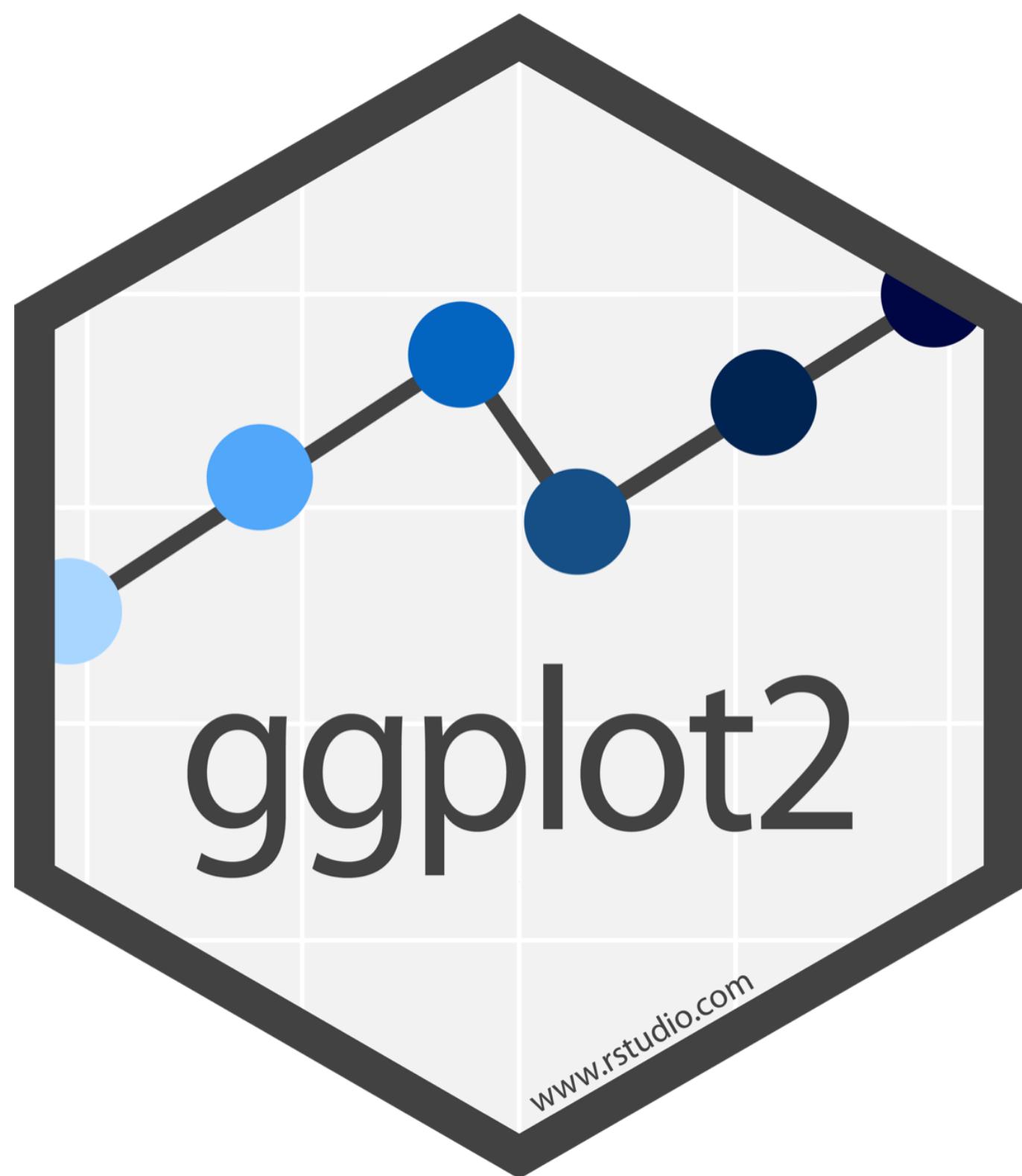


Visualize Data with



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

- John Tukey

Exercise: What do you already know?

- Split into Groups of 2
- Ask your partner:
 - *What do you already know about ggplot2?*
 - *How do you currently visualize data?*
- 5 minutes



ggplot2

Main Ideas

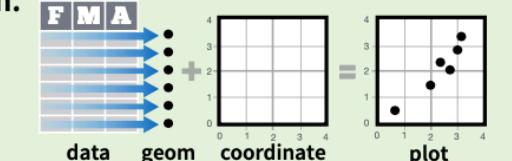
Notes

Data Visualization with ggplot2 Cheat Sheet

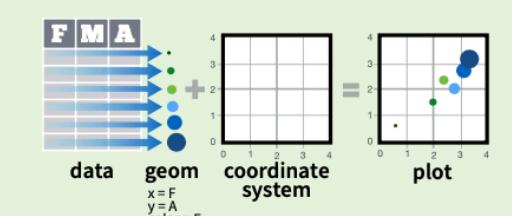


Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data set**, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



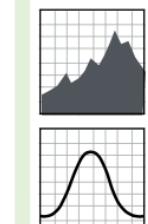
Build a graph with `qplot()` or `ggplot()`

Geoms - Use a geom to represent data points, use the geom's aesthetic properties

One Variable

Continuous

`a <- ggplot(mpg, aes(hwy))`



`a + geom_area(stat = "bin")`

`x, y, alpha, color, fill, linetype, size`

`b + geom_area(aes(y = ..density..), stat = "bin")`

`a + geom_density(kernel = "gaussian")`

`x, y, alpha, color, fill, linetype, size, weight`

`b + geom_density(aes(y = ..county..))`

`a + geom_dotplot()`

`x, y, alpha, color, fill`

`a + geom_freqpoly()`

`x, y, alpha, color, linetype, size`

`b + geom_freqpoly(aes(y = ..density..))`

`a + geom_histogram(binwidth = 5)`

`x, y, alpha, color, fill, linetype, size, weight`

`b + geom_histogram(aes(y = ..density..))`

`Discrete`

`b <- ggplot(mpg, aes(fct))`

`b + geom_bar()`

`x, alpha, color, fill, linetype, size, weight`

Graphical Primitives

Continuous X, Continuous Y

`f <- ggplot(mpg, aes(cty, hwy))`

`f + geom_blank()`

`f + geom_jitter()`

`x, y, alpha, color, fill, shape, size`

`f + geom_point()`

`x, y, alpha, color, fill, shape, size`

`f + geom_quantile()`

`x, y, alpha, color, linetype, size, weight`

`f + geom_rug(sides = "bl")`

`alpha, color, linetype, size`

`f + geom_smooth(model = lm)`

`x, y, alpha, color, fill, linetype, size,`

`f + geom_text(aes(label = cty))`

`x, y, label, alpha, angle, color, family`

`A B`

`hjust, lineheight, size, vjust`

Discrete X, Continuous Y

`g <- ggplot(mpg, aes(class, hwy))`

`g + geom_bar(stat = "identity")`

Notes form & Cheat Sheet

Please take out

Your Turn

Click on **exercises/01-ggplot-exercises.Rmd**



Setup

The setup chunk is always run once before anything else

The screenshot shows the RStudio interface with the following details:

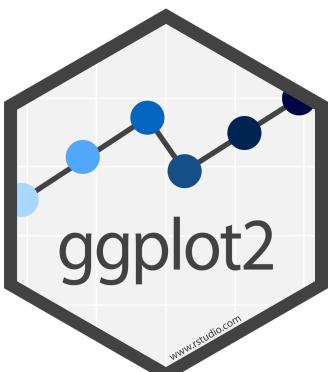
- Title Bar:** The title is "01-Visualize-Data.Rmd".
- Toolbar:** Includes icons for back, forward, file, preview, insert, run, and other common functions.
- Code Editor:** The main area contains R code. Lines 6-8 show a code chunk setup: ````{r setup}`` followed by `library(tidyverse)`. Lines 10-12 show another setup chunk: ````{r}``` followed by `mpg` and `````. Lines 15-17 are instructions: `## Your Turn 1` followed by "Run the code on the slide to make a graph. Pay strict attention to spelling, capitalization, and parentheses!". Lines 23-25 show another turn section: `## Your Turn 2` followed by "Add `color`, `size`, `alpha`, and `shape` aesthetics to your graph. Experiment.". Lines 27-30 show a third turn section: `## Your Turn 3` followed by "Replace this scatterplot with one that draws boxplots. Use the cheatsheet. Try your best guess.". Lines 36-39 show a final turn section: `## Your Turn 4` followed by "Replace this scatterplot with one that draws boxplots. Use the cheatsheet. Try your best guess.".
- Preview Pane:** A large preview pane on the right side displays the rendered content of the R code. It shows the `library(tidyverse)` command and three sets of three dots indicating continuation of the code.

chunk labels are optional,
the setup label is special

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!



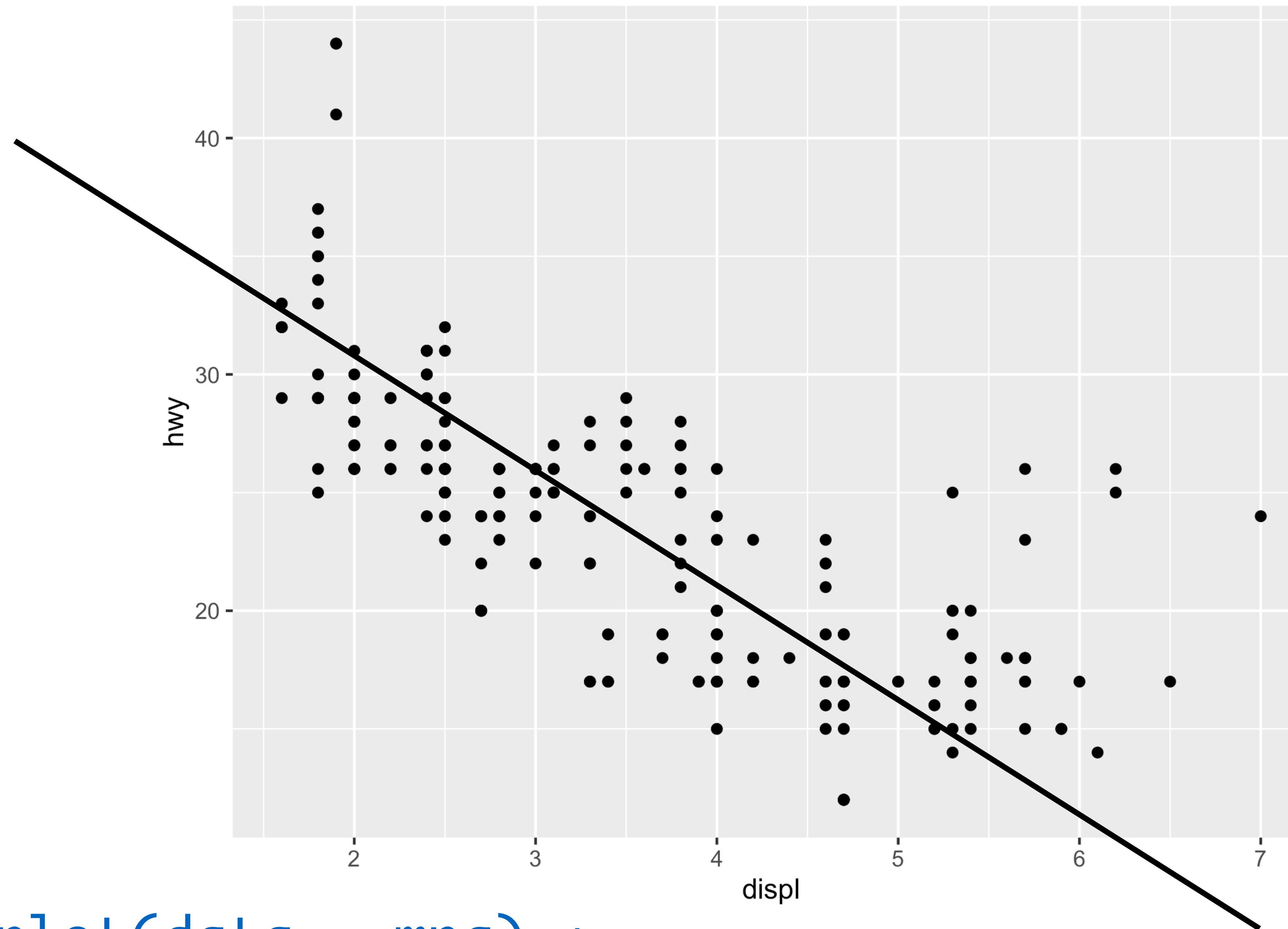
Your Turn 1

Run this code in your notebook to make a graph.

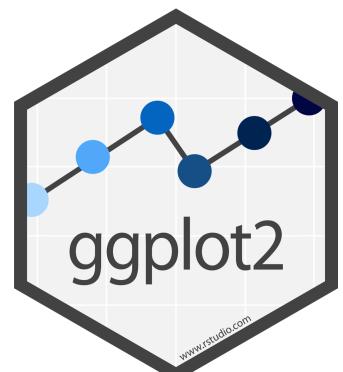
Pay strict attention to spelling, capitalization, and parentheses!

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



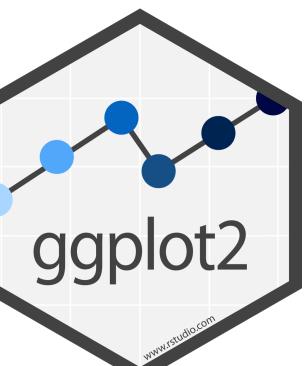


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```



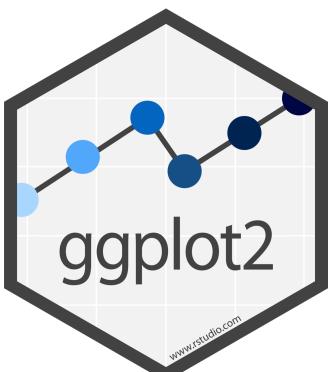
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

```
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

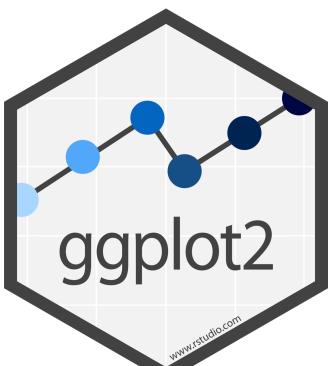
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

type of layer

aes()

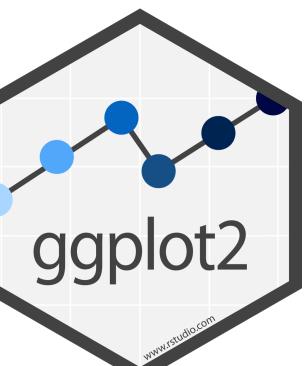
x variable

y variable



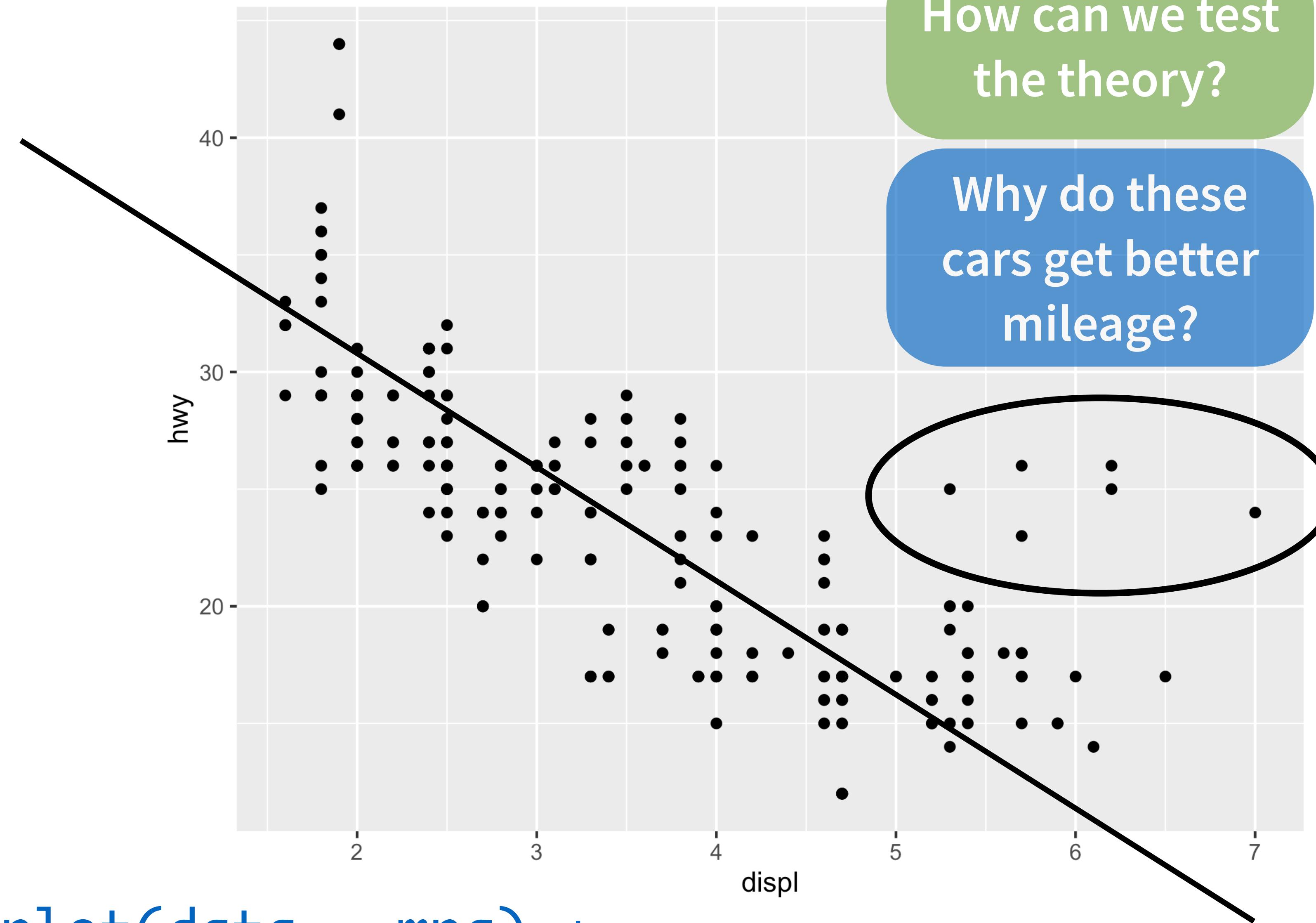
A template

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



Mappings

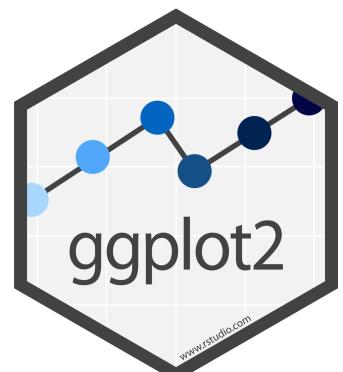
R



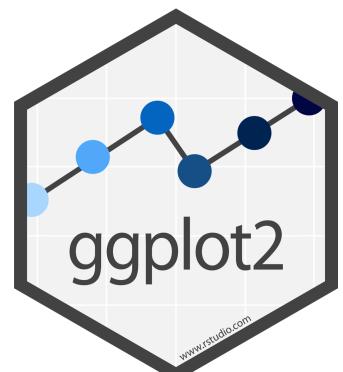
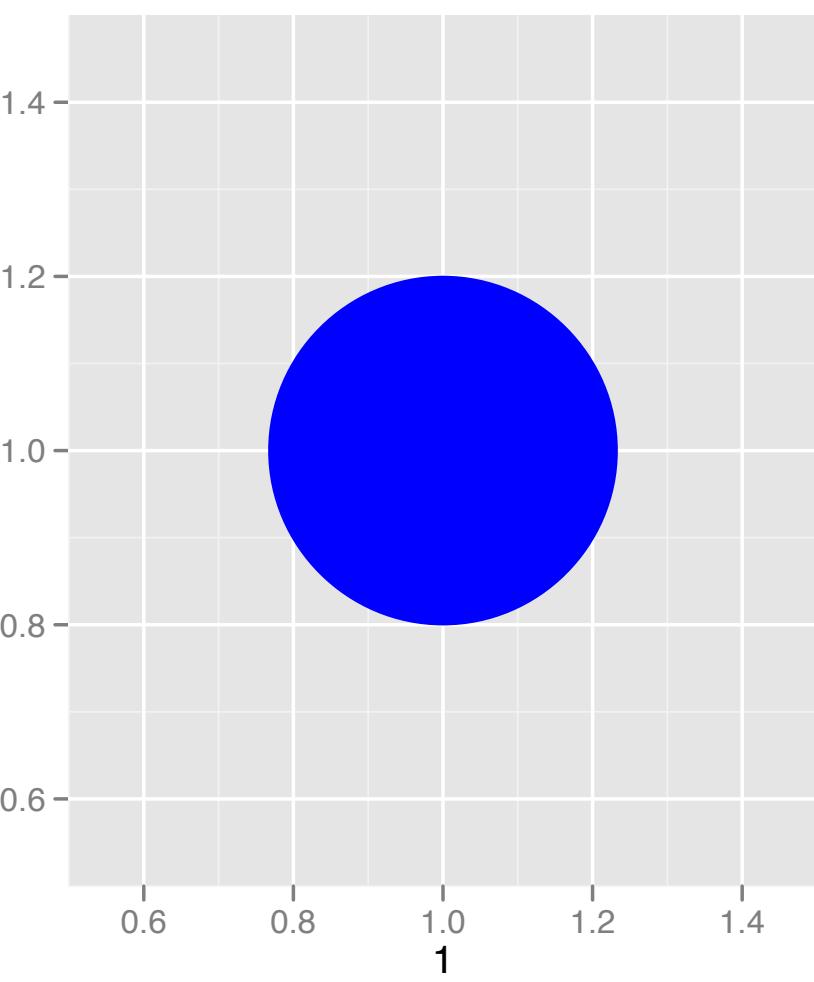
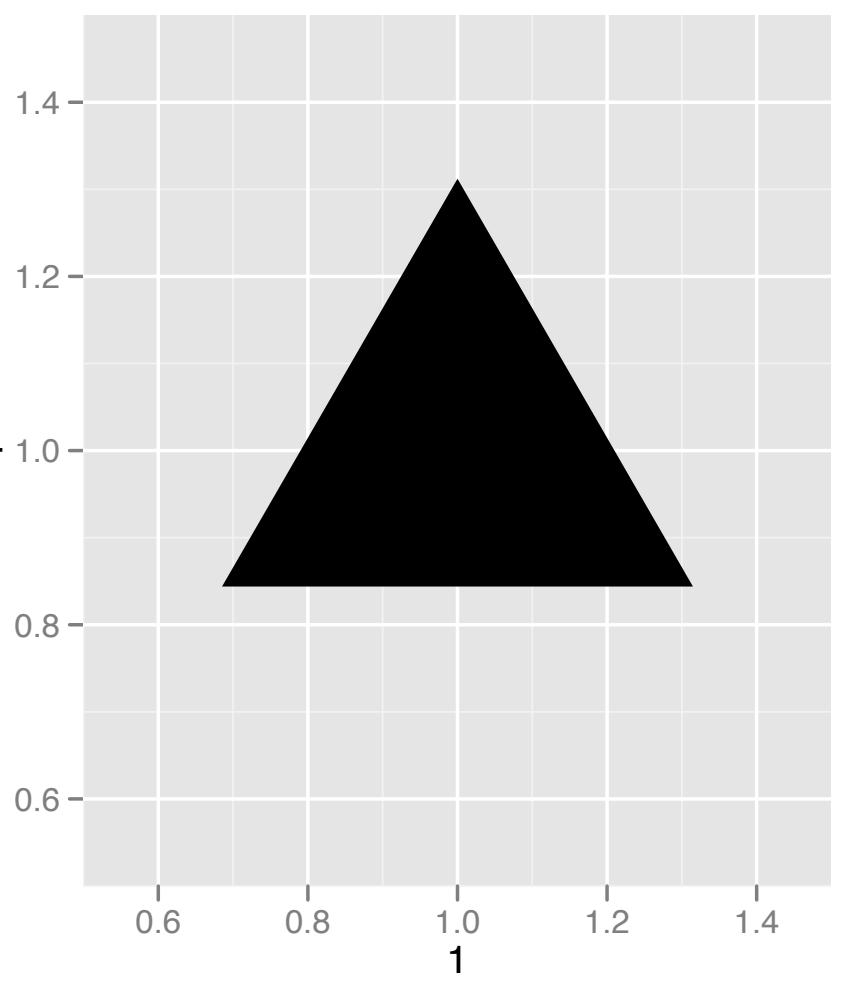
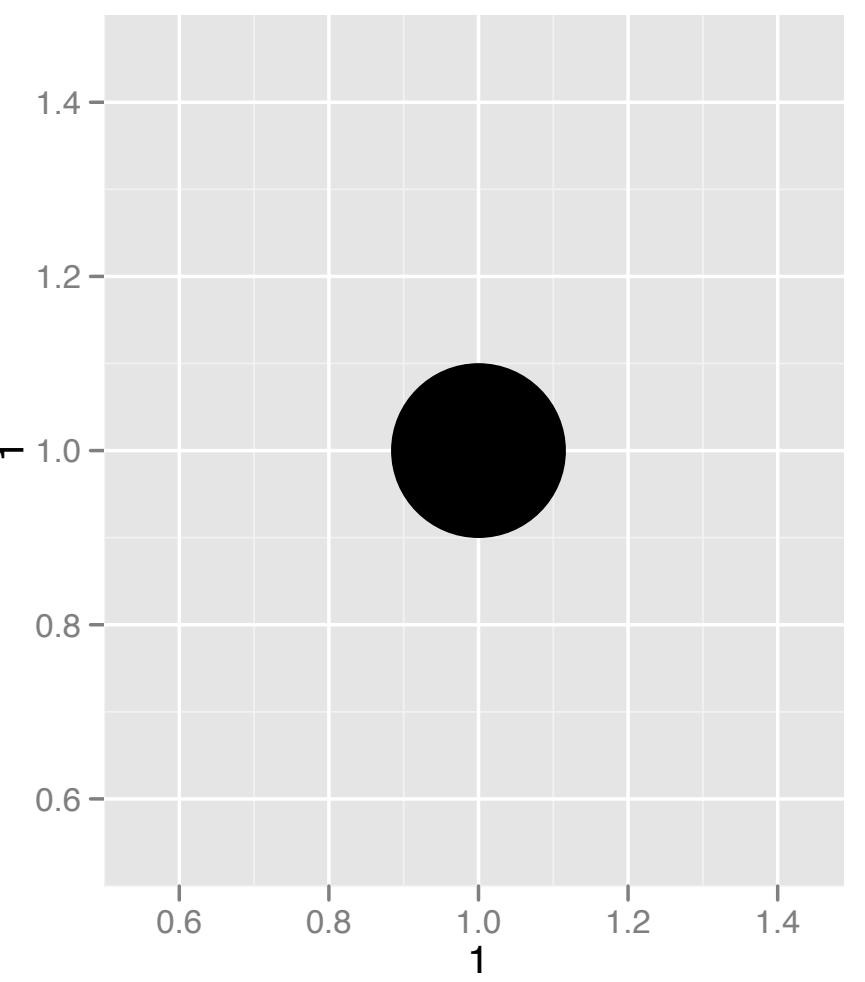
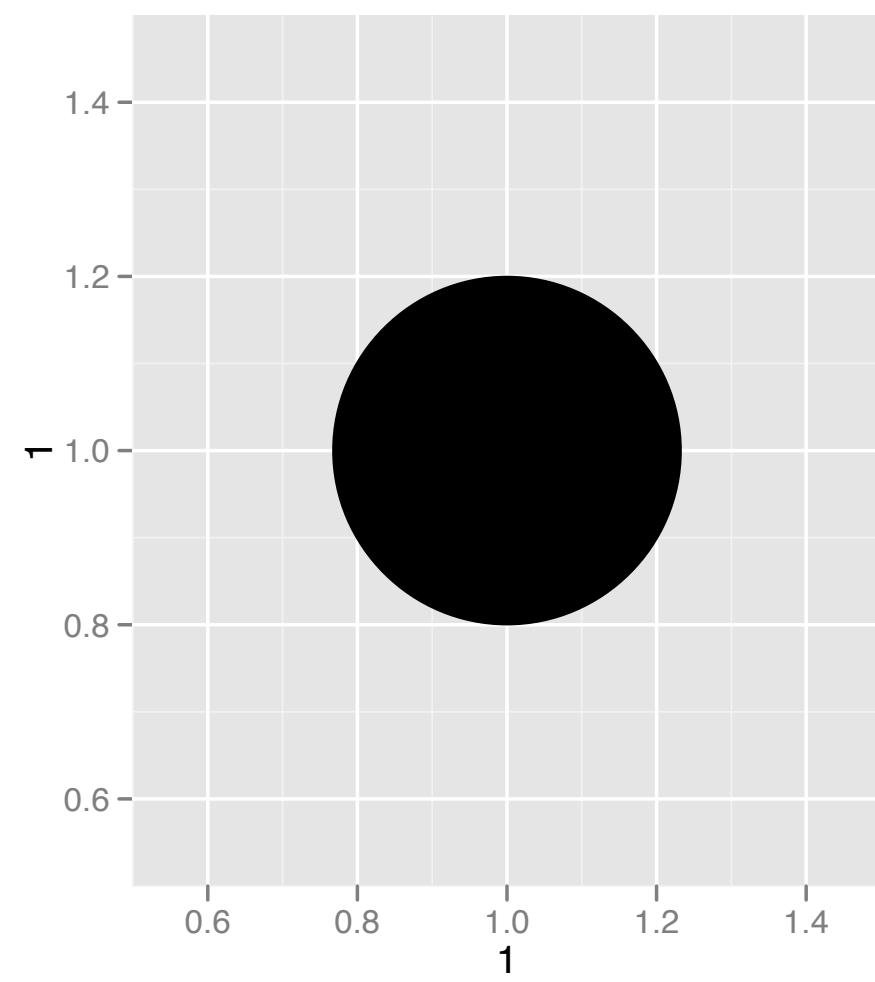
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can we test
the theory?

Why do these
cars get better
mileage?



Aesthetics



Visual Space

color

Red

Brown

Green

Aqua

Blue

Violet

Pink

Data Space

class

2seater

compact

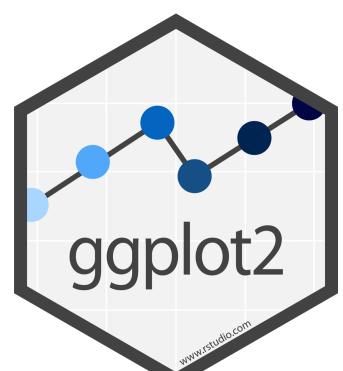
midsize

minivan

pickup

subcompact

suv

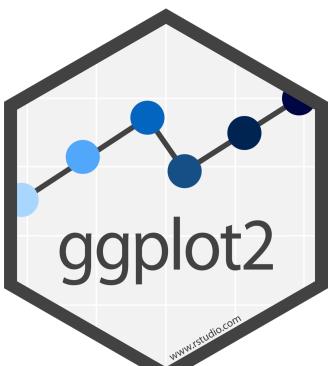


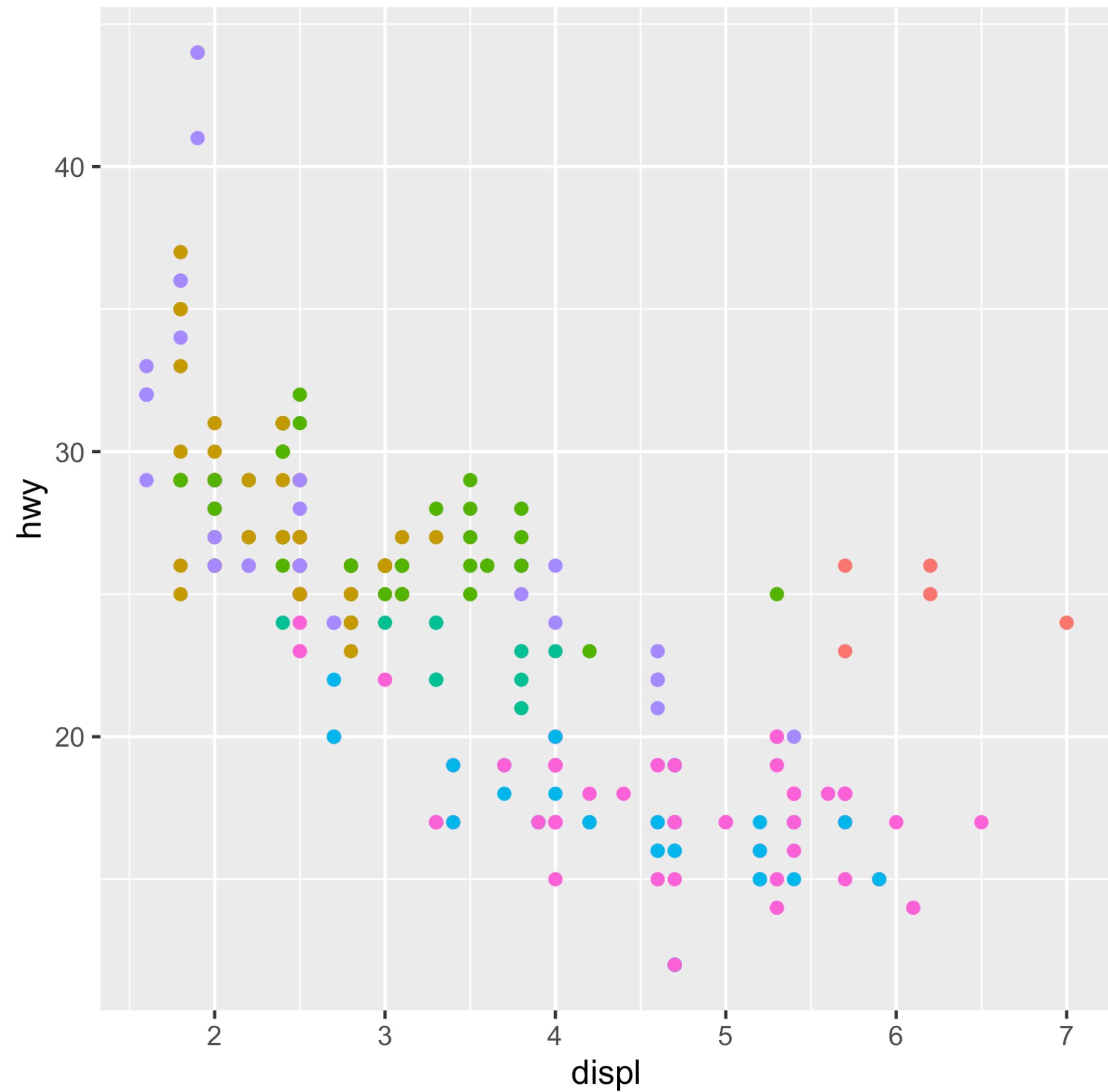
Aesthetics

aesthetic
property

Variable to
map it to

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

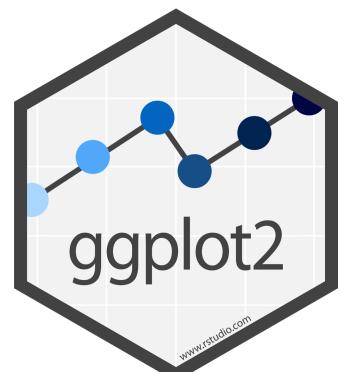




