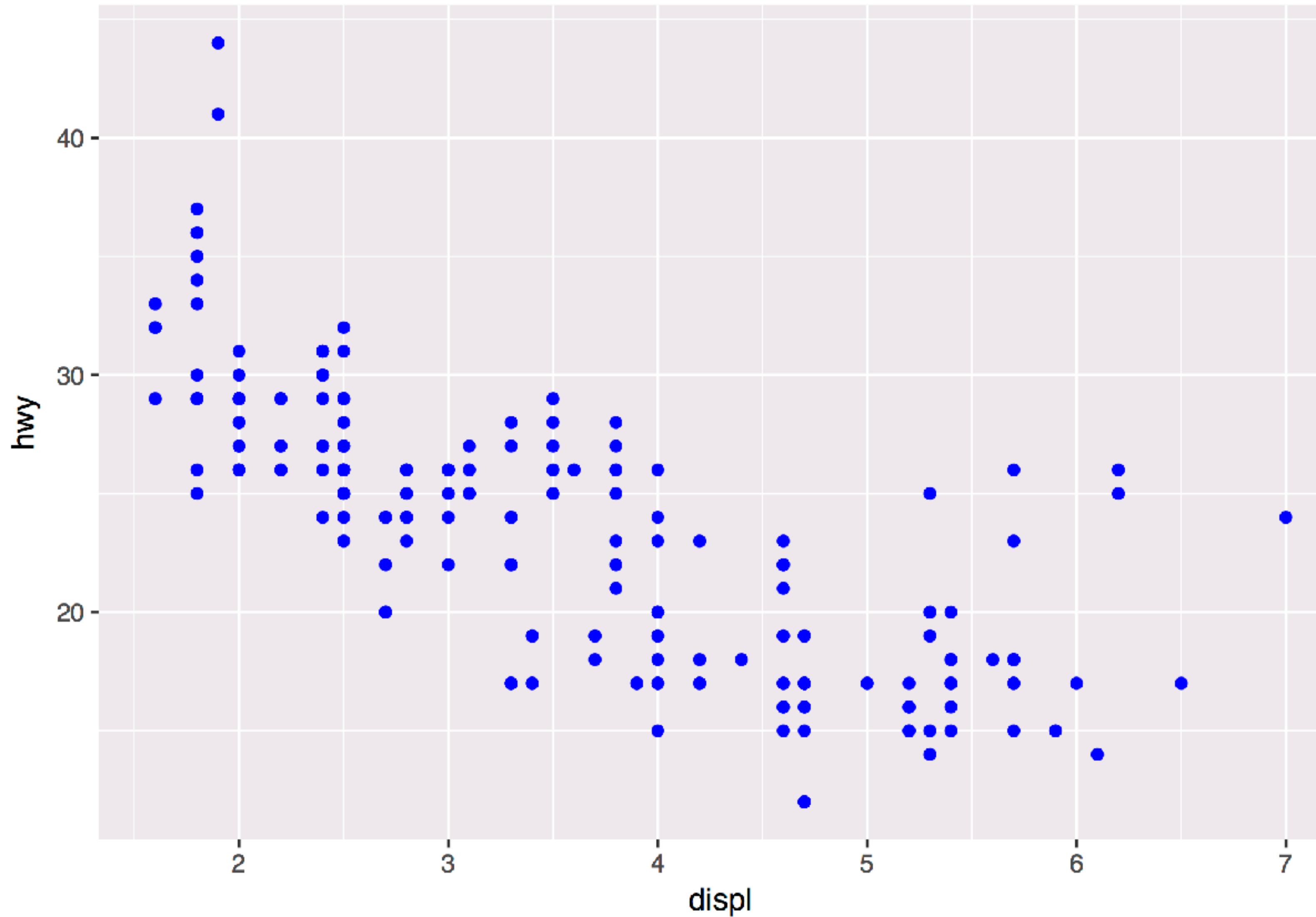


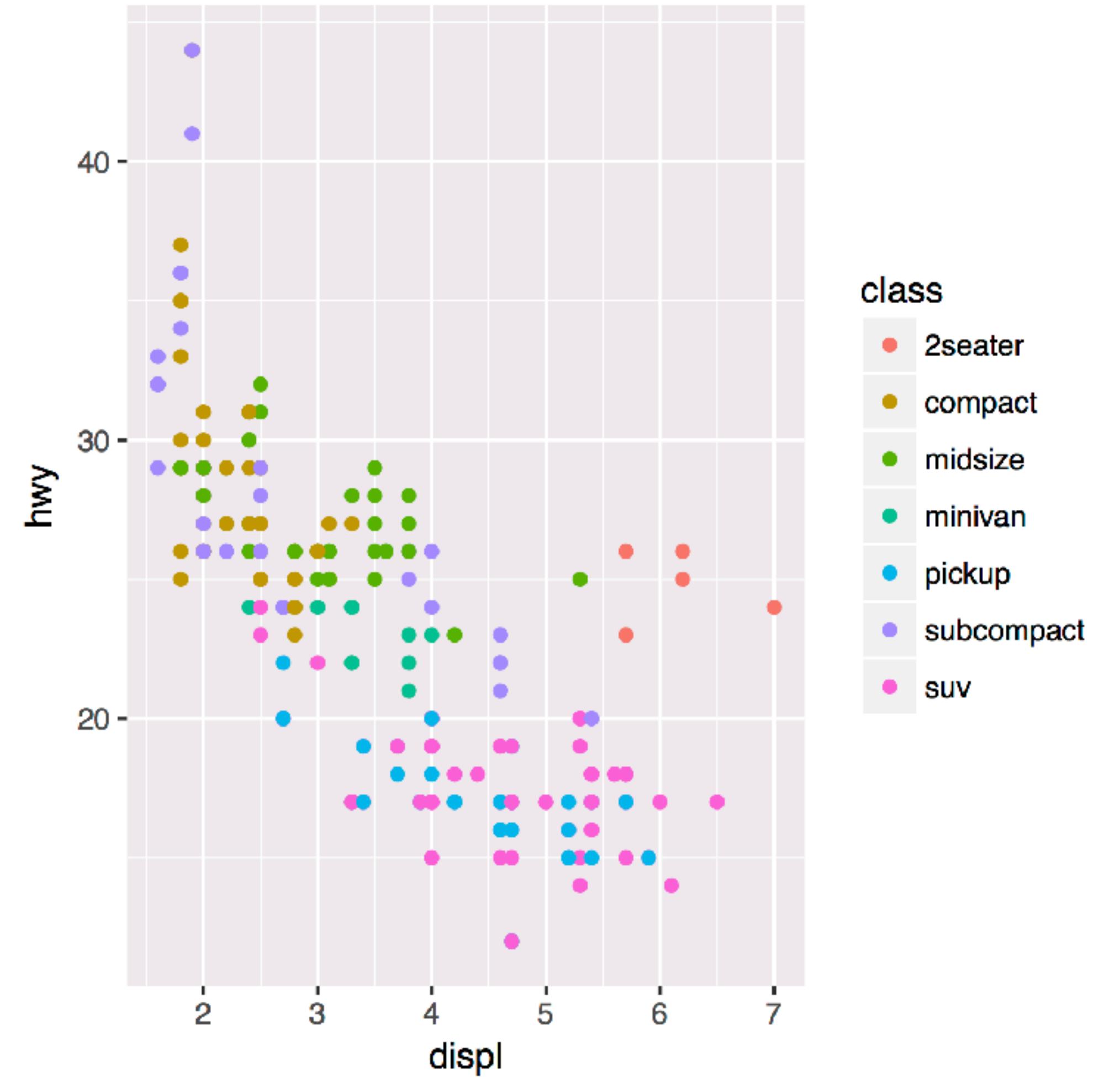
Continuous



Set vs. Map

How would you make this plot?



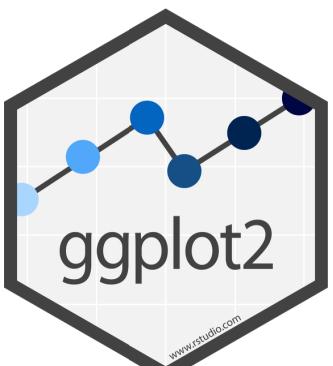


class

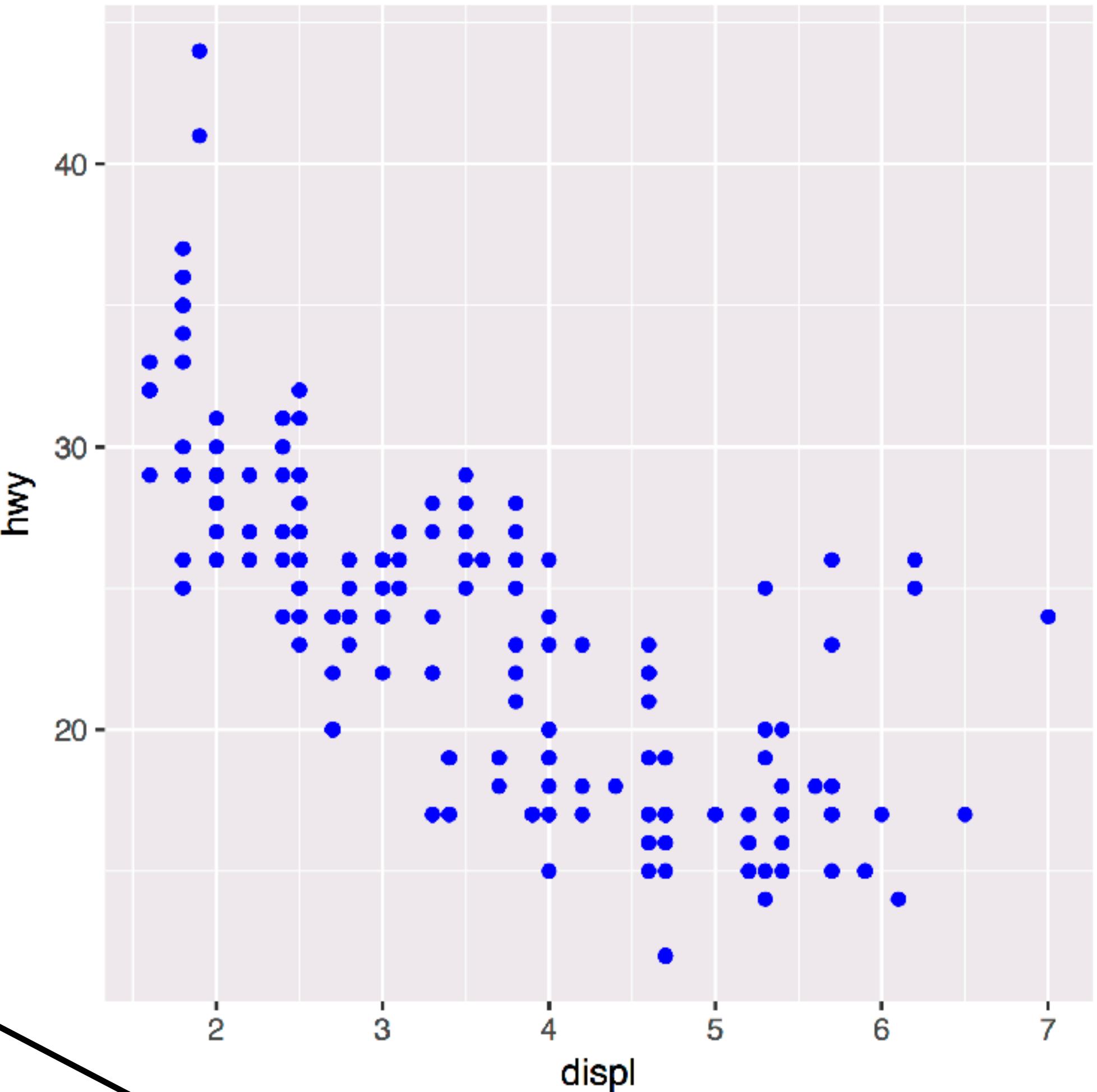
- 2seater
- compact
- midsize
- minivan
- pickup
- subcompact
- SUV

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

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

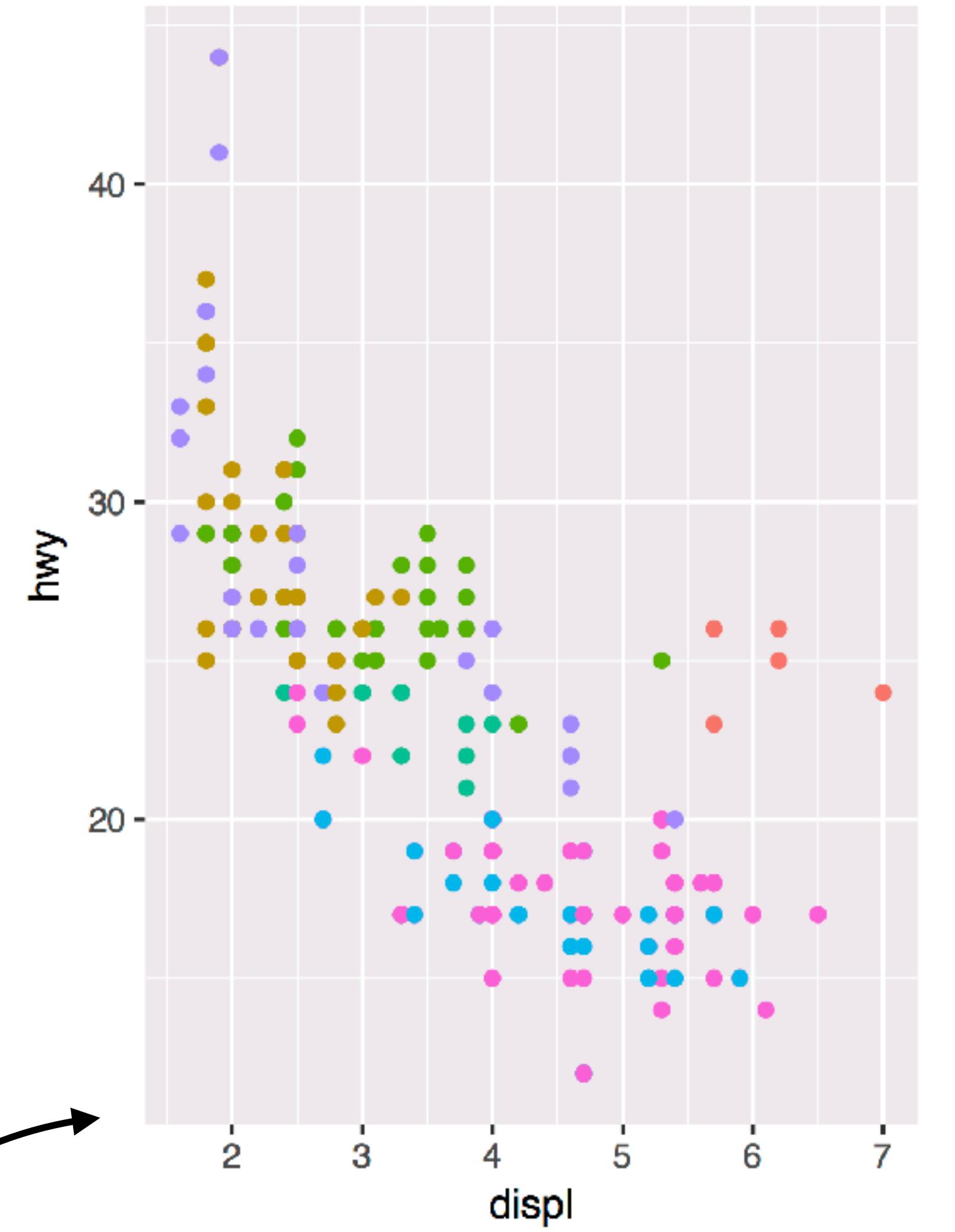


Outside of aes(): sets
an aesthetic to a value



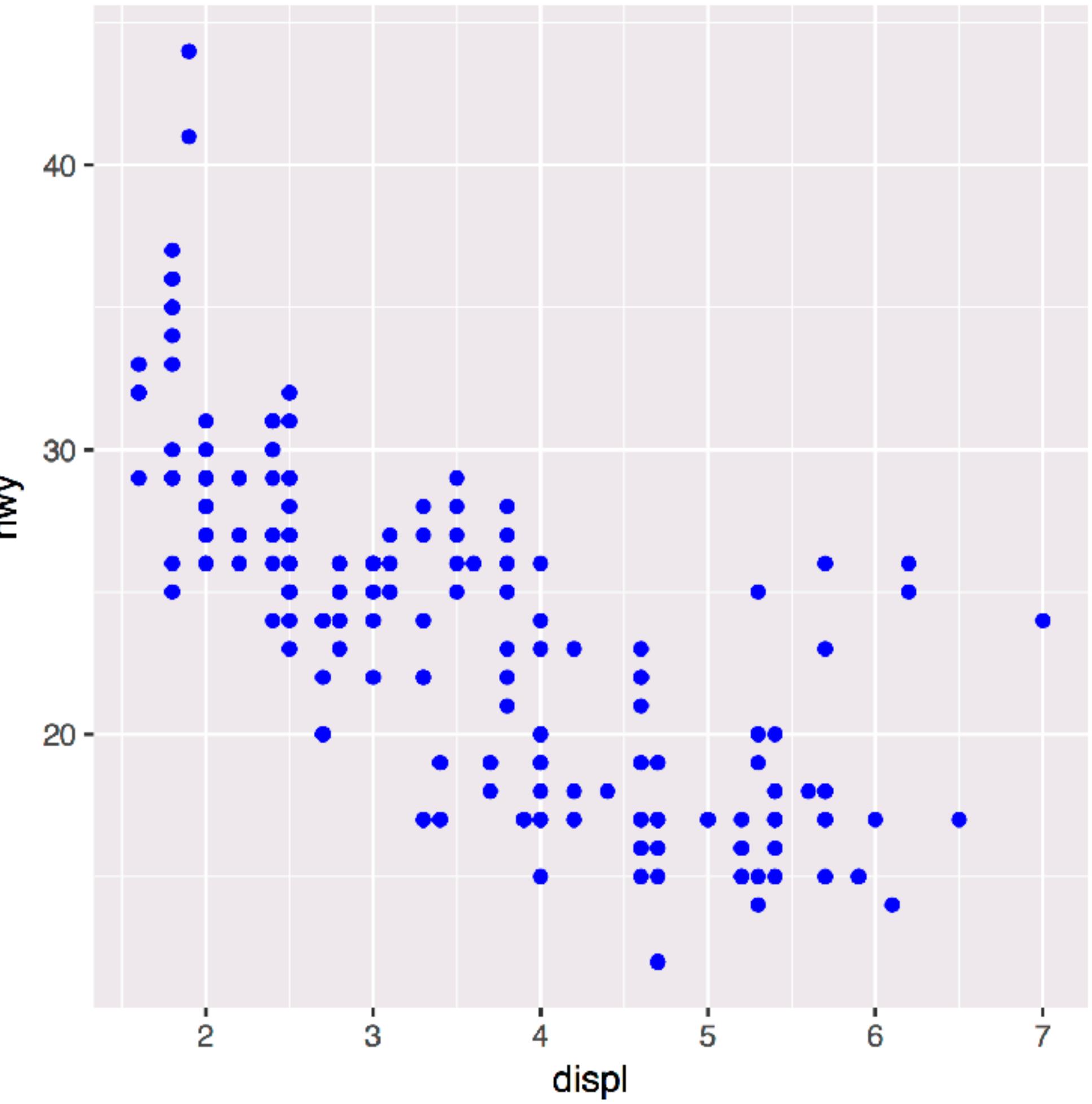
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



