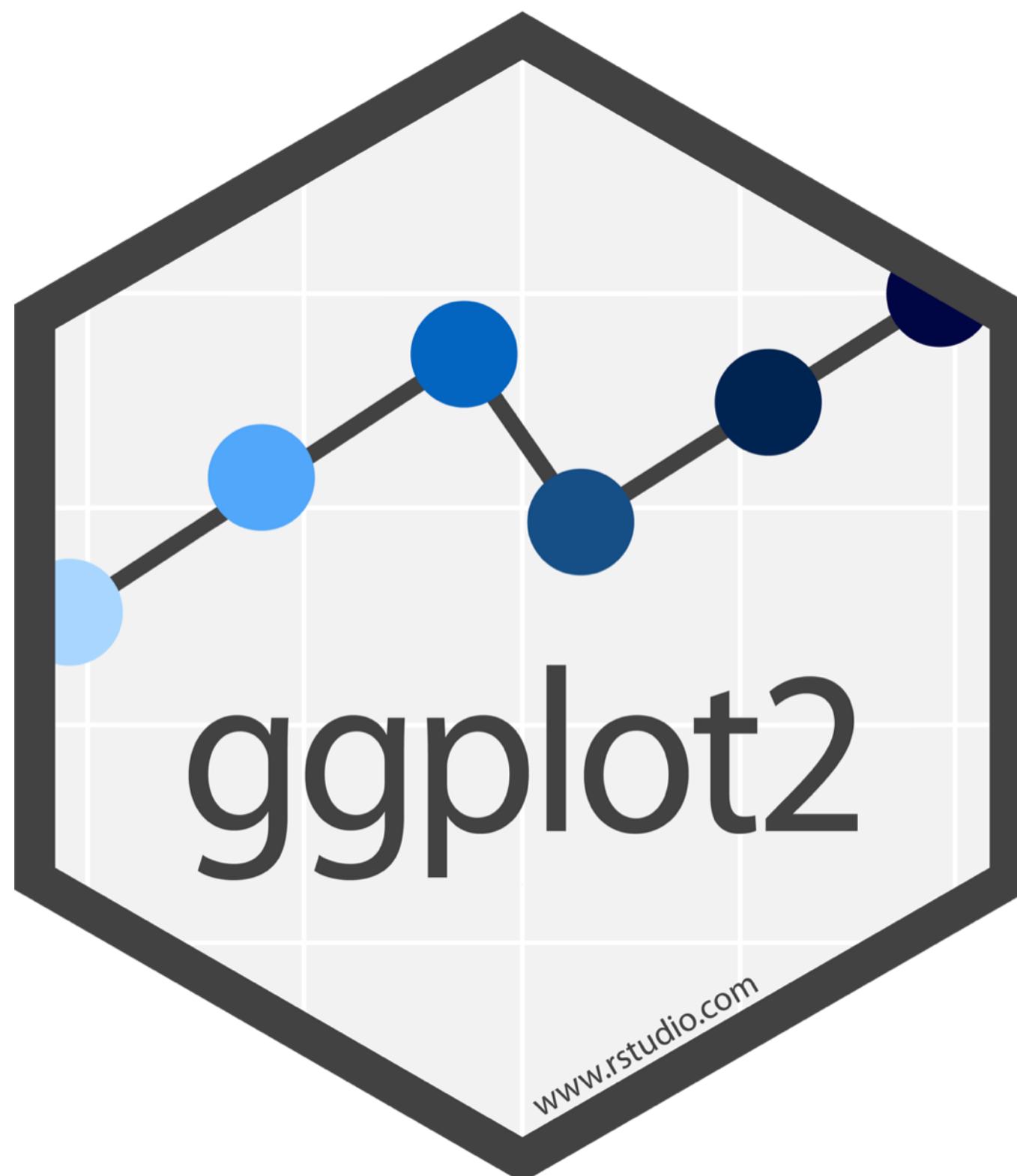


Visualize Data with

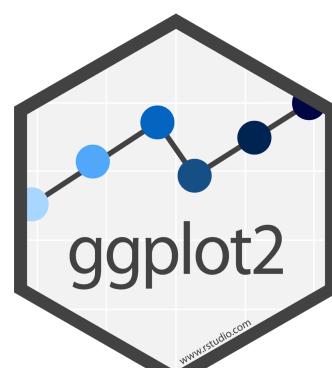
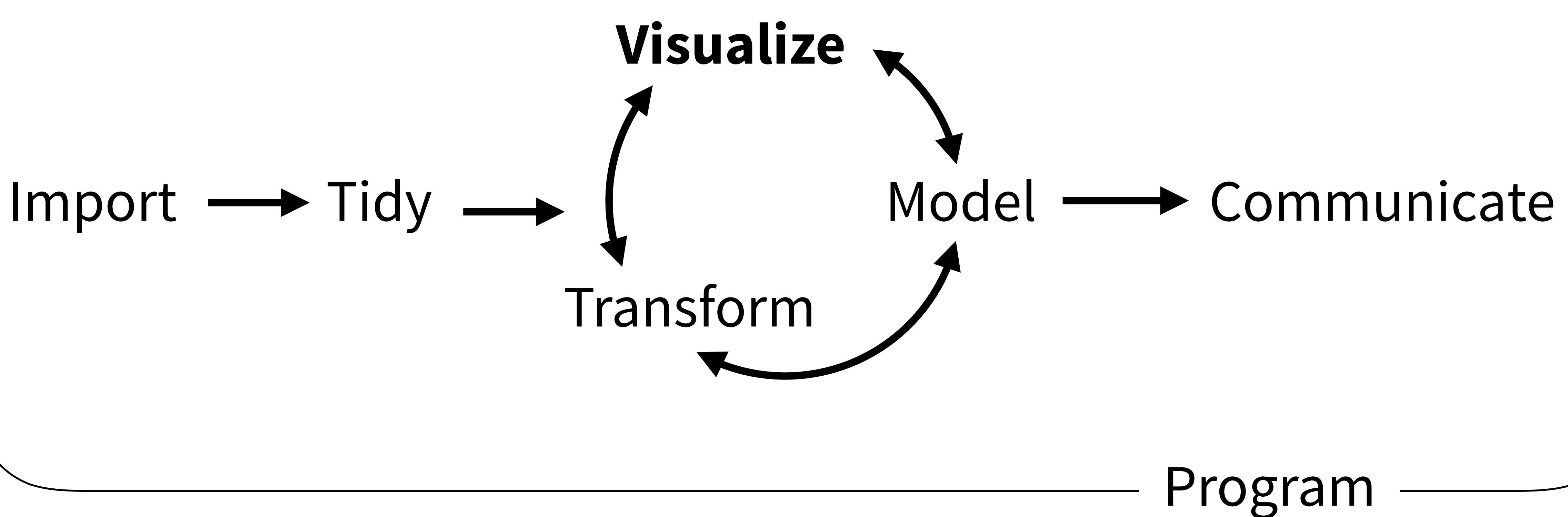


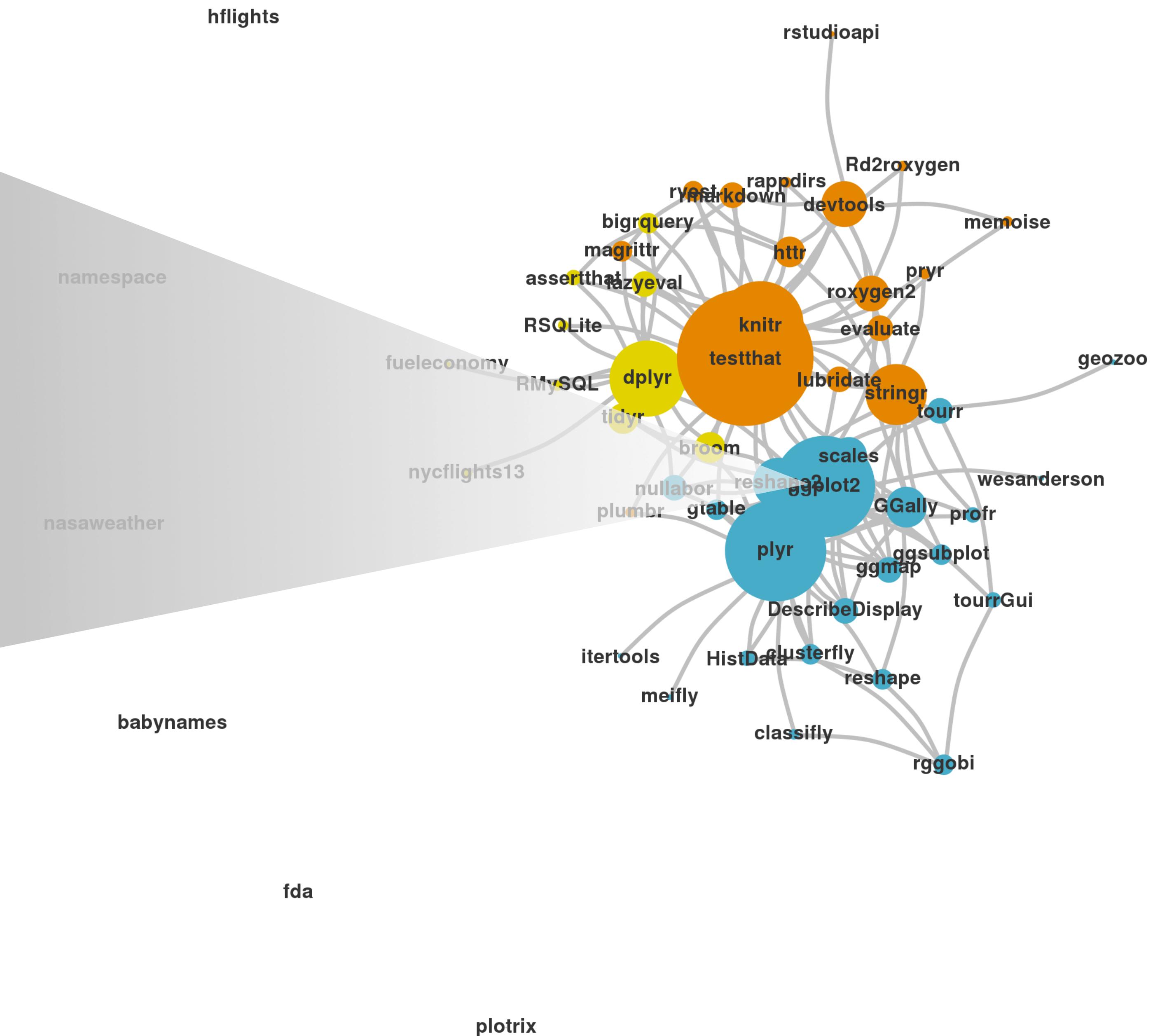
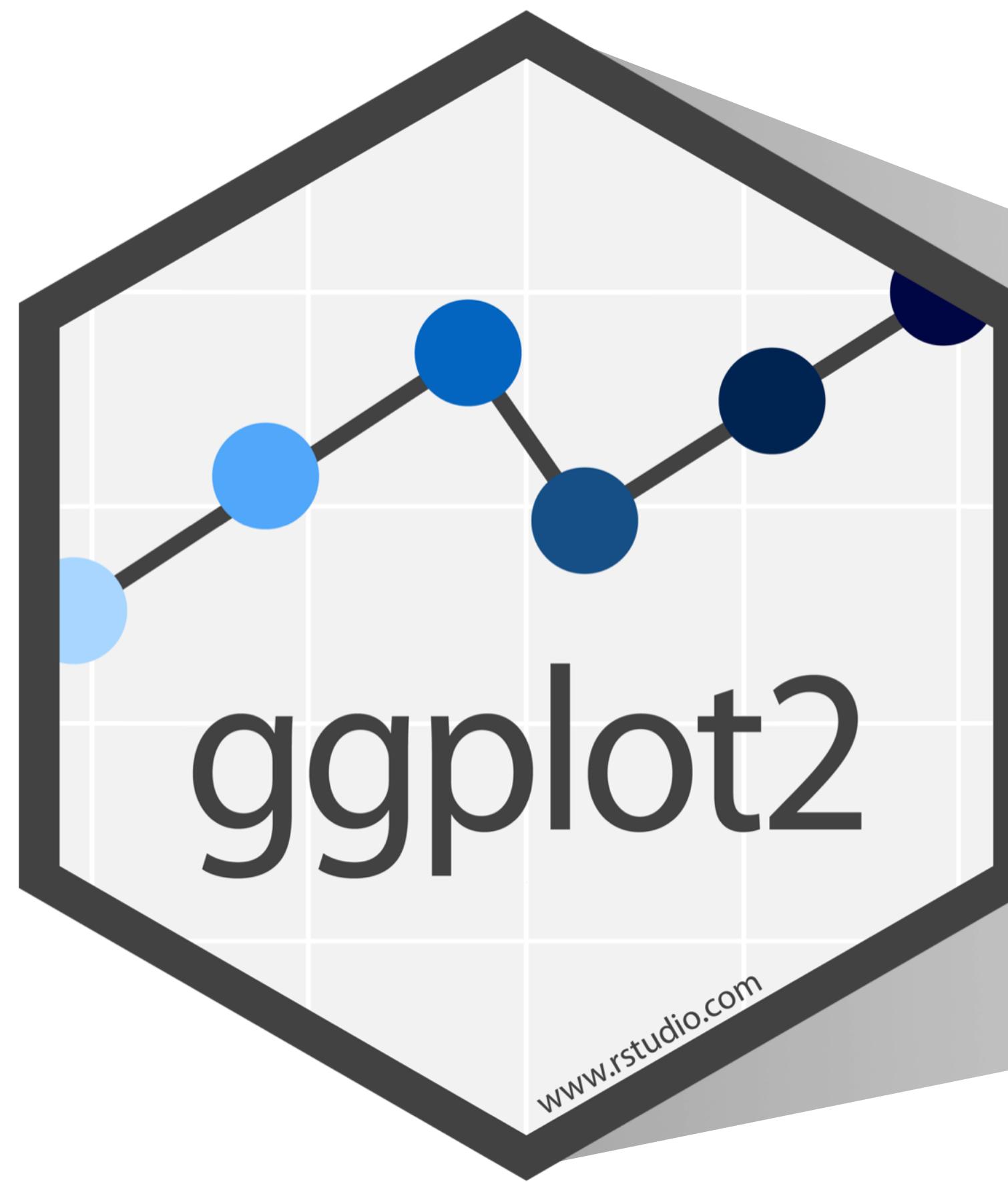
In R4DS
Visualizing Data

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

- John Tukey

(Applied) Data Science





RStudio

Project: (None)

02-Visualize-Data.Rmd

```
1 ---  
2 title: "Visualize Data"  
3 output:  
4   html_document:  
5     df_print: paged  
6 ---  
7  
8 ## Setup  
9  
10 The first chunk in an R Notebook is usually titled  
"setup," and by convention includes the R packages  
you want to load. Remember, in order to use an R  
package you have to run some `library()` code every  
session. Execute these lines of code to load the  
packages.  
11  
12 ```{r setup}  
13 library(ggplot2)  
14 library(fivethirtyeight)  
15 ```  
16  
17 ## Bechdel test data
```

10:33 # Setup R Markdown

Console

Environment History Connections

Import Dataset

Global Environment

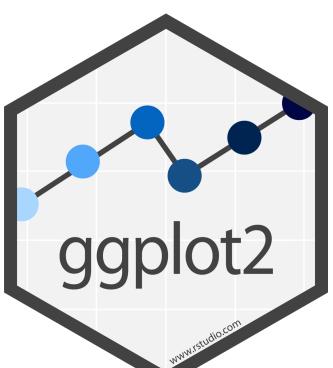
Open the R Notebook 01-Visualize.Rmd

Files Plots Packages Help Viewer

New Folder Delete Rename More

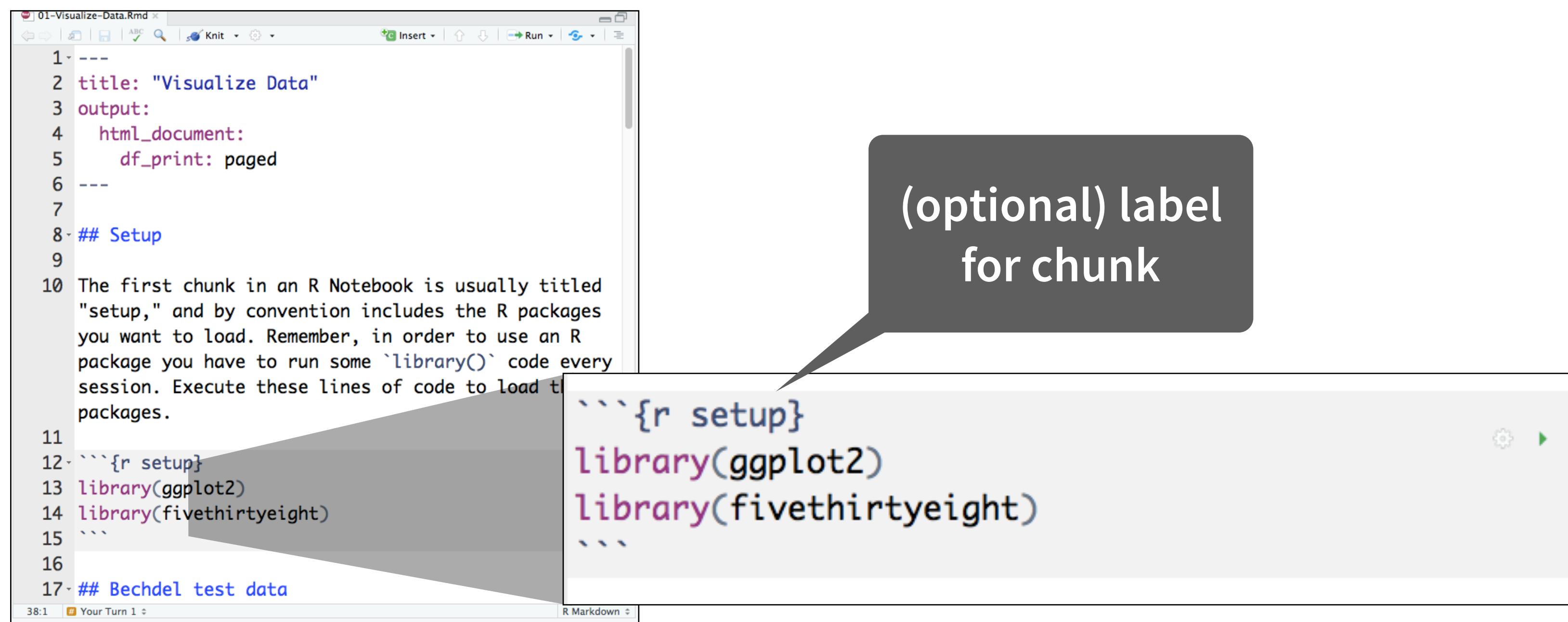
Home > Dropbox > Intro_to_R_and_RStudio > Day1

Name	Size	Modified
..		
code		
keynotes		
slides		



Setup

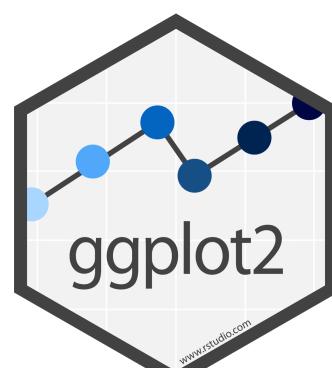
The setup chunk is always run once before anything else



```
1 ---  
2 title: "Visualize Data"  
3 output:  
4   html_document:  
5     df_print: paged  
6 ---  
7  
8 ## Setup  
9  
10 The first chunk in an R Notebook is usually titled  
11 "setup," and by convention includes the R packages  
12 you want to load. Remember, in order to use an R  
13 package you have to run some `library()` code every  
14 session. Execute these lines of code to load the  
15 packages.  
16  
17 # Bechdel test data
```

(optional) label for chunk

```
```{r setup}  
library(ggplot2)
library(fivethirtyeight)
...
Bechdel test data
```

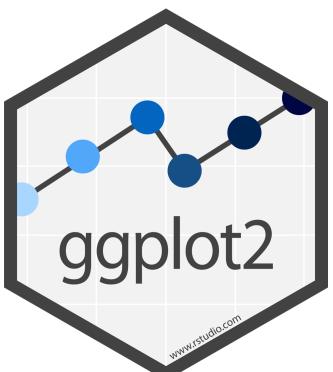


# bechdel

## Data on movies and the Bechtel test

bechdel

?bechdel



# Consider

Confer with the people around you.

What relationship do you expect to see  
between movie budget (budget) and  
domestic gross(domgross)?

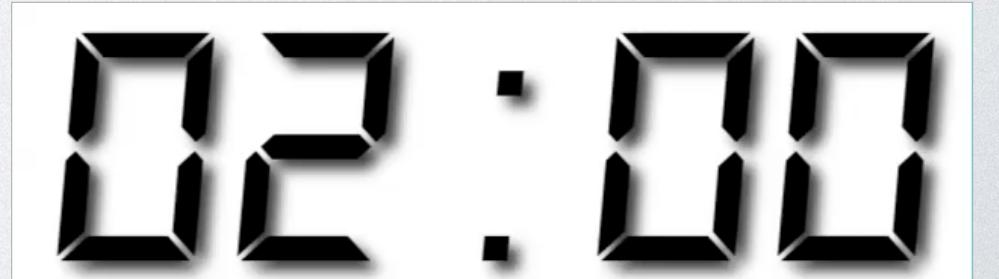


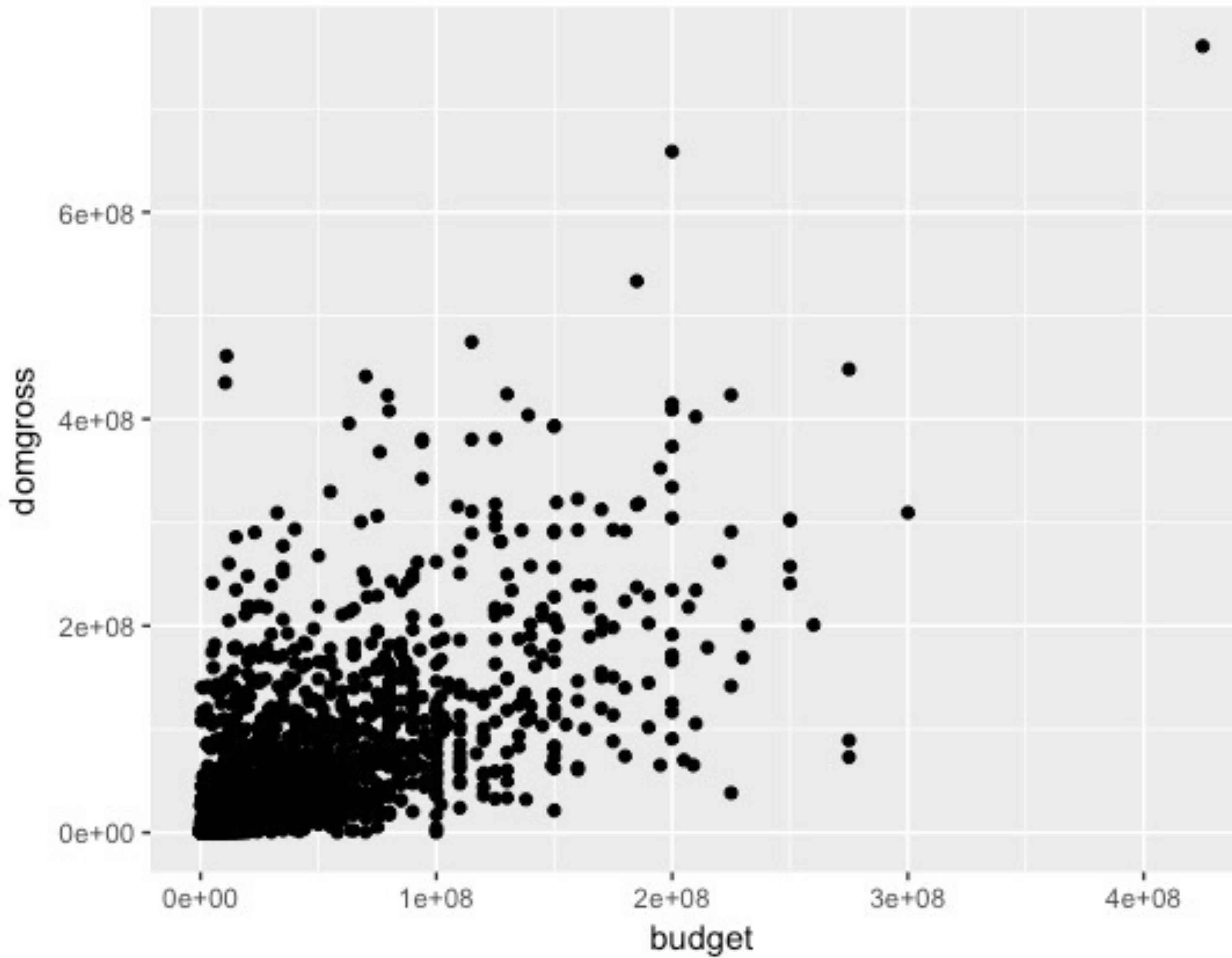
# Your Turn 1

Run this code in your notebook to make a graph.

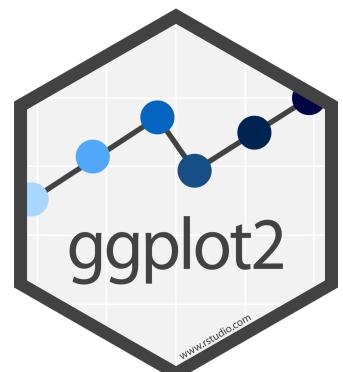
Pay strict attention to spelling, capitalization, and parentheses!

```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```

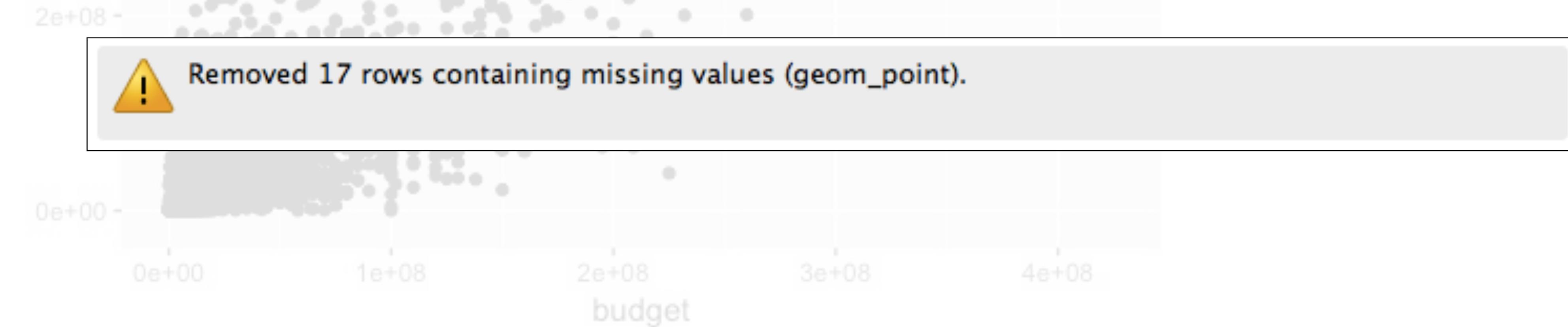




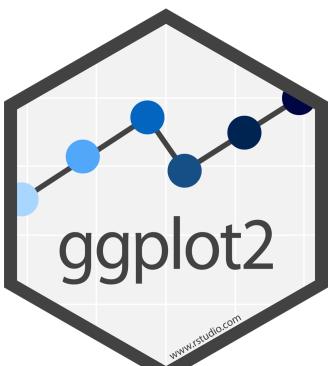
```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```



When you run this code, you will get what looks like an error, but is actually just a message from R. Some of the rows in the dataset didn't contain information for budget and/or domgross, so they're not plotted.

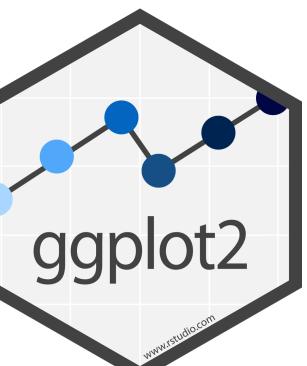


```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```



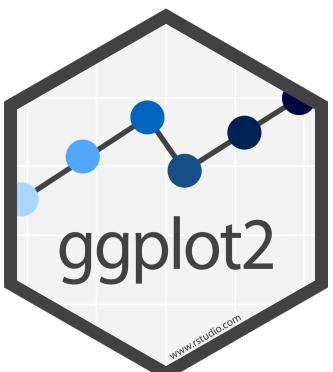
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```



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

```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```



```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```

data

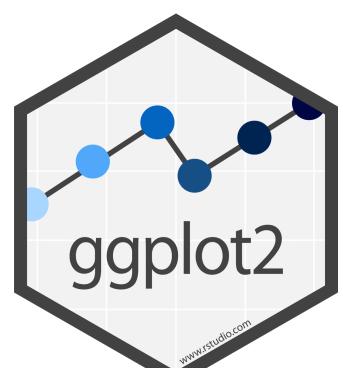
+ before new line

type of layer

aes()

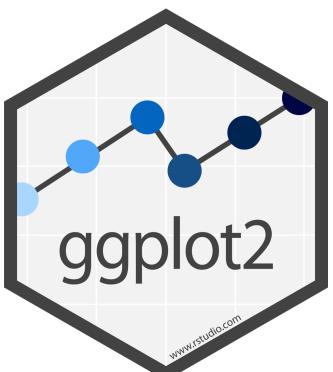
x variable

y variable



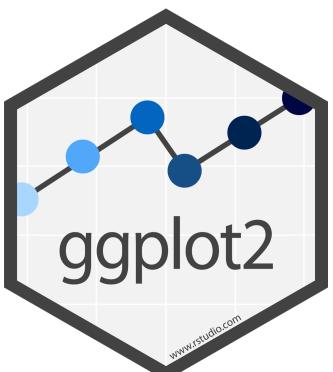
# A template

```
ggplot(data = <DATA>) +
 <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
 geom_point(mapping = aes(x = budget, y = domgross))
```



