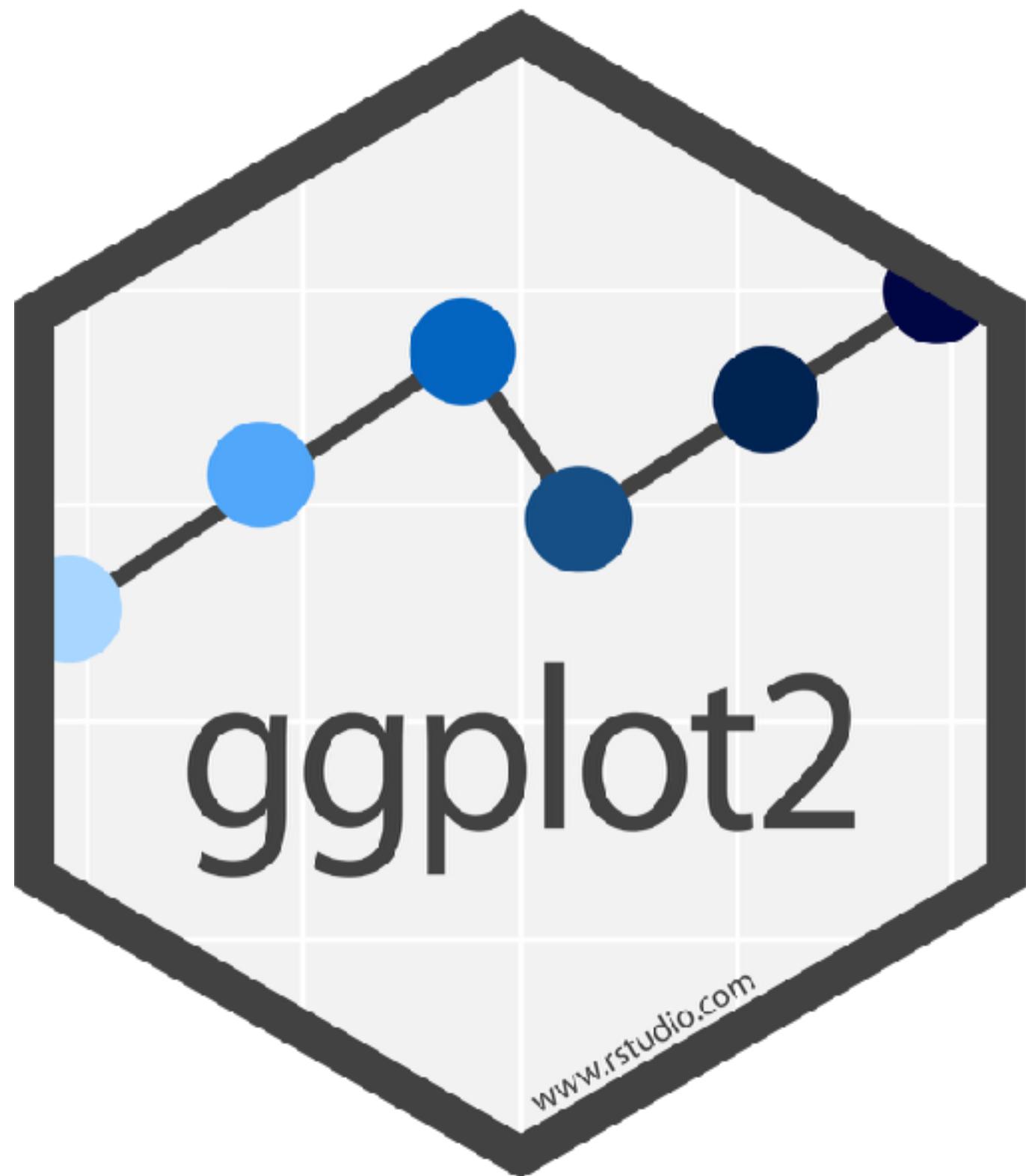
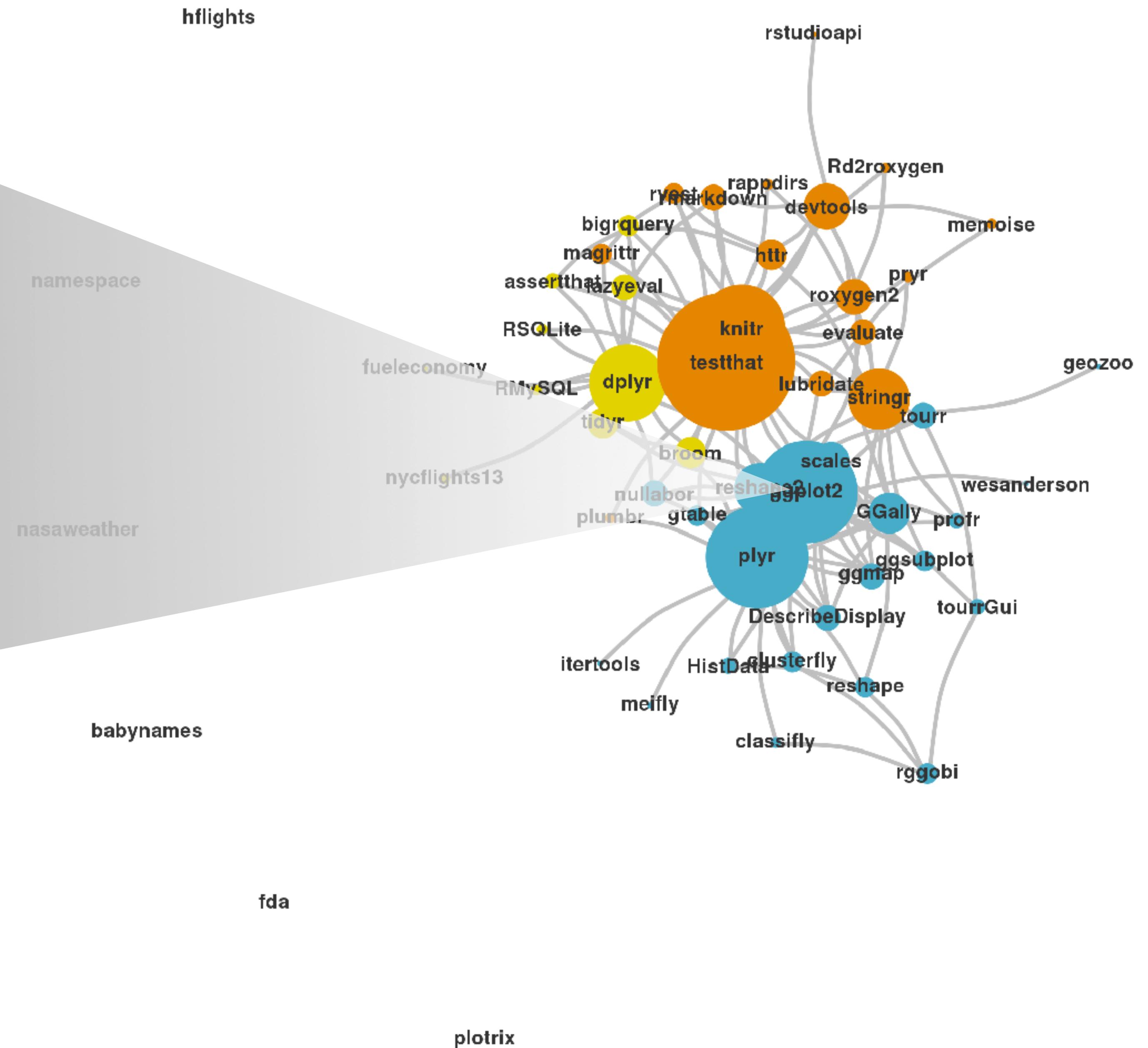
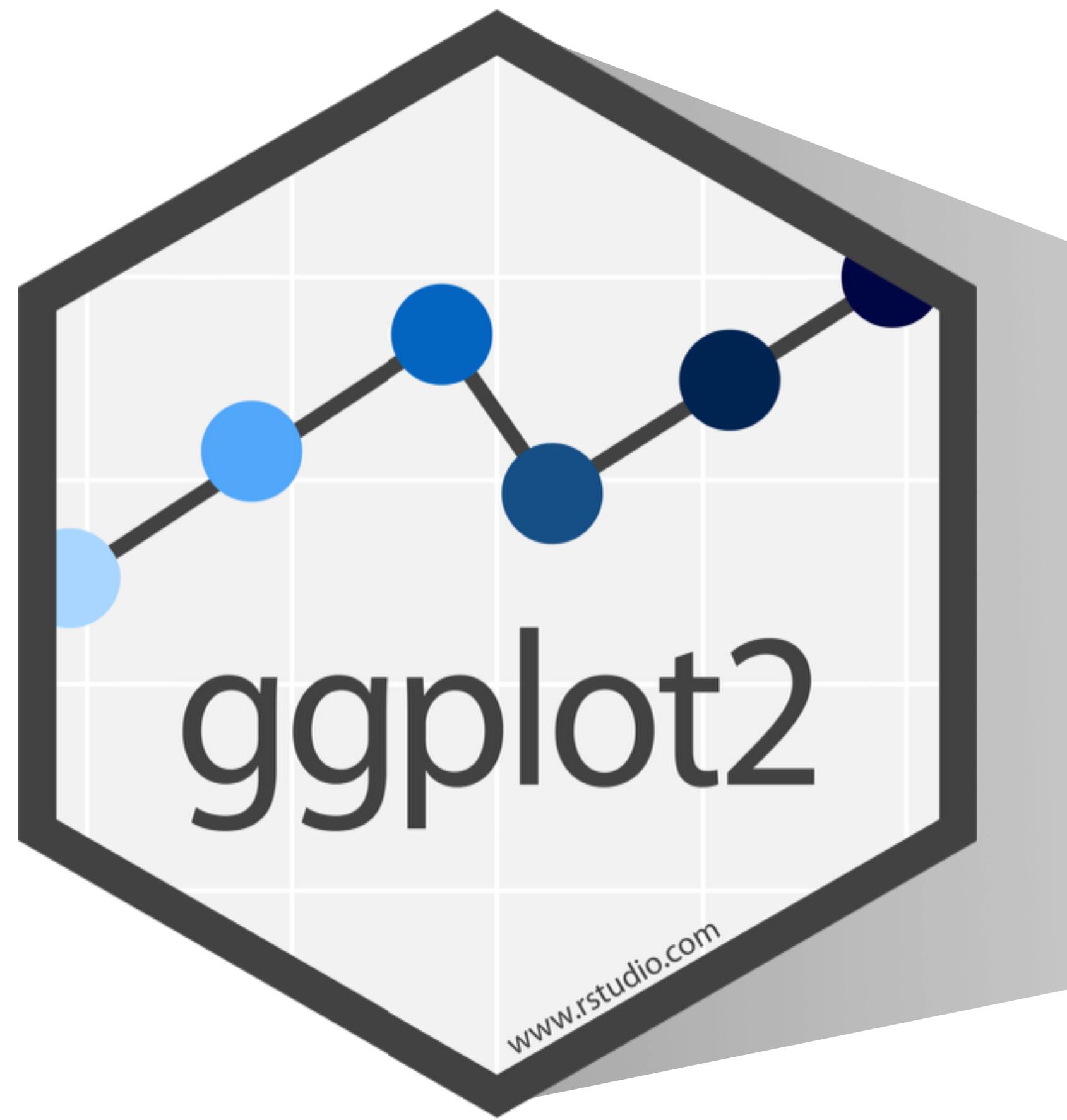


Visualize Data with



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

- John Tukey



Setup

The setup chunk is always run once before anything else



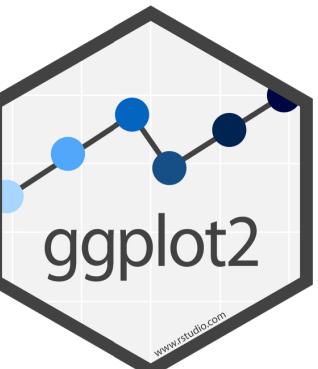
(optional) label for chunk

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

mpg

Fuel economy data for 38 models of car.

mpg



Quiz

Confer with your group.

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

No peeking ahead!



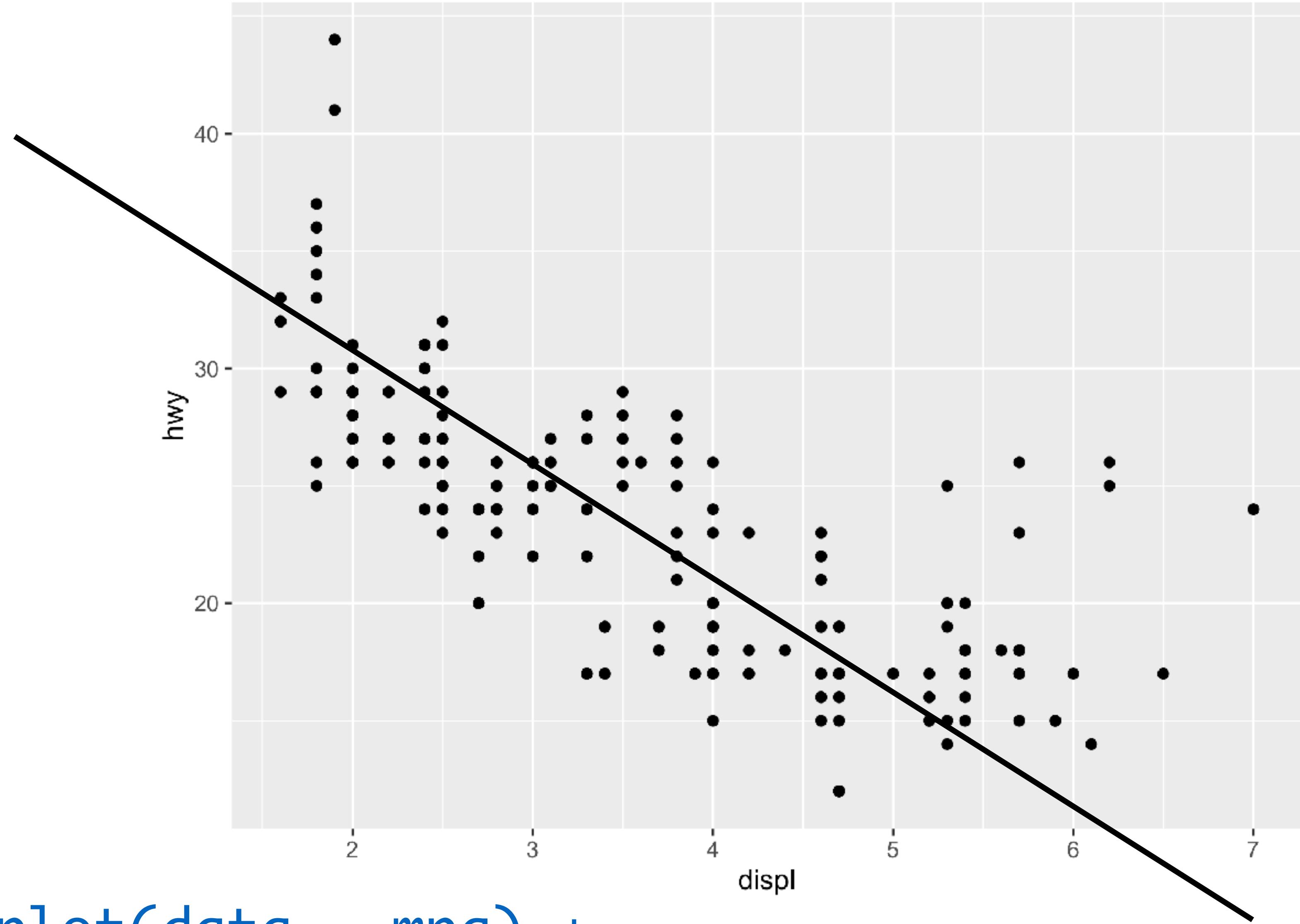
Your Turn 1

Run this code in your notebook to make a graph.

Pay strict attention to spelling, capitalization, and parentheses!

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

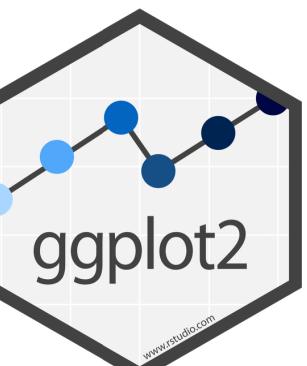




```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

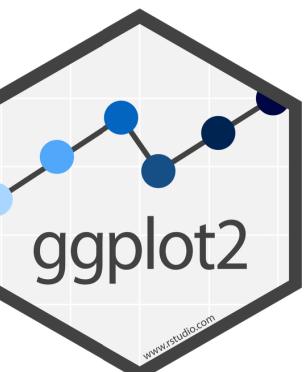
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

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



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

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



data

+ before new line

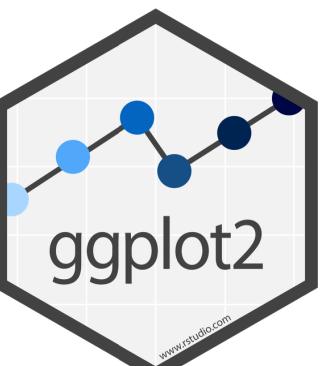
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

type of layer

aes()

x variable

y variable



A template

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

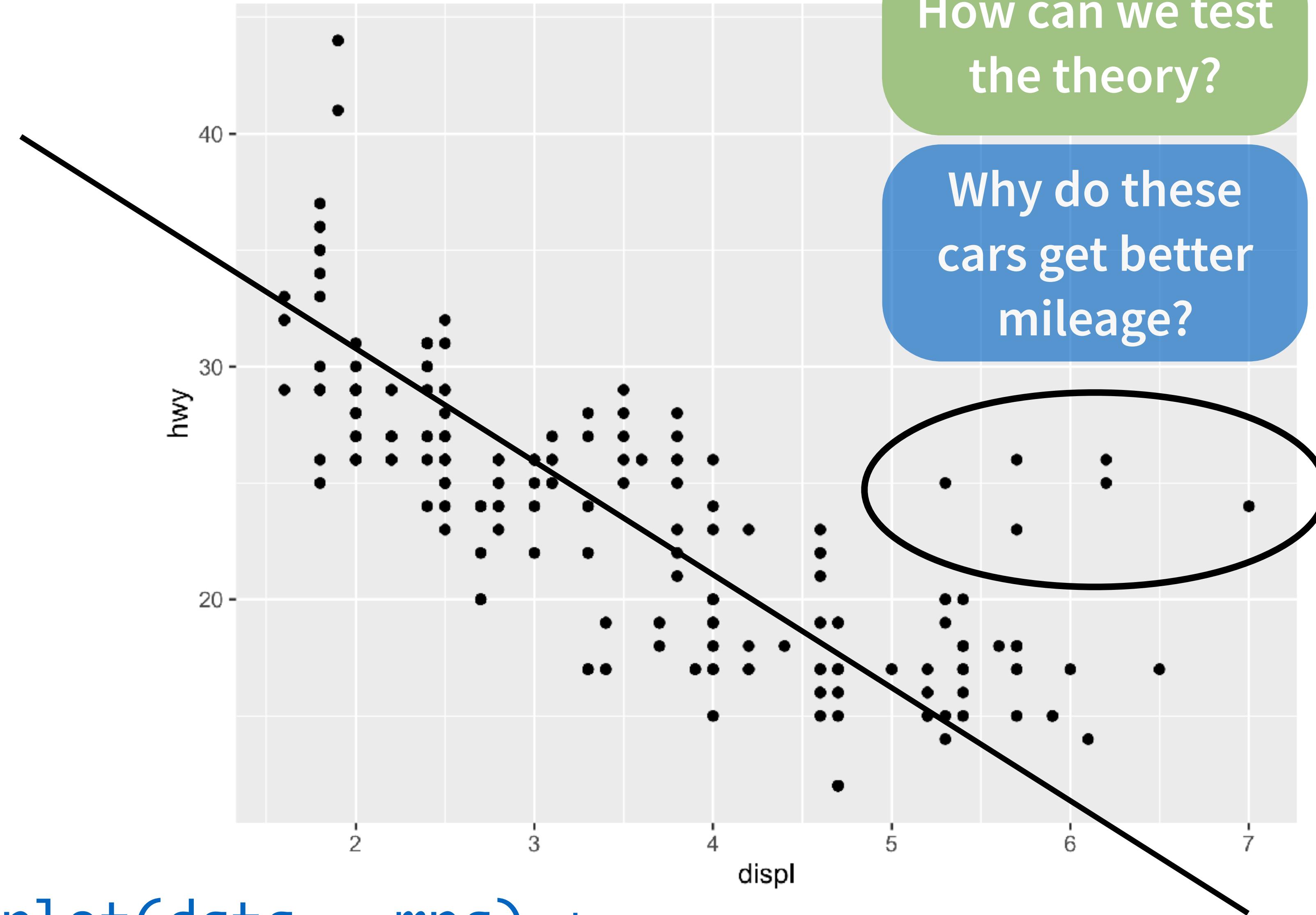
```
geom_point(mapping = aes(x = displ, y = hwy))
```

A template

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

Mappings



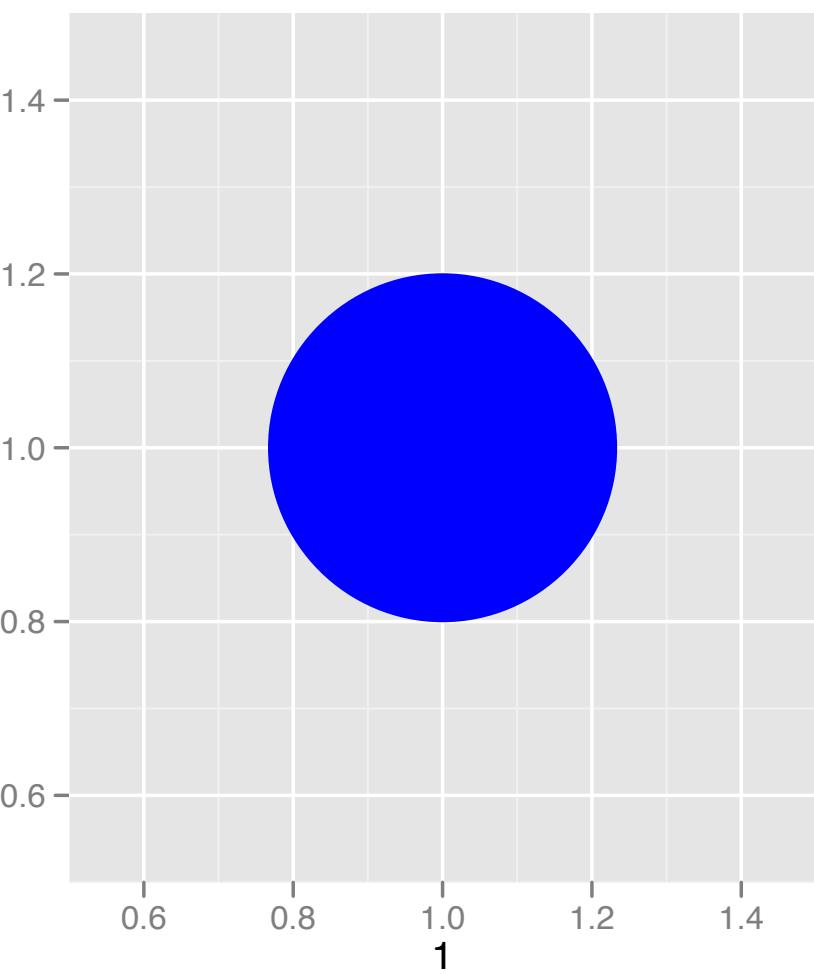
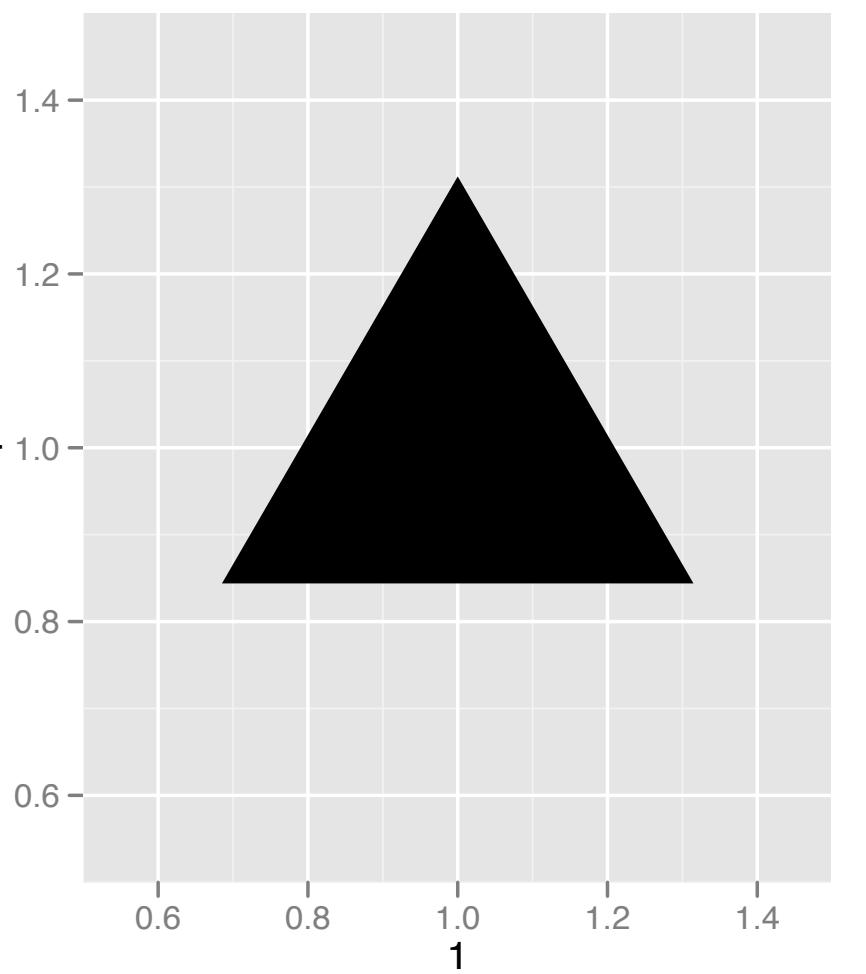
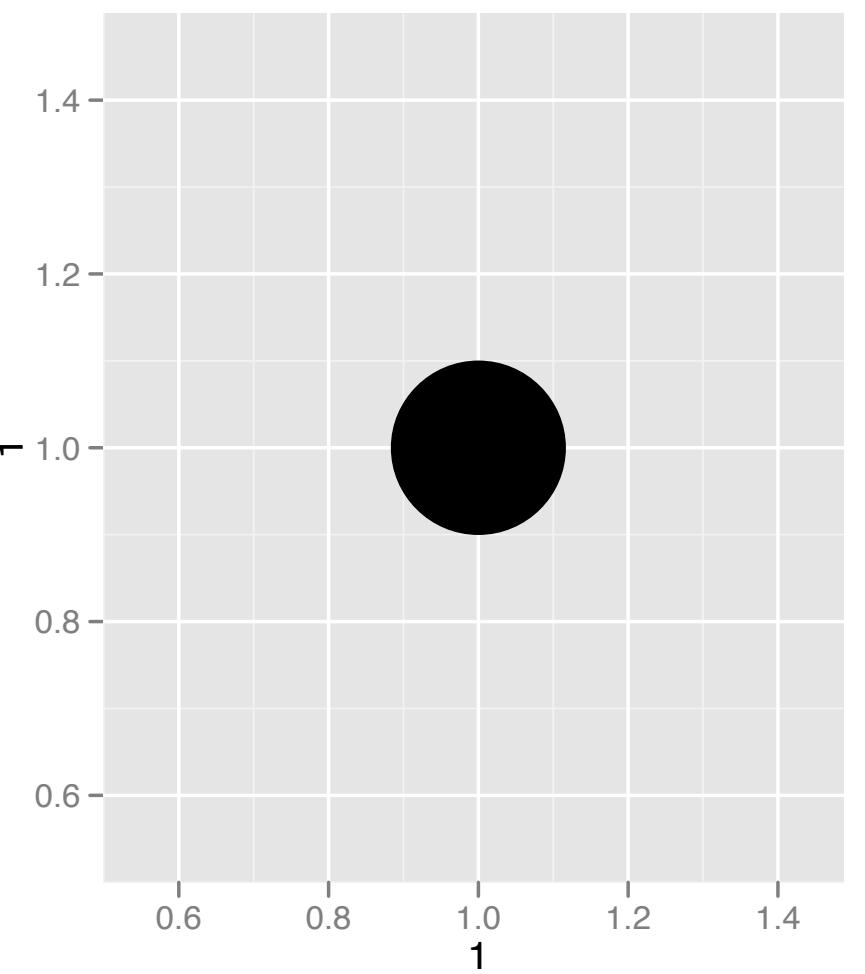
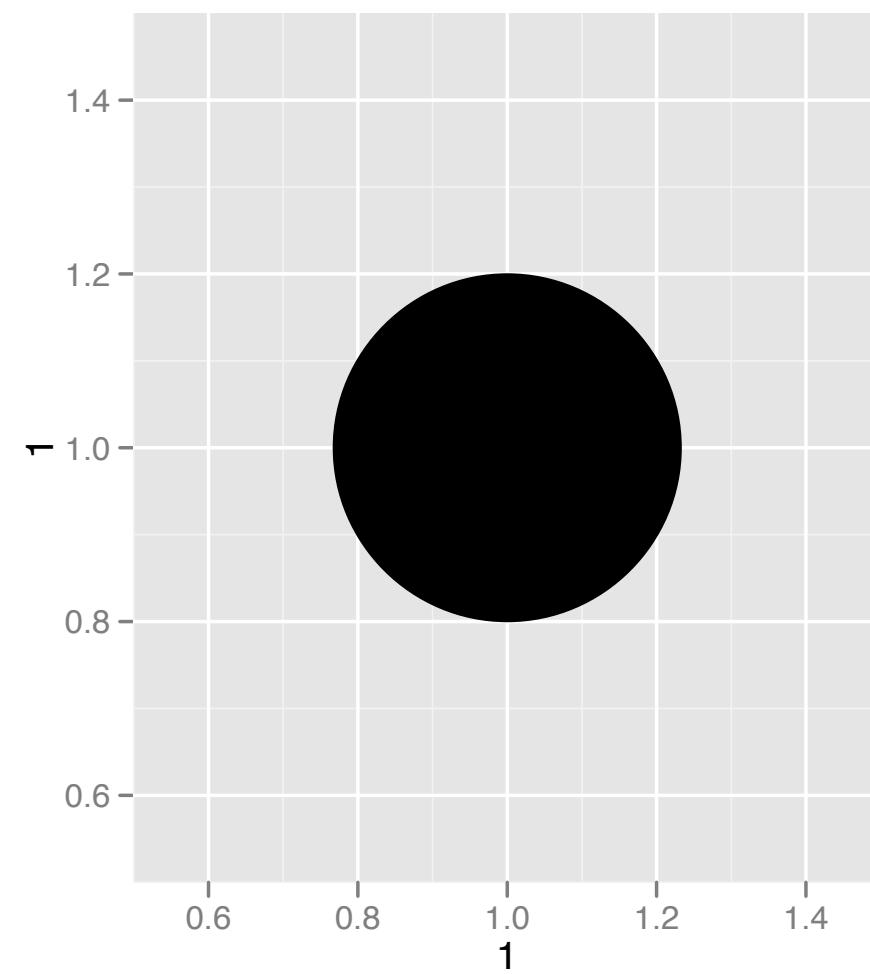


How can we test
the theory?

Why do these
cars get better
mileage?

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

Aesthetics



Visual Space

color

Red

Brown

Green

Aqua

Blue

Violet

Pink

Data Space

class

2seater

compact

midsize

minivan

pickup

subcompact

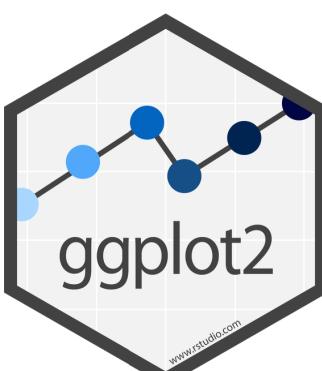
suv

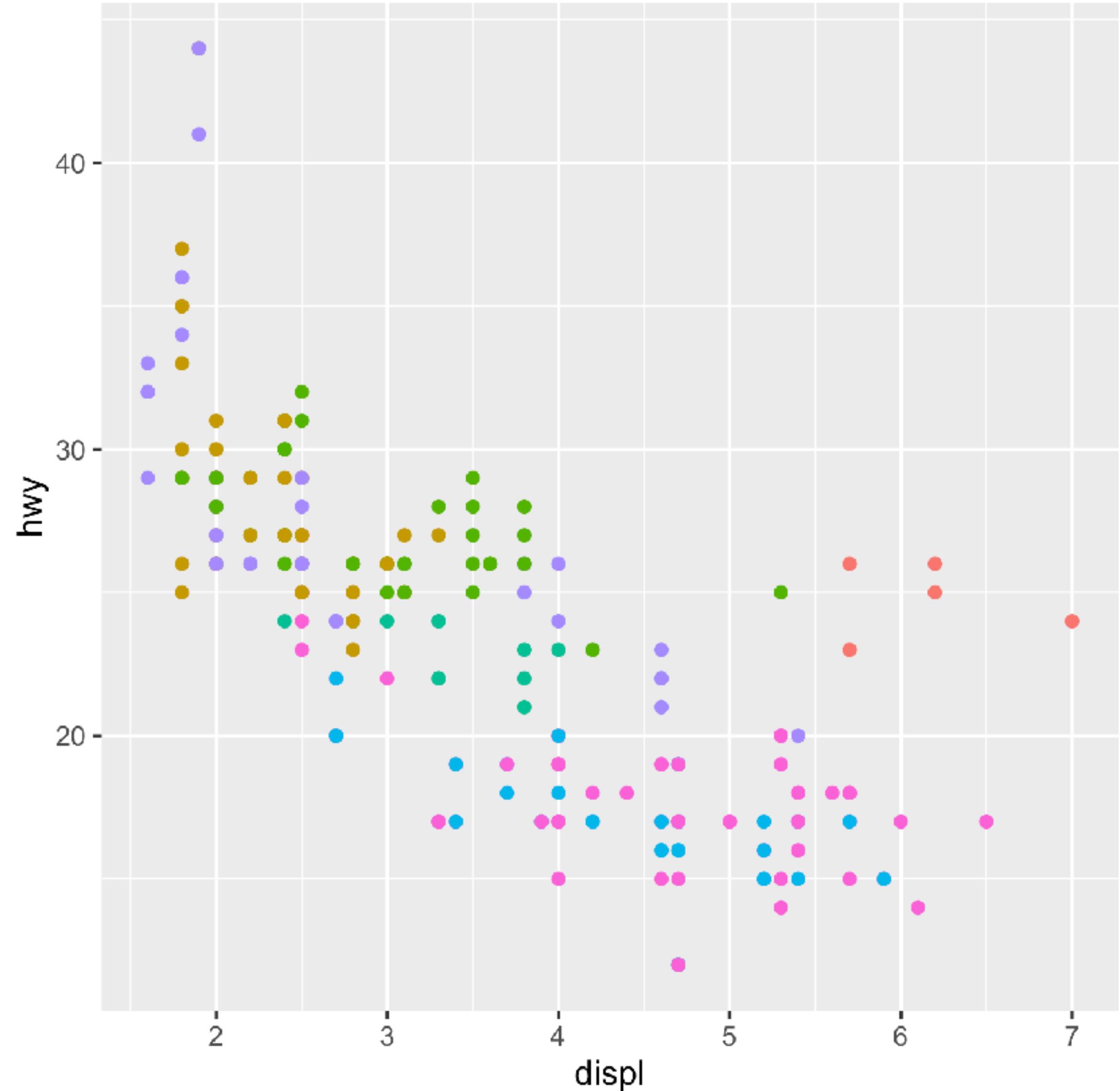
Aesthetics

aesthetic
property

Variable to
map it to

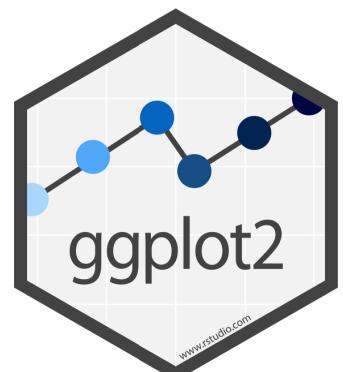
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))  
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, size = class))  
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, shape = class))  
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, alpha = class))
```