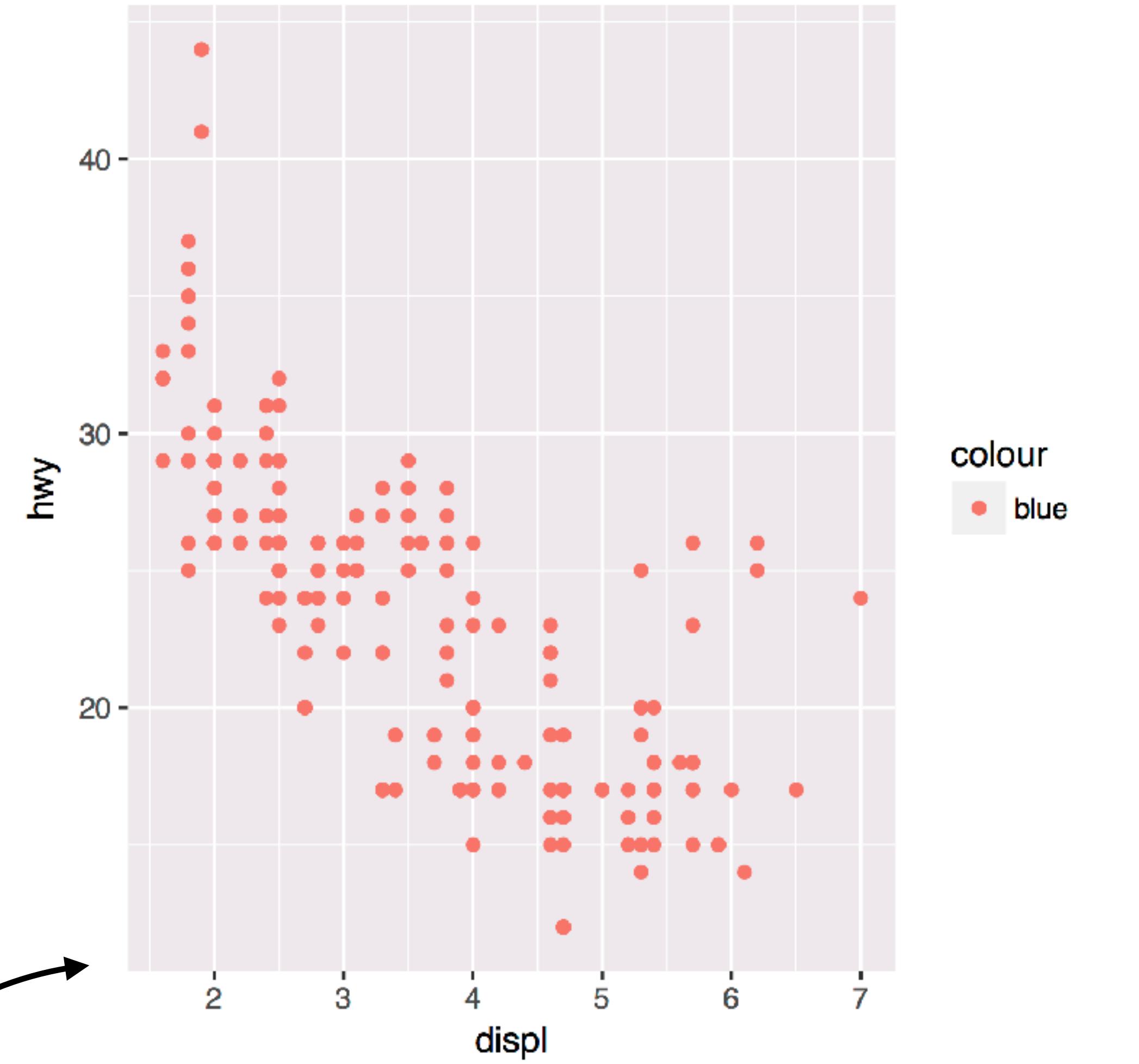
class

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

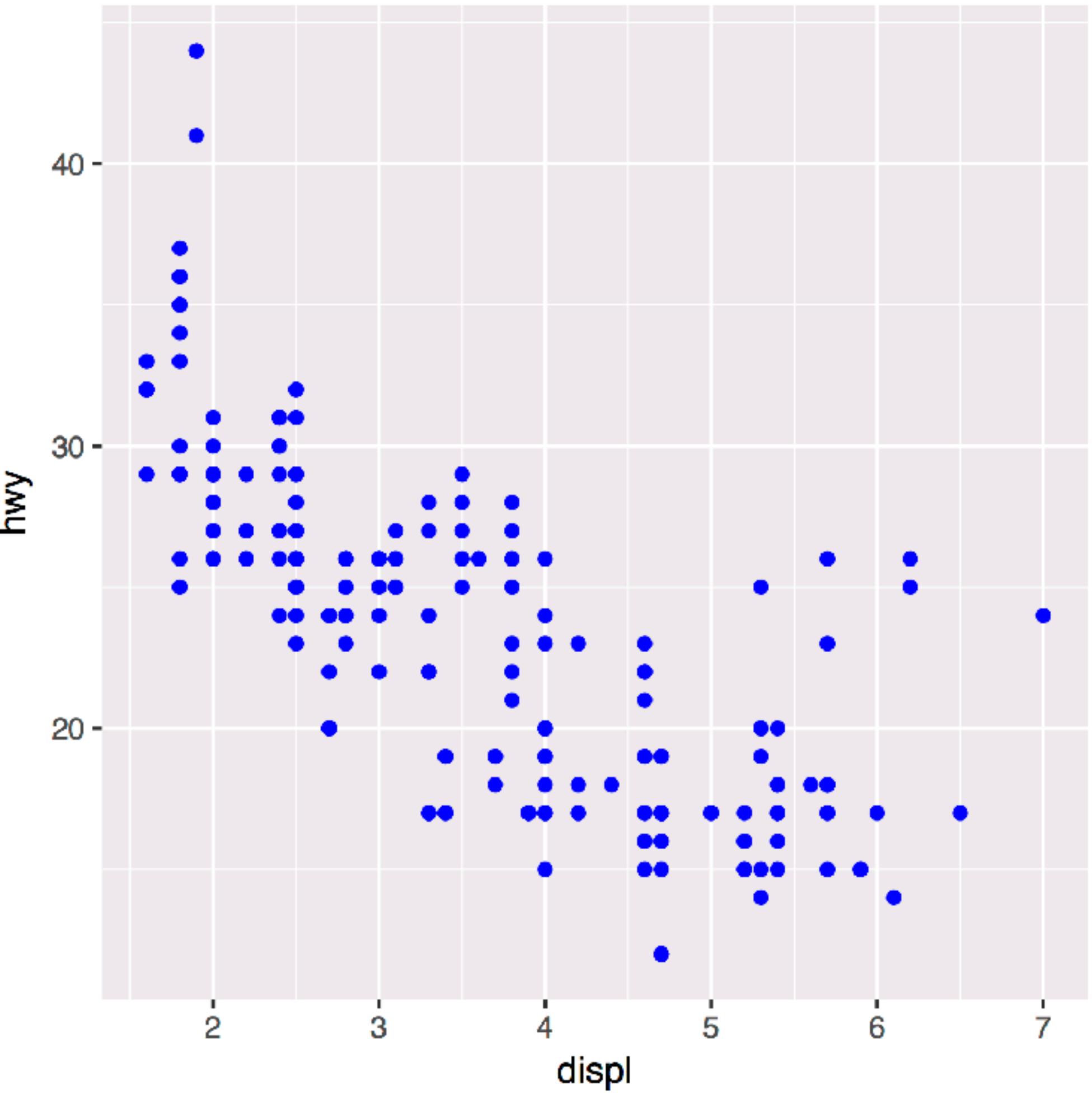


```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



colour
● blue



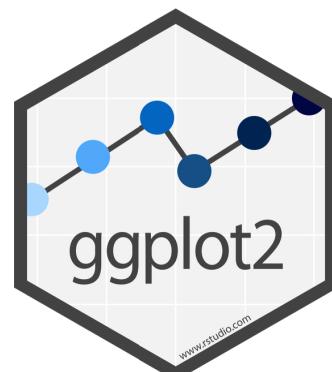
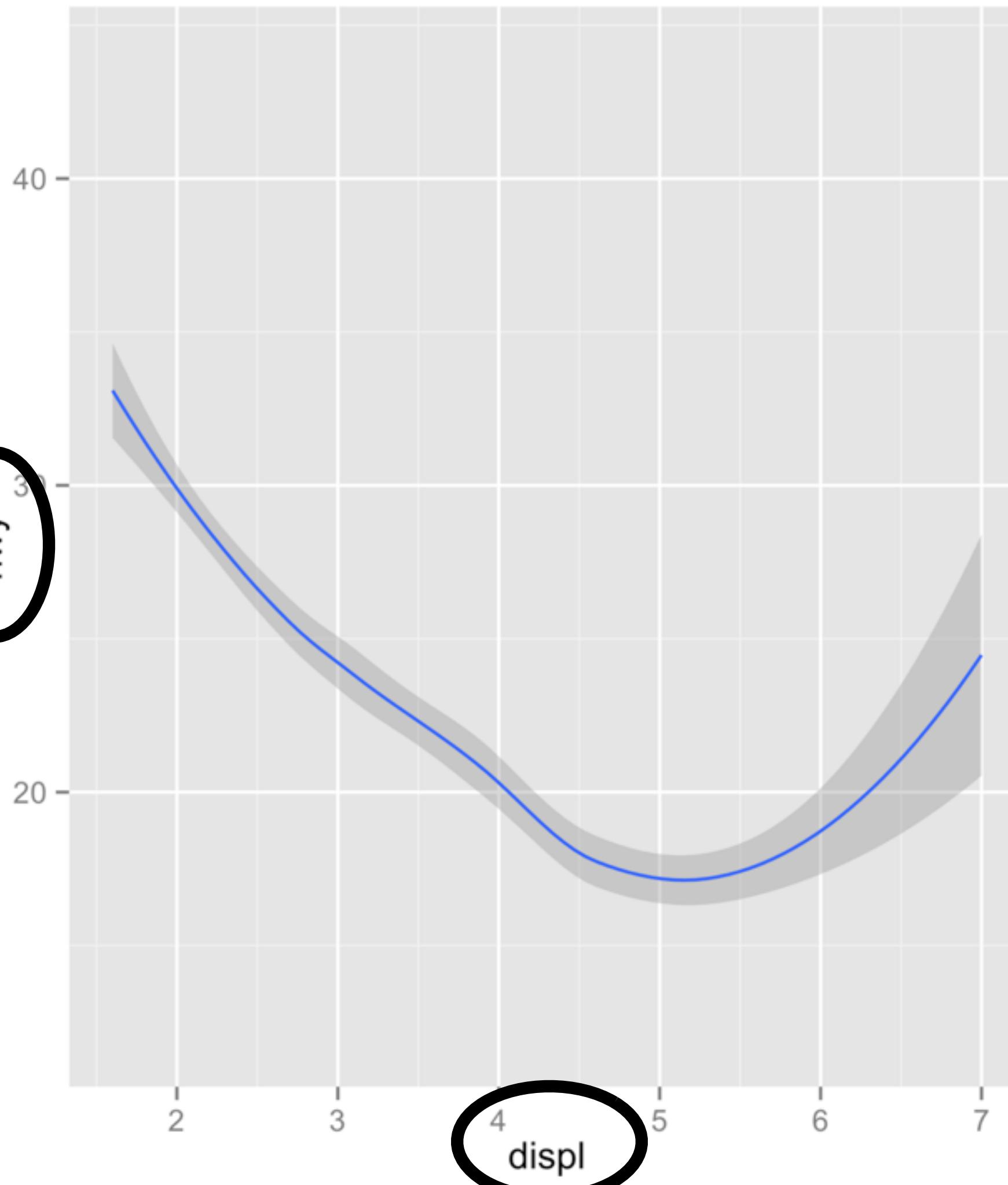
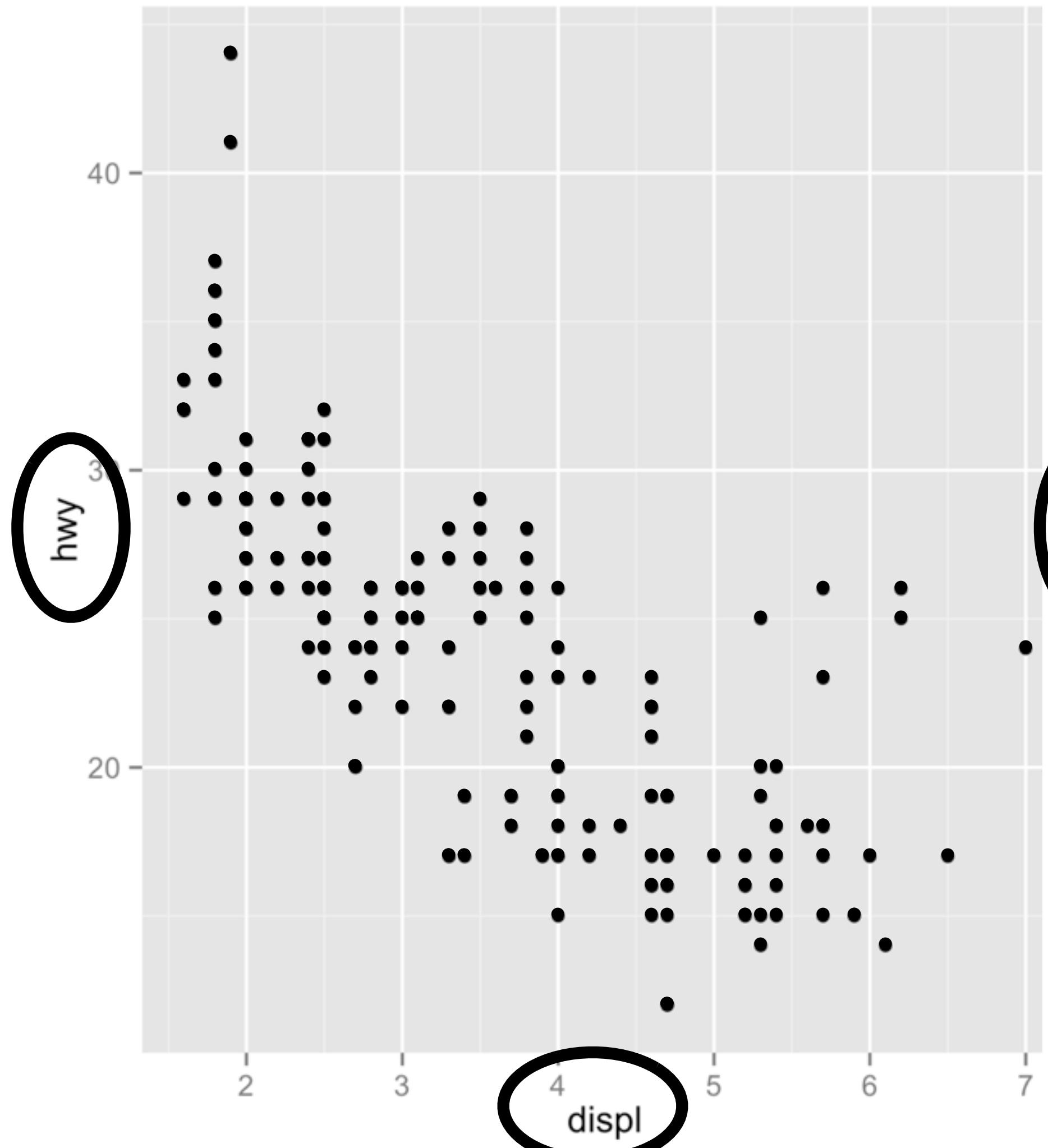
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = "blue"))
```

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

Geoms

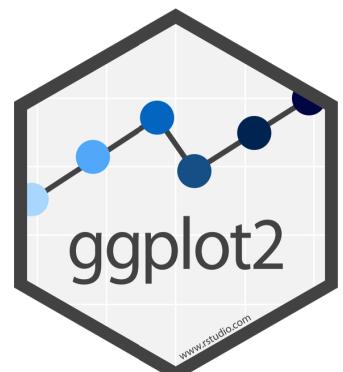
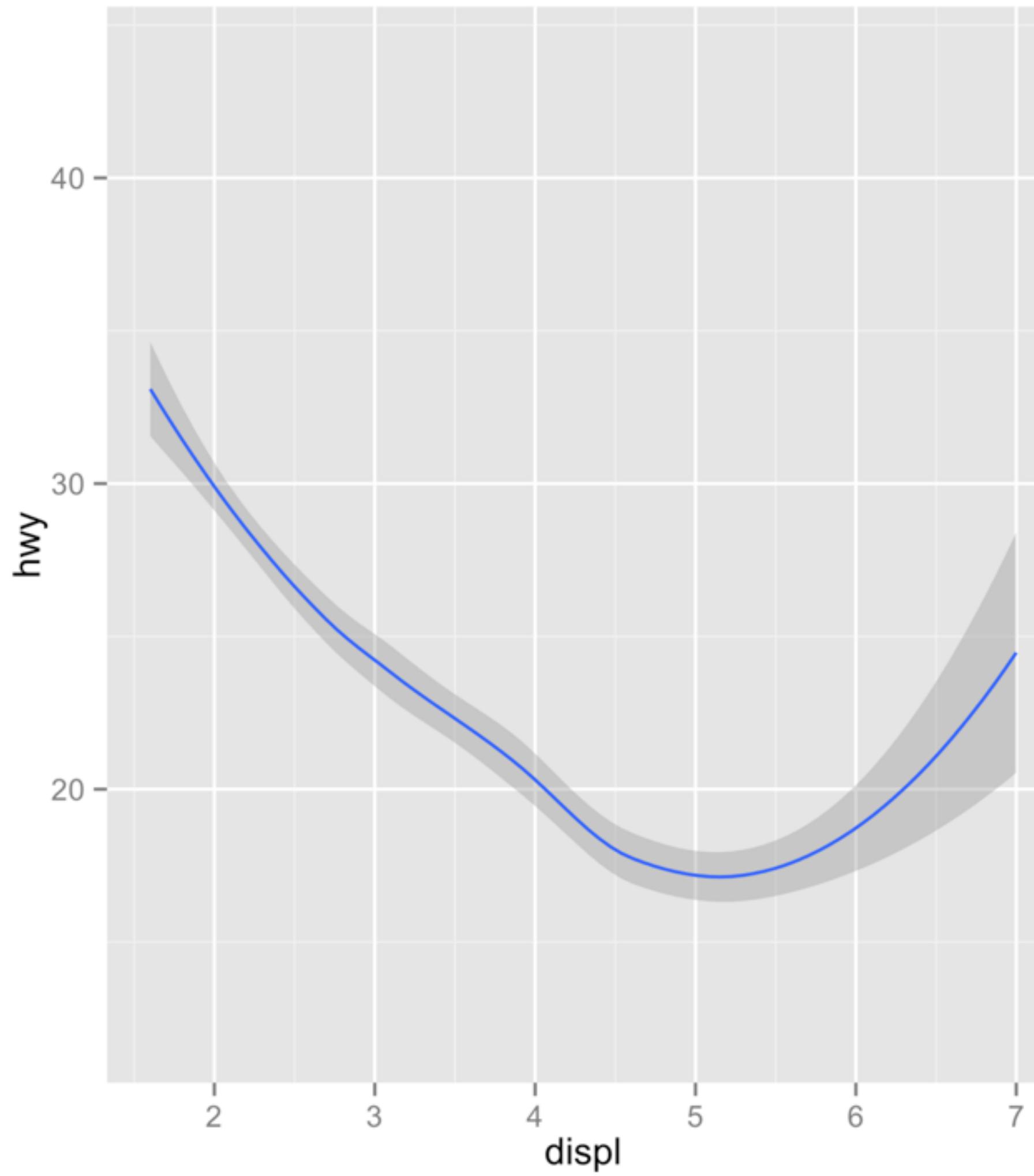
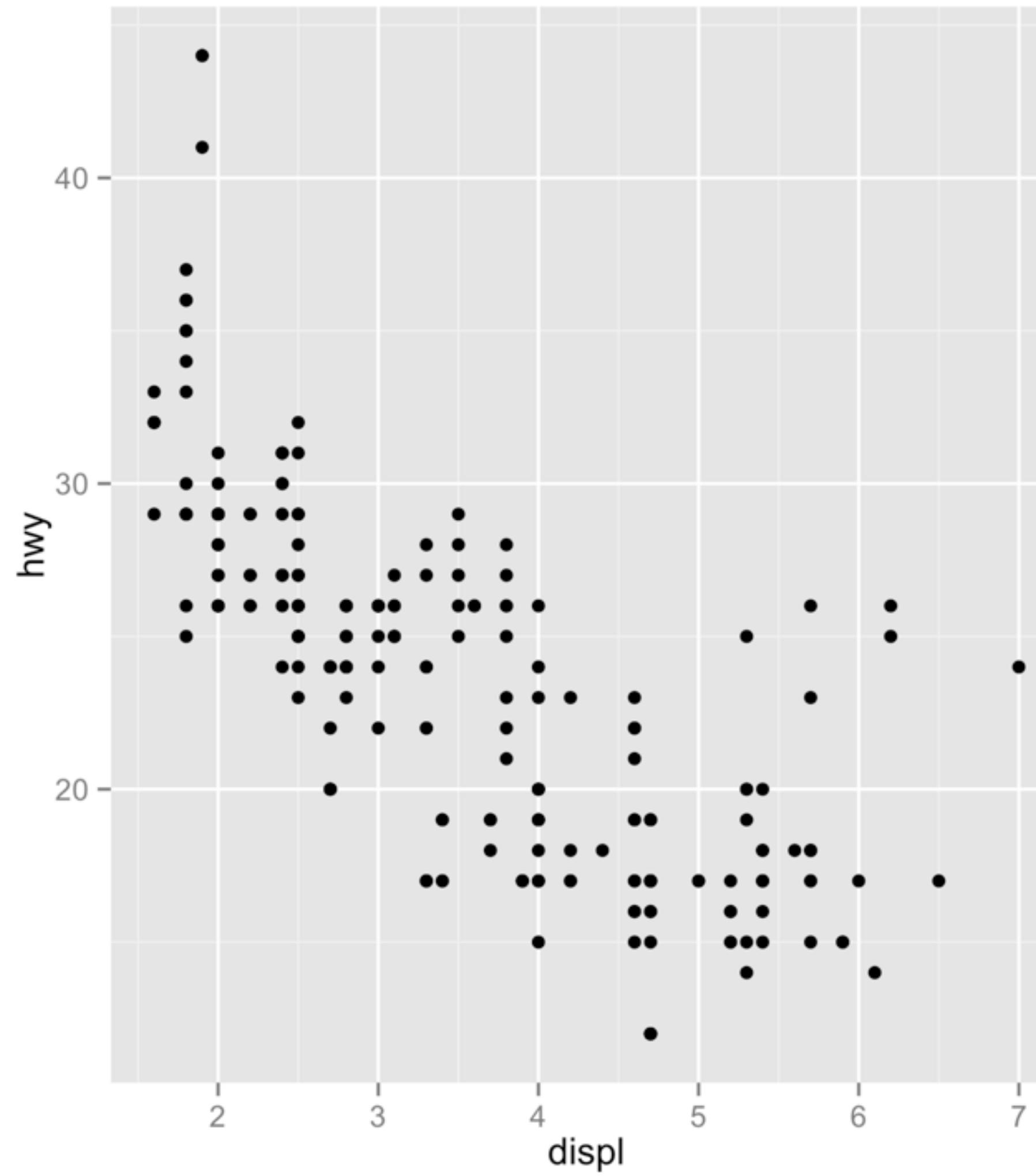
How are these plots similar?