```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

Legend added
automatically

class
2seater
compact
midsize
minivan
pickup
subcompact
suv



Your Turn 2

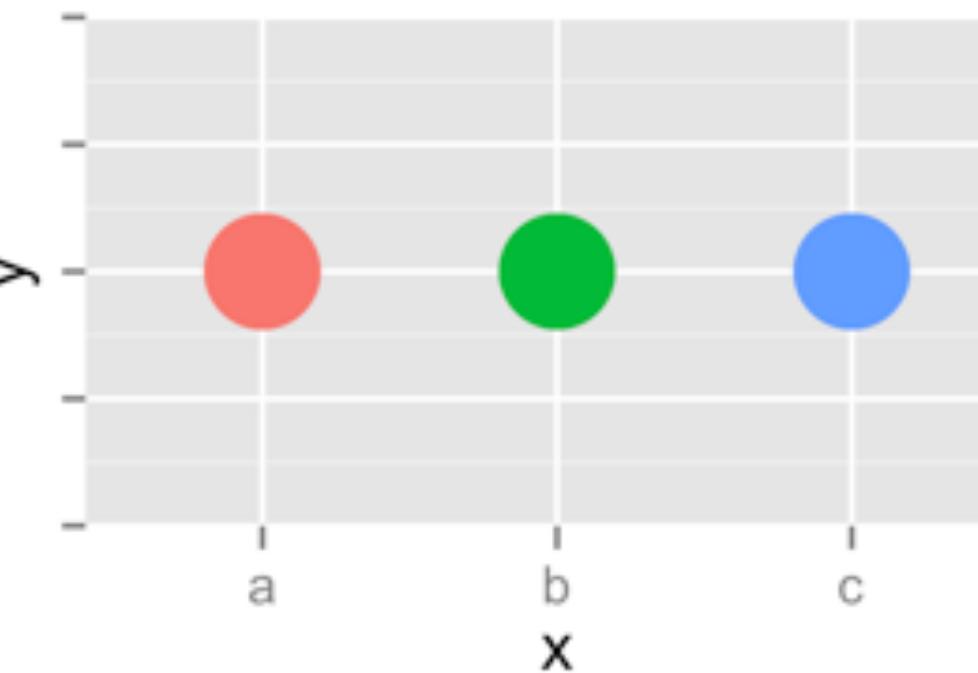
In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

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

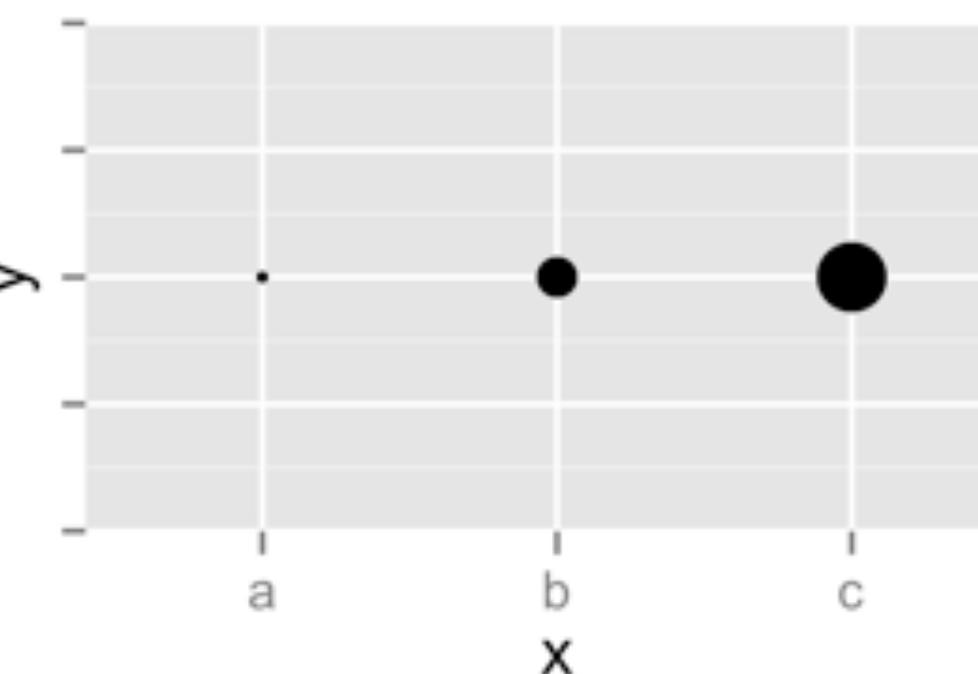
What happens when you use more than one aesthetic?



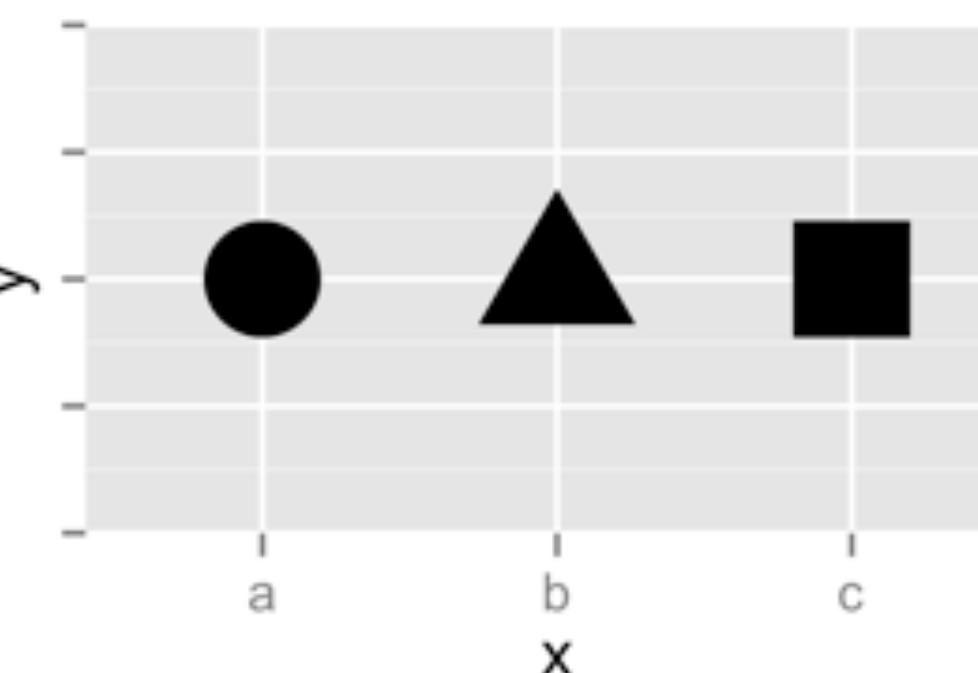
Color



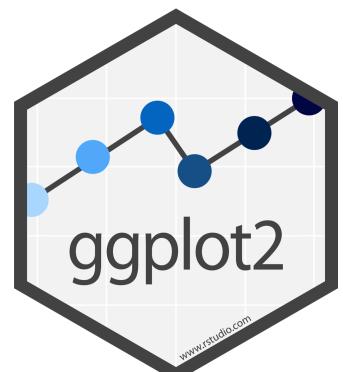
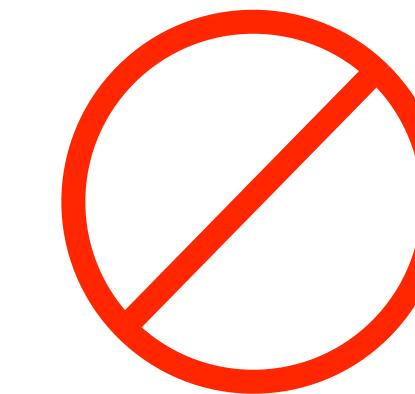
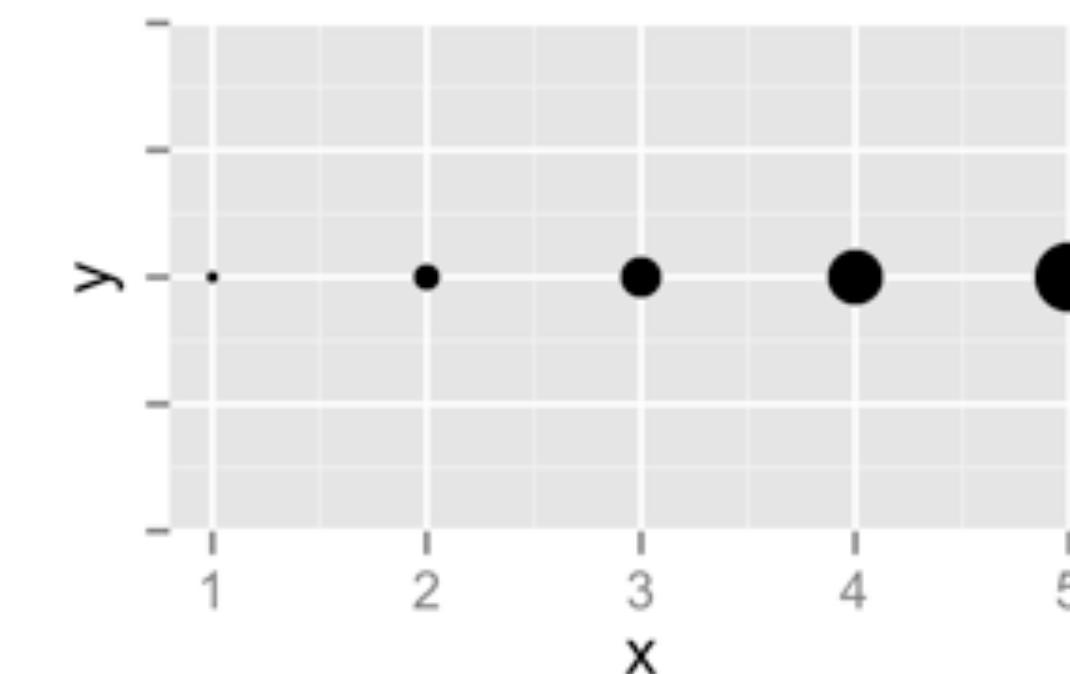
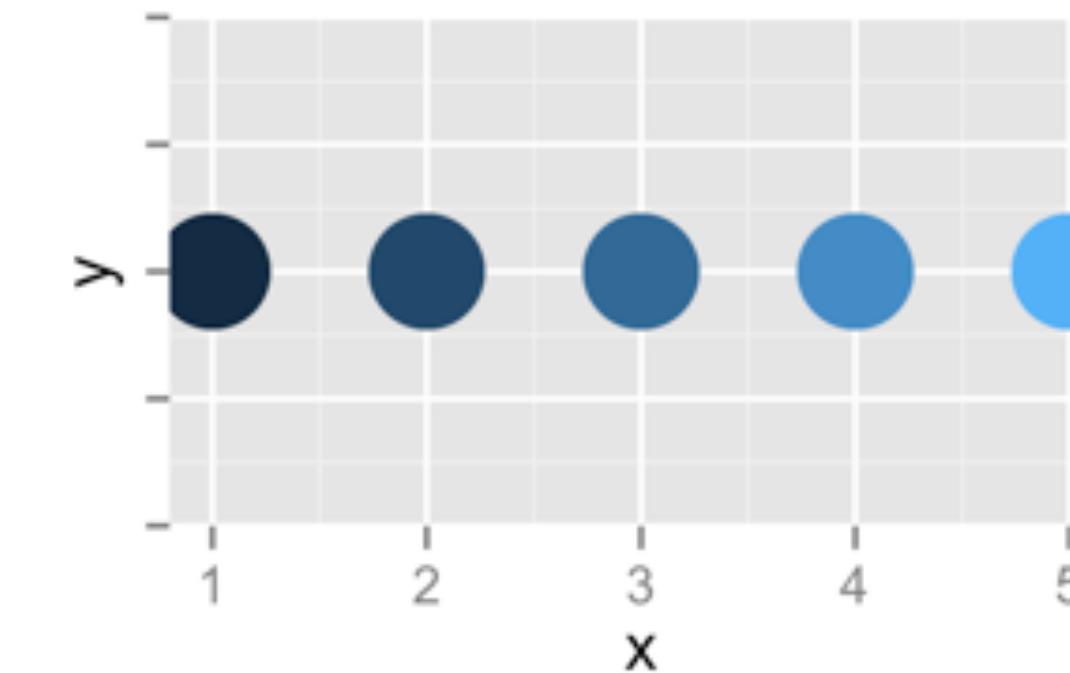
Size



Shape



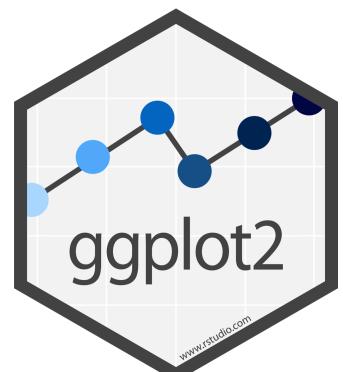
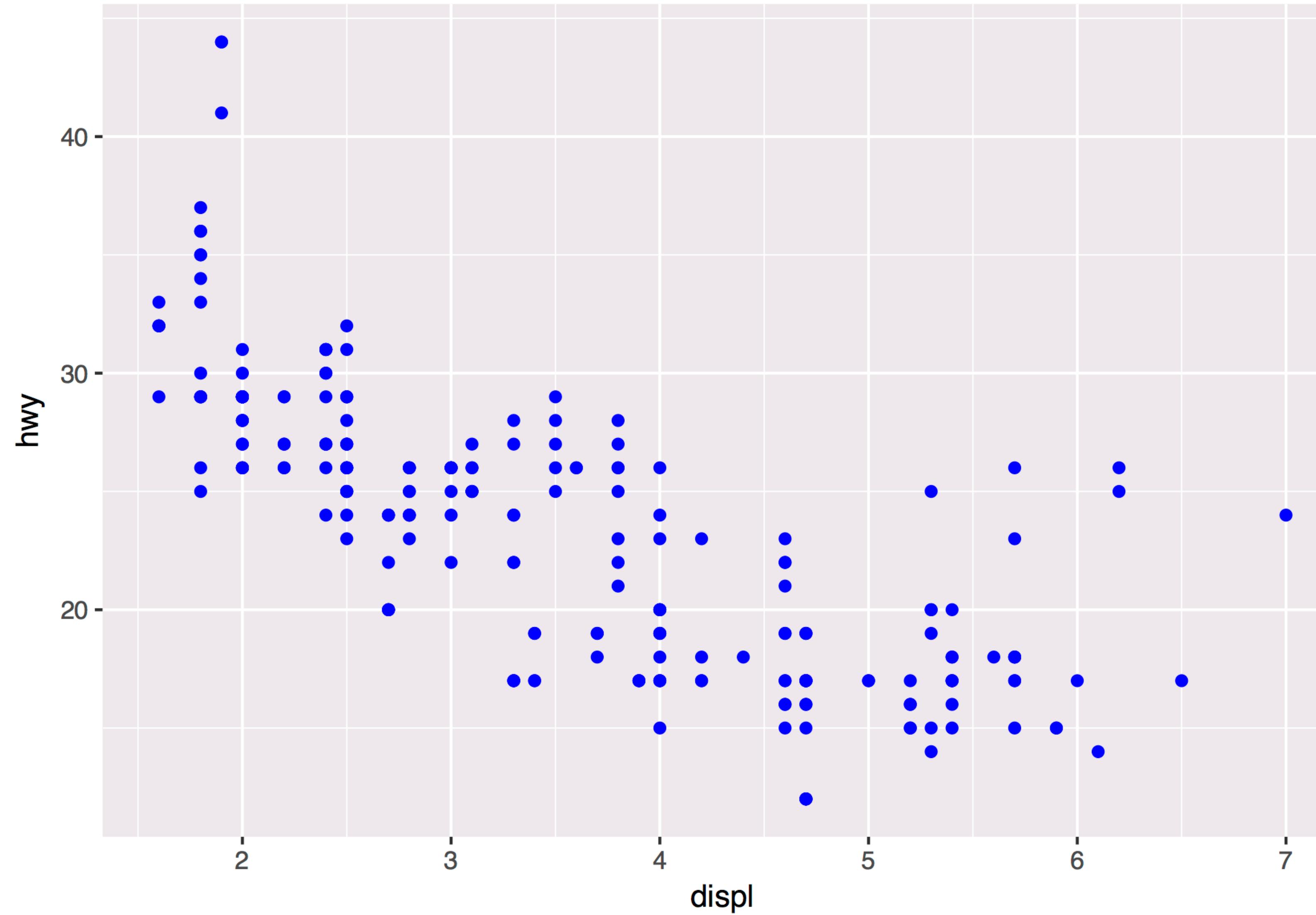
Continuous

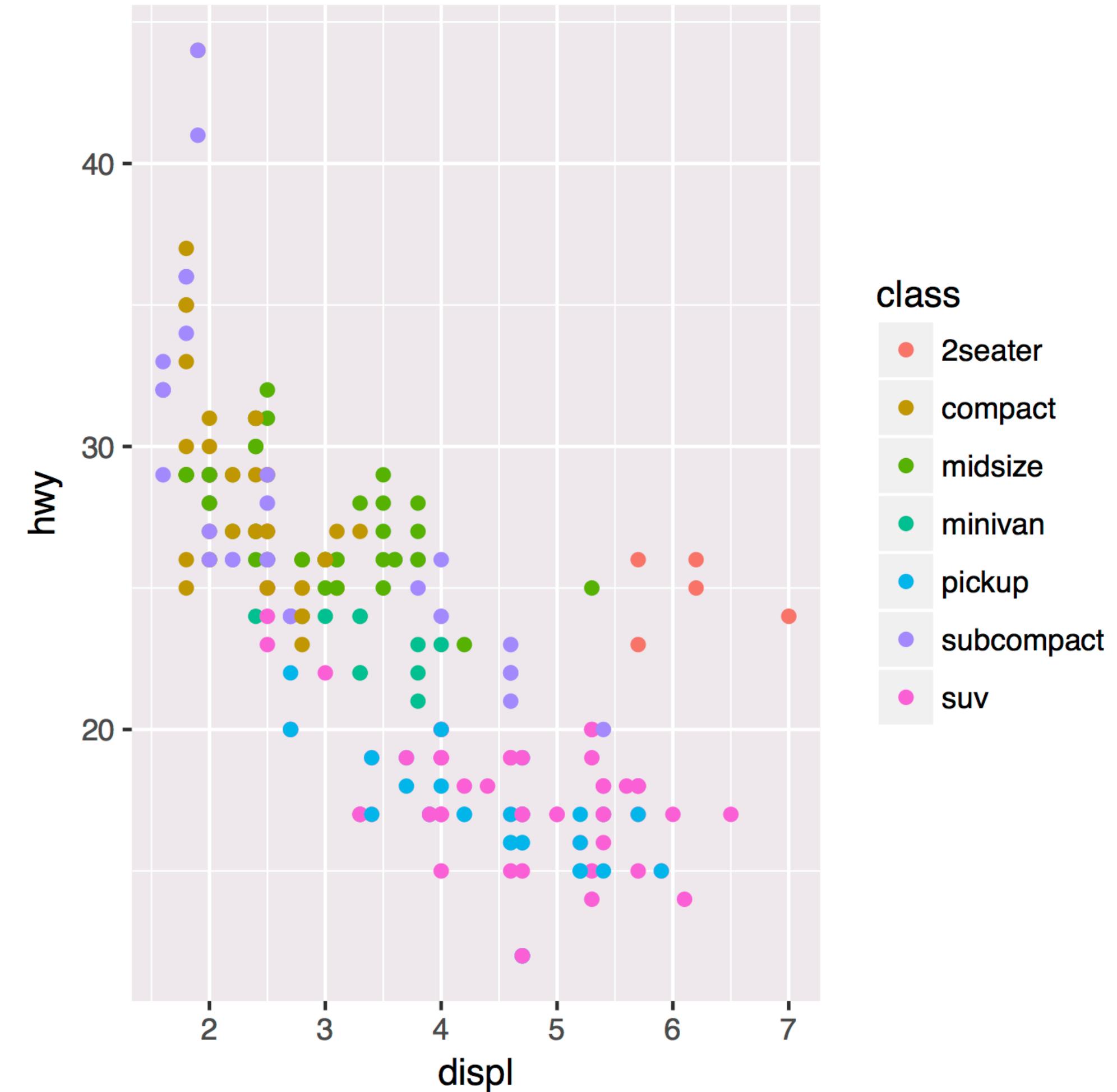


set vs. map

A large, semi-transparent watermark of the R logo is positioned in the bottom right corner. The logo consists of a circular emblem with the letters "R" inside.

How would you make this plot?