Legend added
automatically

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



Your Turn 2

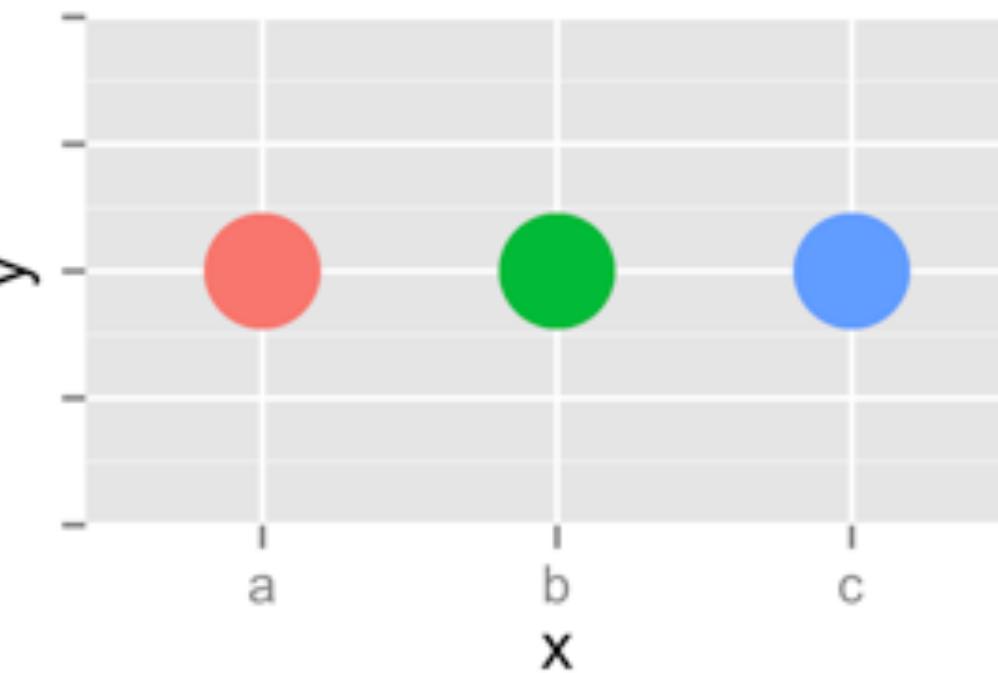
In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

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

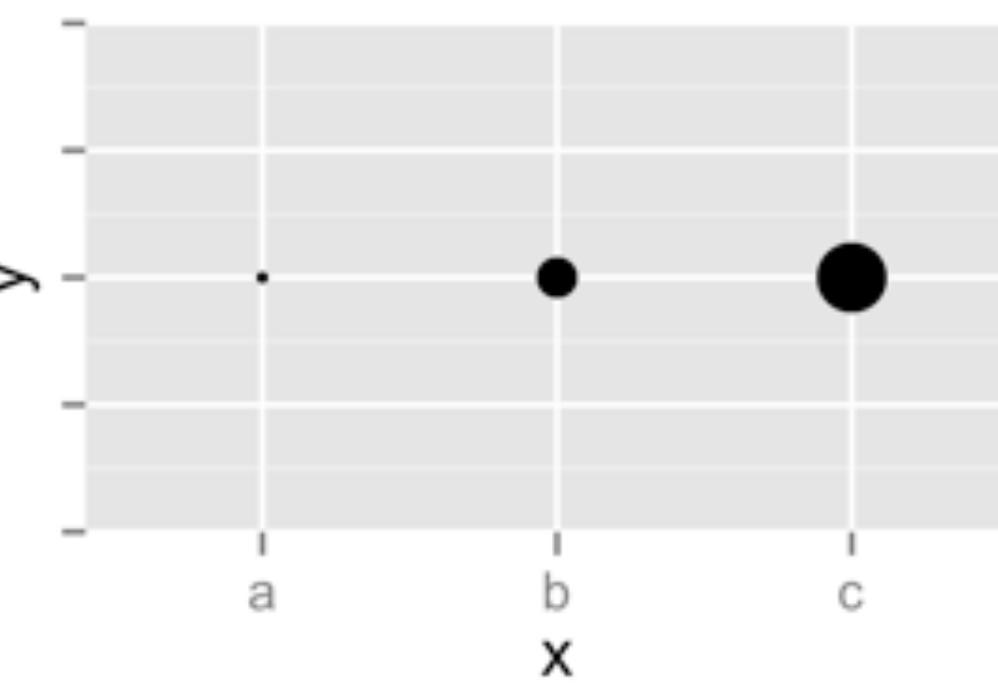
What happens when you use more than one aesthetic?



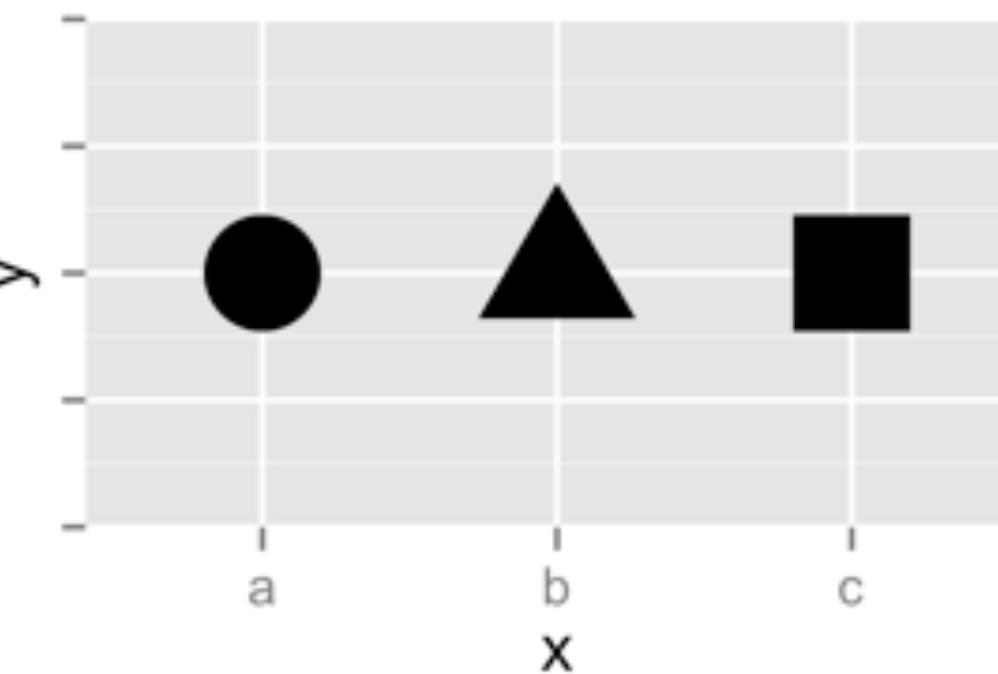
Color



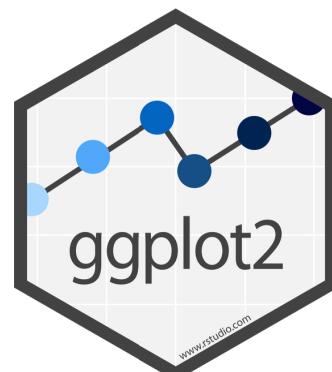
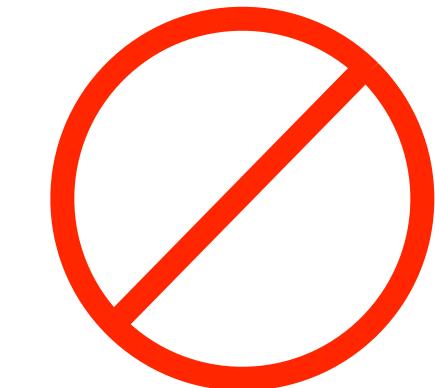
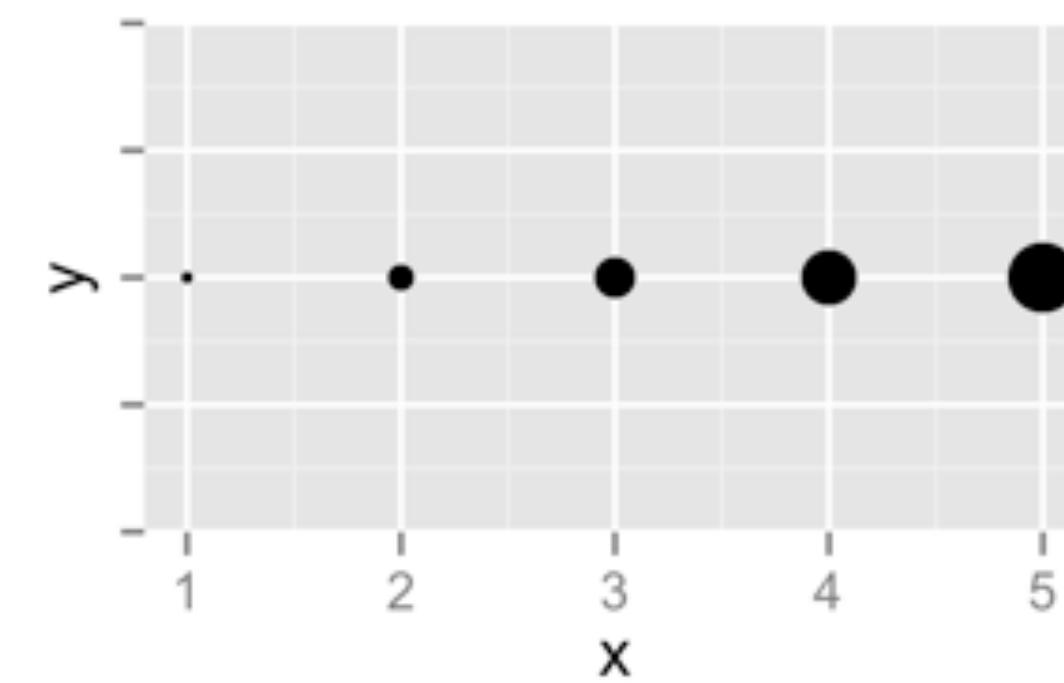
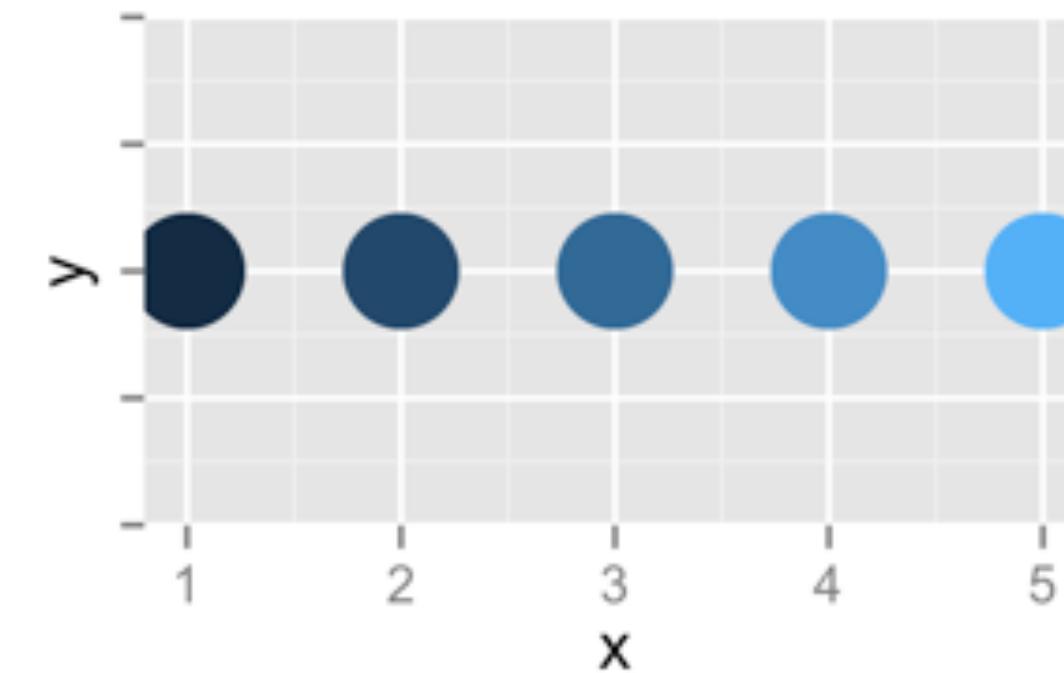
Size



Shape



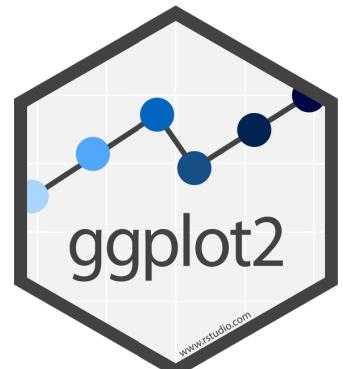
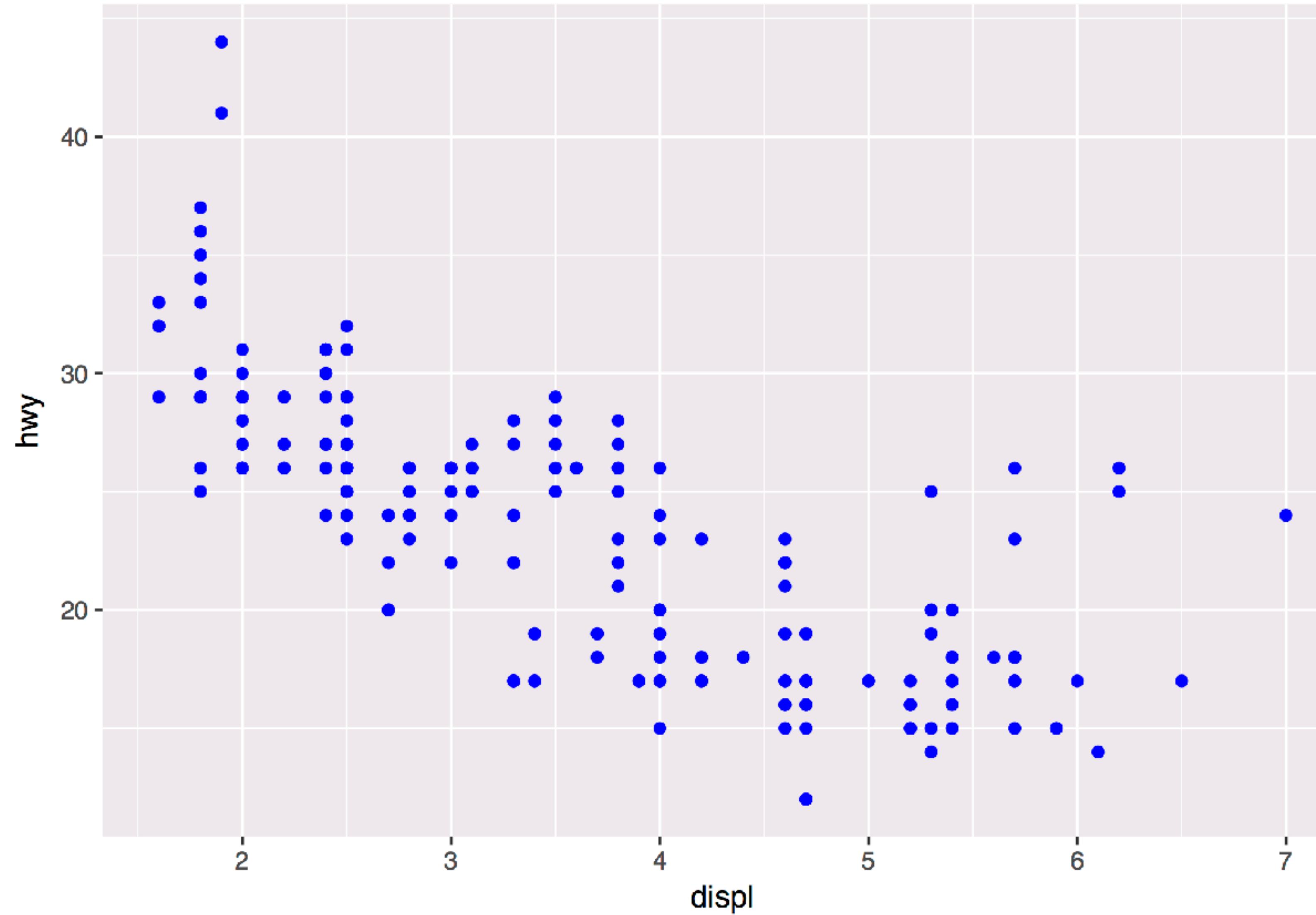
Continuous

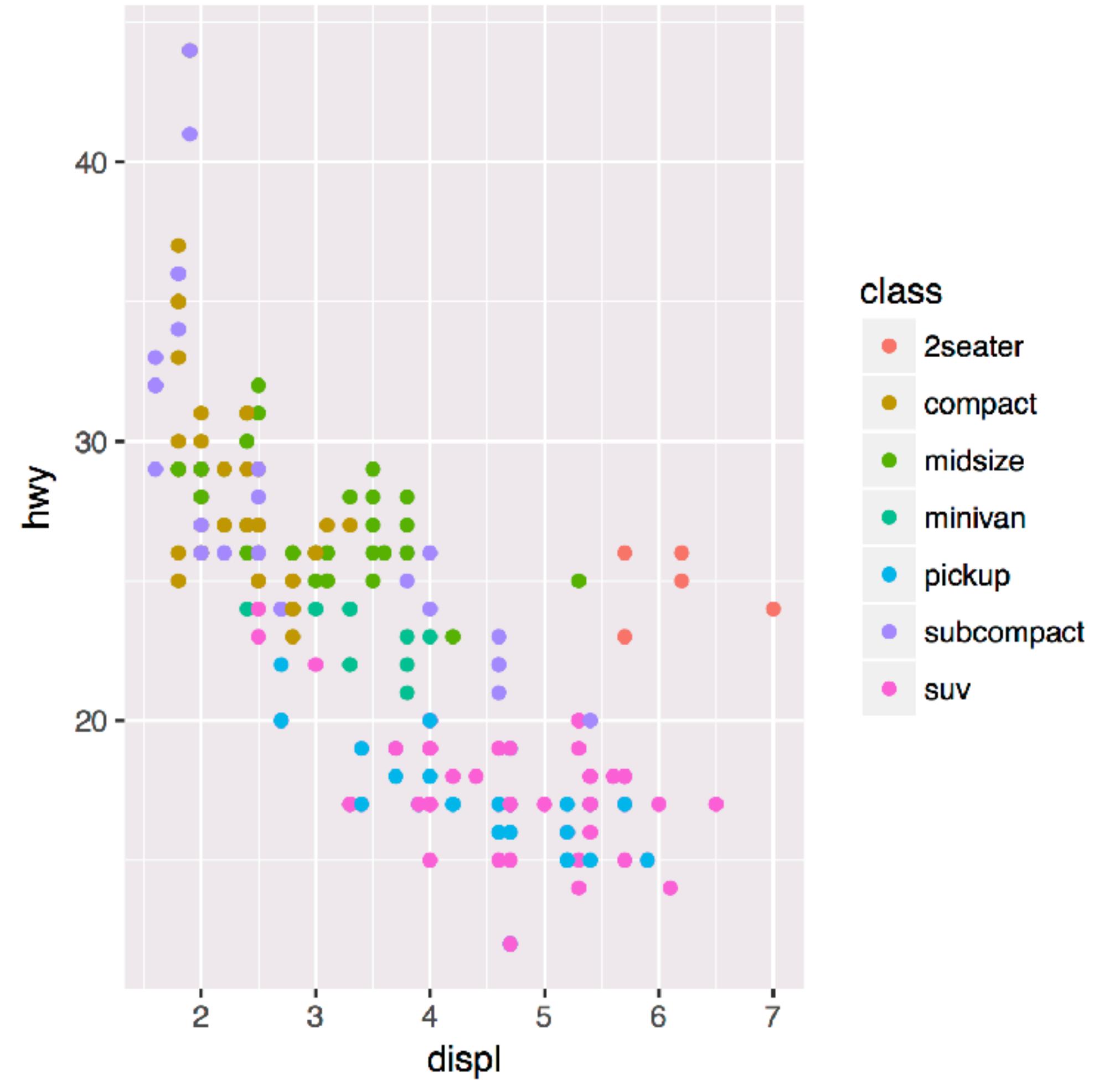


set vs. map

R

How would you make this plot?

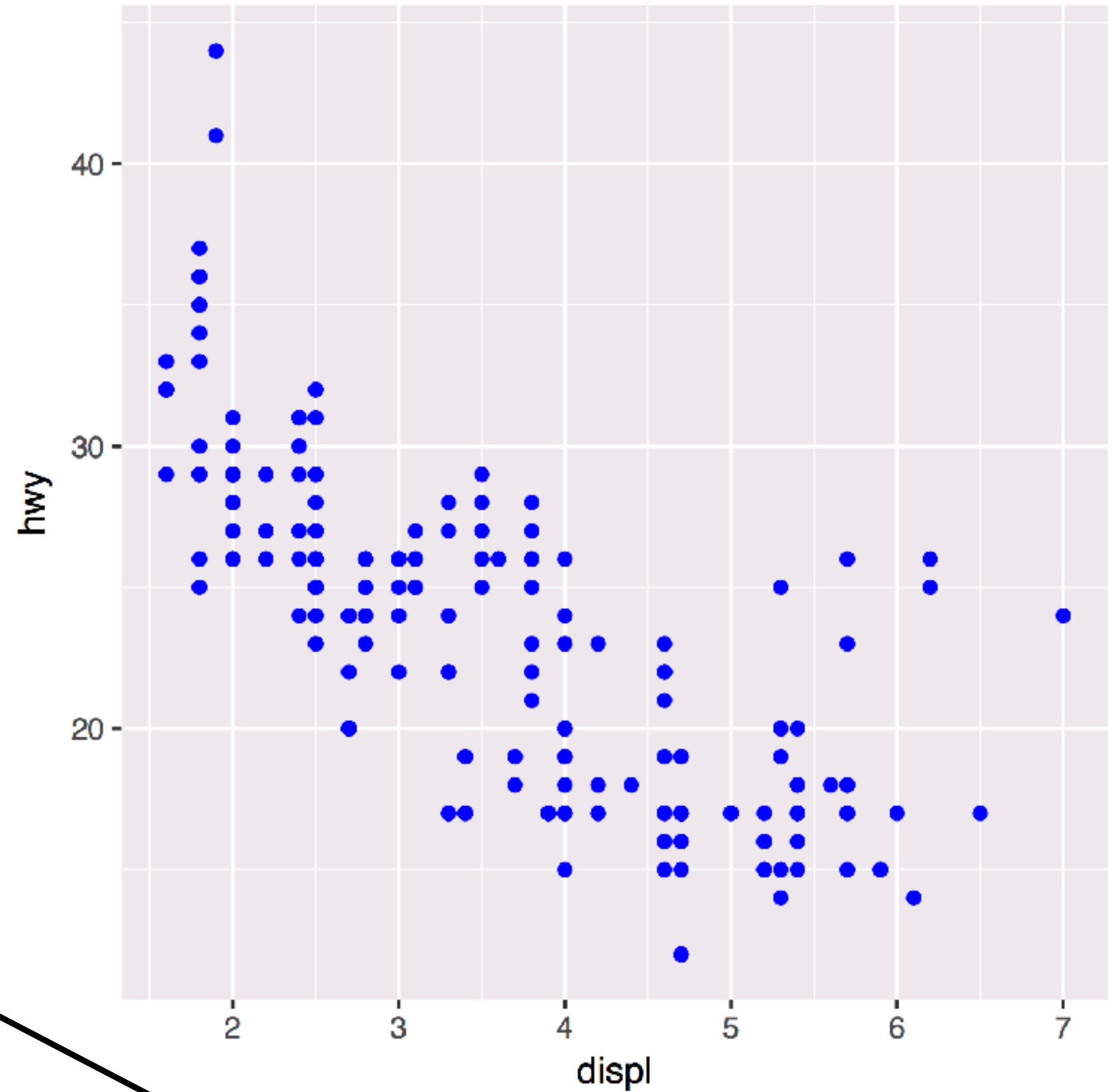




Inside of aes(): maps an aesthetic to a variable

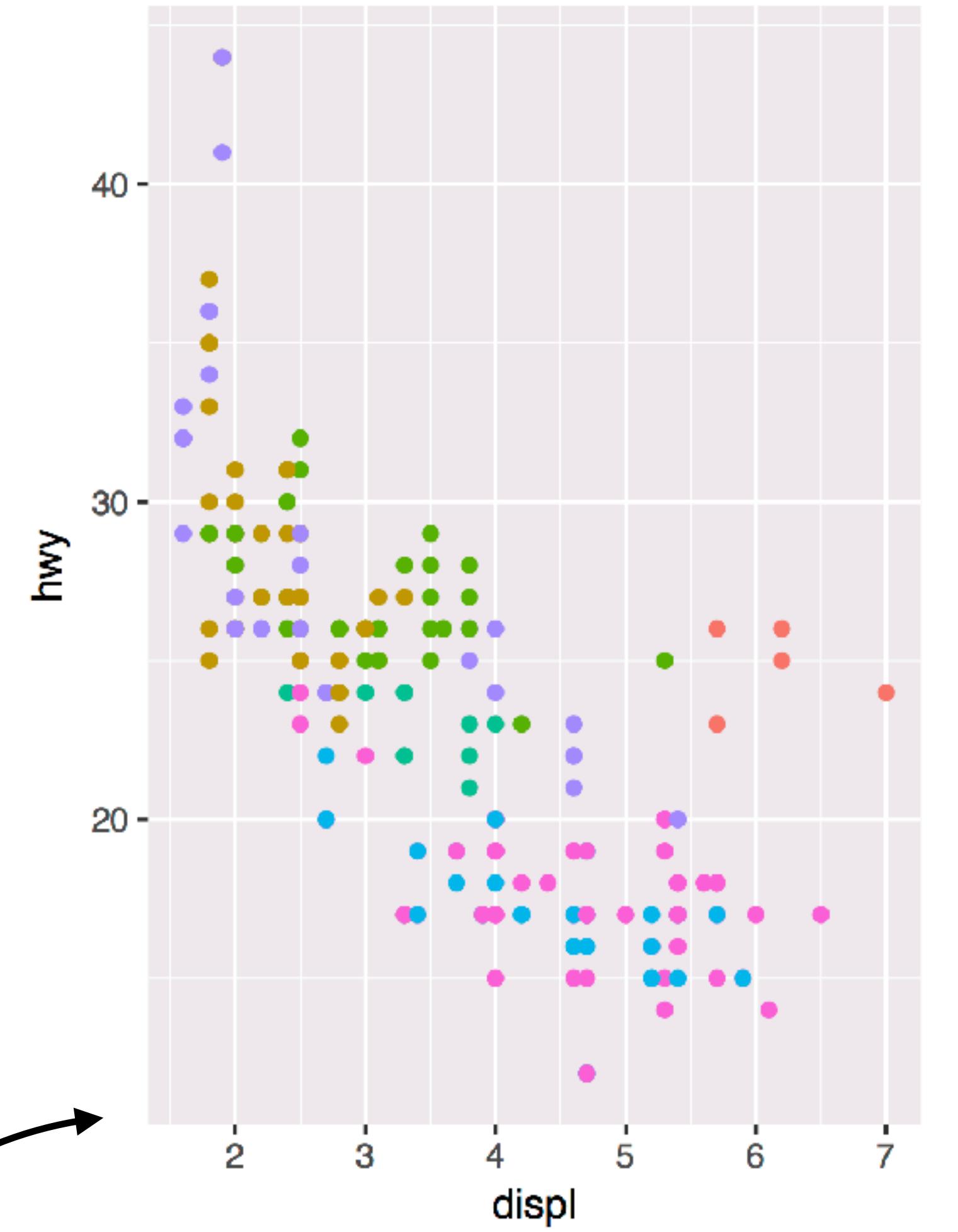
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