Same: x var , y var , data



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>))
```

Every geom requires a mapping argument.

Data Visualization with ggplot2 :: CHEAT SHEET

Basics

ggplot() based on the grammar of graphics, the idea that you can build every graph the same way, using a coordinate system, and then add marks that represent data points.

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

Complete the required bits to build a graph:

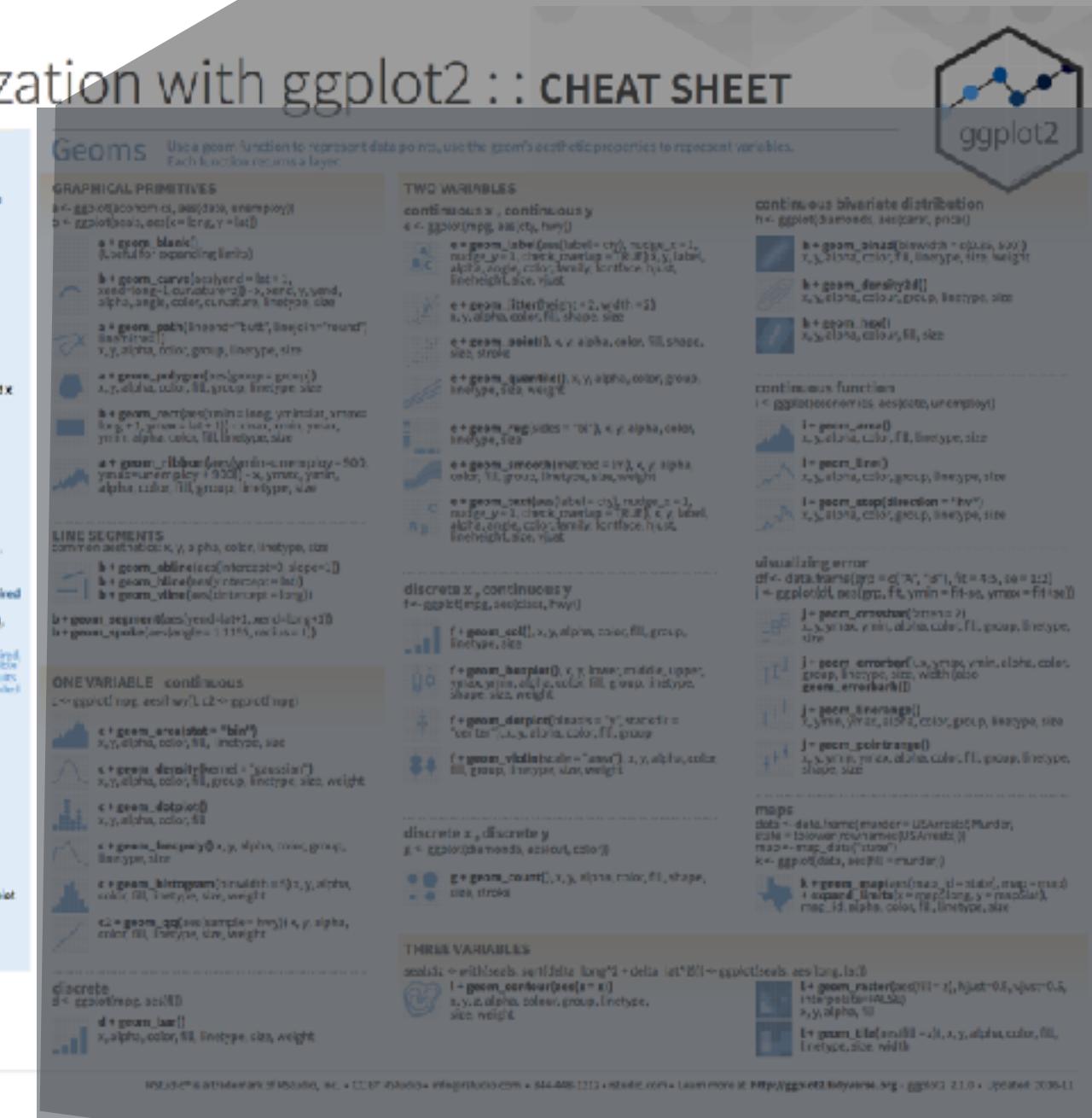
- geom (data = DATA) +
- geom function (mapping = aes(...)) +
- stat (stat = "identity") +
- continuous x, continuous y +
- discrete x, continuous y +
- one variable +
- three variables +

ggplot(data = mpg, aes(x=displ, y=hwy)) # Create a complete plot with given data, geom, and mapping. Supplies many useful defaults.

au_hex() # Returns the hex grid

ggplot(hexgrid, width = 5, height = 5). # Generates a 5x5 hexagonal grid. Matches file types to file extensions.

R Studio



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 = 0)) - x, xend, y, yend, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

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, ymn, 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(yintercept = 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, 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

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

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 = "y", stackdir = "center") x, y, alpha, color, fill, group

f + geom_violin(scale = "area") x, y, alpha, color, fill, group, 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

sealsSz <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

l + geom_raster(aes(z = seals\$z))

ggplot2

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, 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

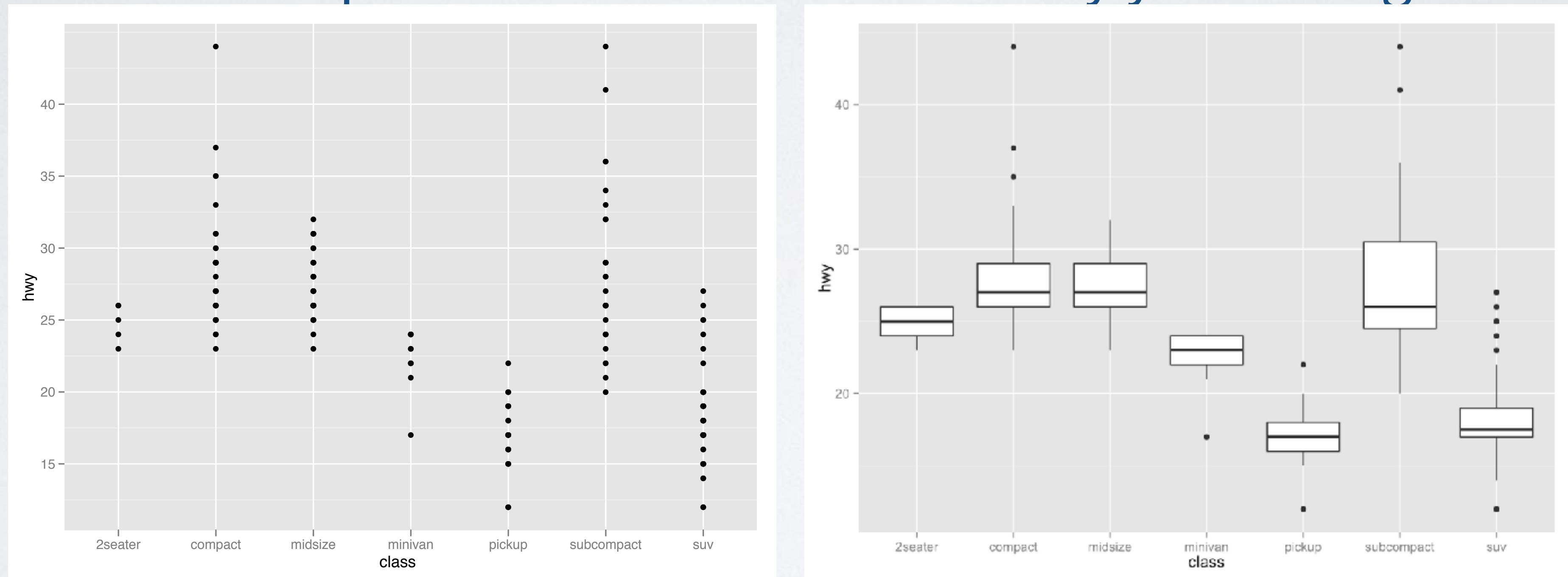
Your Turn

Pair up.



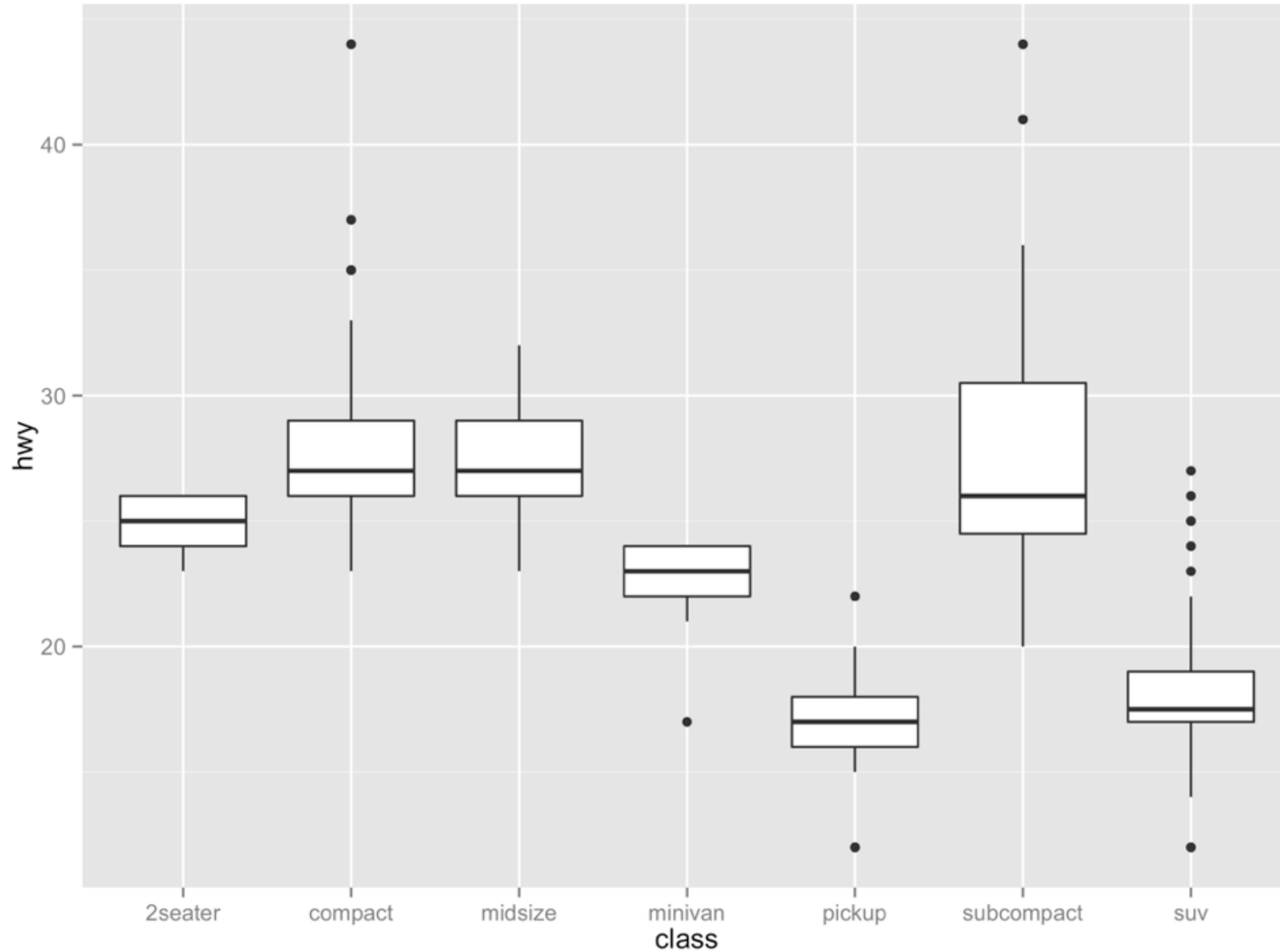
Your Turn 3

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))
```