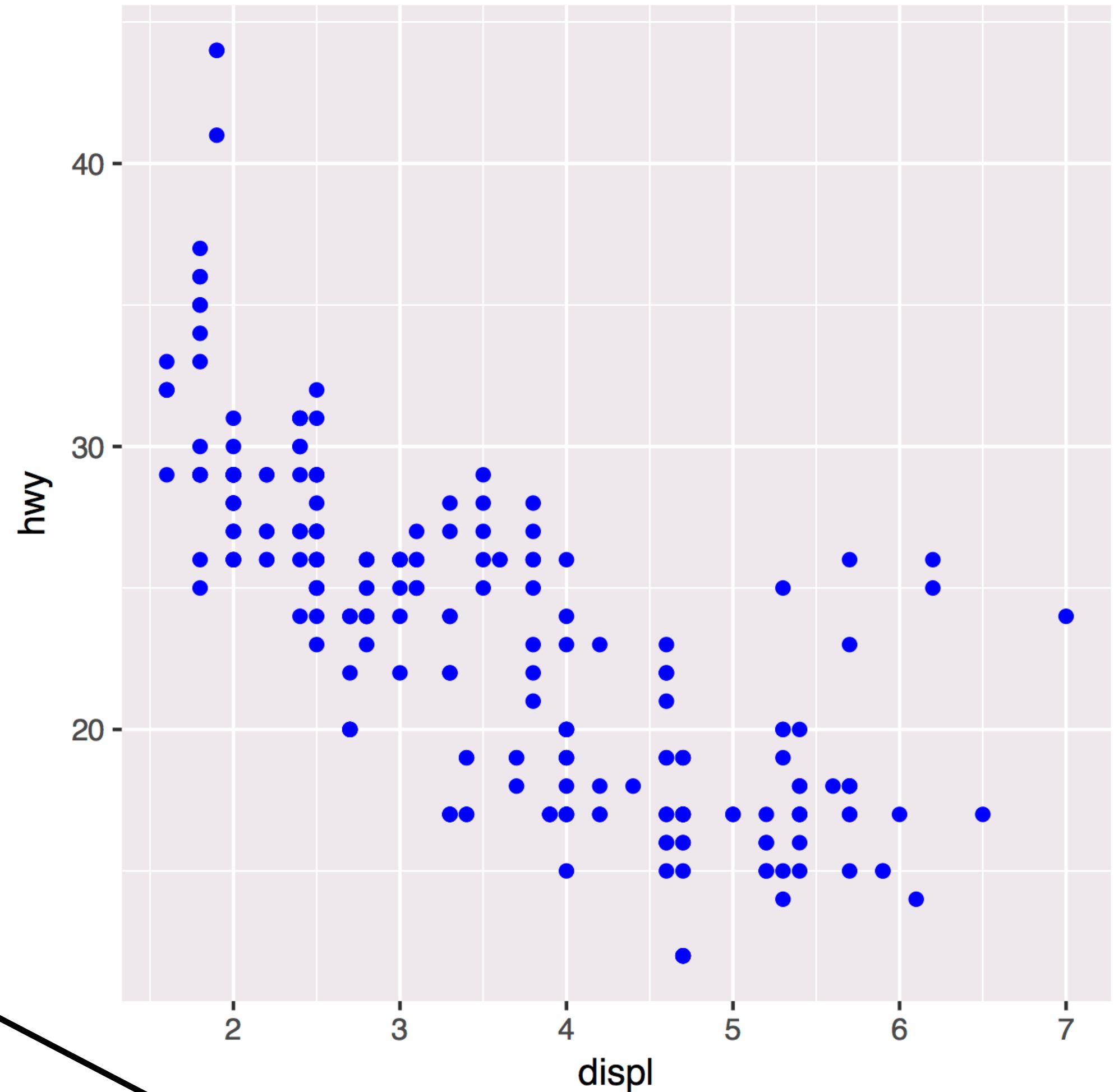




Inside of aes(): maps an aesthetic to a variable

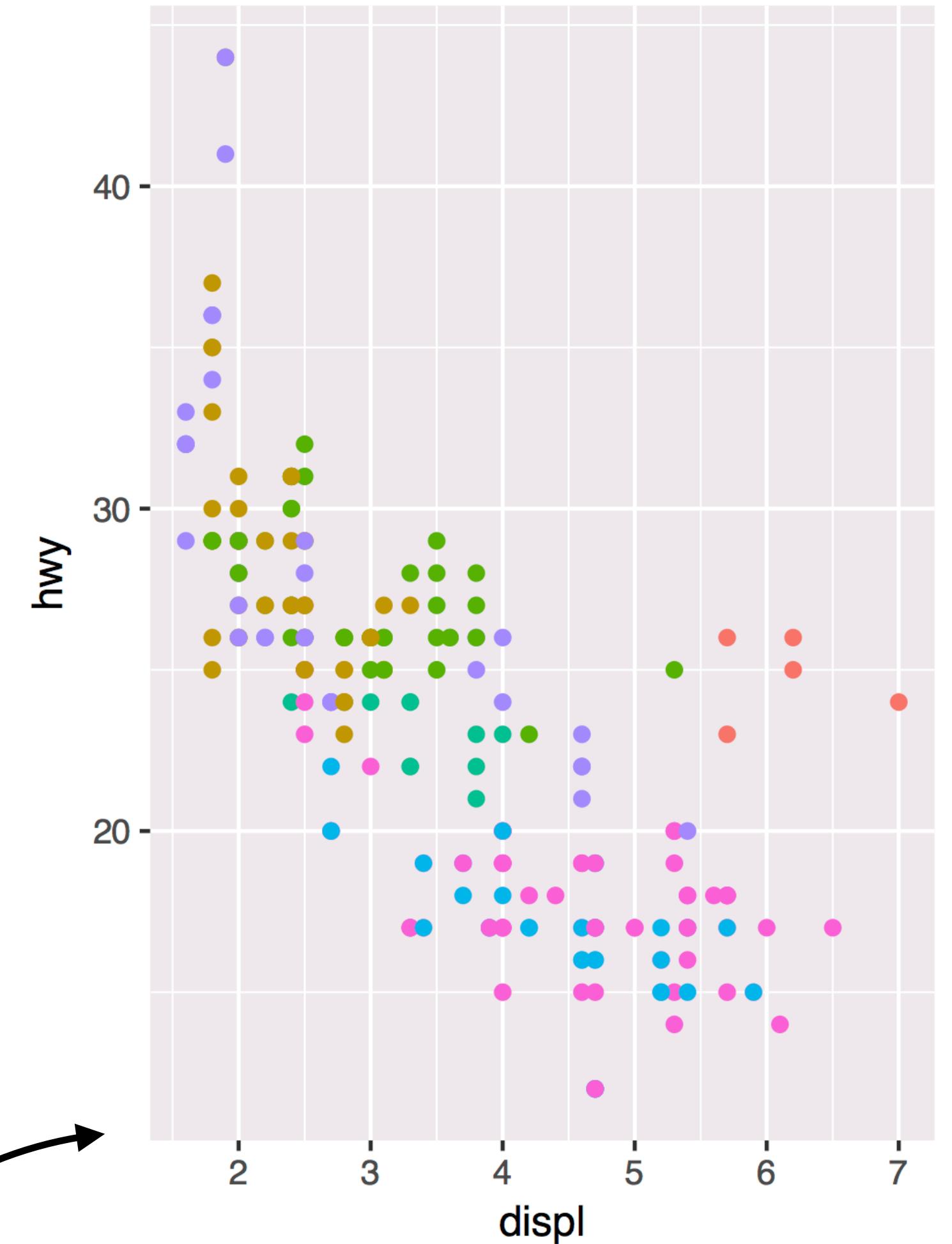
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



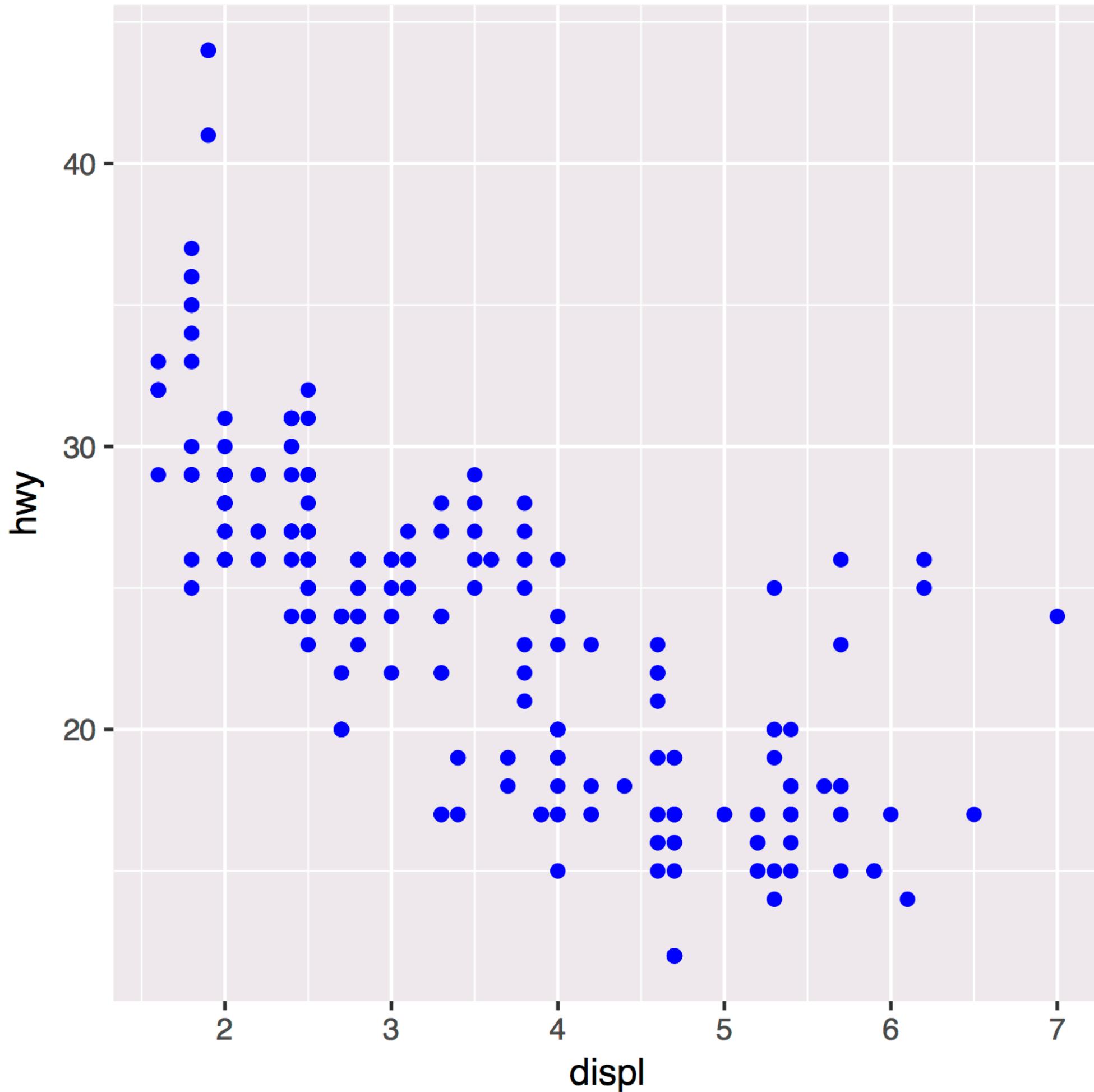
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



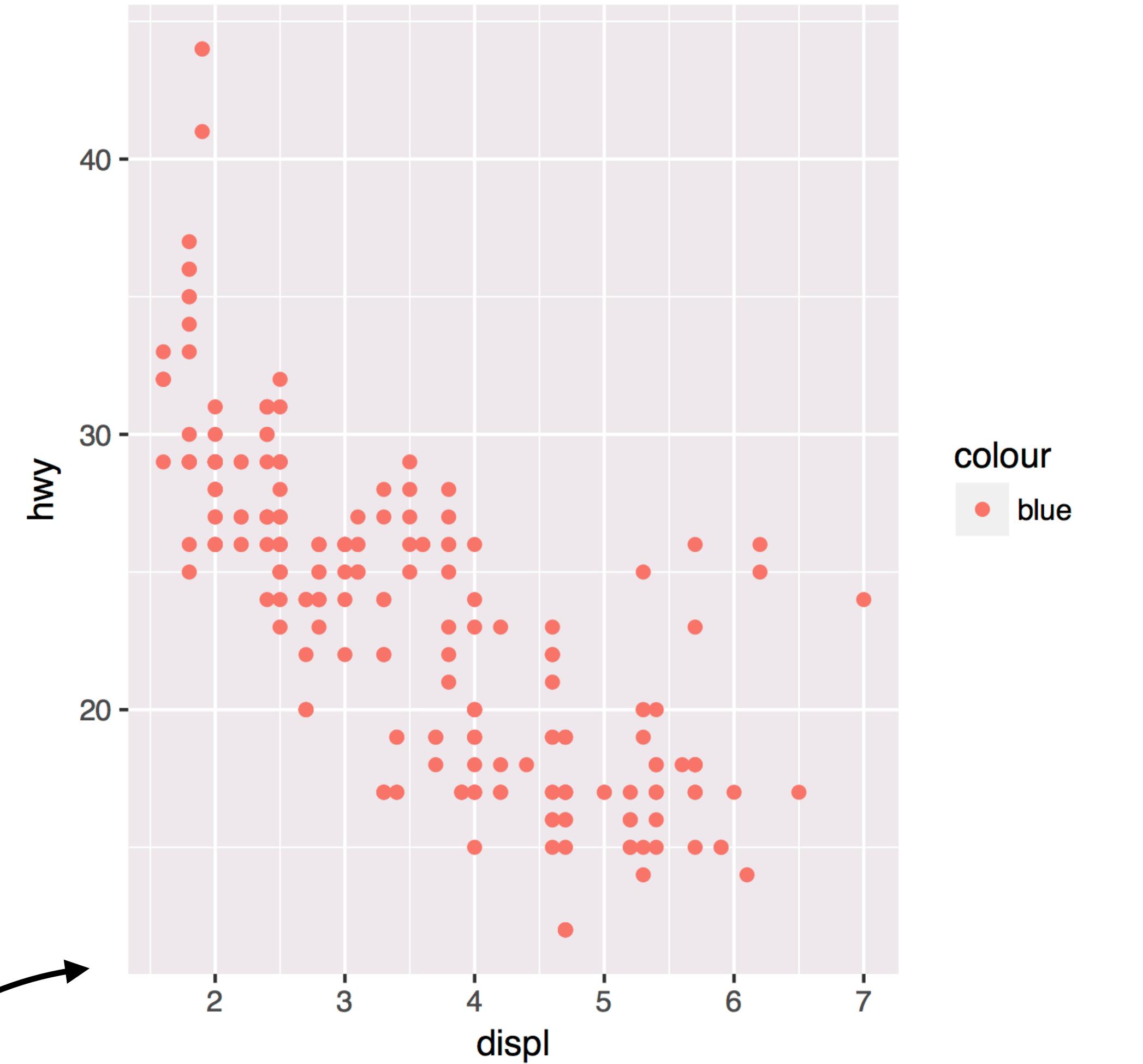
class

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

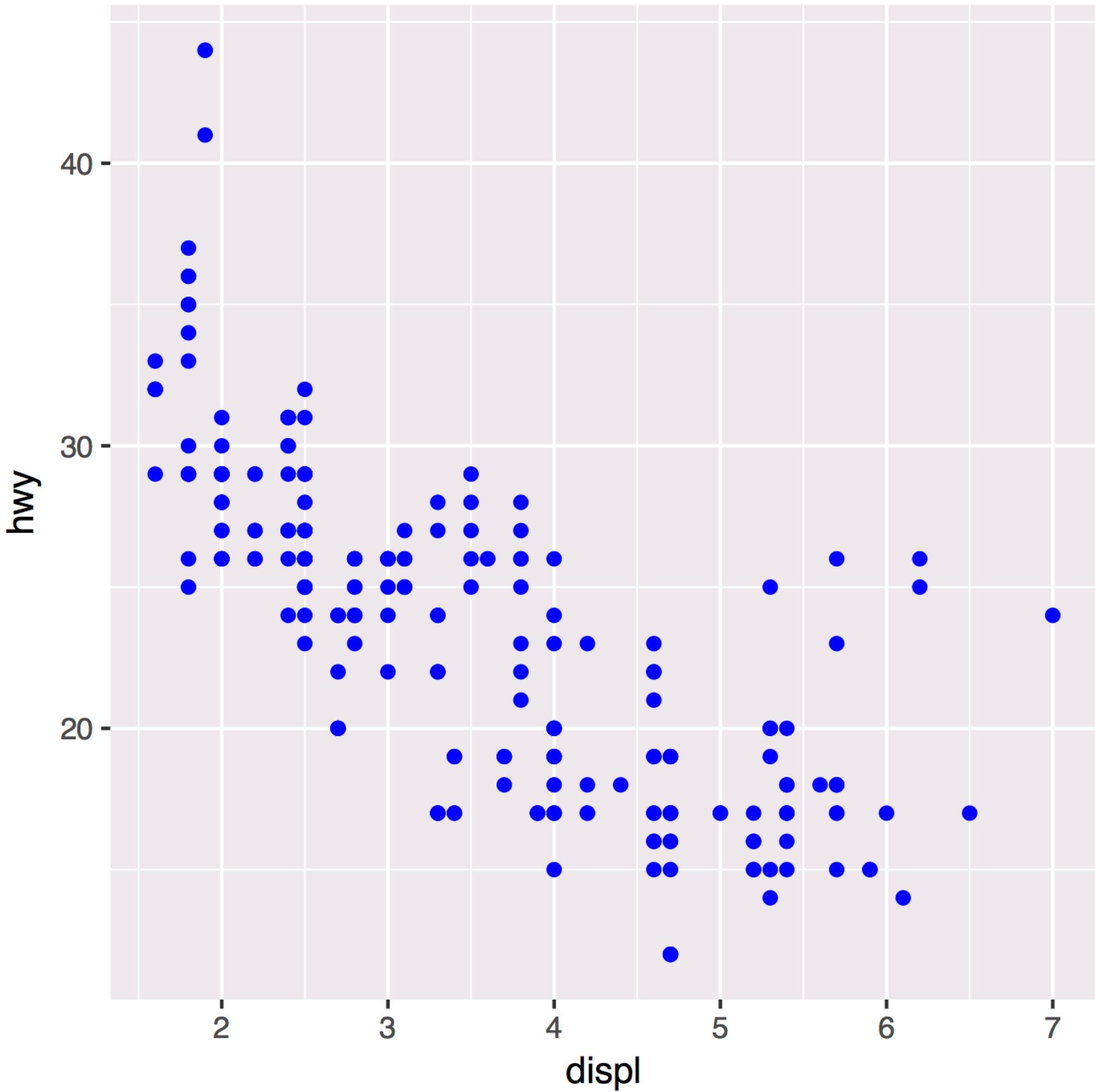


```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
```