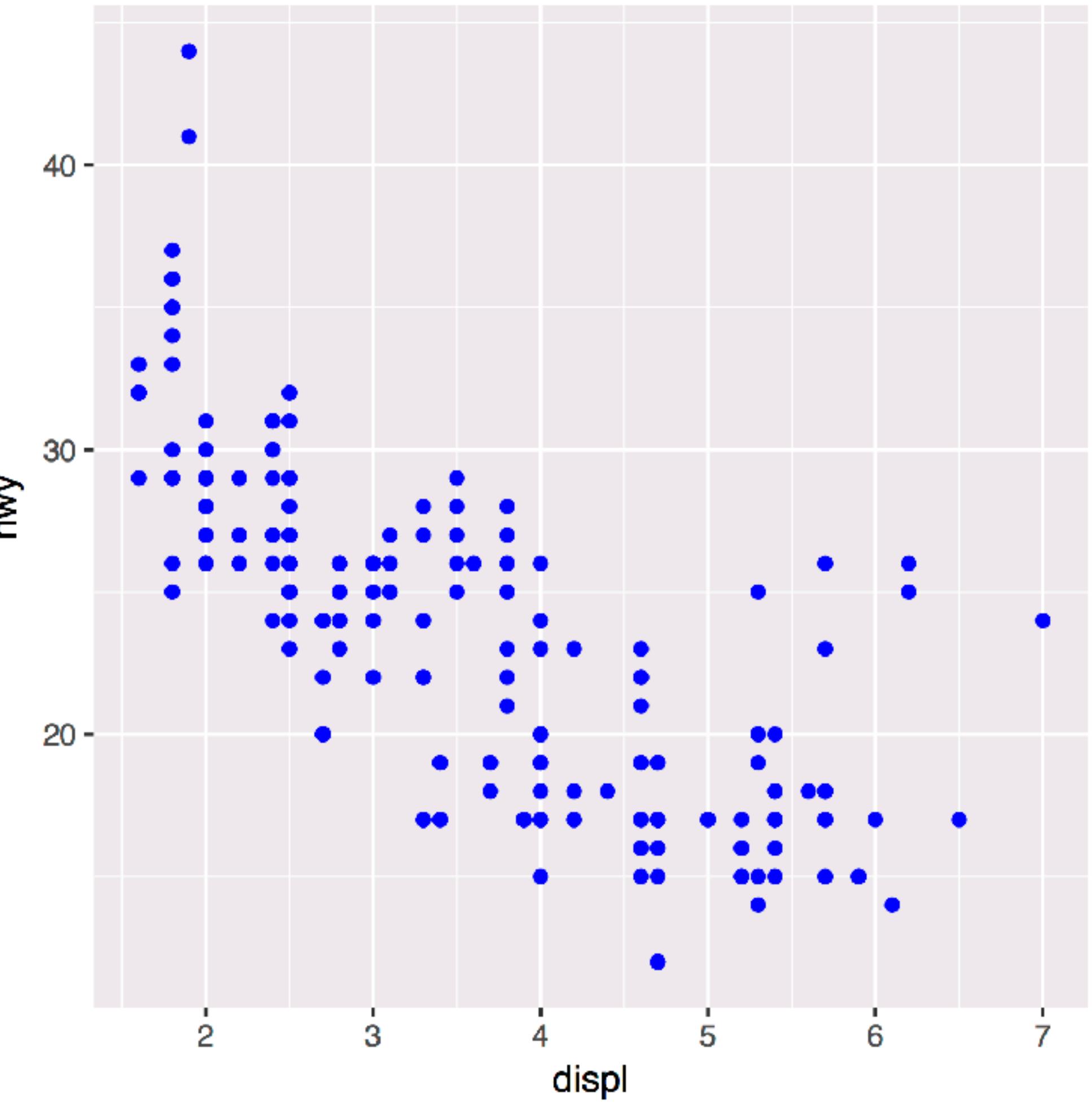
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



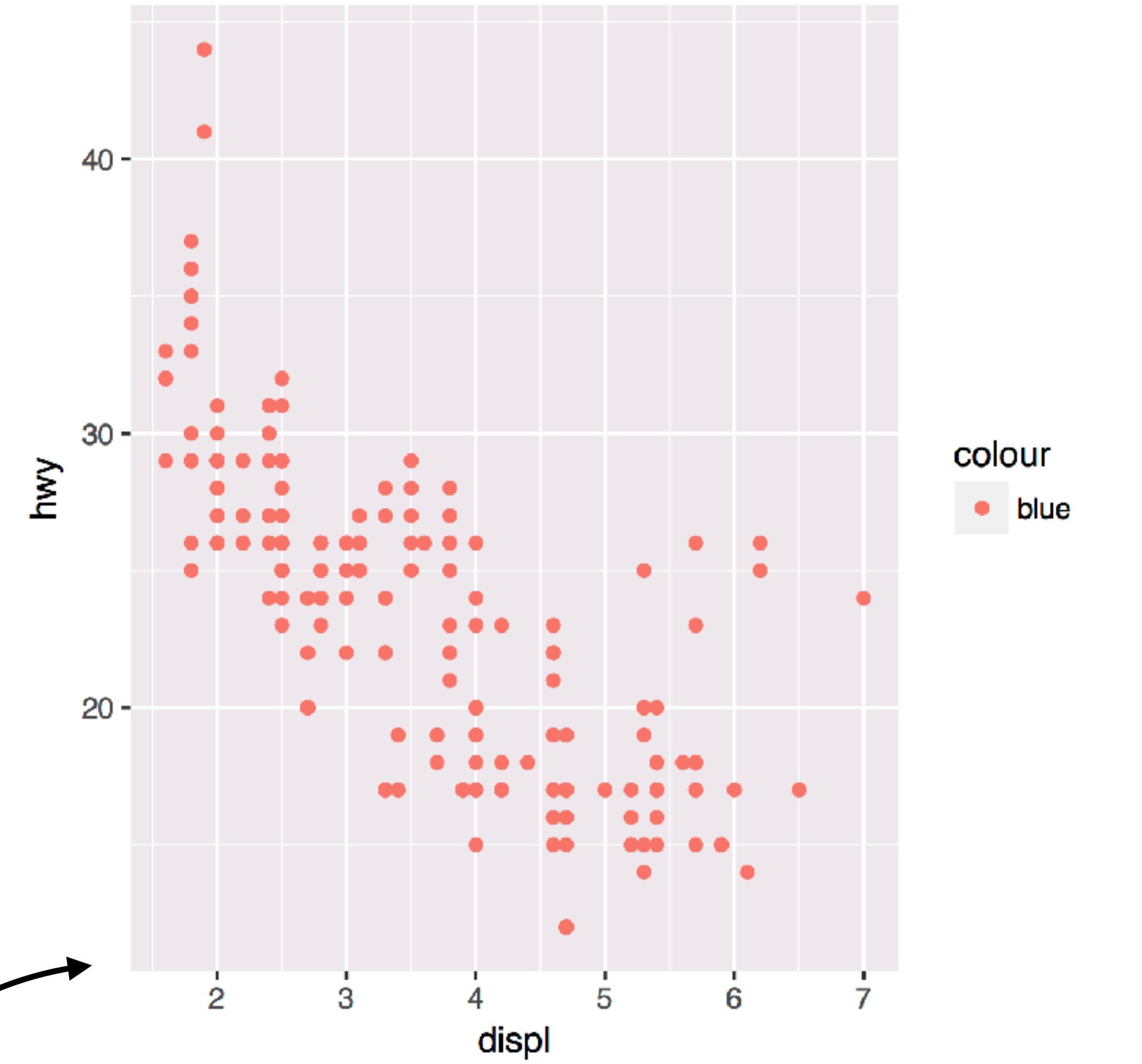
class

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

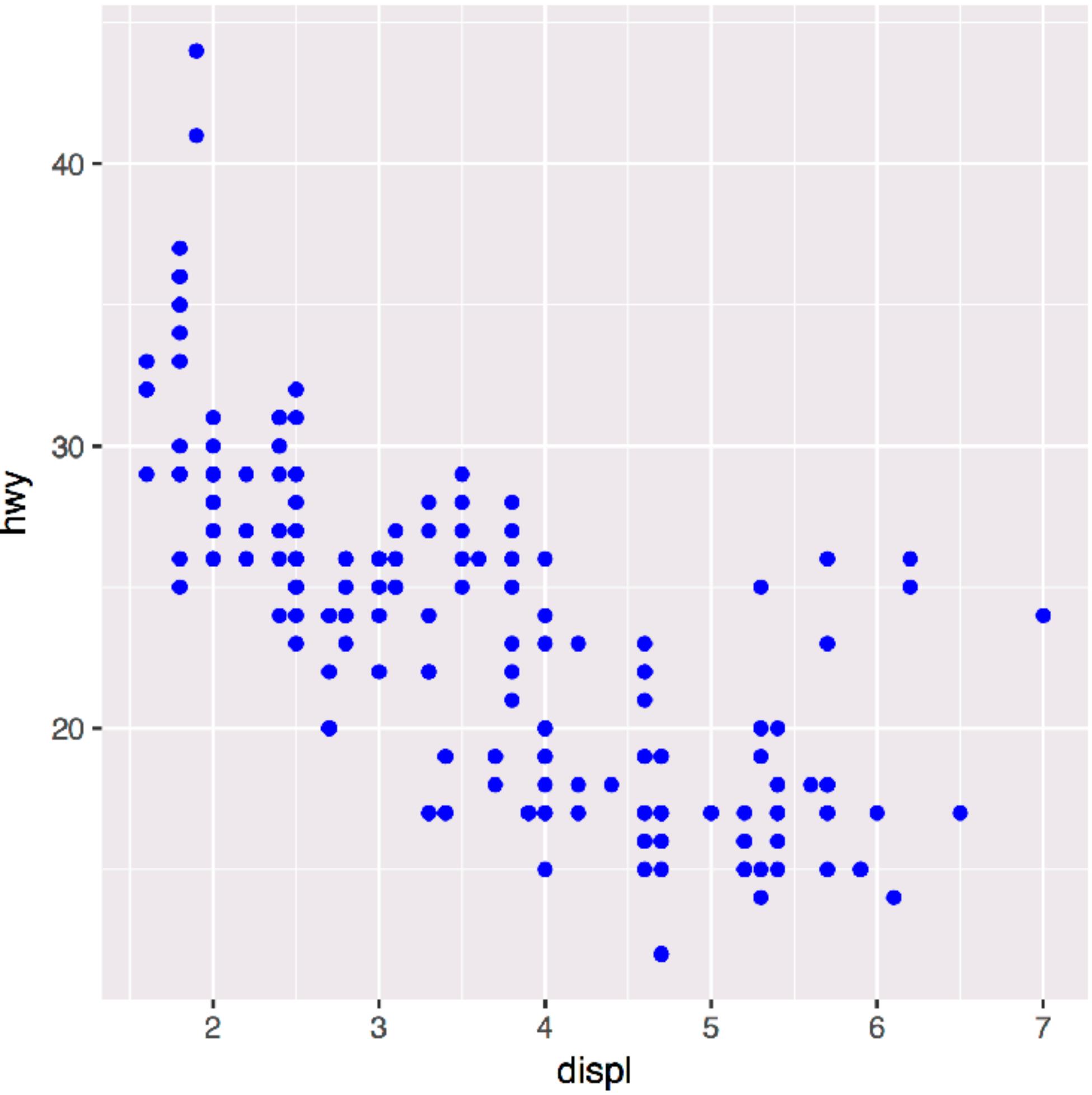


```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



colour
● blue



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

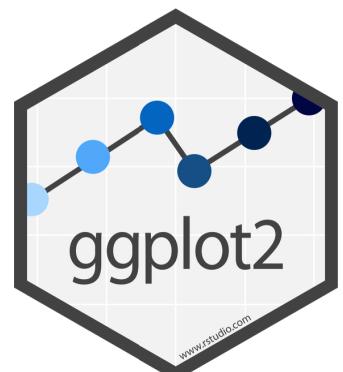
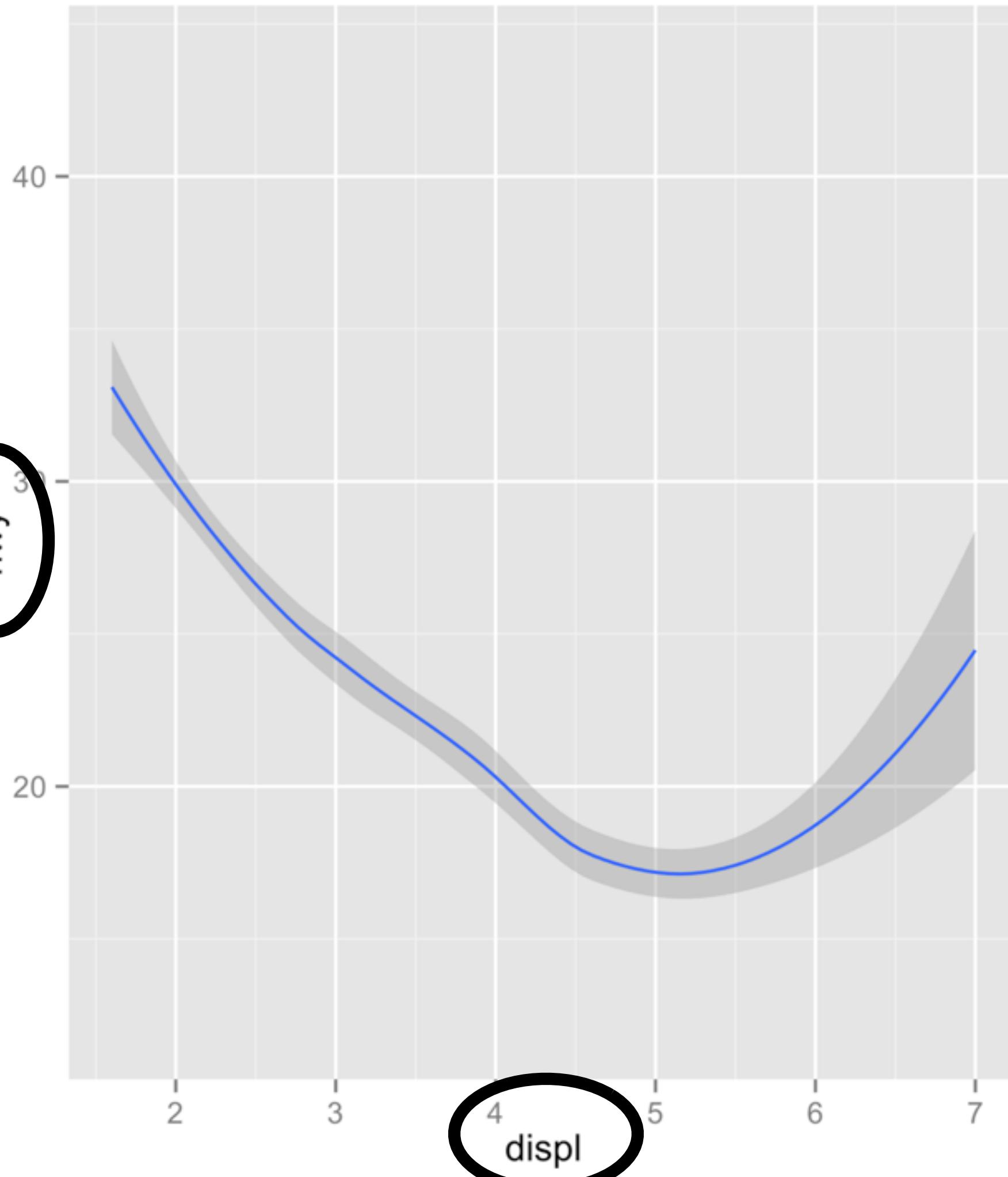
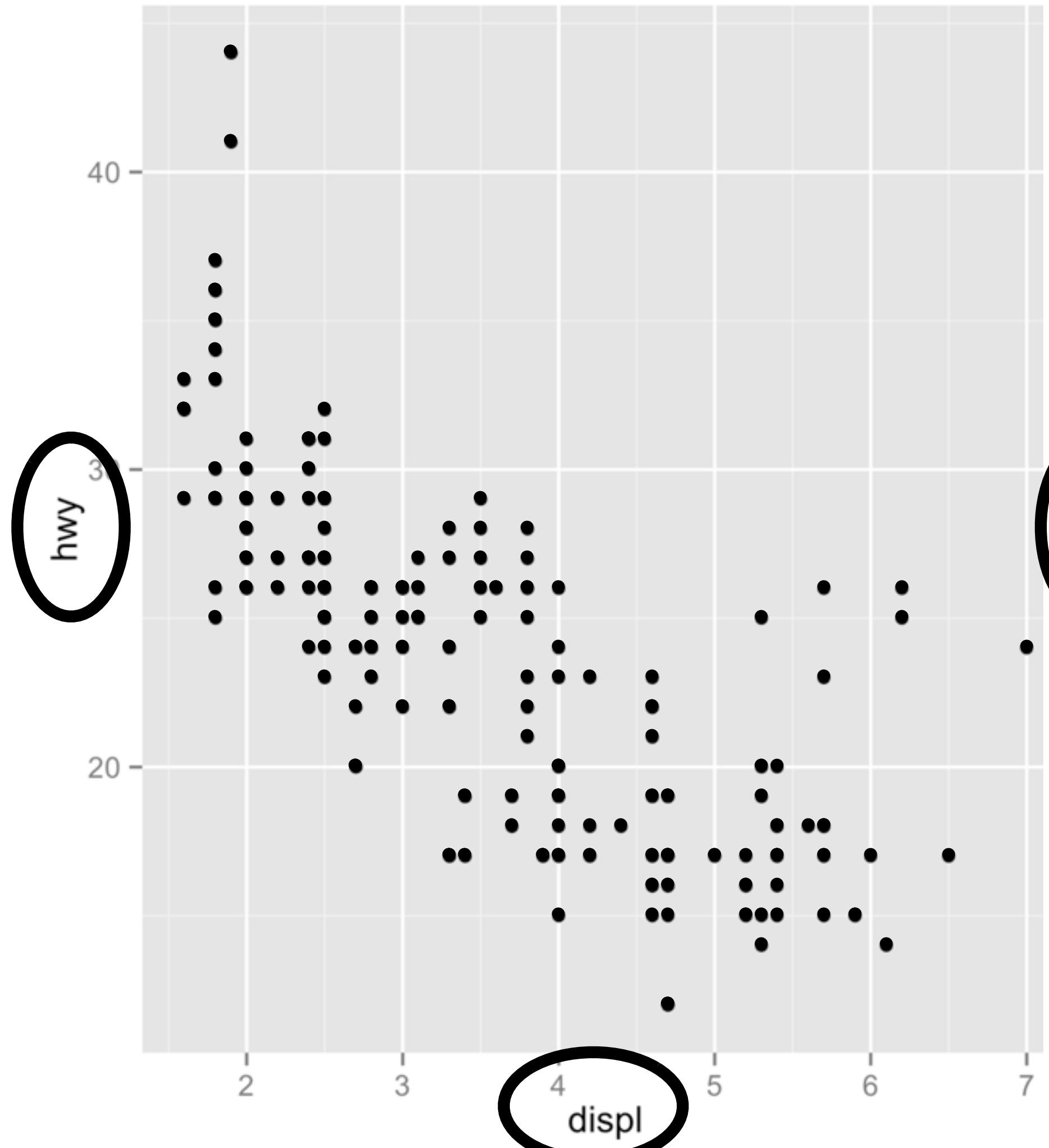
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), color = "blue")
```

Geoms

R

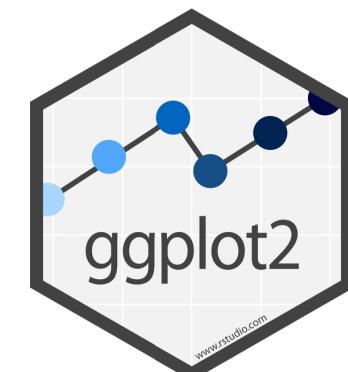
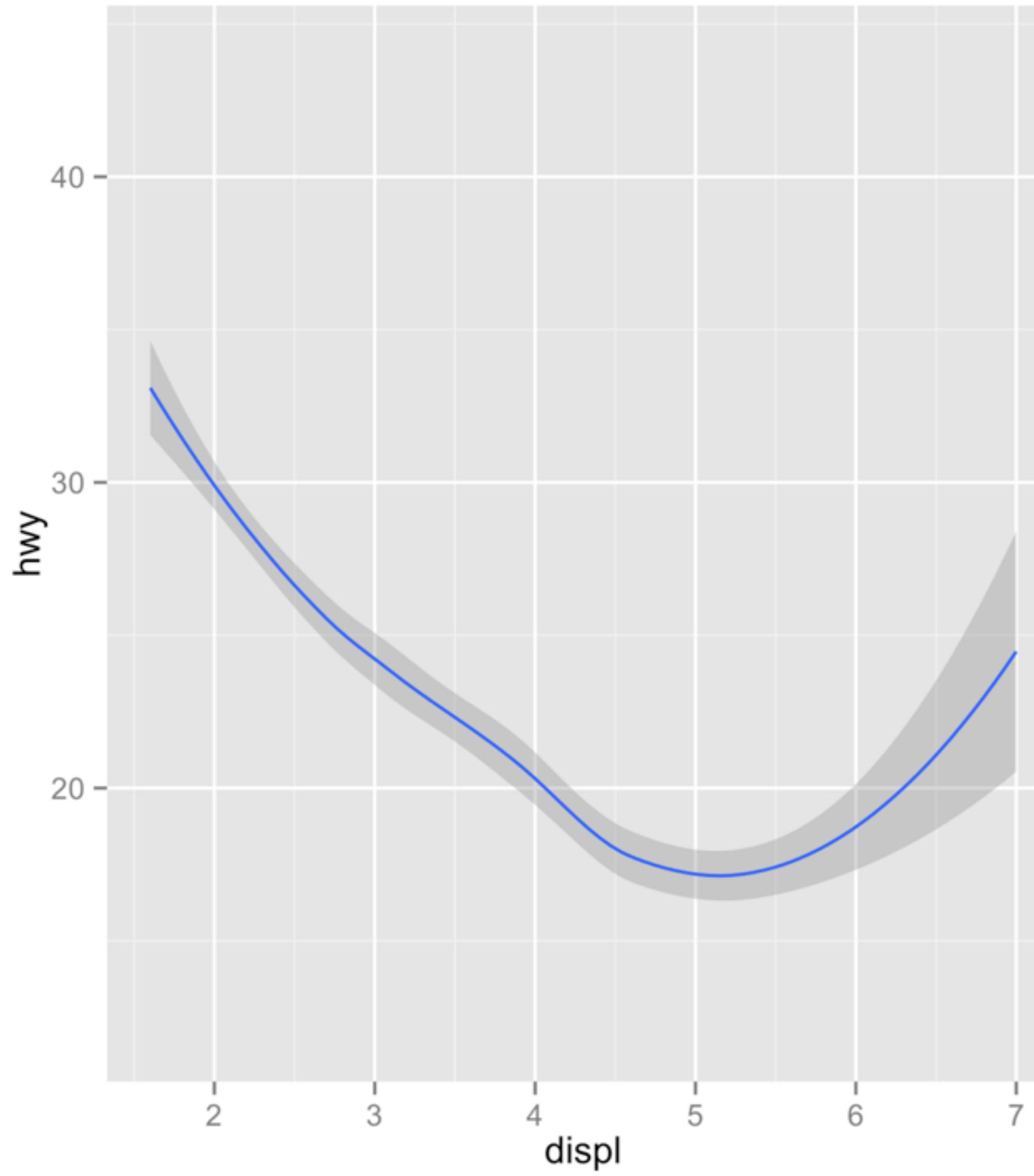
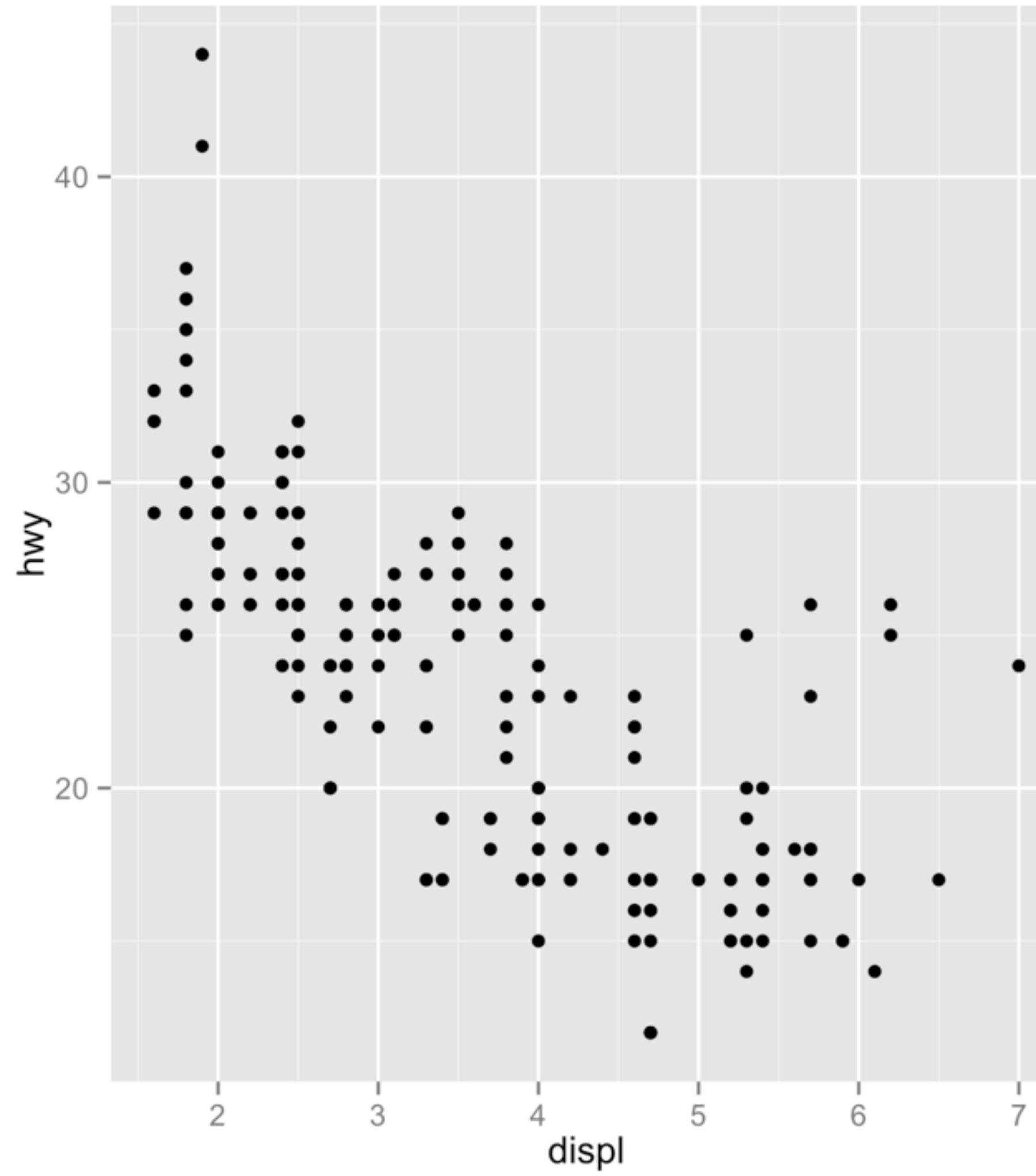
How are these plots similar?

Same: x var , y var , data



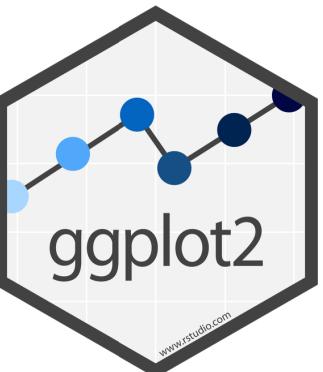
How are these plots different?

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



geoms

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

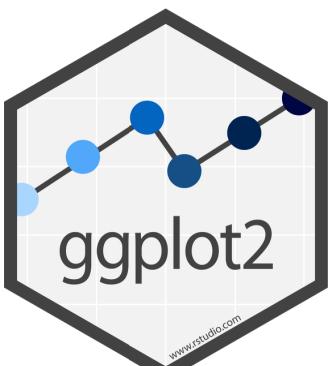


geom_ functions

Each requires a mapping argument.



Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.	
One Variable <ul style="list-style-type: none"> Continuous <ul style="list-style-type: none"> a + <code>geom_area(stat = "bin")</code> x, y, alpha, color, fill, linetype, size b + <code>geom_area(aes(y = ..density..), stat = "bin")</code> a + <code>geom_density(kernel = "gaussian")</code> x, y, alpha, color, fill, linetype, size, weight b + <code>geom_density(aes(y = ..count..))</code> a + <code>geom_dotplot()</code> x, y, alpha, color, fill a + <code>geom_freqpoly()</code> x, y, alpha, color, linetype, size b + <code>geom_freqpoly(aes(y = ..density..))</code> a + <code>geom_histogram(binwidth = 5)</code> x, y, alpha, color, fill, linetype, size, weight b + <code>geom_histogram(aes(y = ..density..))</code> Discrete <ul style="list-style-type: none"> b <- <code>ggplot(mpg, aes(f1))</code> b + <code>geom_bar()</code> x, alpha, color, fill, linetype, size, weight 	Two Variables <ul style="list-style-type: none"> Continuous X, Continuous Y <ul style="list-style-type: none"> f + <code>geom_blank()</code> (Useful for expanding limits) f + <code>geom_jitter()</code> x, y, alpha, color, fill, shape, size f + <code>geom_point()</code> x, y, alpha, color, fill, shape, size f + <code>geom_quantile()</code> x, y, alpha, color, linetype, size, weight f + <code>geom_rug(sides = "bl")</code> alpha, color, linetype, size f + <code>geom_smooth(method = lm)</code> x, y, alpha, color, fill, linetype, size, weight f + <code>geom_text(aes(label = cty))</code> x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust Continuous Function <ul style="list-style-type: none"> j <- <code>ggplot(economics, aes(date, unemploy))</code> j + <code>geom_area()</code> x, y, alpha, color, fill, linetype, size j + <code>geom_line()</code> x, y, alpha, color, linetype, size j + <code>geom_step(direction = "hv")</code> x, y, alpha, color, linetype, size
Graphical Primitives <ul style="list-style-type: none"> map <- <code>map_data("state")</code> c <- <code>ggplot(map, aes(long, lat))</code> c + <code>geom_polygon(aes(group = group))</code> x, y, alpha, color, fill, linetype, size d <- <code>ggplot(economics, aes(date, unemploy))</code> d + <code>geom_path(lineend = "butt", linejoin = "round", linemitre = 1)</code> x, y, alpha, color, linetype, size d + <code>geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))</code> x, ymax, ymin, alpha, color, fill, linetype, size e <- <code>ggplot(seals, aes(x = long, y = lat))</code> e + <code>geom_segment(aes(xend = long + delta_long, yend = lat + delta_lat))</code> x, xend, y, yend, alpha, color, linetype, size e + <code>geom_rect(aes(xmin = long, ymin = lat, xmax = long + delta_long, ymax = lat + delta_lat))</code> xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size 	Visualizing error <ul style="list-style-type: none"> df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2) k <- <code>ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))</code> k + <code>geom_crossbar(fatten = 2)</code> x, y, ymax, ymin, alpha, color, fill, linetype, size k + <code>geom_errorbar()</code> lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight k + <code>geom_linerange()</code> x, y, min, max, alpha, color, linetype, size k + <code>geom_pointrange()</code> x, y, min, max, alpha, color, fill, linetype, shape, size
Discrete X, Continuous Y <ul style="list-style-type: none"> g <- <code>ggplot(mpg, aes(class, hwy))</code> g + <code>geom_bar(stat = "identity")</code> x, y, alpha, color, fill, linetype, size, weight g + <code>geom_boxplot()</code> lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight g + <code>geom_dotplot(binaxis = "y", stackdir = "center")</code> x, y, alpha, color, fill g + <code>geom_violin(scale = "area")</code> x, y, alpha, color, fill, linetype, size, weight 	Maps <ul style="list-style-type: none"> data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests))) map <- <code>map_data("state")</code> l <- <code>ggplot(data, aes(fill = murder))</code> l + <code>geom_map(aes(map_id = state), map = map) + expand_limits(x = map\$long, y = map\$lat)</code> map_id, alpha, color, fill, linetype, size
Discrete X, Discrete Y <ul style="list-style-type: none"> h <- <code>ggplot(diamonds, aes(cut, color))</code> h + <code>geom_jitter()</code> x, y, alpha, color, fill, shape, size 	Three Variables <ul style="list-style-type: none"> seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)) m <- <code>ggplot(seals, aes(long, lat))</code> m + <code>geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)</code> x, y, alpha, fill (fast) m + <code>geom_contour(aes(z = z))</code> x, y, z, alpha, colour, linetype, size, weight m + <code>geom_tile(aes(fill = z))</code> x, y, alpha, color, fill, linetype, size (slow)



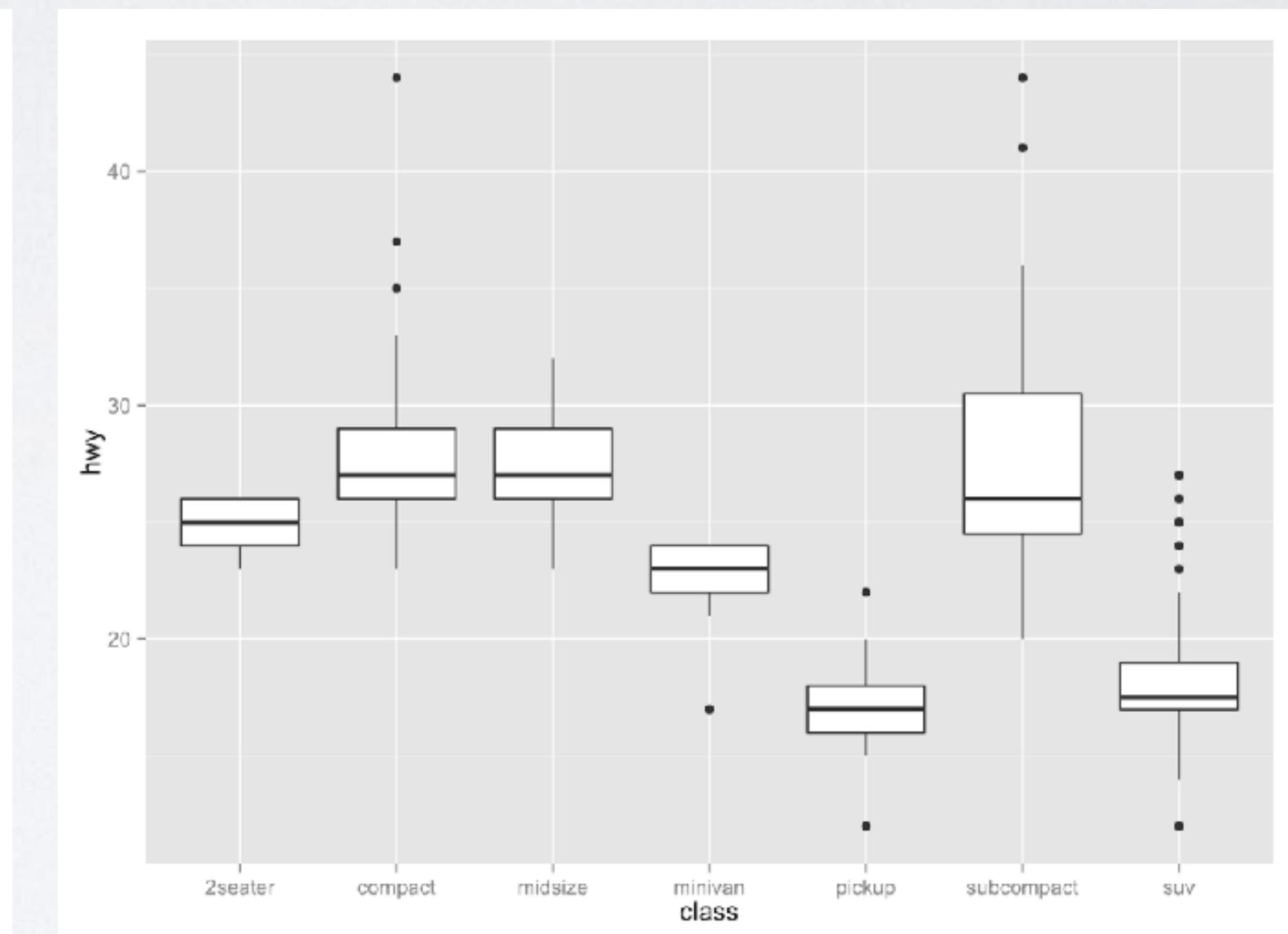
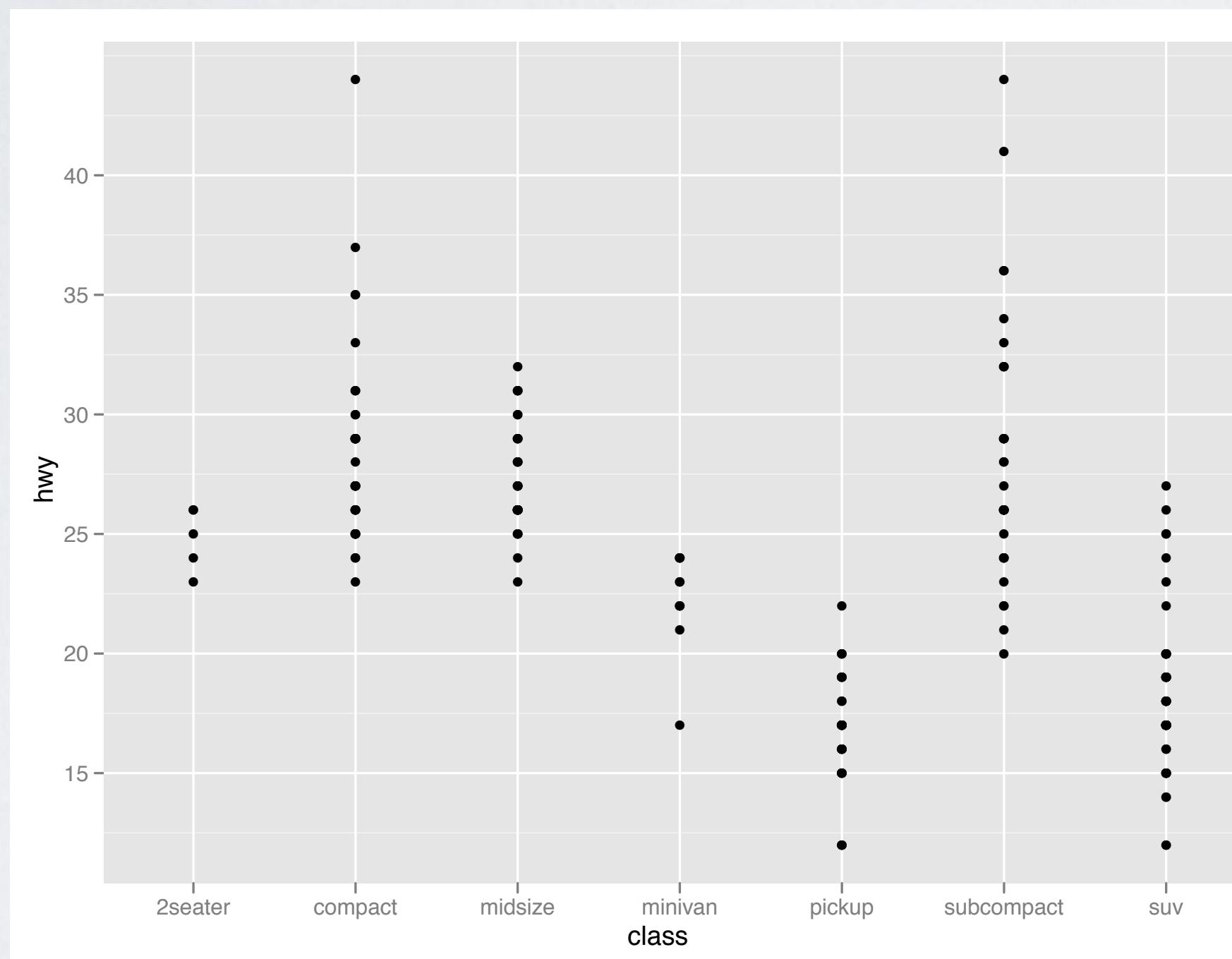
Your Turn

Pair up.



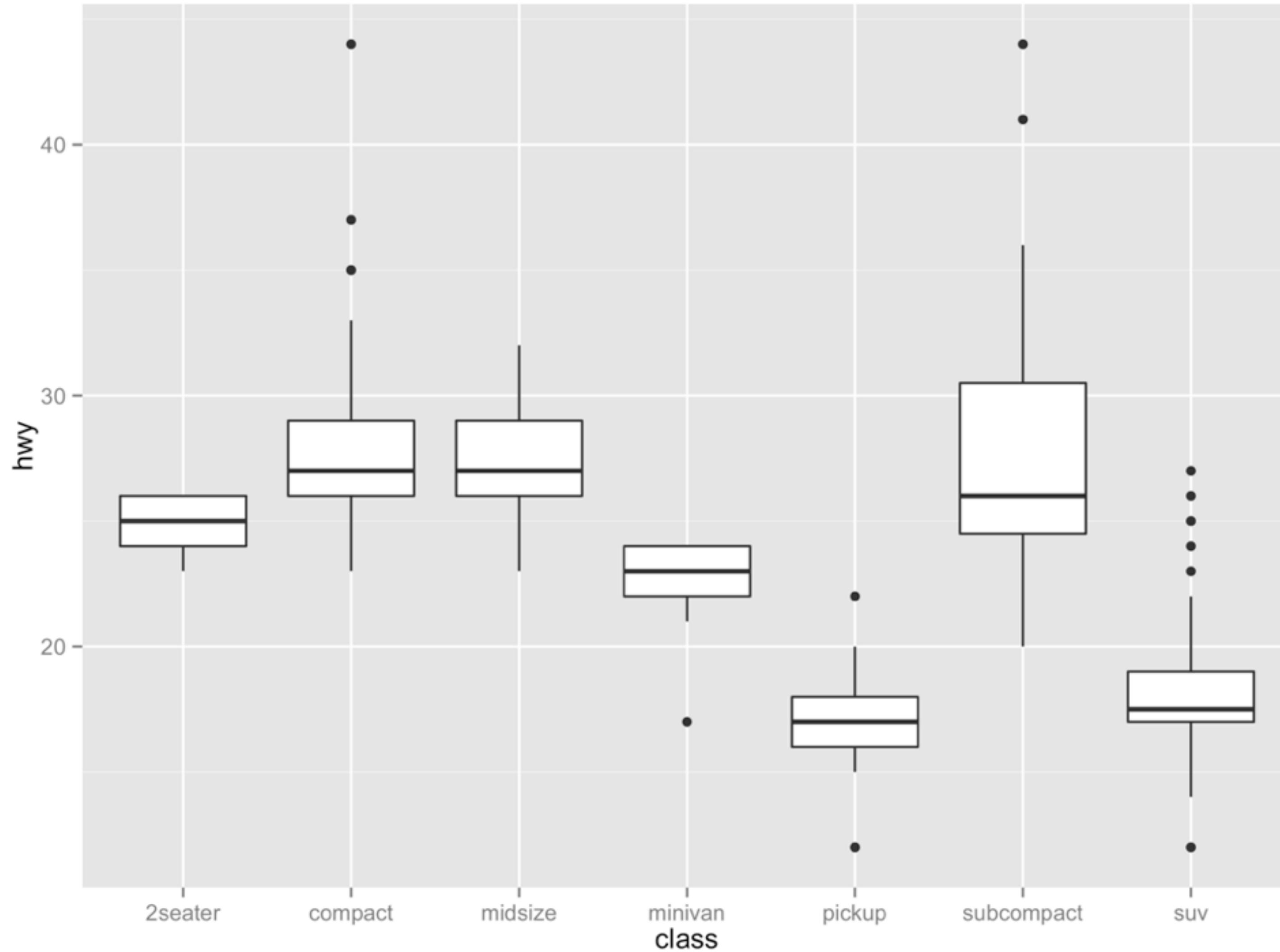
Your Turn 3

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

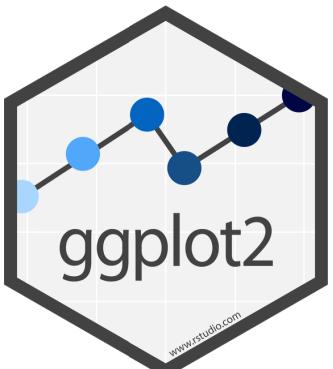


```
ggplot(mpg) + geom_point(aes(class, hwy))
```



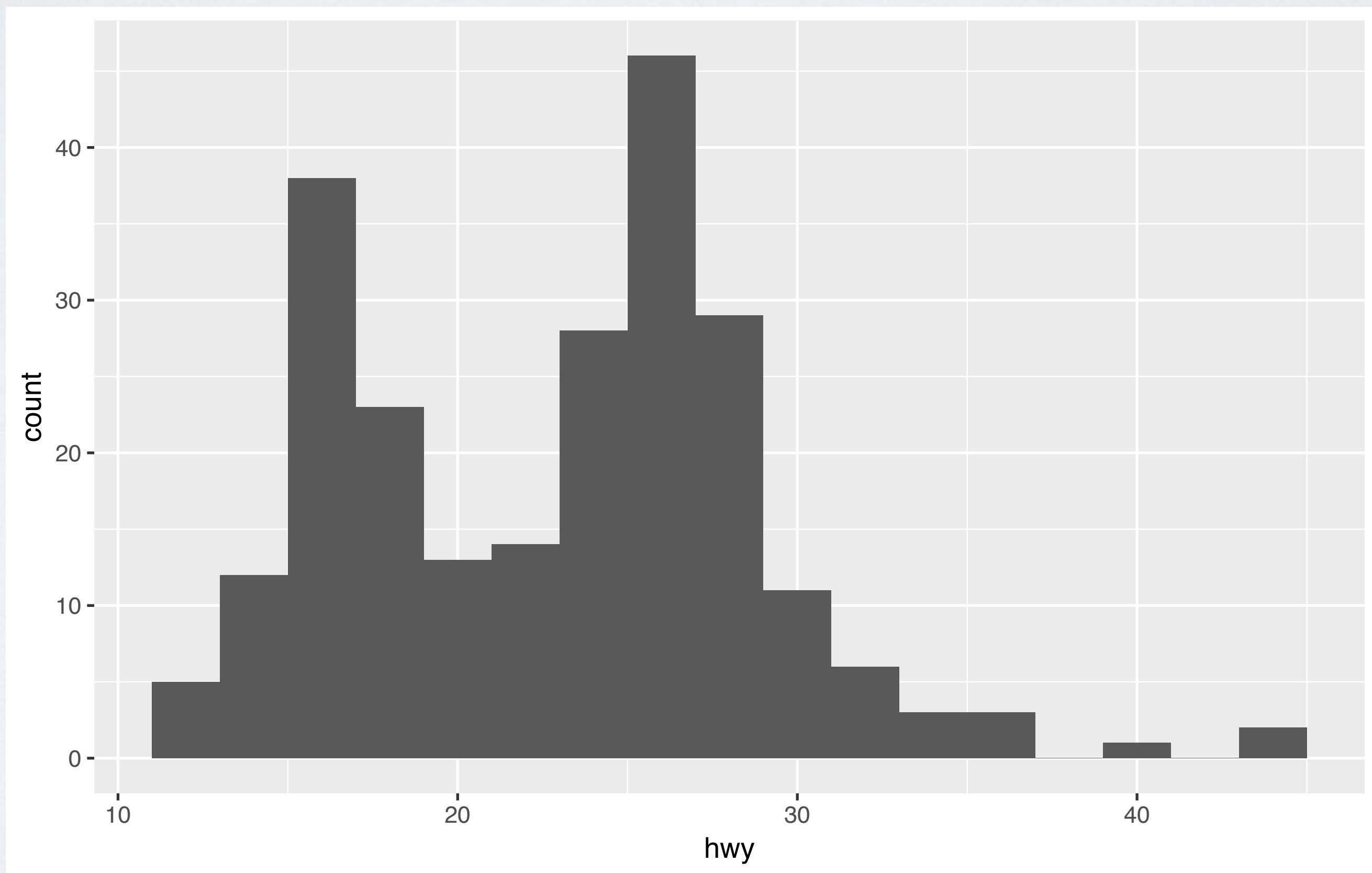


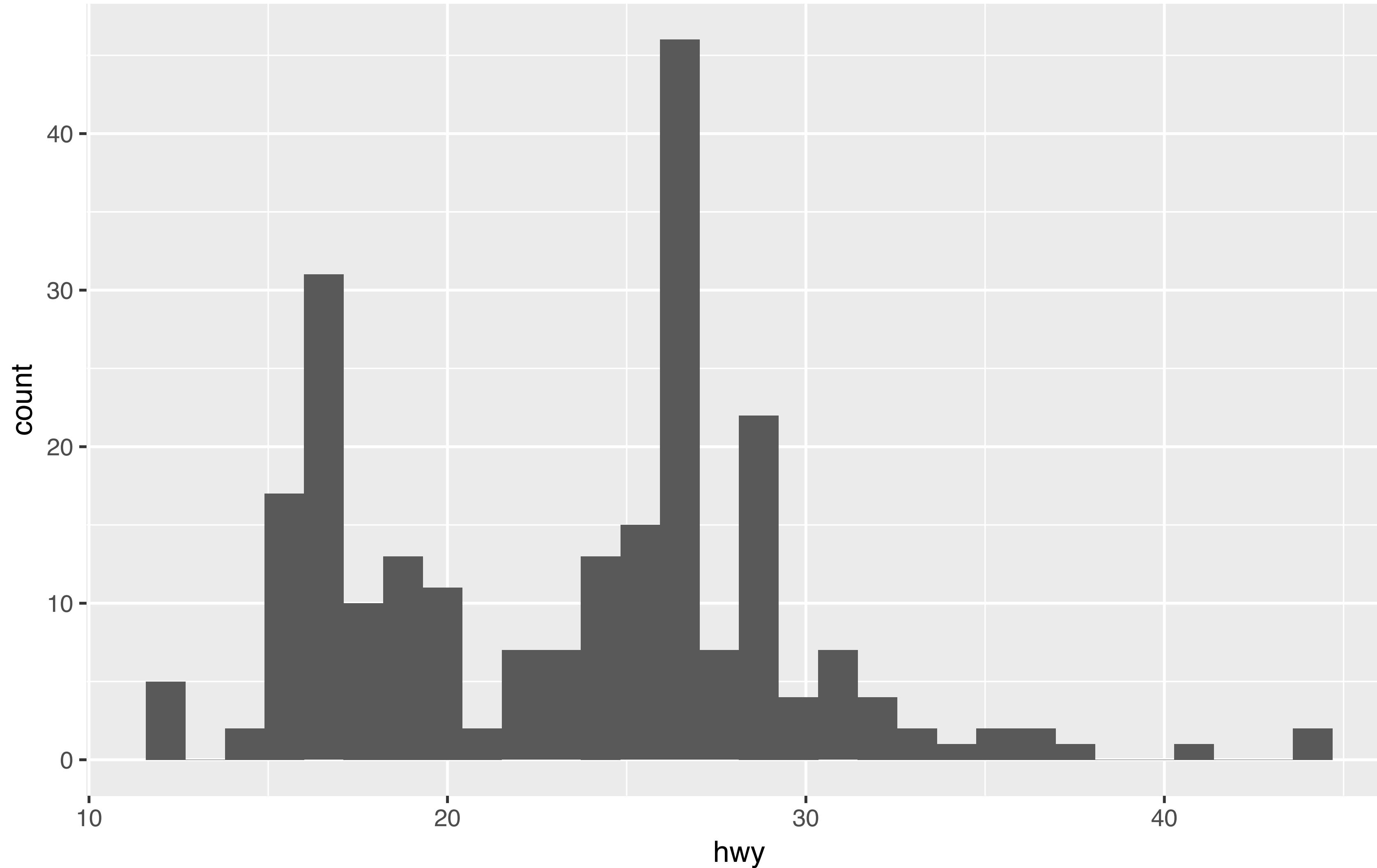
```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = class, y = hwy))
```



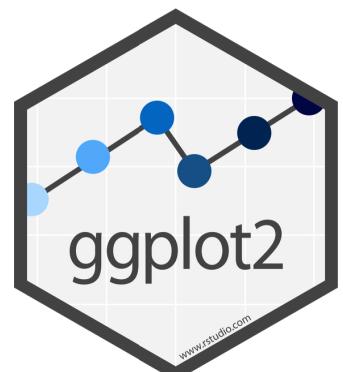
Your Turn 4

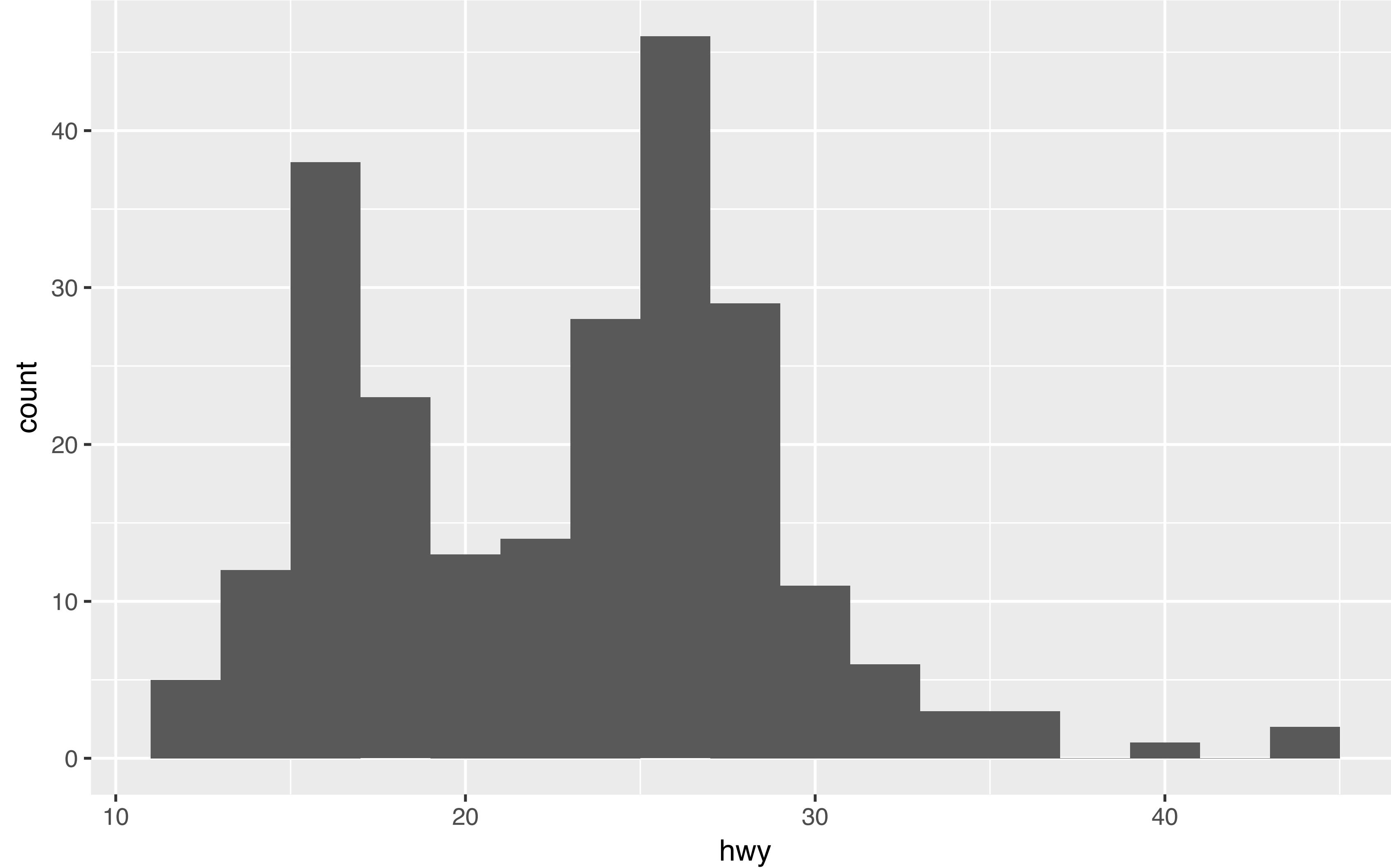
With your partner, make the histogram of `hwy` below. Use the cheatsheet. Hint: do not supply a `y` variable.



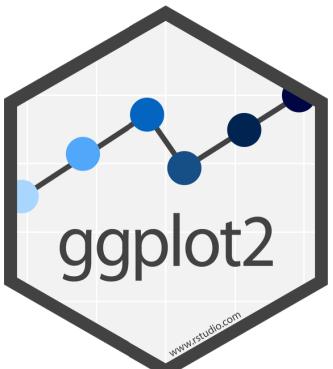


```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy))
```



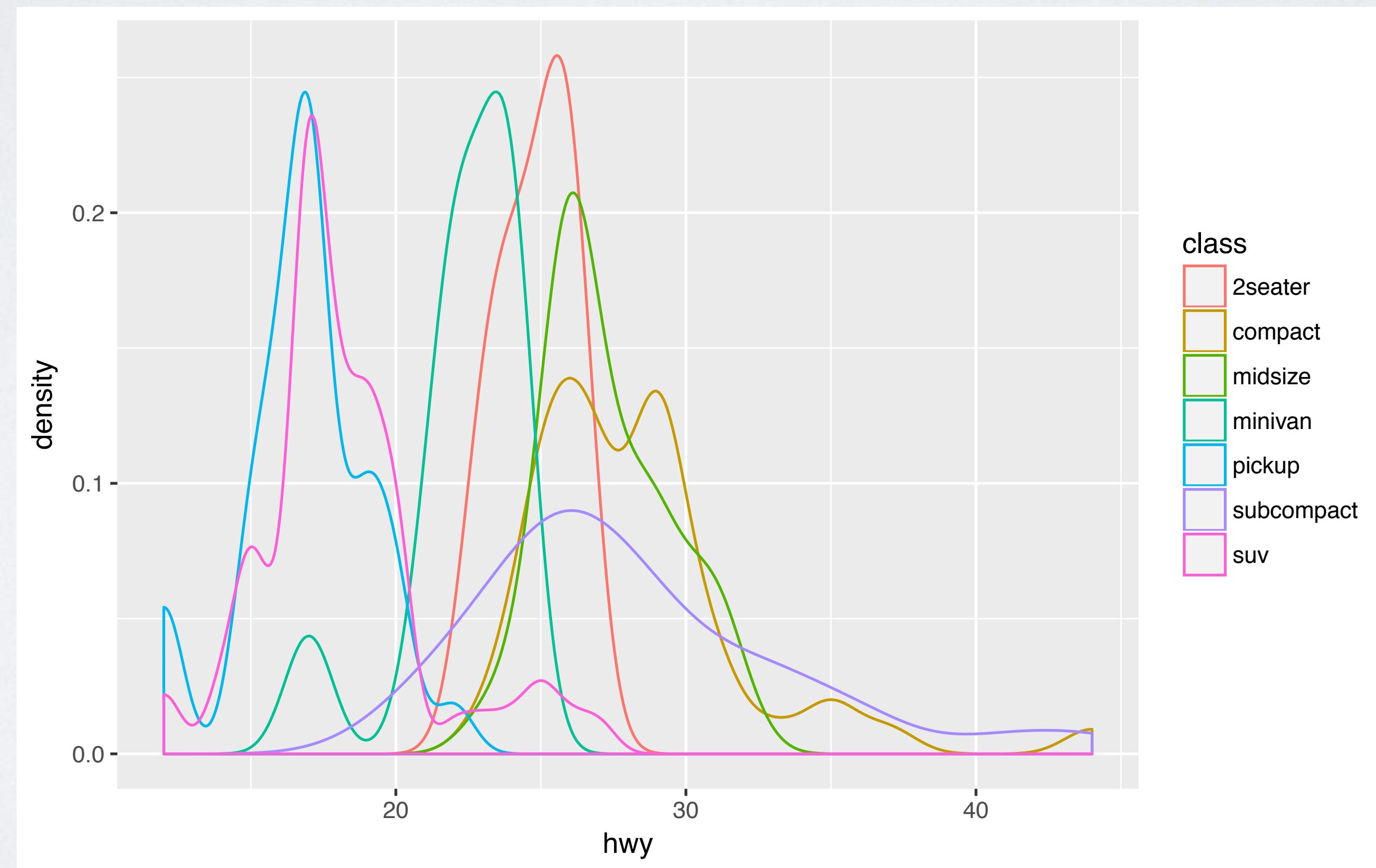


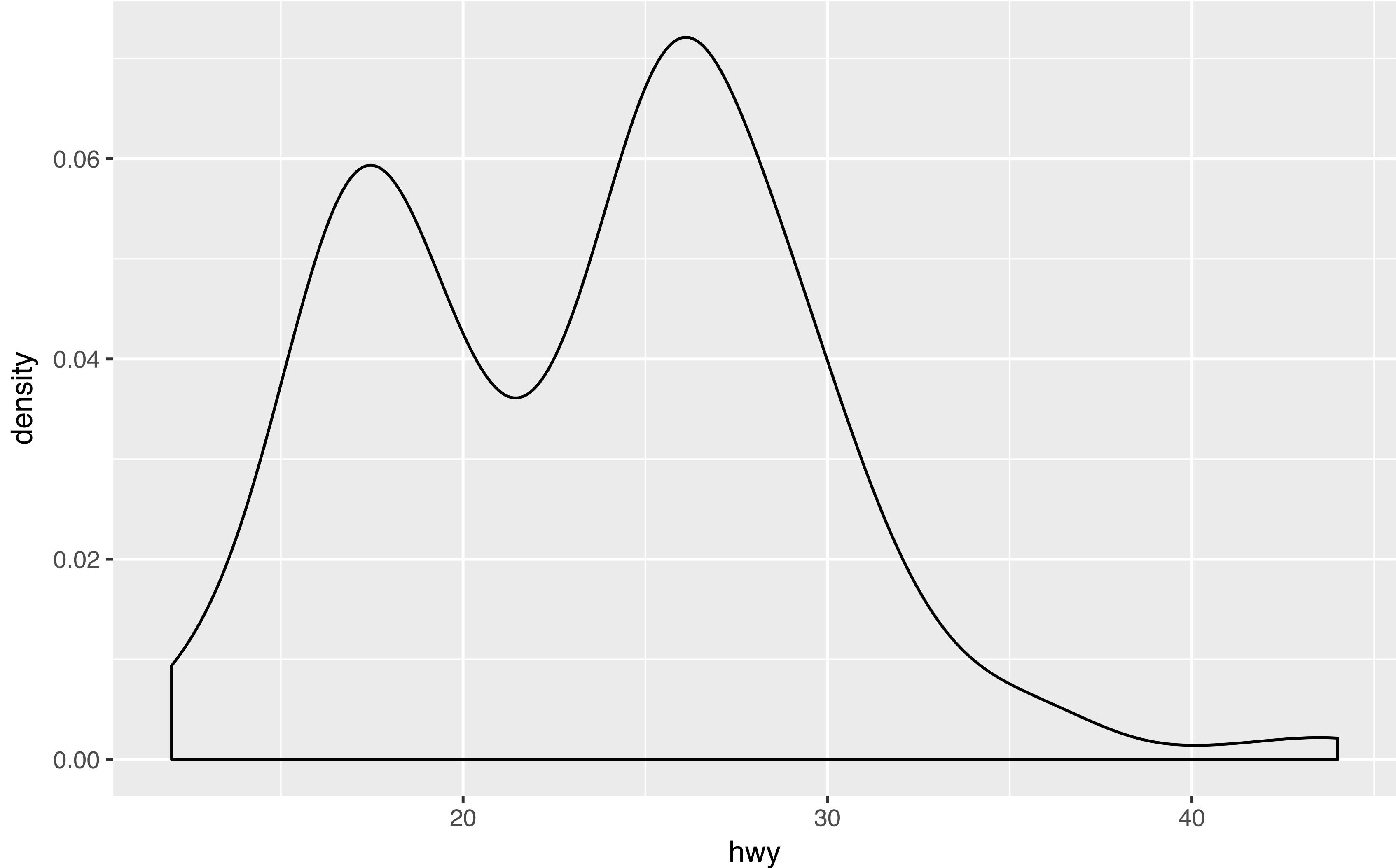
```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy), binwidth = 2)
```