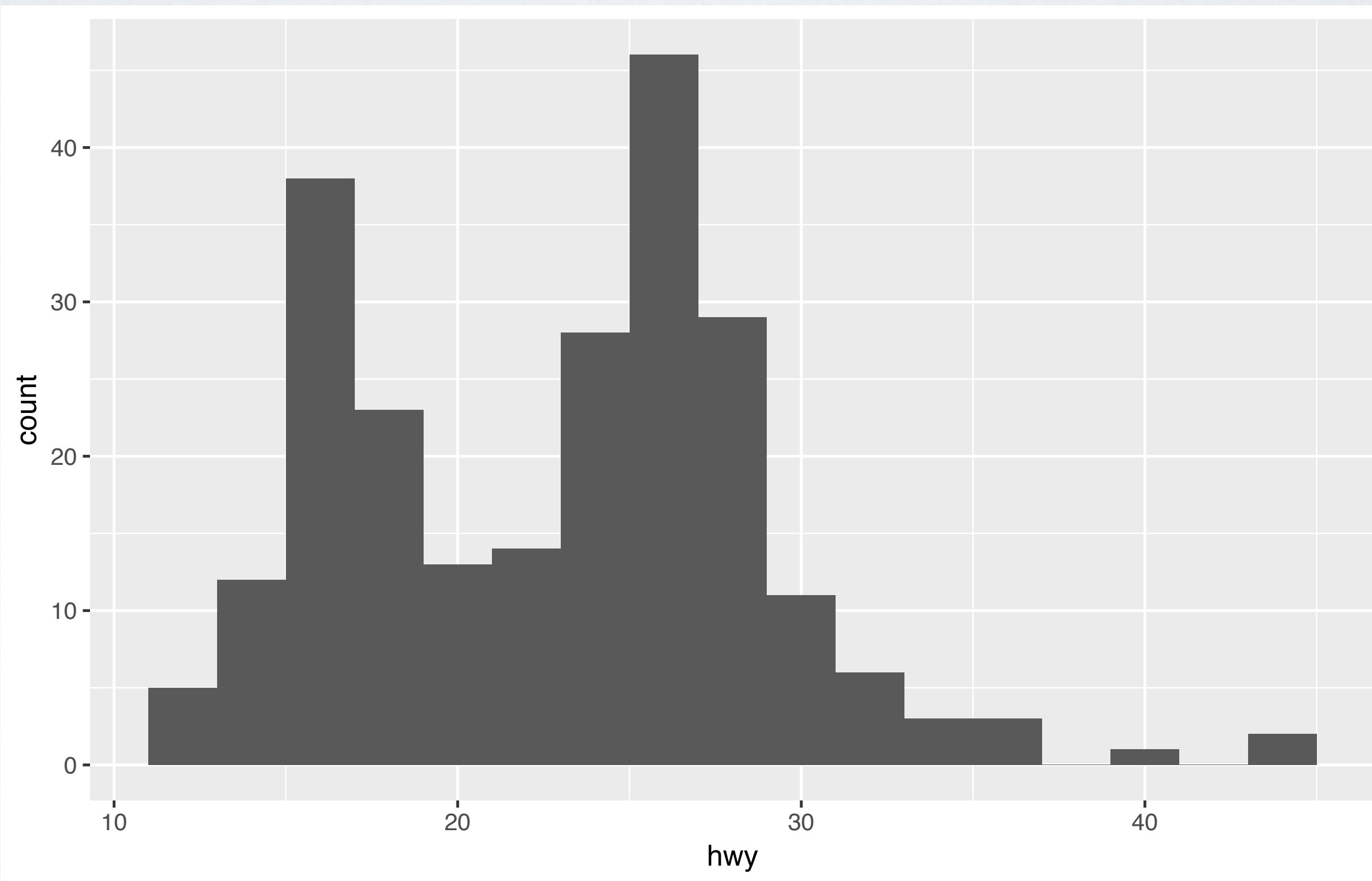
02 : 00

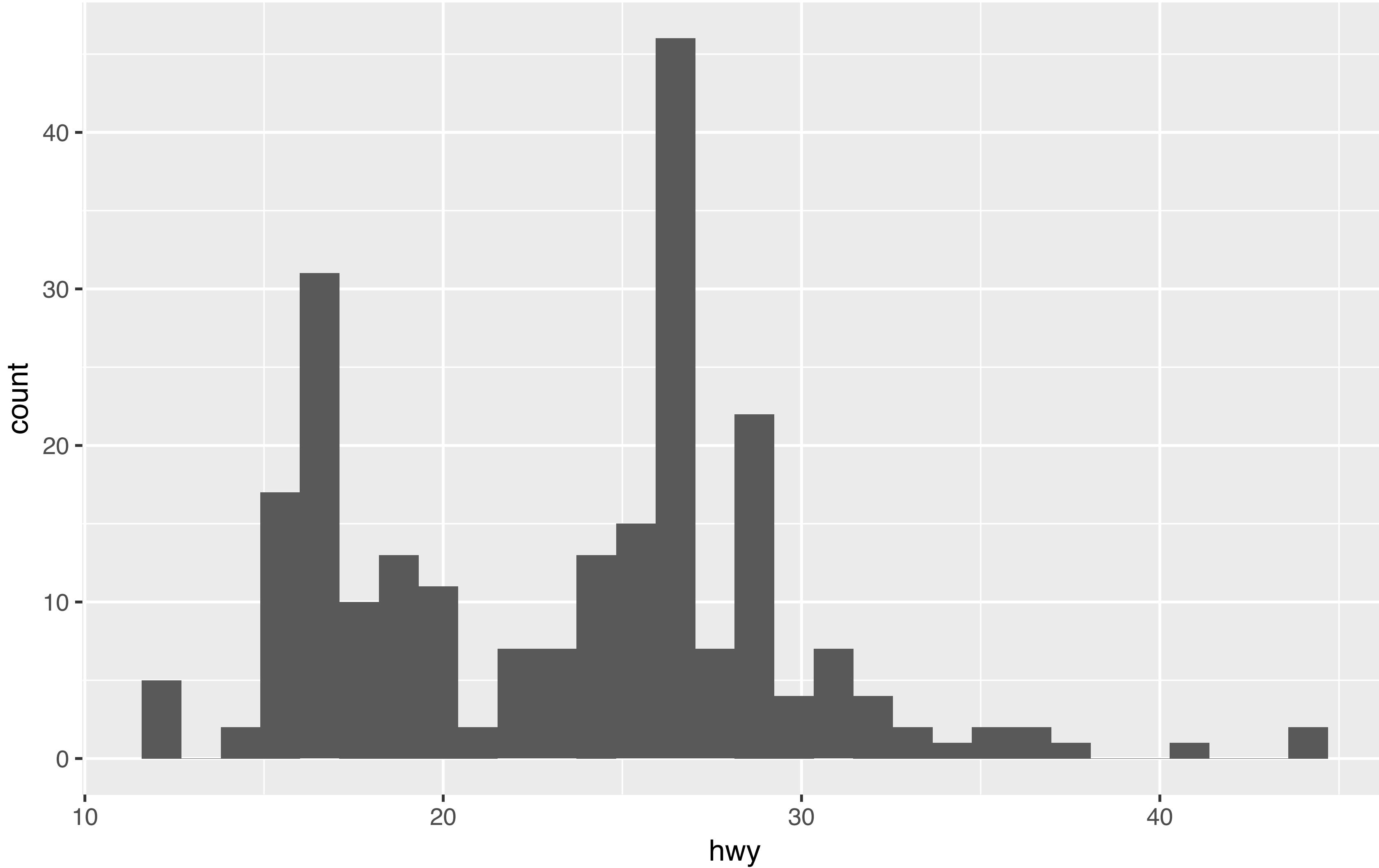


```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

Your Turn 4

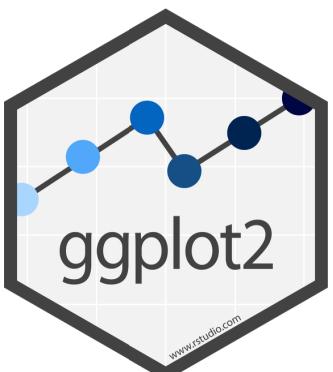
With your partner, make the **histogram** of hwy below. Use the cheatsheet. **Hint:** do not supply a y variable.

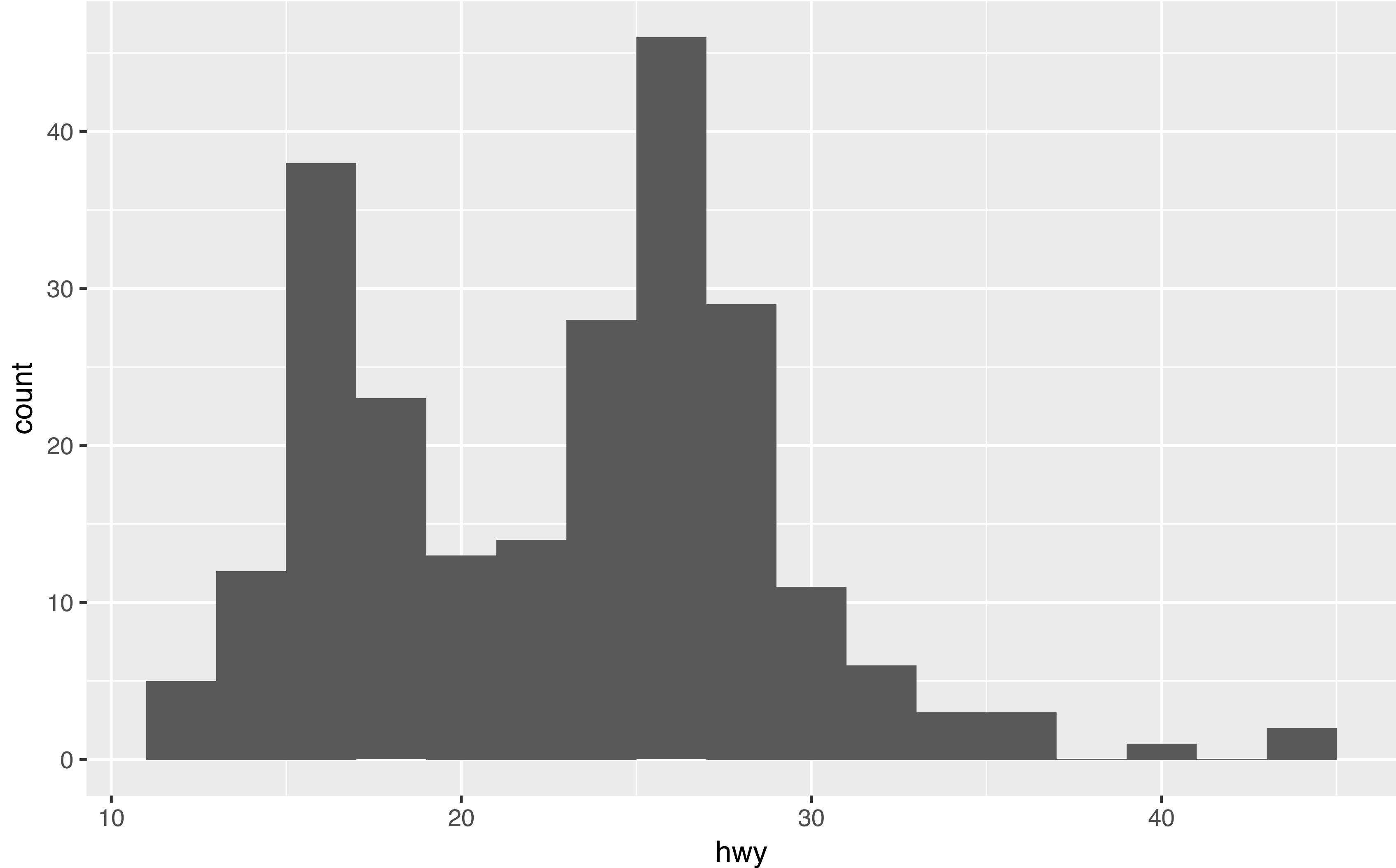




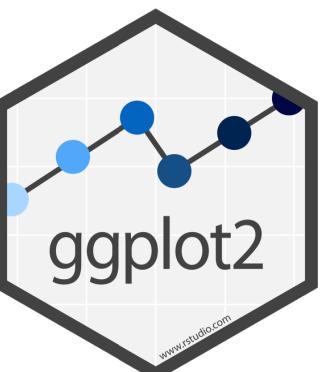
```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy))
```

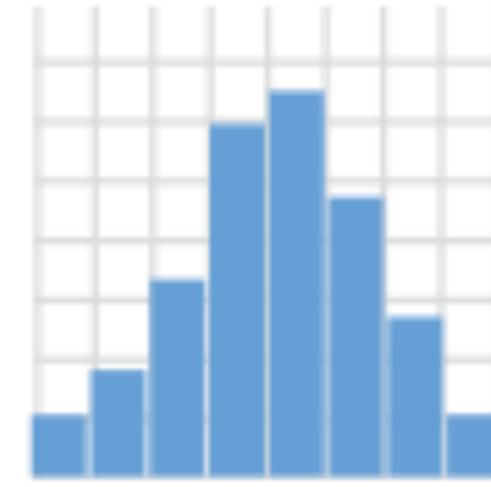
No y aesthetic





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

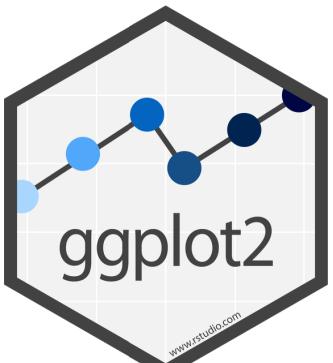