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



colour
● blue



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

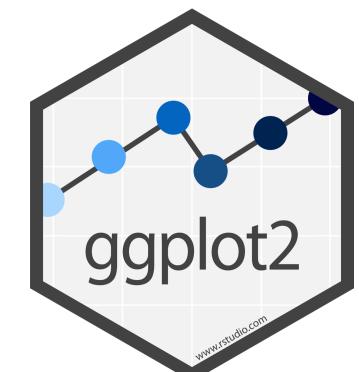
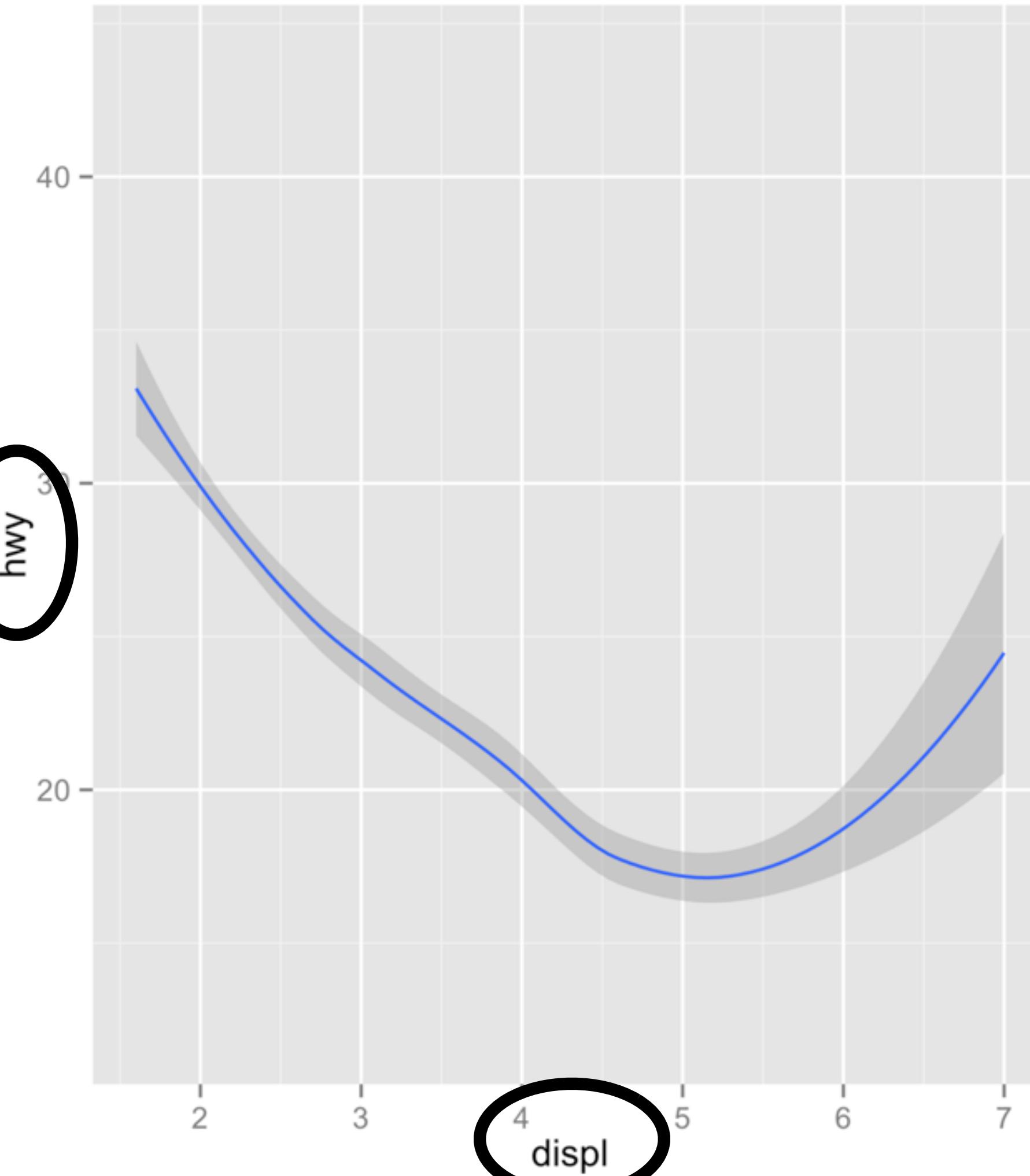
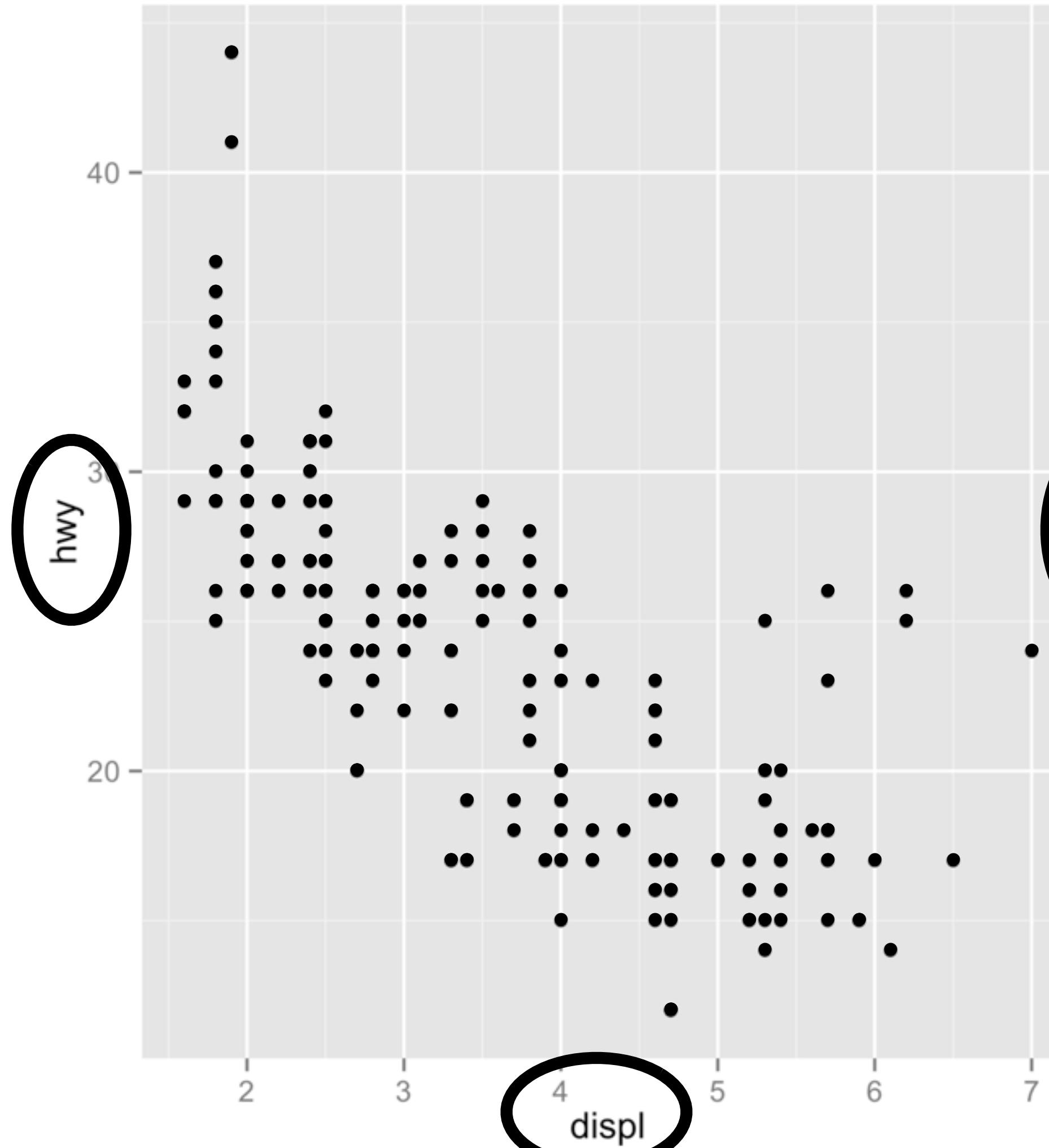
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), color = "blue")
```

Geoms



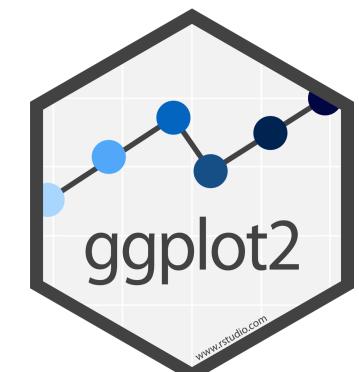
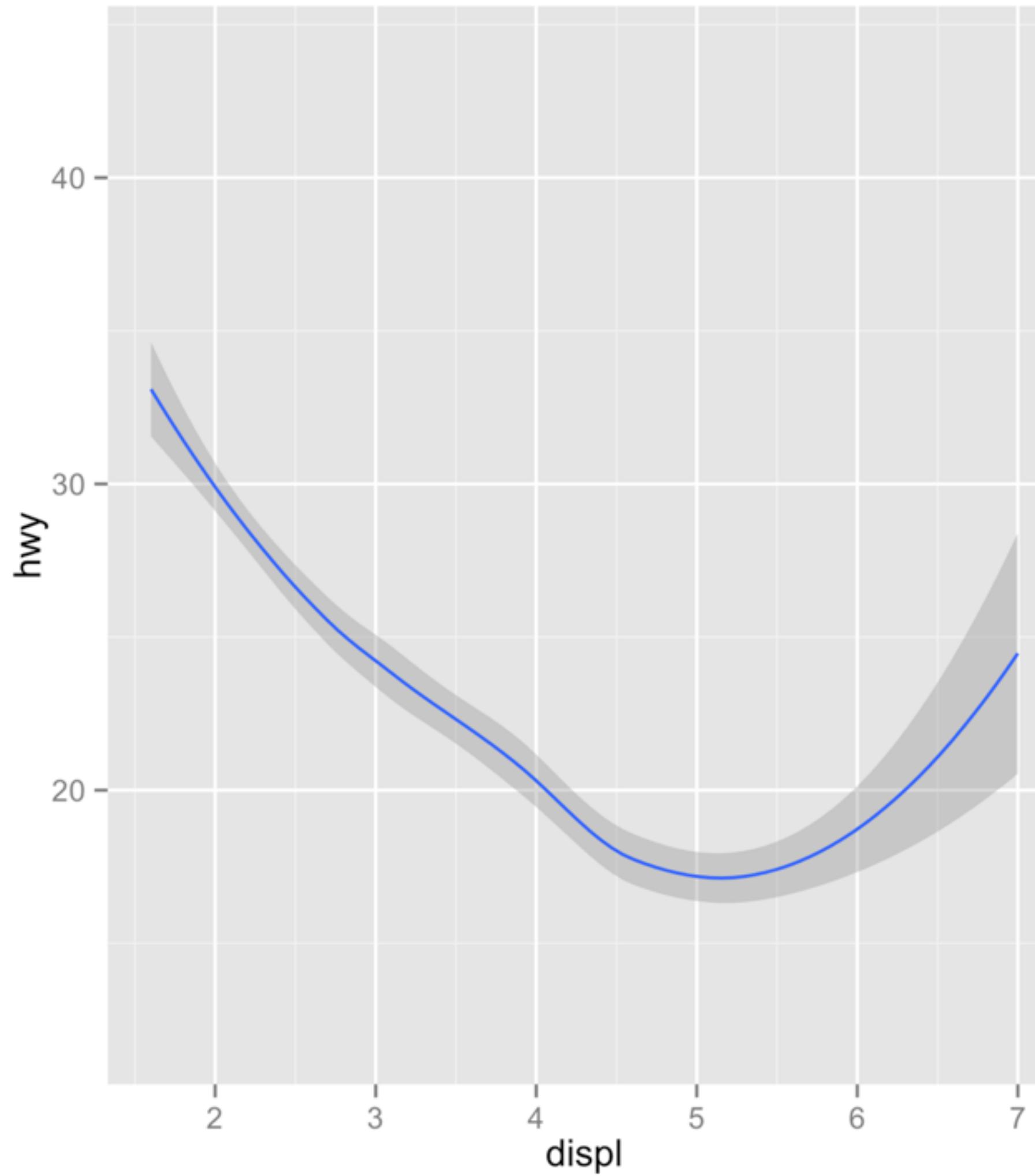
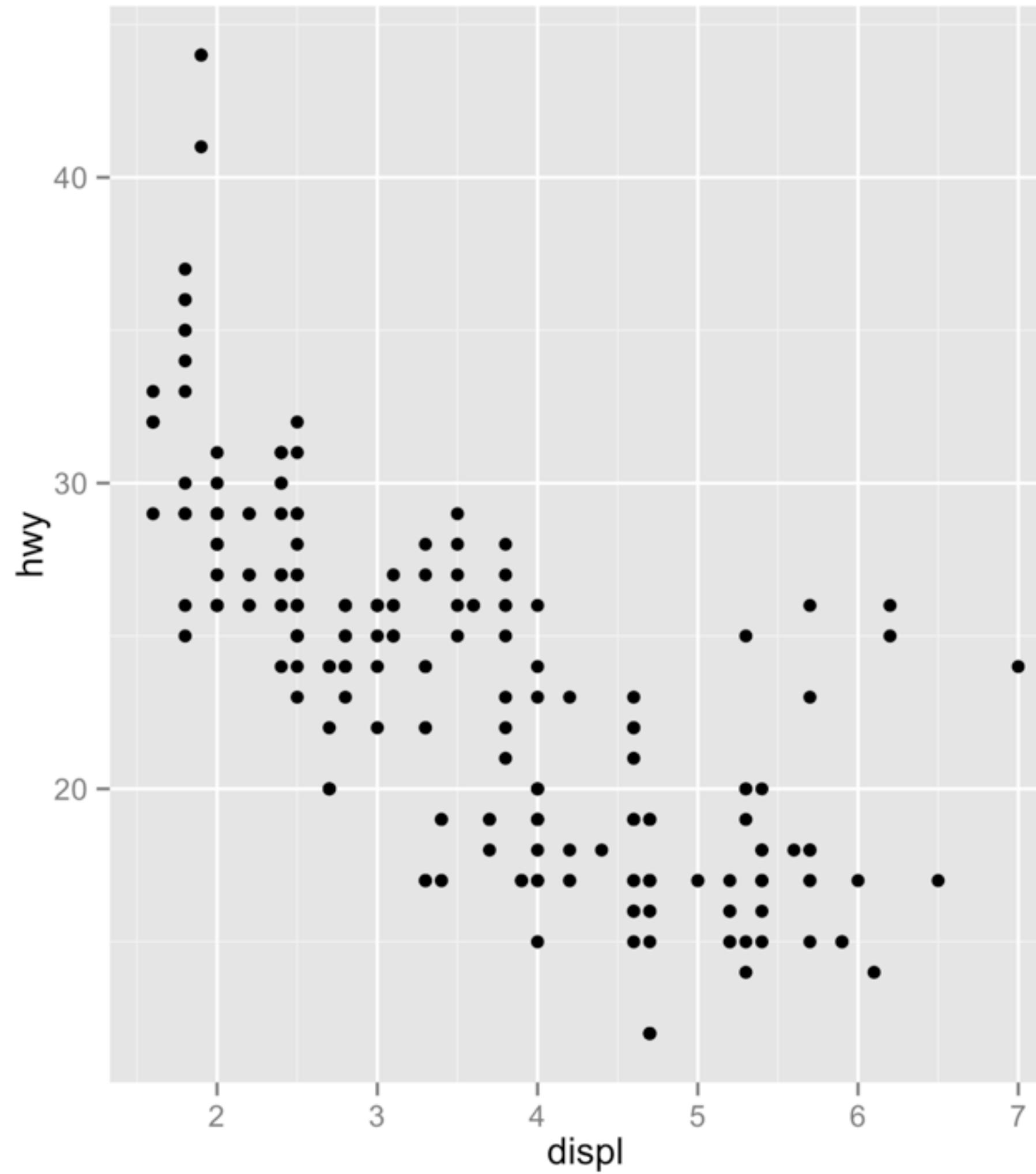
How are these plots similar?

Same: x var , y var , data



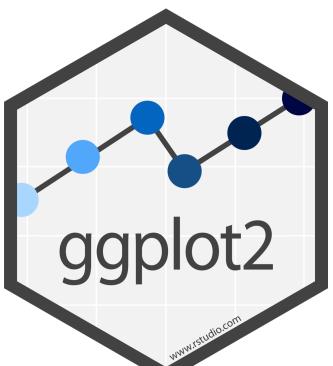
How are these plots different?

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



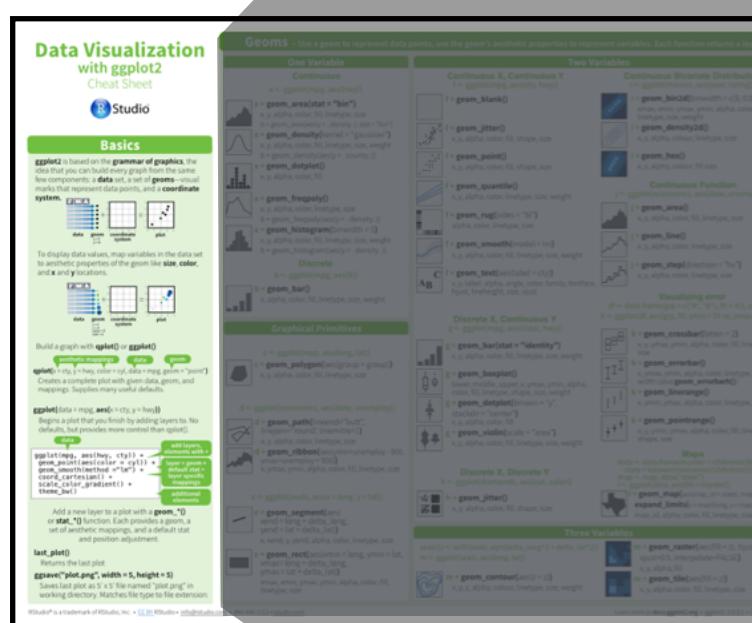
geoms

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



geom_ functions

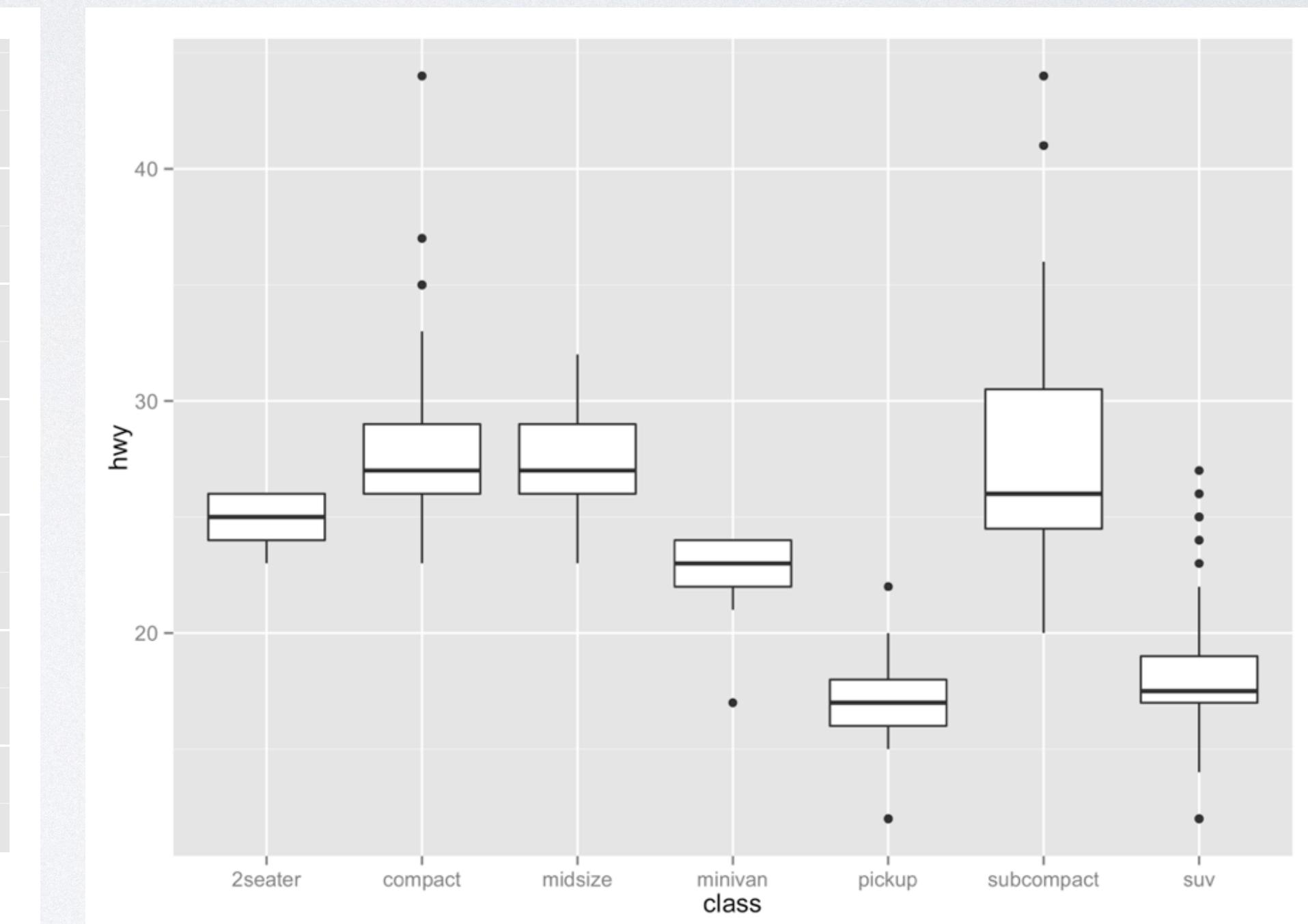
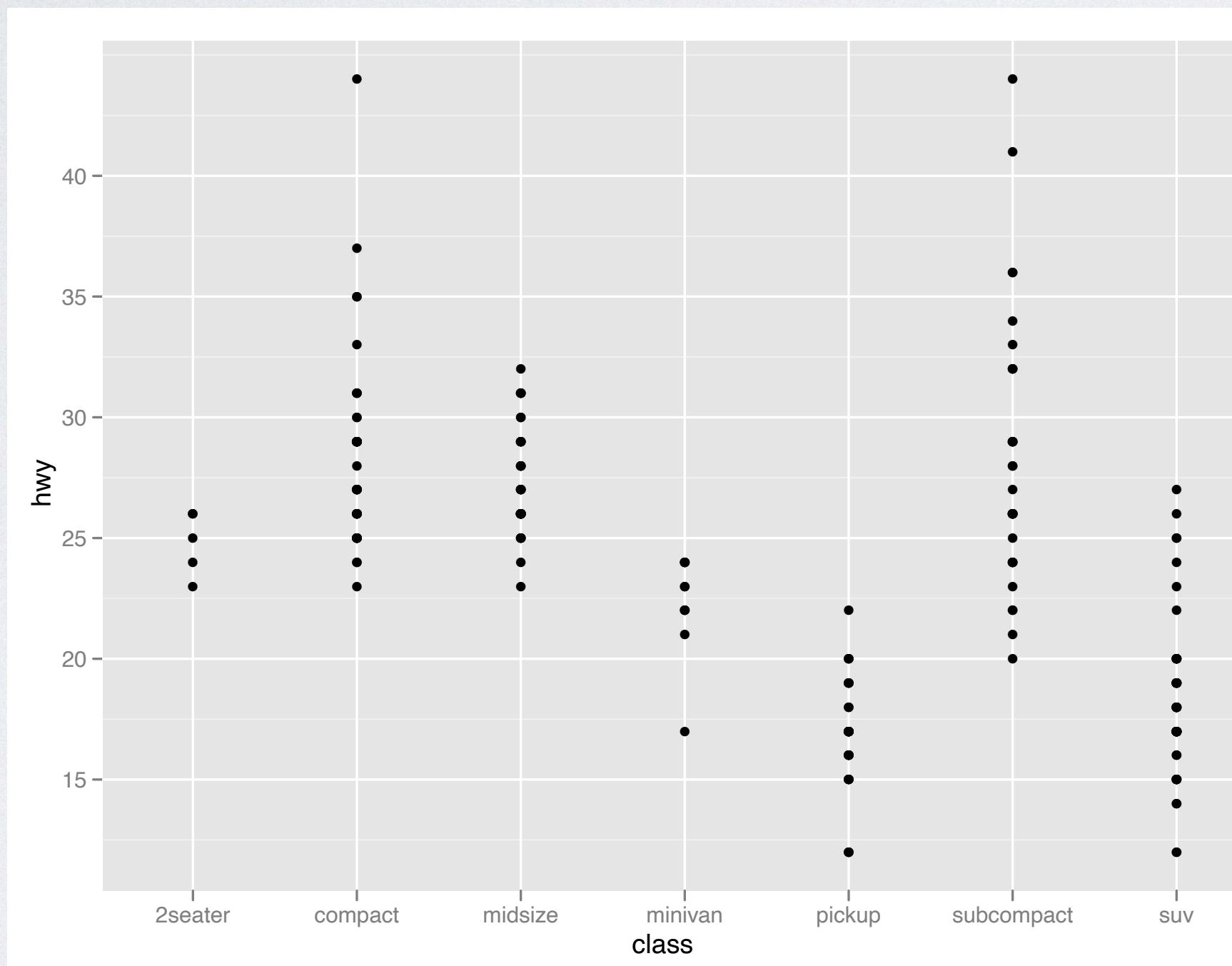
Each requires a mapping argument.



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 <pre>a <- ggplot(mpg, aes(hwy))</pre> <pre>a + geom_area(stat = "bin")</pre> <pre>a + geom_density(kernel = "gaussian")</pre> <pre>a + geom_dotplot()</pre> <pre>a + geom_freqpoly()</pre> <pre>a + geom_histogram(binwidth = 5)</pre> Discrete <pre>b <- ggplot(mpg, aes(fl))</pre> <pre>b + geom_bar()</pre> 	Two Variables <ul style="list-style-type: none"> Continuous X, Continuous Y <pre>f <- ggplot(mpg, aes(cty, hwy))</pre> <pre>f + geom_blank()</pre> <pre>f + geom_jitter()</pre> Continuous Function <pre>j <- ggplot(economics, aes(date, unemploy))</pre> <pre>j + geom_area()</pre> <pre>j + geom_line()</pre> <pre>j + geom_step(direction = "hv")</pre>
Graphical Primitives <ul style="list-style-type: none"> Continuous Y <pre>map <- map_data("state")</pre> <pre>c <- ggplot(map, aes(long, lat))</pre> <pre>c + geom_polygon(aes(group = group))</pre> Continuous X, Continuous Y <pre>g <- ggplot(mpg, aes(class, hwy))</pre> <pre>g + geom_bar(stat = "identity")</pre> <pre>g + geom_boxplot()</pre> <pre>g + geom_dotplot(binaxis = "y", stackdir = "center")</pre> <pre>g + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))</pre> <pre>g + geom_path(lineend = "butt", linejoin = "round", linemetre = 1)</pre> <pre>g + geom_rect(aes(xmin = long + delta_long, xmax = long + delta_long, ymin = lat + delta_lat, ymax = lat + delta_lat))</pre> Discrete X, Discrete Y <pre>h <- ggplot(diamonds, aes(cut, color))</pre> <pre>h + geom_jitter()</pre> Three Variables <pre>seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))</pre> <pre>m <- ggplot(seals, aes(long, lat))</pre> 	Continuous Bivariate Distribution <ul style="list-style-type: none"> Continuous X, Continuous Y <pre>i <- ggplot(movies, aes(year, rating))</pre> <pre>i + geom_hex()</pre> Continuous Function <pre>j <- ggplot(economics, aes(date, unemploy))</pre> <pre>j + geom_area()</pre> <pre>j + geom_line()</pre> <pre>j + geom_step(direction = "hv")</pre> Visualizing error <pre>df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)</pre> <pre>k <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))</pre> <ul style="list-style-type: none"> Continuous X, Continuous Y <pre>g + geom_crossbar(fatten = 2)</pre> Continuous X, Continuous Y <pre>g + geom_errorbar()</pre> Continuous X, Continuous Y <pre>g + geom_linerange()</pre> Continuous X, Continuous Y <pre>g + geom_pointrange()</pre>
Three Variables <ul style="list-style-type: none"> Continuous X, Continuous Y <pre>m + geom_raster(aes(fill = z), hijst = 0.5, vjust = 0.5, interpolate = FALSE)</pre> <pre>m + geom_contour(aes(z = z))</pre> Continuous X, Continuous Y <pre>m + geom_tile(aes(fill = z))</pre> 	

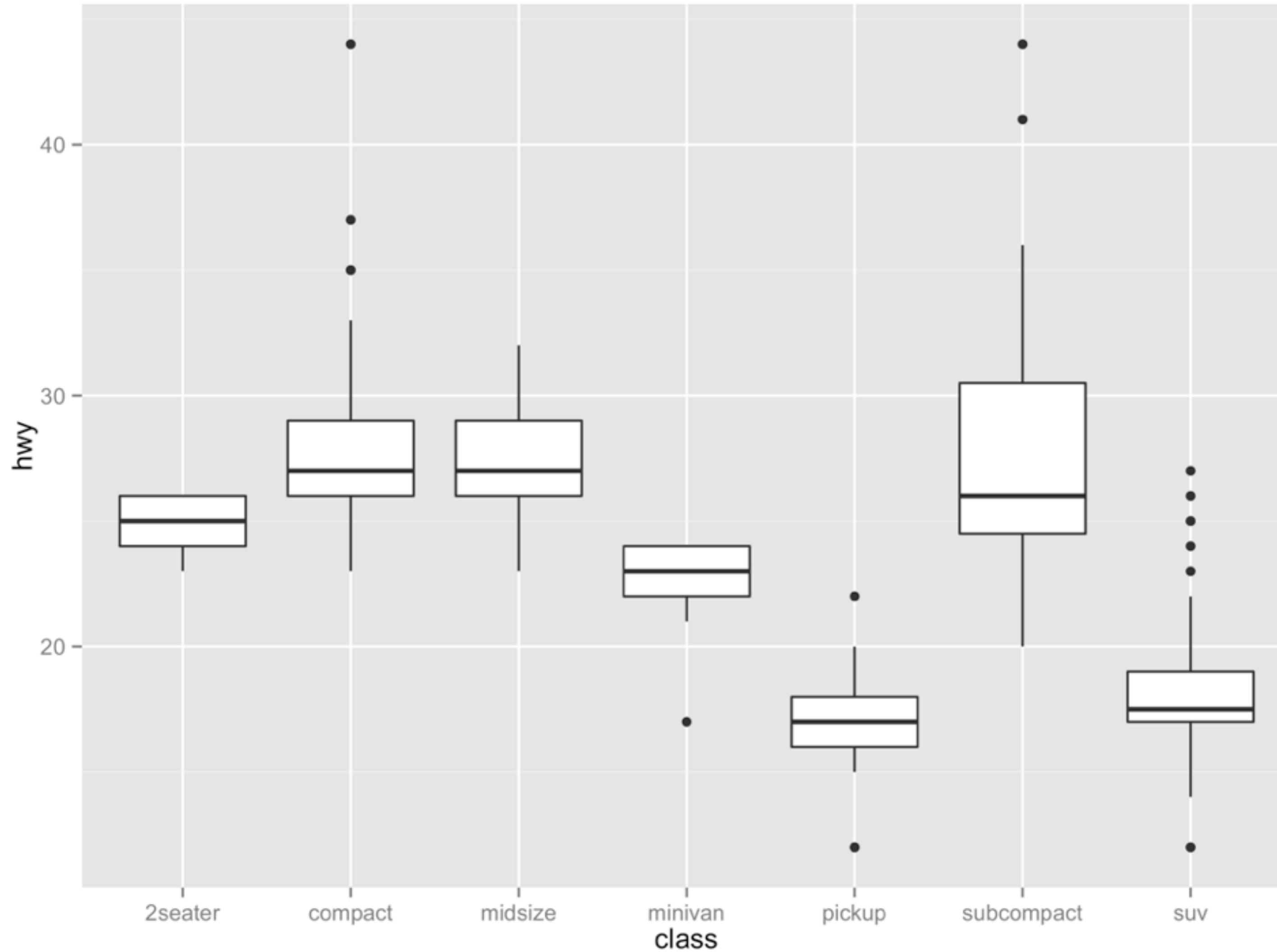
Your Turn 3

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

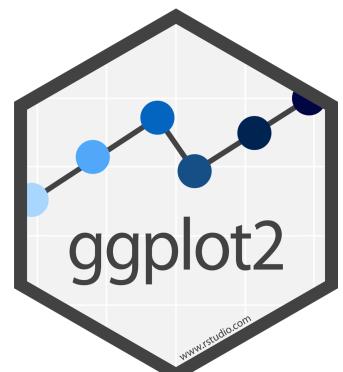


```
ggplot(mpg) + geom_point(aes(class, hwy))
```

03 : 00

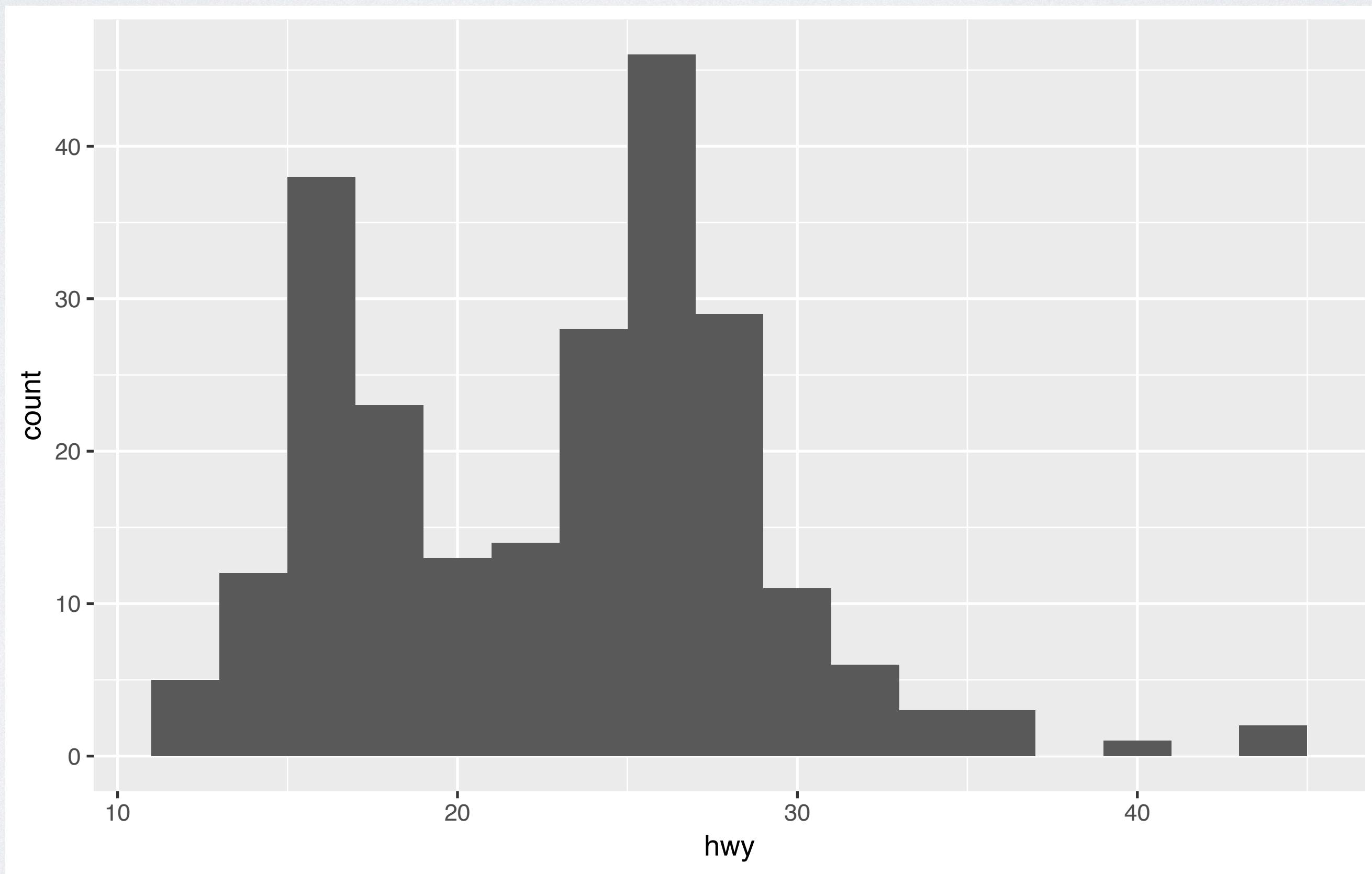


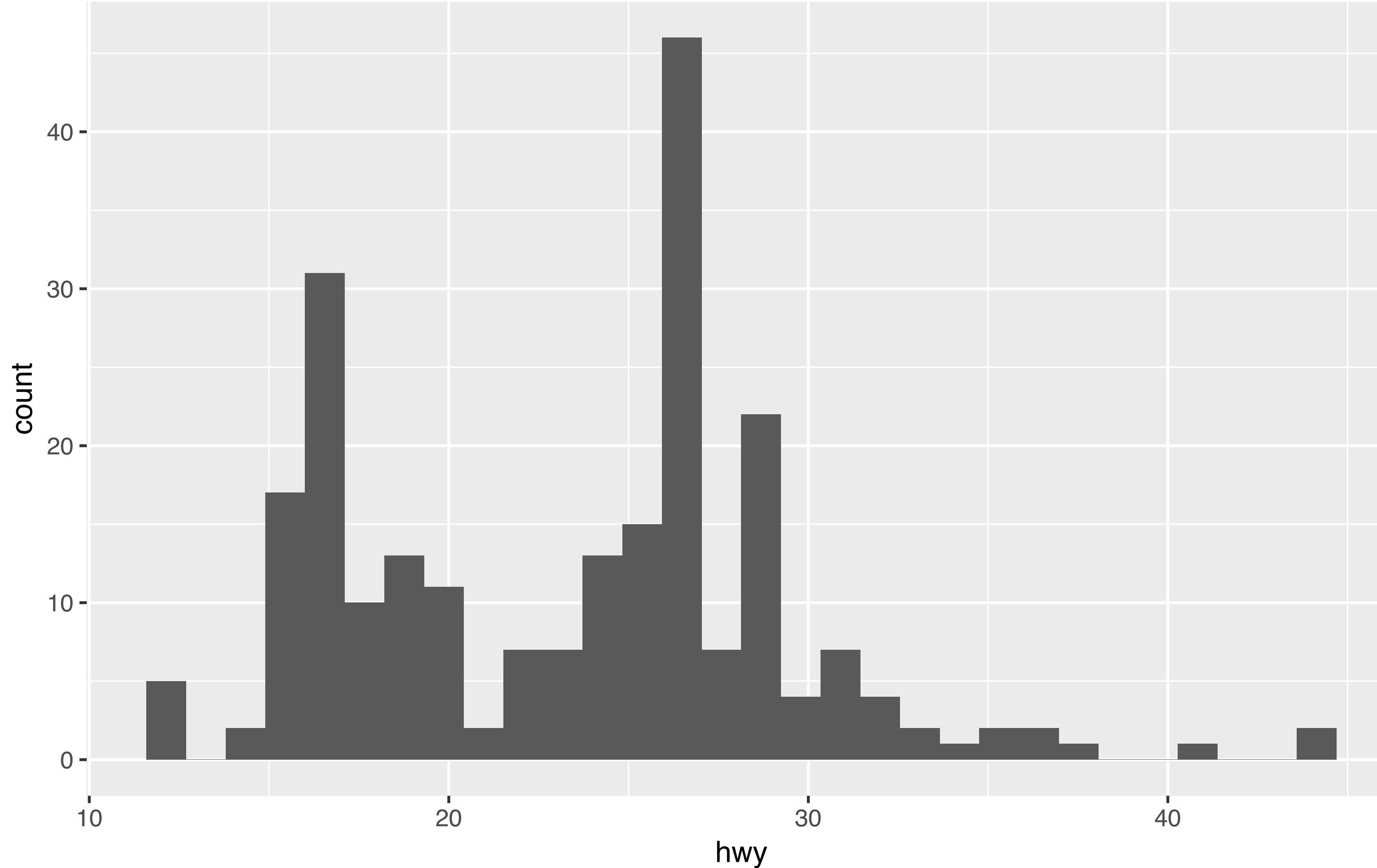
```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = class, y = hwy))
```



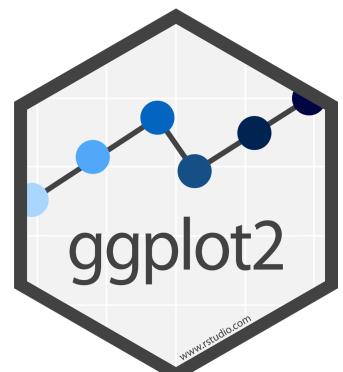
Your Turn 4

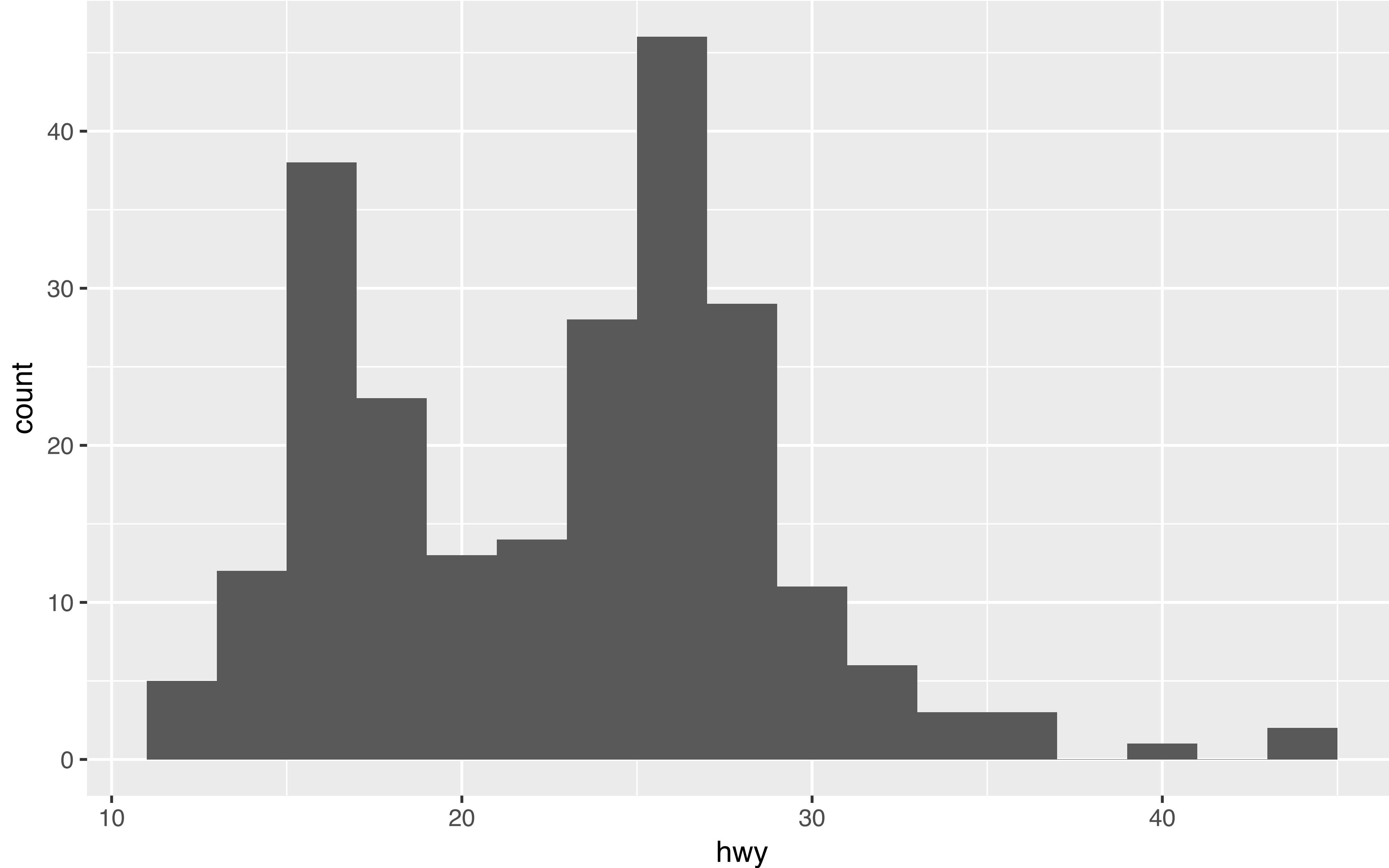
With your partner, make the histogram of `hwy` below. Use the cheatsheet. Hint: do not supply a `y` variable.



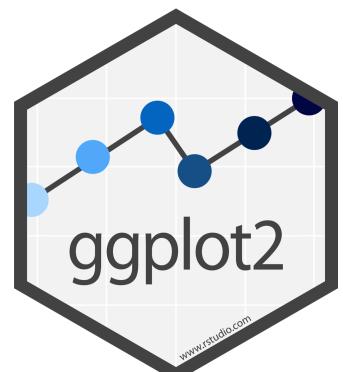