# A template

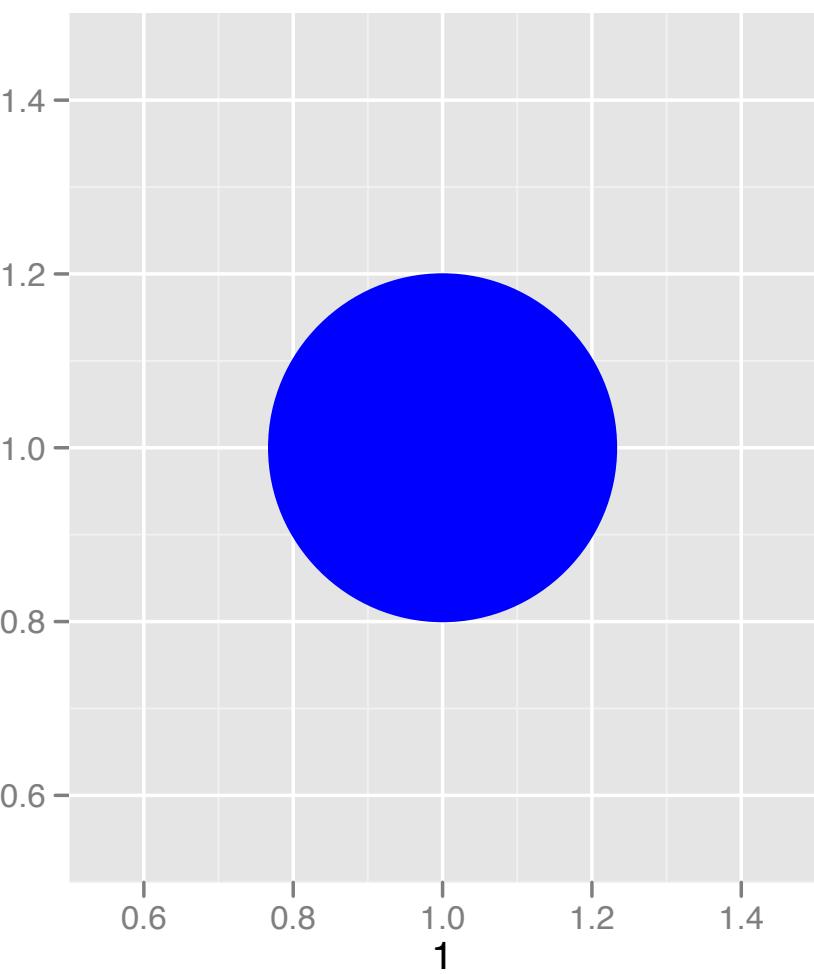
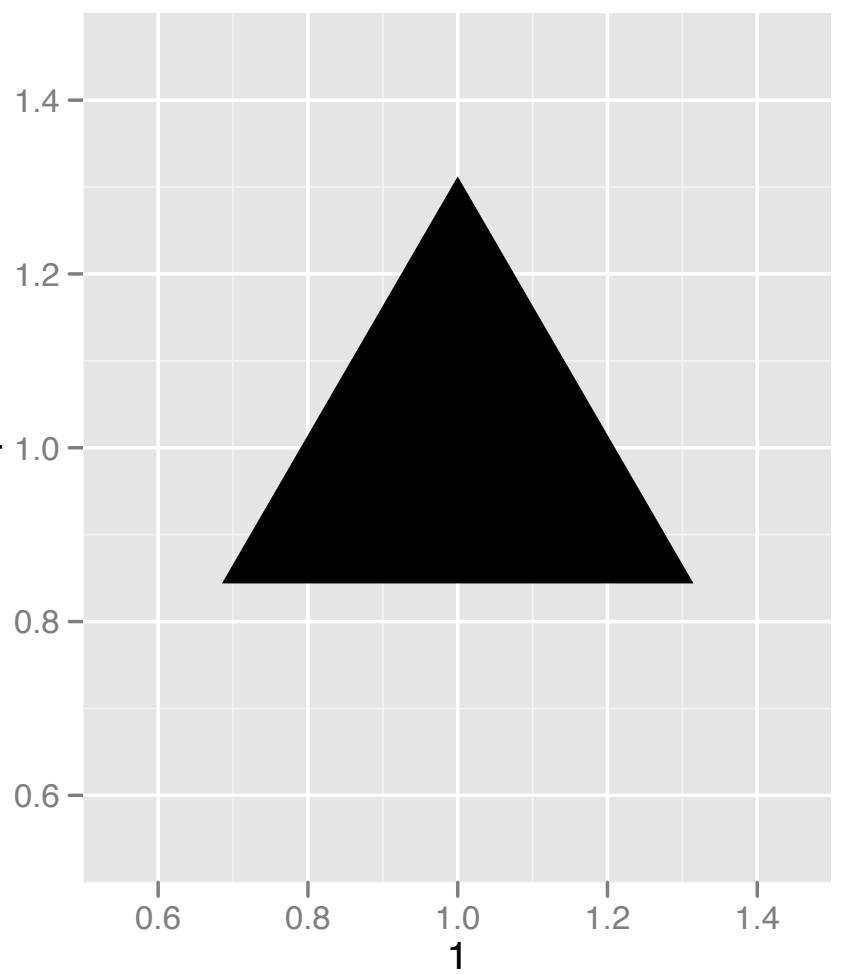
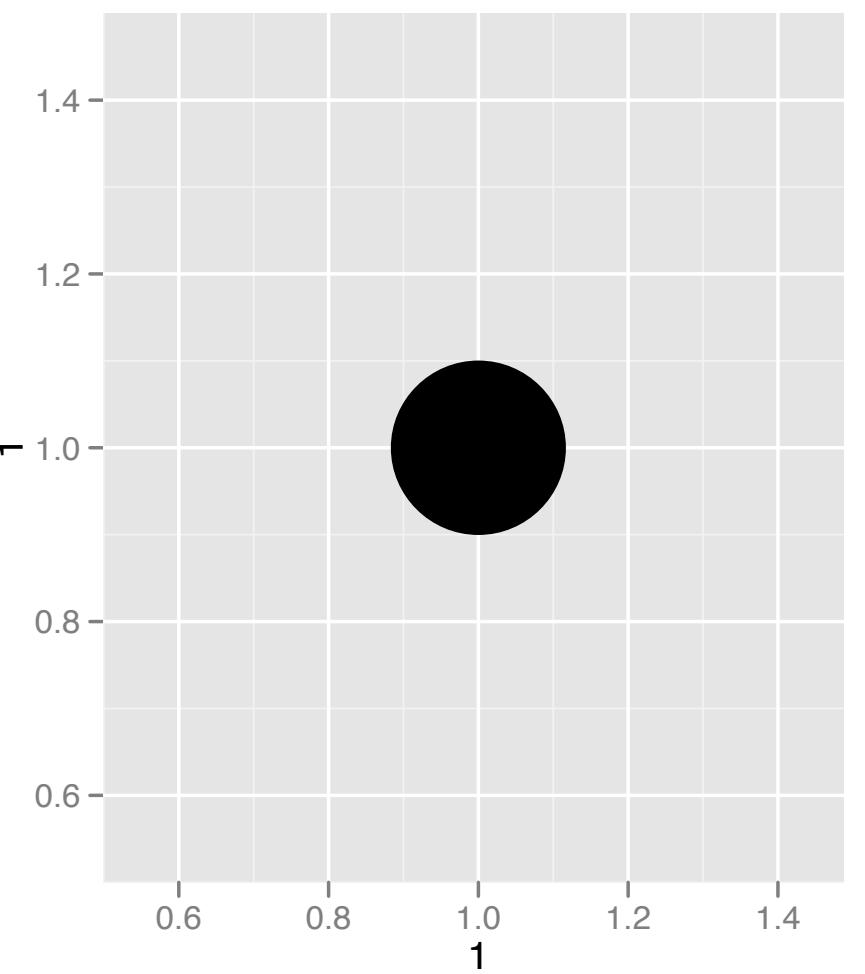
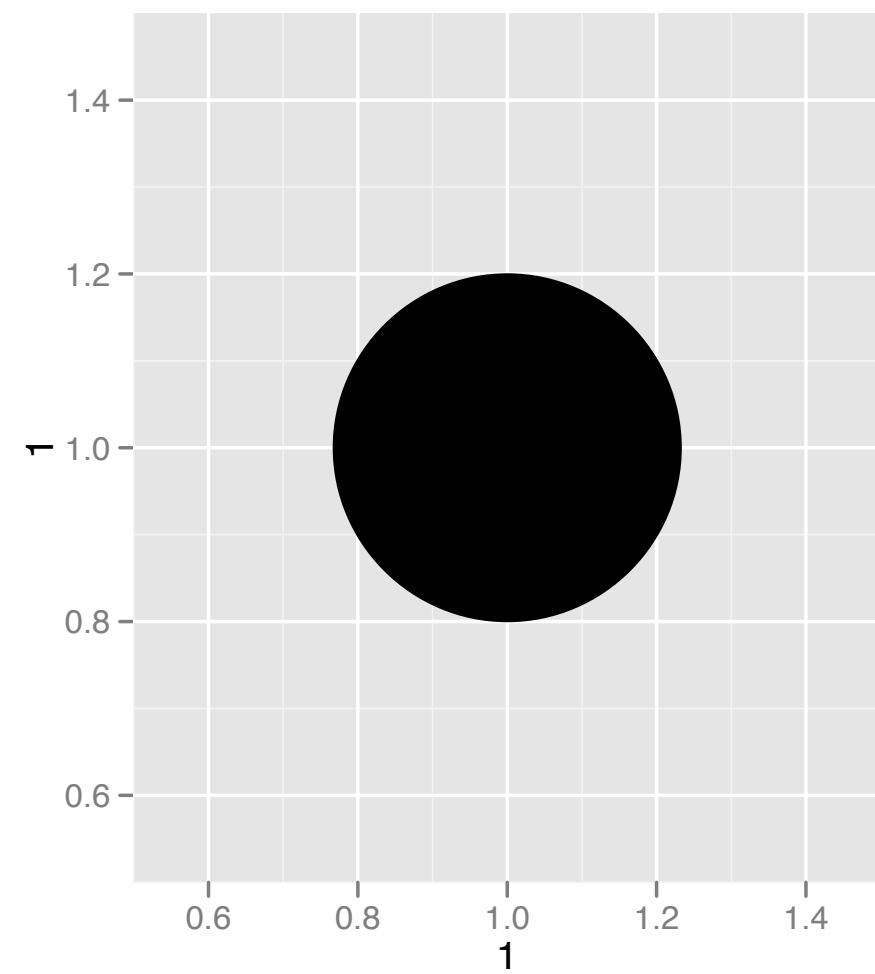
```
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```



# Aesthetics

R

# Aesthetics



## Visual Space      Data Space

color ←→ clean\_test

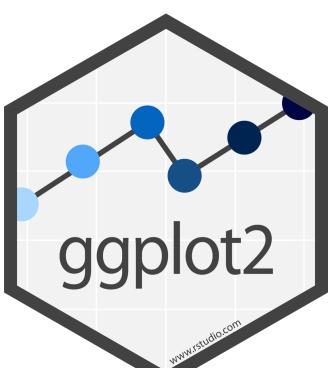
Purple ←→ nowomen

Blue ←→ notalk

Teal ←→ men

Lime ←→ dubious

Yellow ←→ ok



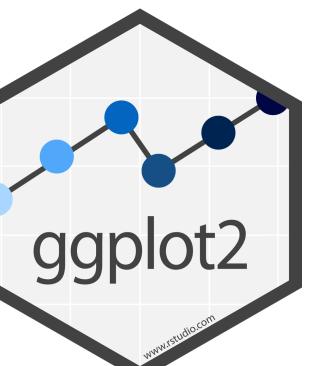
# Aesthetics

```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color = clean_test))
```

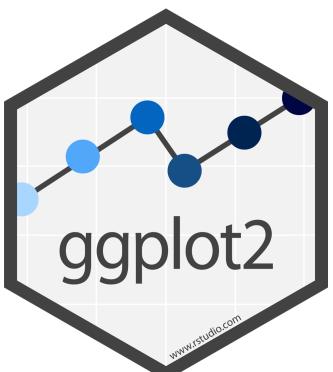
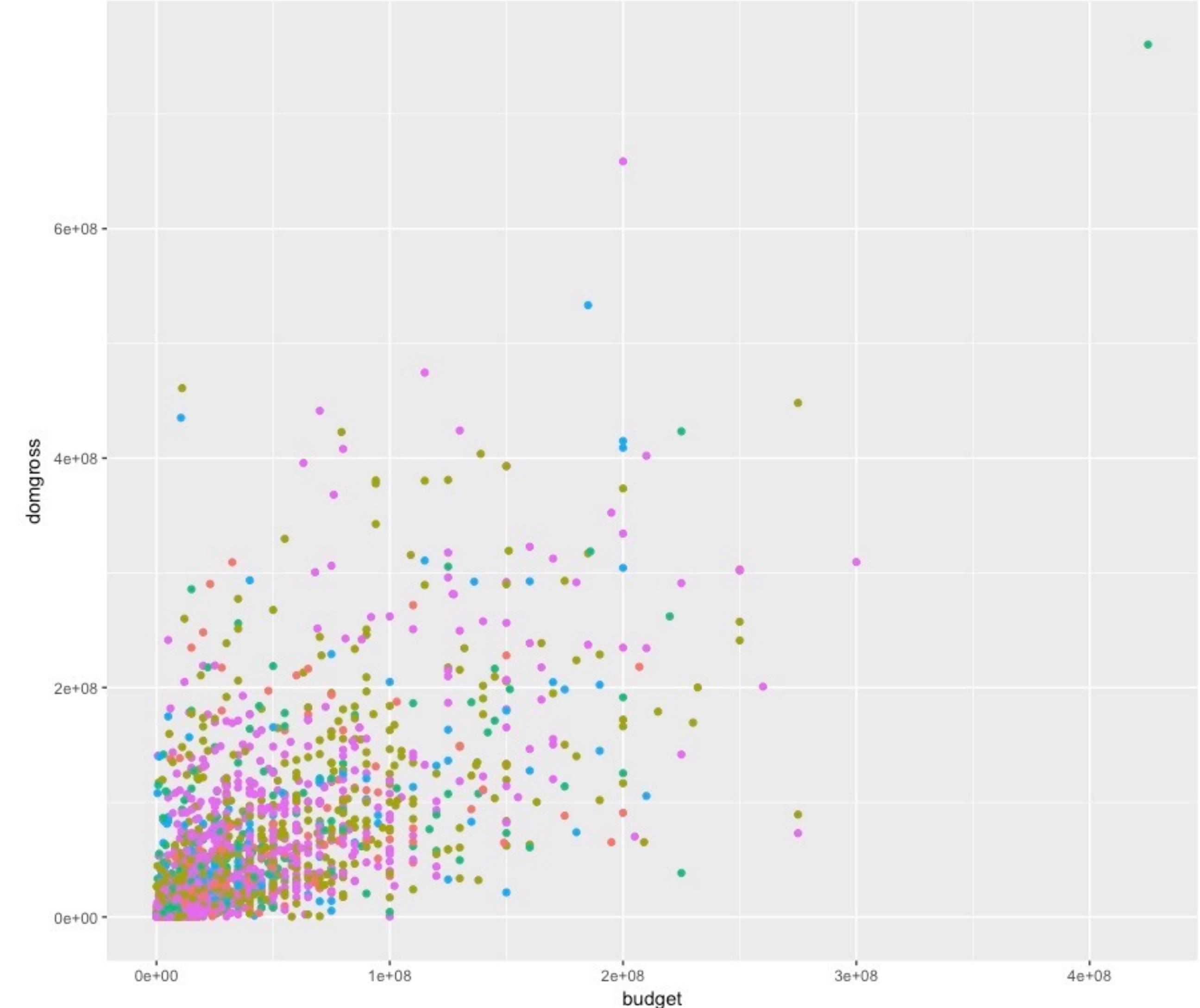
aesthetic  
property

Variable to  
map it to

```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, size = clean_test))
```



```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```



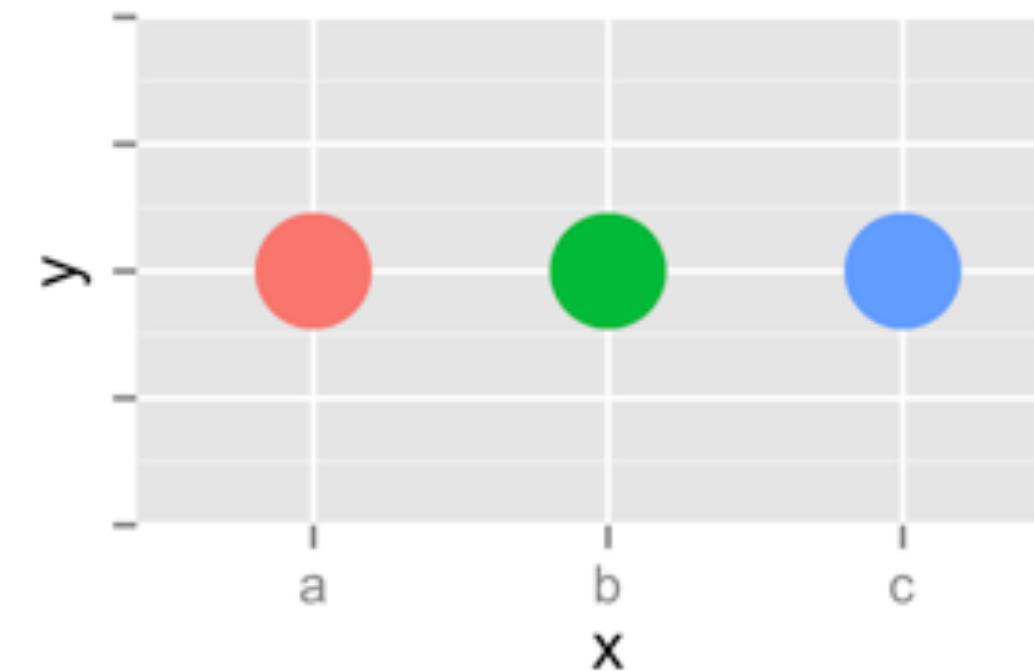
# Your Turn 2

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

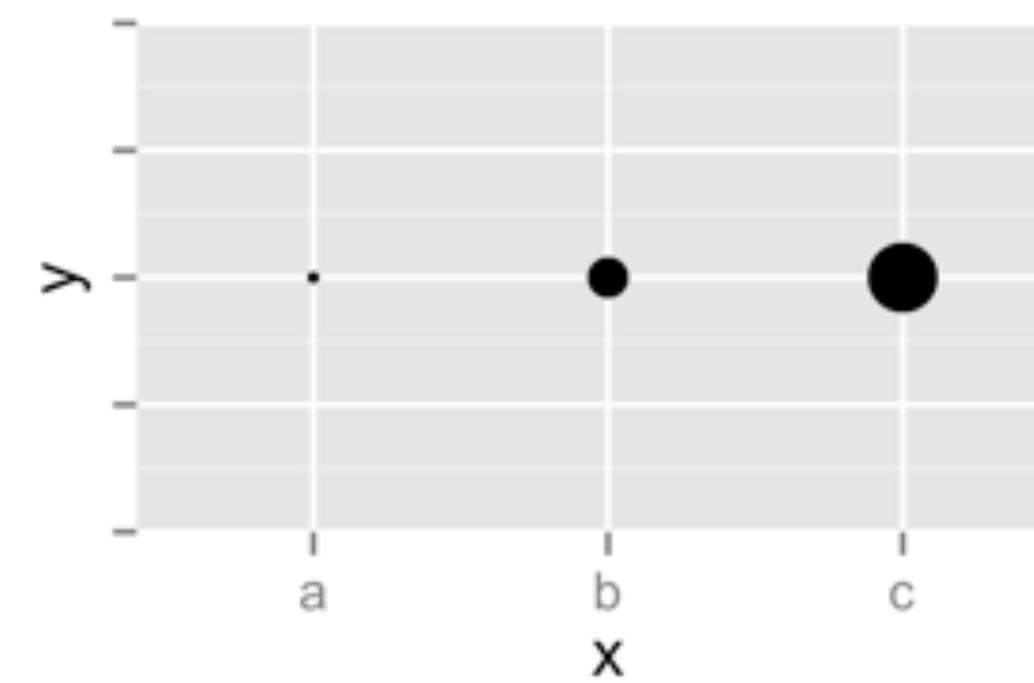
- Do different things happen when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?



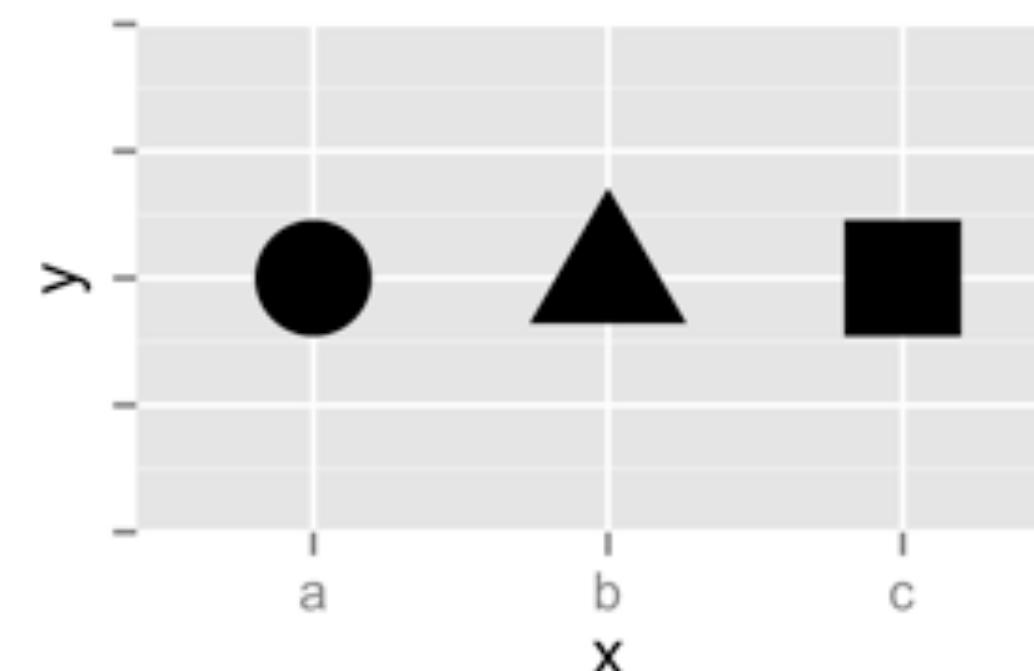
Color



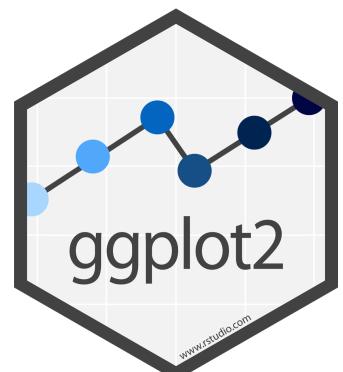
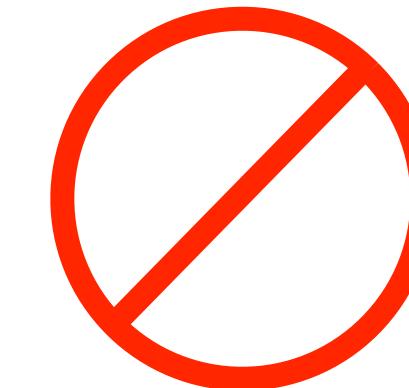
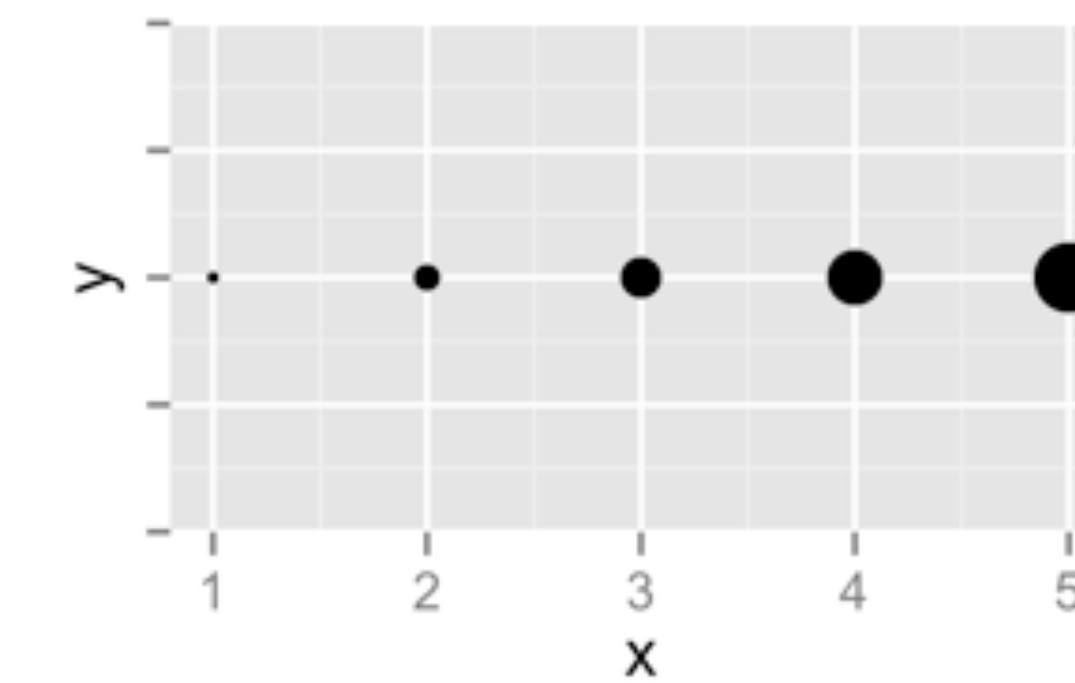
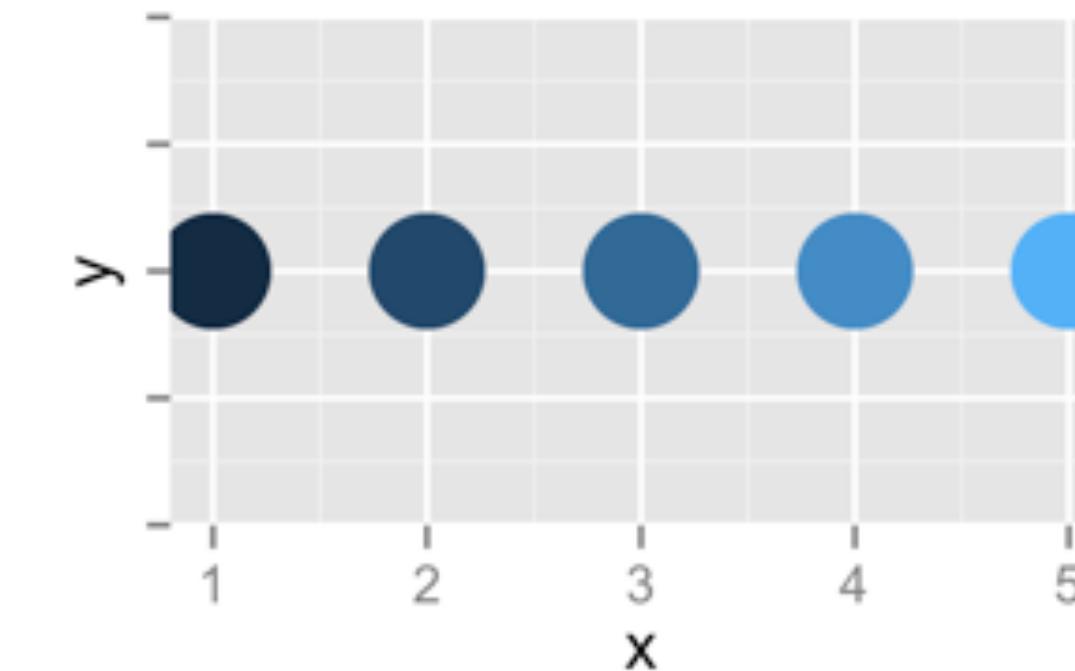
Size



Shape



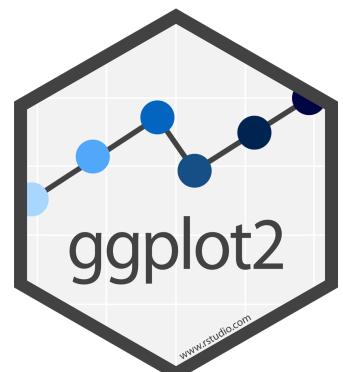
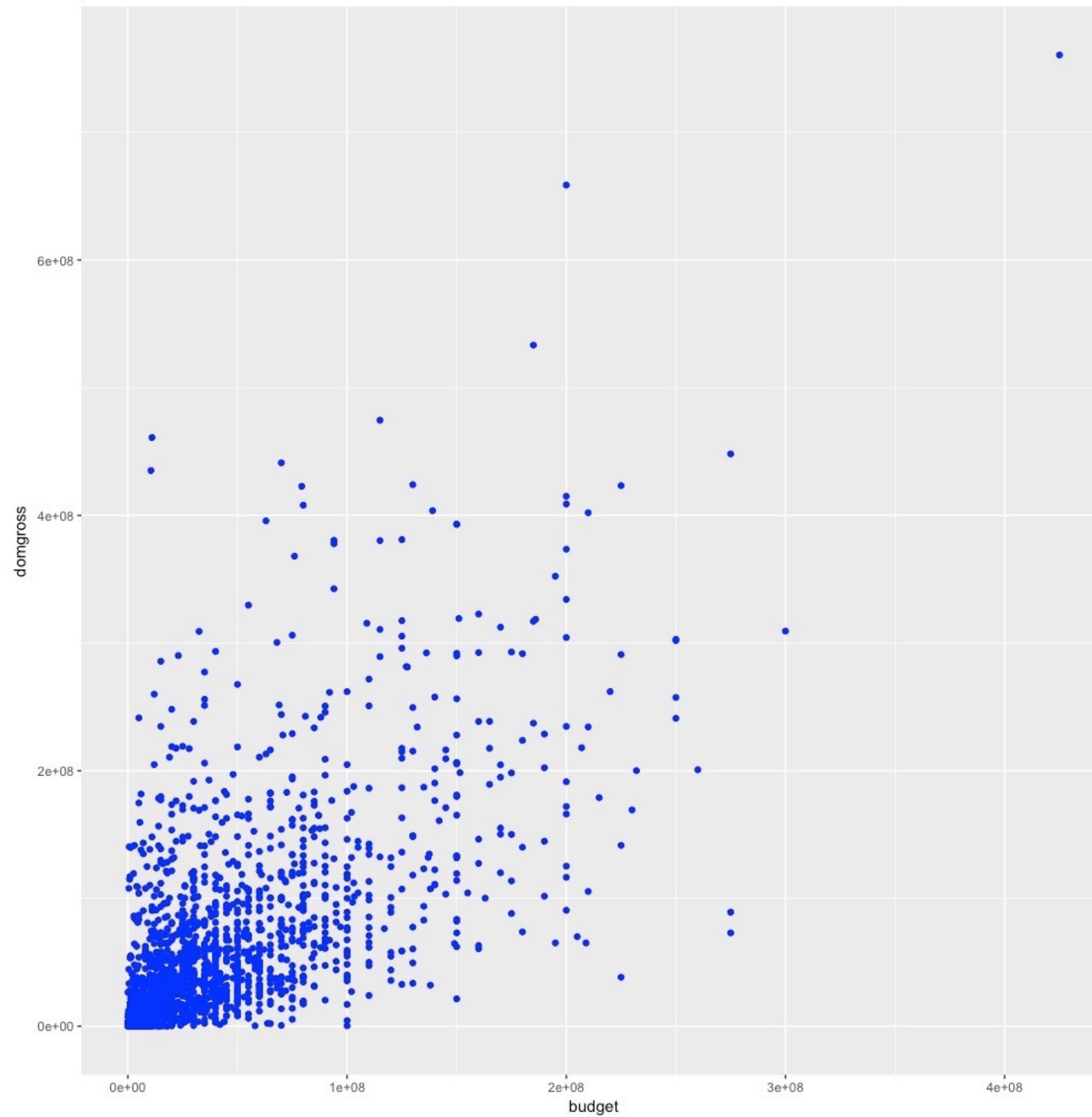
Continuous

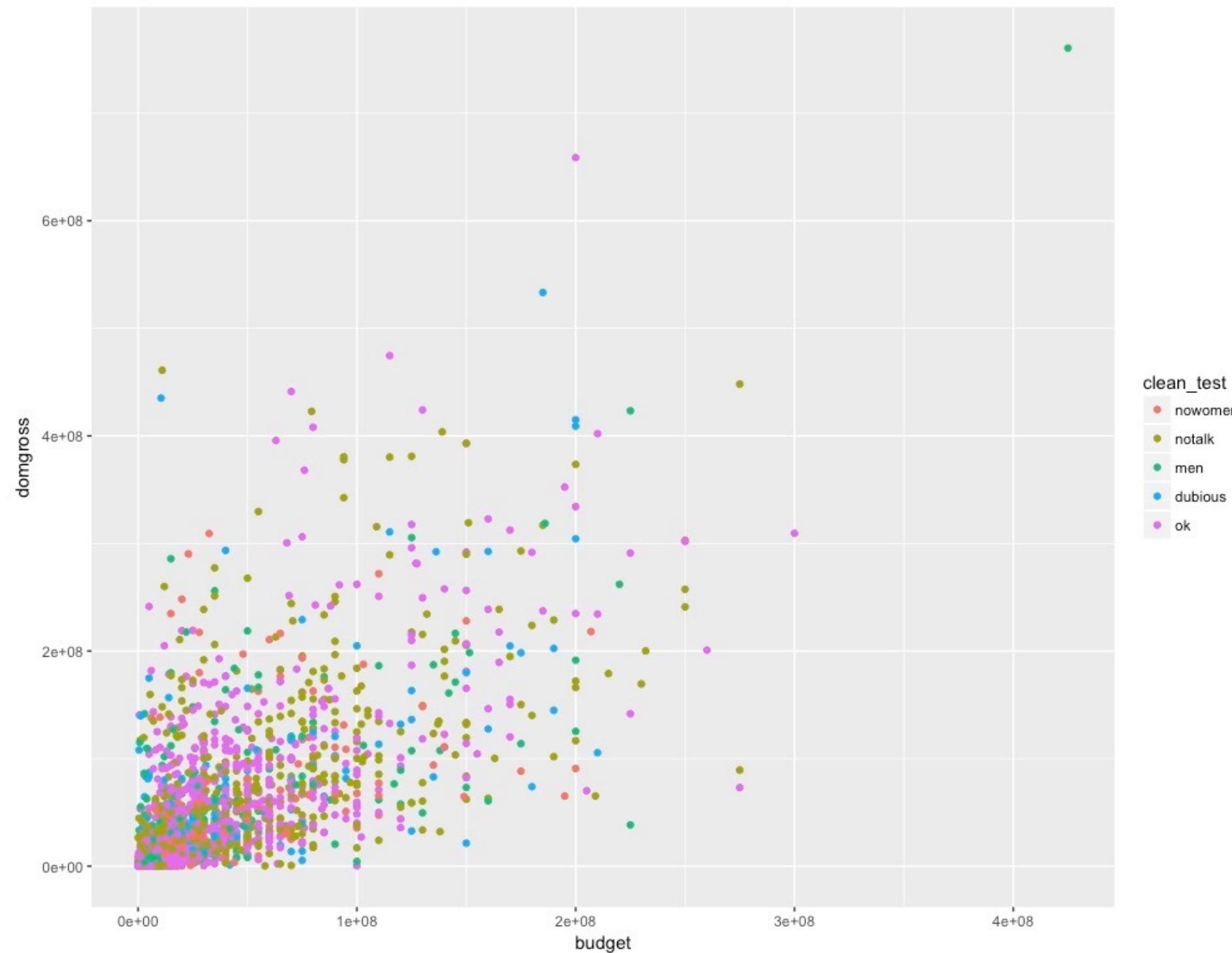


# set vs. map

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.

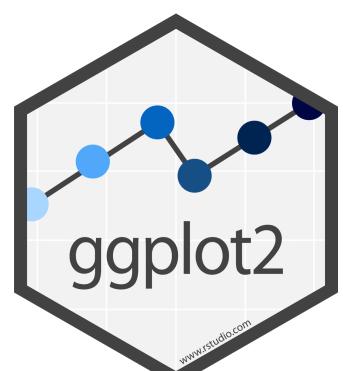
# How would you make this plot?