```
c + geom_histogram(binwidth = 5) x, y, alpha,  
color, fill, linetype, size, weight
```

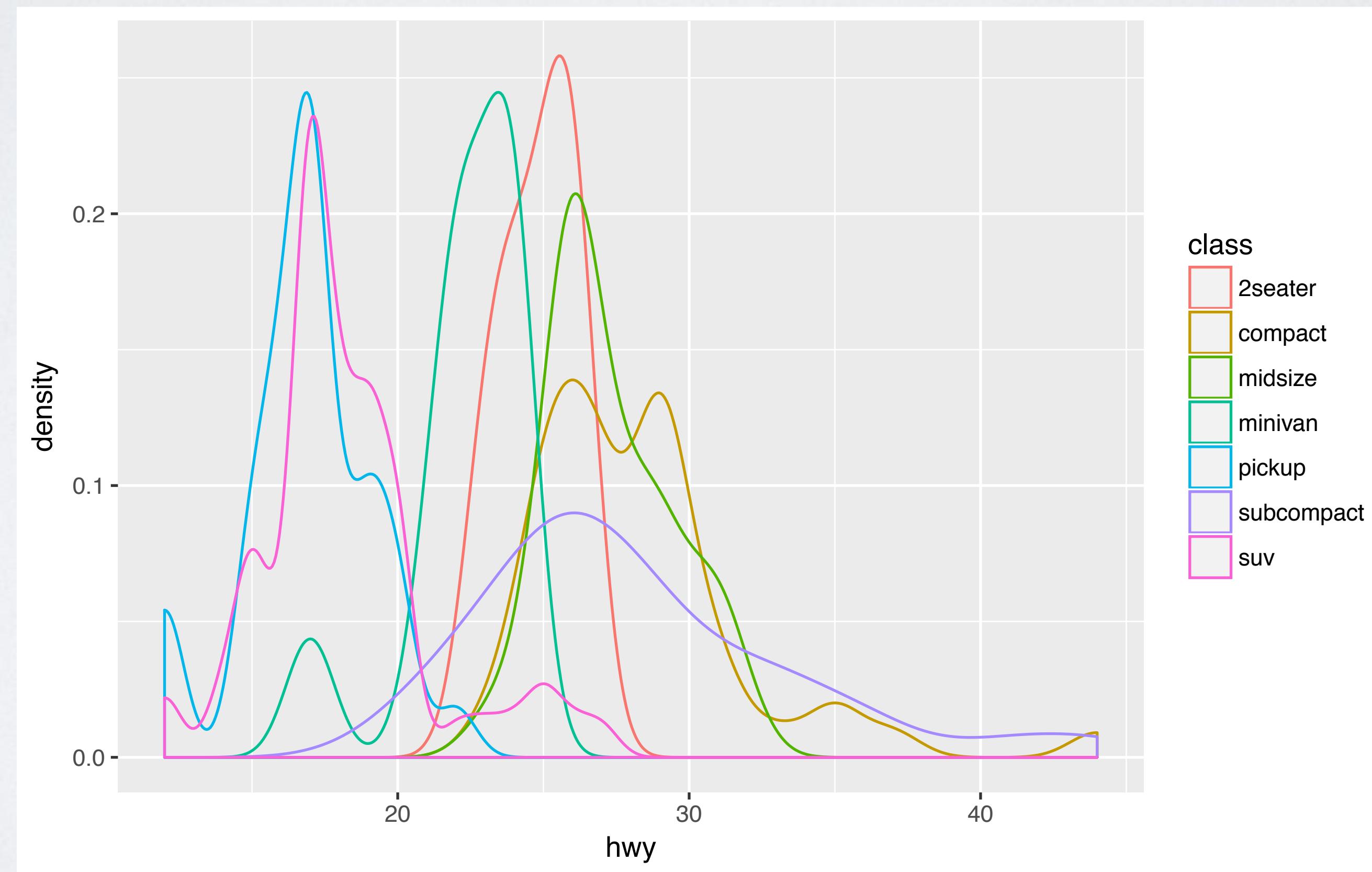
Arguments inside (), are
geom specific options

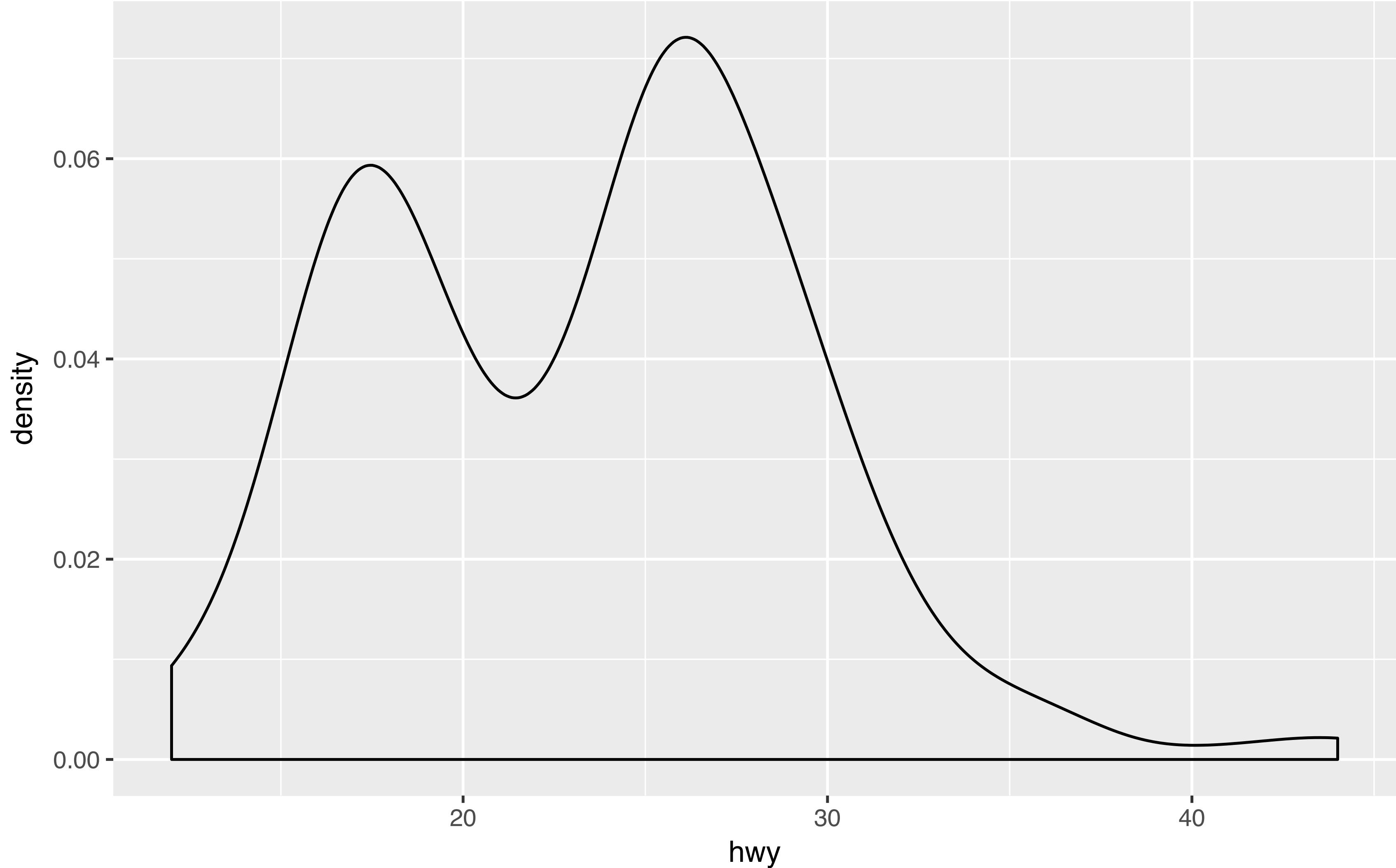
Outside the (), are
aesthetics that can be
mapped or set.



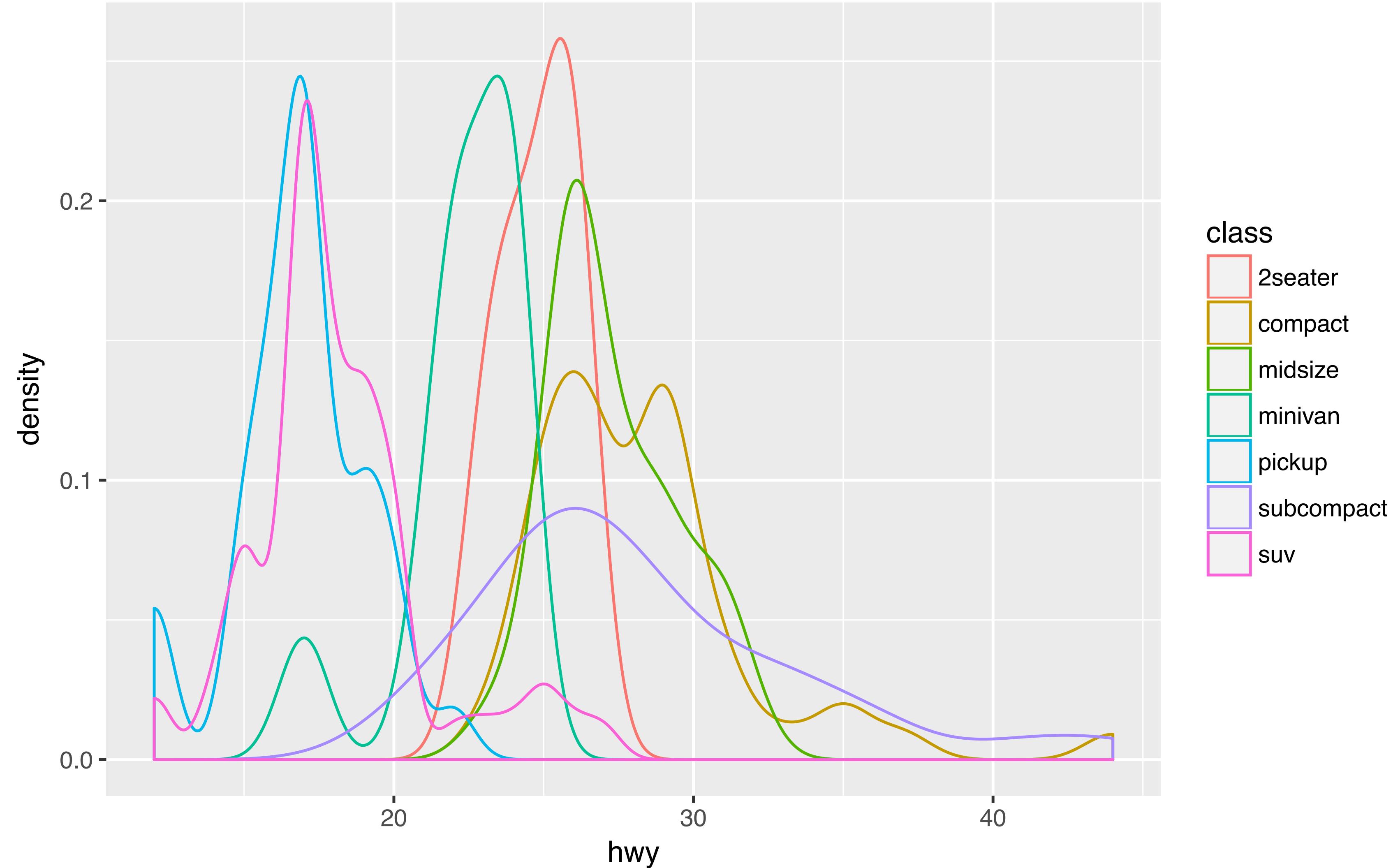
Your Turn 5

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

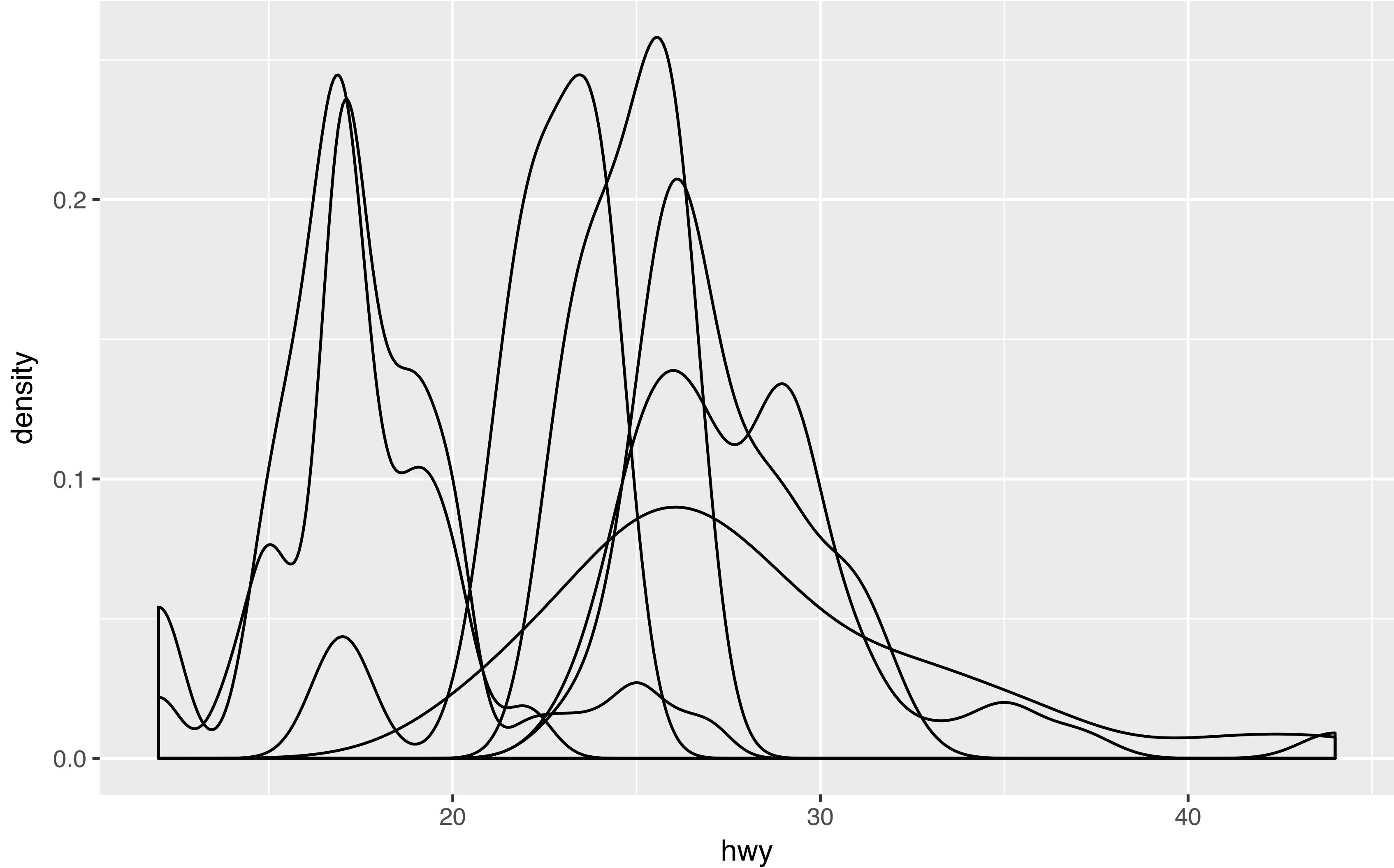




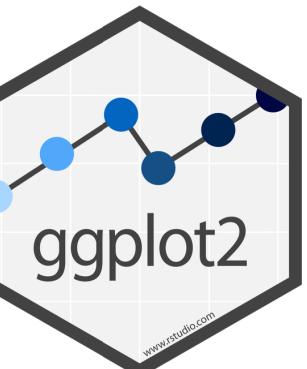
```
ggplot(data = mpg) +  
  geom_density(mapping = aes(x = hwy))
```



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

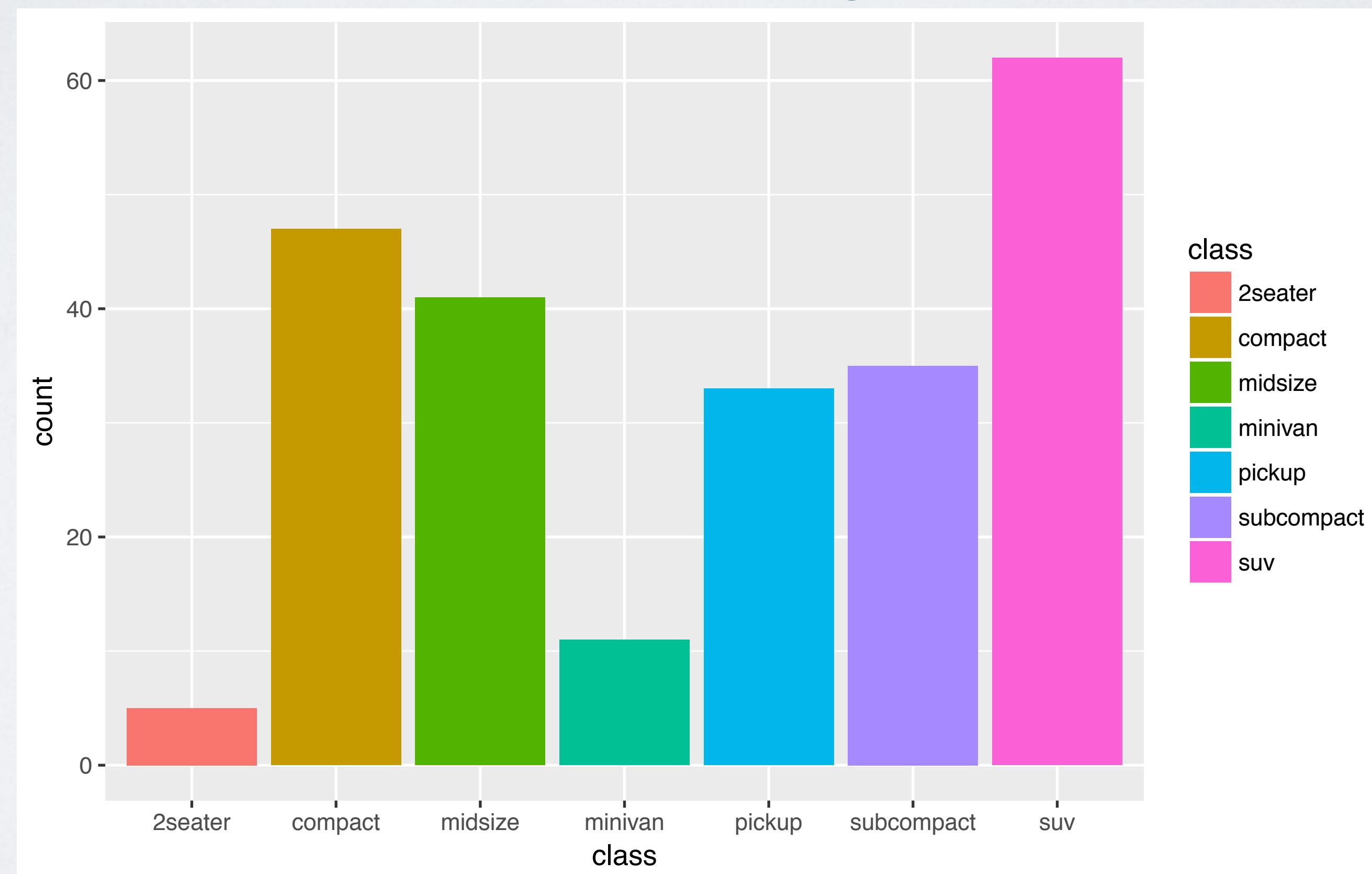


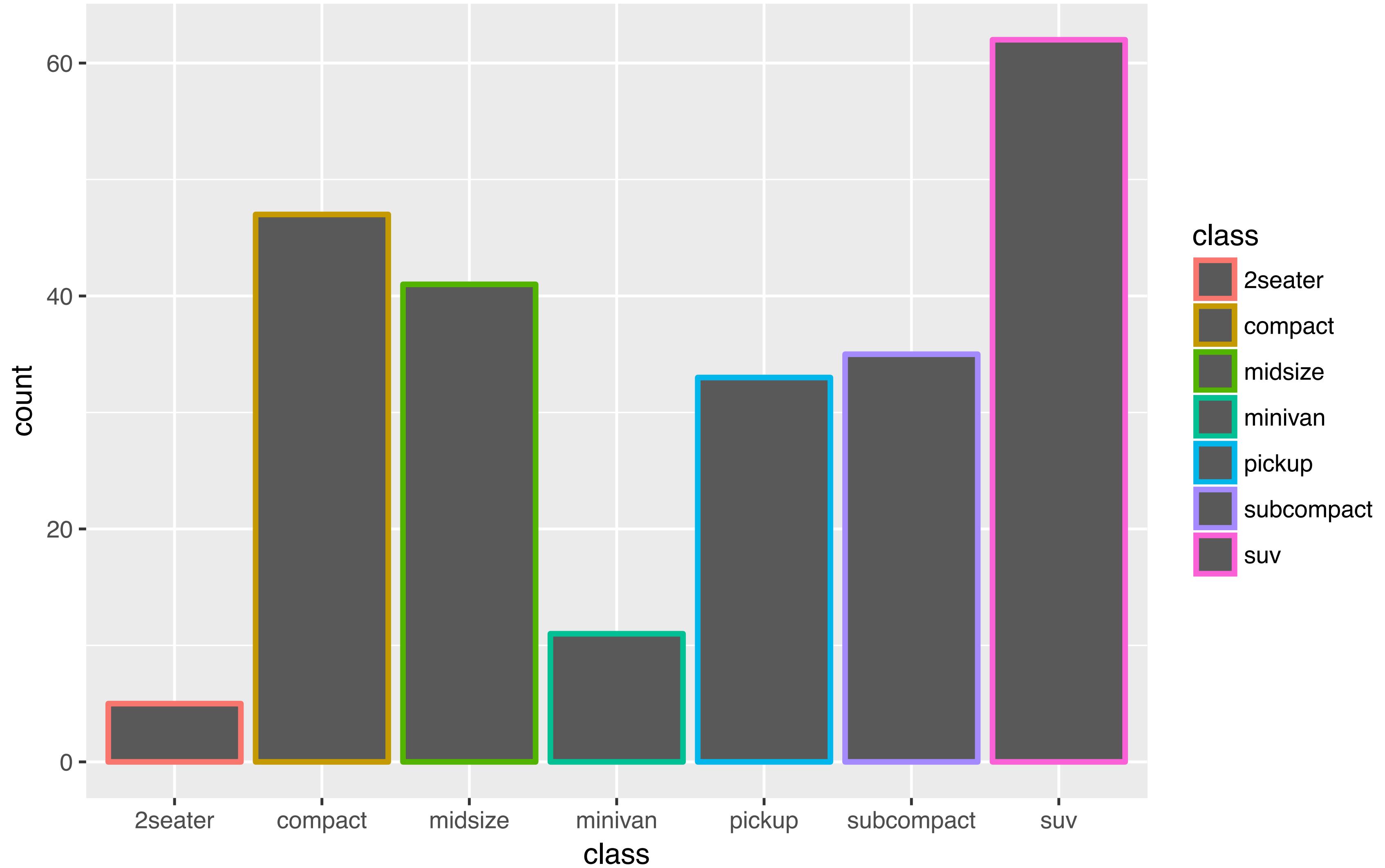
```
ggplot(data = mpg) +  
  geom_density(mapping = aes(x = hwy, group = class))
```



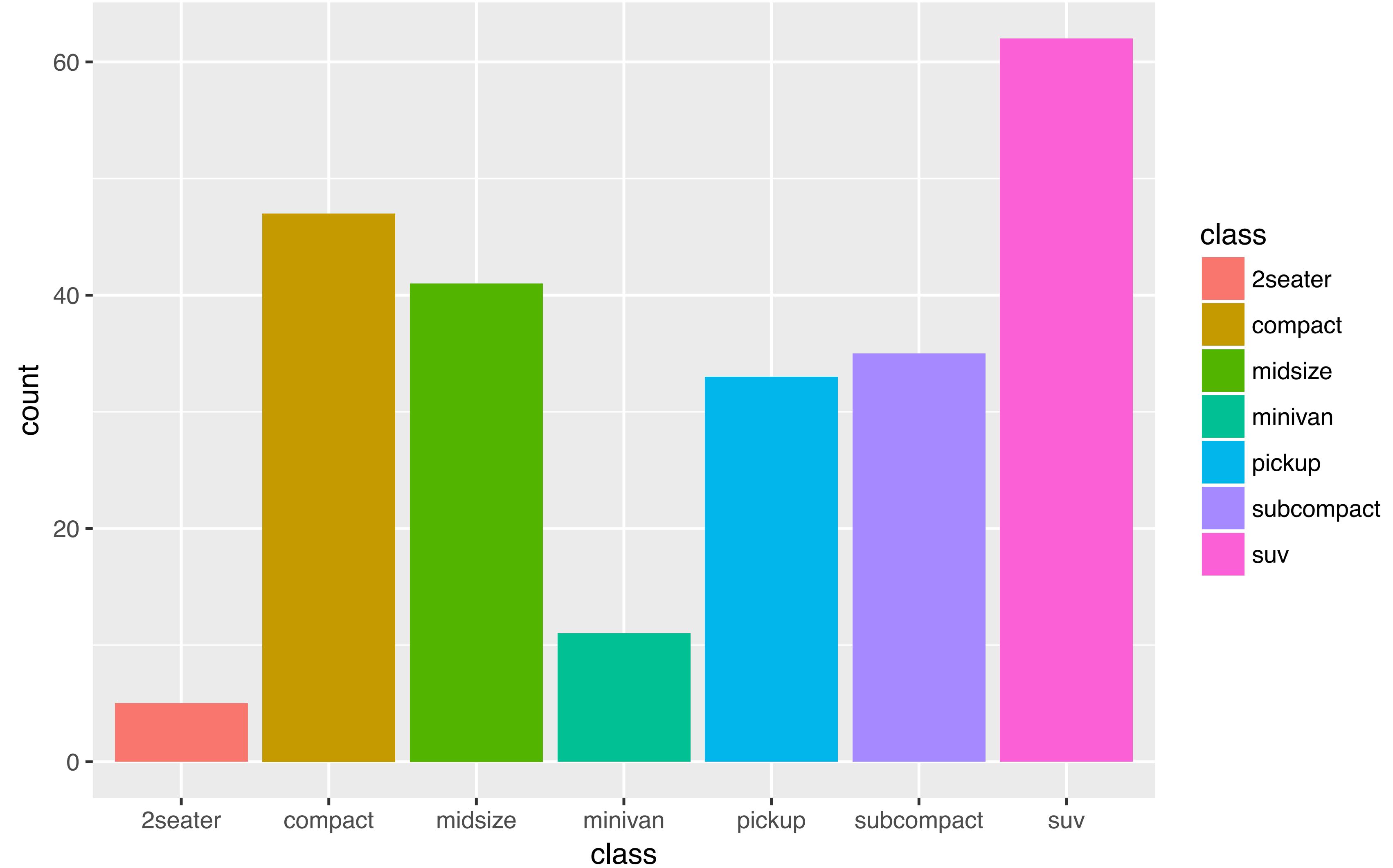
Your Turn 6

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





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

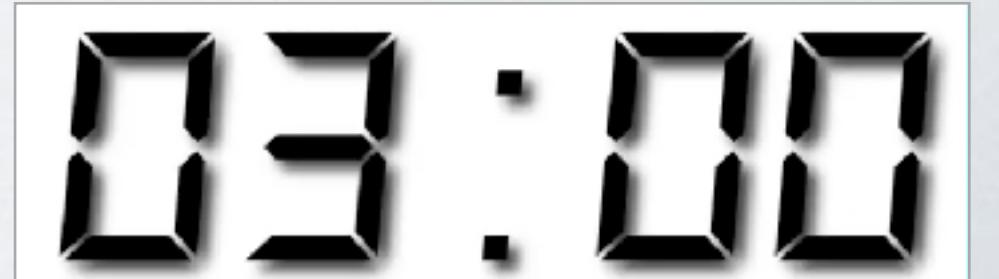


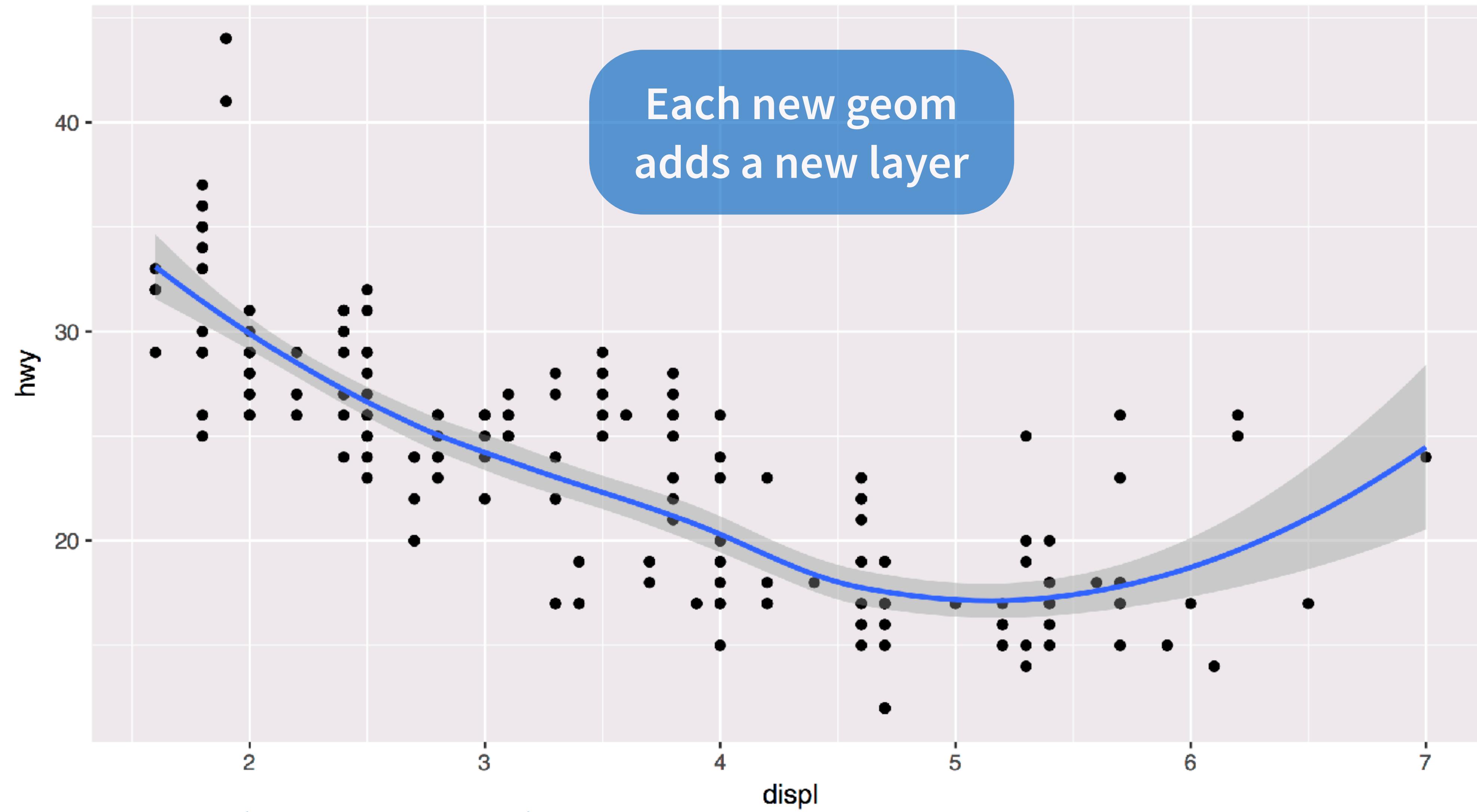
```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = class, fill = class))
```

Your Turn 7

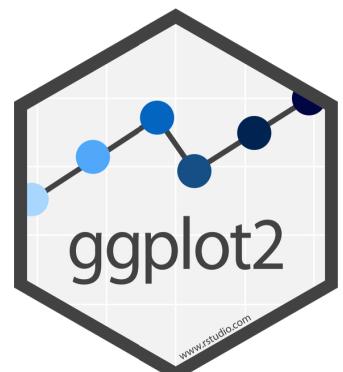
With your partner, predict what this code will do.
Then run it.

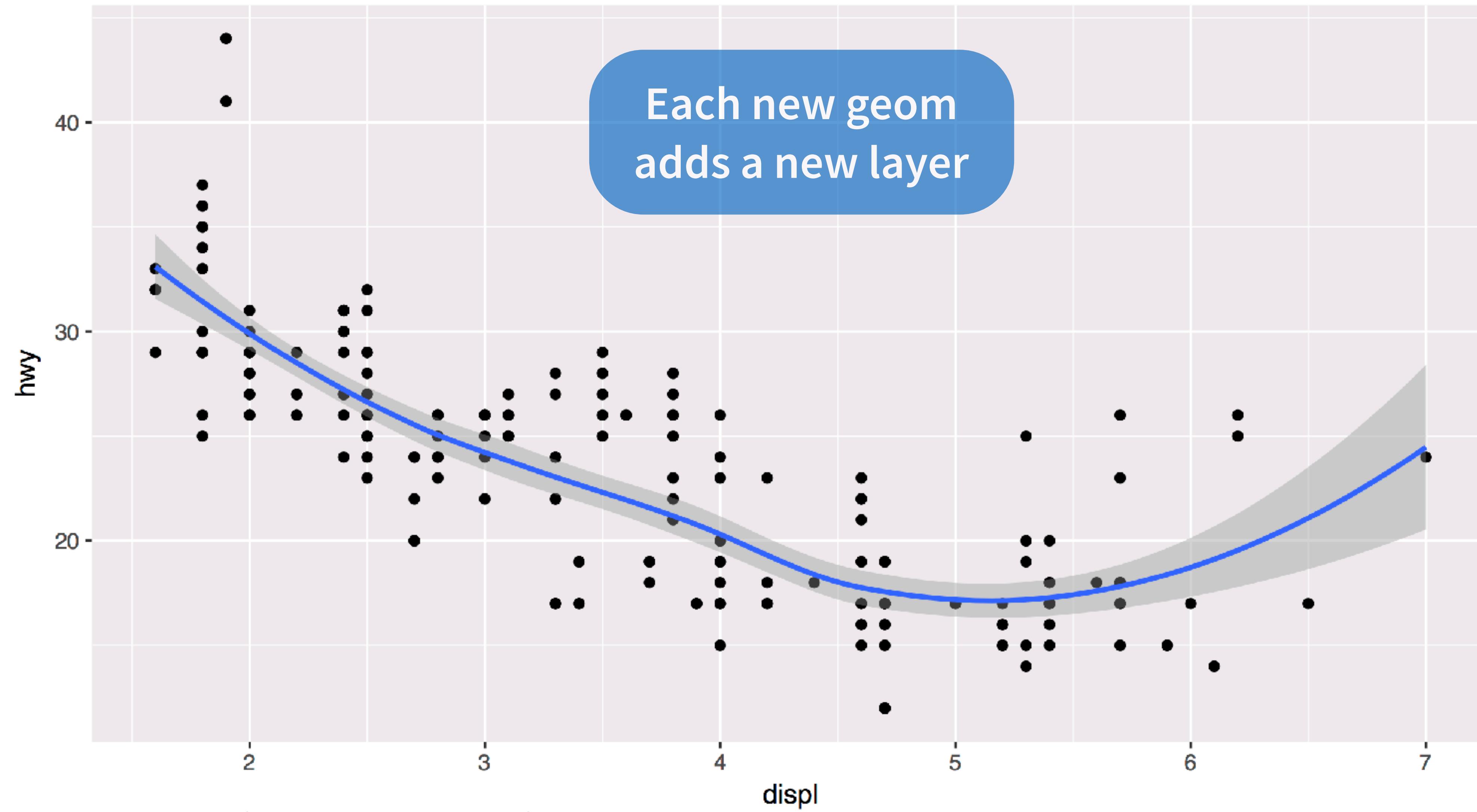
```
ggplot(mpg) +  
  geom_point(aes(displ, hwy)) +  
  geom_smooth(aes(displ, hwy))
```



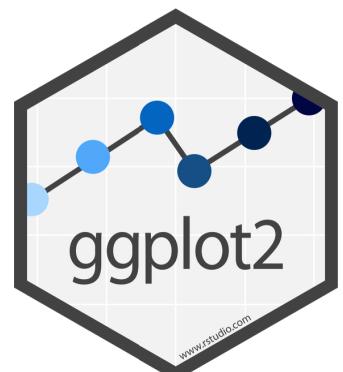


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

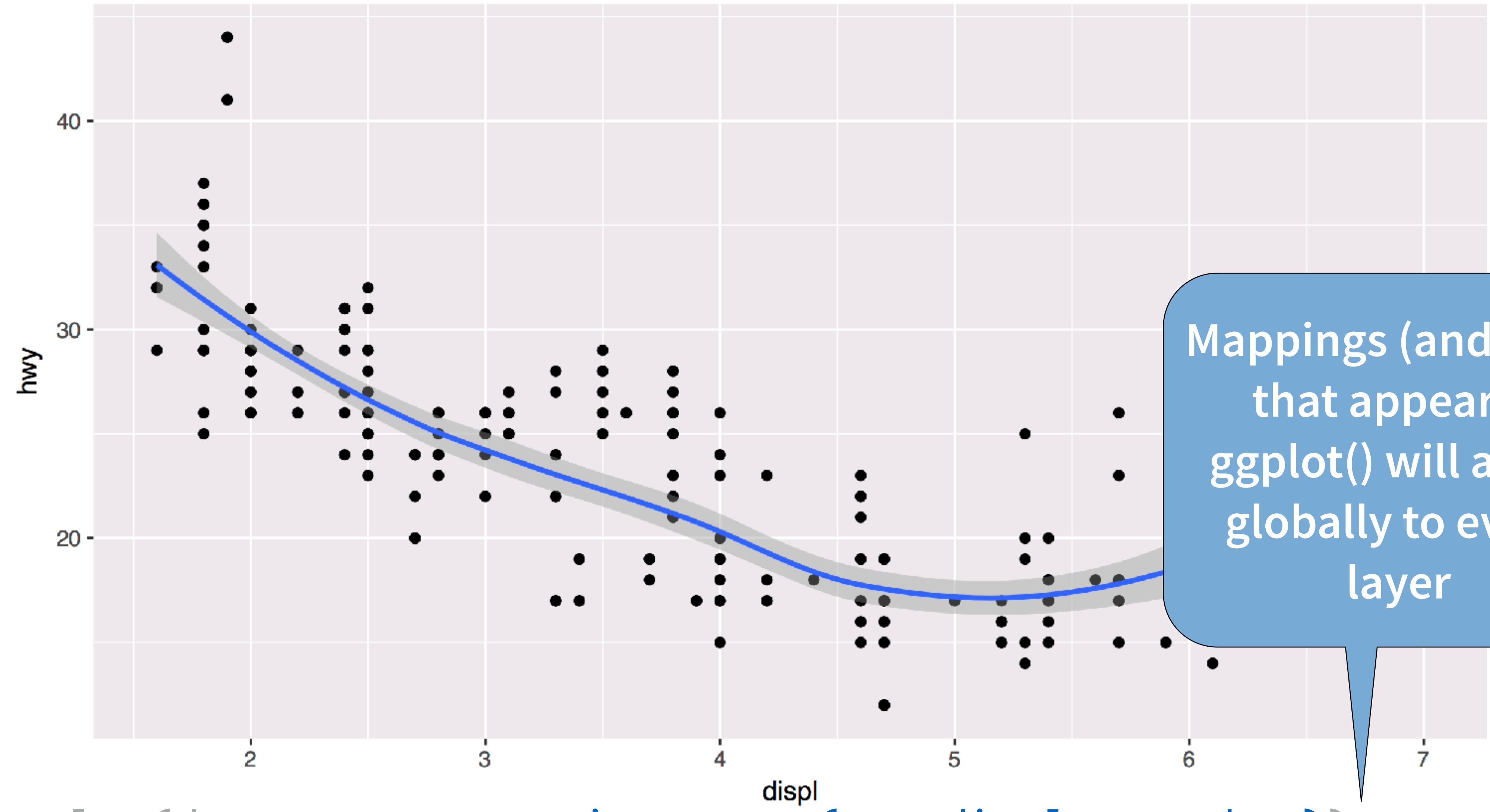