```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy))
```



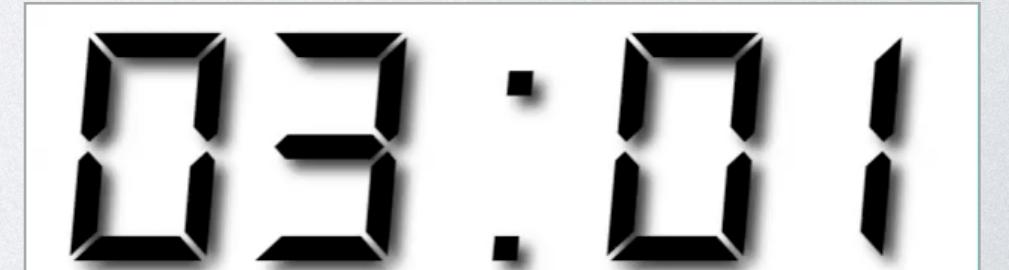
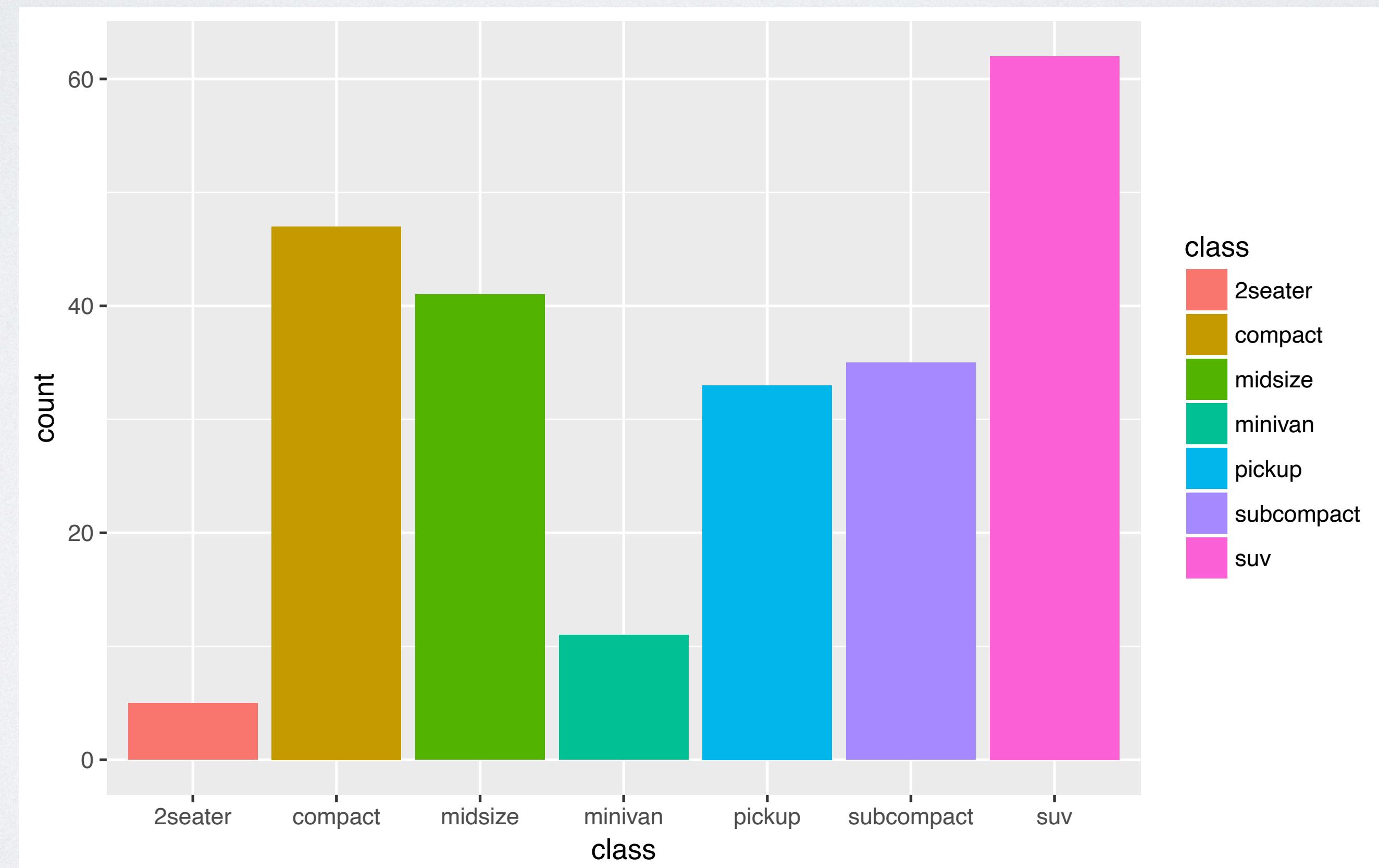


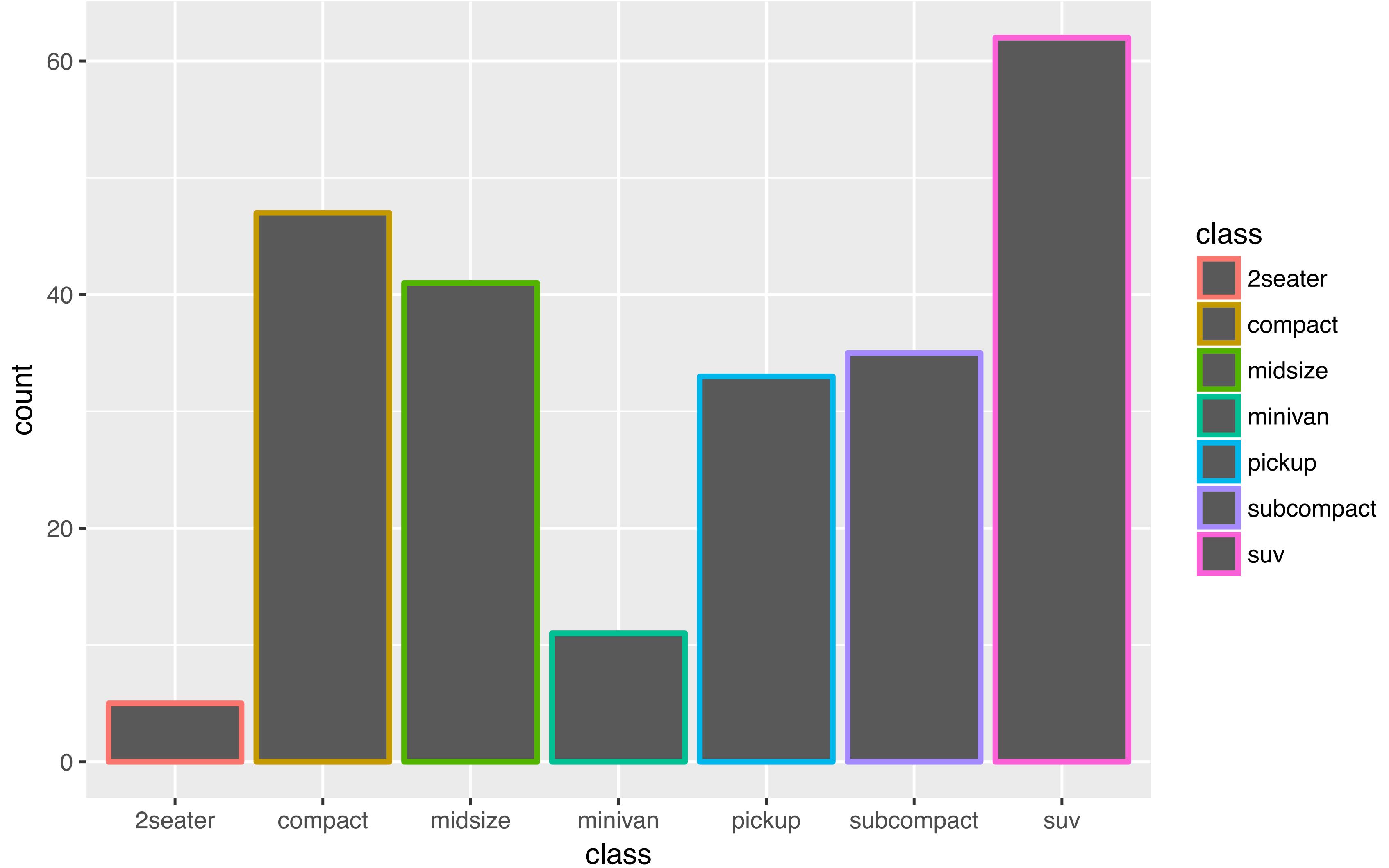
```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy), binwidth = 2)
```



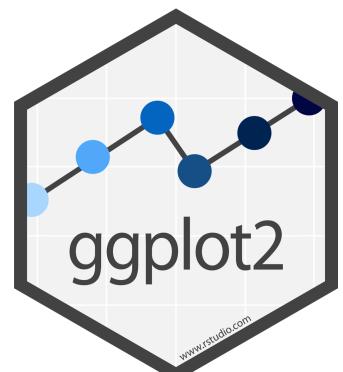
Your Turn 5

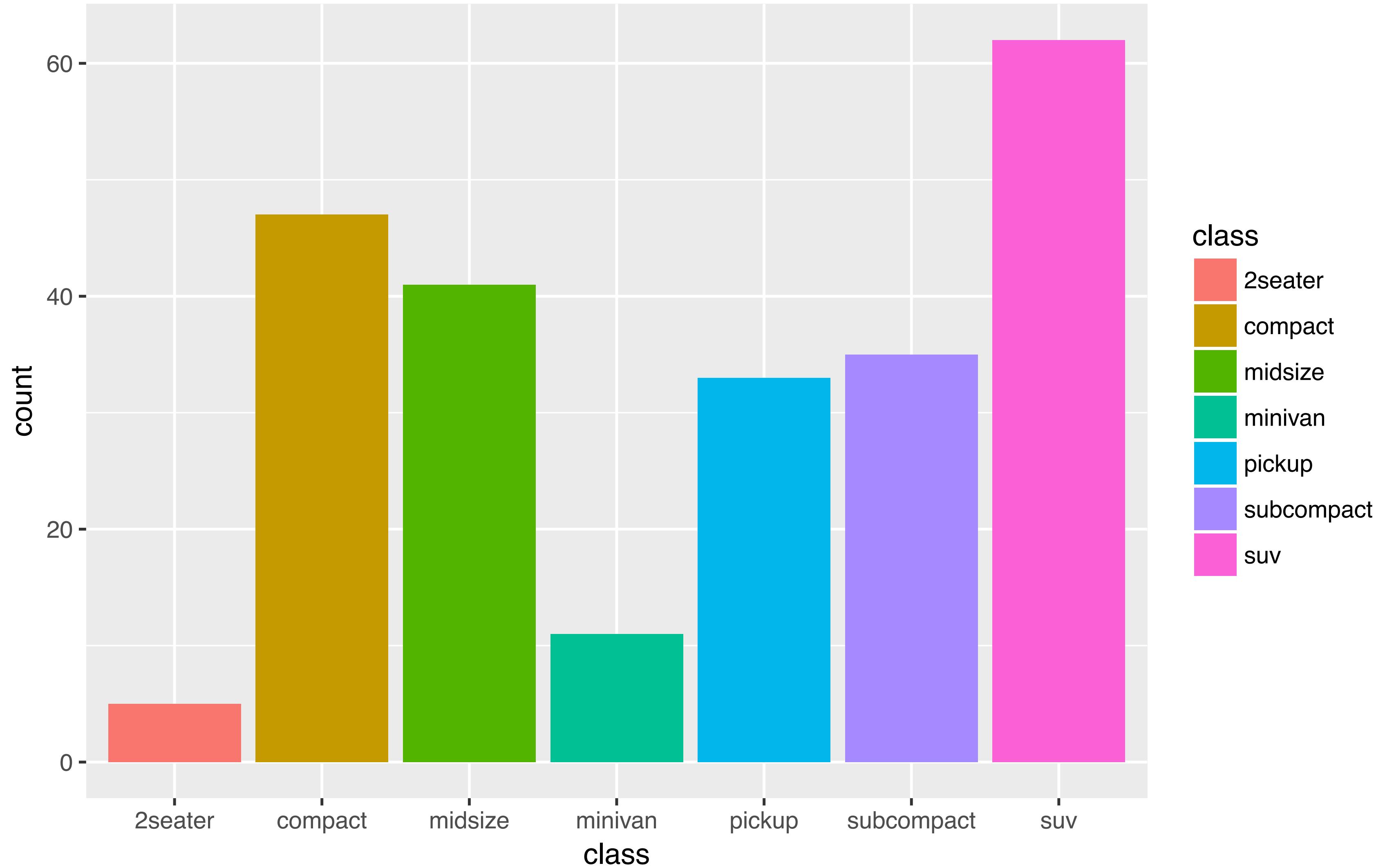
With your partner, make the bar chart of **class** colored by **class** below. Use the cheatsheet. Try your best guess.



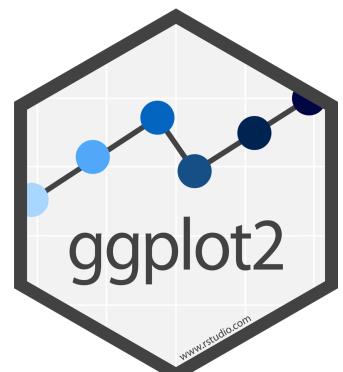


```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = class, color = class))
```





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



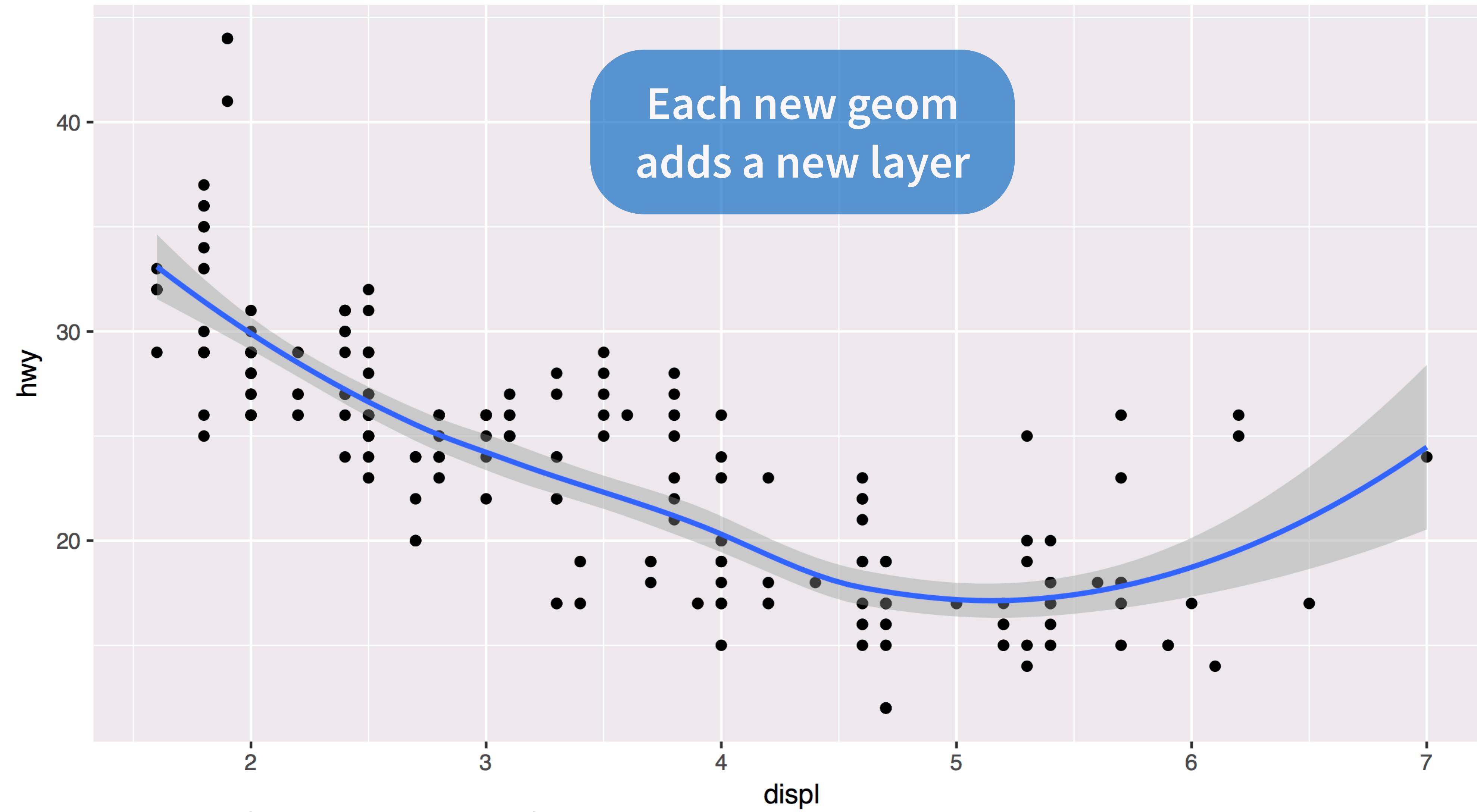
Your Turn 6

With your partner, predict what this code will do.

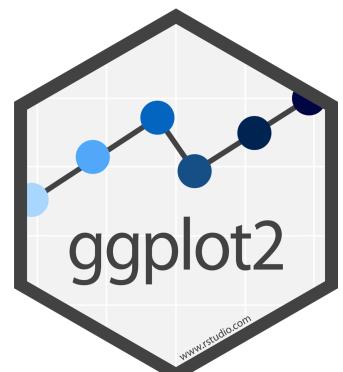
Then run it.

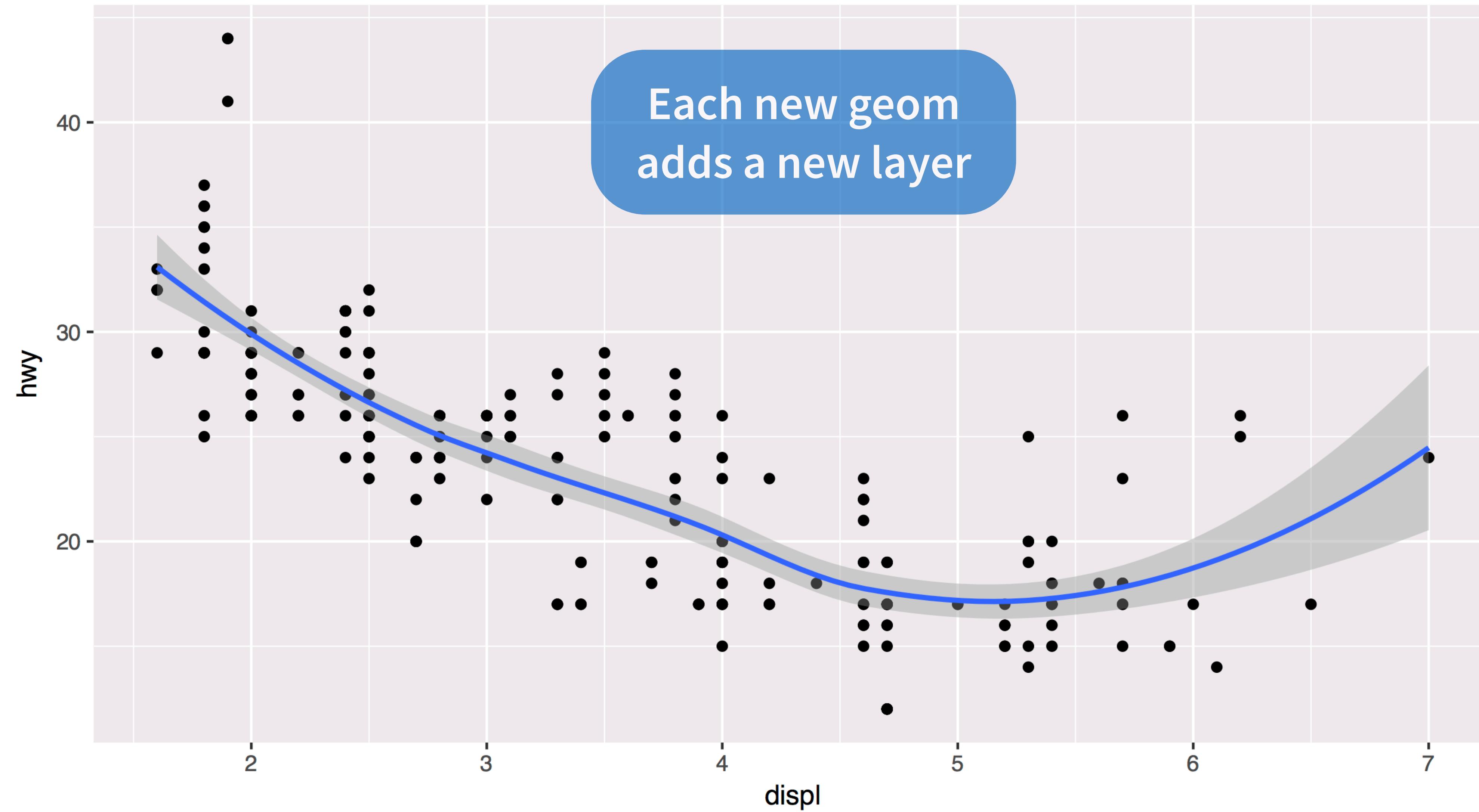
```
ggplot(mpg) +  
  geom_point(aes(displ, hwy)) +  
  geom_smooth(aes(displ, hwy))
```



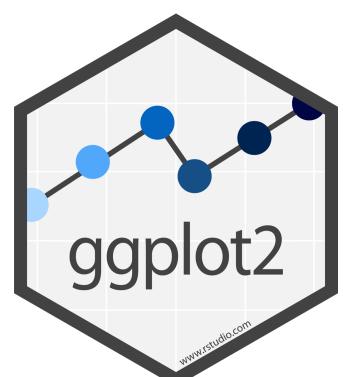


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



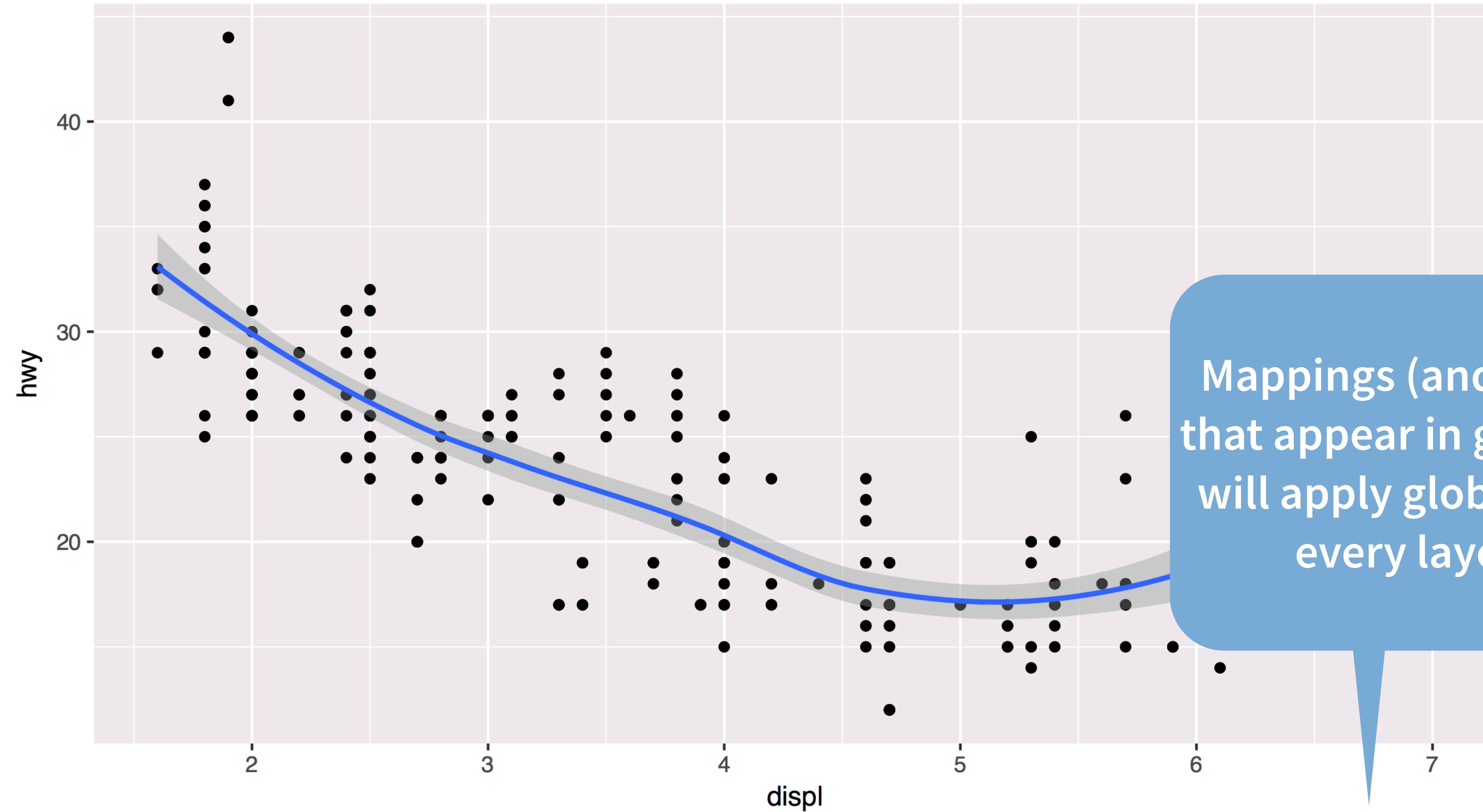