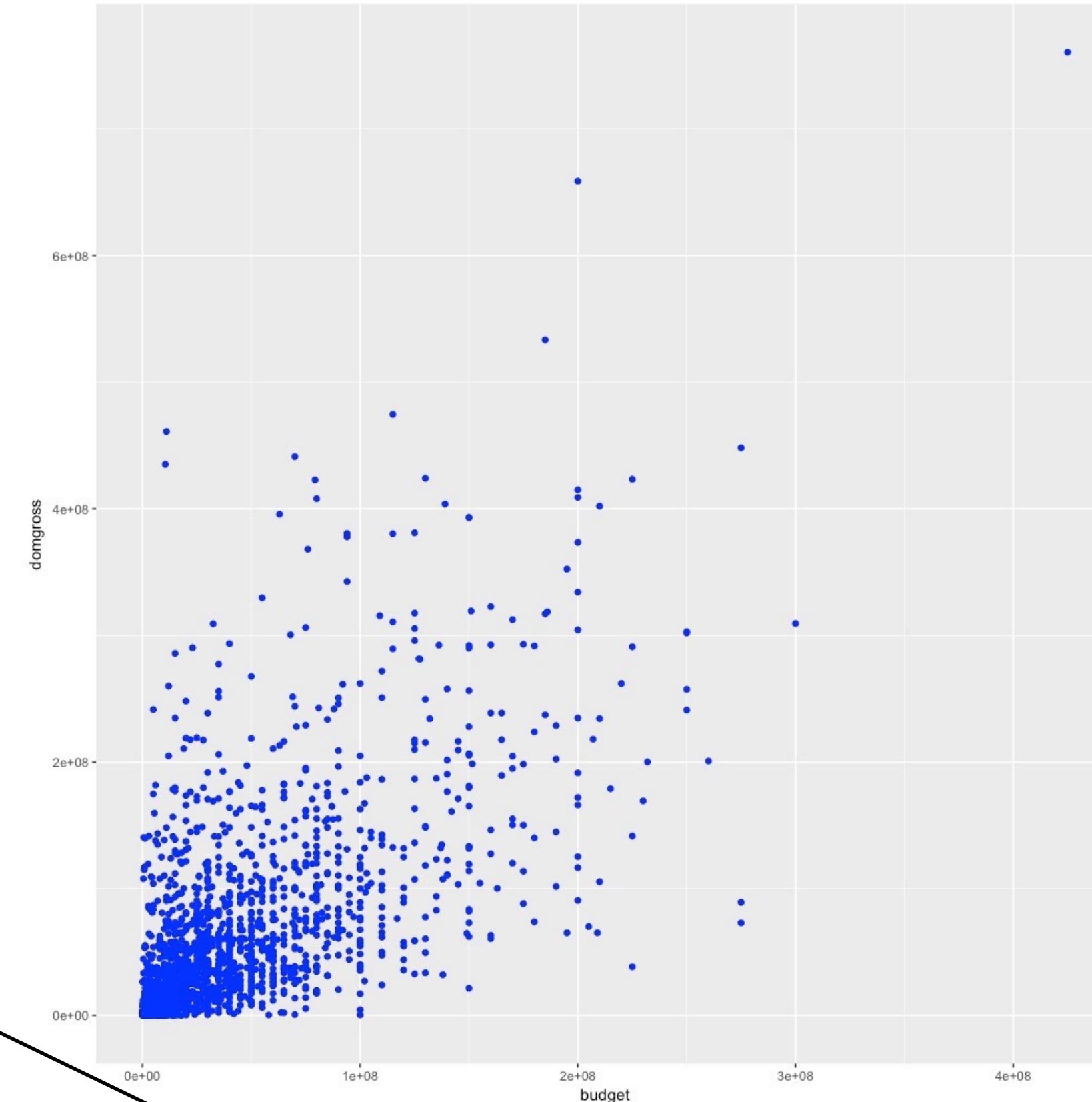


**Inside of aes(): maps an aesthetic to a variable**

```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

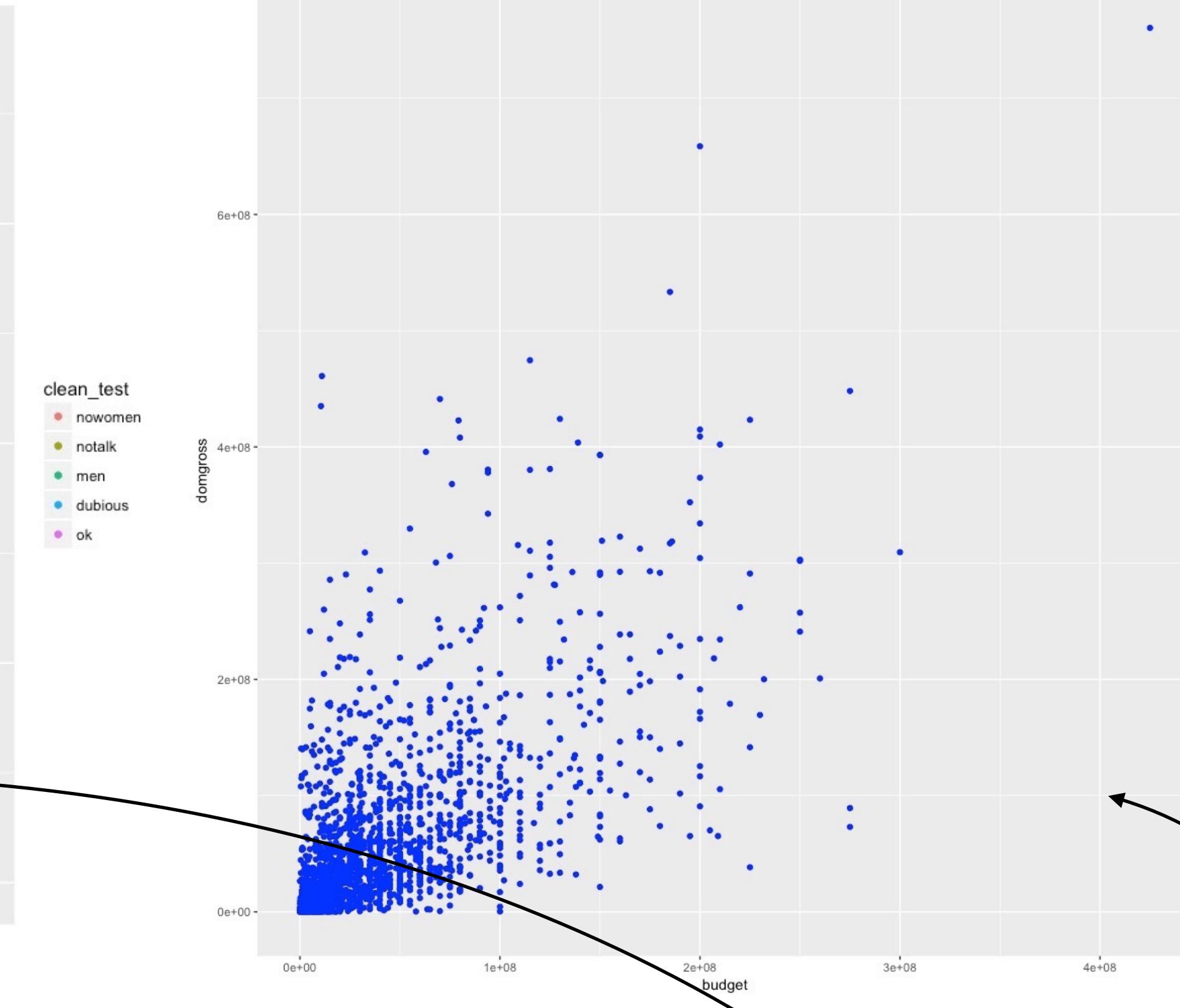
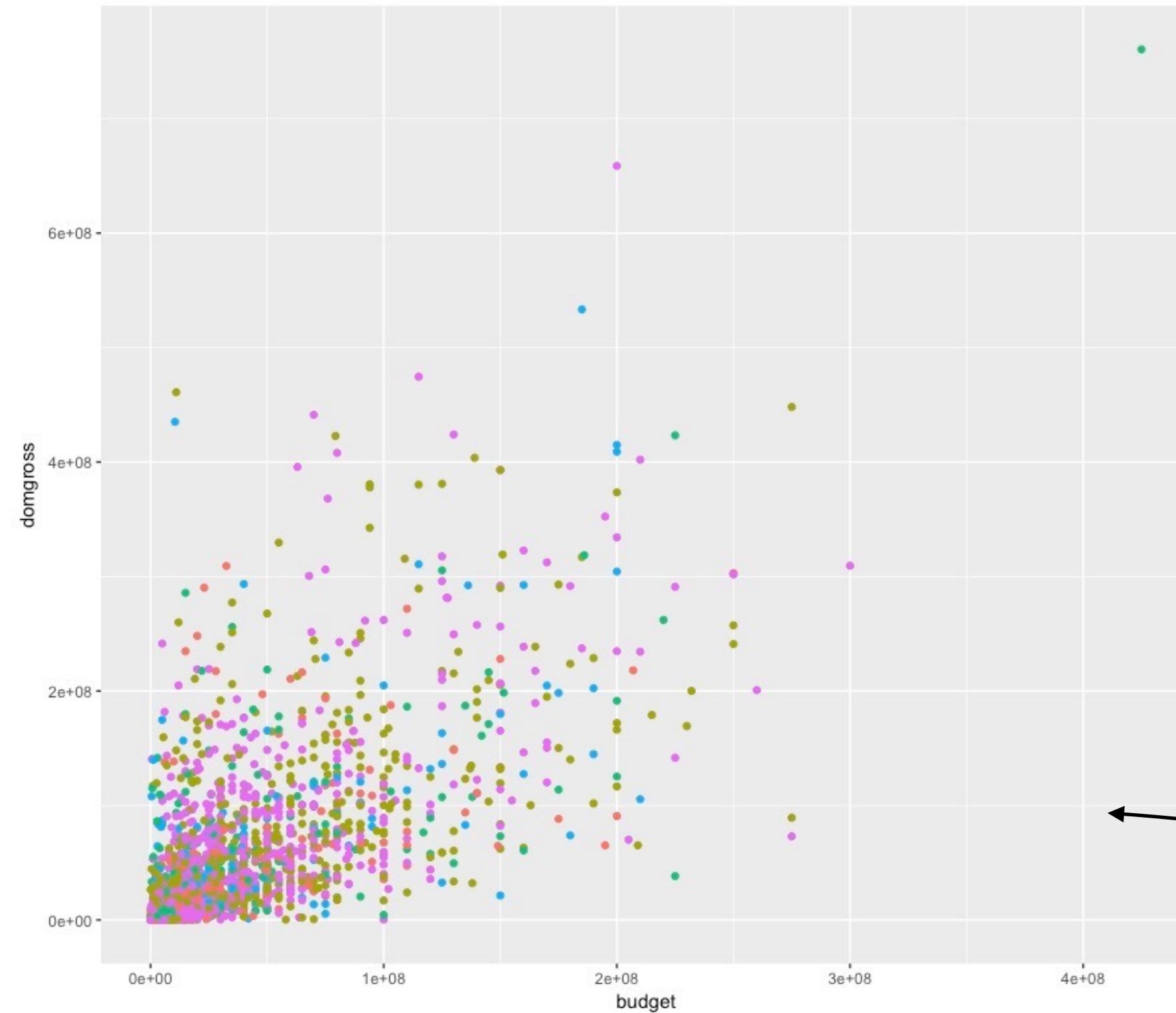


**Outside of aes():** sets  
an aesthetic to a value



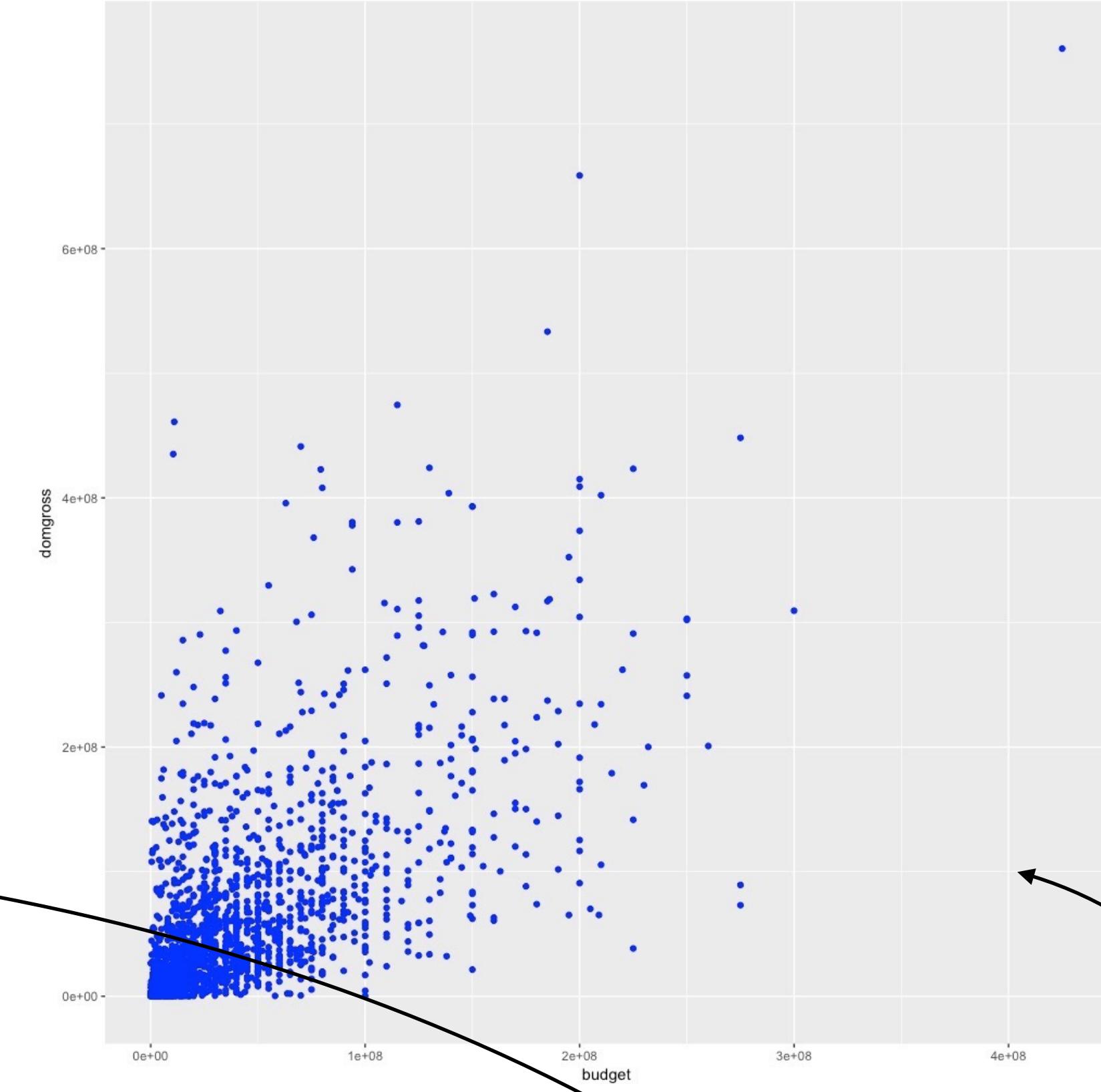
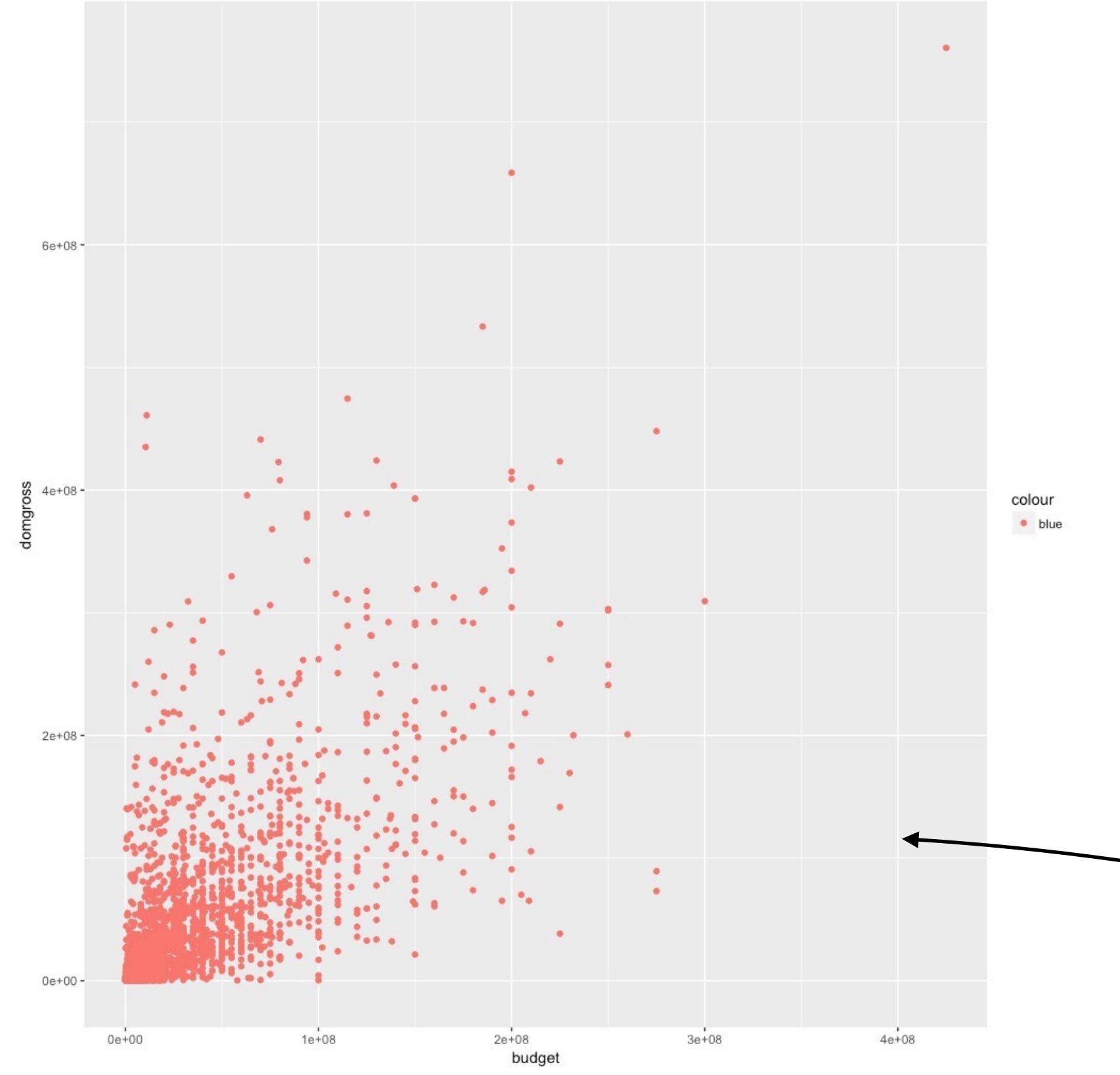
```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```



```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```



```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color="blue"))
```

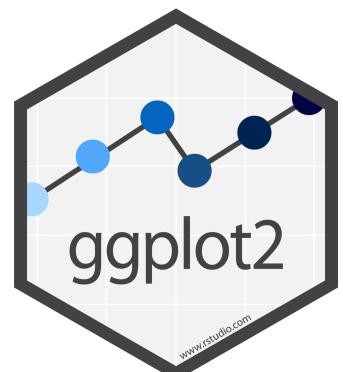
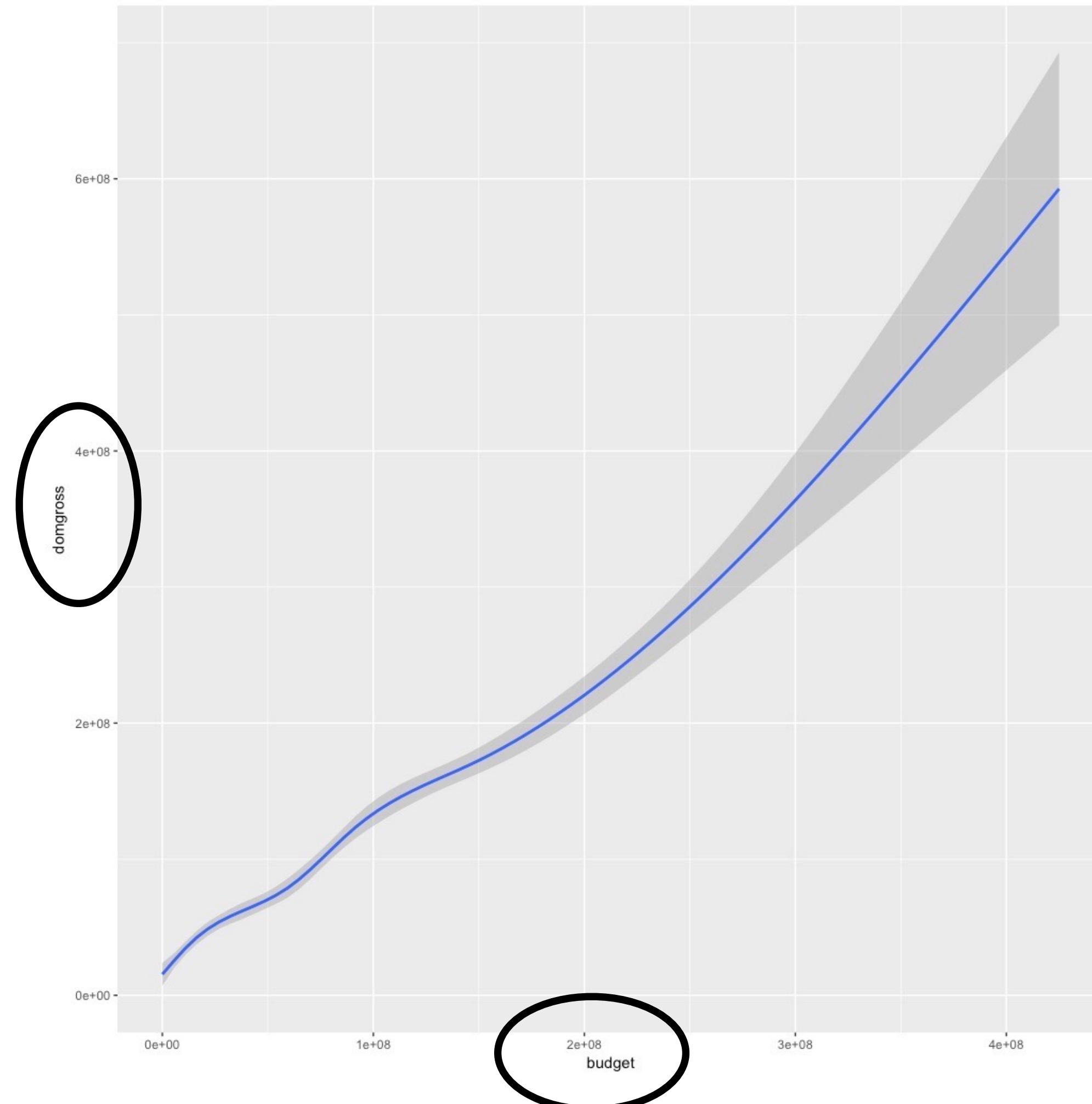
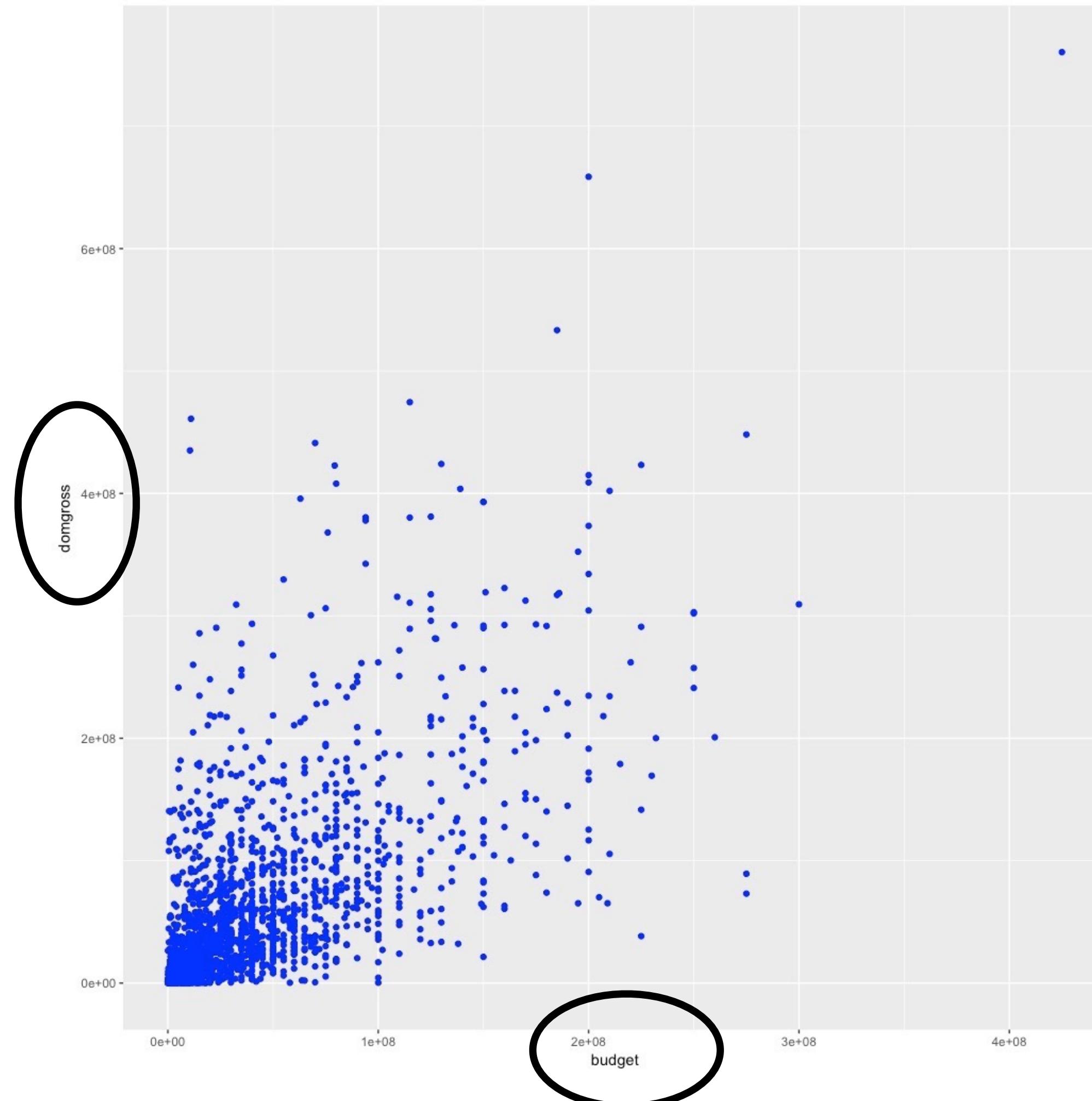
```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

# Geoms



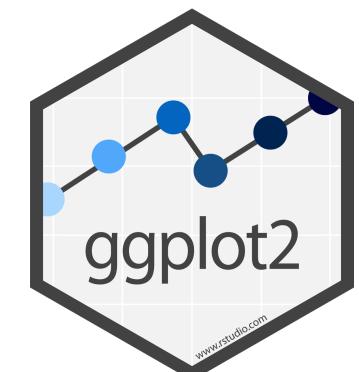
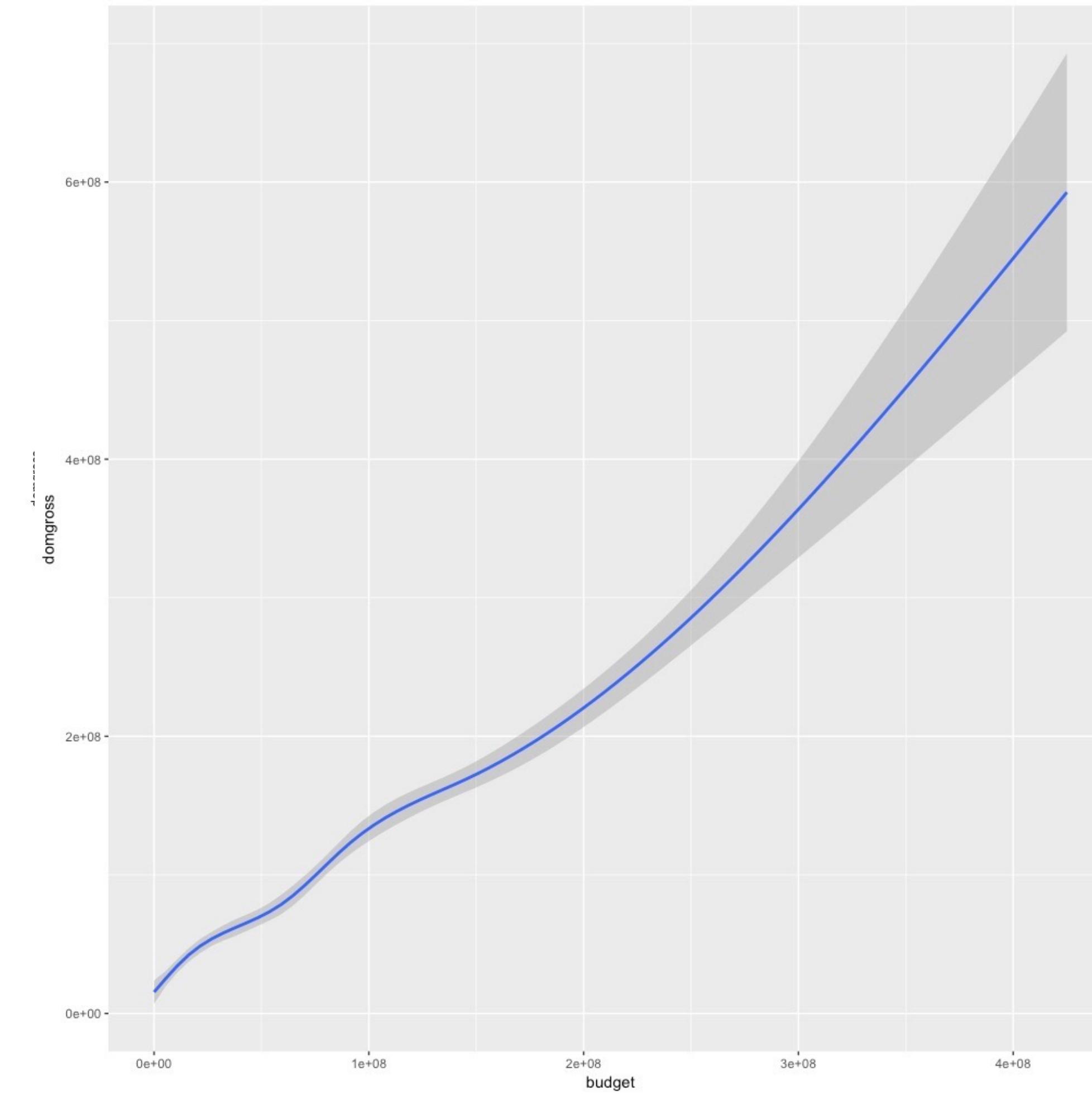
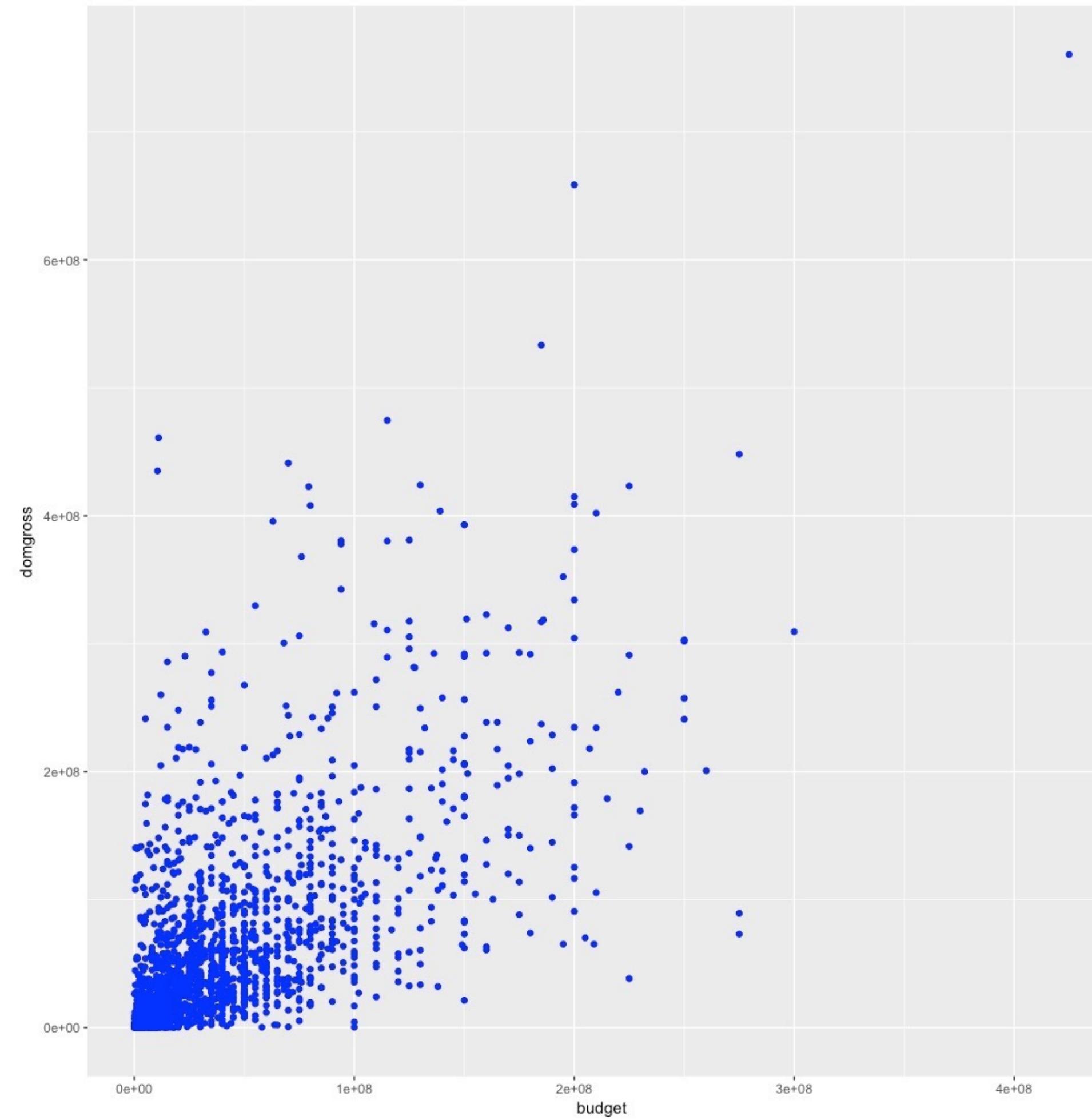
How are these plots similar?