```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

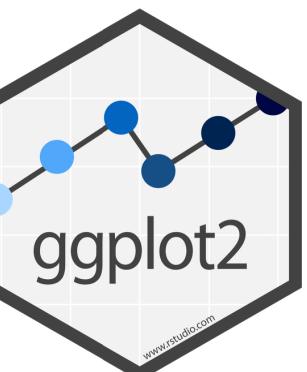


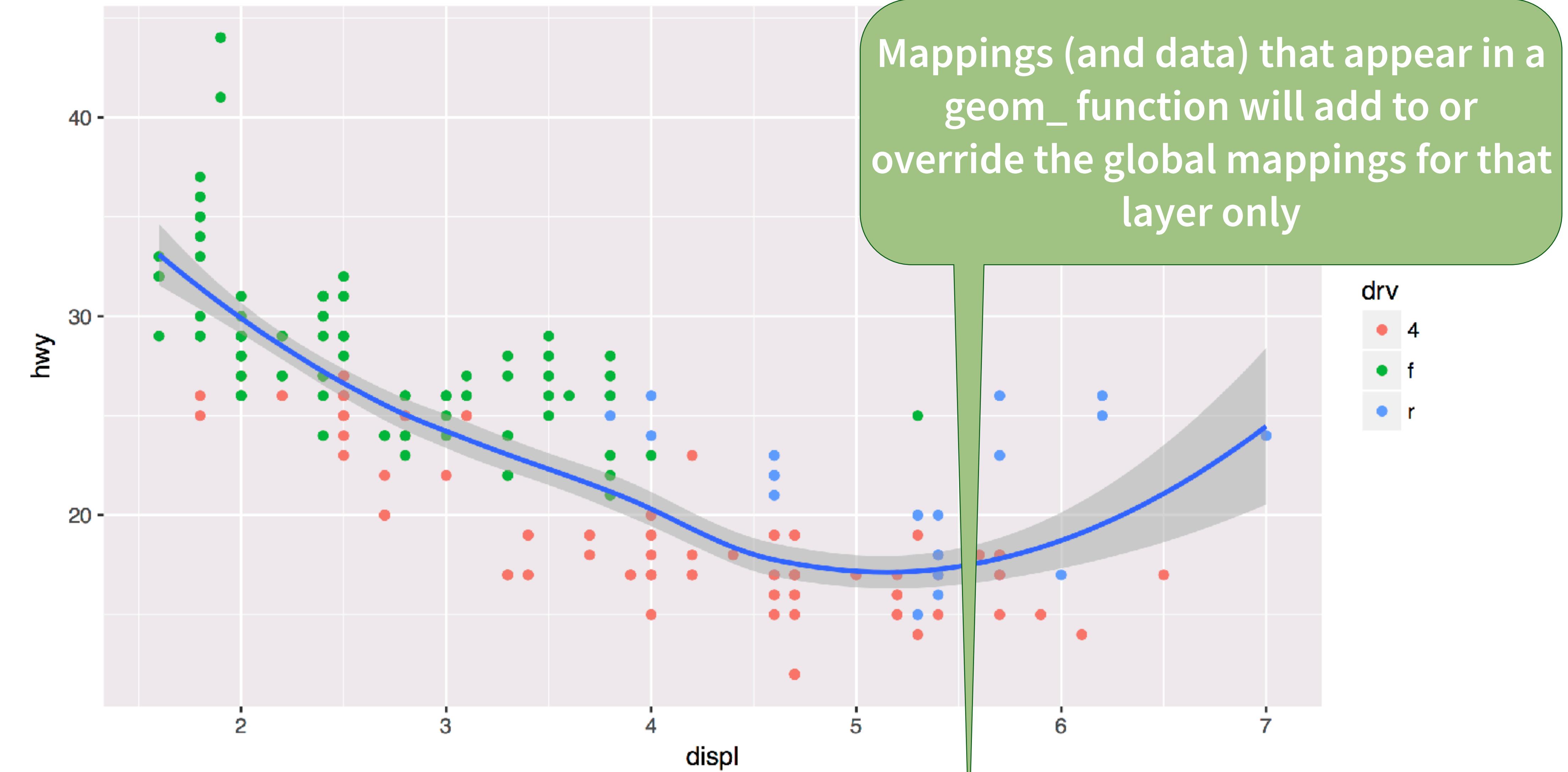
Global vs. Local



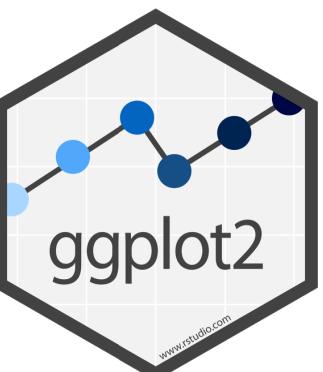
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
```

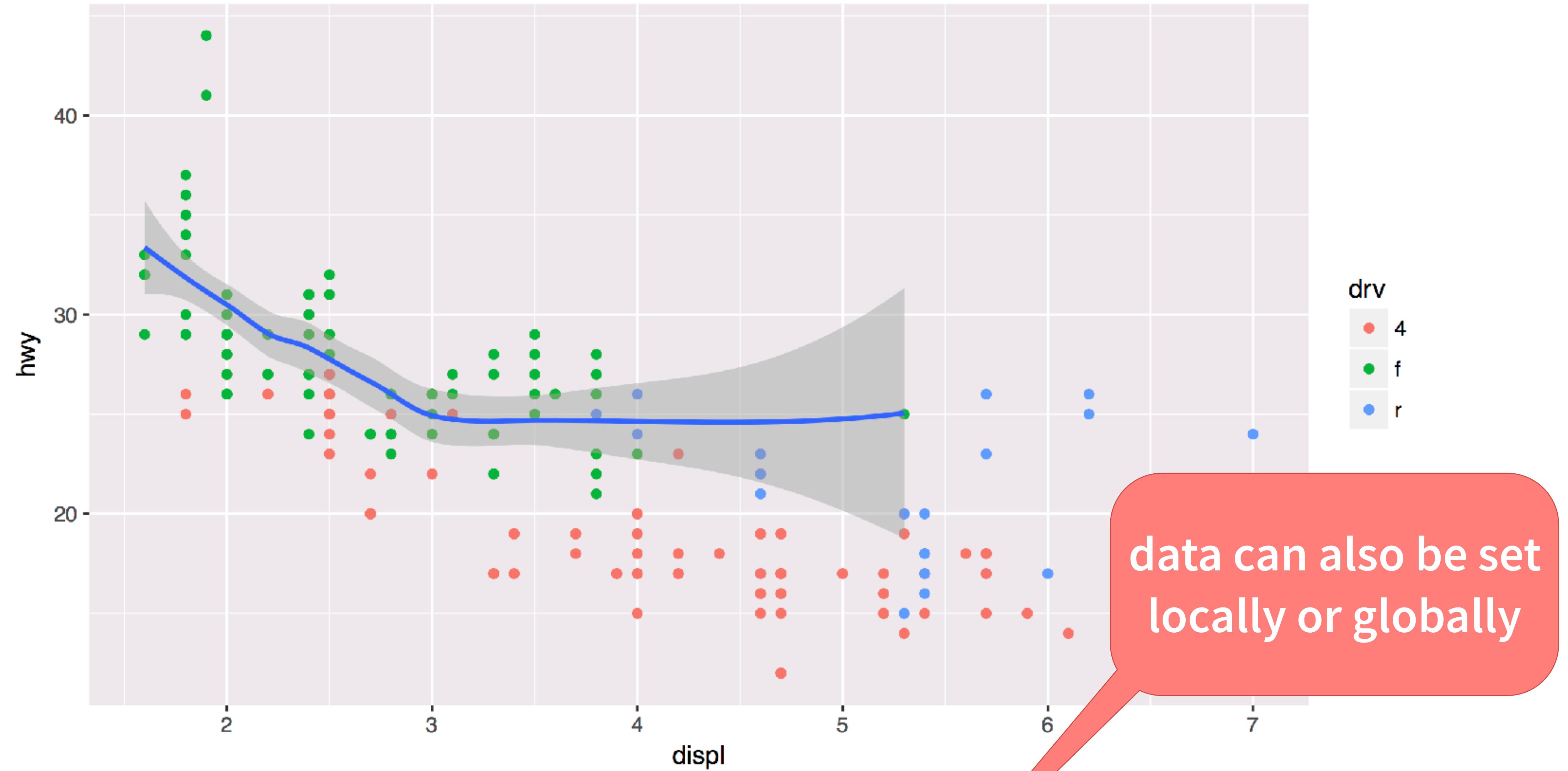
```
geom_point() +  
geom_smooth()
```





```
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()`

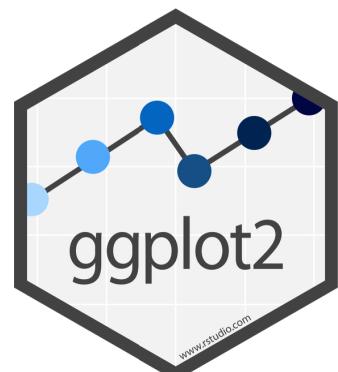
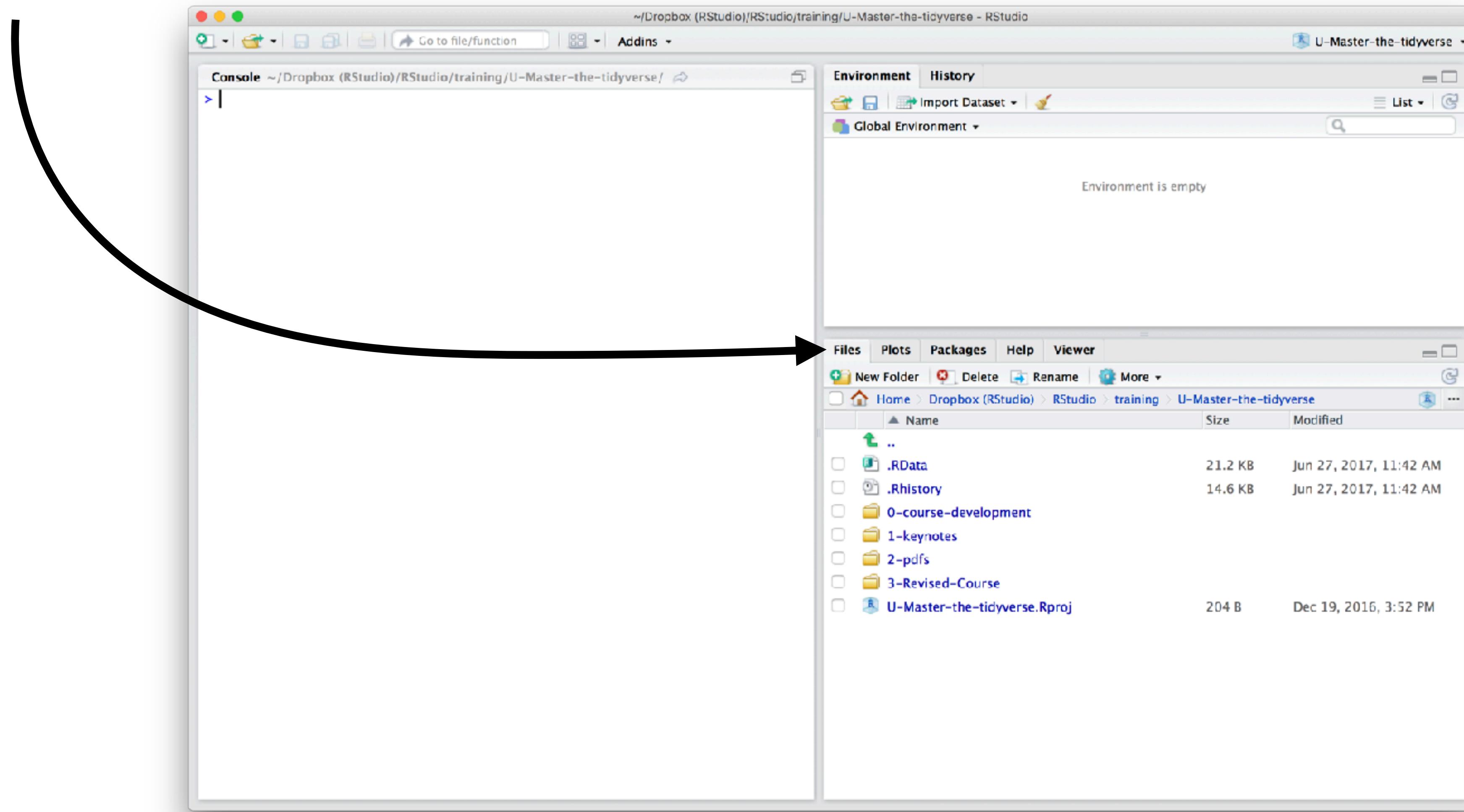


Working Directory

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

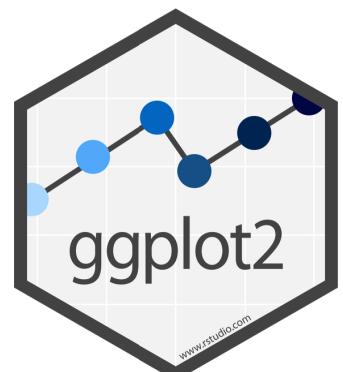
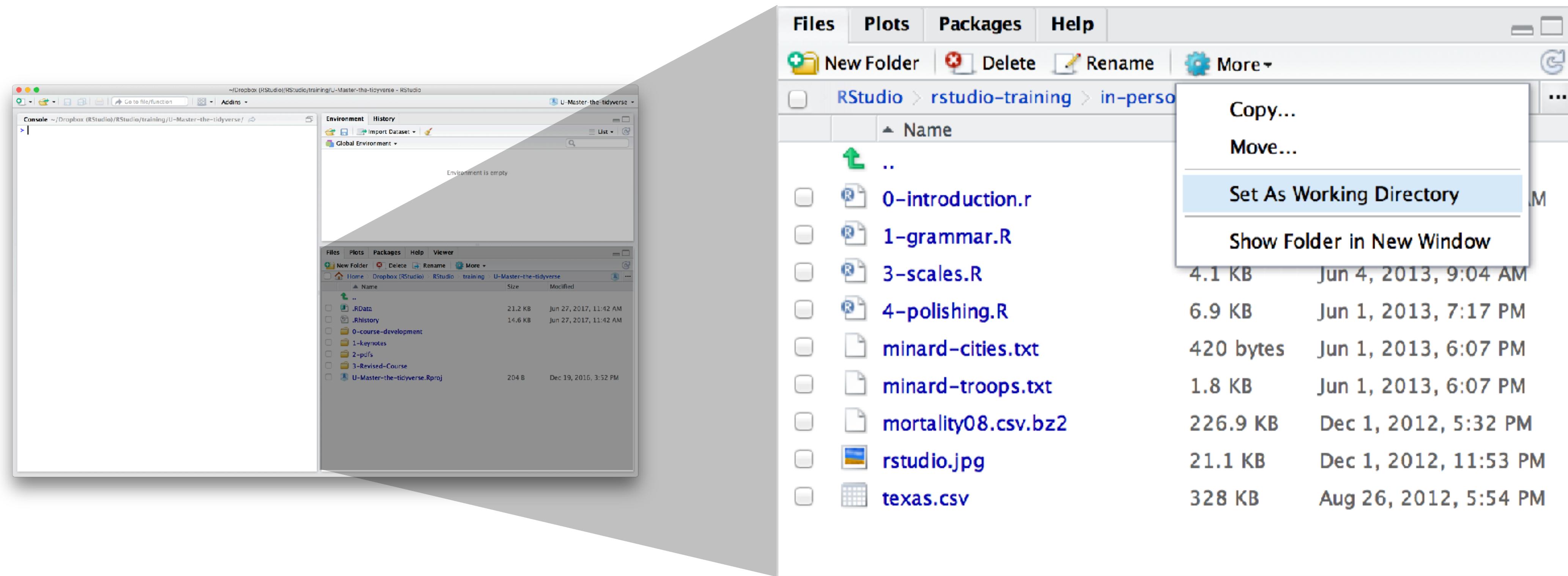
- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here

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

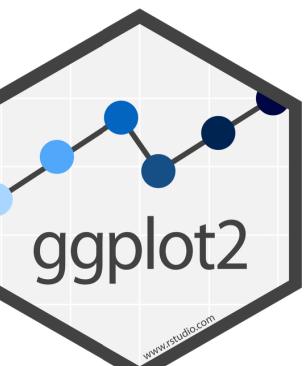
ggsave() saves the last plot.