```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

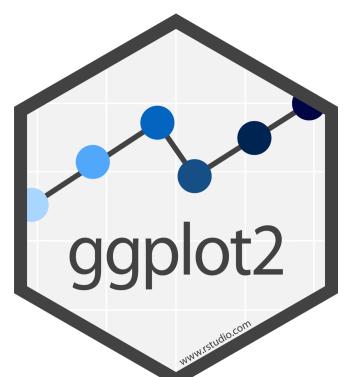


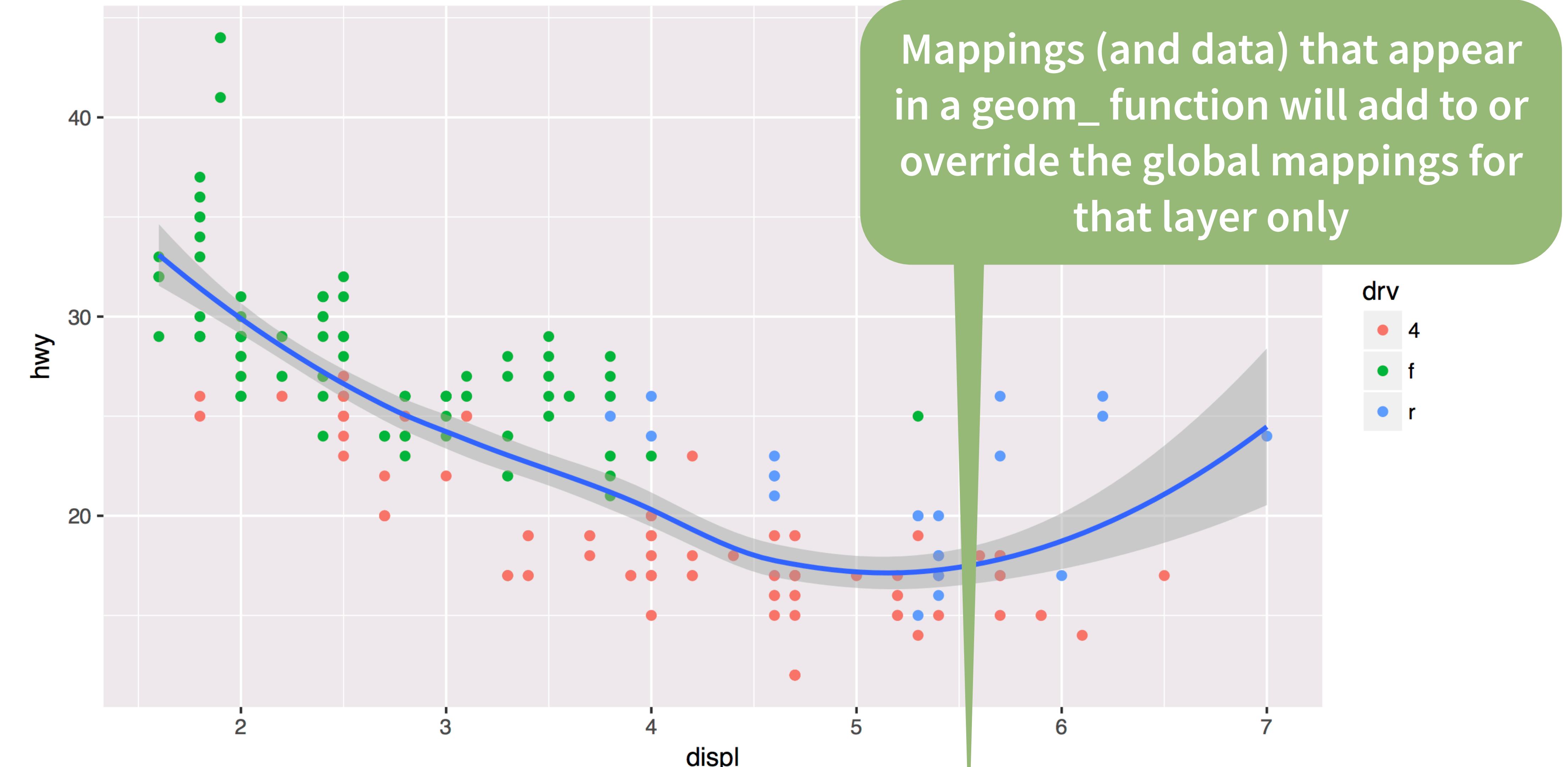
global vs. local

R

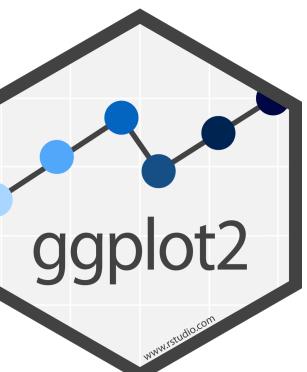


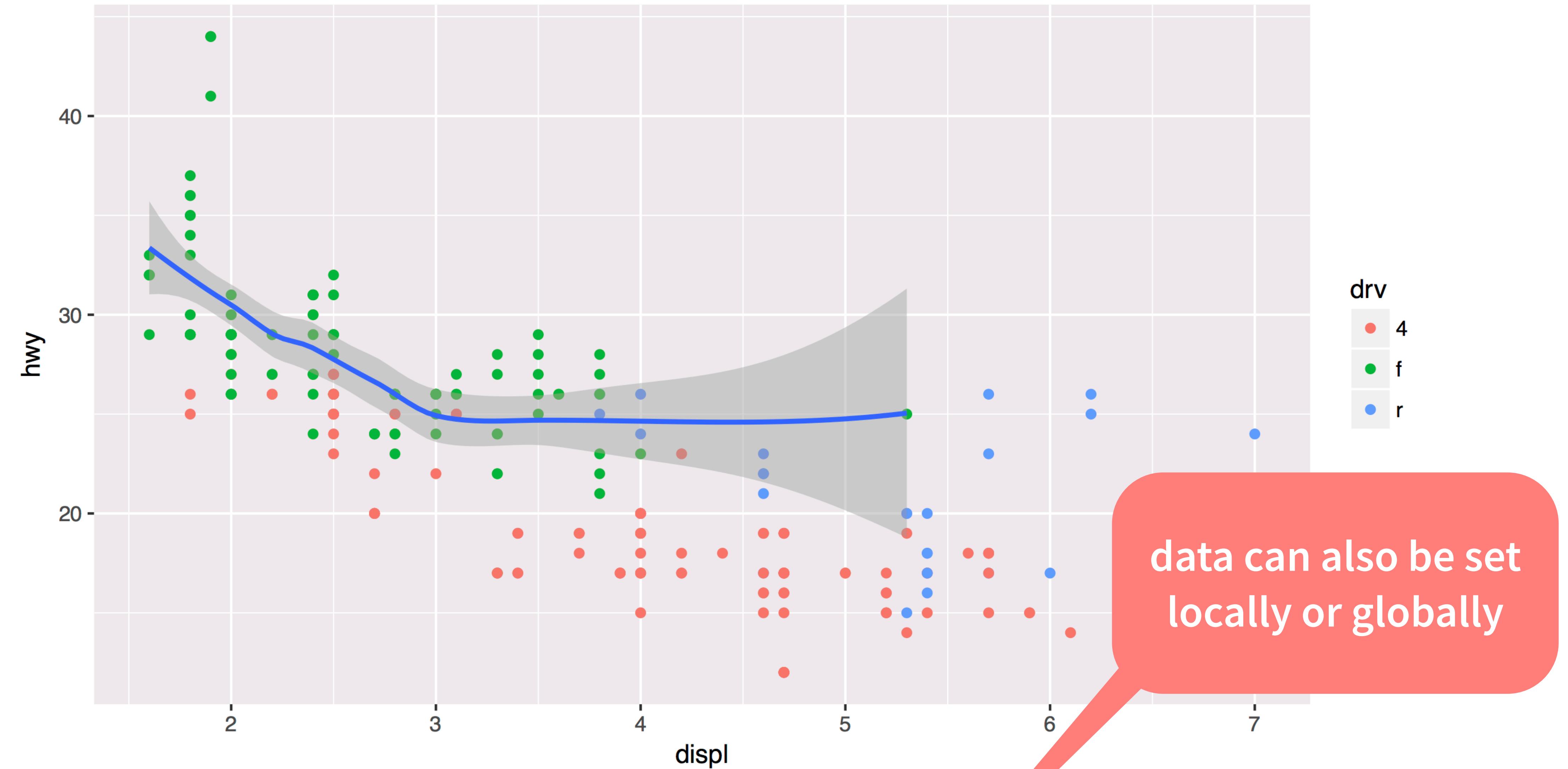
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth()
```



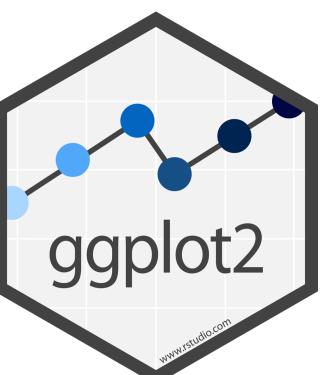