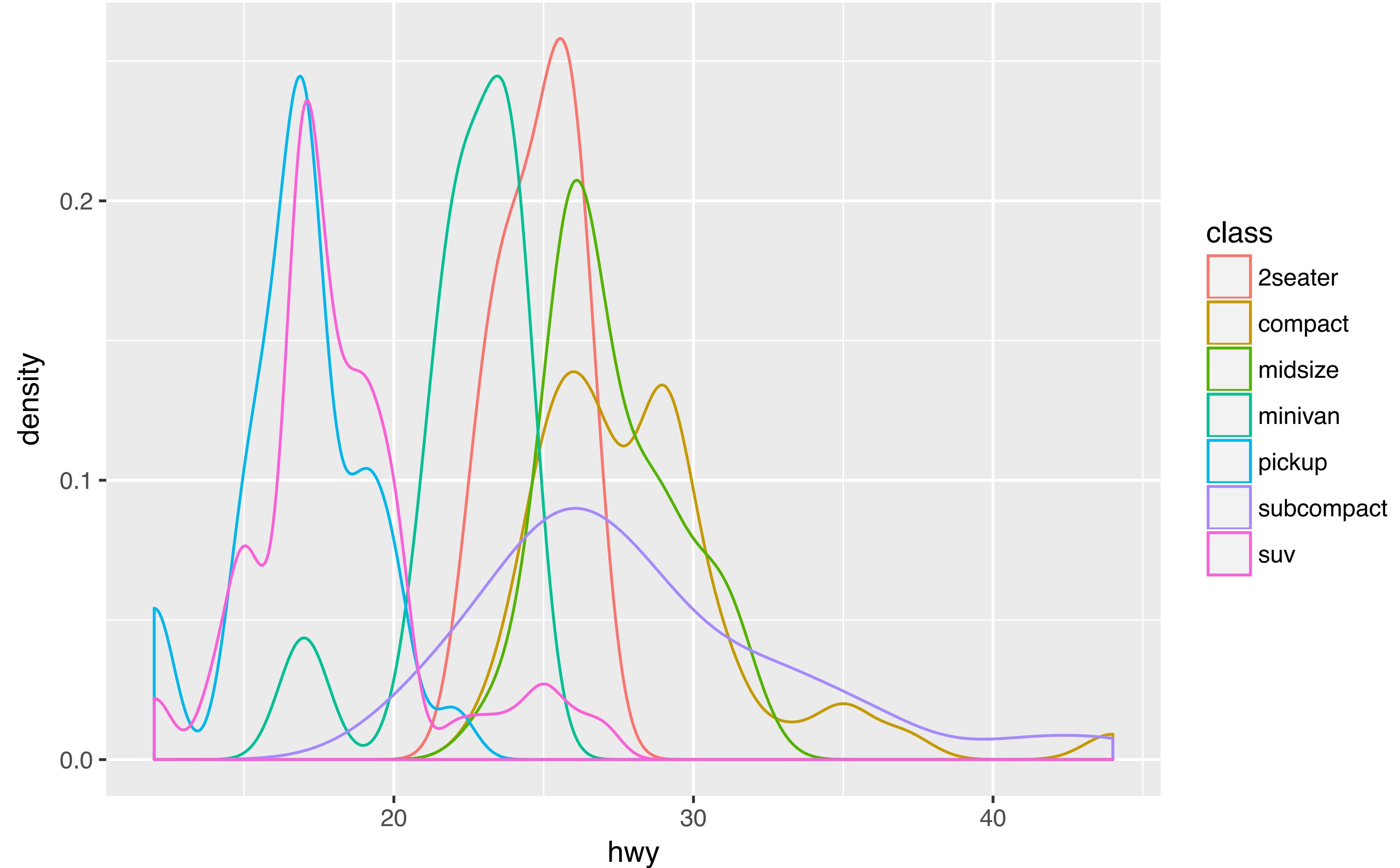
Your Turn 5

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

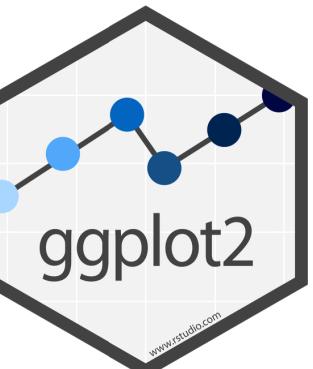


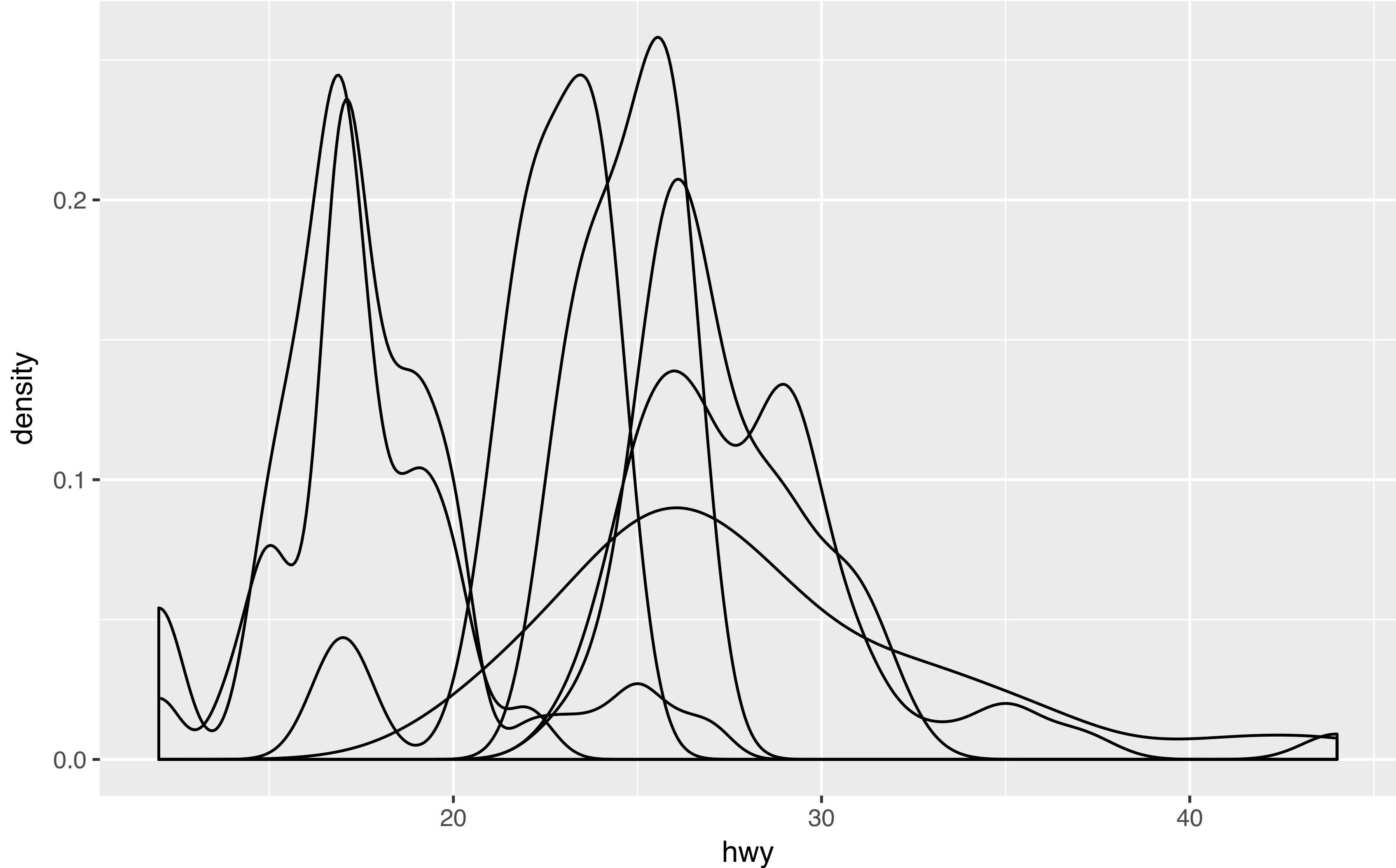


```
ggplot(data = mpg) +  
  geom_density(mapping = aes(x = hwy))
```

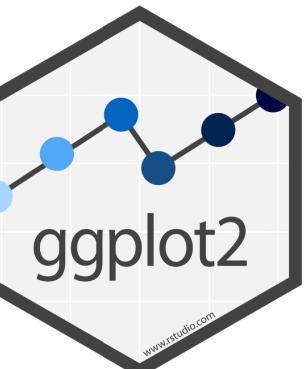


```
ggplot(data = mpg) +  
  geom_density(mapping = aes(x = hwy, color = class))
```



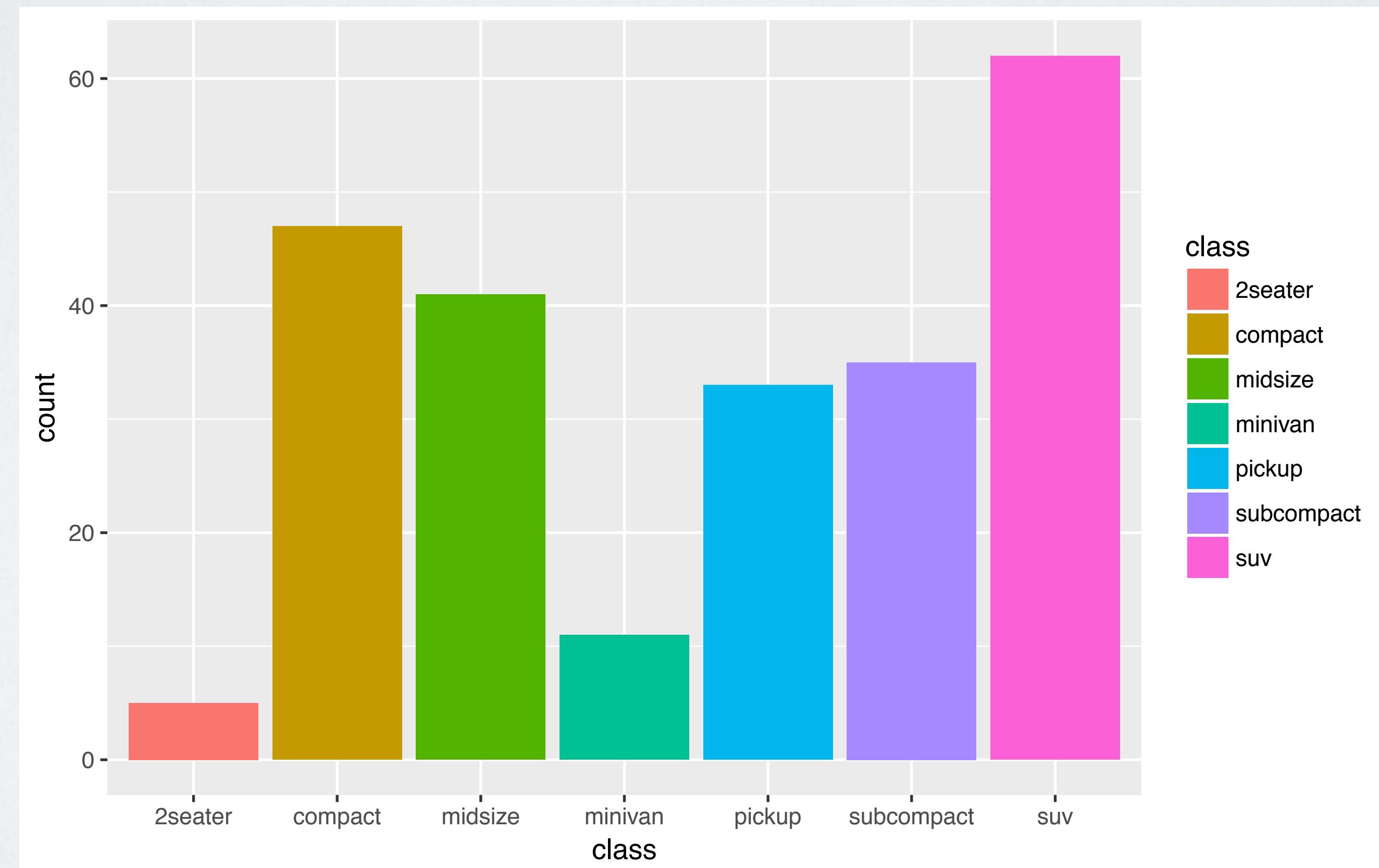


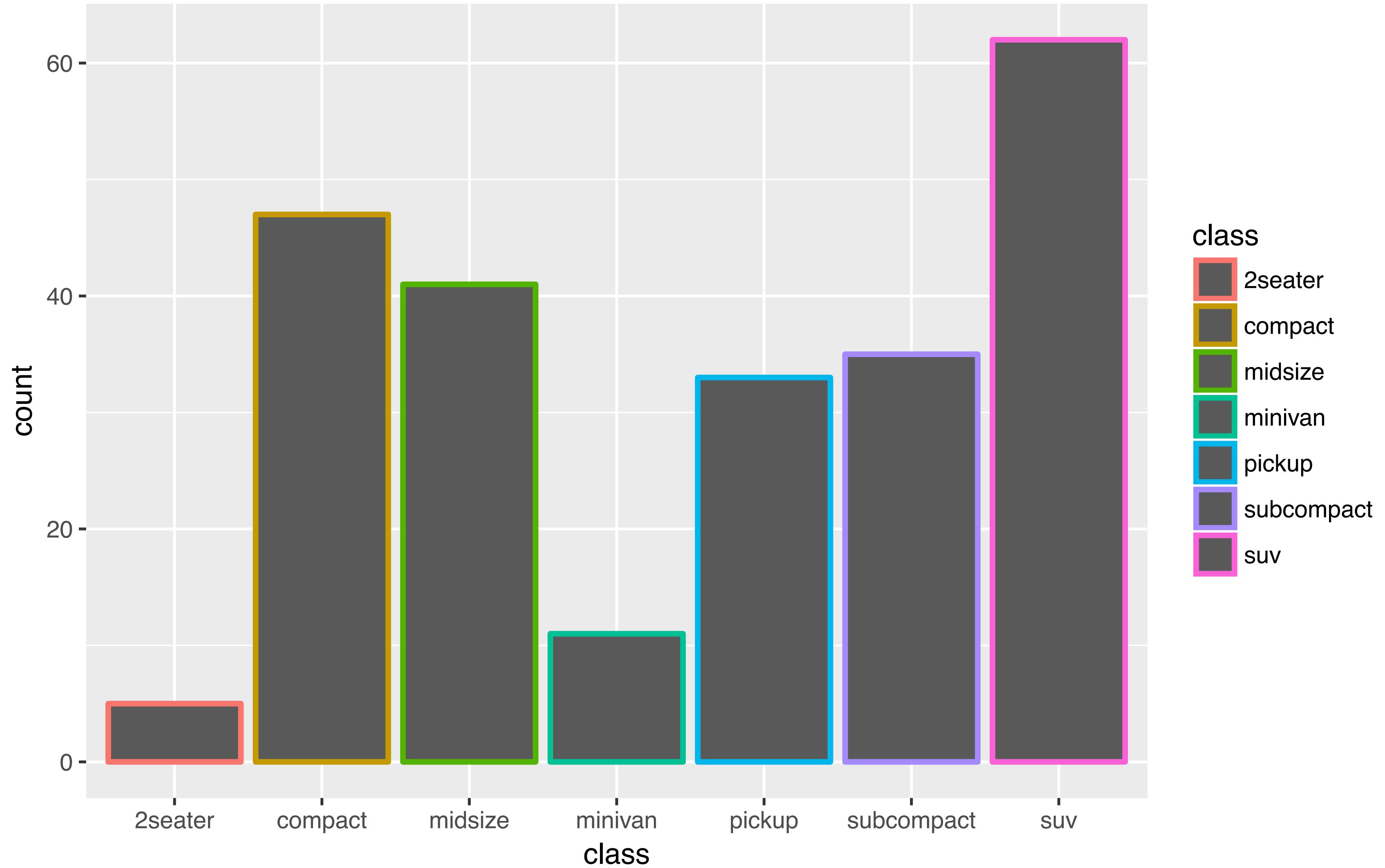
```
ggplot(data = mpg) +  
  geom_density(mapping = aes(x = hwy, group = class))
```



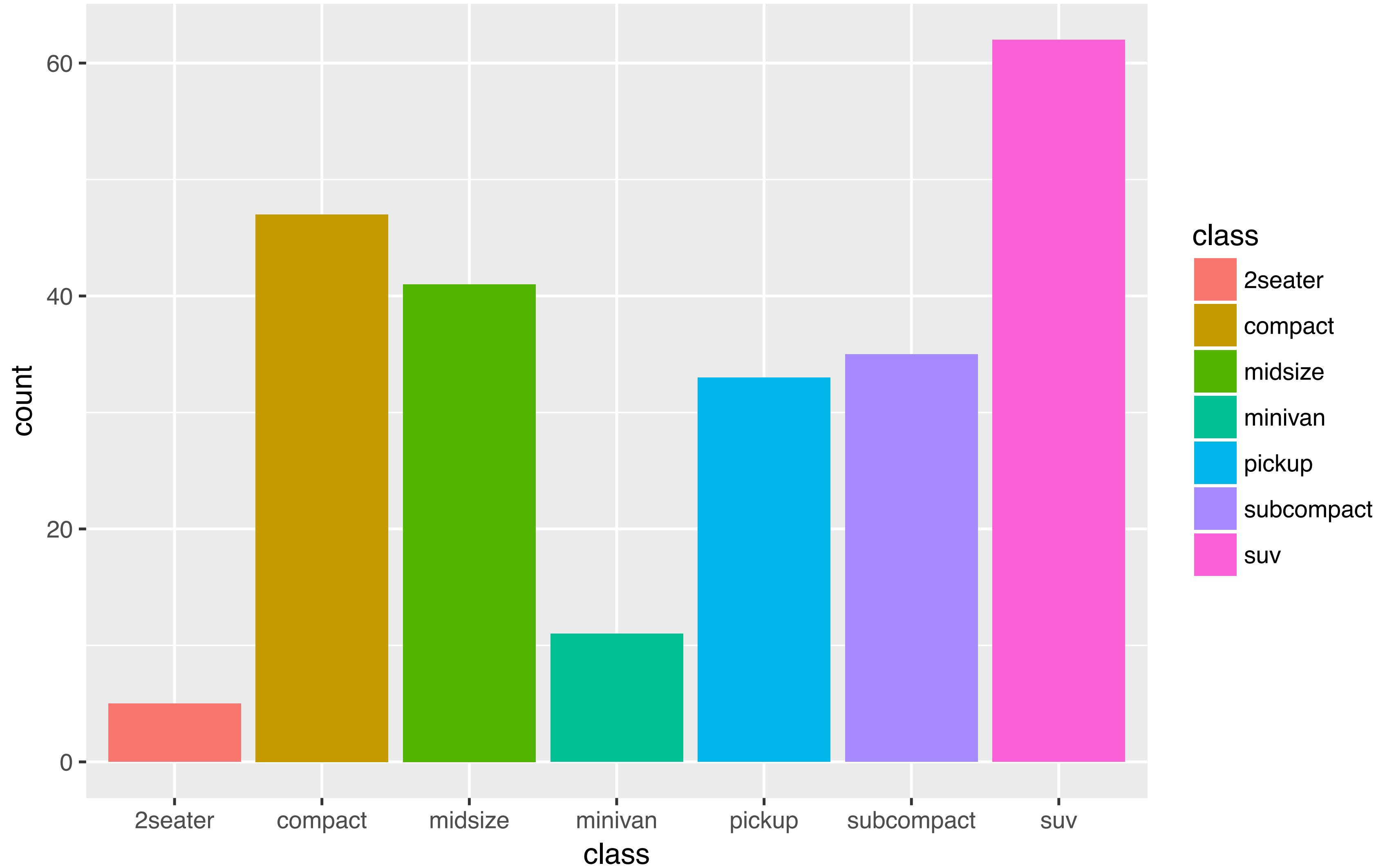
Your Turn 6

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

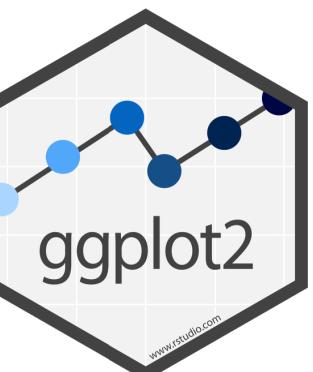




```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = class, color = class))
```



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



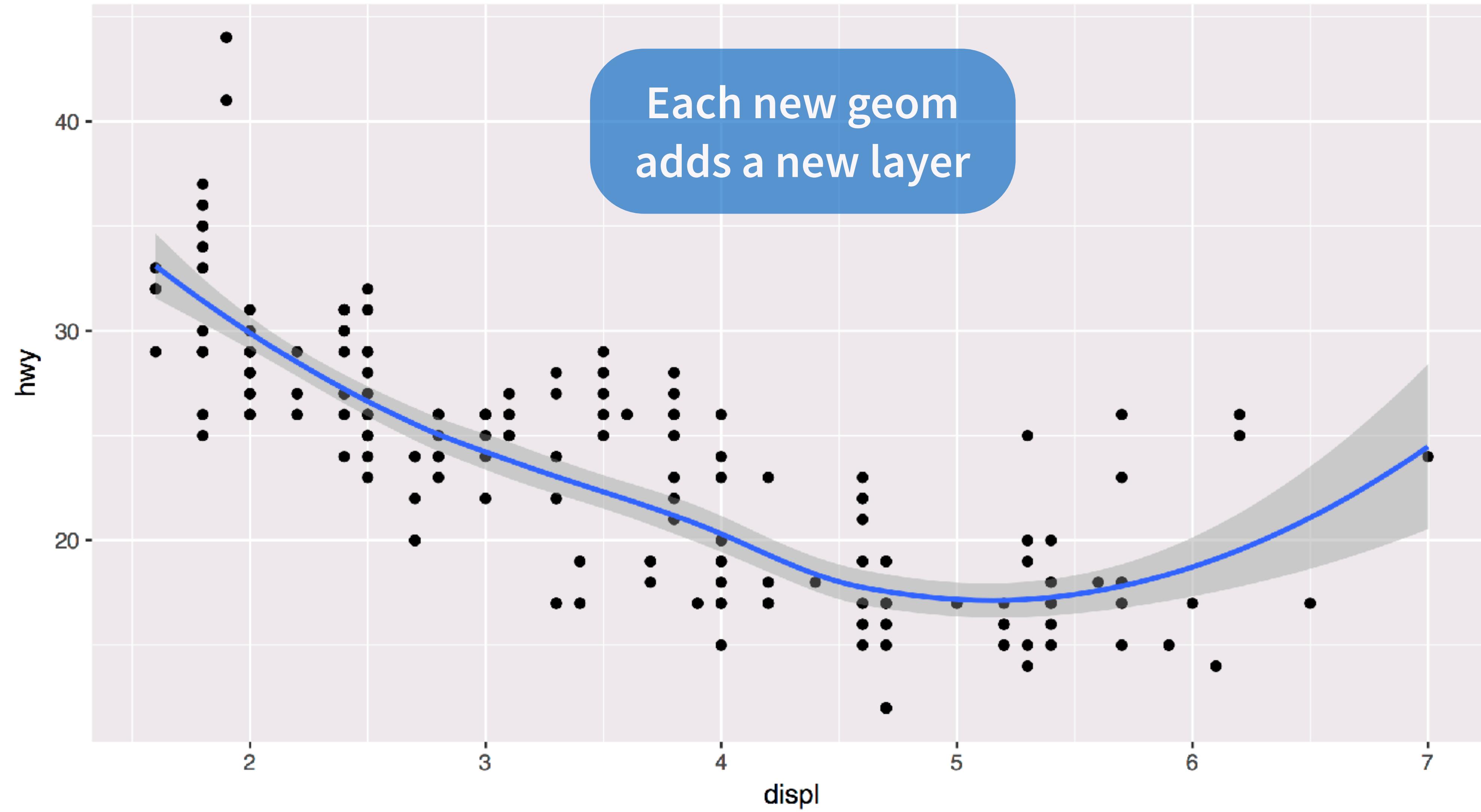
Your Turn 7

With your partner, predict what this code will do.

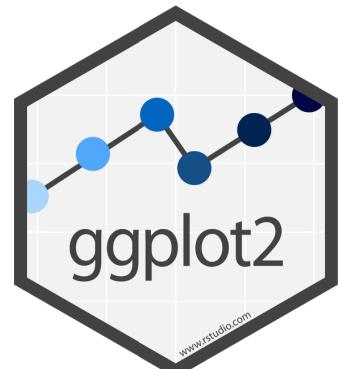
Then run it.

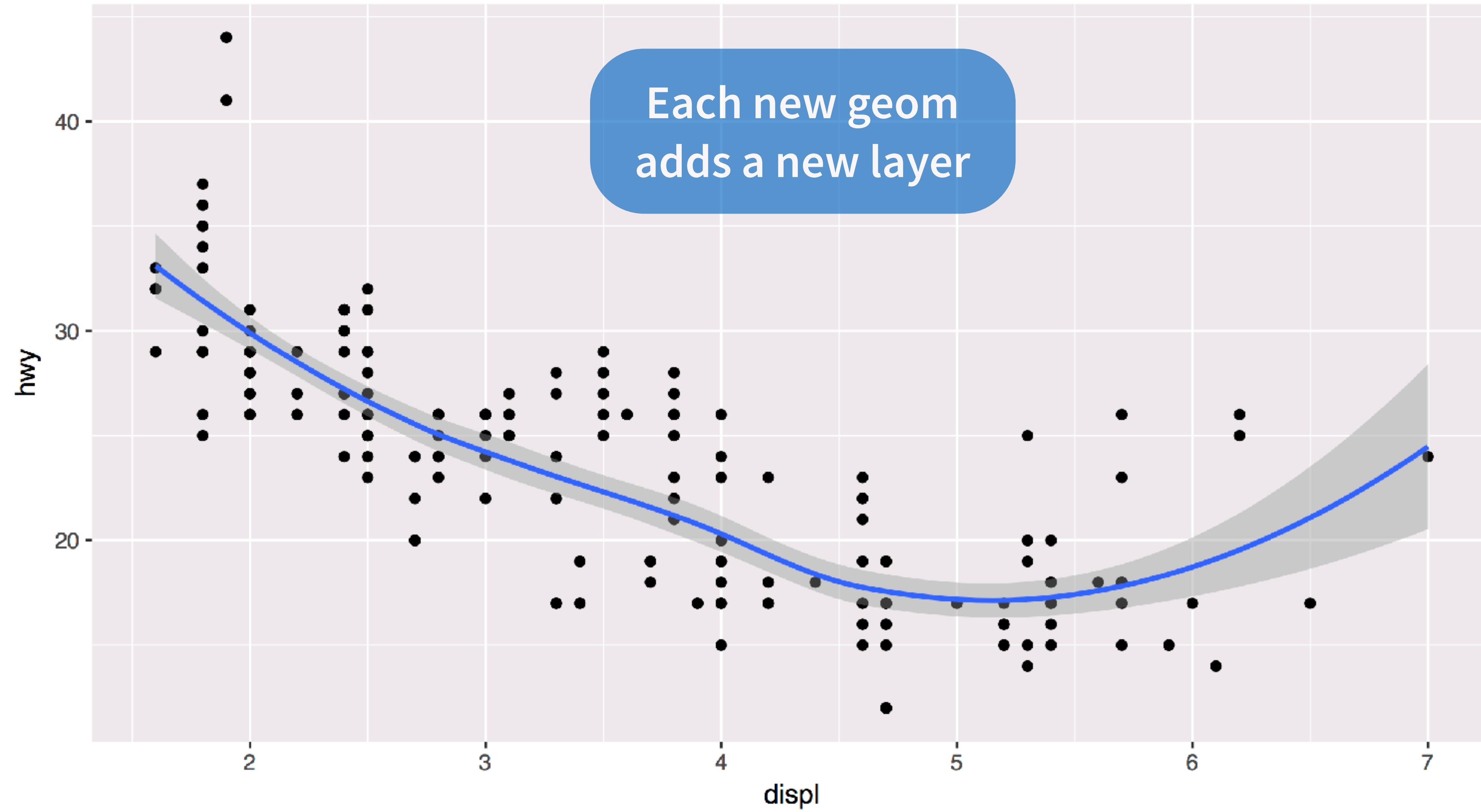
```
ggplot(mpg) +  
  geom_point(aes(displ, hwy)) +  
  geom_smooth(aes(displ, hwy))
```