Uses size on screen:

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

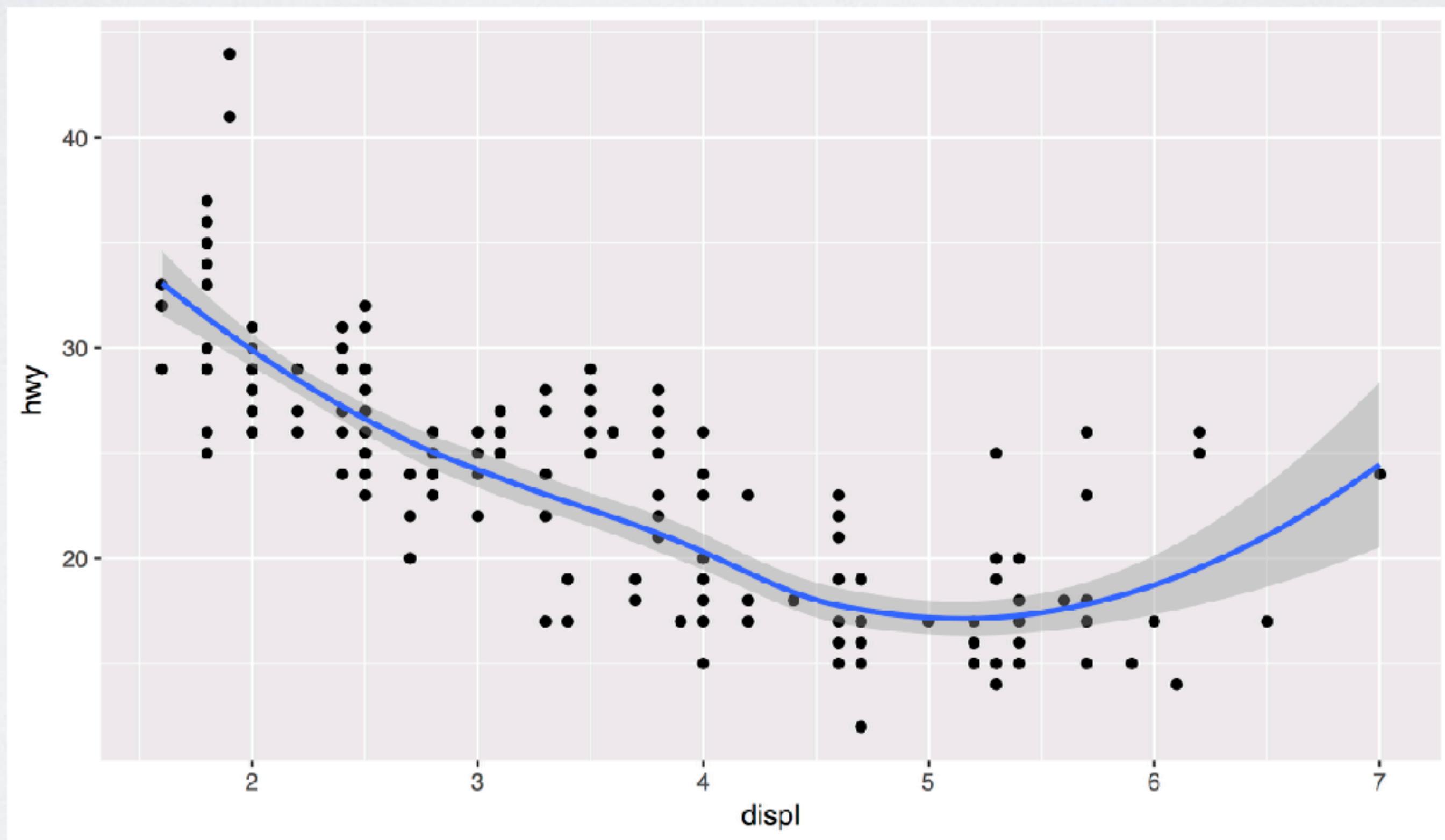
Specify size in inches

```
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>))
```

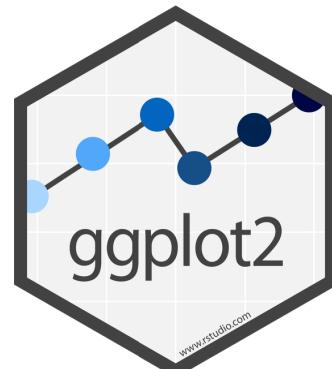
To make a graph

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

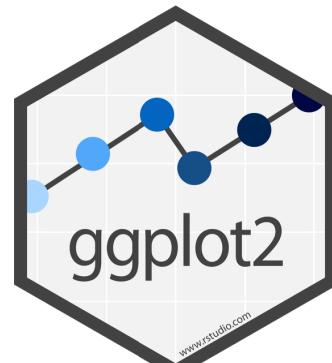
data

geom

1. Pick a **data** set

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

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



mappings

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

data

geom

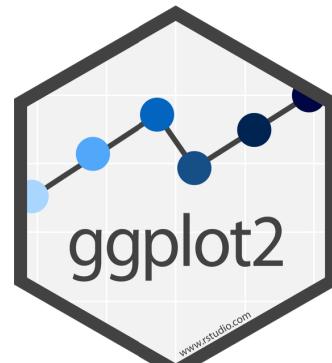
To make a graph

1. Pick a **data** set

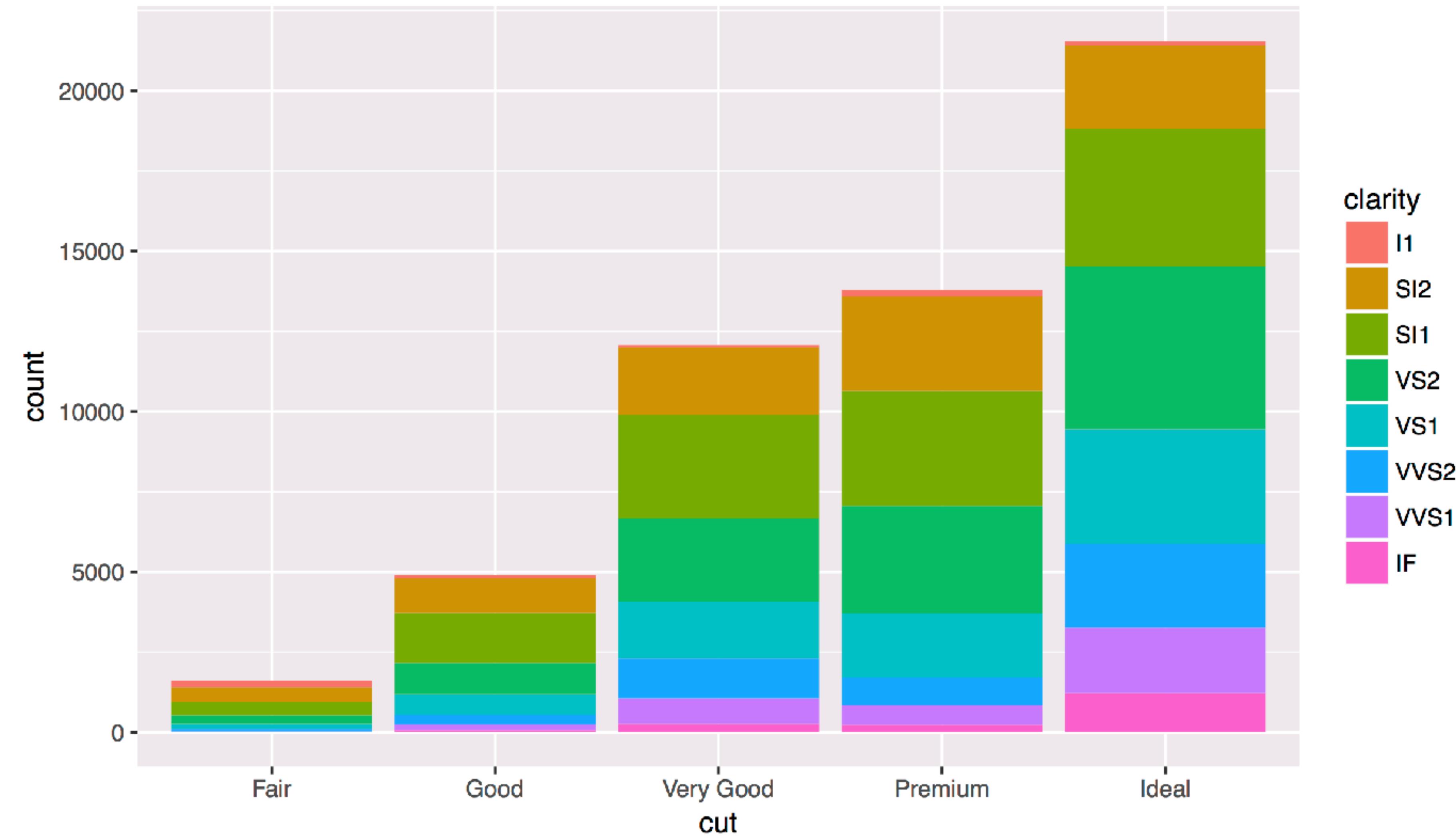
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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

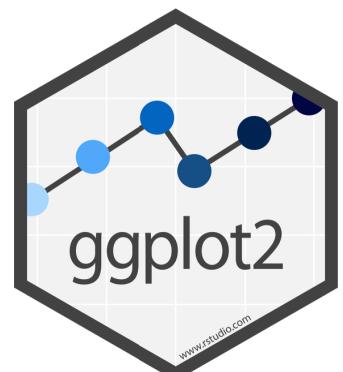
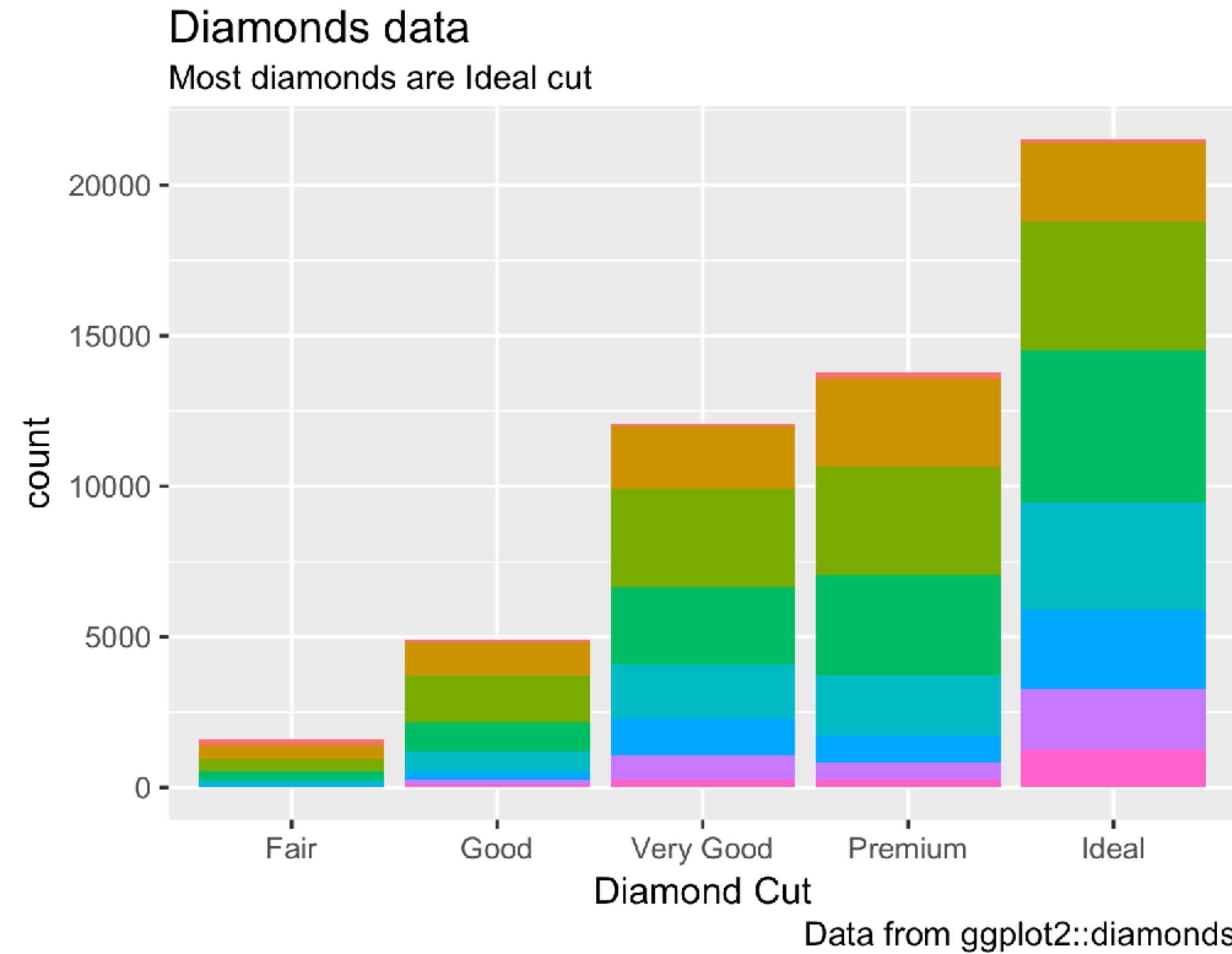
3. **Map** aesthetic
properties to
variables



What else?

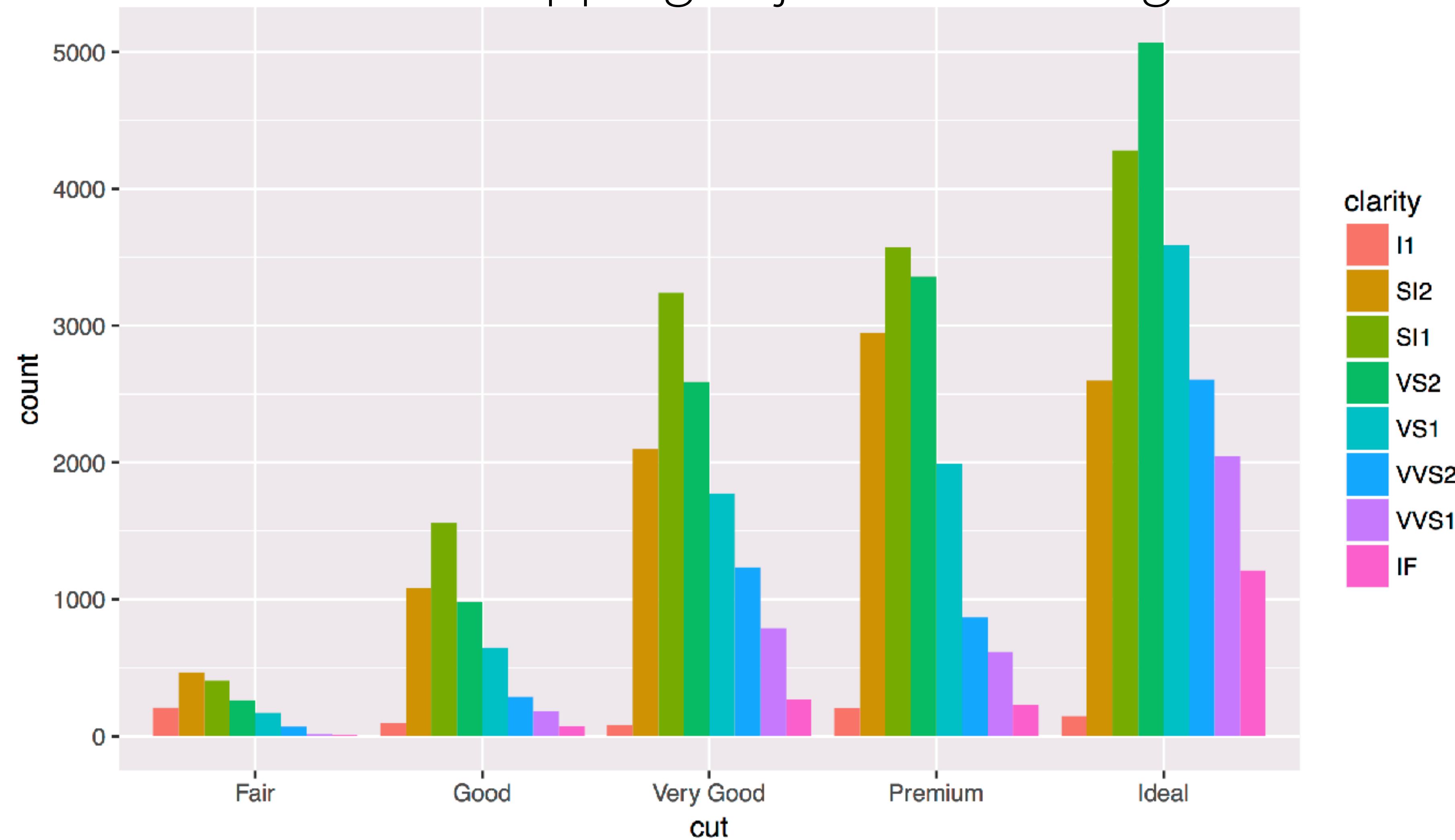


Titles and captions



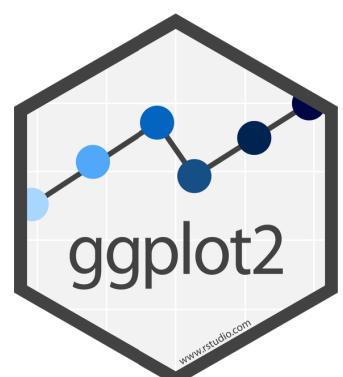
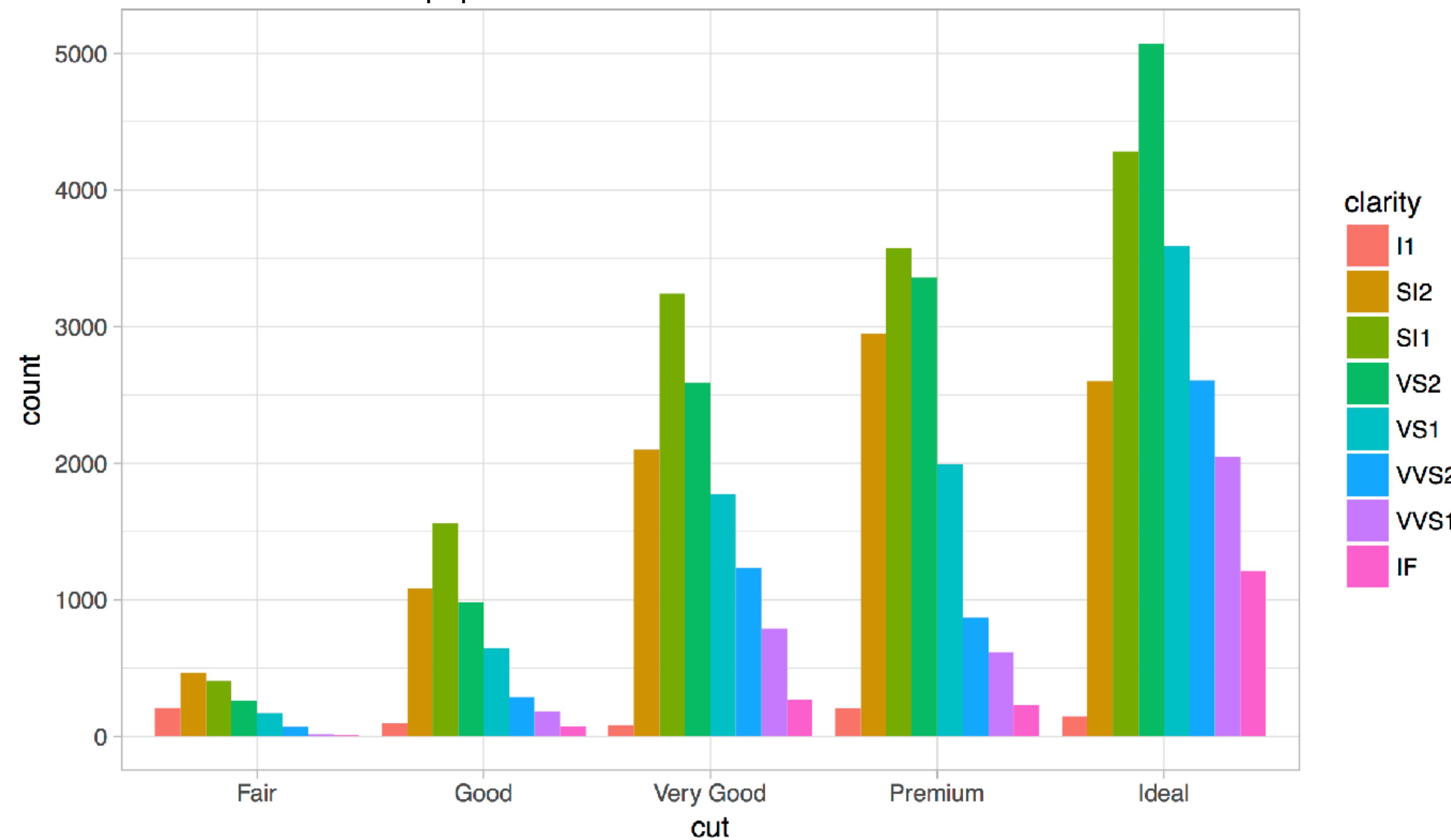
Position Adjustments