Same: x var , y var , data



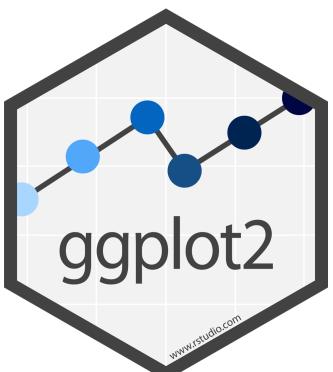
How are these plots different?

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

**ggplot2**

**GRAPHICAL PRIMITIVES**

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

- a + geom\_blank()** (Useful for expanding limits)
- b + geom\_curve(aes(yend = lat + 1, xend = long + 1, curvature = z))** - x, yend, alpha, angle, color, curvature, linetype, size
- a + geom\_path(linewidth = "butt", linejoin = "round", linemiter = 1)** x, y, alpha, color, group, linetype, size
- a + geom\_polygon(aes(group = group))** x, y, alpha, color, fill, group, linetype, size
- b + geom\_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + geom\_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))** - x, ymax, ymin, alpha, color, fill, group, linetype, size

**TWO VARIABLES**

**continuous x , continuous y**

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

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

**continuous bivariate distribution**

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

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

**continuous function**

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

- i + geom\_area()** x, y, alpha, color, fill, linetype, size
- i + geom\_line()** x, y, alpha, color, group, linetype, size
- i + geom\_step(direction = "hv")** x, y, alpha, color, group, linetype, size

**visualizing error**

```
df <- data.frame(grp = c("A", "B"), fit = 4.5, se = 1.2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

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

**maps**

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

- k + geom\_map(aes(map\_id = state), map = map)**
  - \*expand\_limits(x = map\$long, y = map\$lat)**, map\_id, alpha, color, fill, linetype, size

**ONE VARIABLE continuous**

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

- c + geom\_area(stat = "bin")** x, y, alpha, color, fill, linetype, size
- c + geom\_density(kernel = "gaussian")** x, y, alpha, color, fill, group, linetype, size, weight
- c + geom\_dotplot()** x, y, alpha, color, fill
- c + geom\_freqpoly()** x, y, alpha, color, group, linetype, size
- c + geom\_histogram(binwidth = 5)** x, y, alpha, color, fill, linetype, size, weight
- c2 + geom\_qq(aes(sample = hwy))** x, y, alpha, color, fill, linetype, size, weight

**discrete x , discrete y**

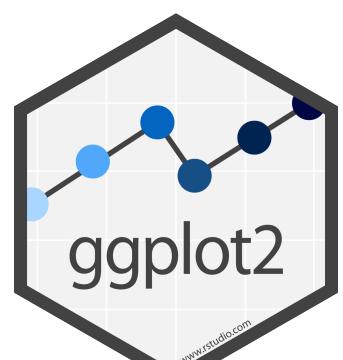
```
g <- ggplot(diamonds, aes(cut, color))
```

- g + geom\_count()** x, y, alpha, color, fill, shape, size, stroke

**THREE VARIABLES**

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

- l + geom\_contour(aes(z = z))** x, y, z, alpha, colour, group, linetype, size, weight
- l + geom\_raster(aes(fill = z))** x, y, alpha, fill
- l + geom\_tile(aes(fill = z))** x, y, alpha, color, fill, linetype, size, width



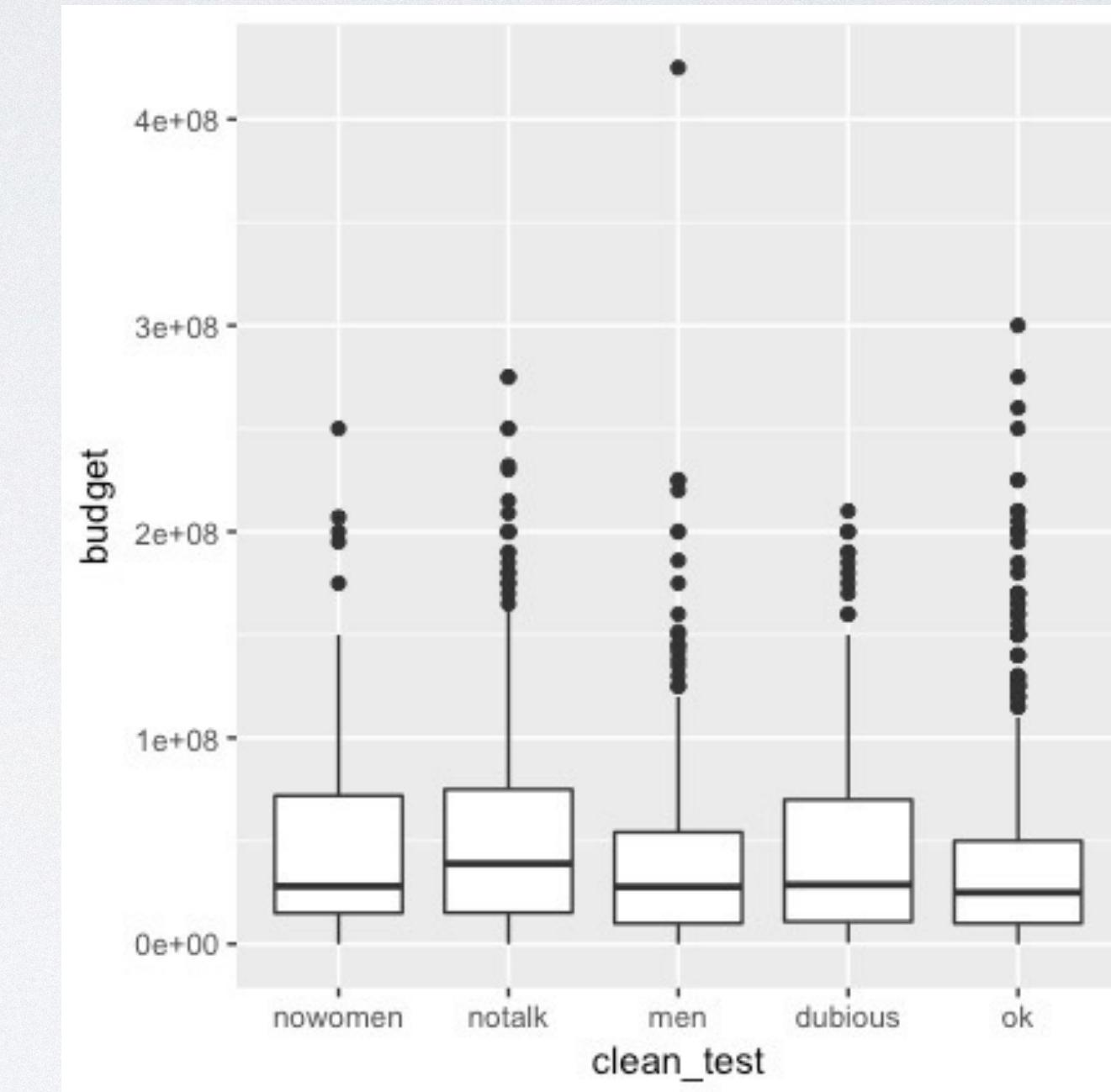
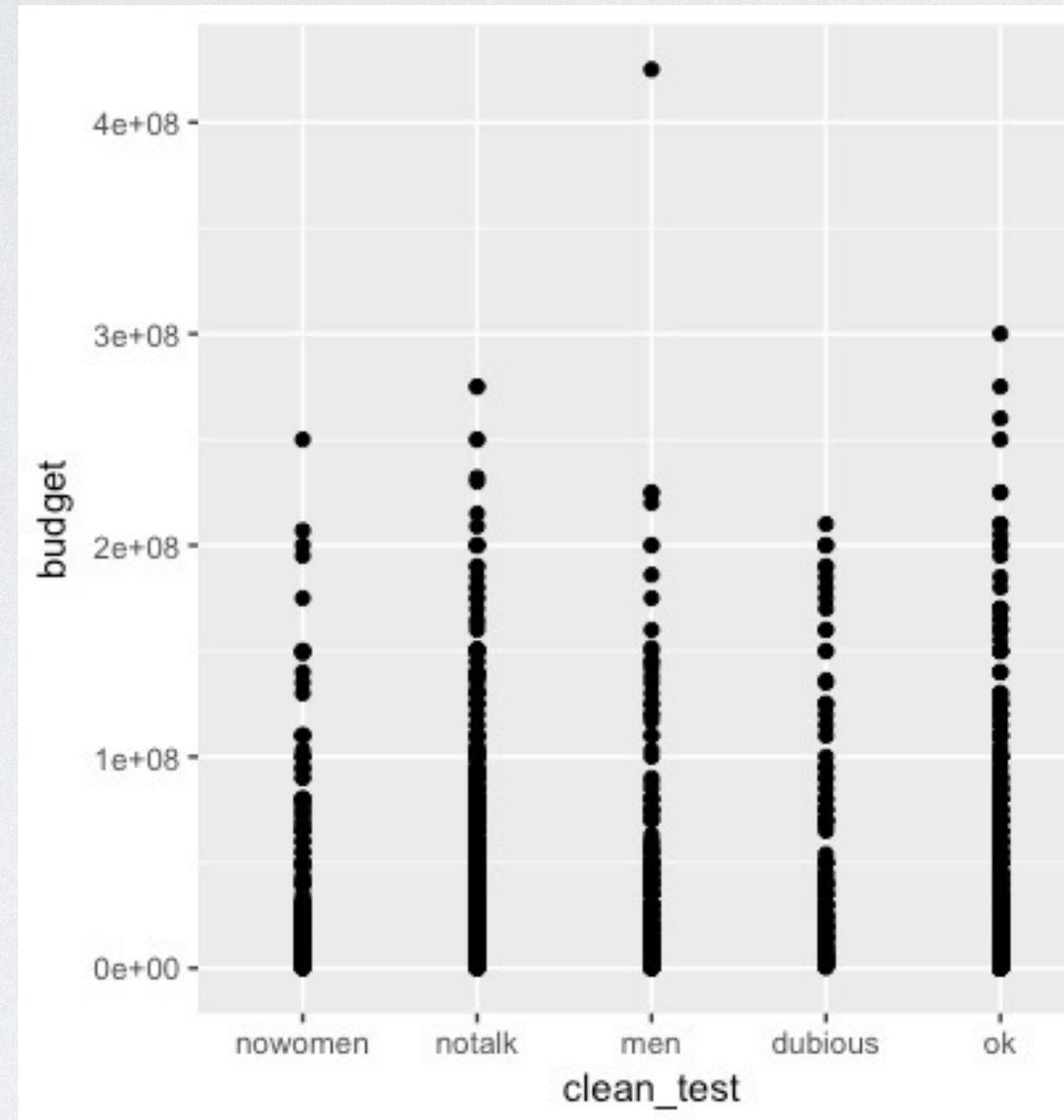
# Your Turn

Pair up.

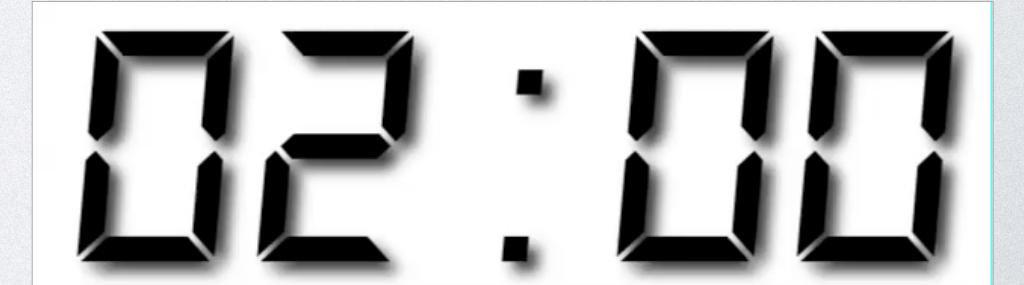


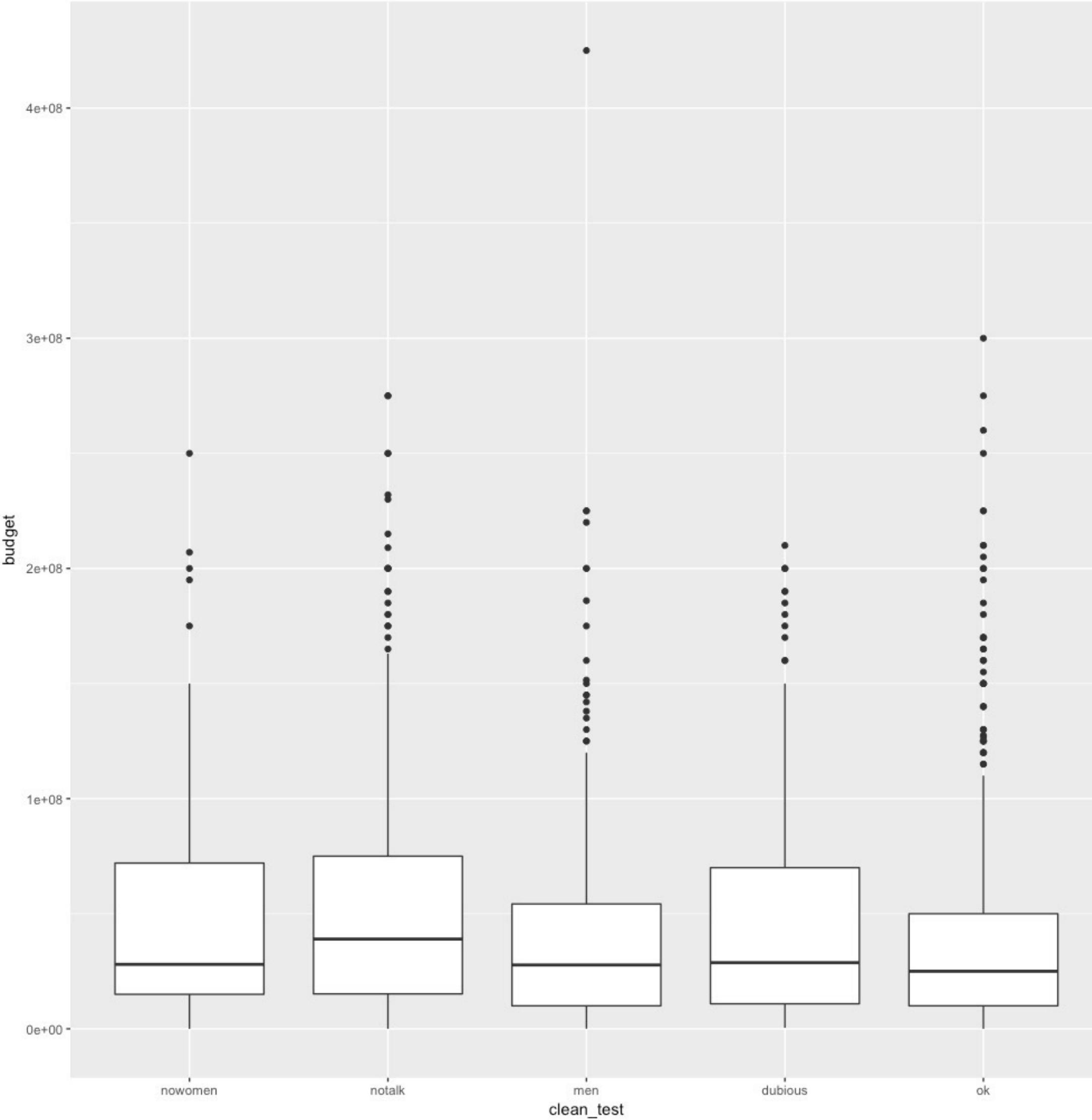
# Your Turn 3

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



```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = clean_test, y = budget))
```

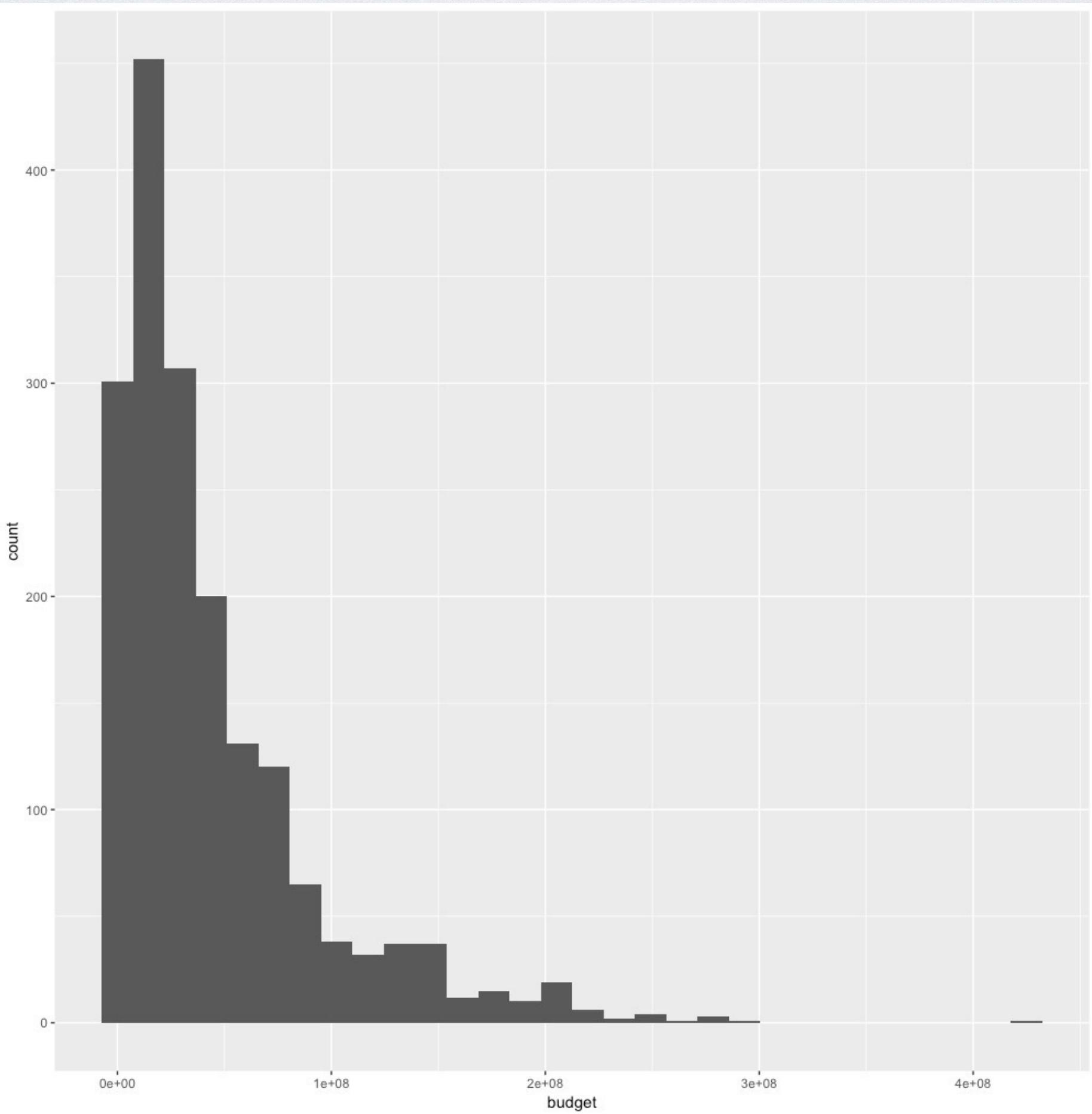


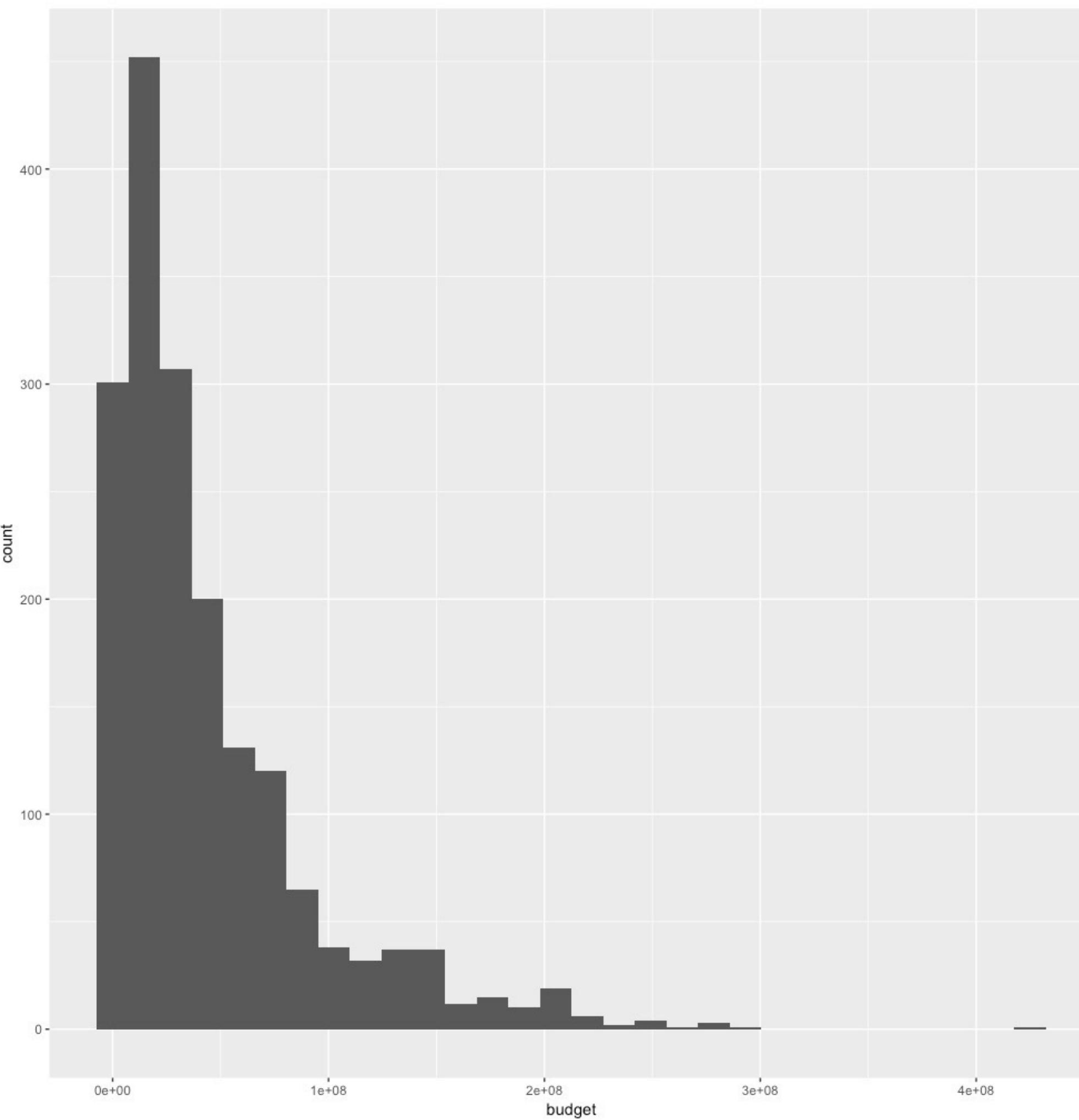


```
ggplot(data = bechdel) +
 geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

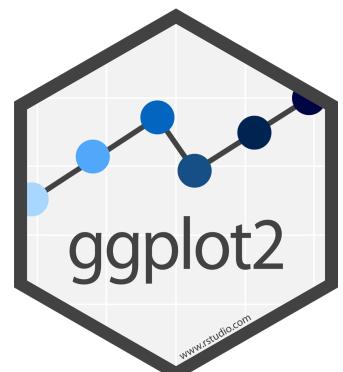
# Your Turn 4

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

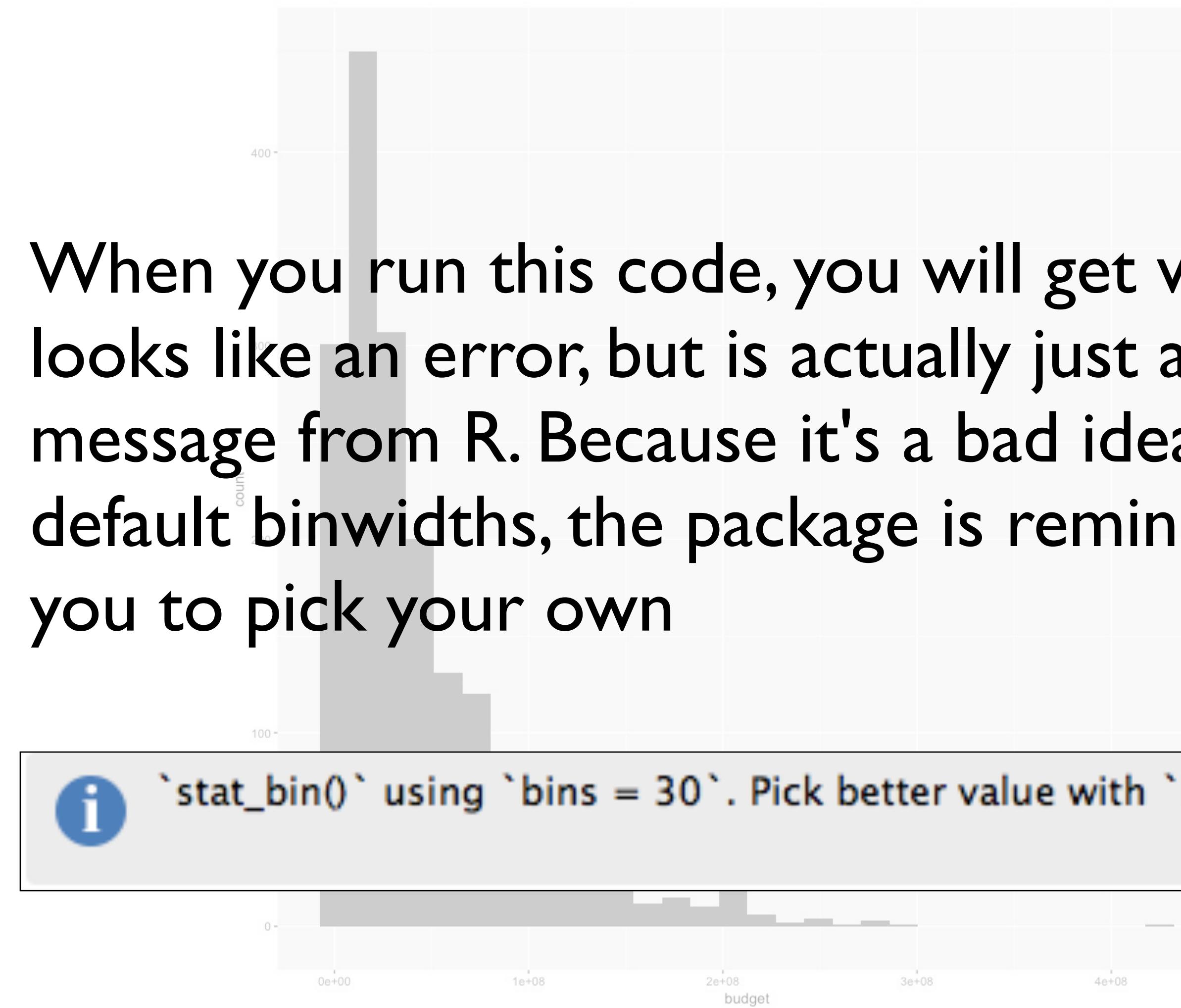




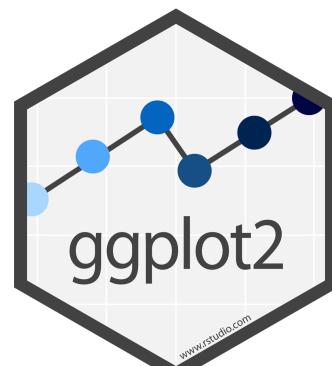
```
ggplot(data = bechdel) +
 geom_histogram(mapping = aes(x = budget))
```



When you run this code, you will get what looks like an error, but is actually just a message from R. Because it's a bad idea to use default binwidths, the package is reminding you to pick your own



```
ggplot(data = bechdel) +
 geom_histogram(mapping = aes(x = budget))
```

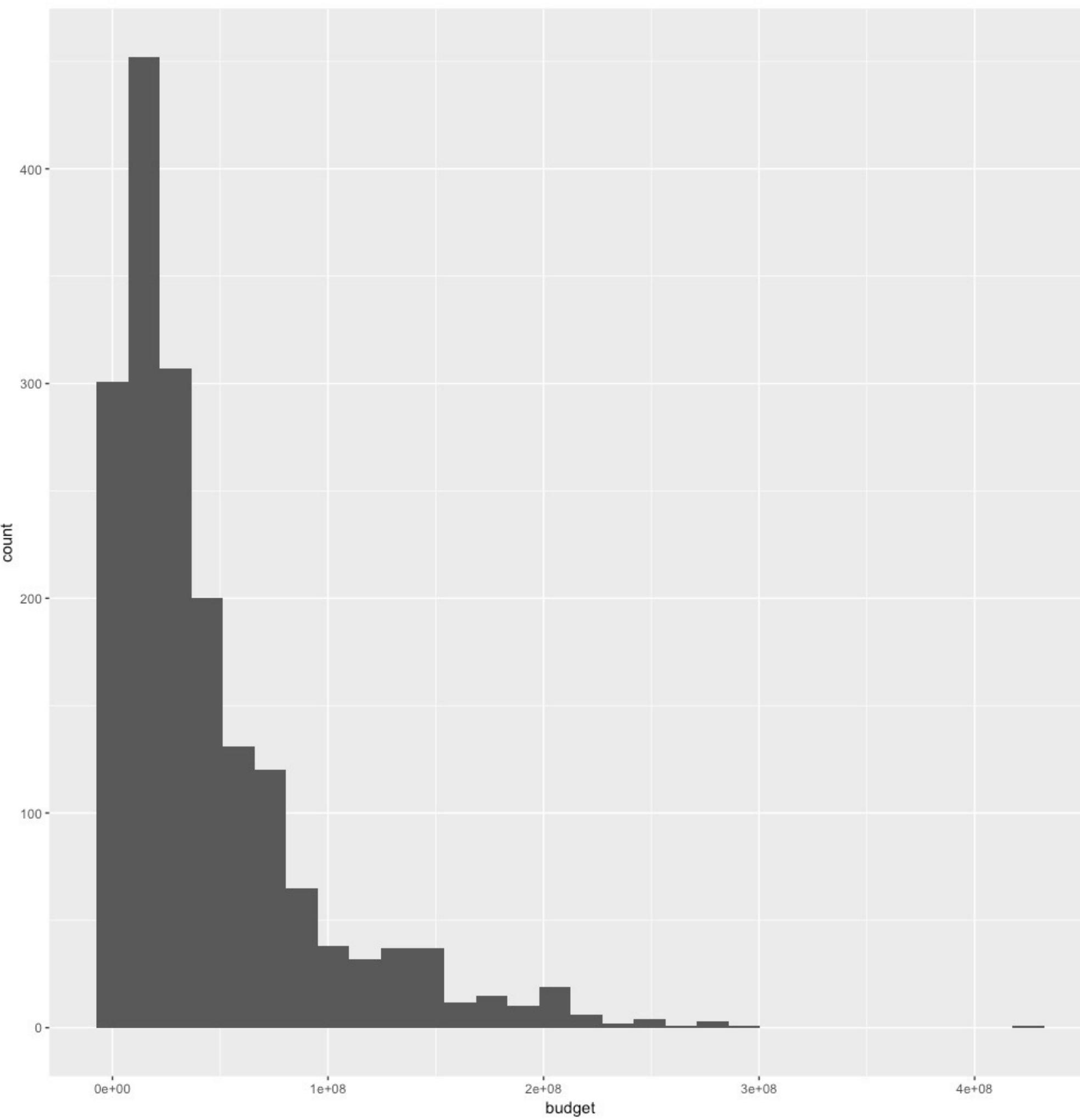


# Your Turn 5

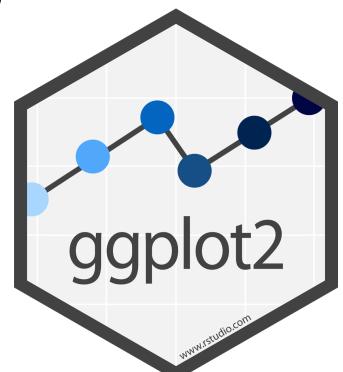
What would be a reasonable  
binwidth for budget?