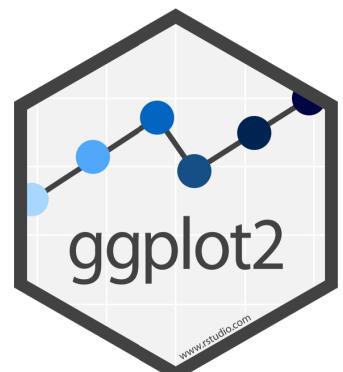


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



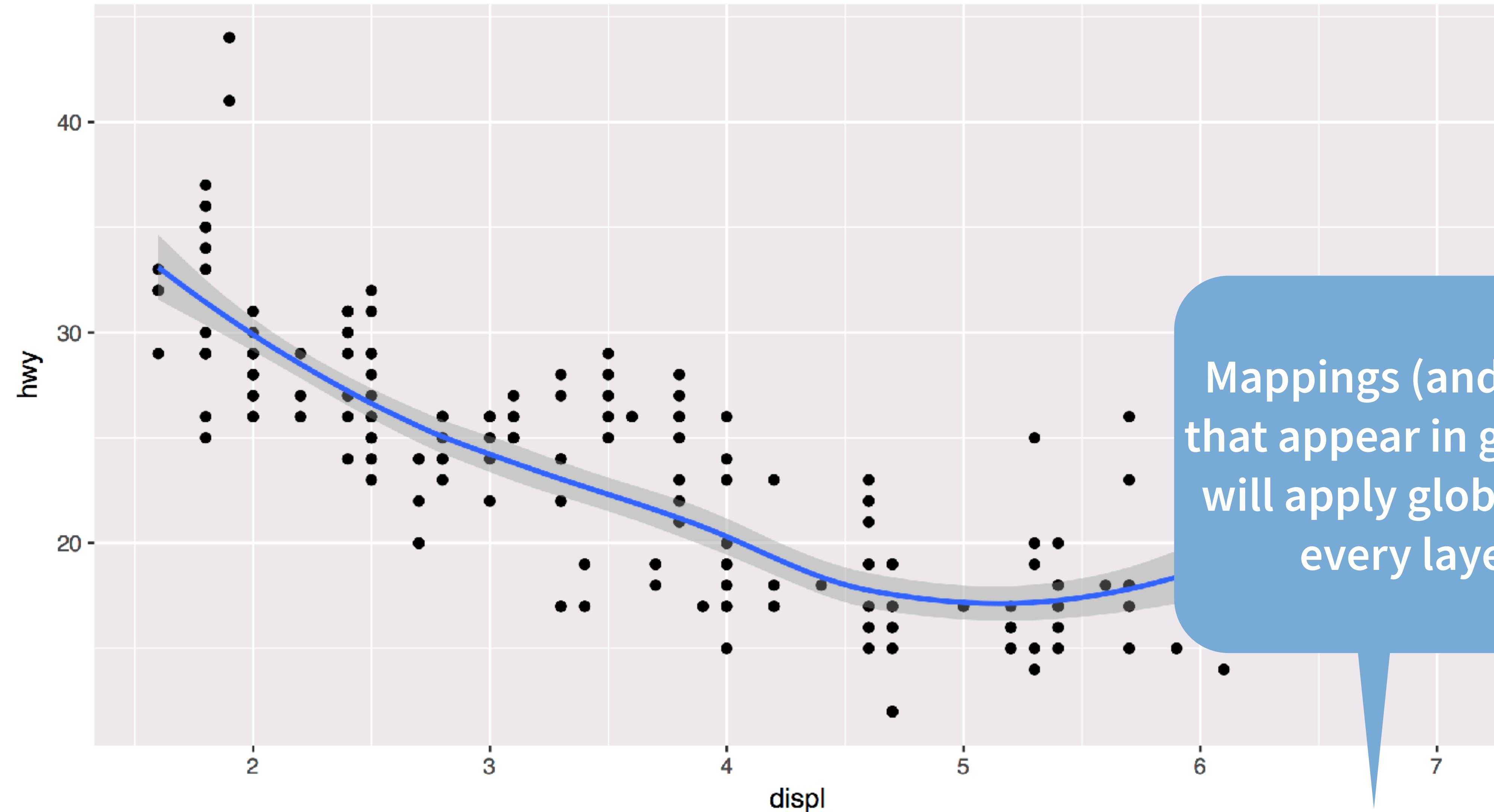


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

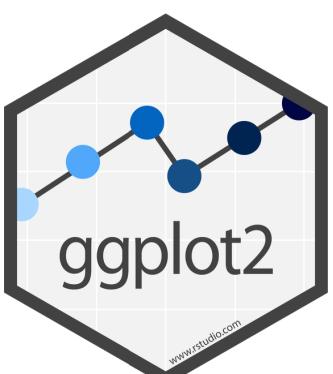


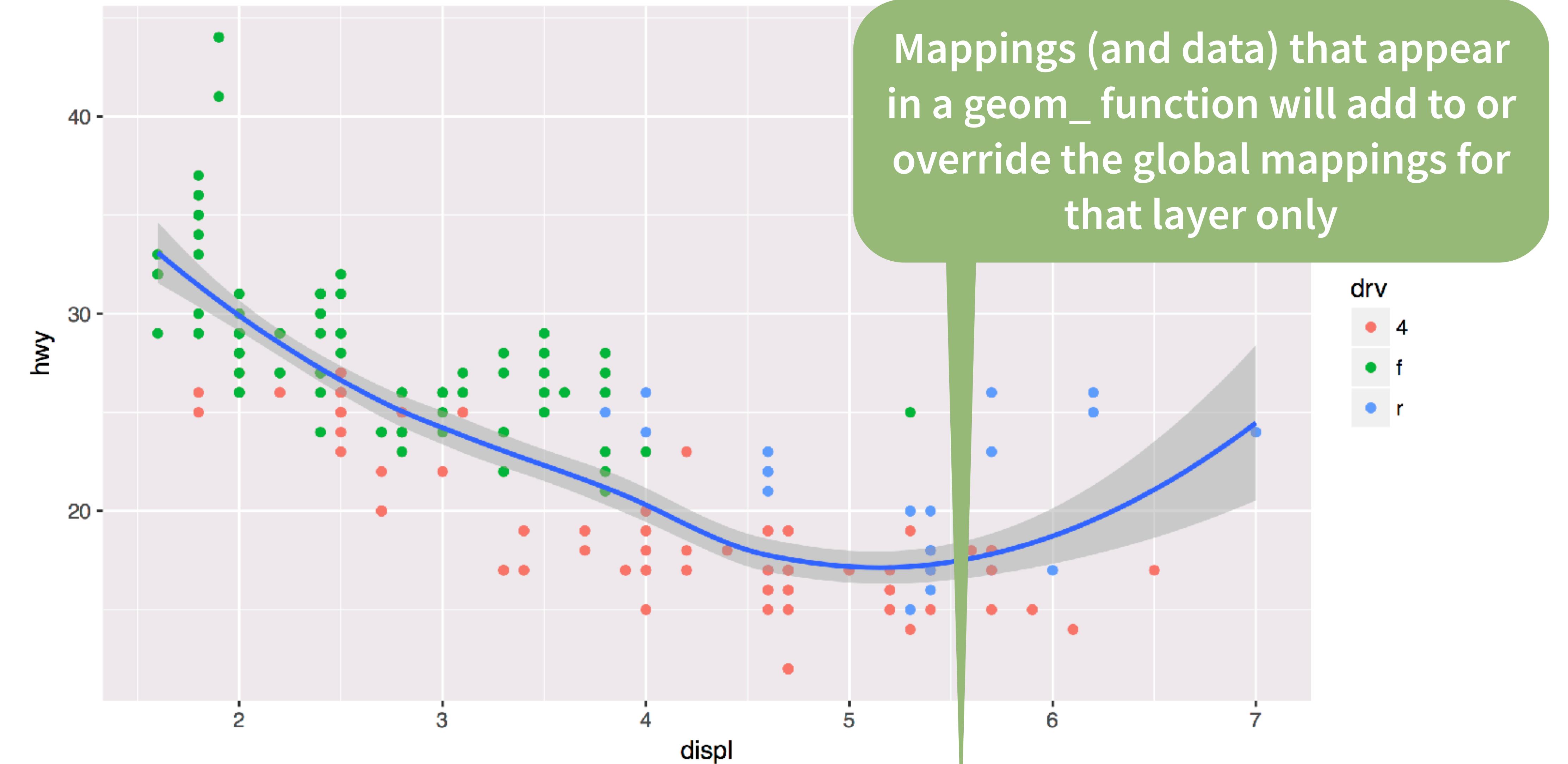
global vs. local

R

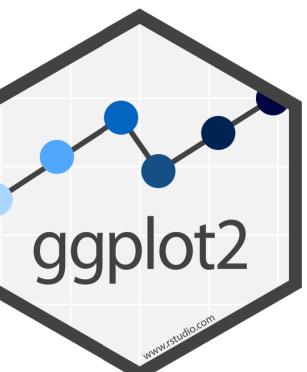


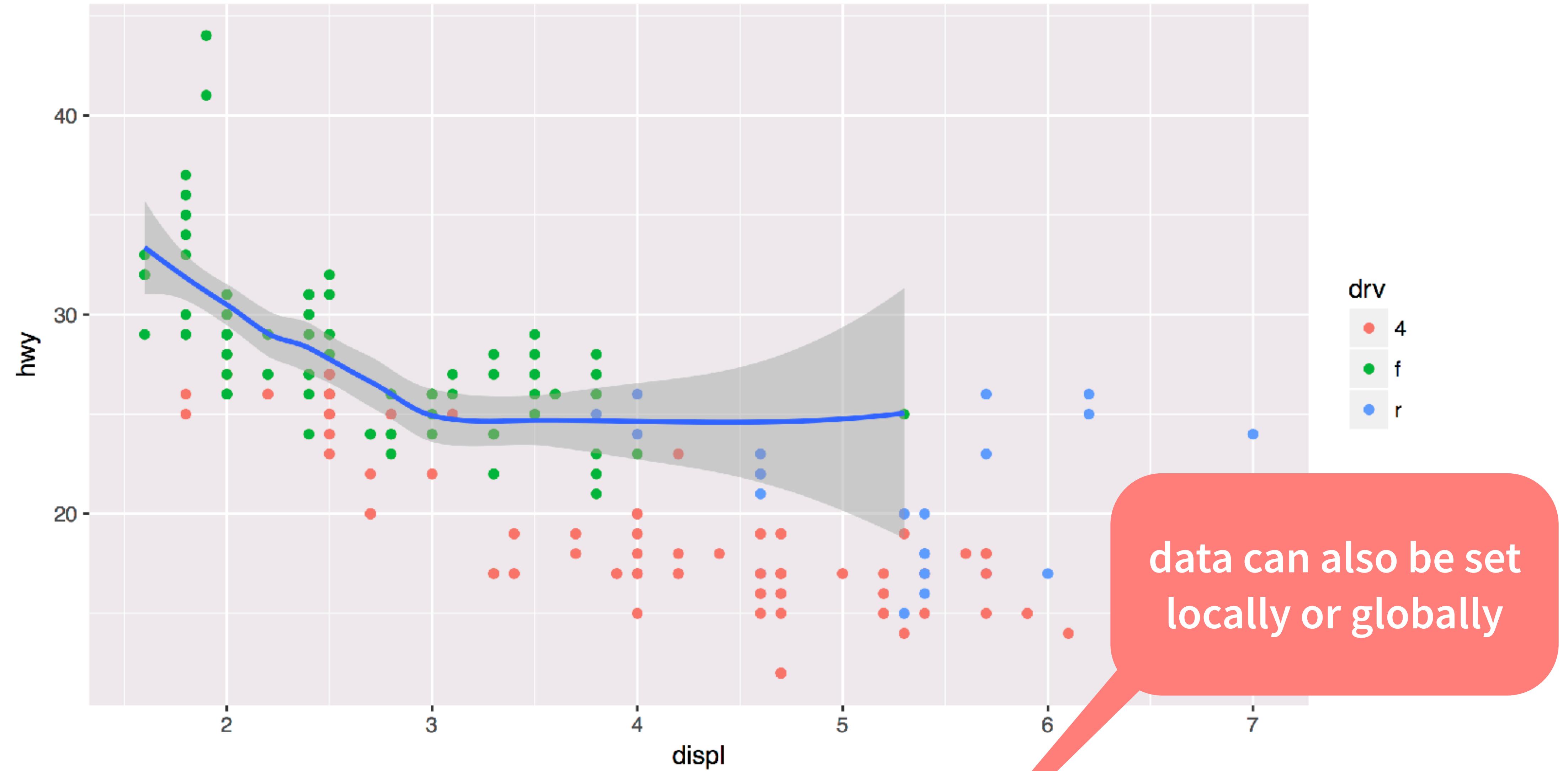
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth()
```





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





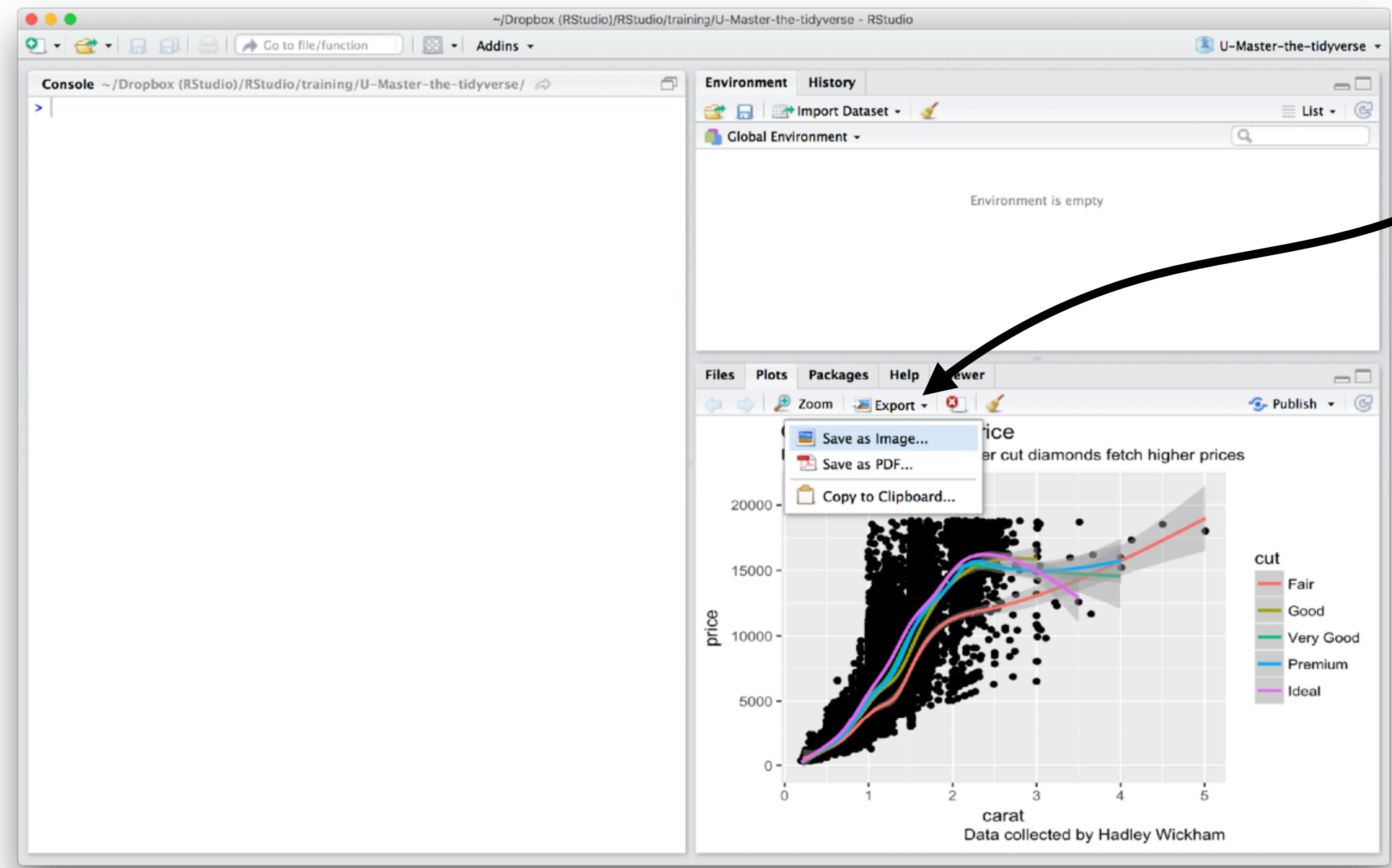
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth(data = filter(mpg, drv == "f"))
```

Saving graphs



Manually saving plots

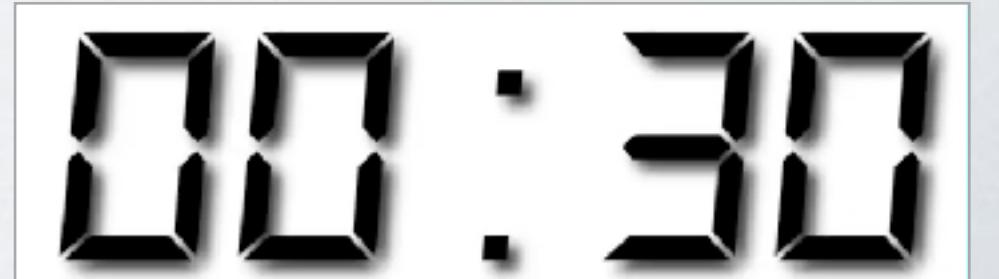
Save plots manually with the export menu



Your Turn 8

What does this command return?

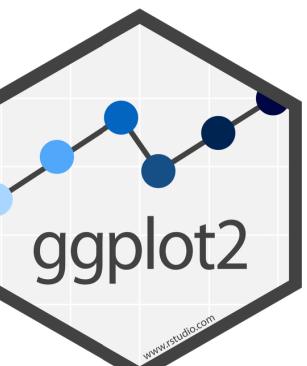
`getwd()`



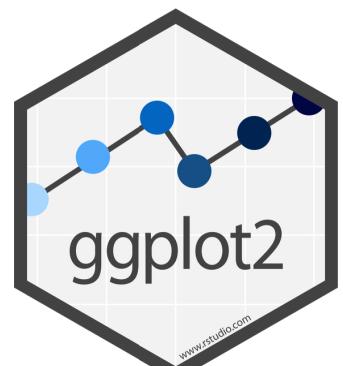
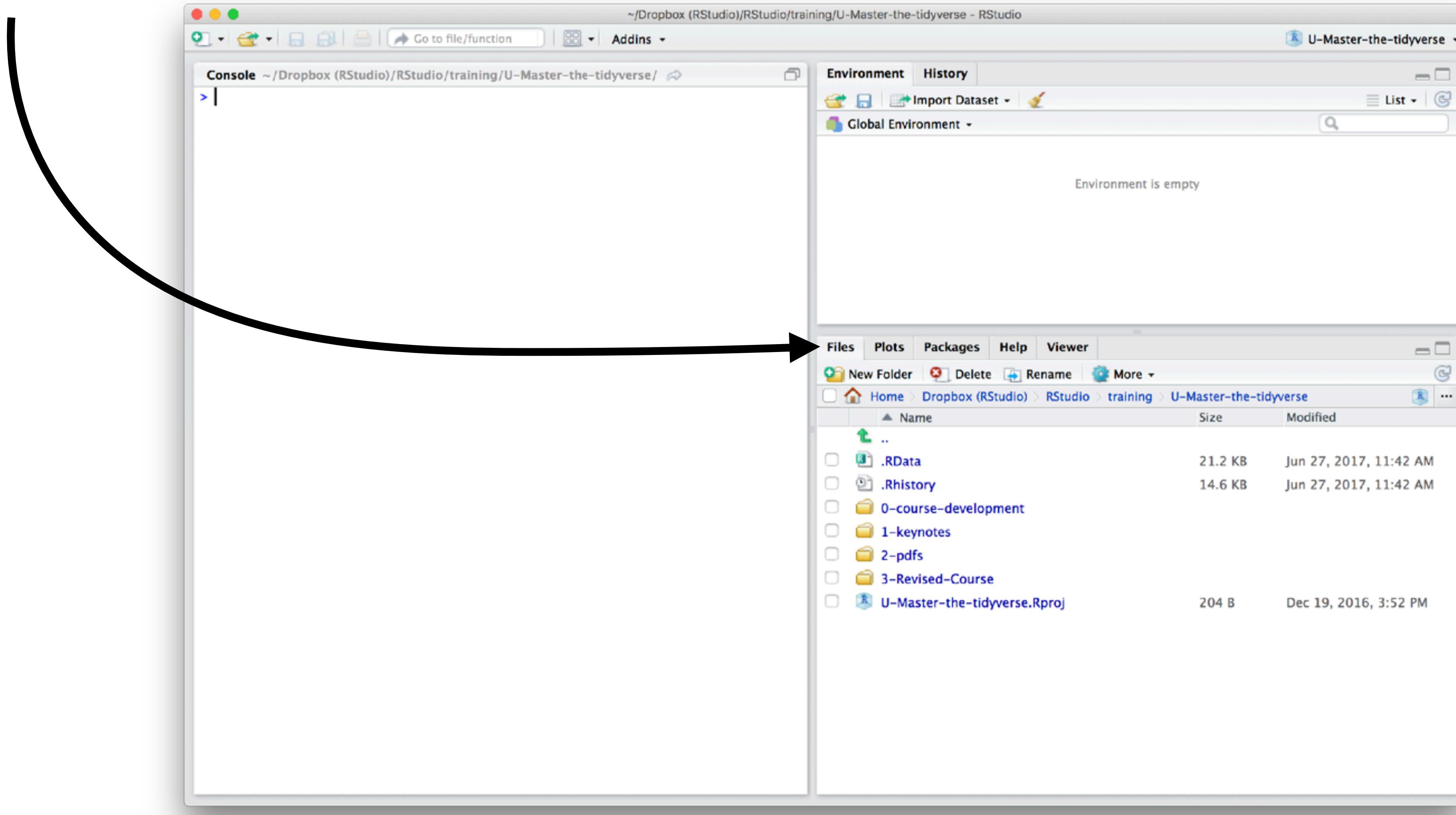
Working directory

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

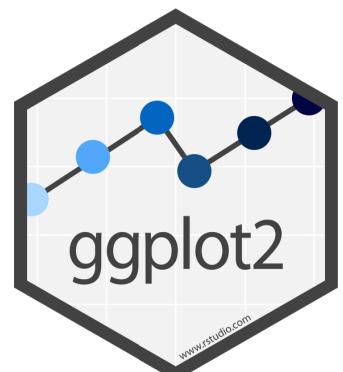
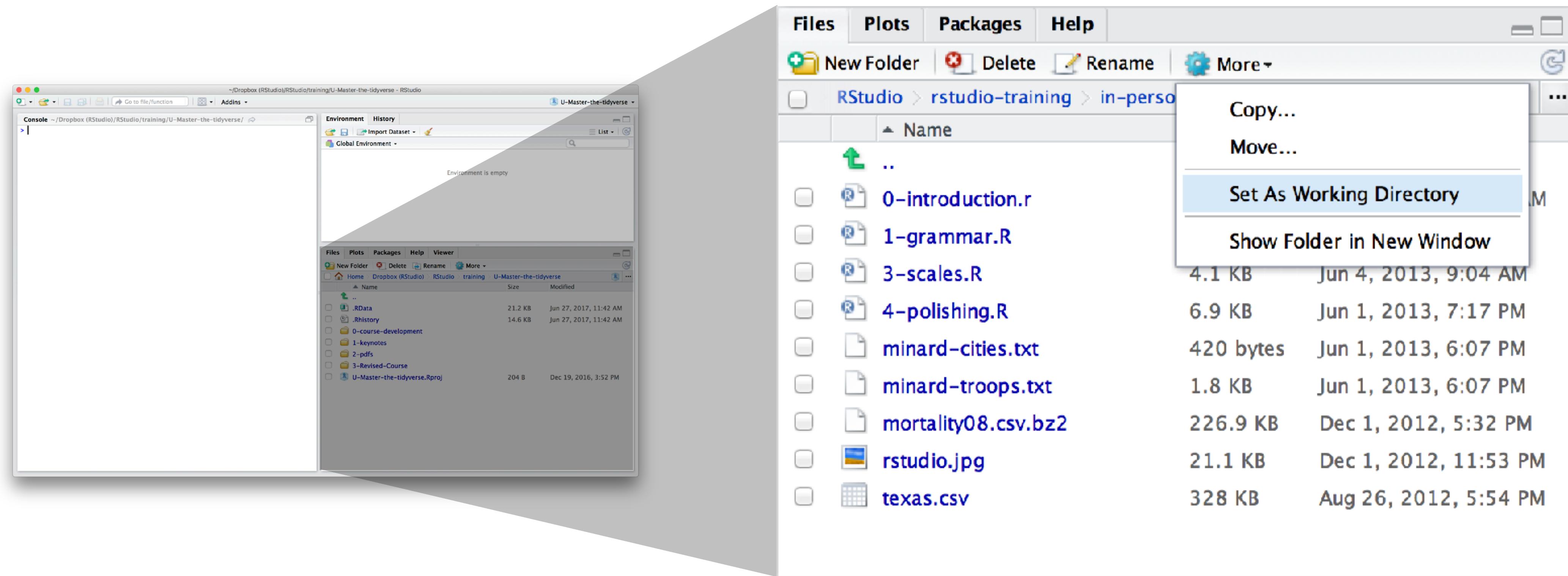


The files pane of the IDE displays the contents of your working directory



Changing the Working directory

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



Saving plots

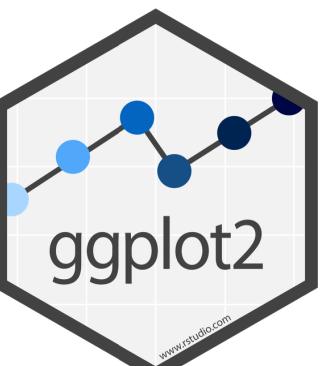
ggsave() saves the last plot.

Uses size on screen:

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

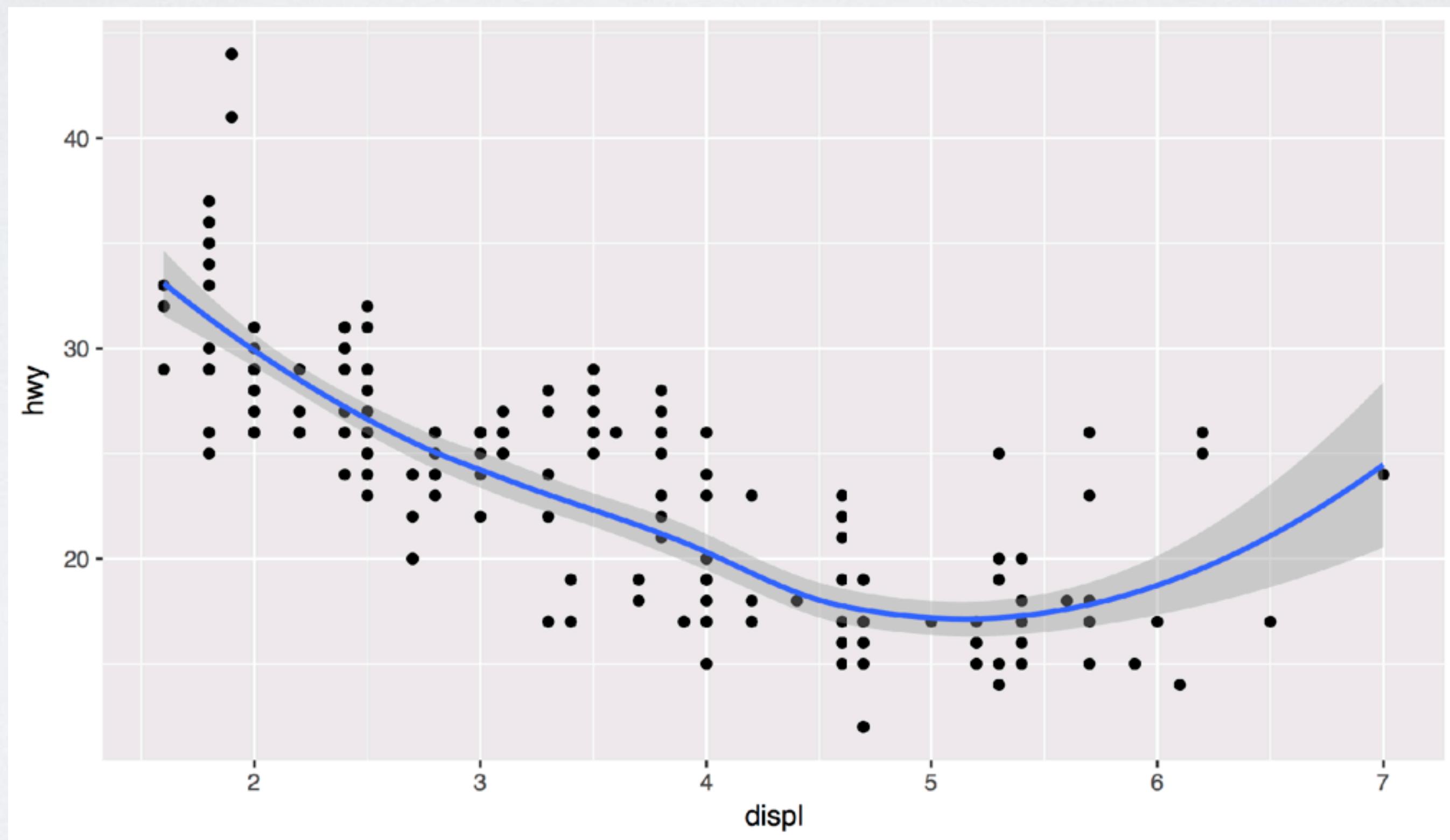
Specify size in inches

```
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).



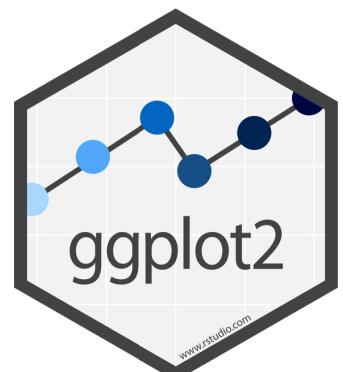
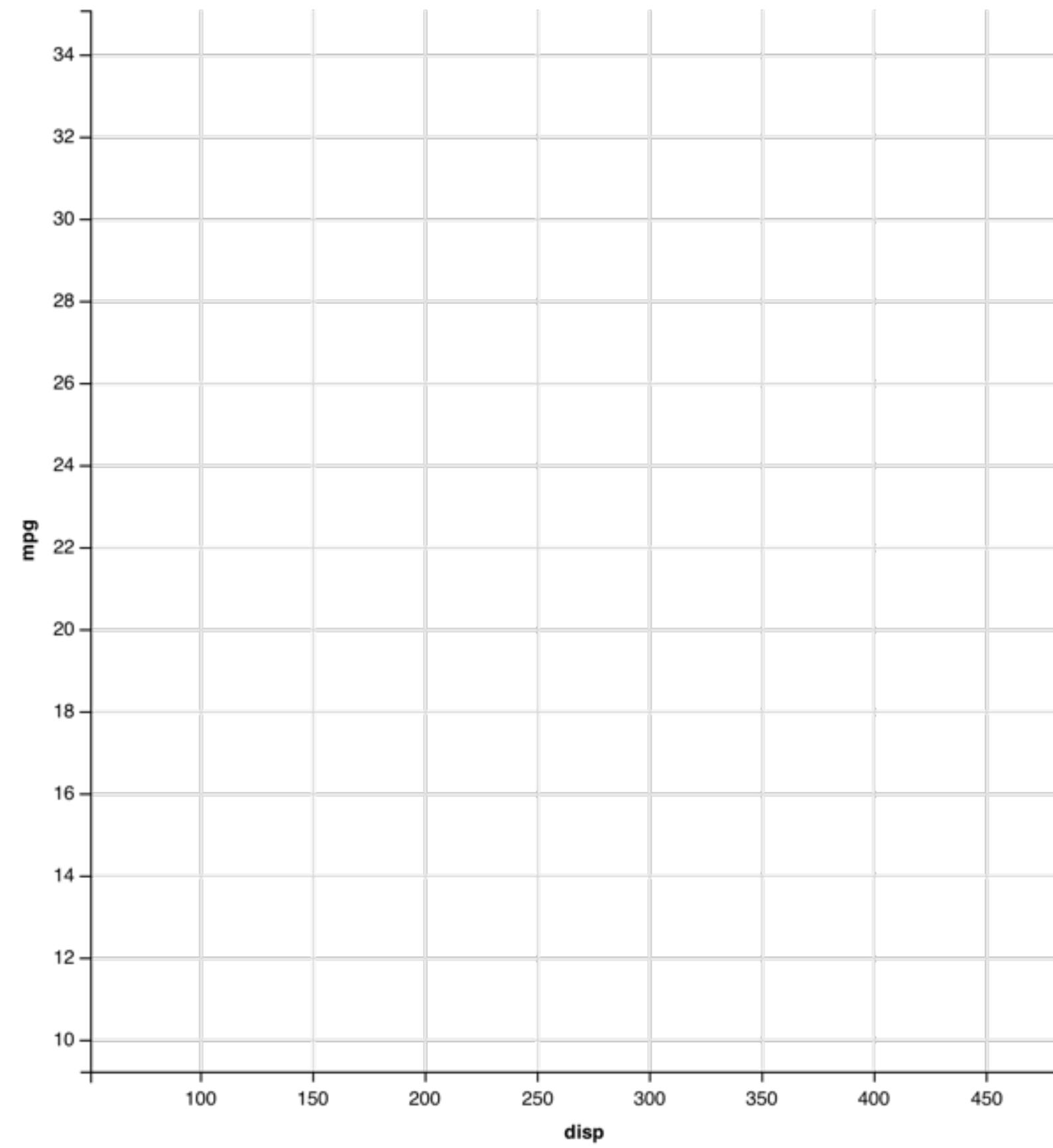
Grammar of Graphics



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

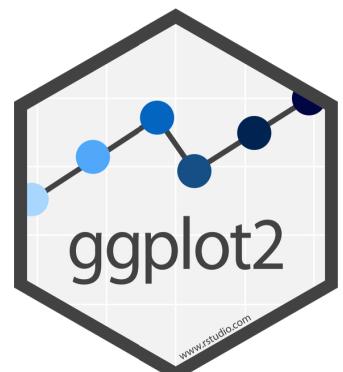
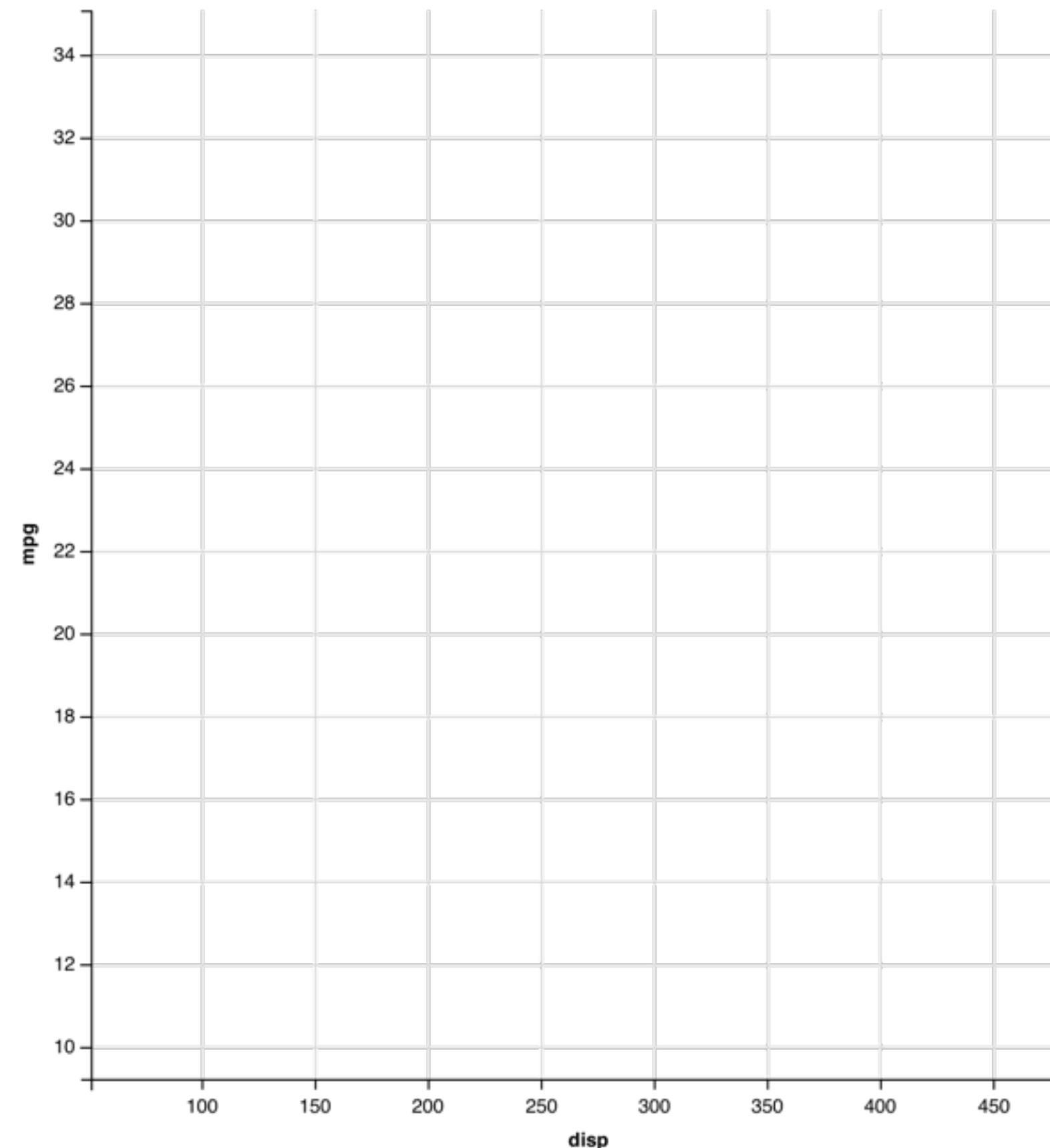
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

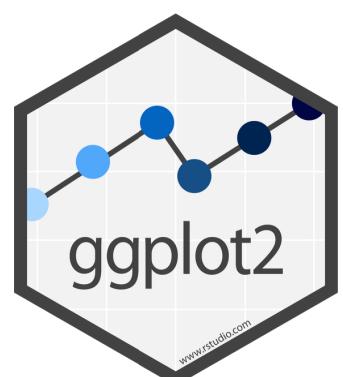
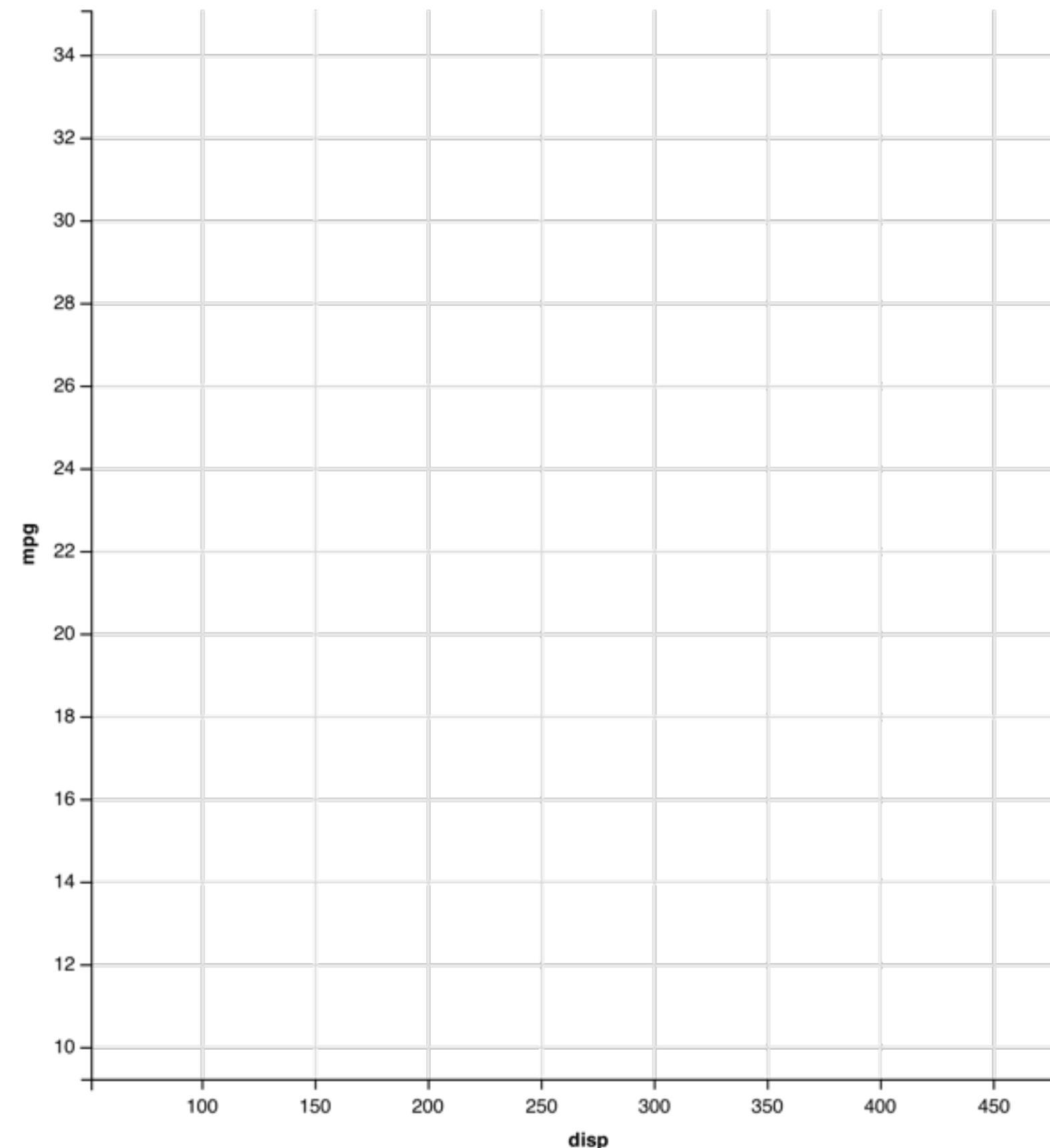


mappings

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

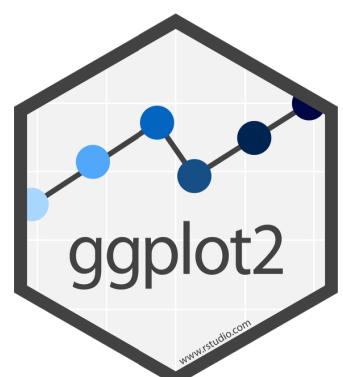
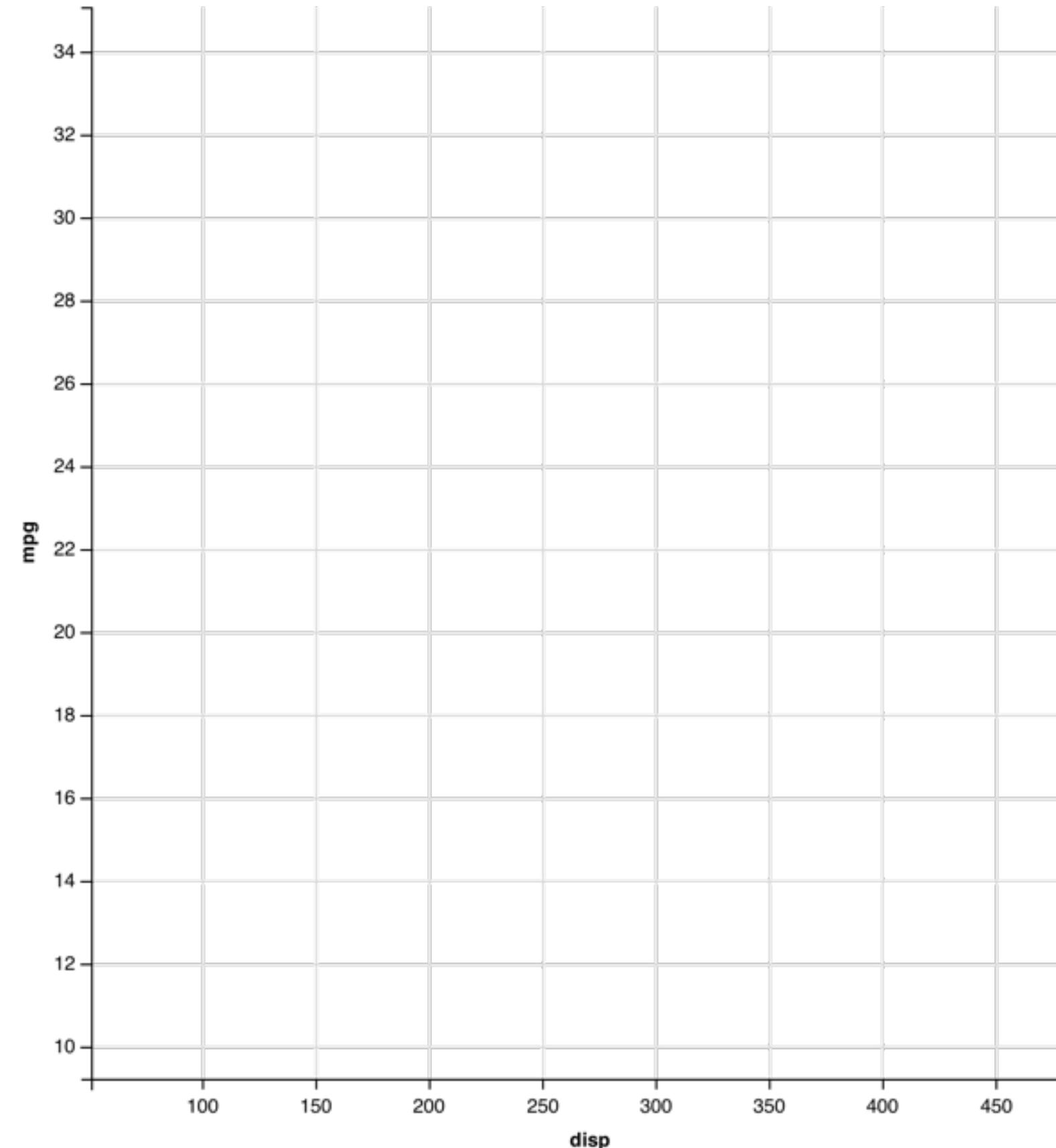


mappings

	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

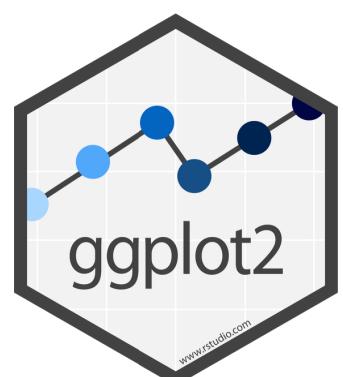
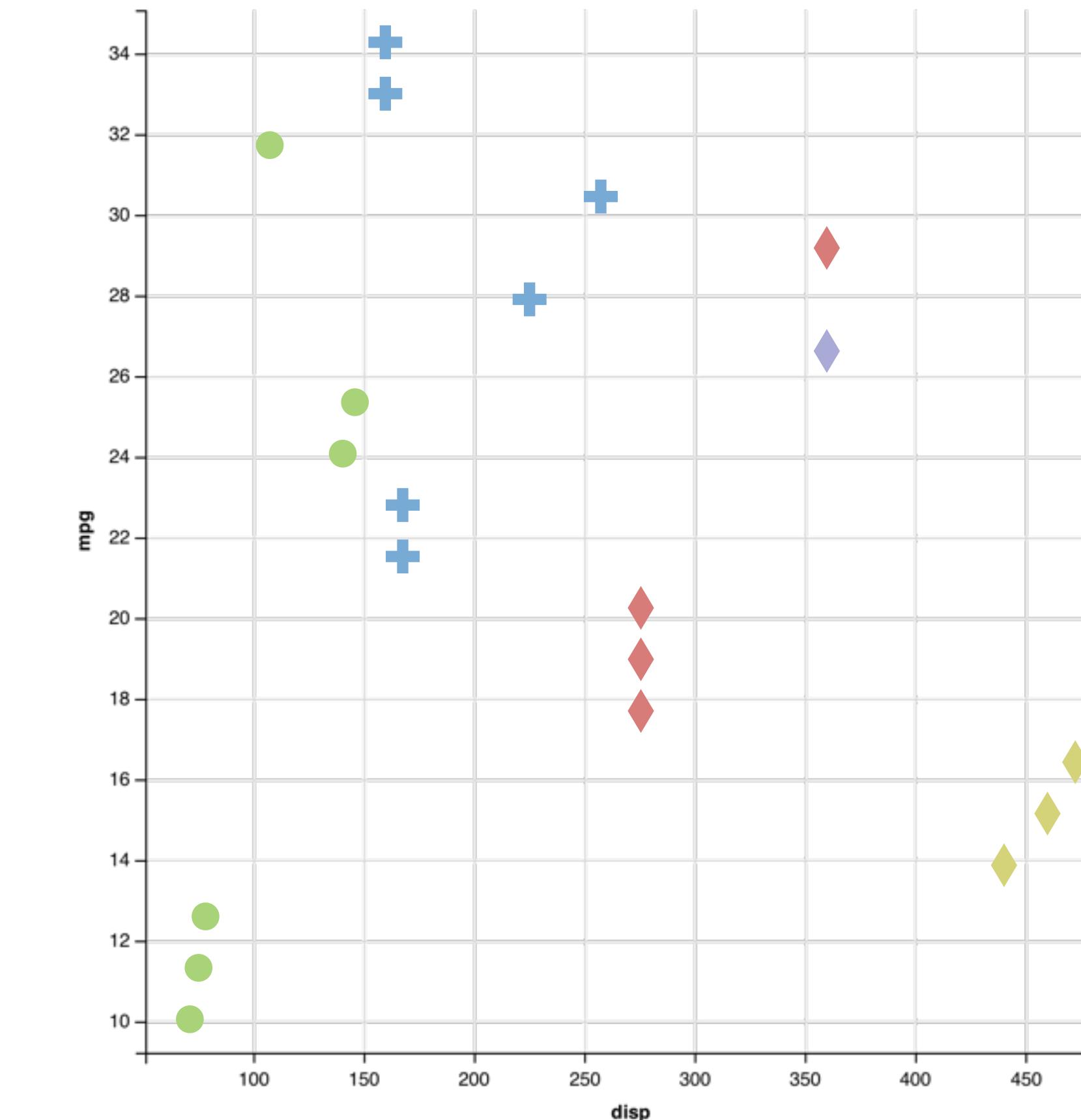


mappings

	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

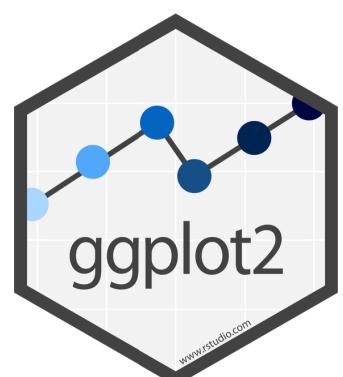
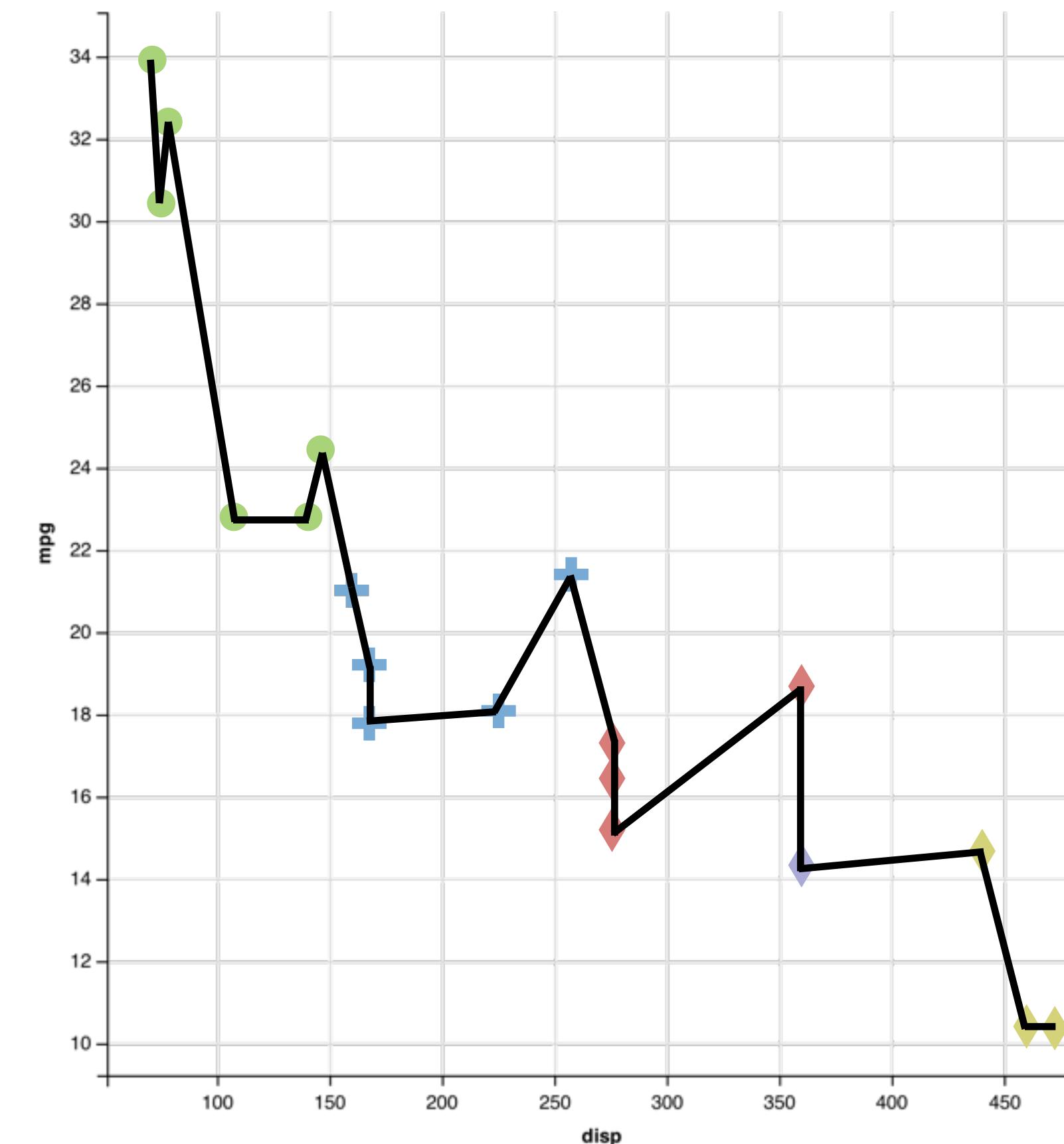


mappings

	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

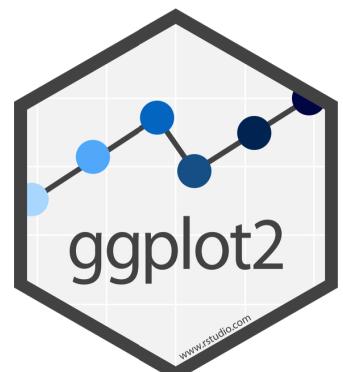
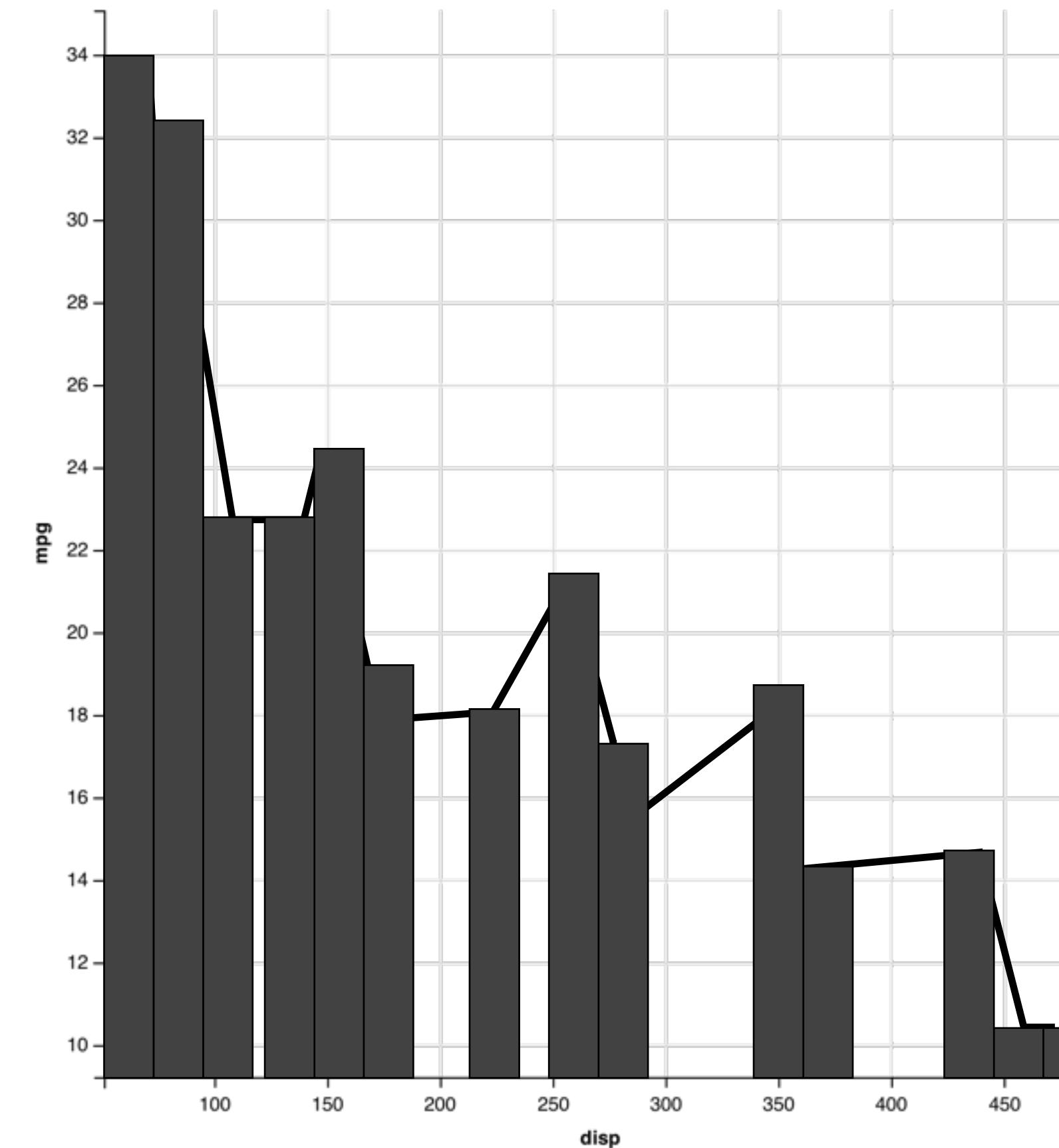


mappings

	y	x		
	mpg	cyl	disp	hp
1	21.0	6	160.0	2
2	21.0	6	160.0	2
3	22.8	4	108.0	1
4	21.4	6	258.0	2
5	18.7	8	360.0	3
6	18.1	6	225.0	2
7	14.3	8	360.0	5
8	24.4	4	146.7	1
9	22.8	4	140.8	1
10	19.2	6	167.6	2
11	17.8	6	167.6	2
12	16.4	8	275.8	3
13	17.3	8	275.8	3
14	15.2	8	275.8	3
15	10.4	8	472.0	4
16	10.4	8	460.0	4
17	14.7	8	440.0	4
18	32.4	4	78.7	1
19	30.4	4	75.7	1
20	33.9	4	71.1	1

data

geom
points
lines
bars

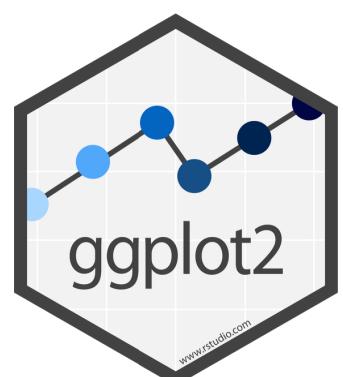
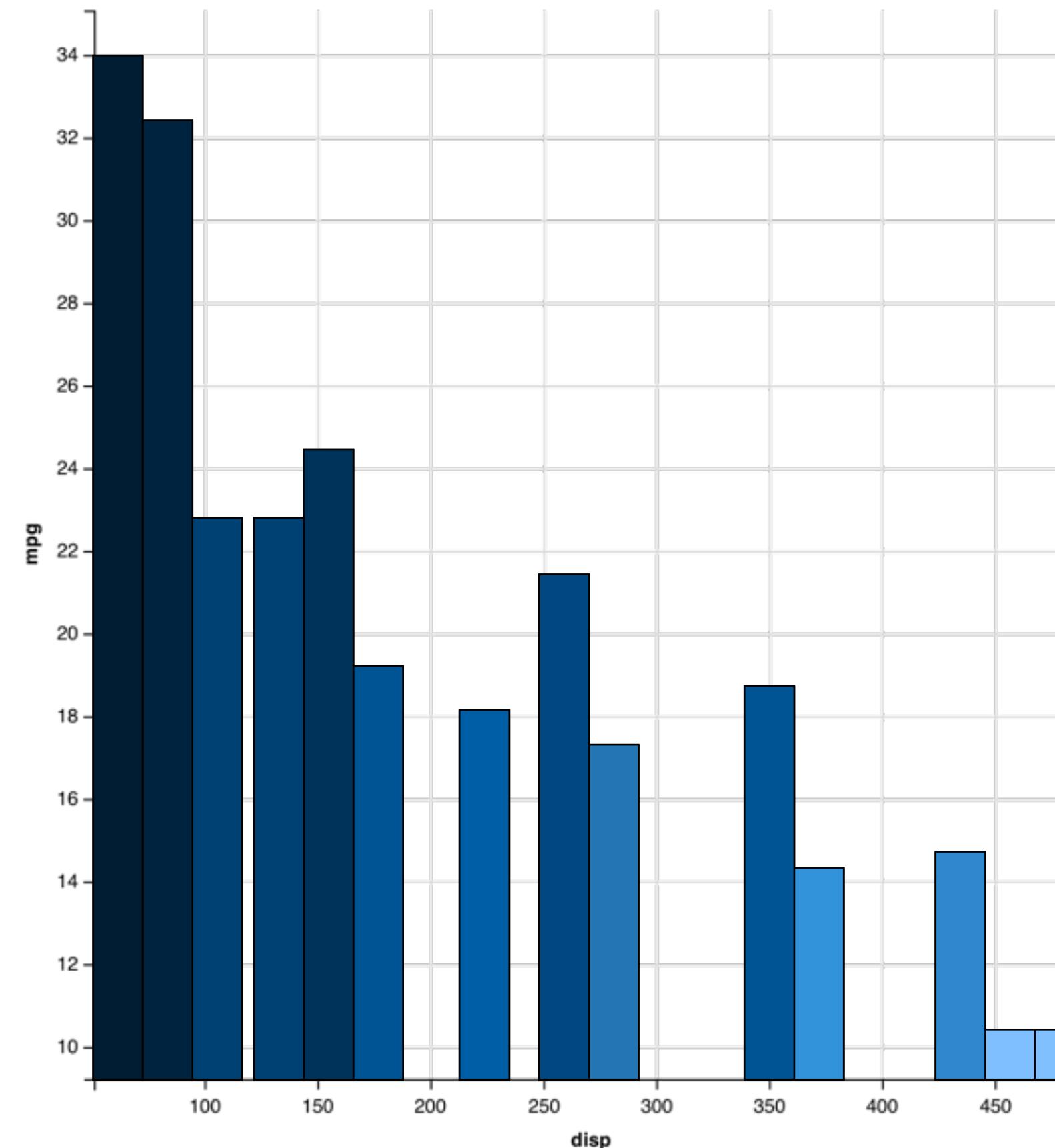


mappings

	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

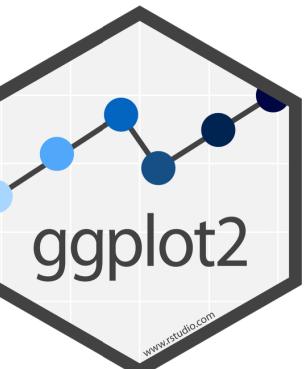
geom
points
lines
bars



To make a graph

[template]

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



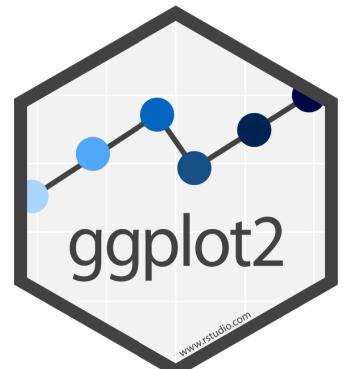
To make a graph

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	●

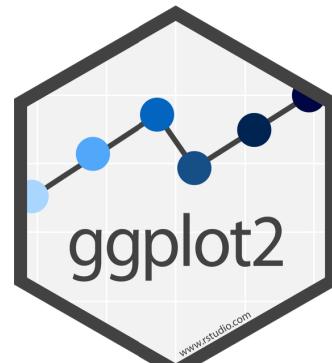