```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth()
```





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

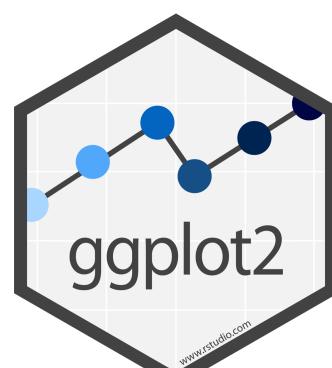
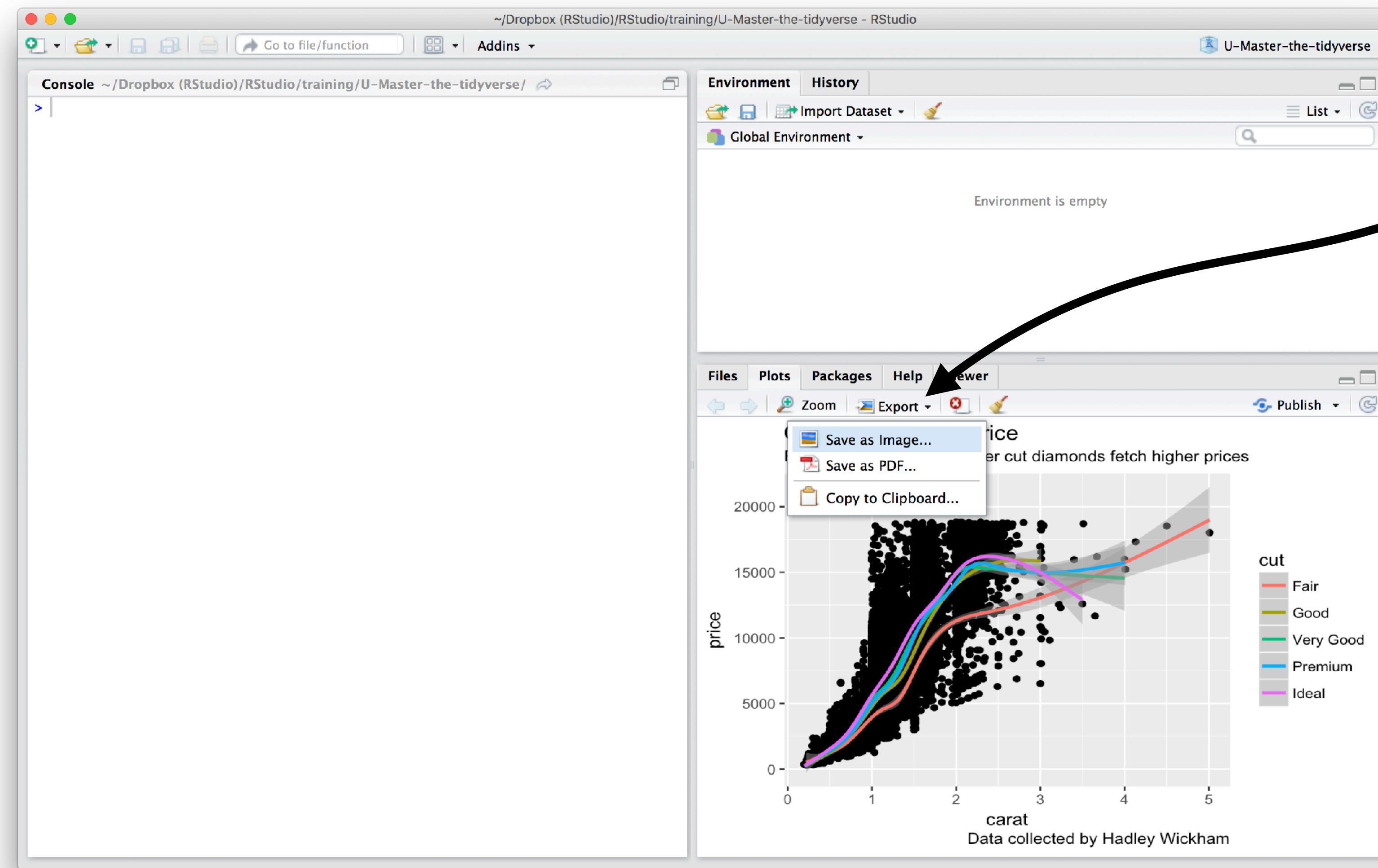


Saving graphs



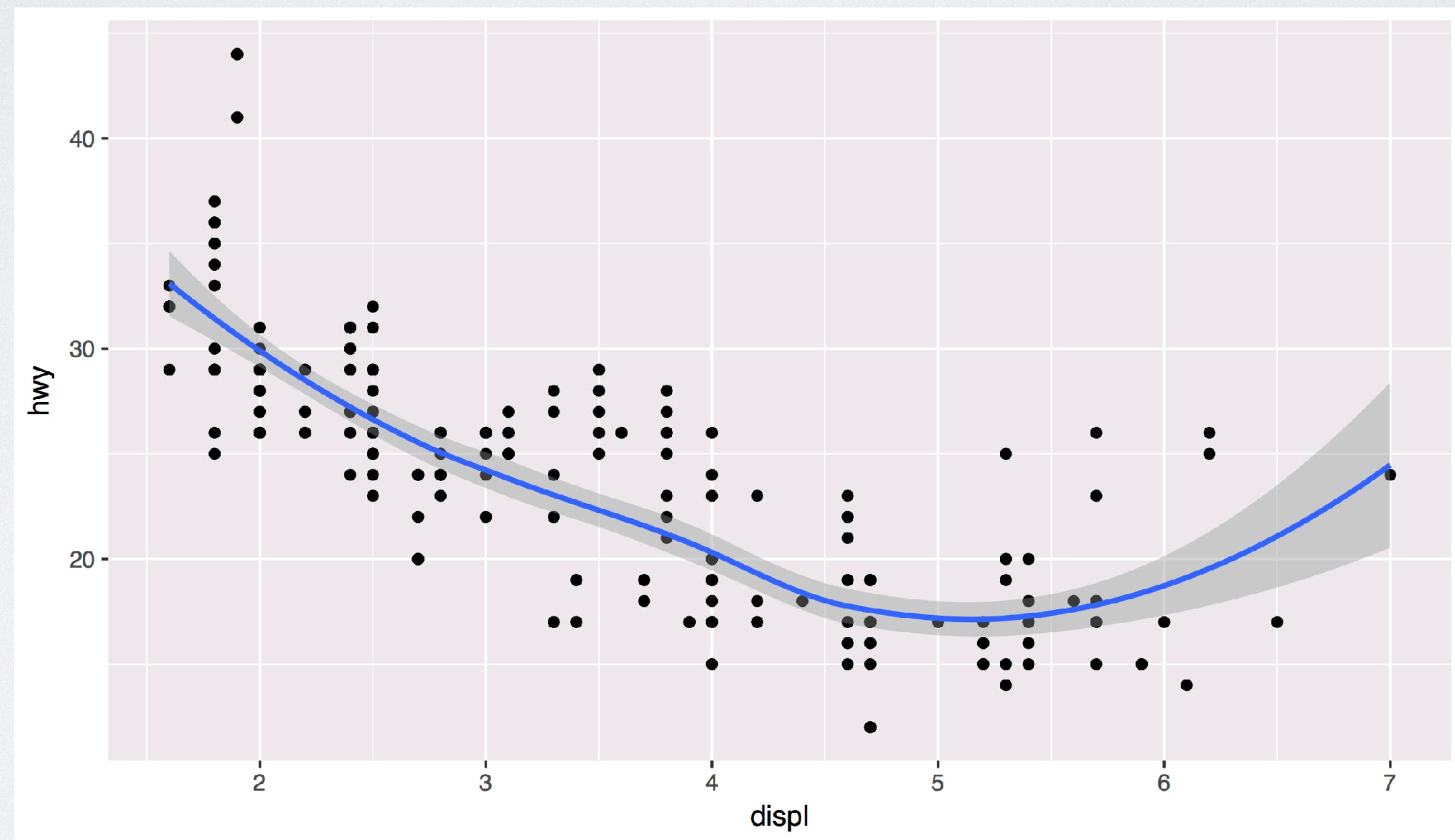
Manually saving plots

Save plots manually with the export menu



Your Turn 7

Run your code from the last exercise in the Console. Click "Export" in the window that it appears in, and then save the graph!



Grammar of Graphics



amazon

All grammar of graphics

Hello, Sign in Account & Lists Returns & Orders

Hello Select your address Holiday Deals Gift Cards Best Sellers Customer Service Find a Gift New Releases Whole Foods AmazonBasics Free Shipping Sell Registry Coupons

Books Advanced Search New Releases Best Sellers & More Children's Books Textbooks Textbook Rentals Best Books of the Month

audible plus * Only \$4.95 a month for the first 6 months › audible

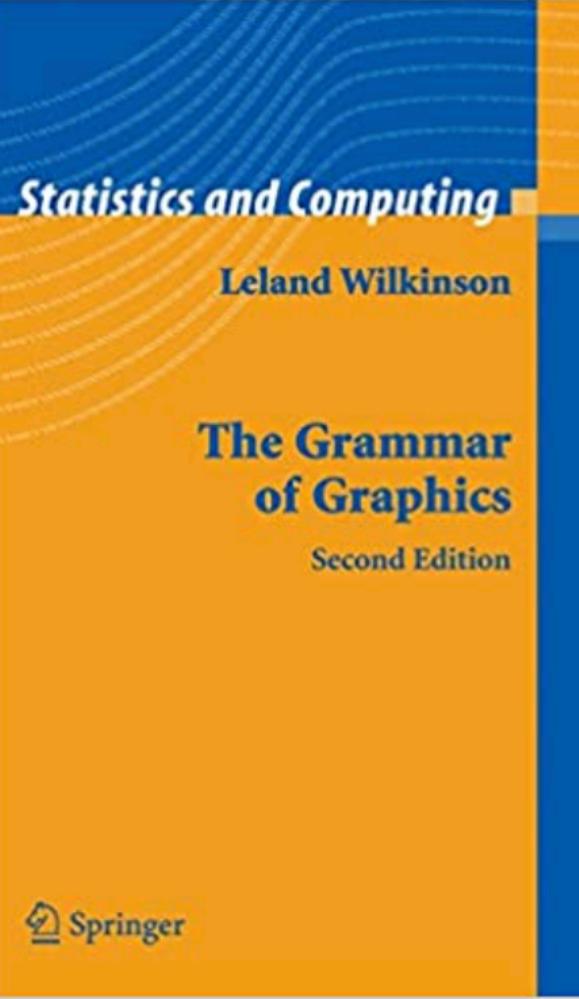
Back to results

The Grammar of Graphics (Statistics and Computing) 2nd Edition

by Leland Wilkinson ~ (Author), D. Wills (Contributor), D. Rope (Contributor), A. Norton (Contributor), R. Dubbs (Contributor)

★★★★★ 8 ratings

Look inside ↗



eTextbook \$36.57 - \$100.33

Hardcover \$87.43 - \$91.43

Paperback \$116.17 - \$121.76

Other Sellers See all 6 versions

Buy new:

Only 1 left in stock - order soon.

Sold by itemspopularonlineindemand and Fulfilled by Amazon.

Available at a lower price from other sellers that may not offer free Prime shipping.

Arrives: Wednesday, Nov 18 Details

Fastest delivery: Tomorrow

Order within 13 hrs and 56 mins Details

\$91.43

List Price: \$179.99
Save: \$88.56 (49%)

25 new from \$91.43 & FREE Shipping. Details

Select delivery location

prime Enjoy fast, FREE delivery, exclusive deals and award-winning movies & TV shows with Prime Try Prime and start saving today with Fast, FREE Delivery

Add to Cart

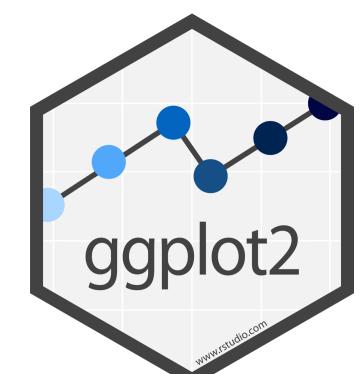
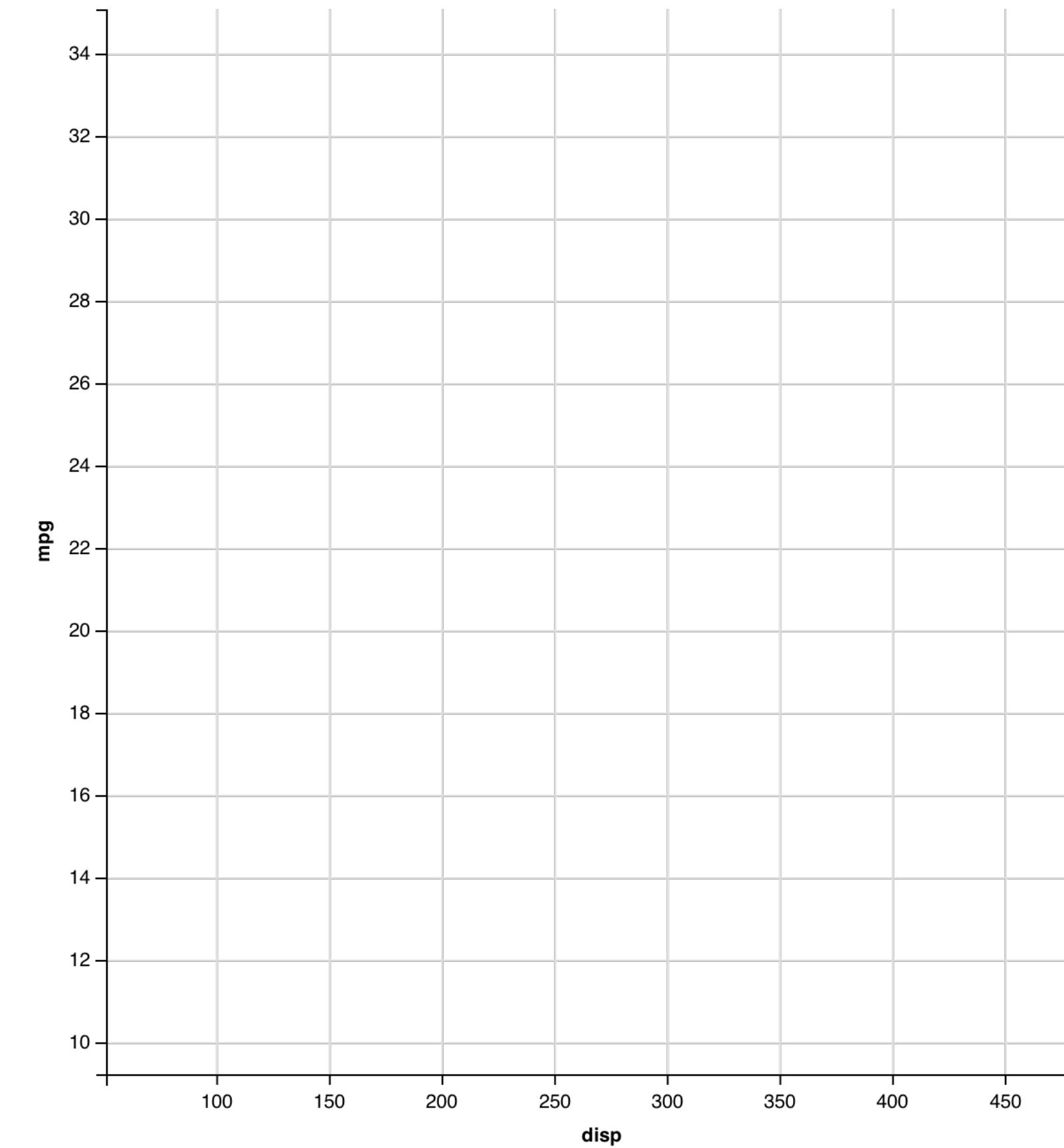
The Grammar of Graphics

Leland Wilkinson

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

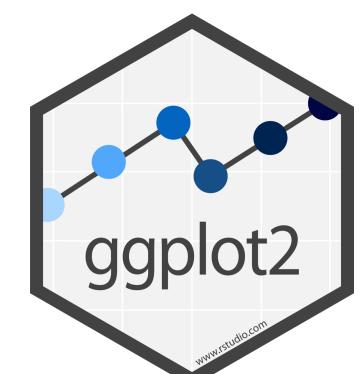
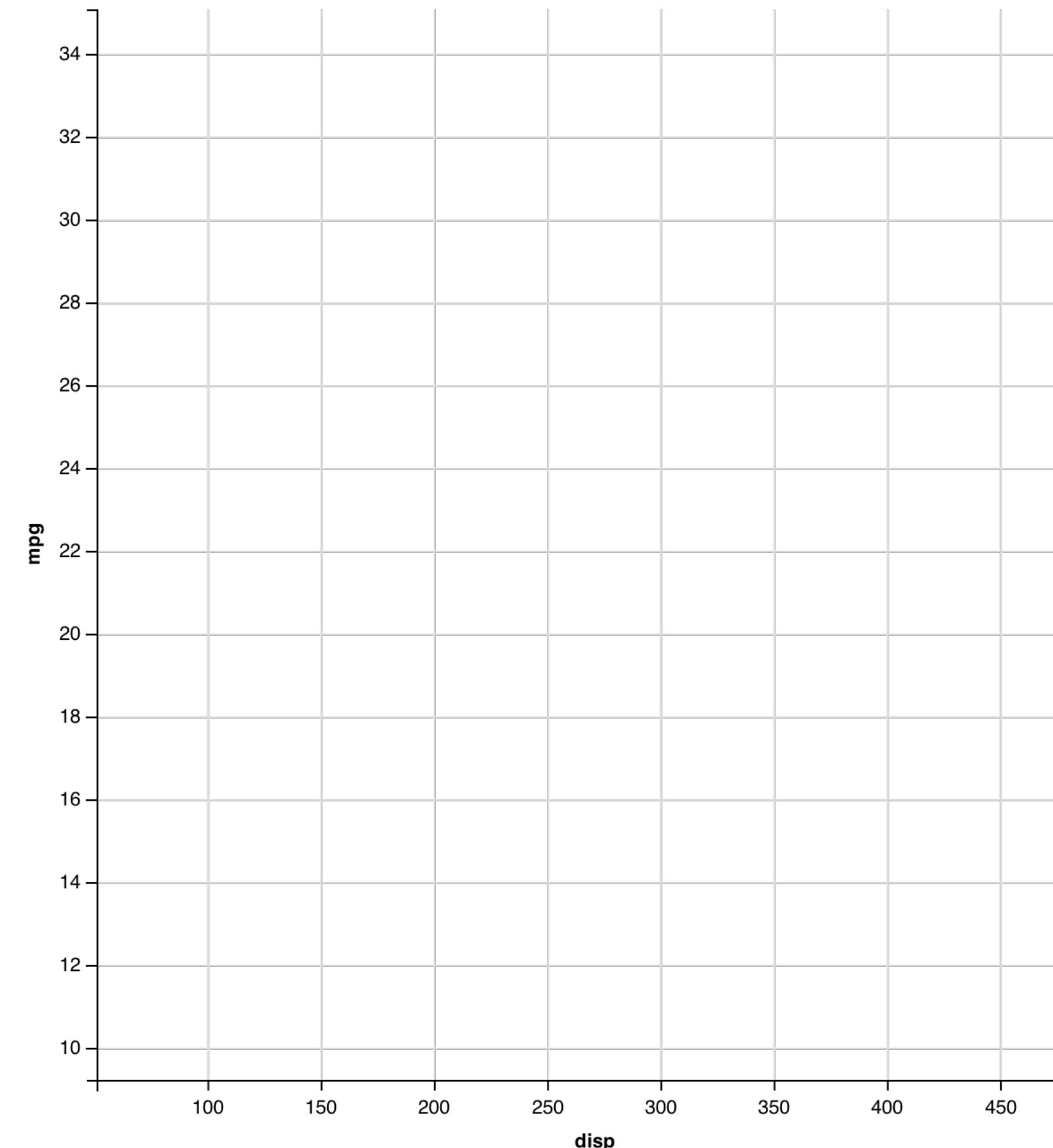
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

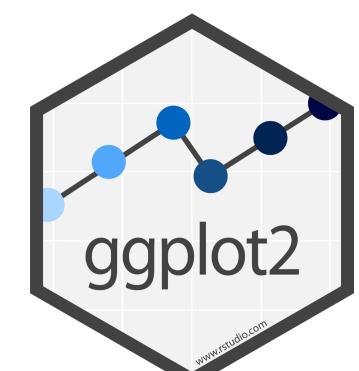
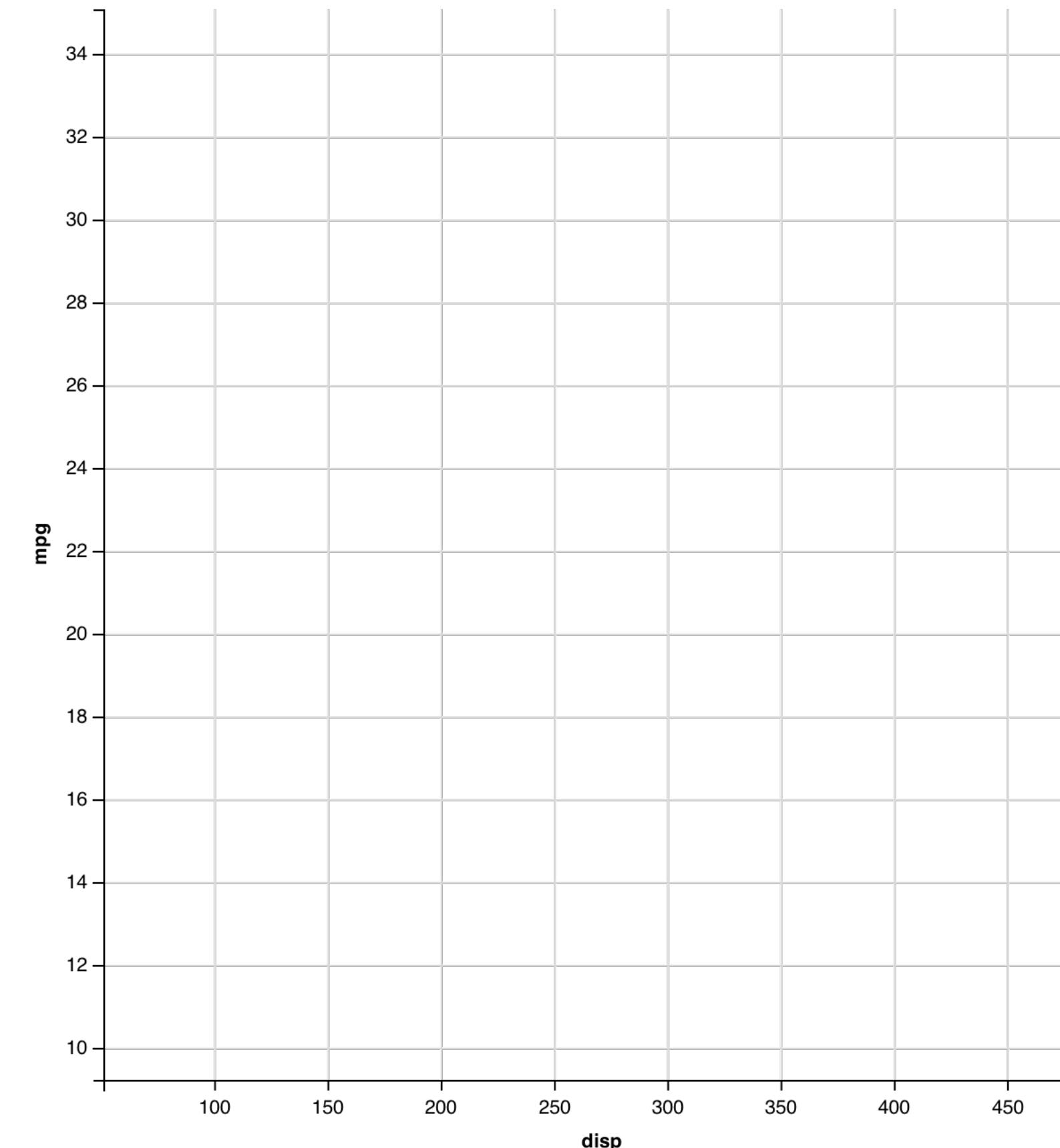


mappings

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

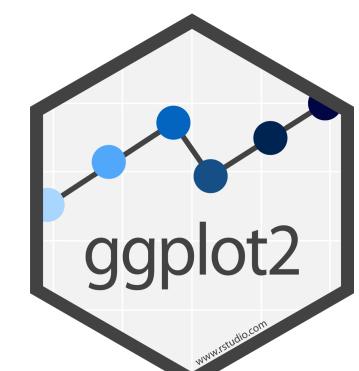
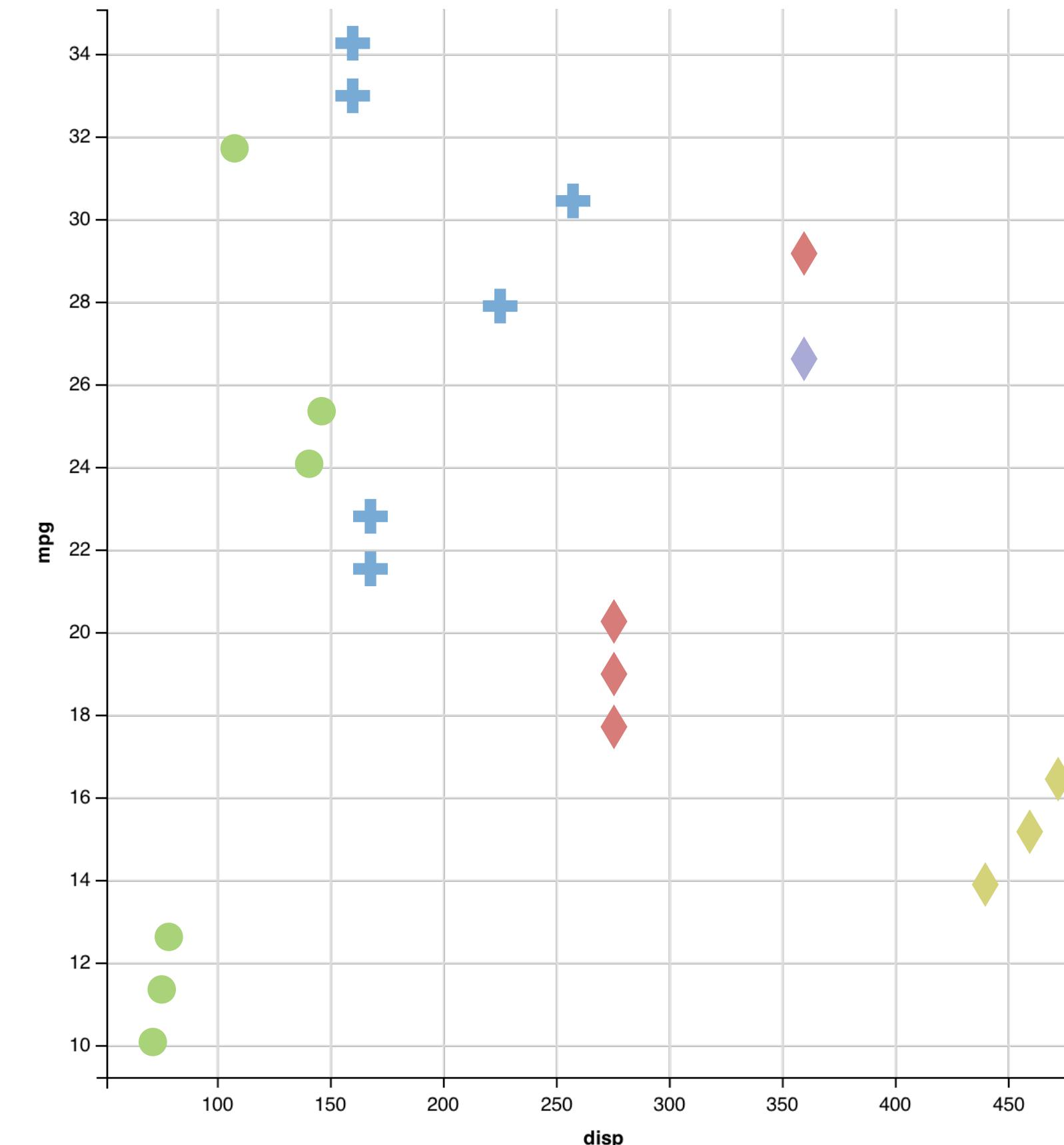


mappings

	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

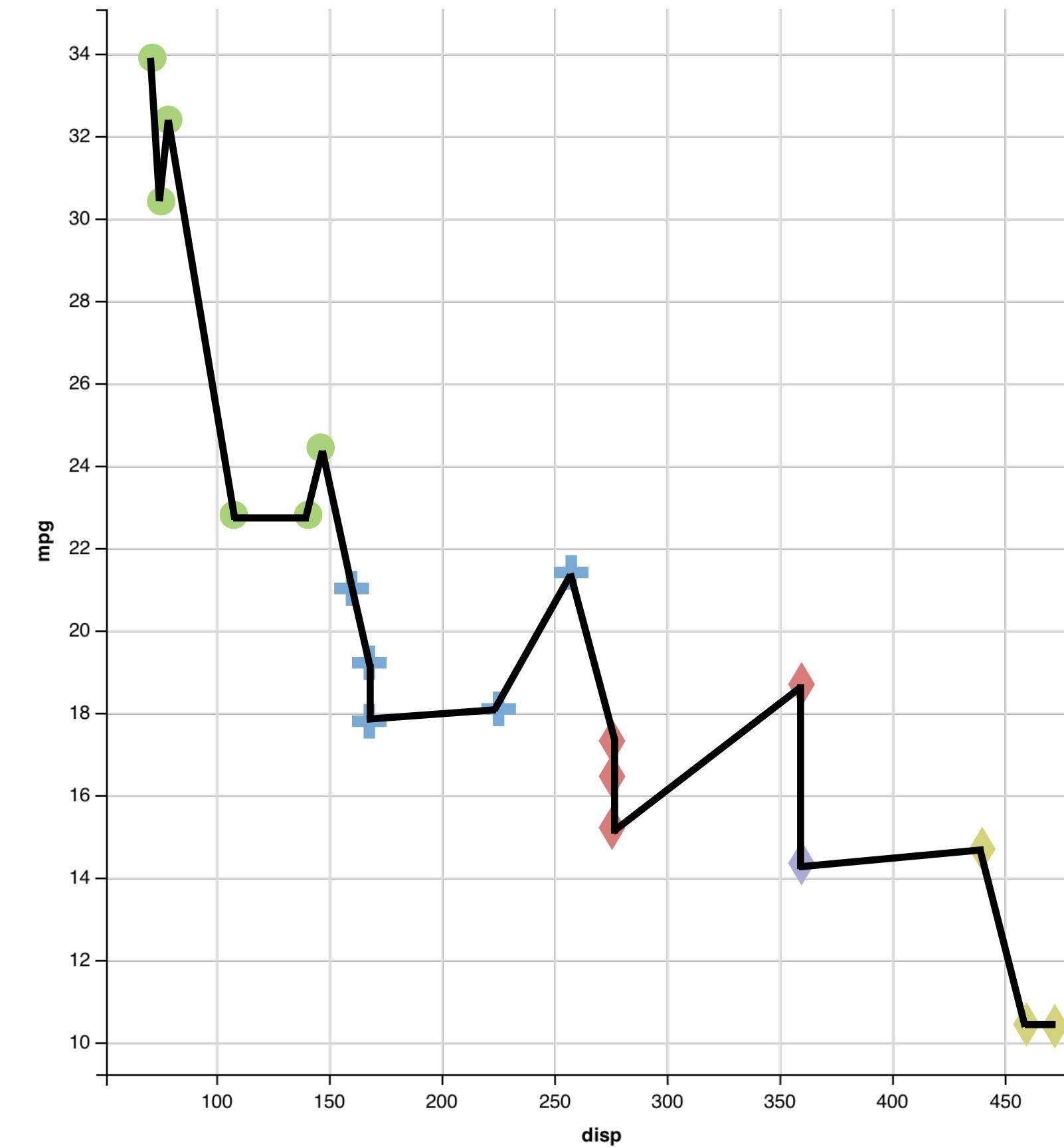


mappings

	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

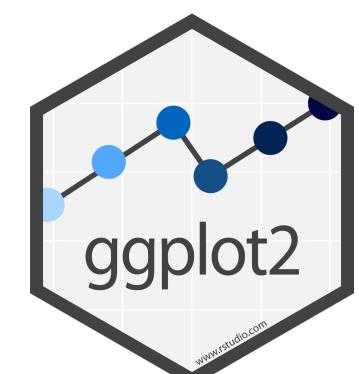
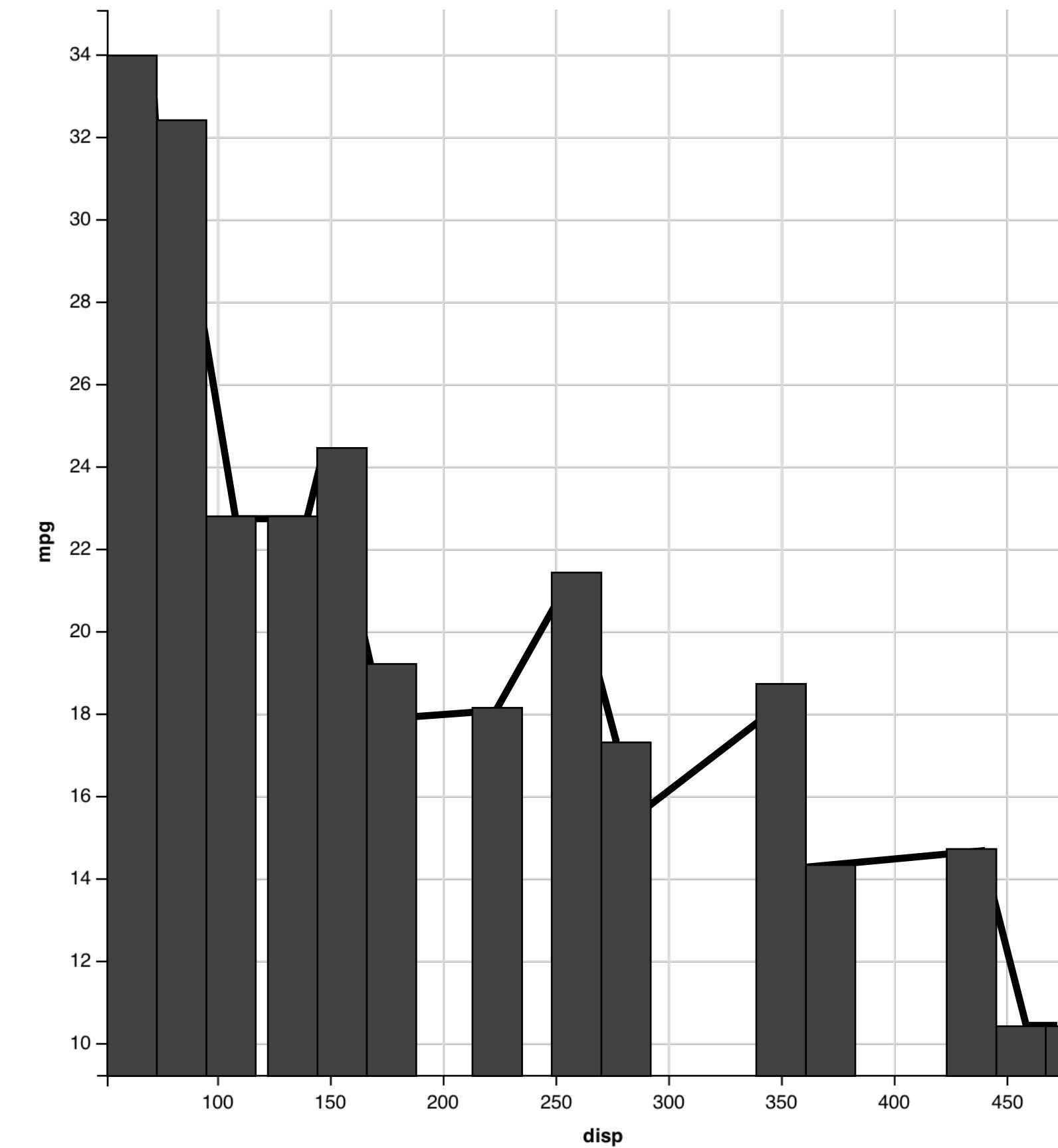


mappings

	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

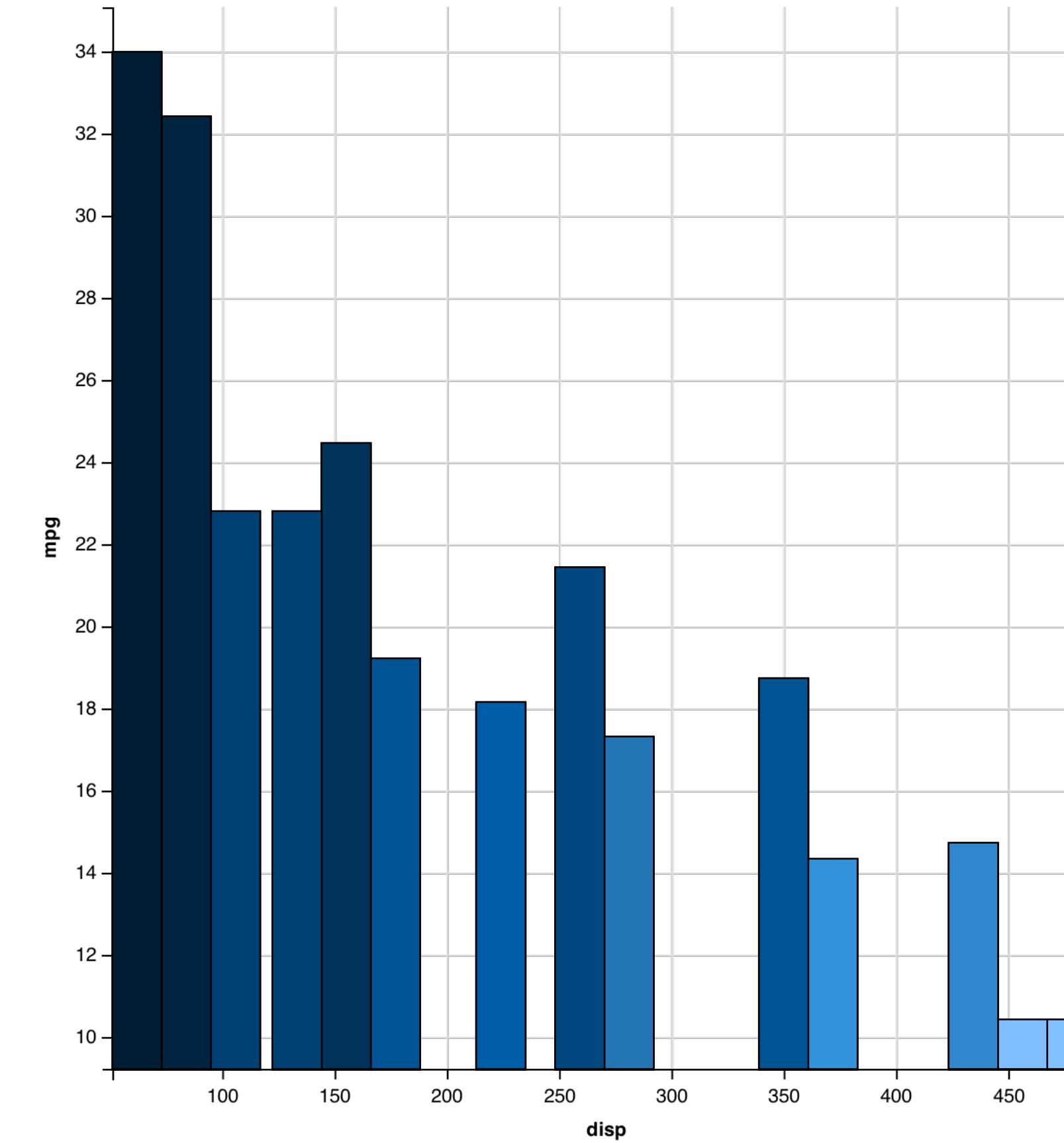
data

geom
points
lines
bars



mappings

	y		fill	
	mpg	cyl	disp	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

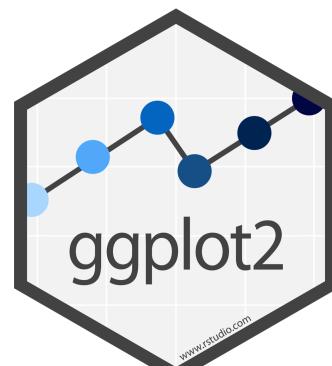
geom
points
lines
bars



To make a graph

[template]

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



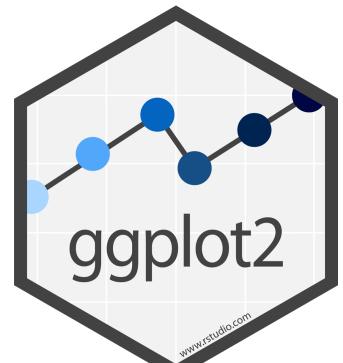
To make a graph

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	●

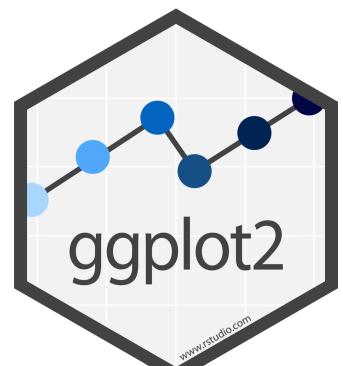
data

geom

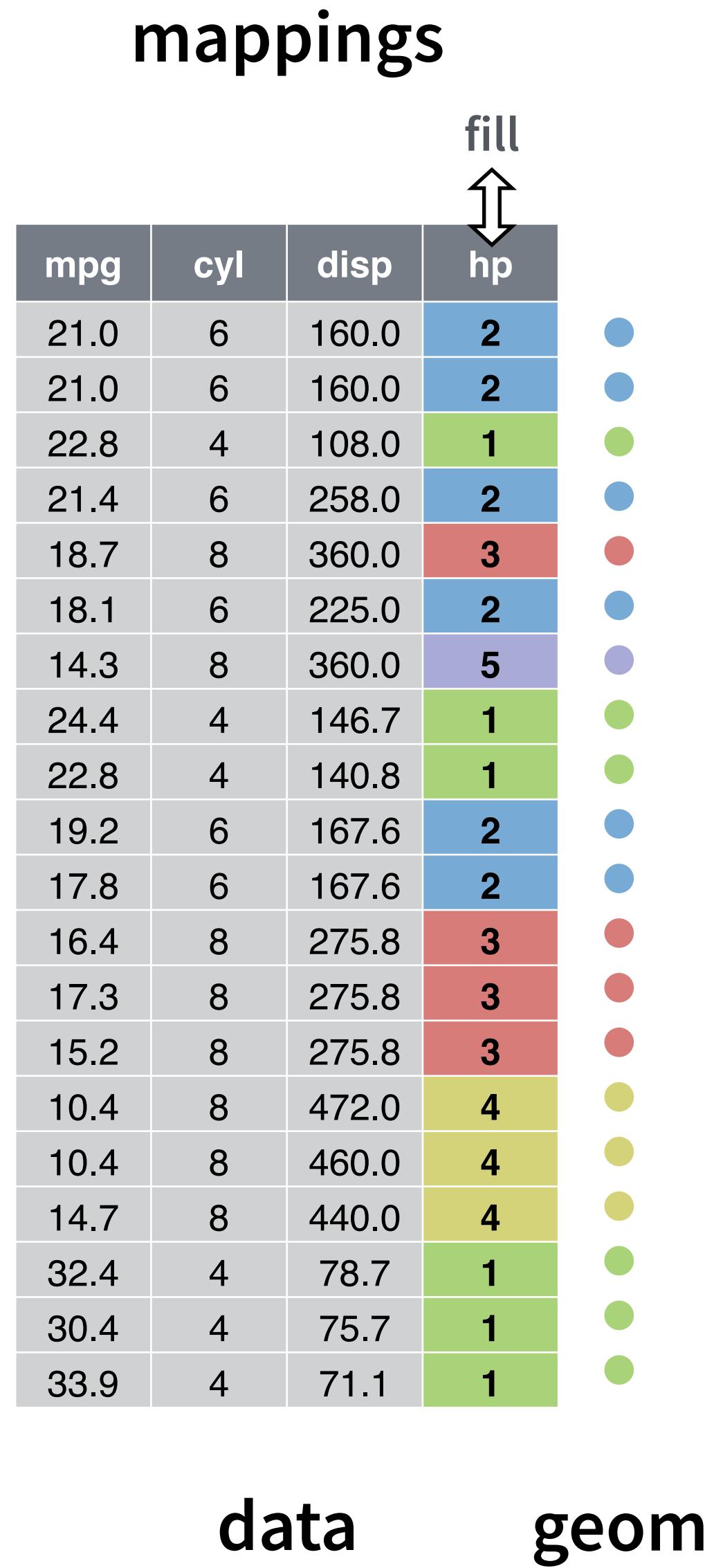
1. Pick a **data** set

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

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



To make a graph

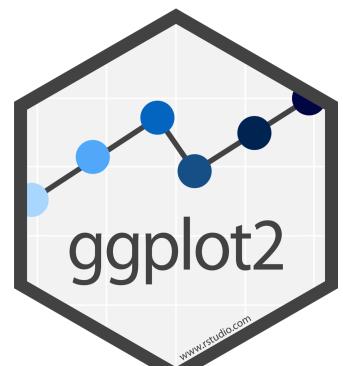


1. Pick a **data** set

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

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

3. **Map** aesthetic
properties to
variables

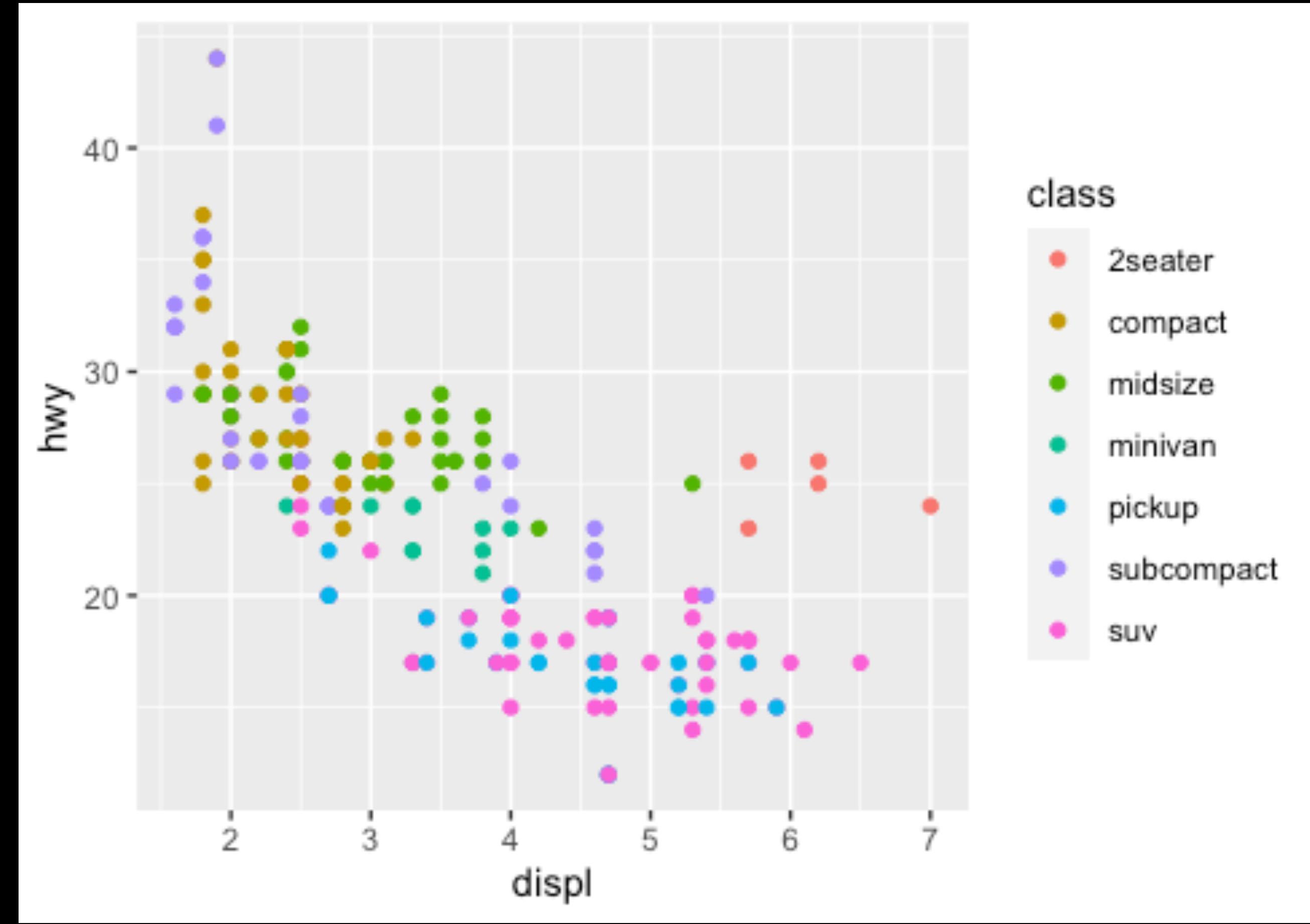


what else?

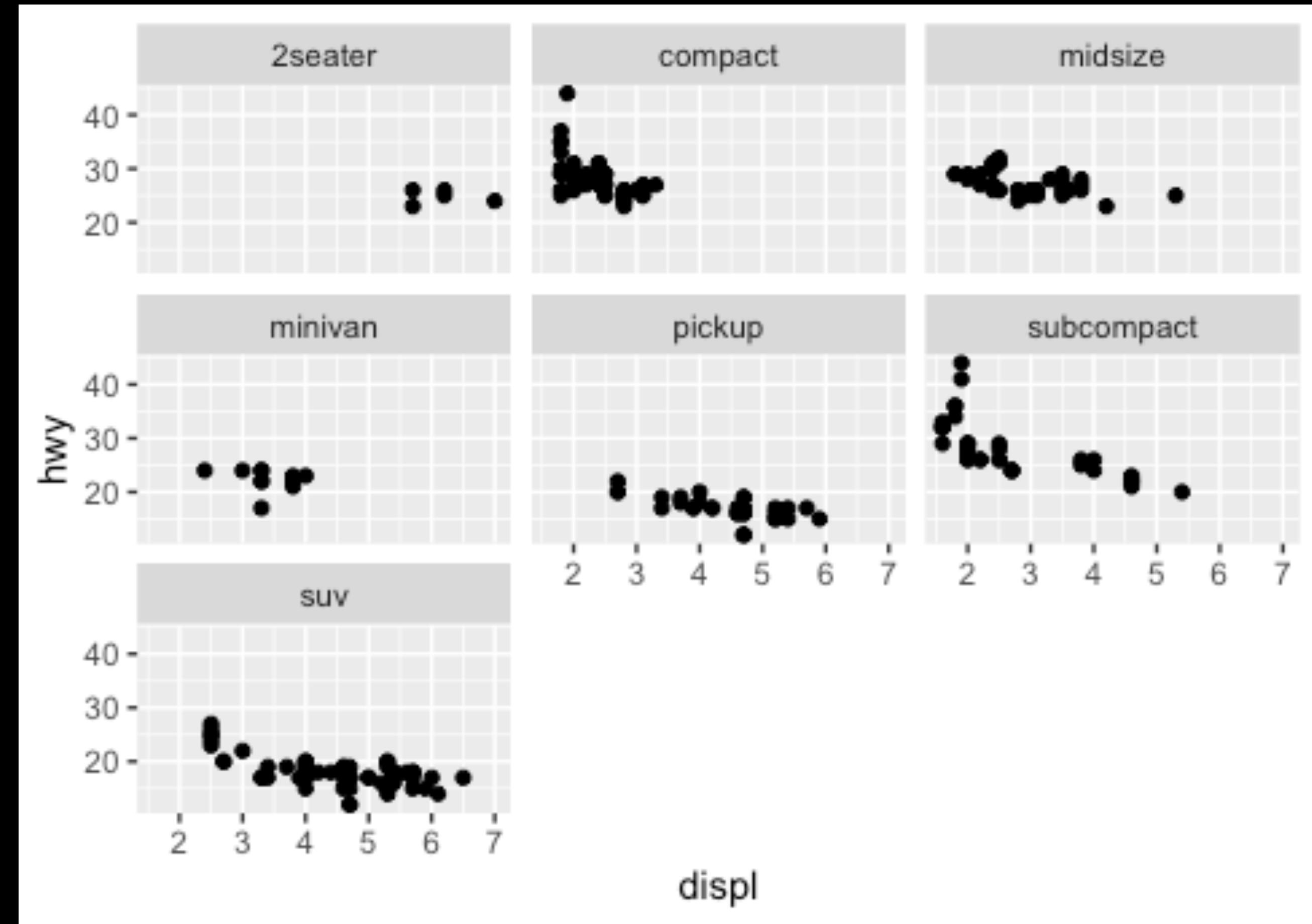
R

Faceting

(AKA "One graph for each sub-group")



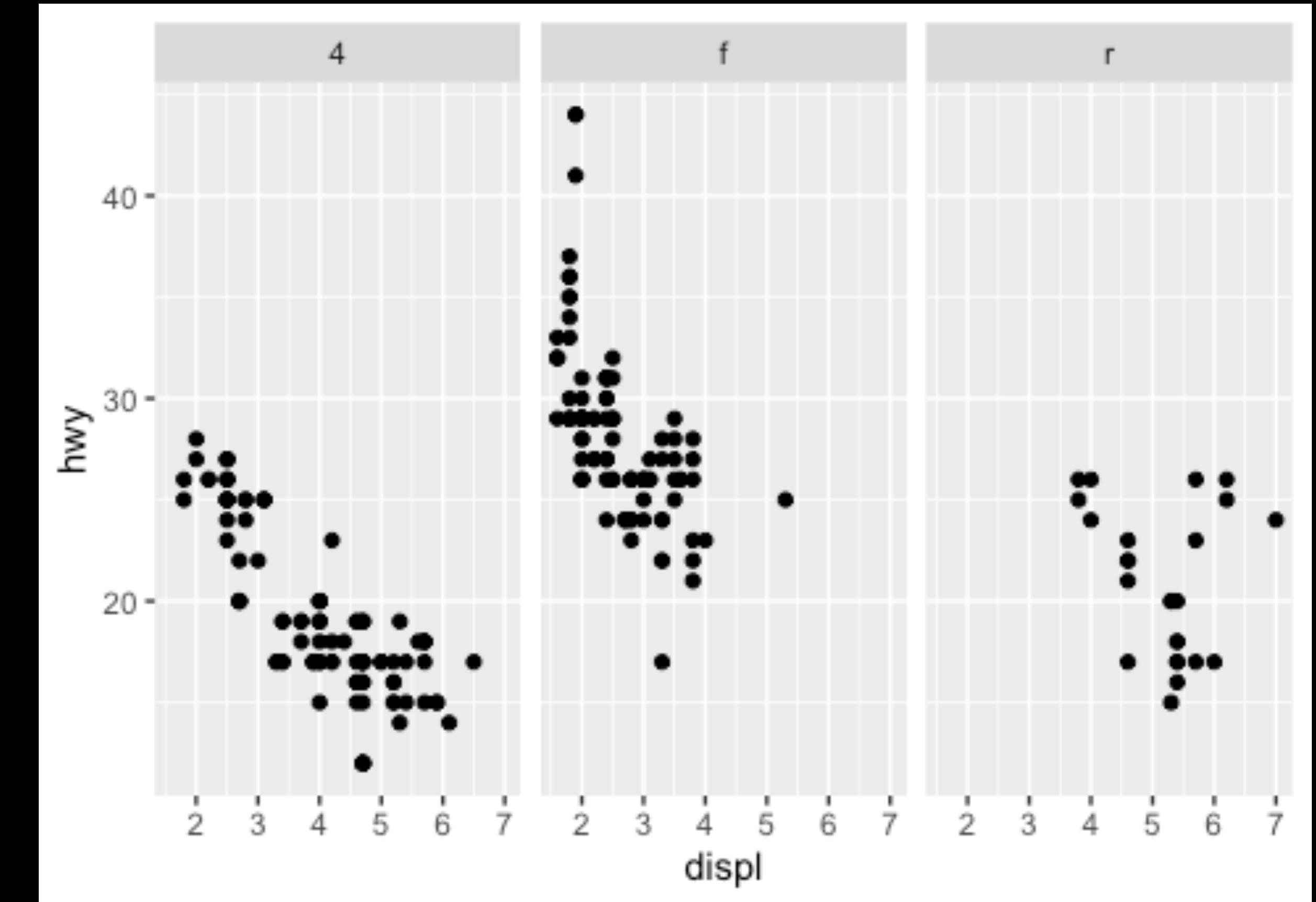
```
ggplot(mpg) +  
  geom_point(aes(displ, hwy, color=class))
```