Try it out



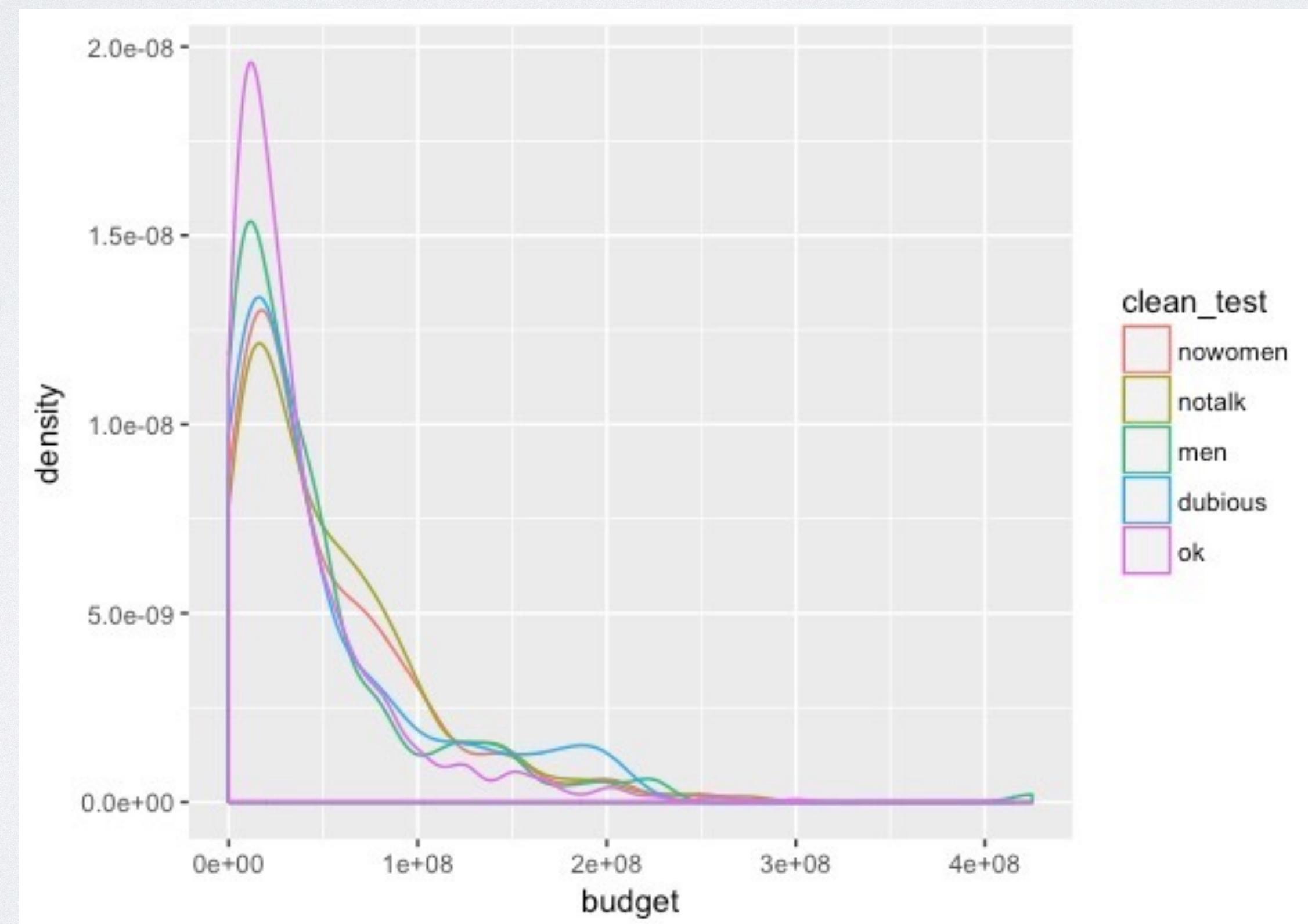


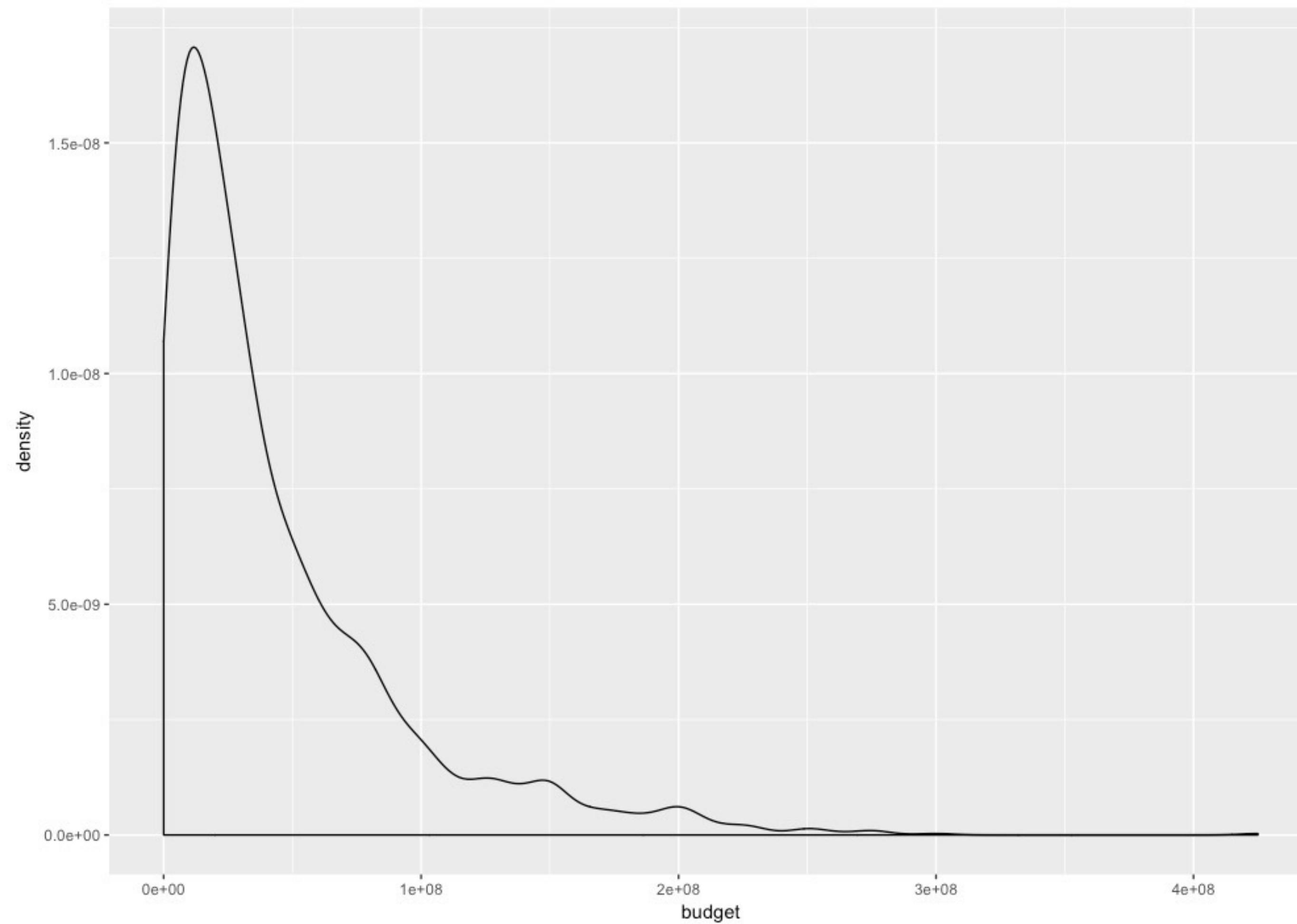
```
ggplot(data = bechdel) +
 geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```



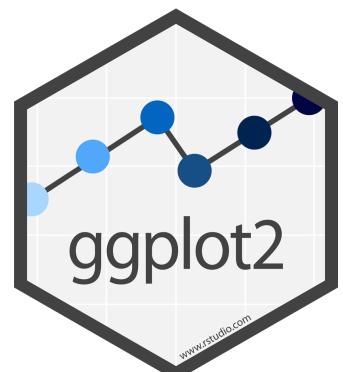
# Your Turn 6

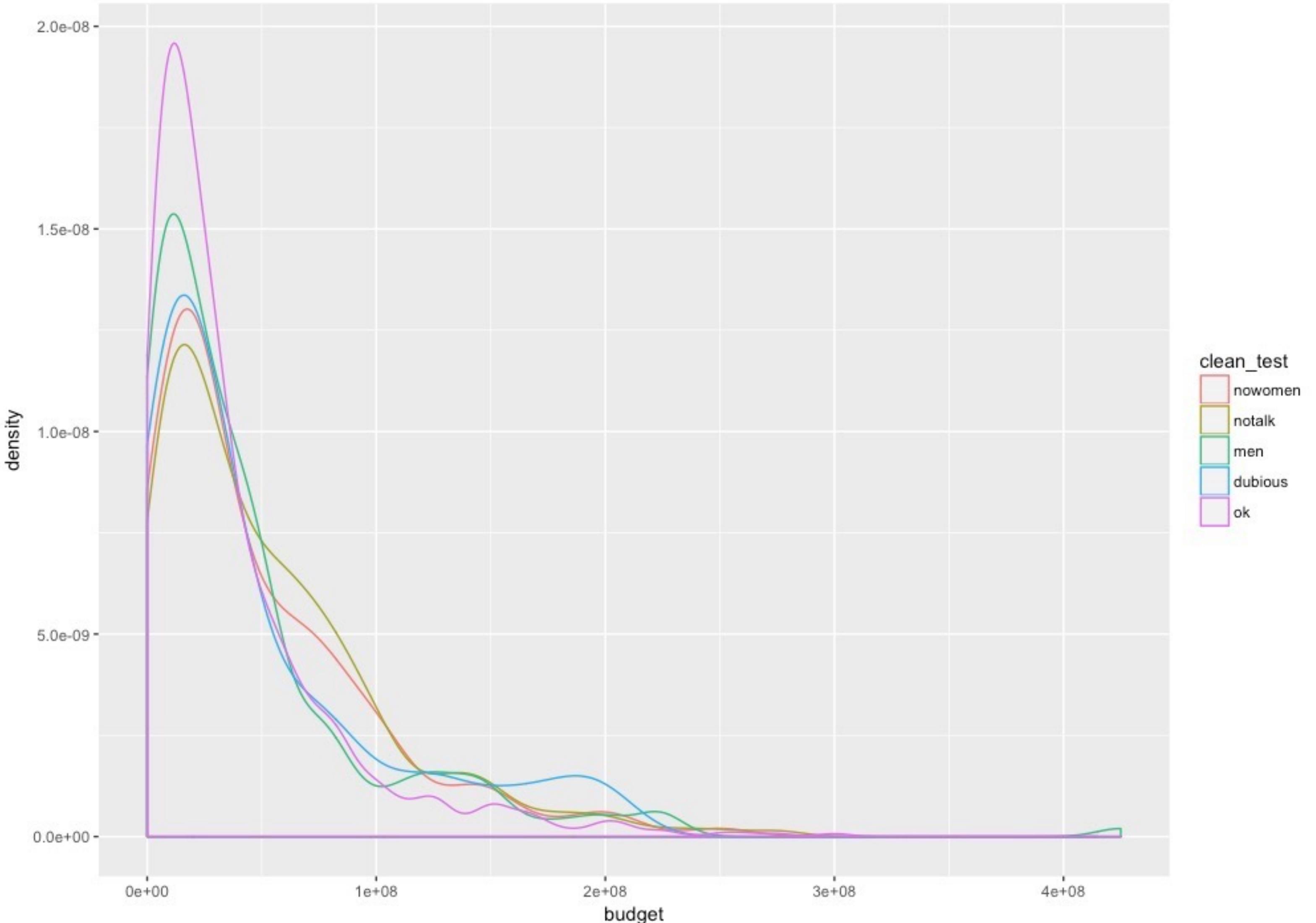
With your partner, make the density plot of **budget** colored by **clean\_test** below. Use the cheatsheet. Try your best guess.



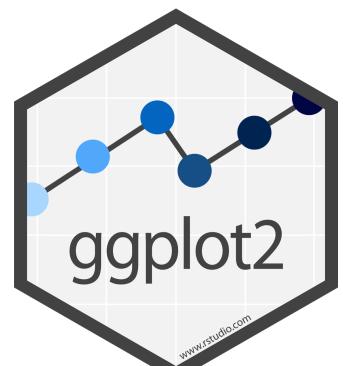


```
ggplot(data = bechdel) +
 geom_density(mapping = aes(x = budget))
```



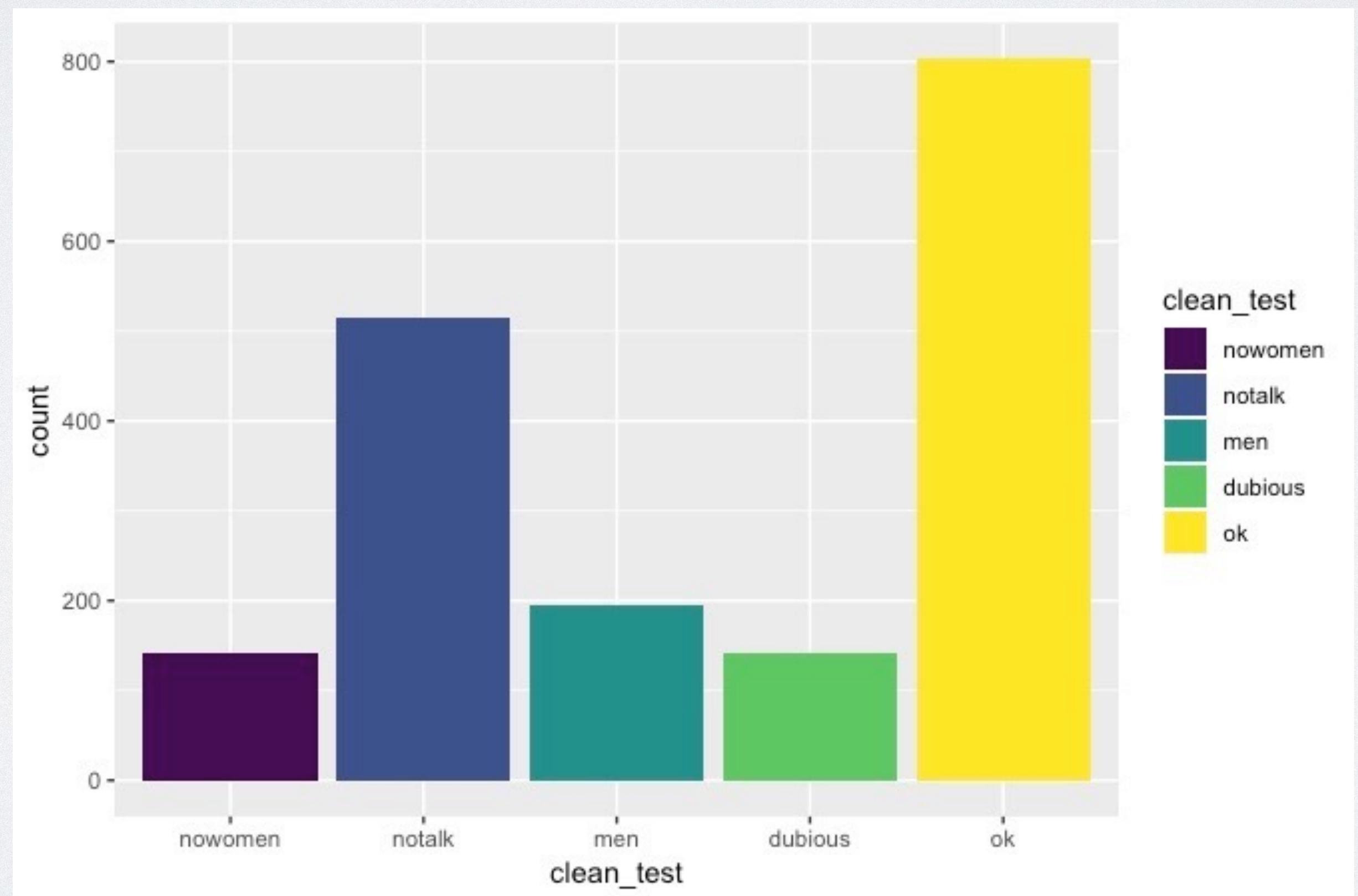


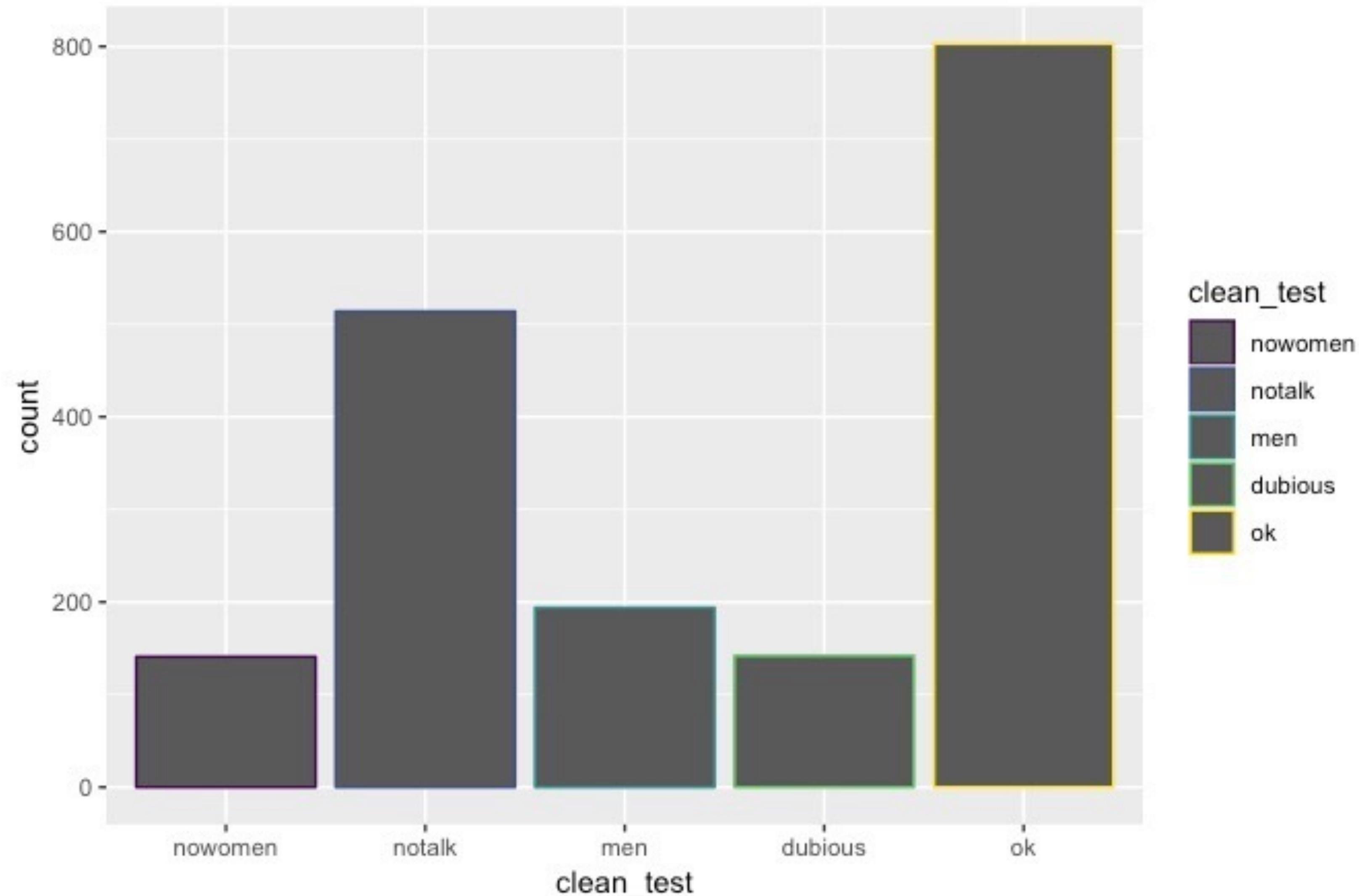
```
ggplot(data = bechdel) +
 geom_density(mapping = aes(x = budget, color=clean_test))
```



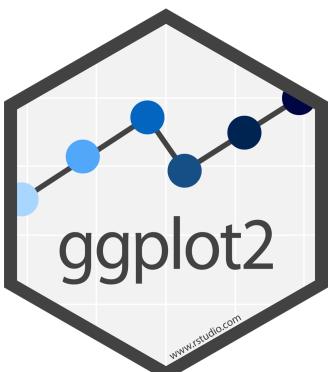
# Your Turn 7

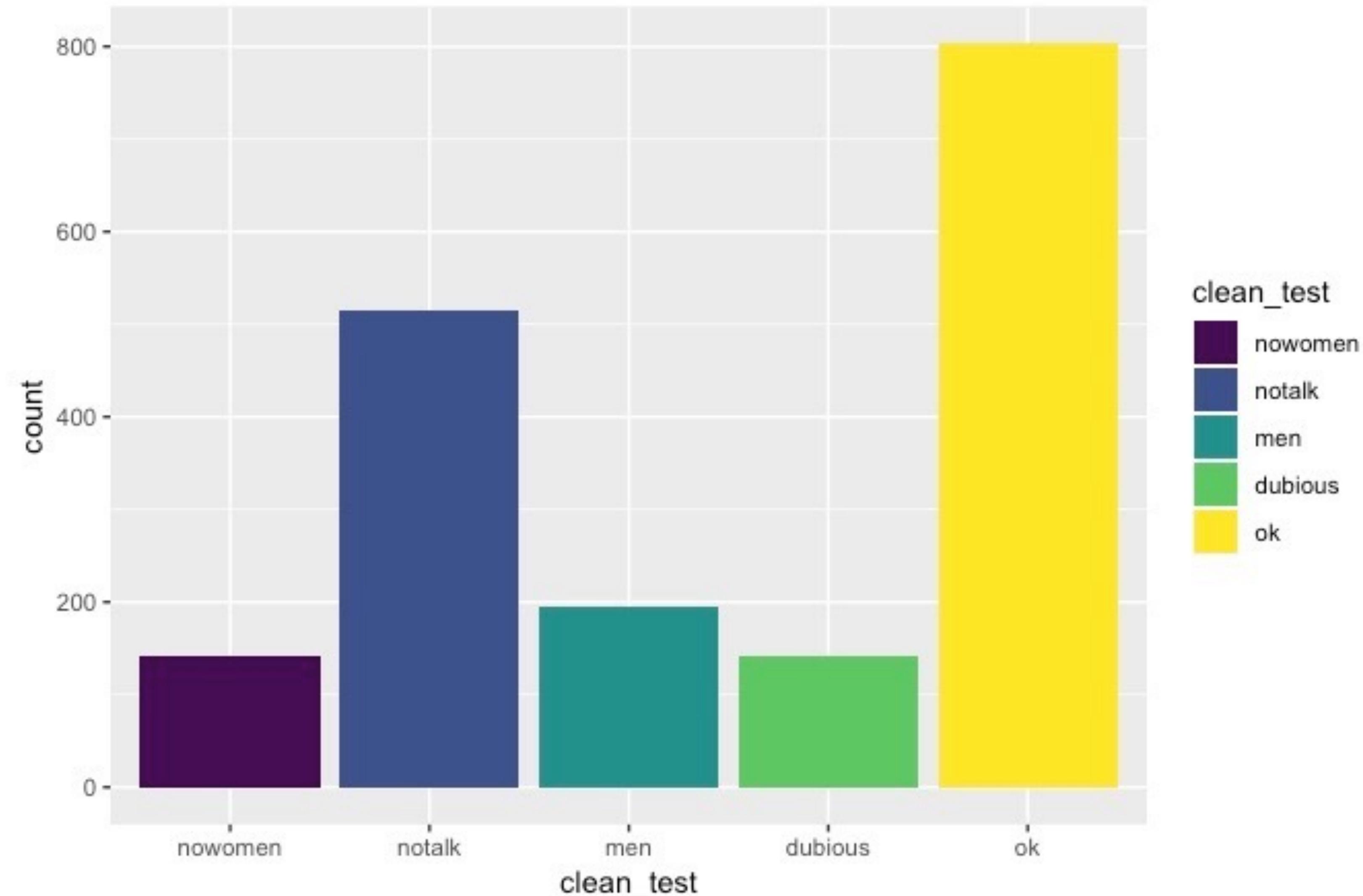
With your partner, make the bar chart of `clean_test` colored by `clean_test` below. Use the cheatsheet. Try your best guess.



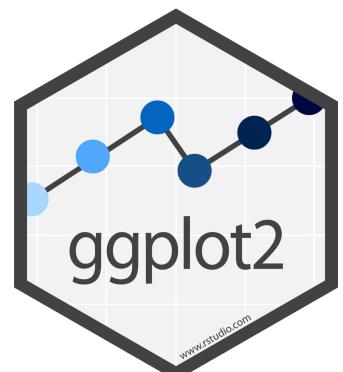


```
ggplot(data=bechdel) +
 geom_bar(aes(x=clean_test, color=clean_test))
```





```
ggplot(data=bechdel) +
 geom_bar(aes(x=clean_test, fill=clean_test))
```

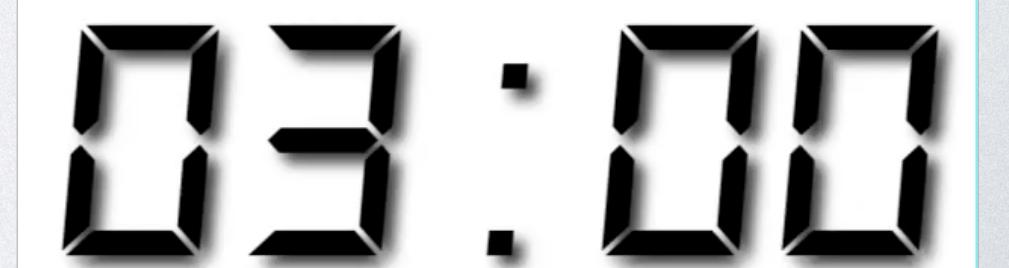


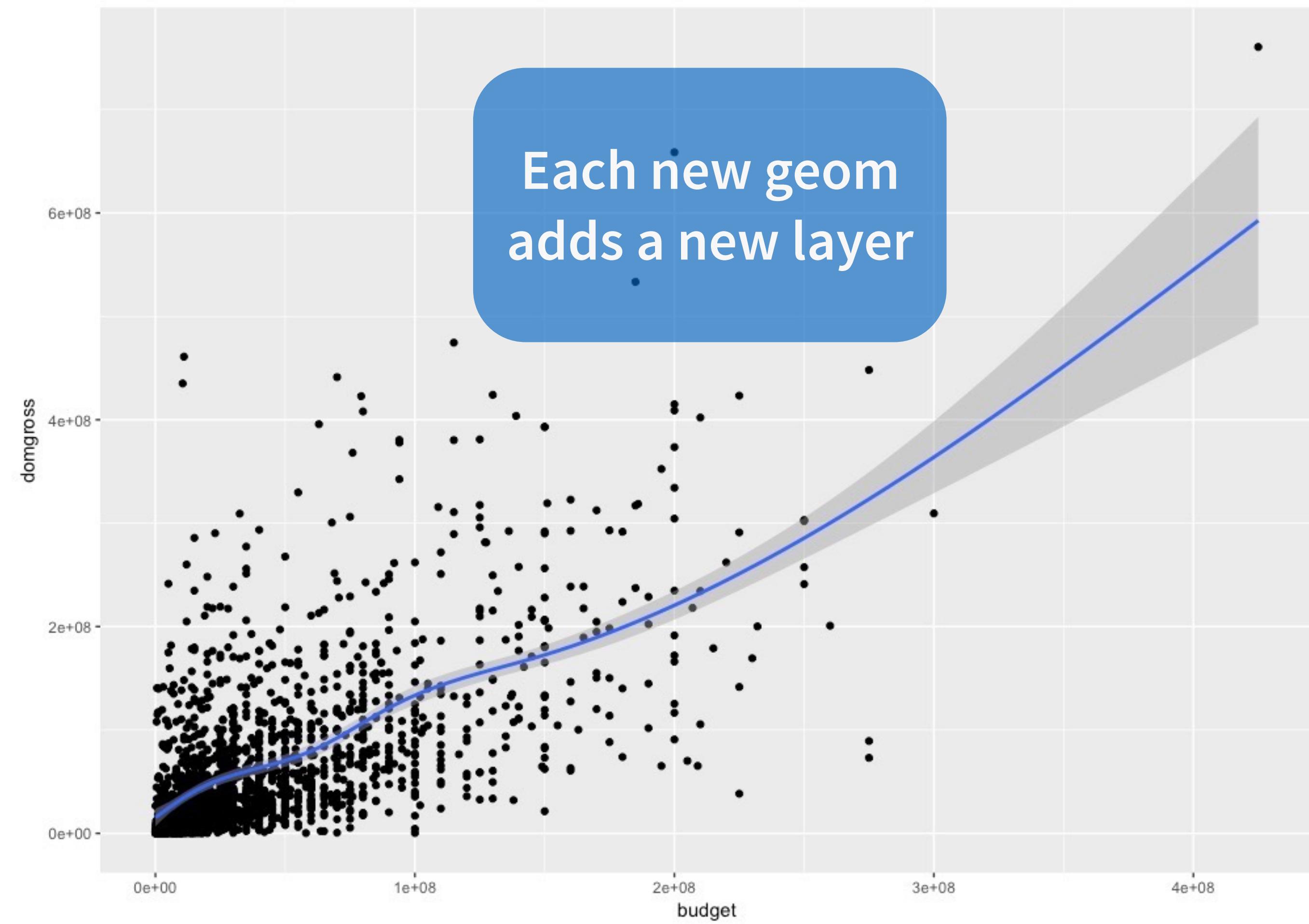
# Your Turn 8

With a partner, predict what this code will do.

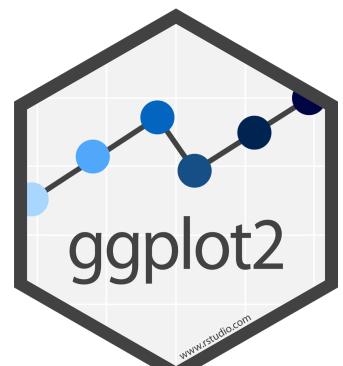
Then run it.