How overlapping objects are arranged



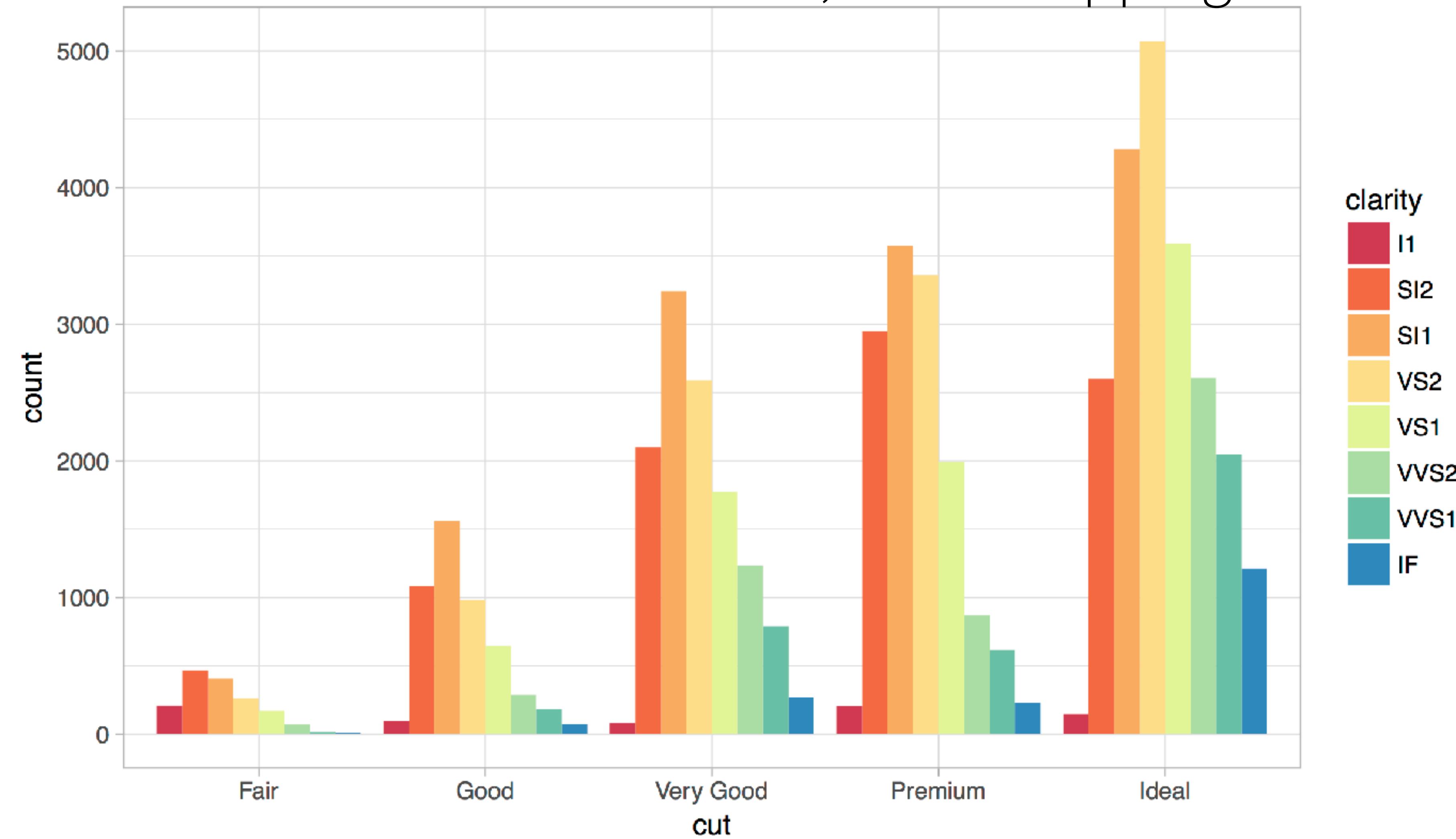
Themes

Visual appearance of non-data elements



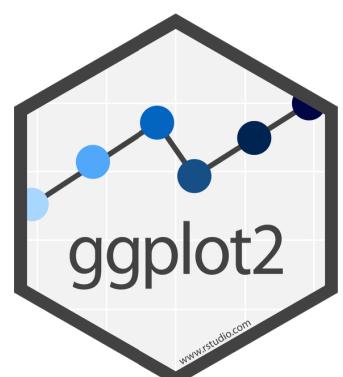
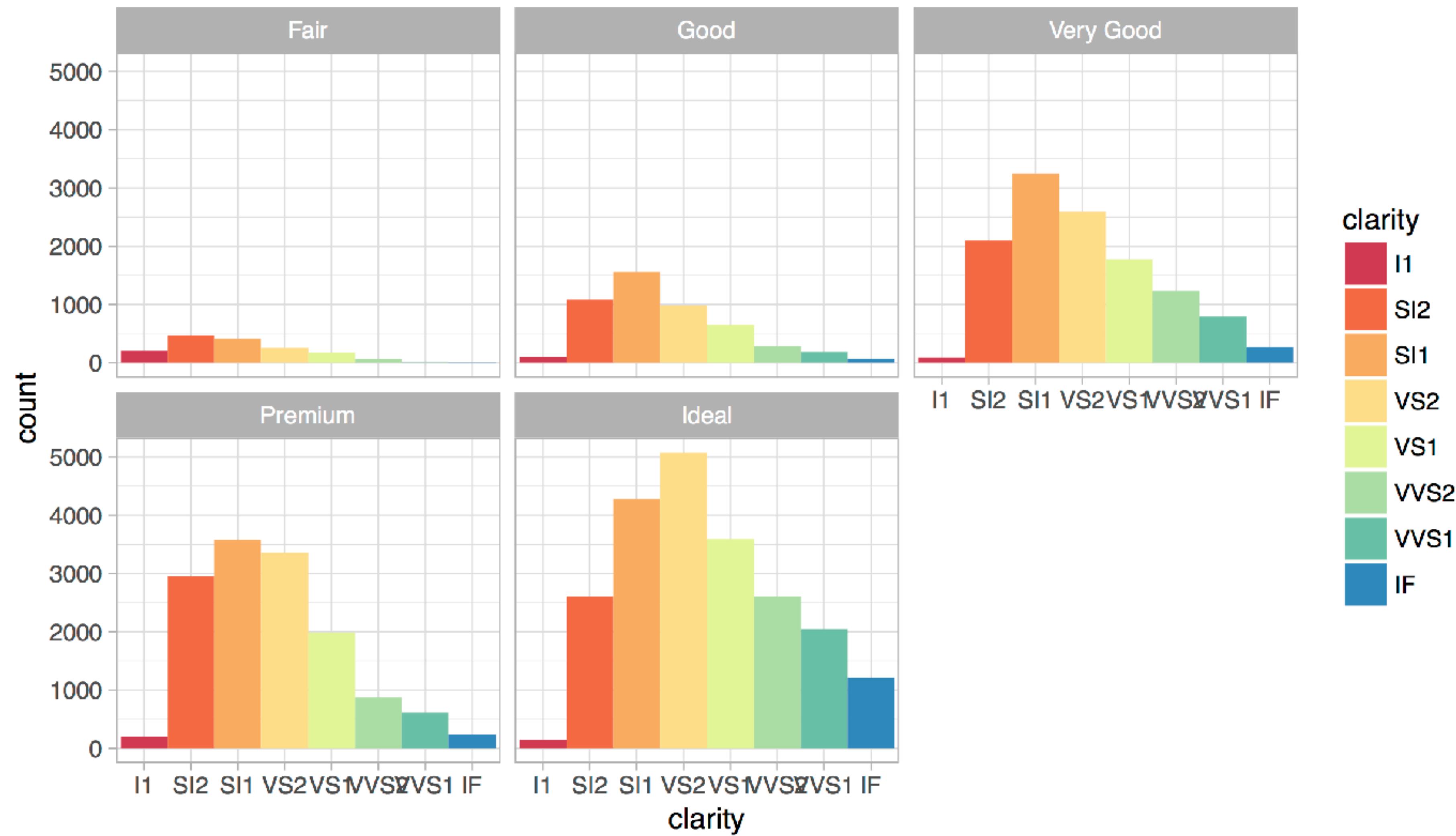
Scales

Customize color scales, other mappings



Facets

Subplots that display subsets of the data.



Coordinate systems



A ggplot2 template

Make any plot by filling in the parameters of this template

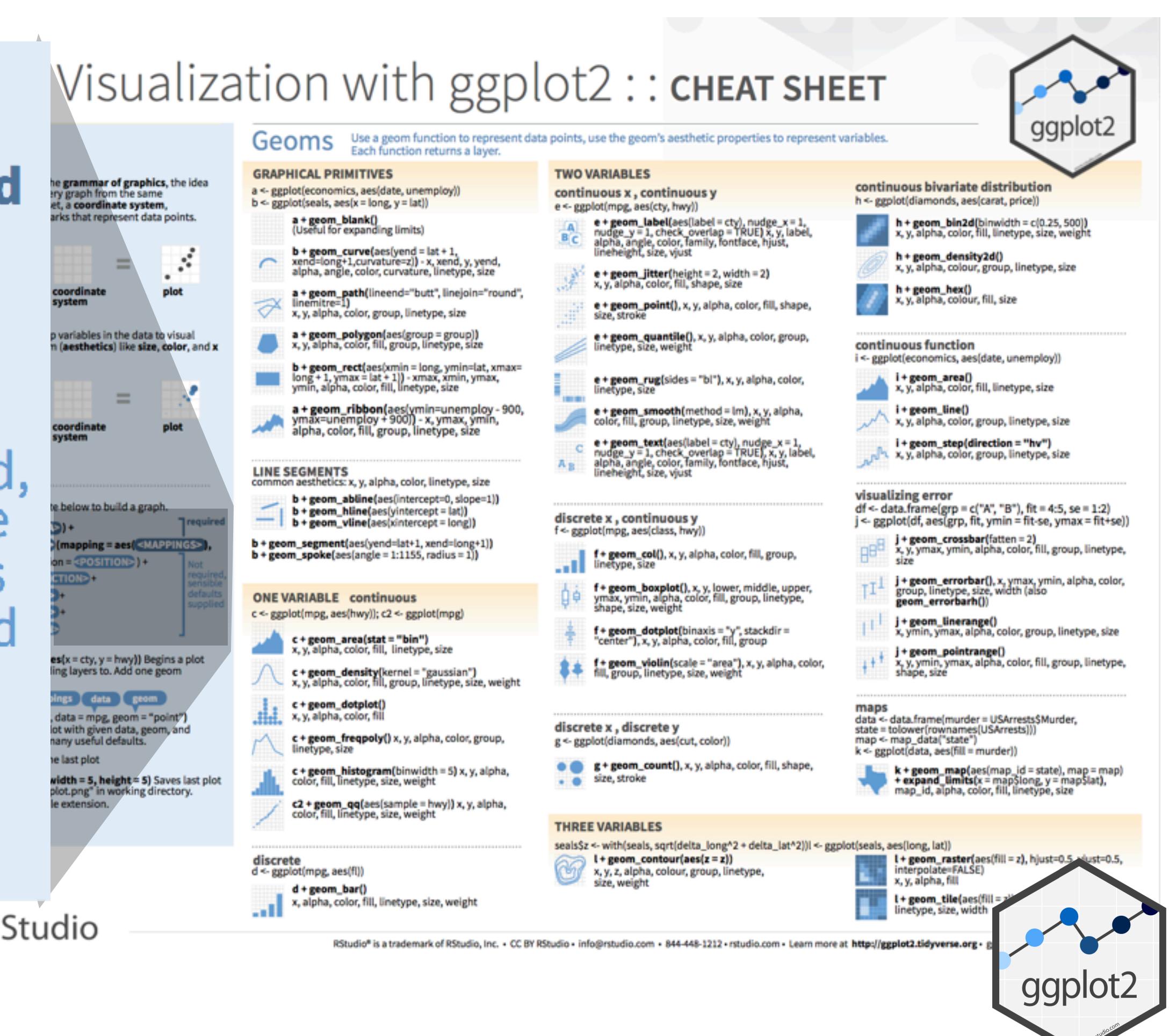
Complete the template below to build a graph.

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

required

Not required,
sensible
defaults
supplied

Visualization with ggplot2 :: CHEAT SHEET



The cheat sheet is organized into several sections:

- 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 list of basic geom functions: geom_blank, geom_label, geom_curve, geom_hex, geom_jitter, geom_point, geom_quantile, geom_rect, geom_ribbon, geom_rug, geom_smooth, geom_step, and geom_text.
- TWO VARIABLES**: A list of geom functions for two variables: geom_abline, geom_crossbar, geom_errorbar, geom_hex, geom_hex2d, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- continuous**: A list of geom functions for continuous data: geom_area, geom_bar, geom_boxplot, geom_dotplot, geom_freqpoly, geom_histogram, geom_qq, and geom_violin.
- discrete**: A list of geom functions for discrete data: geom_bar, geom_col, geom_crossbar, geom_errorbar, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- maps**: A list of geom functions for maps: geom_map, geom_count, geom_raster, geom_contour, and geom_tile.
- continuous function**: A list of geom functions for continuous functions: geom_area, geom_line, geom_smooth, and geom_stepdirection.
- continuous bivariate distribution**: A list of geom functions for bivariate distributions: geom_hex, geom_hex2d, geom_hexbin, geom_hexgrid, and geom_hexjitter.
- visualizing error**: A list of geom functions for visualizing errors: geom_abline, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- continuous x , continuous y**: A list of geom functions for continuous x and y: geom_abline, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- continuous discrete y**: A list of geom functions for continuous x and discrete y: geom_bar, geom_col, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- discrete x , continuous y**: A list of geom functions for discrete x and continuous y: geom_bar, geom_col, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- discrete x , discrete y**: A list of geom functions for discrete x and discrete y: geom_bar, geom_col, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.
- THREE VARIABLES**: A list of geom functions for three variables: geom_bar, geom_hex, geom_hexbin, geom_hexgrid, geom_hexjitter, geom_hexquantile, geom_hexrect, geom_hexribbon, geom_hexrug, geom_hexsmooth, geom_hexstep, and geom_hextext.

R Studio logo

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1232 • rstudio.com • Learn more at <http://ggplot2.tidyverse.org>

ggplot2.tidyverse.org

The screenshot shows a web browser displaying the ggplot2.tidyverse.org website. The page title is "Create Elegant Data Visualisation" and the user is logged in as "Garrett". The address bar shows the URL "ggplot2.tidyverse.org". The main content area features the ggplot2 logo and the text "part of the tidyverse". A large heading "Usage" is followed by a paragraph explaining the philosophy of ggplot2 and how to use it. Below this is a code block showing R code to create a scatter plot. To the right, there is a "Links" sidebar with links to CRAN, GitHub source code, reporting bugs, learning more, and the license. At the bottom, there is a small ggplot2 logo graphic.

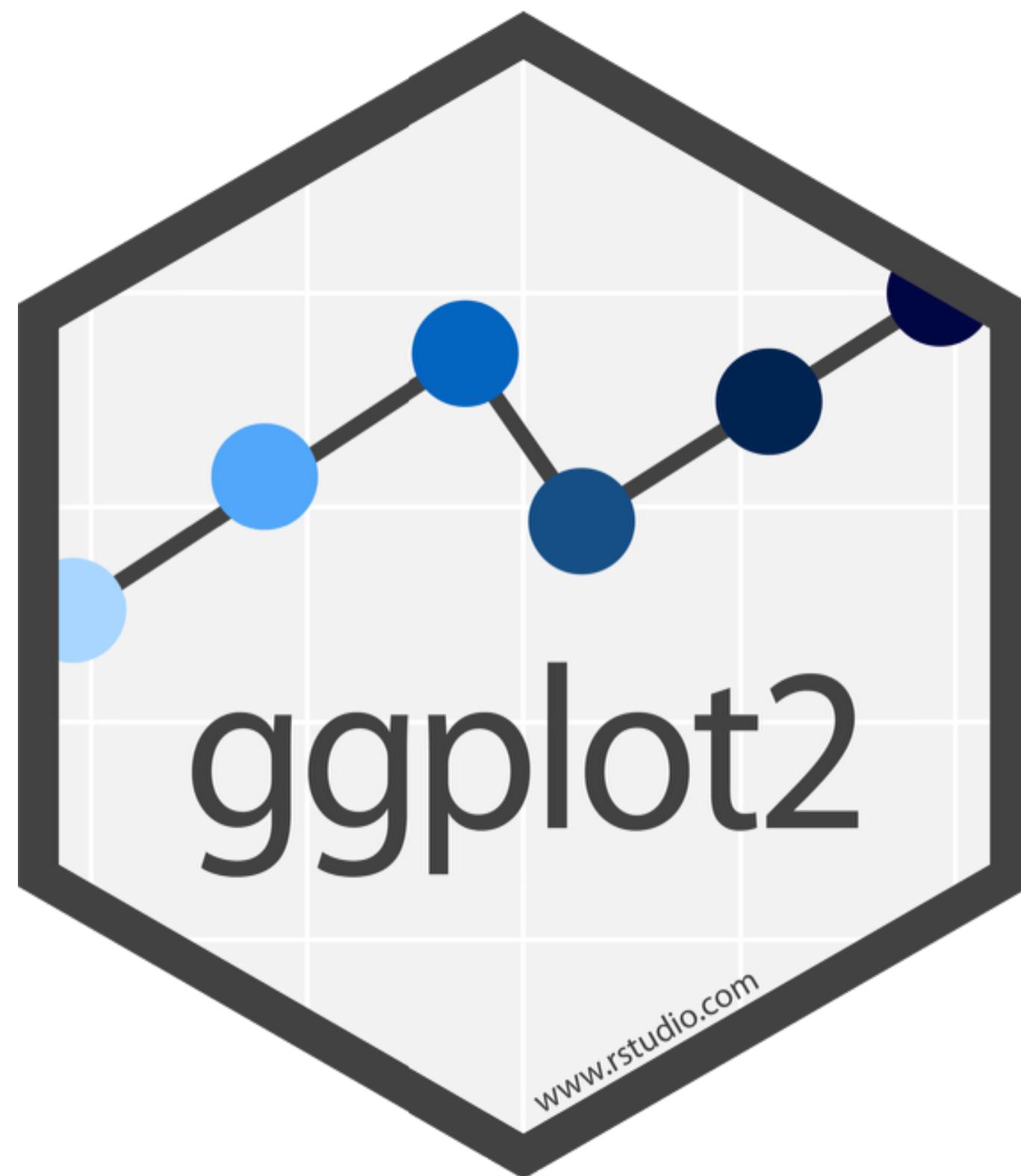
Usage

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()
```

Visualize Data with



www.rstudio.com