```
ggplot(mpg) +  
  geom_point(aes(displ, hwy)) +  
  facet_wrap(~class)
```

Your Turn 8

- Use **facet_wrap** to create this graph
- For mpg, it shows displ vs. hwy.
- It is faceted by drv
- 5 Minutes

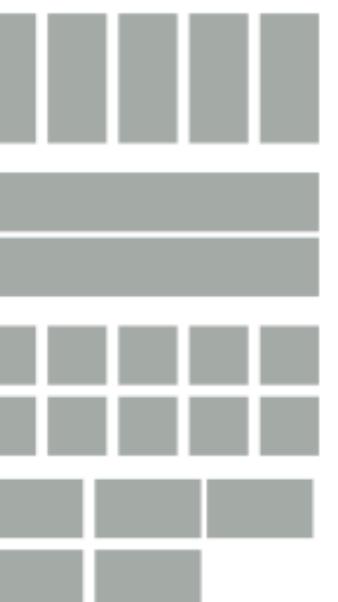


More on Faceting

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
```



t + facet_grid(. ~ fl)
facet into columns based on fl

t + facet_grid(year ~ .)
facet into rows based on year

t + facet_grid(year ~ fl)
facet into both rows and columns

t + facet_wrap(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

```
t + facet_grid(y ~ x, scales = "free")
```

x and y axis limits adjust to individual facets

- **"free_x"** - x axis limits adjust
- **"free_y"** - y axis limits adjust

Set **labeler** to adjust facet labels

```
t + facet_grid(. ~ fl, labeller = label_both)
```

fl: c fl: d fl: e fl: p fl: r

```
t + facet_grid(. ~ fl, labeller = label_bquote(alpha ^ .(x)))
```

α^c α^d α^e α^p α^r

```
t + facet_grid(. ~ fl, labeller = label_parsed)
```

c d e p r

Data Visualization with ggplot2 Cheat Sheet

R Studio

Basics

ggplot2 is based on the grammar of graphics. The idea is you can build every graph from the same components: **a data set**, **a coordinate system**, and **geoms**—visual 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 template below to build a graph.

```
ggplot(data = DATA) +  
  GEOM_FUNCTIONS +  
  mapping = aes(MAPPINGS),  
  stat = STAT,  
  position = POSITION,  
  COORDINATE FUNCTIONS +  
  SCALE FUNCTIONS +  
  THEME FUNCTIONS
```

Required

- Not required, sensible defaults supplied

ggplot(data = mpg, aes(cty, hwy))

Begins a plot that you finish by adding layers to. Add one geom function per layer.

geom_point(mapping = data, geom = "point")

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot()

Returns the last plot.

ggsave("plot.png", width = 5, height = 5)

Saves first plot at 5 x 5 file named "plot.png" in working directory. Matches file type to file extension.

ggplot2.org is a trademark of RStudio, Inc. • [www.RStudio.com](#) • 844-448-1232 • [rstudio.com](#)

Learn more at [docs.ggplot2.org](#) and [www.ggplot2-exts.org](#) • ggplot2 2.1.0 • Updated 11/16

Geoms

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

Two Variables

Continuous X, Continuous Y

e.g. `geom_label(aes(label = cty, nudge_x = 0, nudge_y = 1, check_overlap = TRUE))`

↳ Useful for expanding limits

b + `geom_curve(sezend = lat + 1, xend = long + 1, curvature = -0.5, x, y, label, alpha, angle, color, family, fontface, linejoin = "round", linemrite = 1)`

↳ x, y, label, alpha, color, curve, curvature, linejoin, size, weight

a + `geom_label(aes(label = "bus"))`

↳ x, y, alpha, color, group, linetype, size, weight

b + `geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))`

↳ x, y, alpha, color, fill, group, linetype, size, weight

a + `geom_polygon(aes(group = group))`

↳ x, y, alpha, color, group, linetype, size, weight

b + `geom_rug(sides = "bt")`

↳ x, y, alpha, color, line, size, weight

i + `geom_area()`

↳ x, y, alpha, color, fill, group, linetype, size, weight

e + `geom_line()`

↳ x, y, alpha, color, group, line, size, weight

i + `geom_step(direction = "hv")`

↳ x, y, alpha, color, group, line, size, weight

Line Segments

common aesthetics: x, y, alpha, color, line, size, weight

f + `geom_abline(aes(intercept = slope))`

↳ x, y, alpha, color, fill, group, line, intercept = slope

b + `geom_hline(aes(intercept = long))`

↳ x, y, alpha, color, fill, group, line, intercept = long

b + `geom_segment(aes(endx = lat, endy = long + 1))`

↳ x, y, alpha, color, fill, group, line, segment = segment, size, weight

b + `geom_text(aes(label = cty, nudge_x = 1, nudge_y = 1, check_overlap = TRUE))`

↳ x, y, label, alpha, color, family, fontface, linejoin, size, weight

Continuous Function

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

↳ x, y, alpha, color, group, line, size, weight

i + `geom_area()`

↳ x, y, alpha, color, fill, group, line, size, weight

i + `geom_line()`

↳ x, y, alpha, color, group, line, size, weight

Maps

data <- data.frame(murder = USArrests\$Murder, state = rownames(USArrests))

map <- map_data("us-states")

k <- `ggplot(data, aes(state, map = map)) + expand_limits(x = min(state[, 1]), y = min(state[, 2]))`

↳ x, y, alpha, color, fill, group, line, size, weight

k + `geom_map(aes(fill = murder))`

↳ x, y, min, max, alpha, fill, group, line, size, weight

k + `geom_bar()`

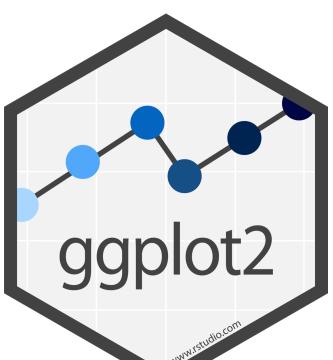
↳ x, y, alpha, color, fill, group, line, size, weight

d + `geom_raster(aes(fill = z))`

↳ x, y, z, alpha, color, group, line, size, weight

d + `geom_tile(aes(fill = z))`

↳ x, y, alpha, color, fill, group, line, size, weight



Interactive Graphics

Search...

On This Page

- Plotly for R
- Installation
- Initialization
- Cutomizing the Layout
- Modifying Layers
- Interactively View the JSON Object
- Modify with Style
- Modify with Build
- Resources
- What About Dash?

Getting Started with Plotly in ggplot2

Get started with Plotly's R graphing library with ggplot2 to make interactive, publication-quality graphs online.

► New to Plotly?

Plotly for R

Plotly is an R package for creating interactive web-based graphs via [plotly](#)'s JavaScript graphing library, `plotly.js`.

The [plotly R package](#) serializes ggplot2 figures into Plotly's [universal graph JSON](#). `plotly::ggplotly` will crawl the ggplot2 figure, extract and translate all of the attributes of the ggplot2 figure into JSON (the colors, the axes, the chart type, etc), and draw the graph with `plotly.js`.

Furthermore, you have the option of manipulating the Plotly object with the `style` function.

Installation

Plotly is now on CRAN!

Plotly (plotly.com)

library(plotly)

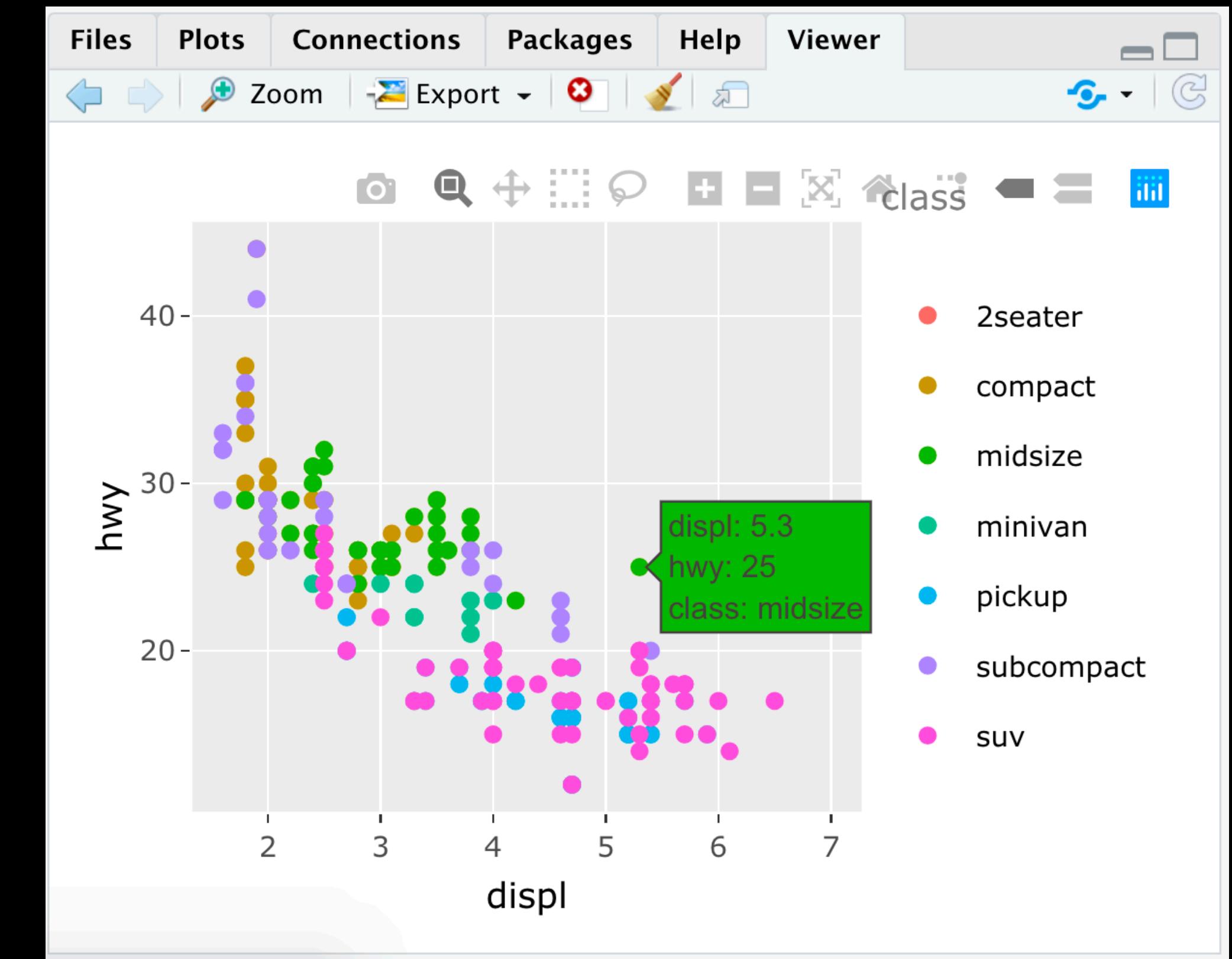
Suggest an edit to this page

?ggplotly

```
library(plotly)
```

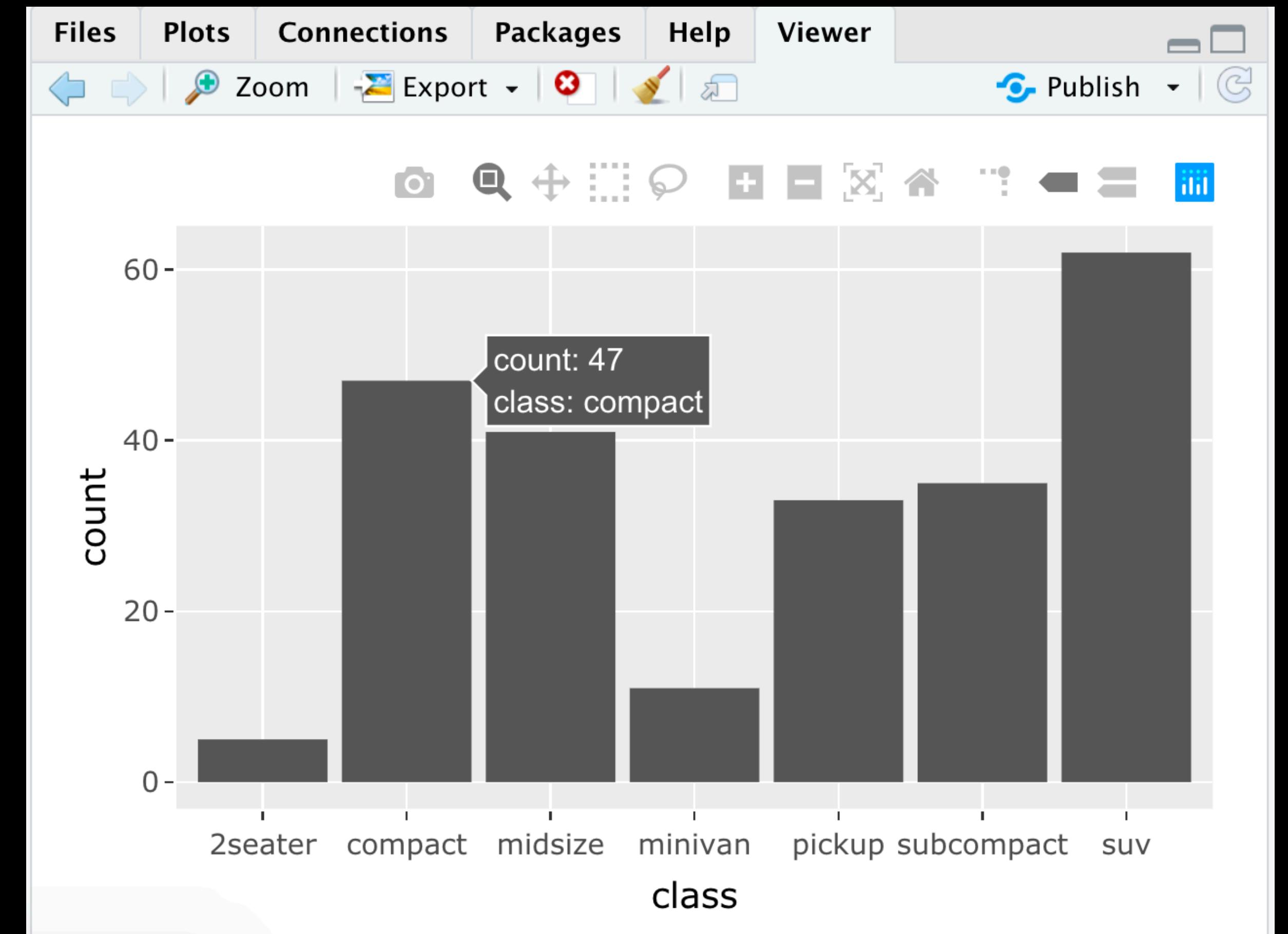
```
chart = ggplot(mpg) +  
  geom_point(aes(displ, hwy, color=class))
```

```
ggplotly(chart)
```



Your Turn 10

- I gave you code to create the static version of this chart
- Make it interactive by passing the ggplot2 object to [?ggplotly](#)
- You will need to load the **plotly** library.
- 5 Minutes



ggplot2.tidyverse.org

The screenshot shows a web browser window displaying the ggplot2.tidyverse.org website. The title bar reads "Create Elegant Data Visualisati x" and "Garrett". The address bar shows the URL "ggplot2.tidyverse.org". The page content includes a header with the ggplot2 logo and "part of the tidyverse". A main section titled "Usage" contains text about the philosophy of ggplot2 and a code snippet. To the right, there are "Links" to CRAN, GitHub, issues, and more. At the bottom, there's a plot and developer information.

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()
```

class

2seater

Links

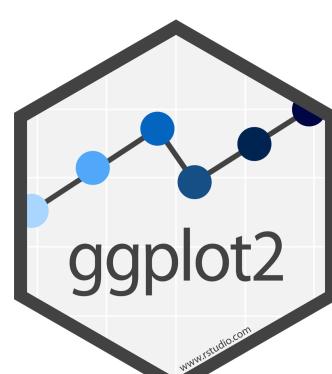
- Download from CRAN at <https://cran.r-project.org/package=ggplot2>
- Browse source code at <https://github.com/tidyverse/ggplot2>
- Report a bug at <https://github.com/tidyverse/ggplot2/issues>
- Learn more at <http://r4ds.had.co.nz/data-visualisation.html>

License

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

Developers

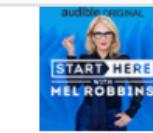
Hadley Wickham
Author, maintainer



Hello
Select your address

Holiday Deals Gift Cards Best Sellers Customer Service Find a Gift New Releases Whole Foods AmazonBasics Free Shipping Sell Registry Coupons

Books Advanced Search New Releases Best Sellers & More Children's Books Textbooks Textbook Rentals Best Books of the Month



audible plus *

Only \$4.95 a month for the first 6 months ›



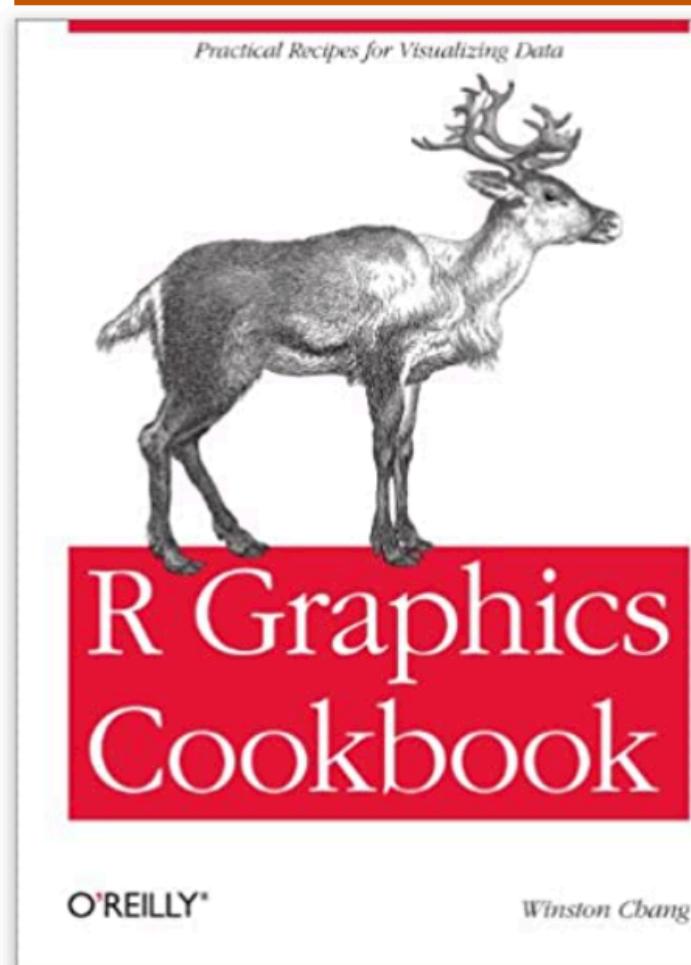
Books > Computers & Technology > Programming

R Graphics Cookbook: Practical Recipes for Visualizing Data 1st Edition

by Winston Chang (Author)

★★★★★ 131 ratings

Related Text



Kindle

\$8.09 - \$18.14

Paperback

\$26.36

Other Sellers

See all 2 versions

More Buying Choices

30 used from \$16.79

30 Used from \$16.79

Select delivery location

See All Buying Options

There is a newer edition of this item:



R Graphics Cookbook: Practical Recipes for Visualizing Data

\$55.05

★★★★★ (33)

In Stock.

R Cookbook

Winston Chang

Closing Exercise

- Fill out the "Summary" section
- Fill out the "How I can apply this to my work"
- Share with your neighbor!

03 : 00

ggplot2	
Main Ideas	Notes _____ _____ _____ _____ _____ _____ _____
Summary	
How I can apply this to my work	

Any Questions?