```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross)) +
 geom_smooth(mapping = aes(x = budget, y = domgross))
```



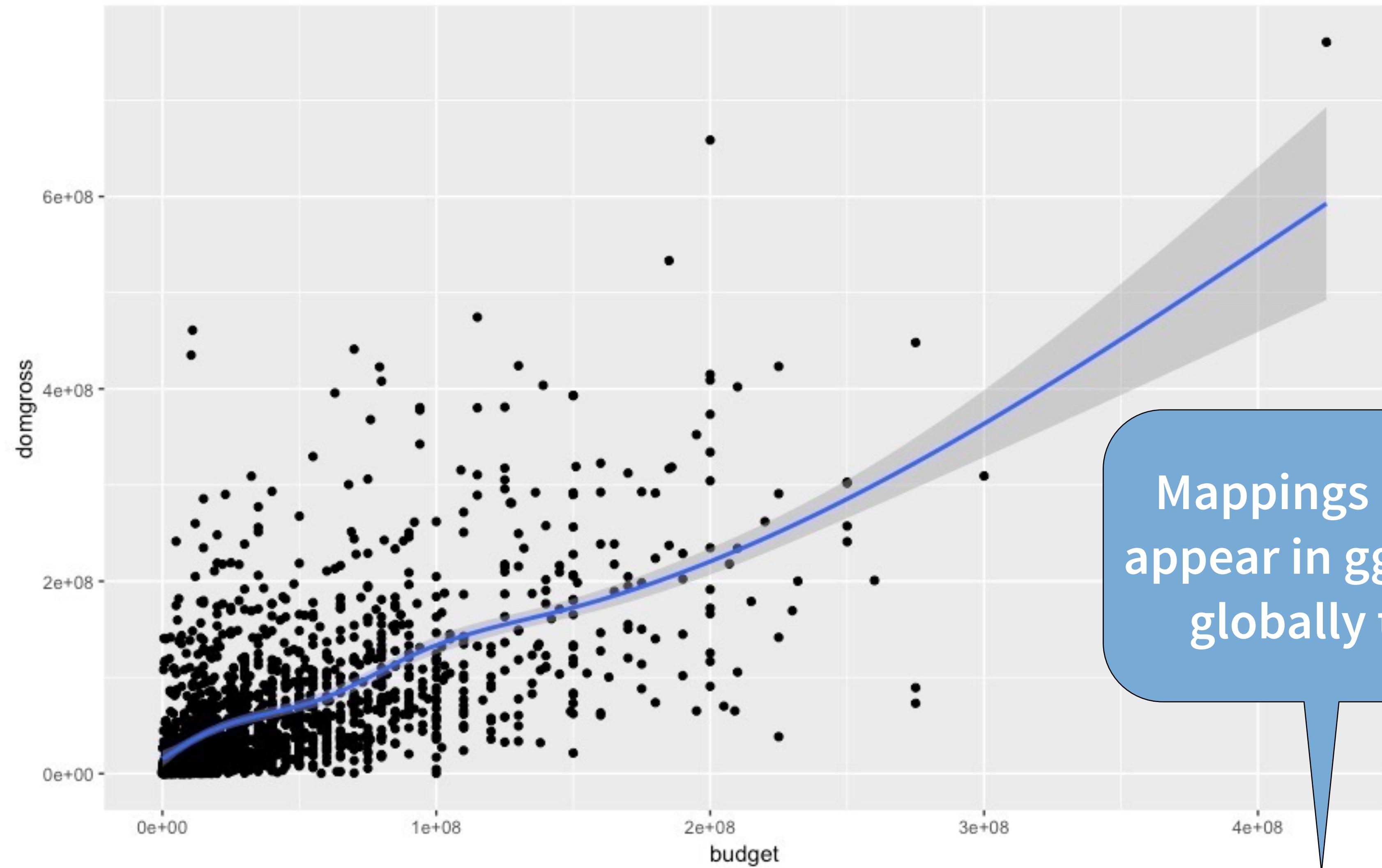


```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross)) +
 geom_smooth(mapping = aes(x = budget, y = domgross))
```



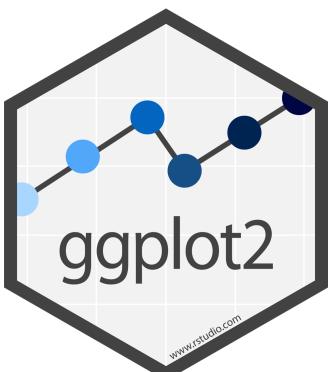
# global vs. local

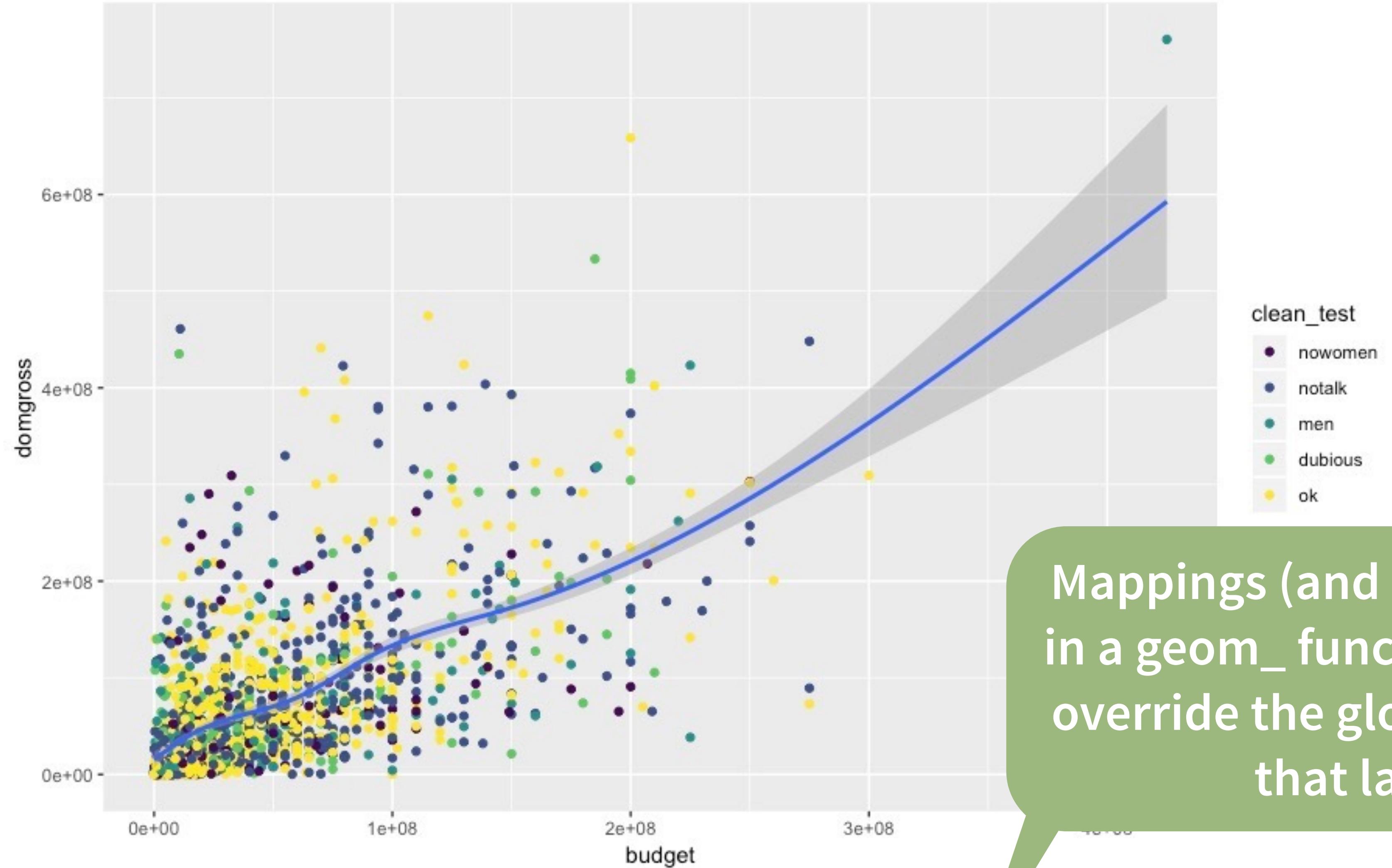
R



```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
 geom_point() +
 geom_smooth()
```

Adapted from [Master the Tidyverse, CC BY RStudio](#)

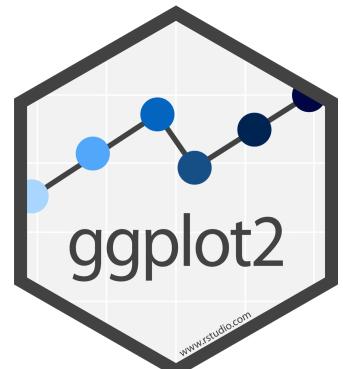


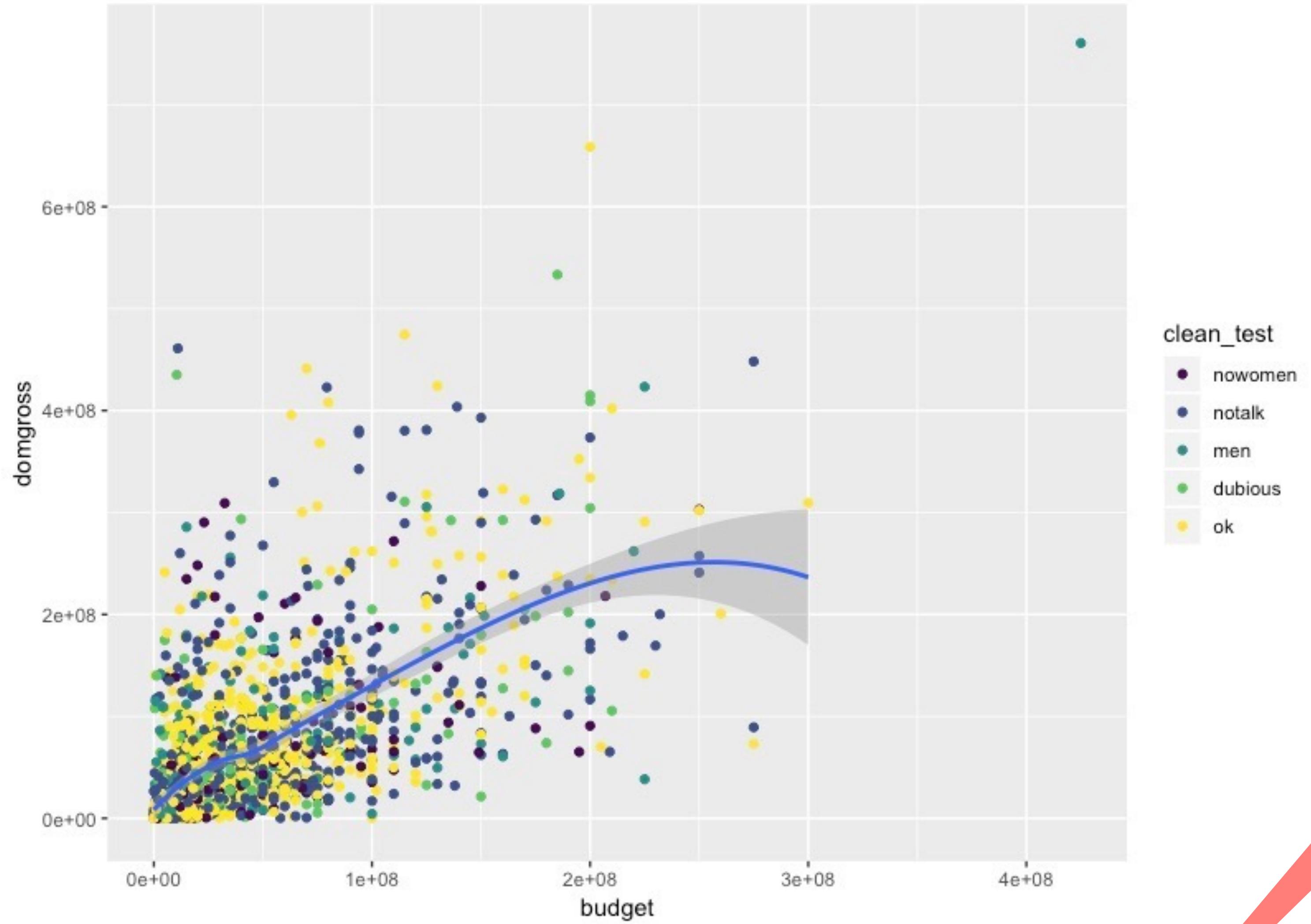


Mappings (and data) that appear in a `geom_` function will add to or override the global mappings for that layer only

```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
 geom_point(mapping = aes(color = clean_test)) +
 geom_smooth()
```

Adapted from [Master the Tidyverse, CC BY RStudio](#)

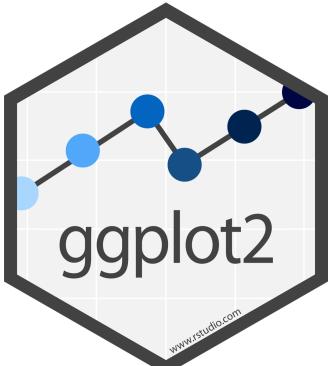




data can also be set locally or globally

```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
 geom_point(mapping = aes(color = clean_test)) +
 geom_smooth(data = filter(bechdel, clean_test == "ok"))
```

Adapted from [Master the Tidyverse, CC BY RStudio](#)



# Saving graphs



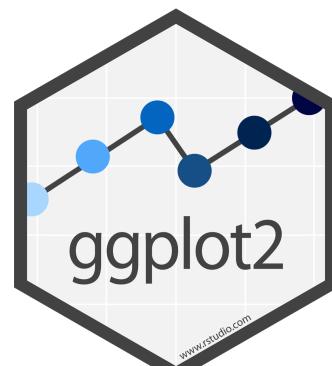
# R Notebook Preview

The easiest way to save all your work (including graphs) is to include it in an R Notebook. While you're working, you can view your notebook by clicking "Preview"

The screenshot shows the RStudio interface with an R Notebook file open. The file contains the following code:

```
1 ---
2 title: "Visualization"
3 output: html_notebook
4 editor_options:
5 chunk_output_type: inline
6 ---
7 |
8 ## Setup
9
10 The first chunk in an R Notebook is usually
11 titled "setup," and by convention includes the
12 R packages you want to load. Remember, in order
13 to use an R package you have to run some
14 `library()` code every session. Execute these
15 lines of code to load the packages.
16
17 ## Bechdel test data
```

A red circle highlights the "Preview" button in the toolbar. To the right, the "Plots" tab is selected, displaying a scatter plot of movie budget versus domestic gross earnings. The x-axis is labeled "budget" and ranges from 0e+00 to 4e+08. The y-axis is labeled "domgross" and ranges from 0e+00 to 6e+08. A blue regression line with a shaded confidence interval is overlaid on the data points.



# R Notebook Preview

Now, you'll see a beautifully typeset version of what you've done!

The screenshot shows the RStudio interface with an R Notebook open. The left pane displays the R Markdown code, and the right pane shows the generated HTML output.

**Code (Left Pane):**

```
1 ---
2 title: "Visualization"
3 output: html_notebook
4 editor_options:
5 chunk_output_type: inline
6 ---
7 |
8 ## Setup
9
10 The first chunk in an R Notebook is usually
11 titled "setup," and by convention includes the
12 R packages you want to load. Remember, in order
13 to use an R package you have to run some
14 library() code every session.
15
16 geom_point(aes(budget, domgross))
17 + geom_smooth(aes(budget, domgross))
18 `geom_smooth()` using method = 'gam'
19 Warning messages:
20 1: Removed 17 rows containing non-finite values
21 (stat_smooth).
22 2: Removed 17 rows containing missing values
23 (geom_point).
24 > |
```

**Preview (Right Pane):**

## Visualization

### Setup

The first chunk in an R Notebook is usually titled "setup," and by convention includes the R packages you want to load. Remember, in order to use an R package you have to run some `library()` code every session. Execute these lines of code to load the packages.

```
library(ggplot2)
library(fivethirtyeight)
```

### Bechdel test data

We're going to start by playing with data collected by the website FiveThirtyEight on movies and the Bechdel test.

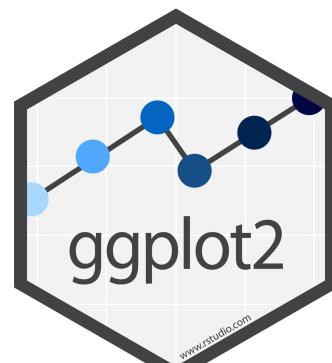
To begin, let's just preview our data. There are a couple ways to do that. One is just to type the name of the data and execute it like a piece of code.