data

geom

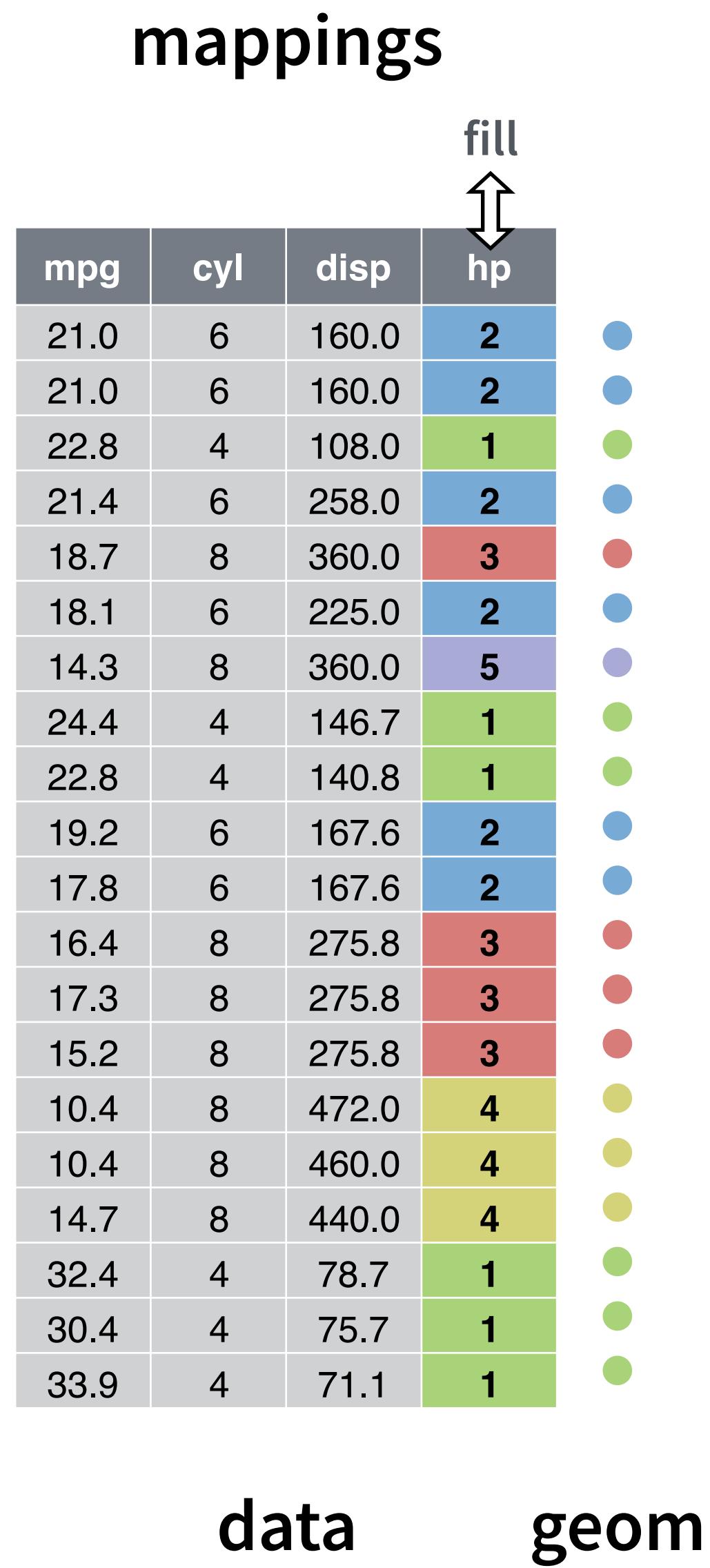
1. Pick a **data** set

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

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



To make a graph

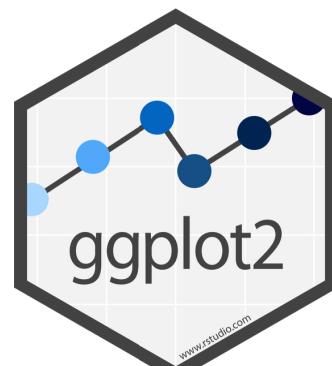


1. Pick a **data** set

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

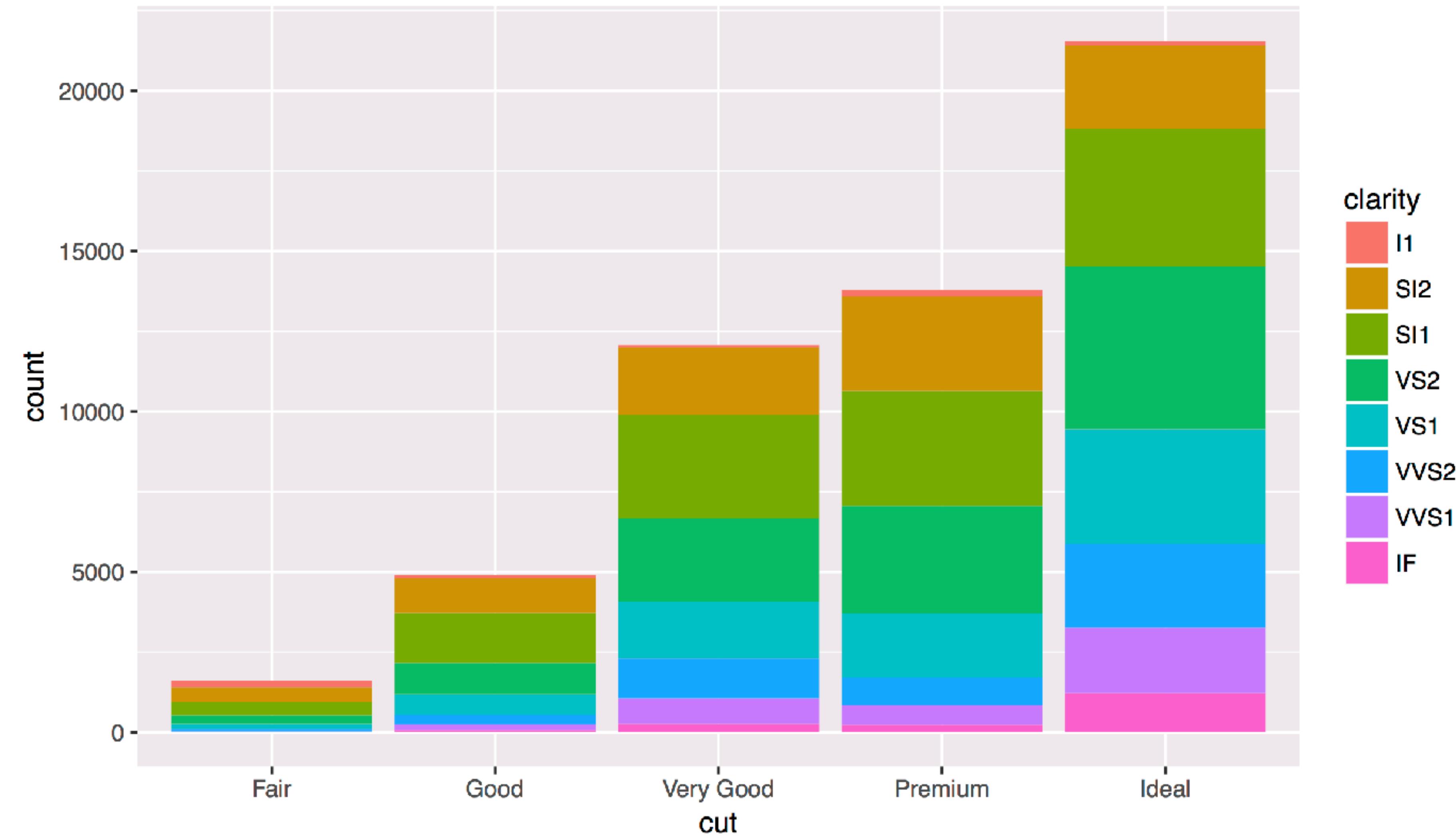
2. Choose a **geom**
to display cases

3. **Map** aesthetic
properties to
variables



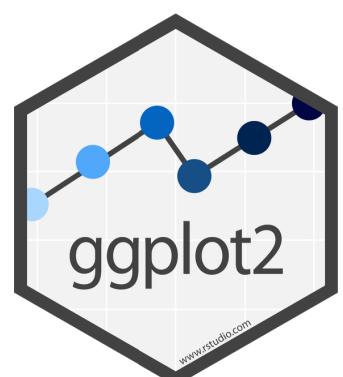
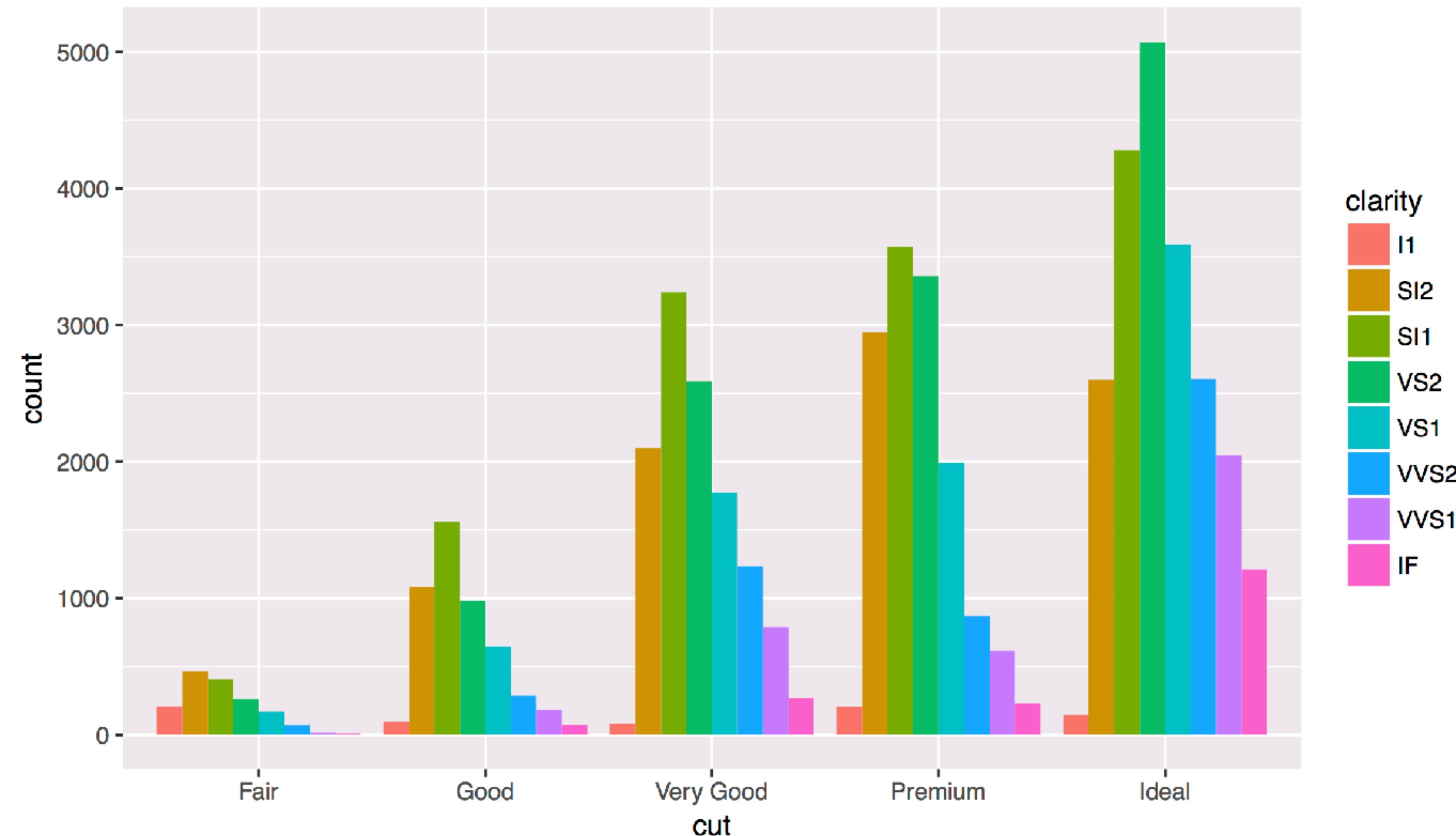
what else?

R



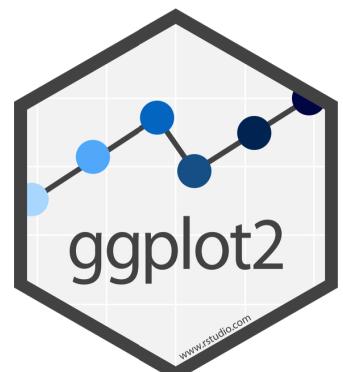
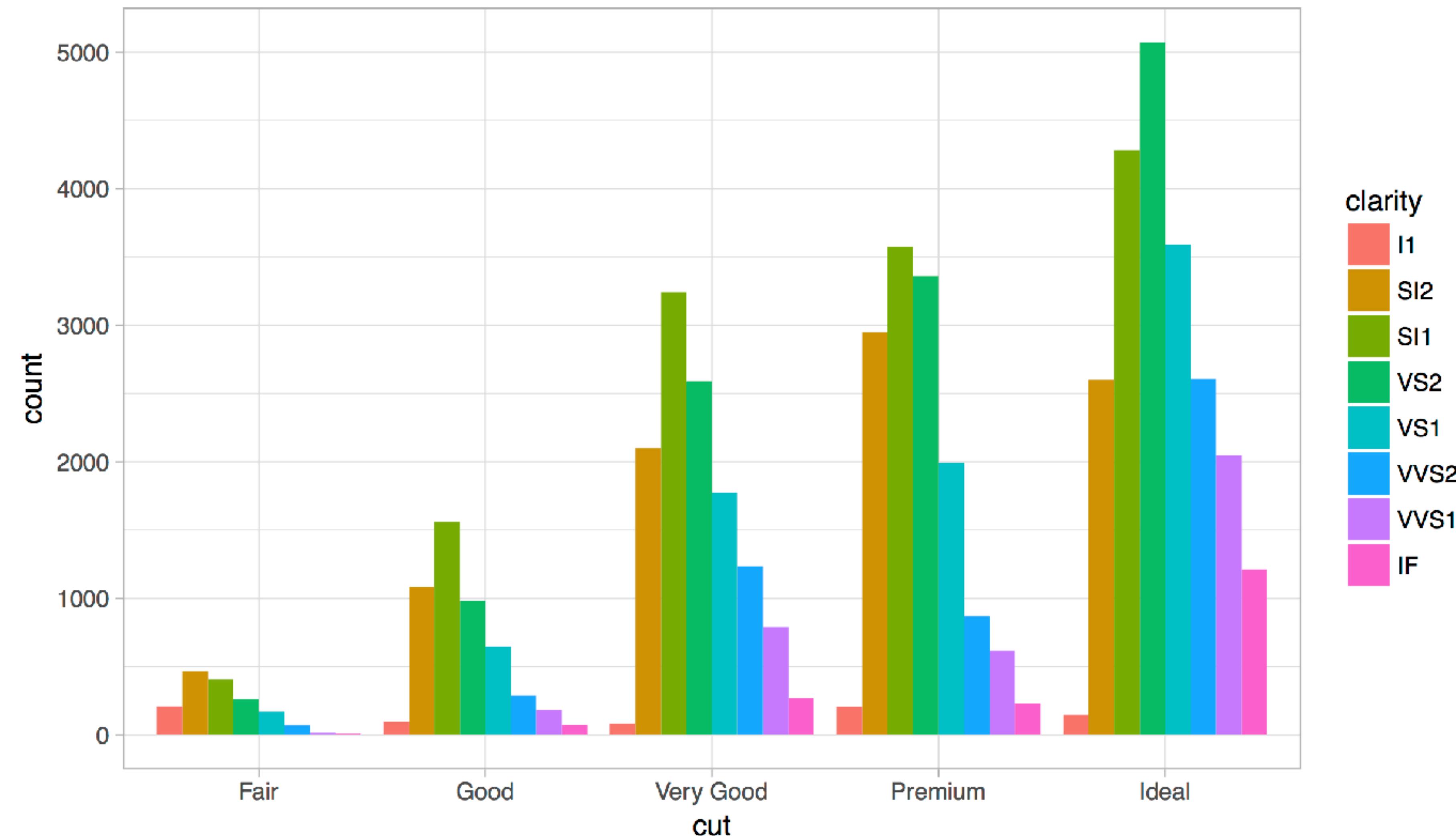
Position Adjustments

How overlapping objects are arranged



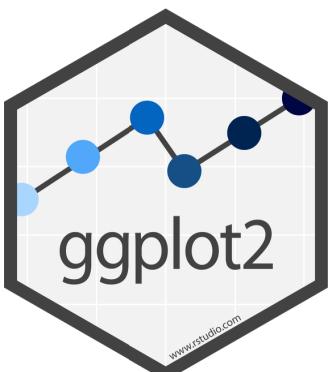
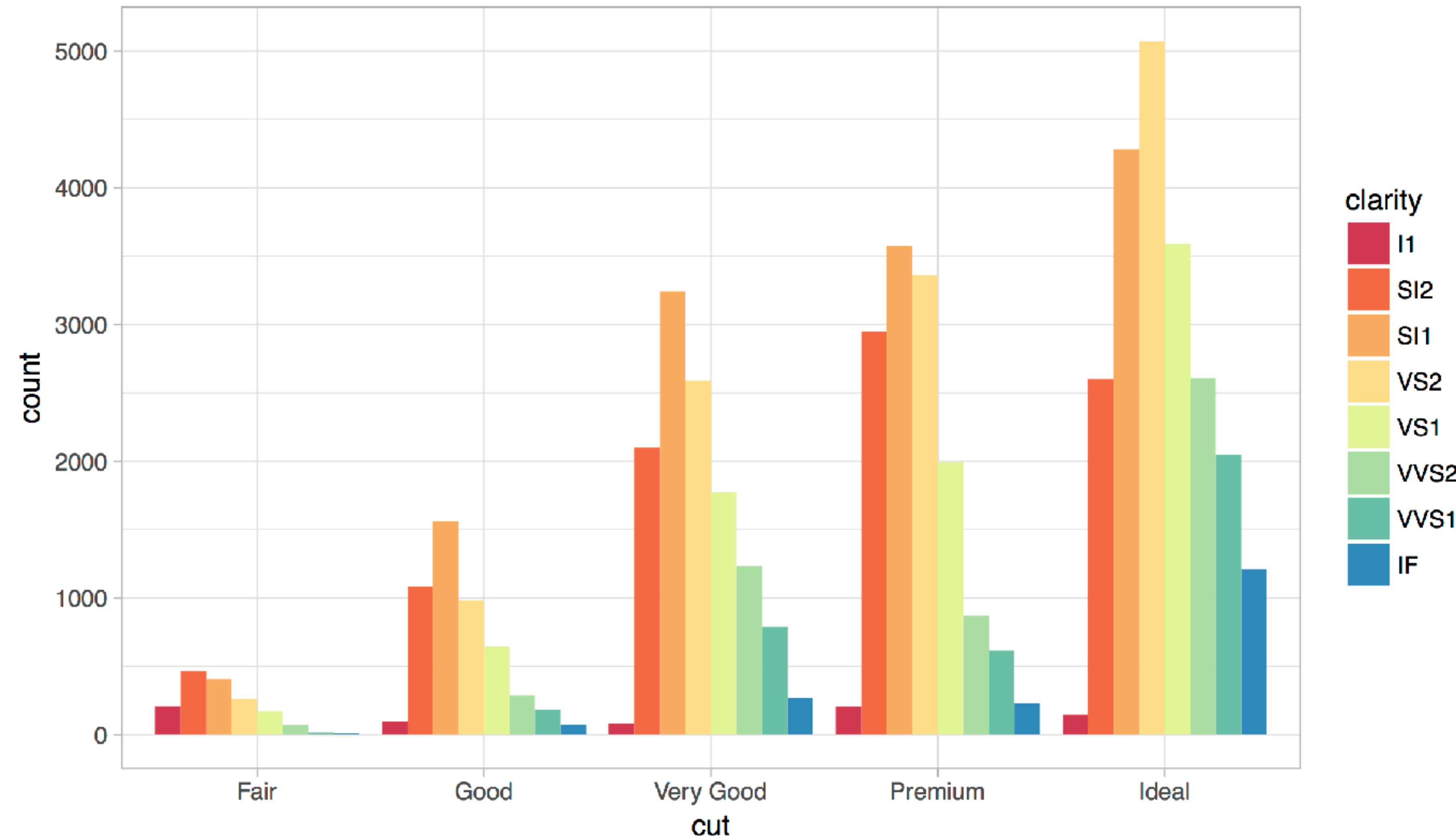
Themes

Visual appearance of non-data elements



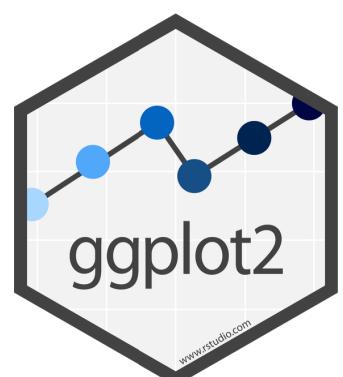
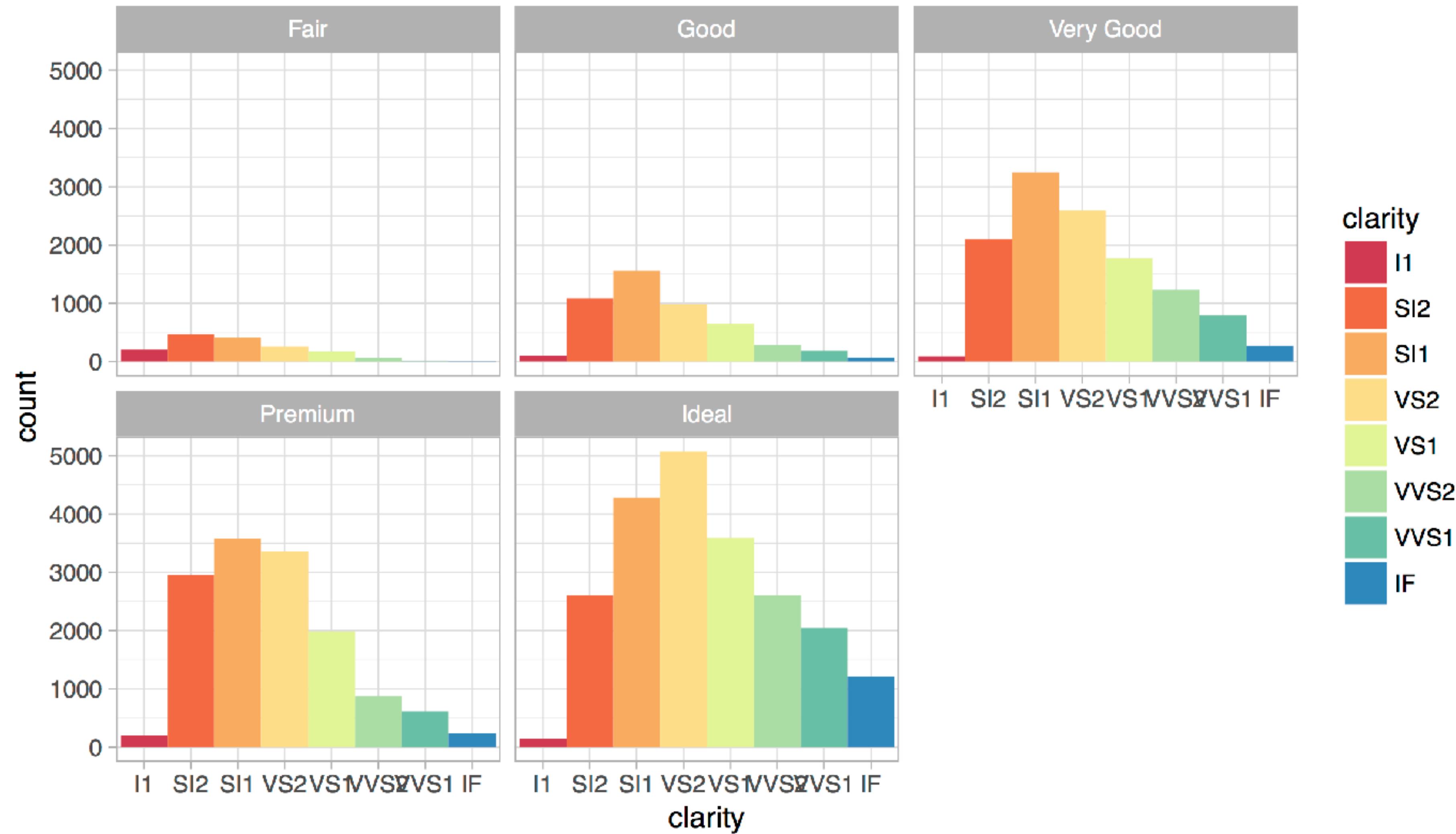
Scales

Customize color scales, other mappings



Facets

Subplots that display subsets of the data.



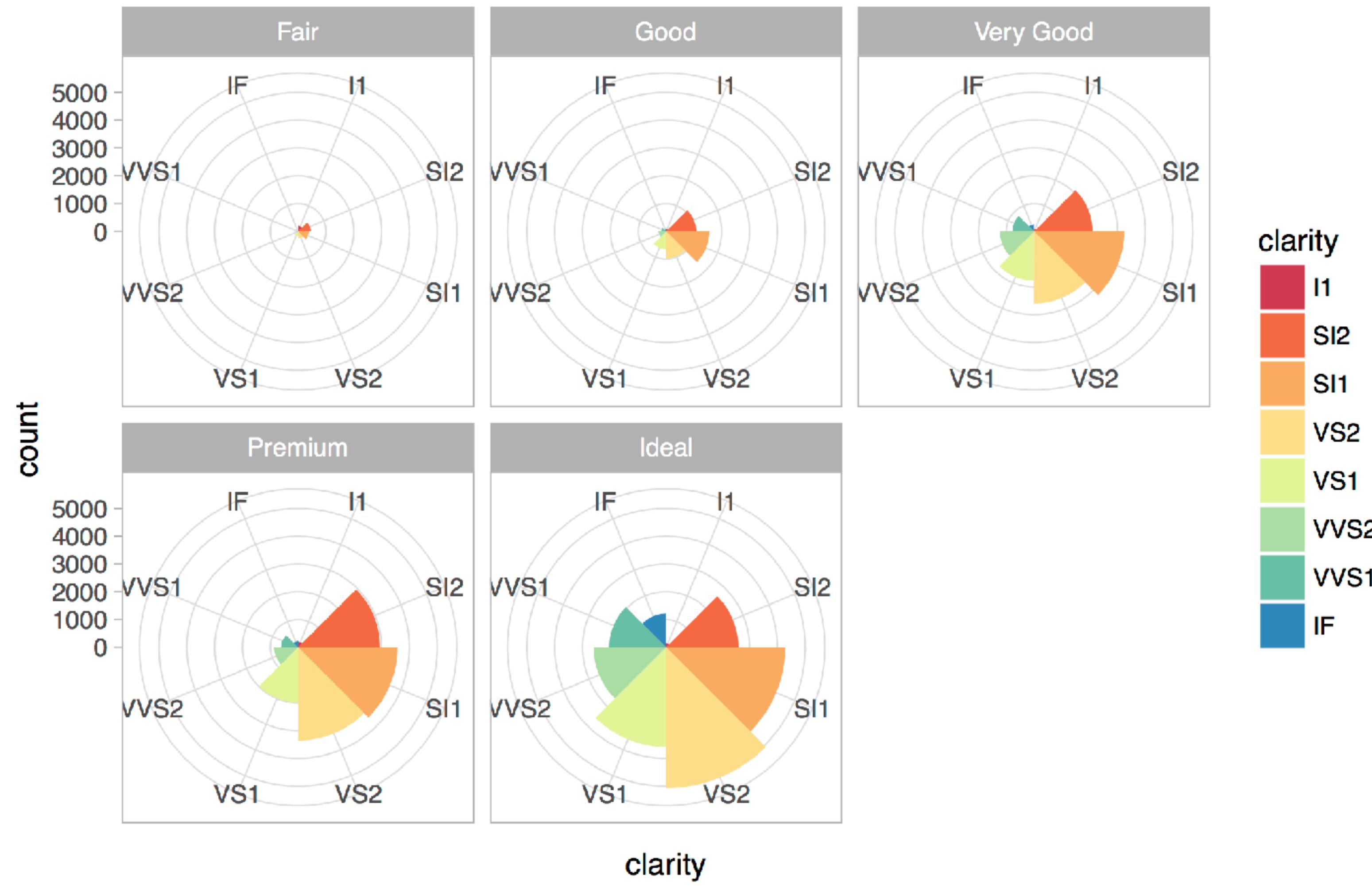
Coordinate systems



Titles and captions

Diamonds data

The data set is skewed towards ideal cut diamonds

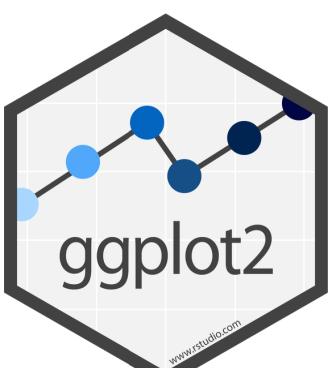
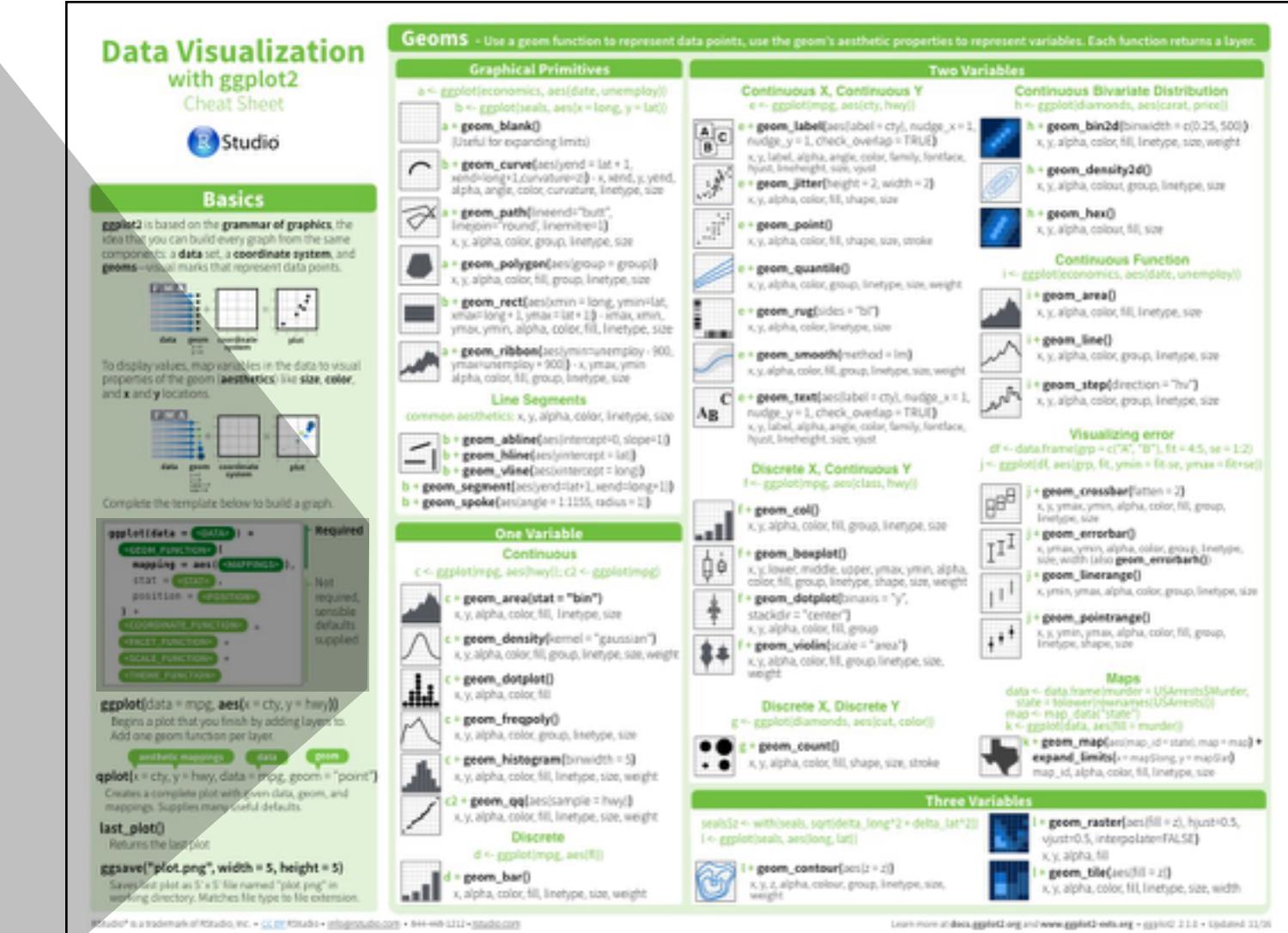


A ggplot2 template

Make any plot by filling in the parameters of this template

```
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



ggplot2.tidyverse.org

The screenshot shows a web browser window with the title "Create Elegant Data Visualisation" and the URL "ggplot2.tidyverse.org". The page content includes:

- ## 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)

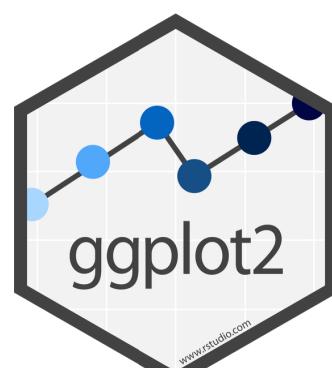
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```

A scatter plot showing fuel efficiency (mpg) versus engine displacement (displ) for different car classes. The plot uses a color scale where purple represents "2seater" cars.
- ## 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

Open **02-Transform-Data.Rmd.**



Visualize Data with