```
bechdel
```

Notice that you can page through to see more of the dataset.

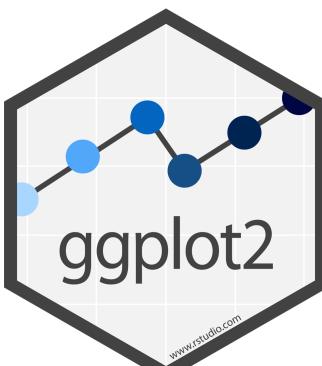
Sometimes, people prefer to see their data in a more spreadsheet-like format, and RStudio provides a way to do that. Go to the Console and type `View(bechdel)` to see the data preview.

(An aside— `View` is a special function. Since it makes something happen in the RStudio



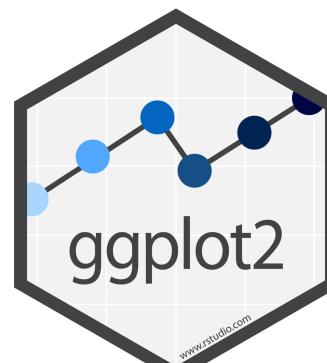
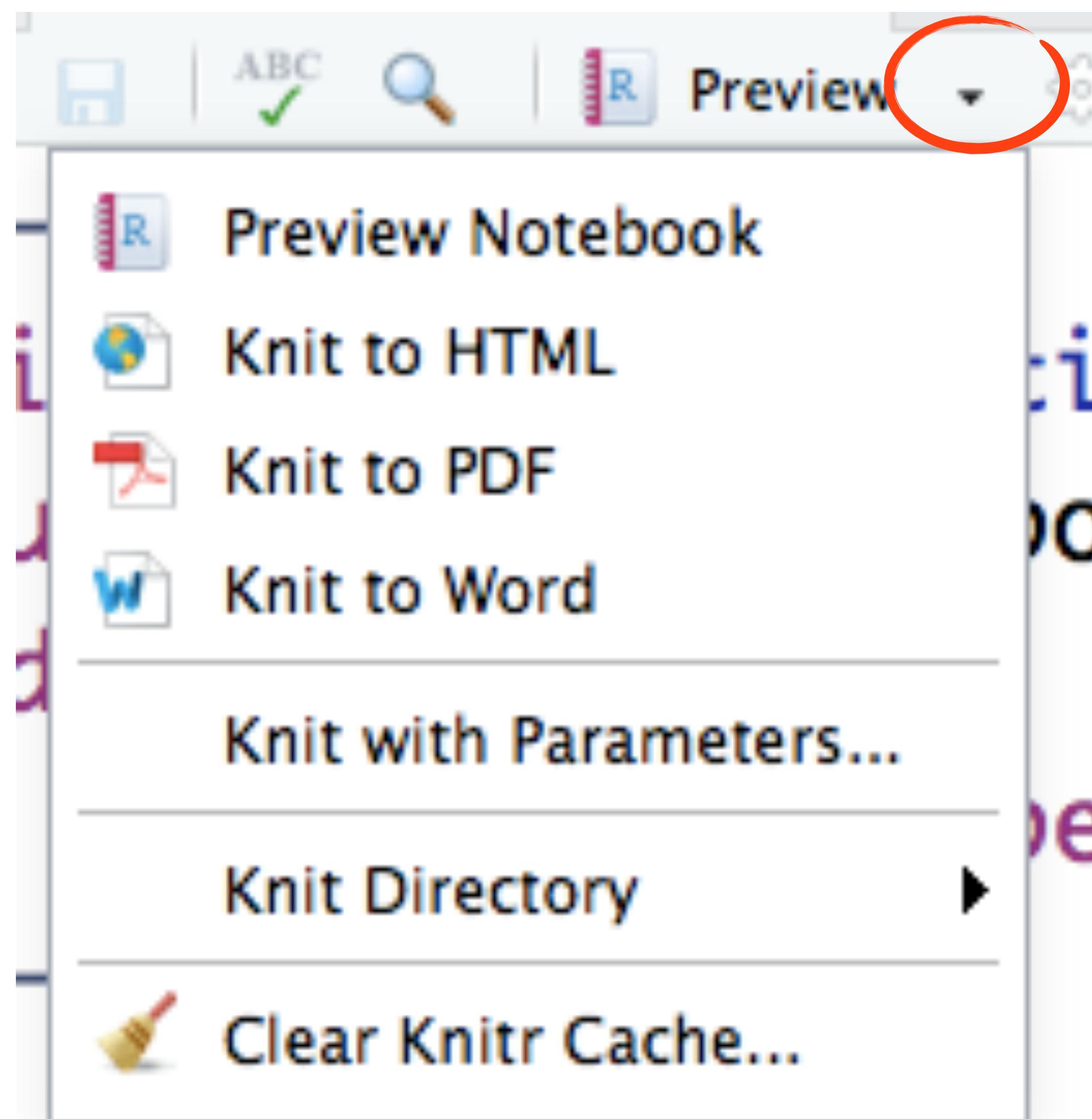
# Sharing your work

The Preview is something only available to you, but RStudio automatically creates a file you can share with others when you save an R Notebook. The file it creates is an HTML file, and it has a name that corresponds to your Rmd



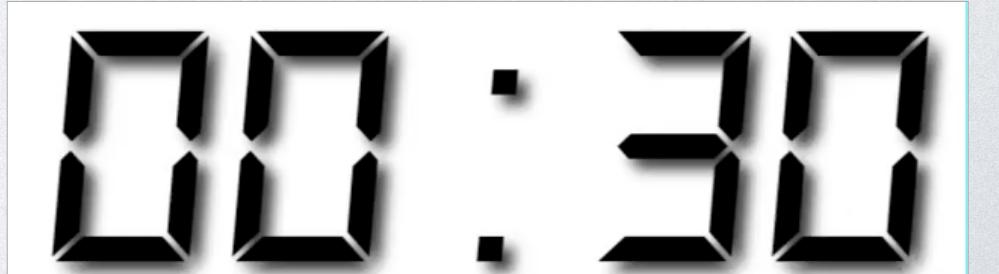
# "knitting" a document

While RStudio automatically generates an HTML file of your work, you might want a different format.  
Clicking the down arrow next to Preview lets you see other options.



# Your Turn

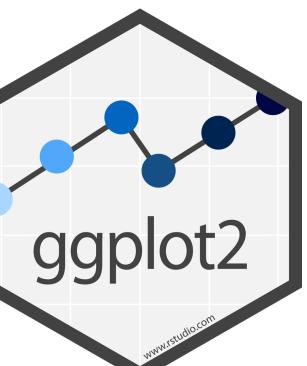
Locate the 01-Visualize.nb.html file in your Files pane. It will be located in your **working directory**



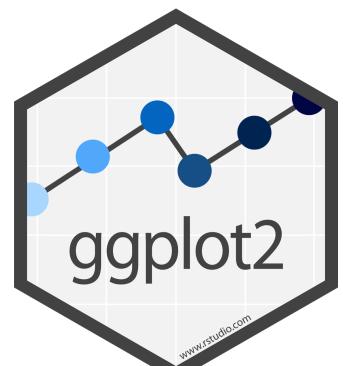
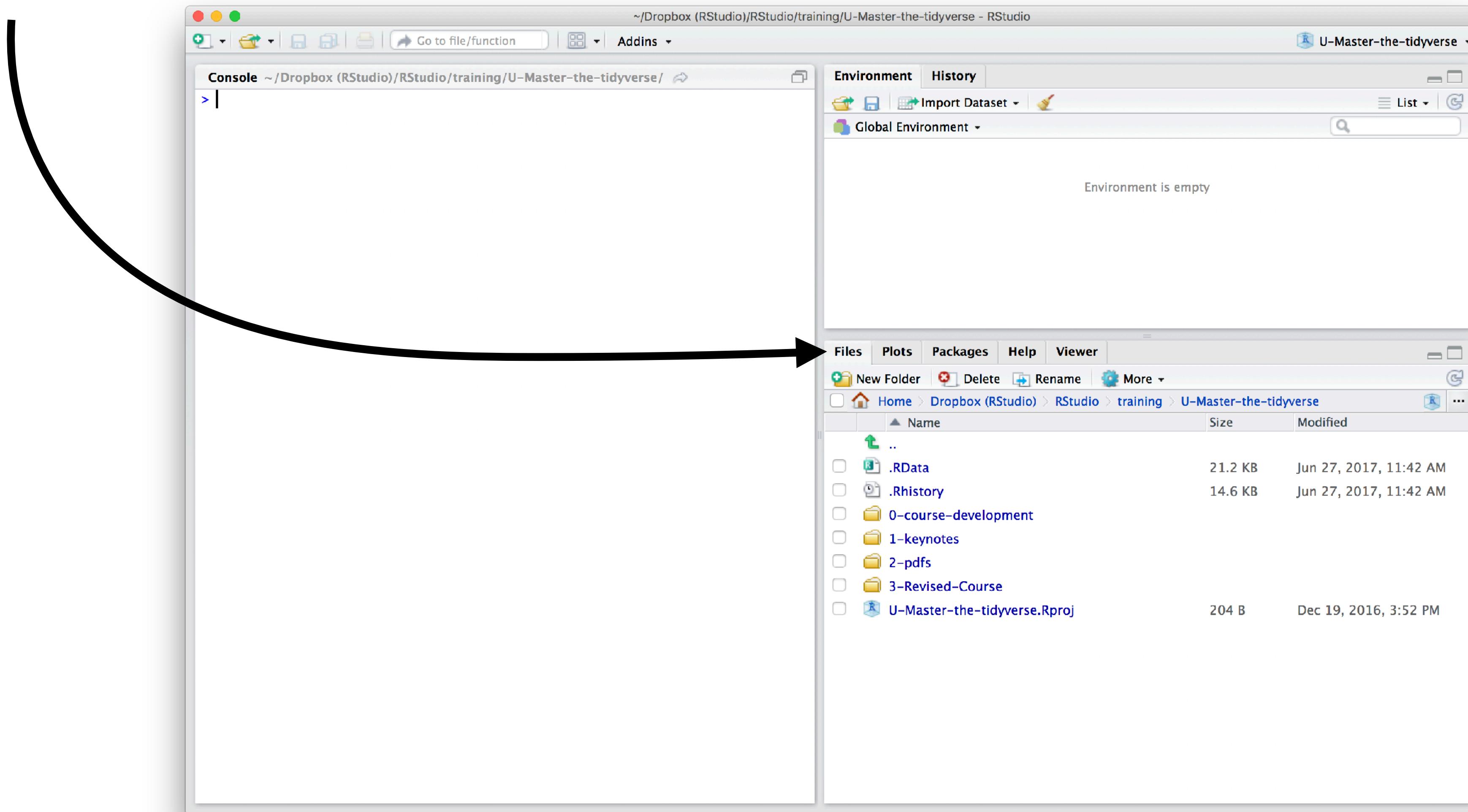
# Working directory

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

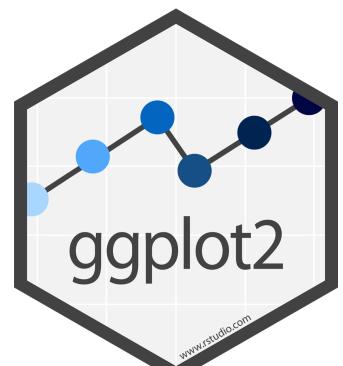
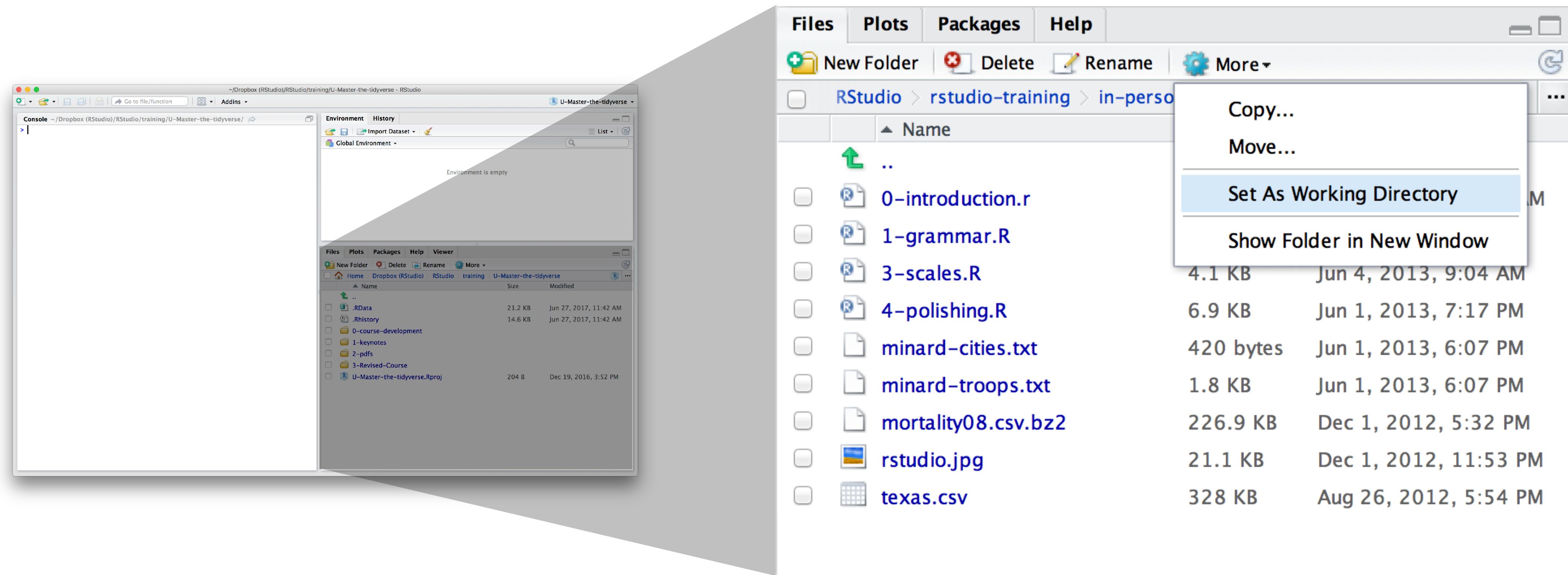


# The files pane of the IDE displays the contents of your working directory



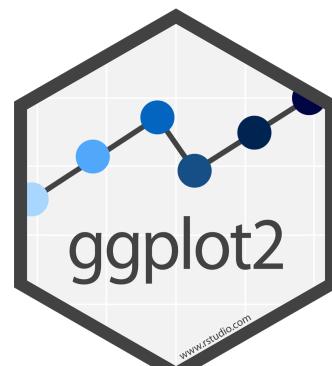
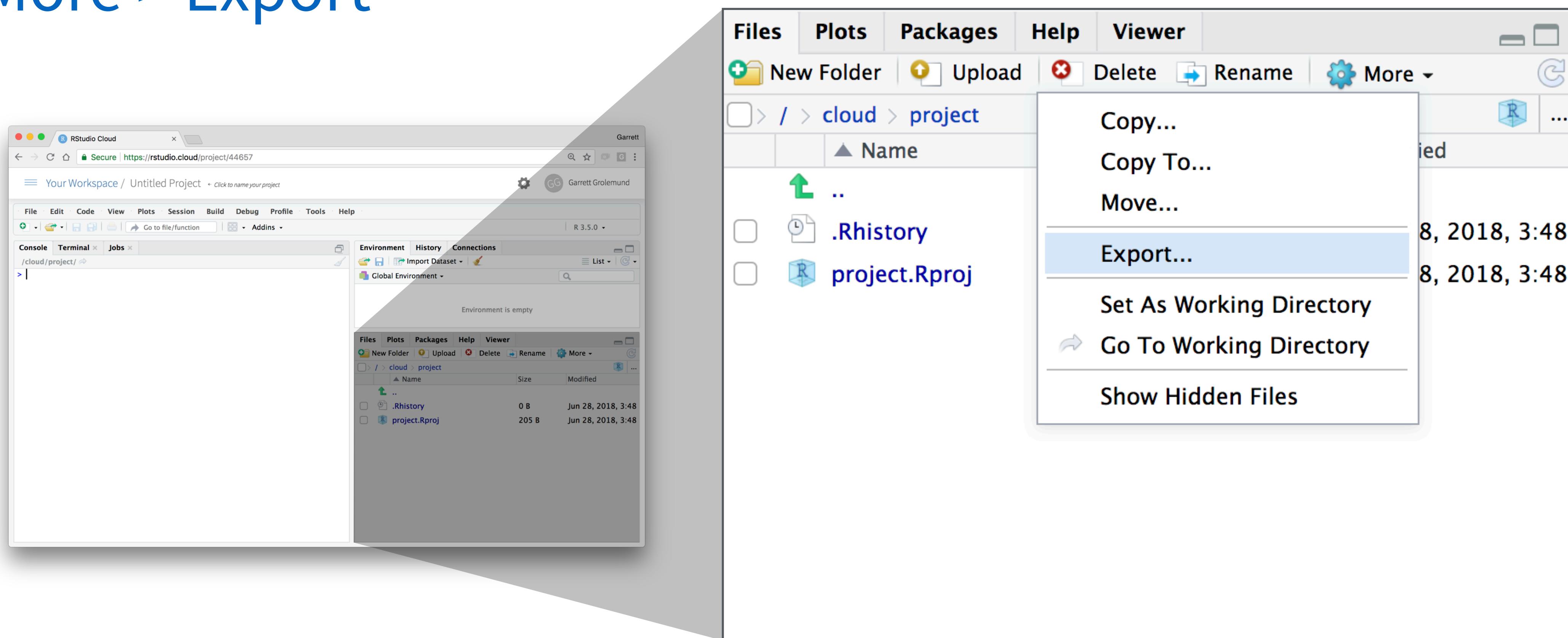
# Changing the Working directory

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



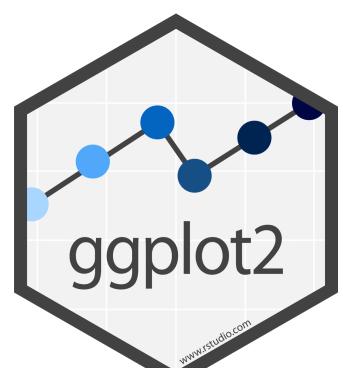
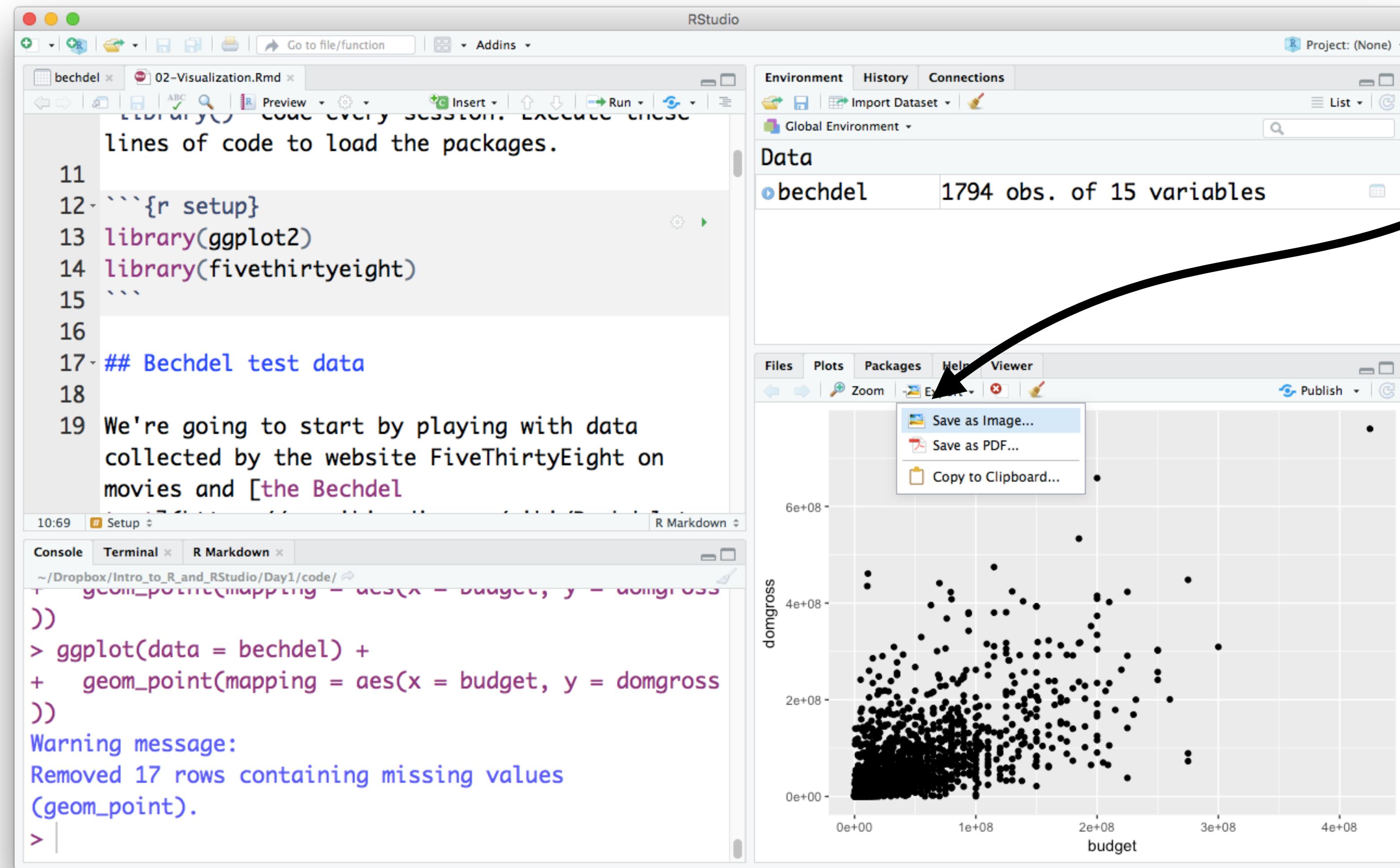
# Download files

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



# Manually saving plots

Save plots manually with the export menu



# Saving plots

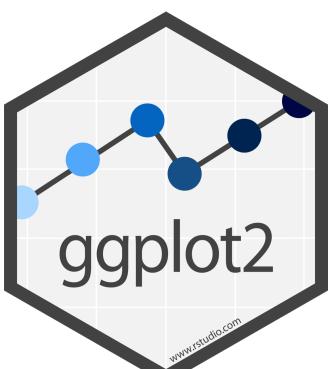
`ggsave()` saves the last plot.

Uses size on screen:

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

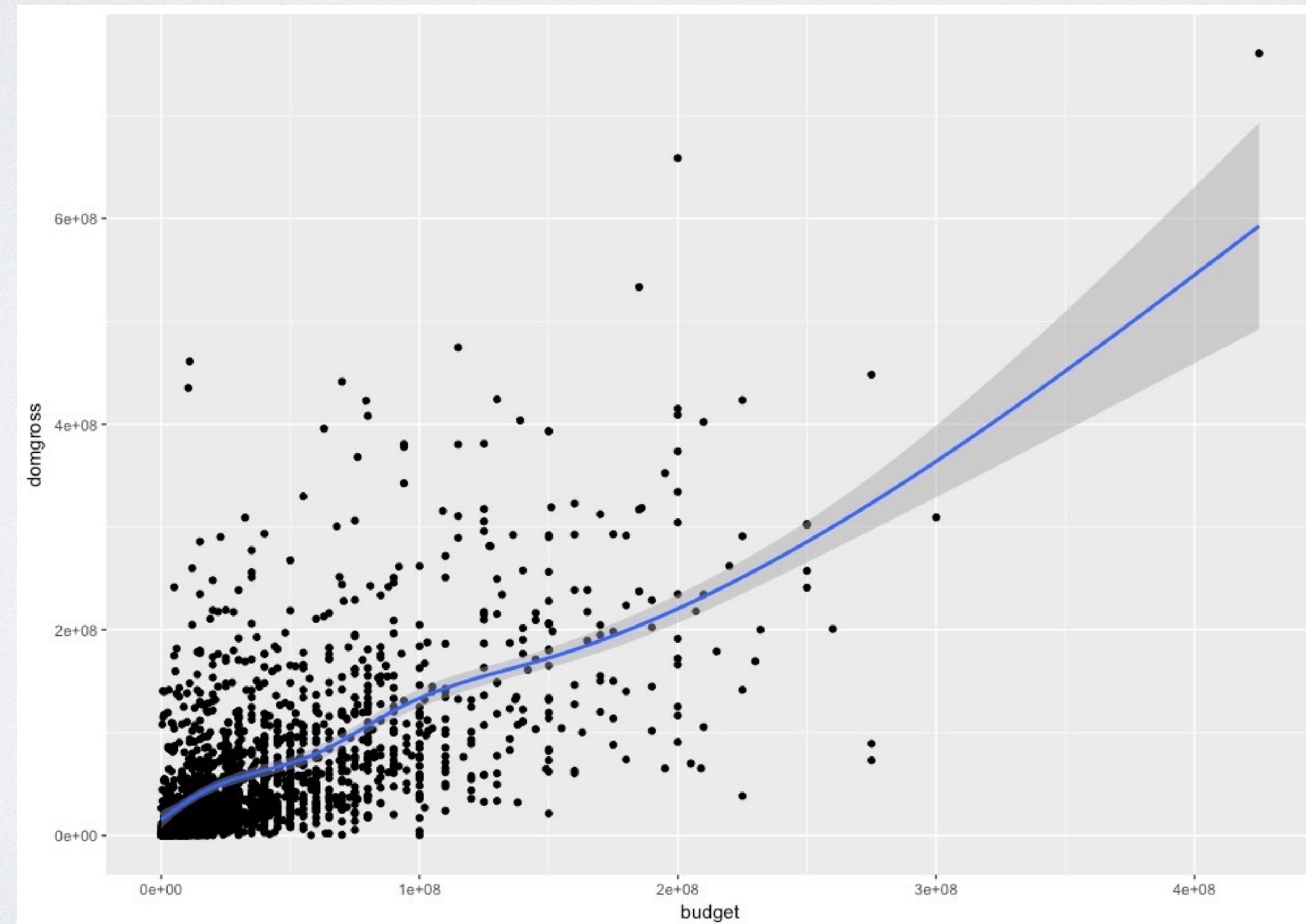
Specify size in inches

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



# Your Turn 9

Save your last plot and then locate it in your files pane and download it. (You may have to refresh the files list).



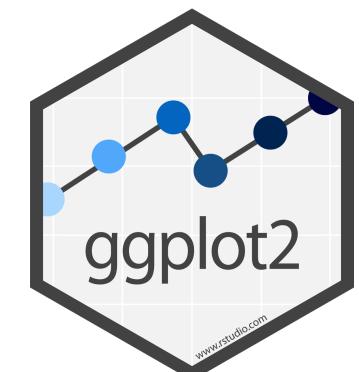
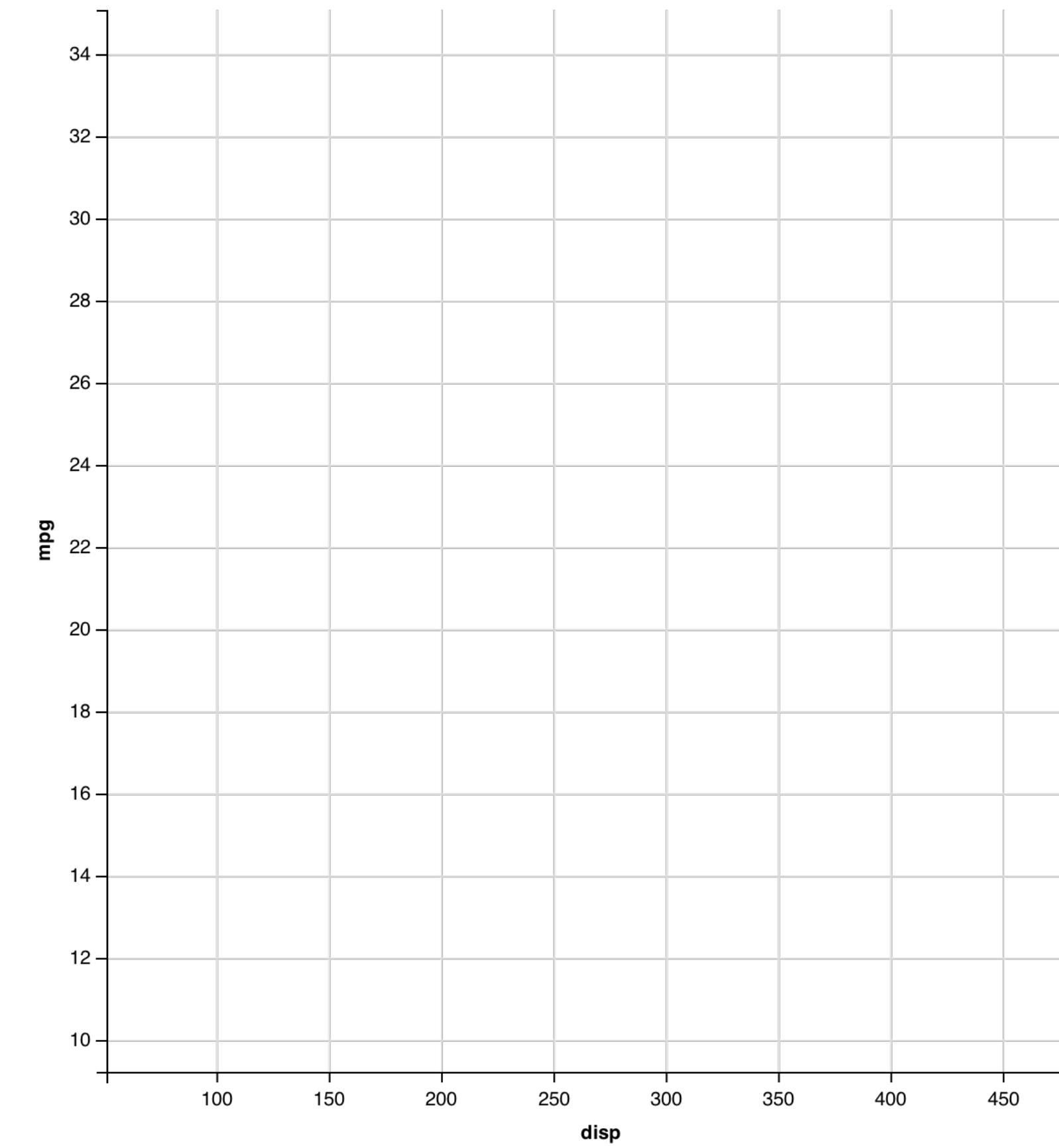
# Grammar of Graphics



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

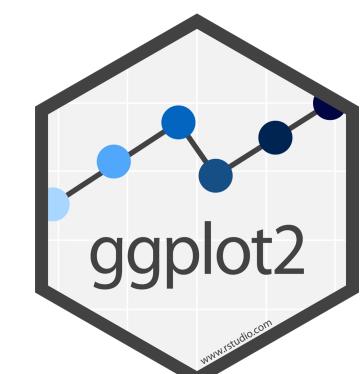
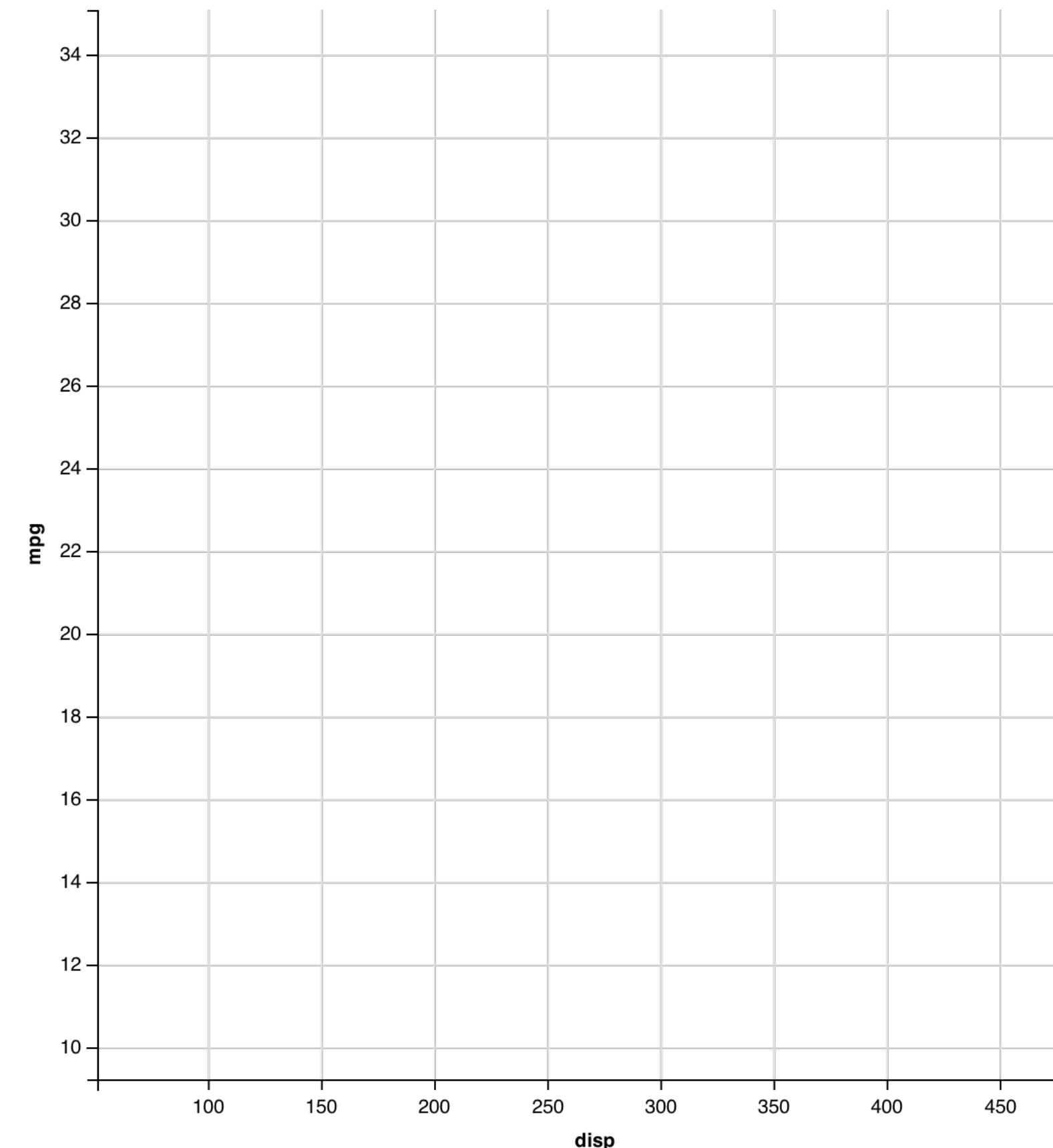
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

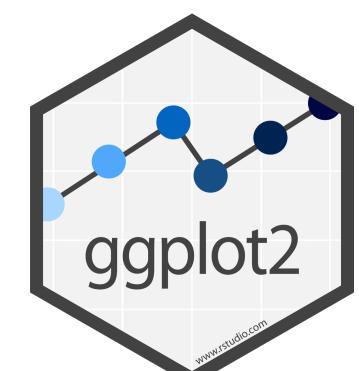
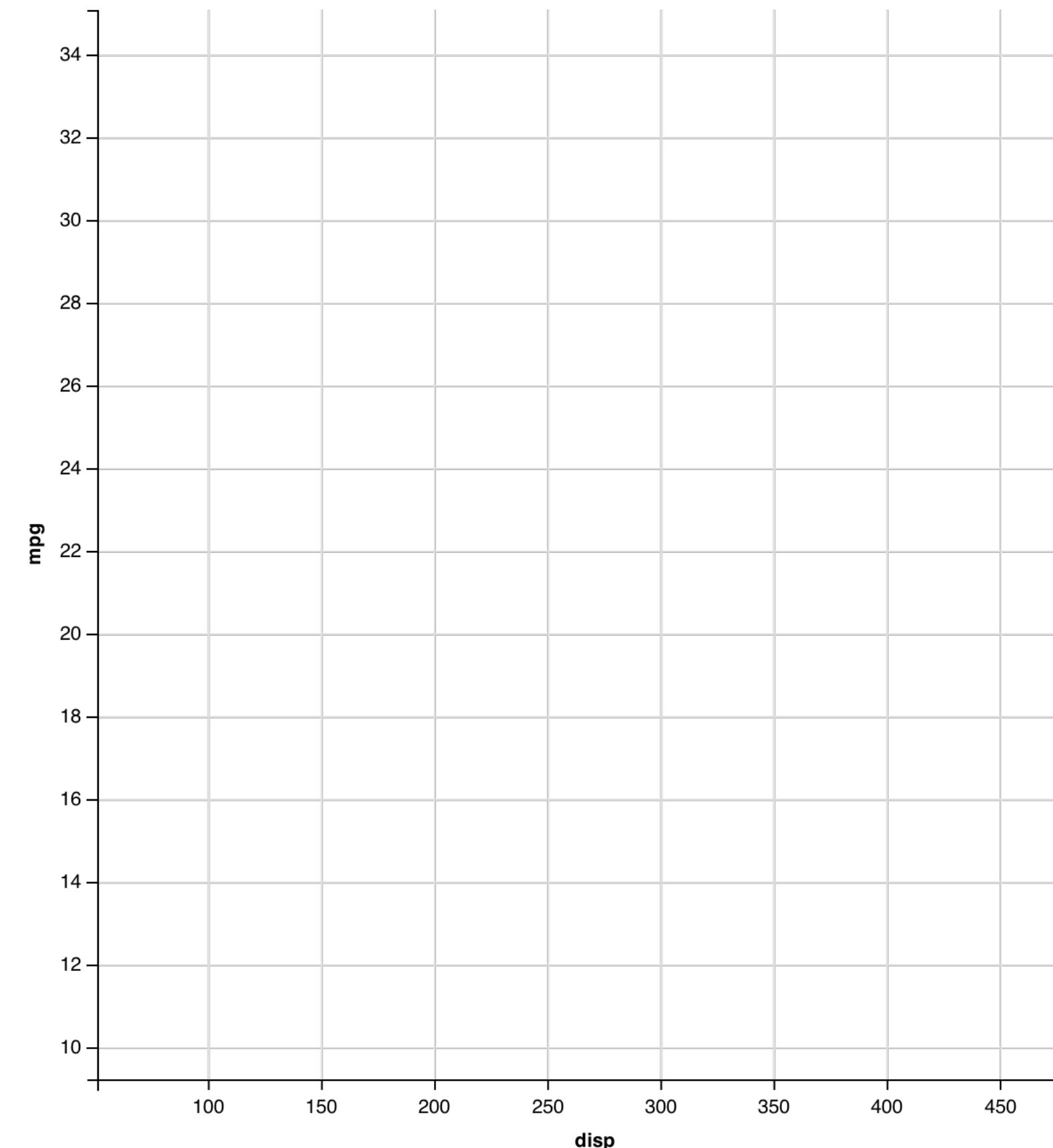


# mappings

	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

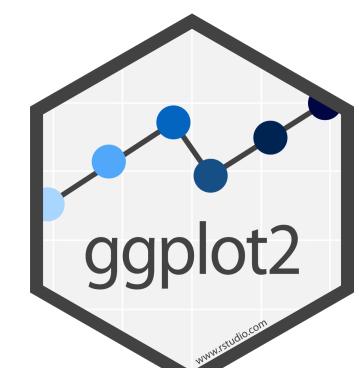
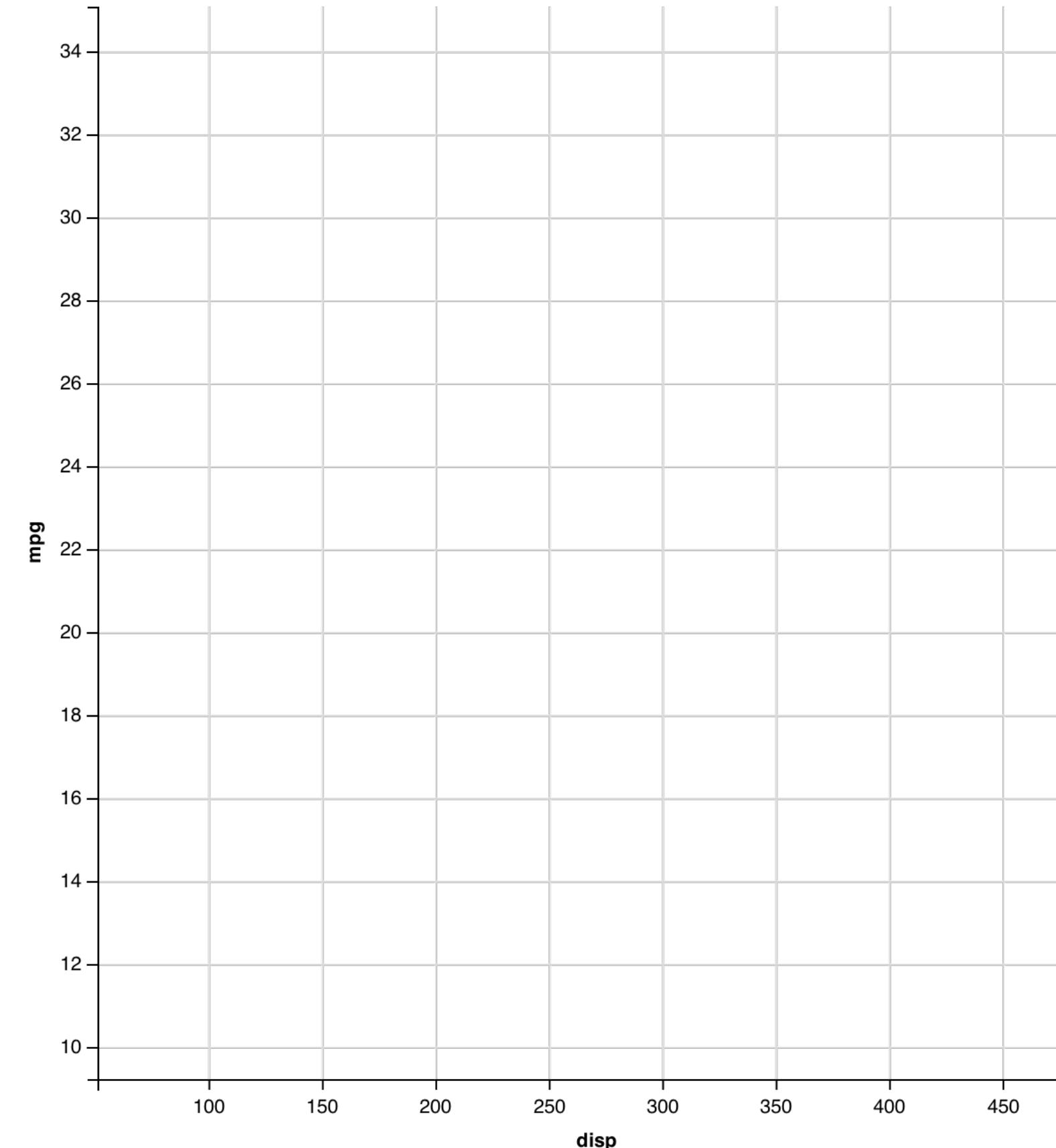


# mappings

	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

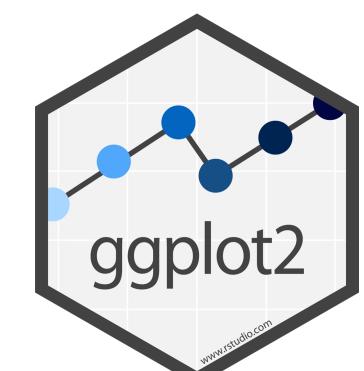
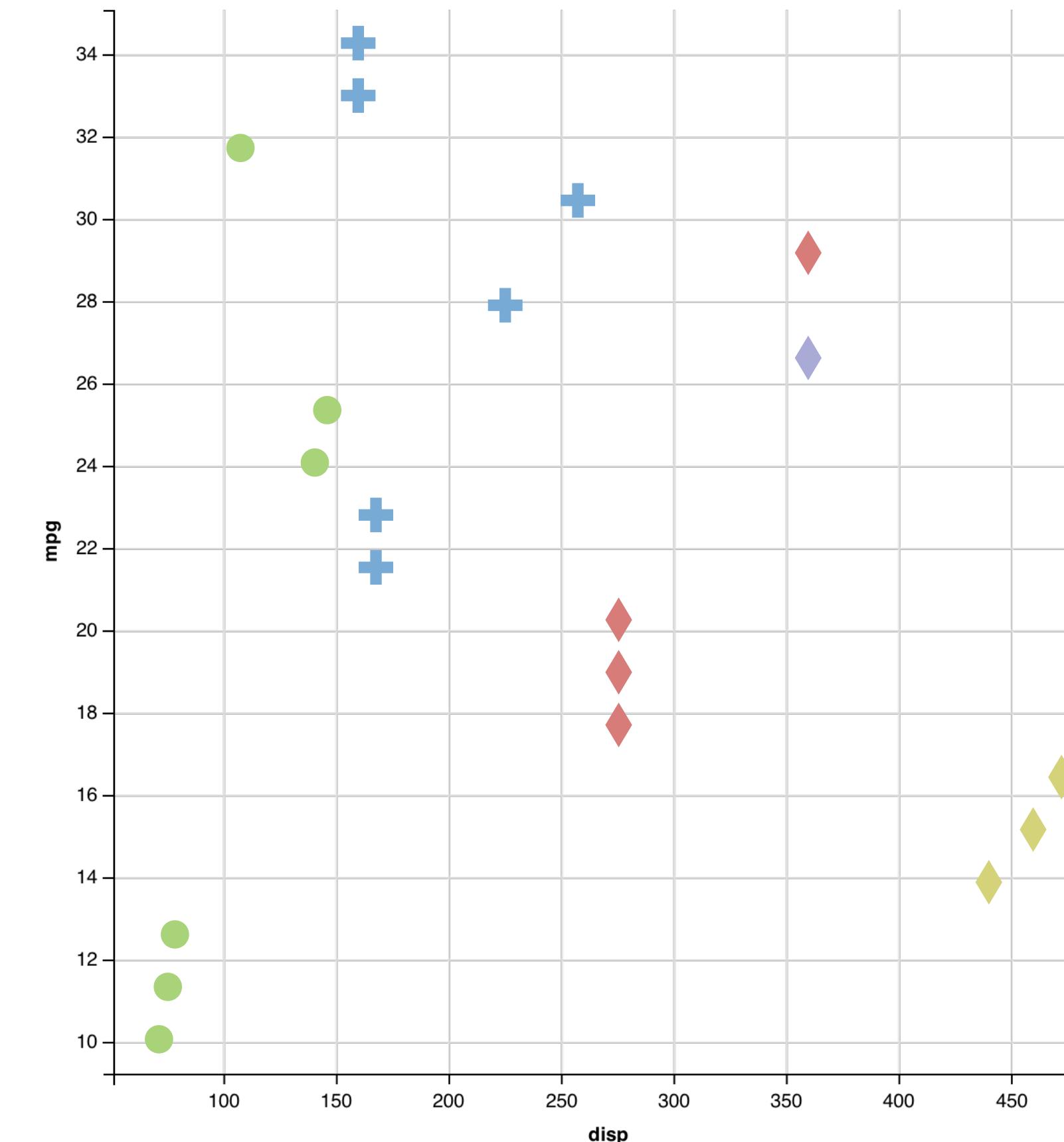


# mappings

	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

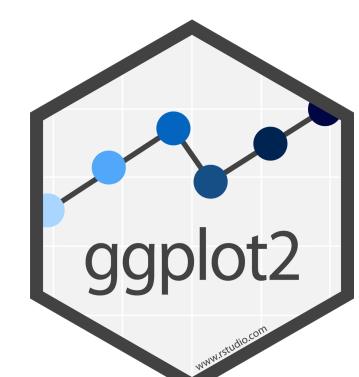
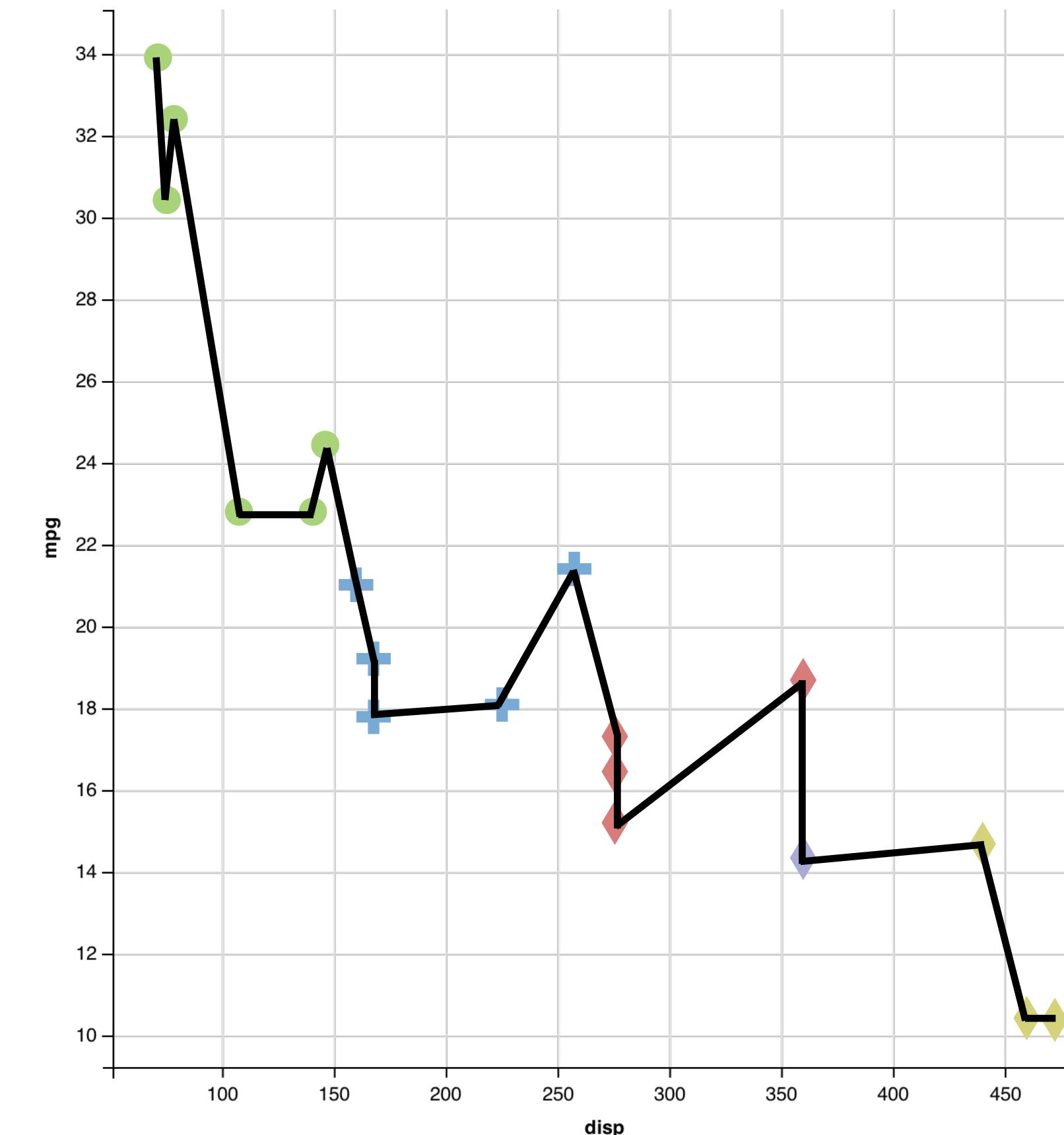


# mappings

	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

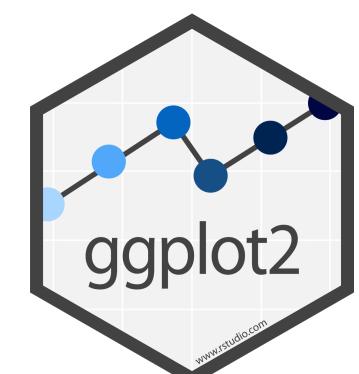
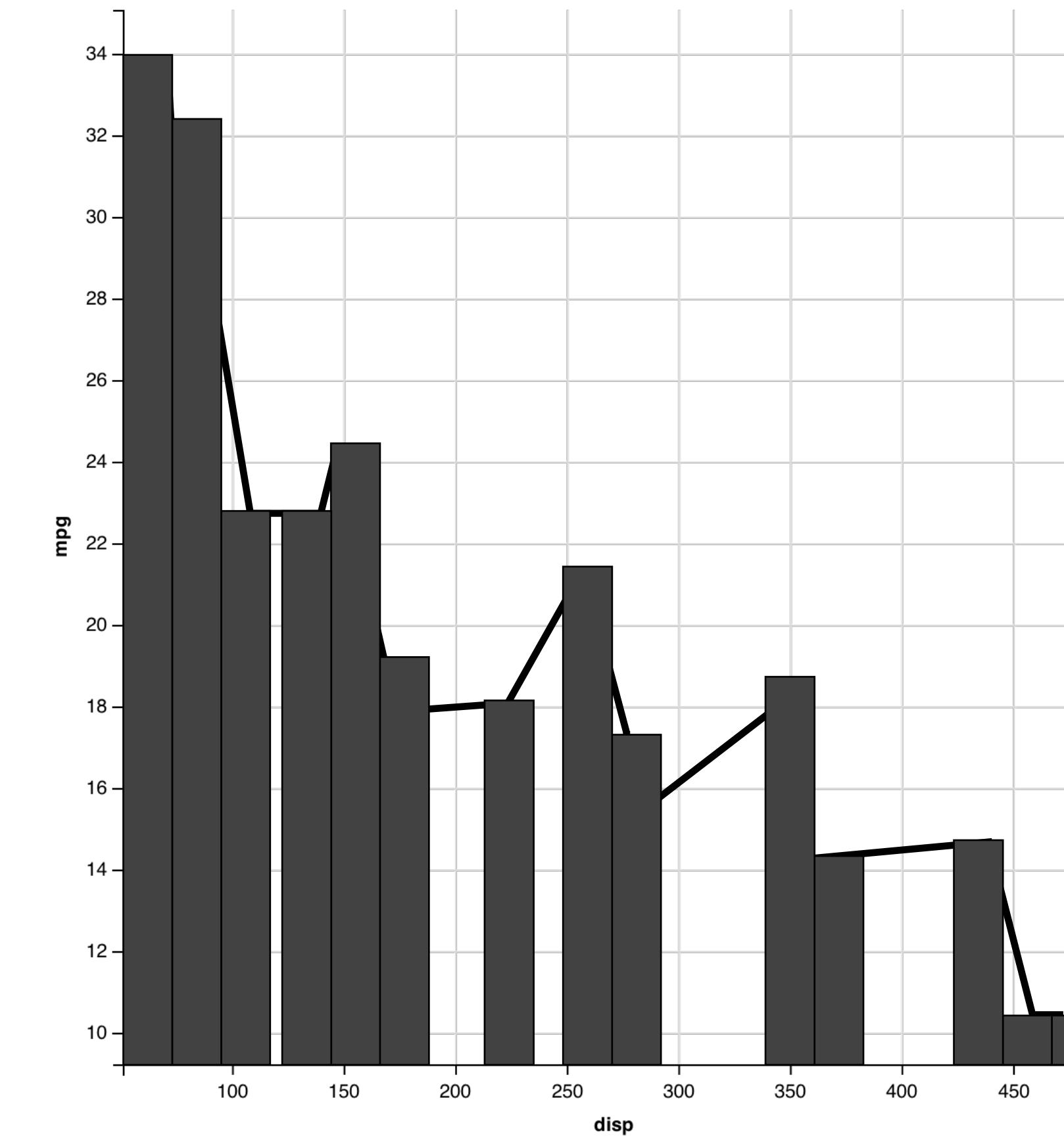


# mappings

	y	x	
mpg	↑	disp	↓
cyl		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

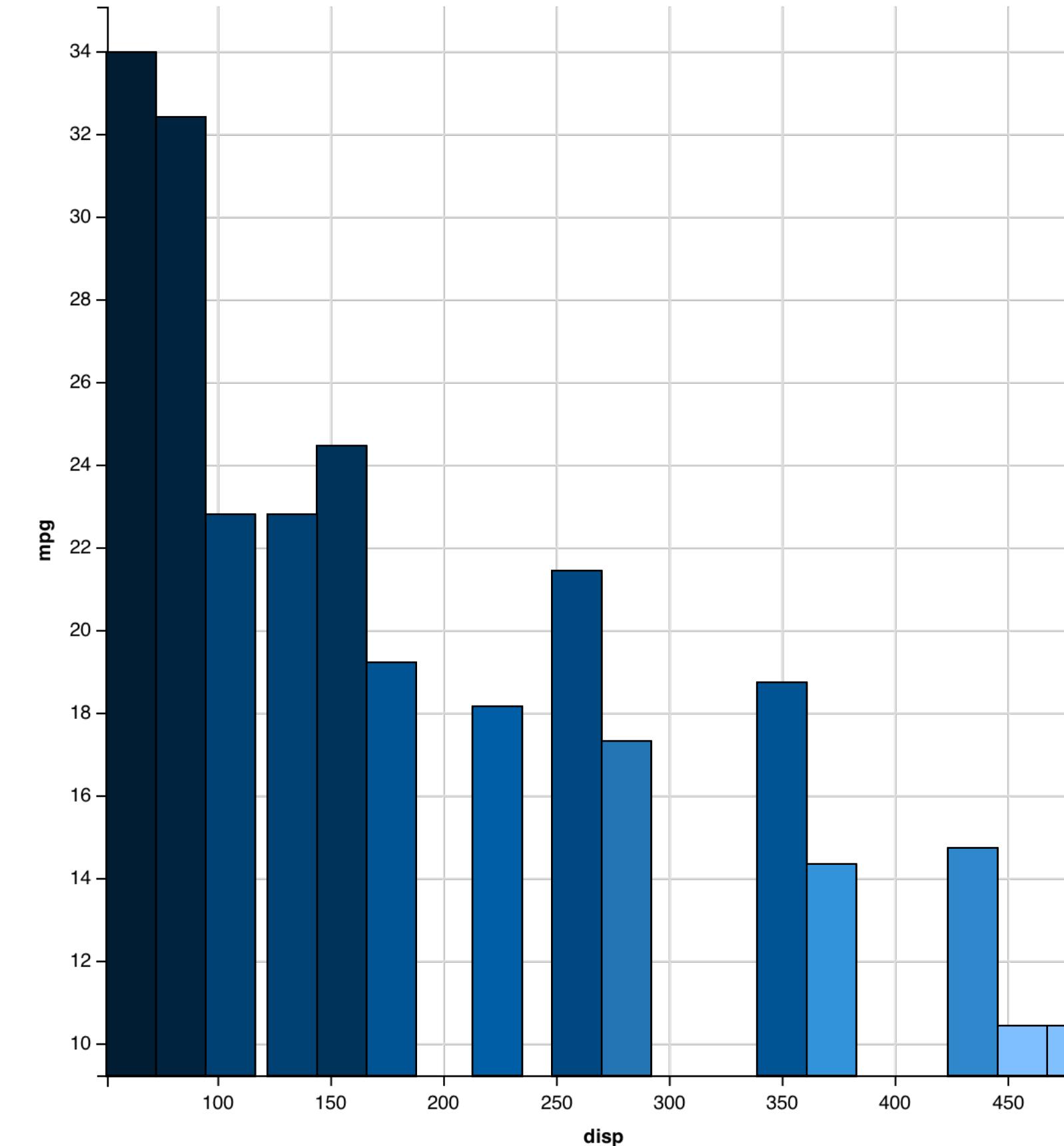
data

geom  
points  
lines  
bars



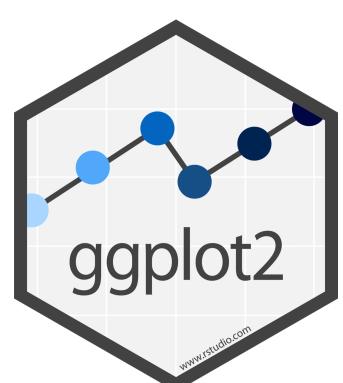
# mappings

	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

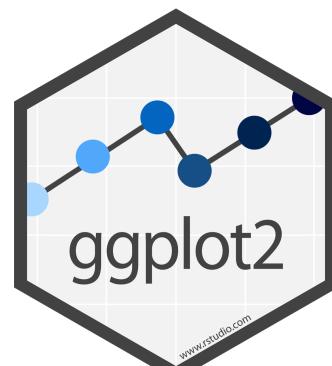
geom  
points  
lines  
bars



# To make a graph

[template]

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



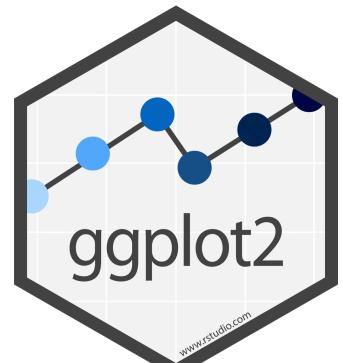
# To make a graph

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

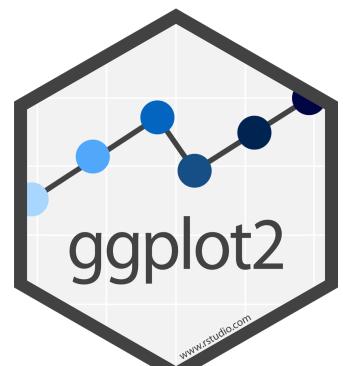
data

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

mappings

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

data

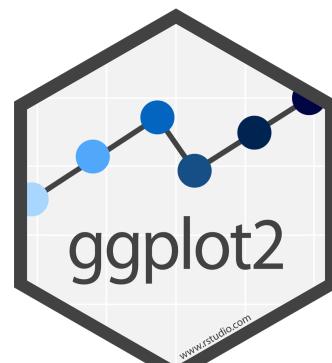
geom

1. Pick a data set

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

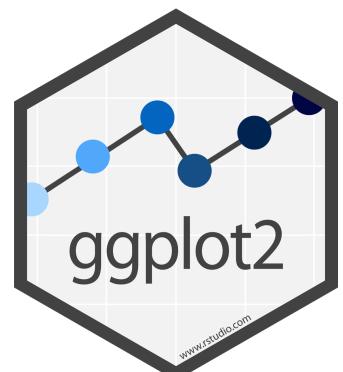
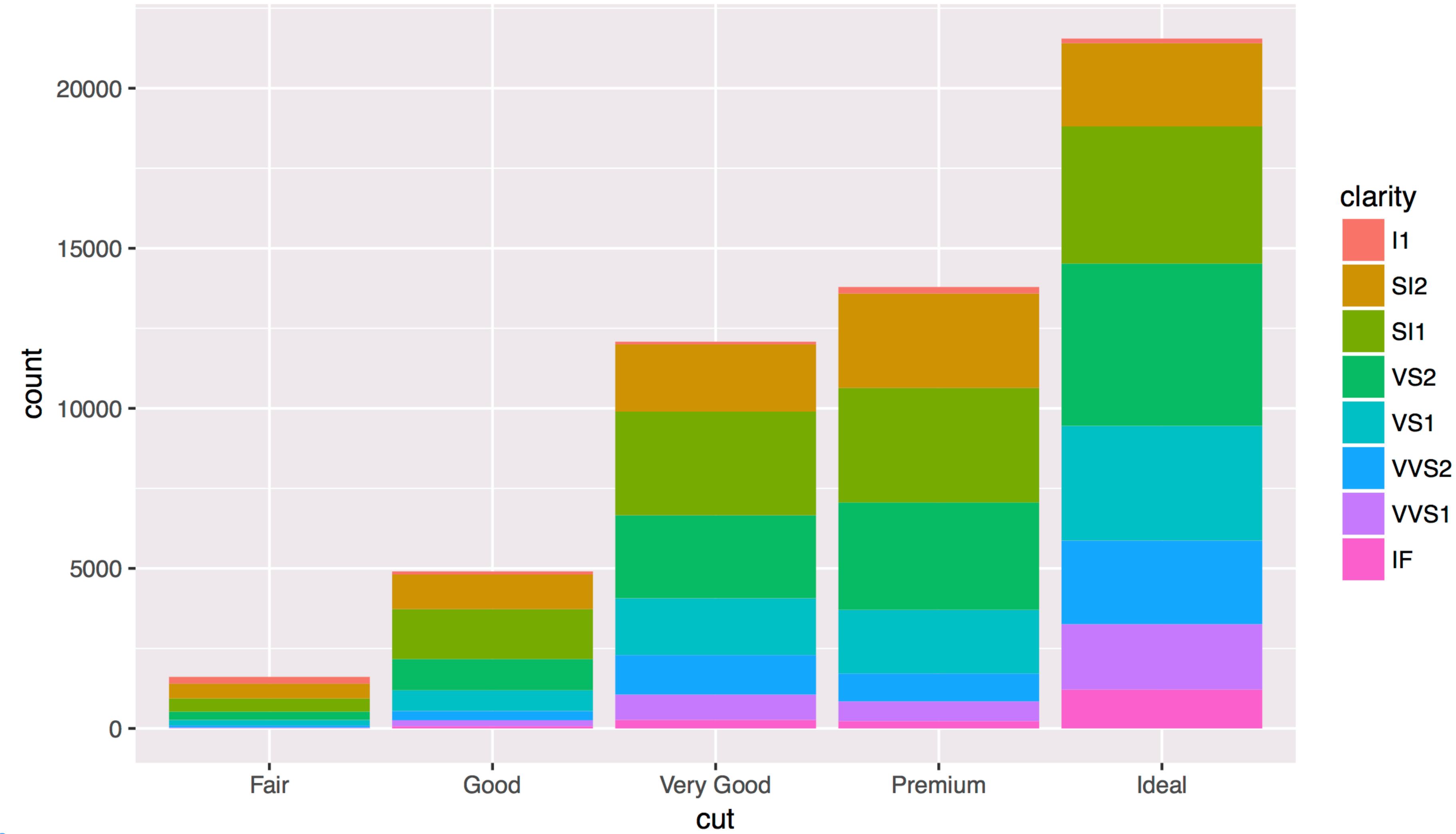
2. Choose a geom  
to display cases

3. Map aesthetic  
properties to  
variables



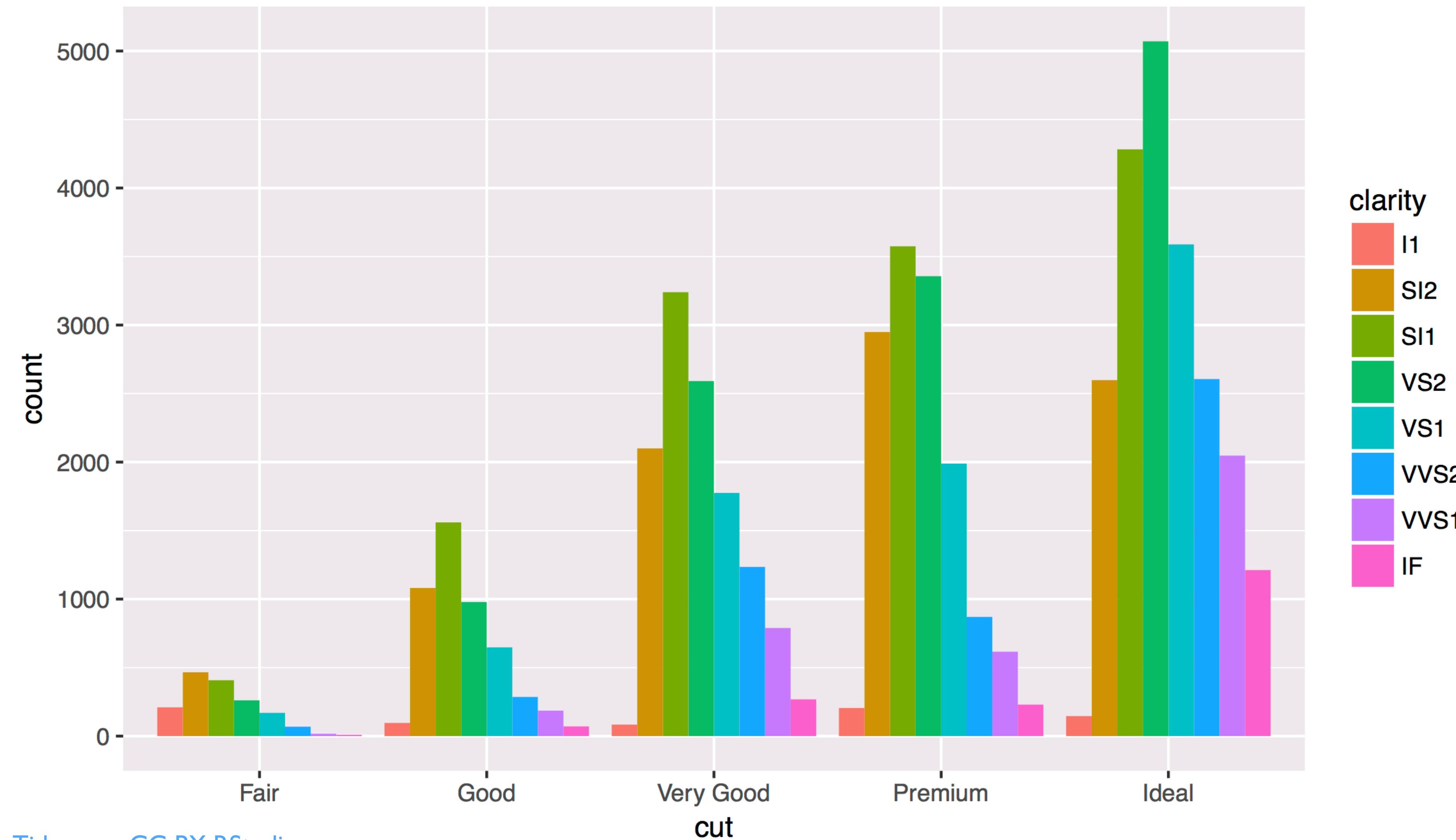
# what else?

R



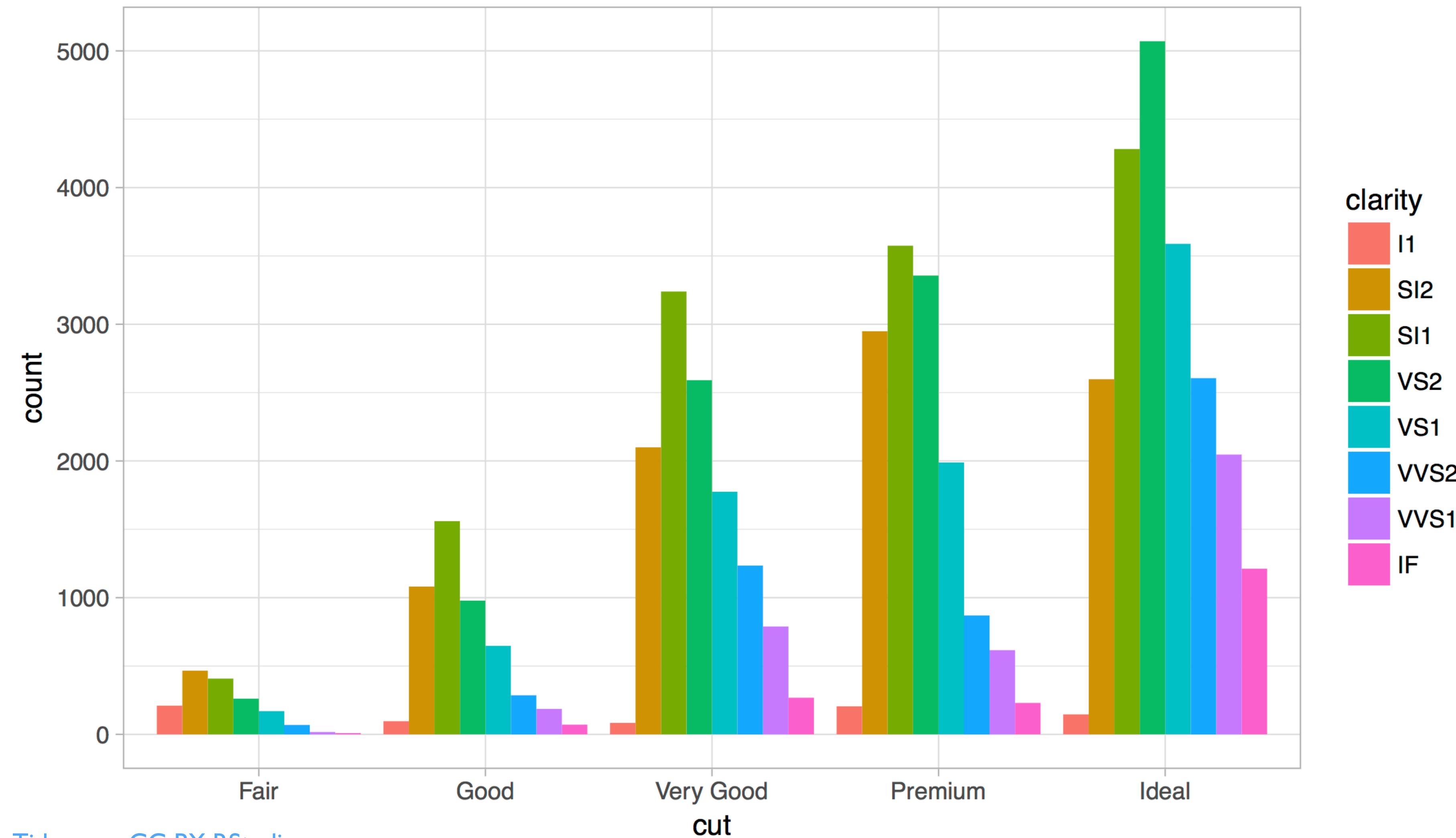
# Position Adjustments

How overlapping objects are arranged



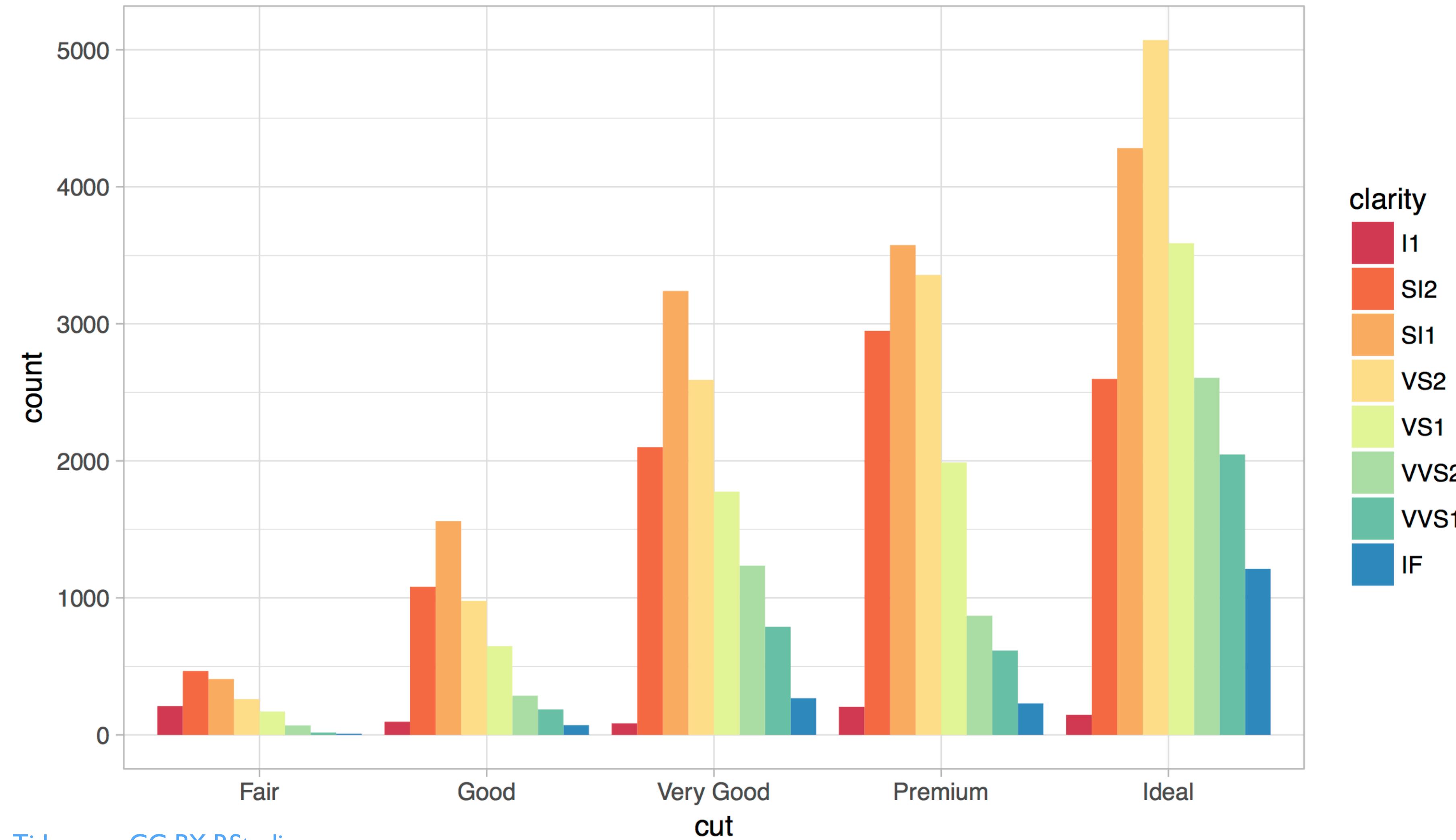
# Themes

## Visual appearance of non-data elements



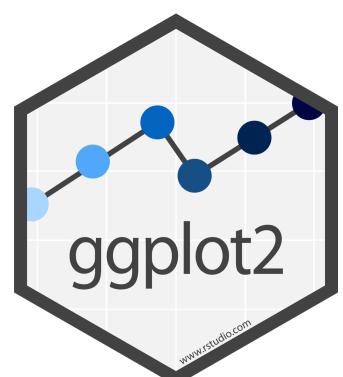
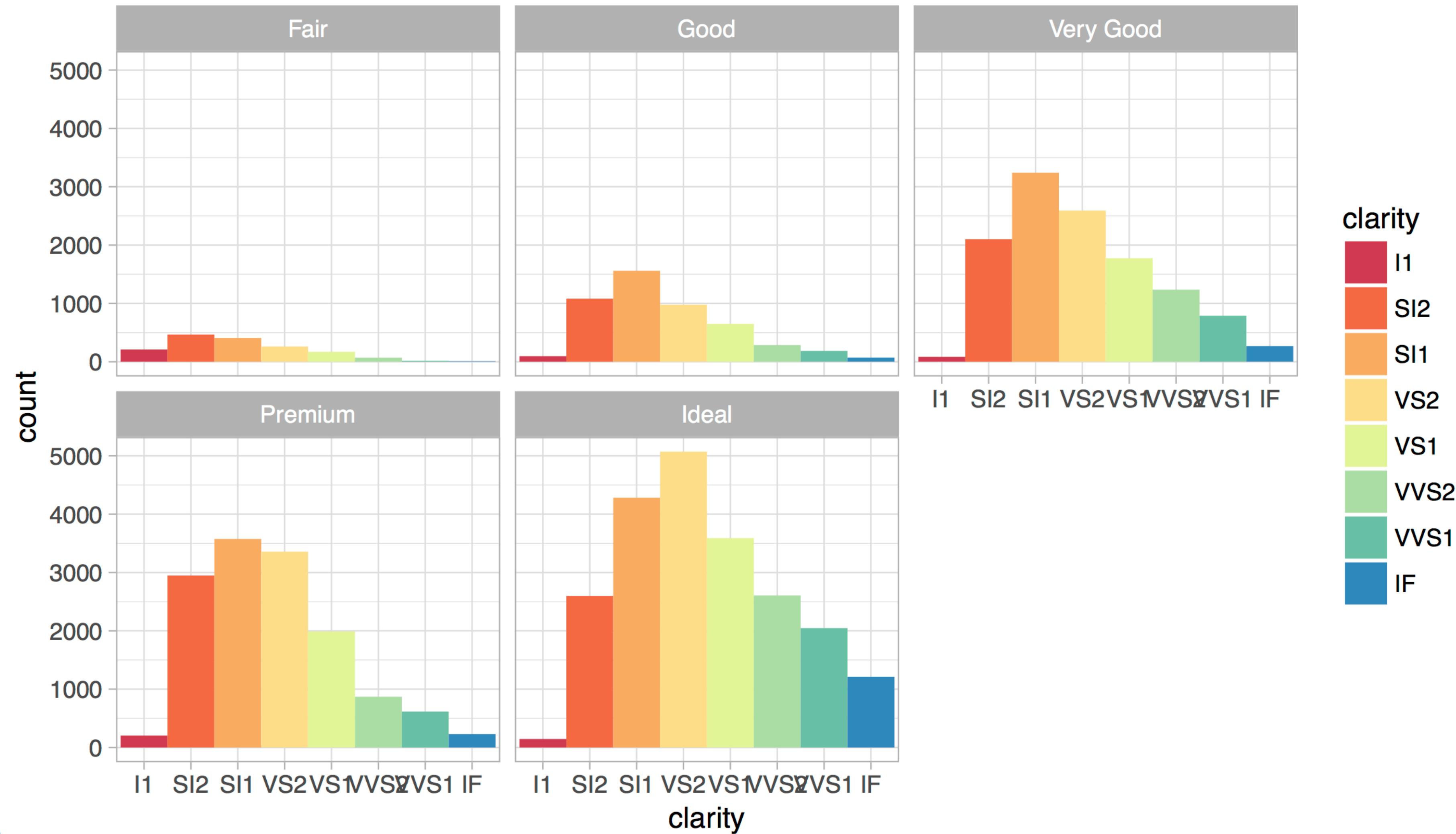
# Scales

Customize color scales, other mappings

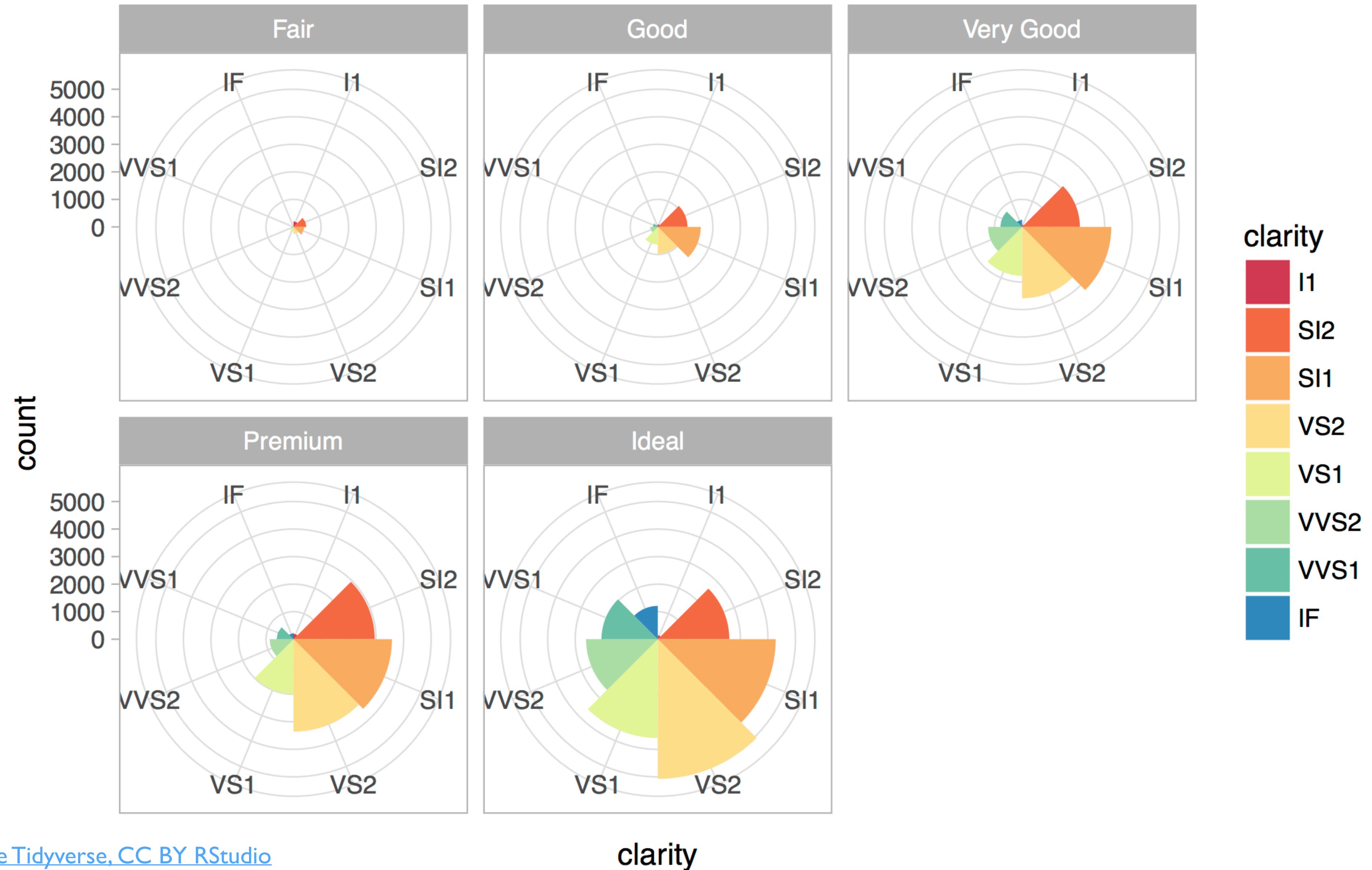


# Facets

Subplots that display subsets of the data.



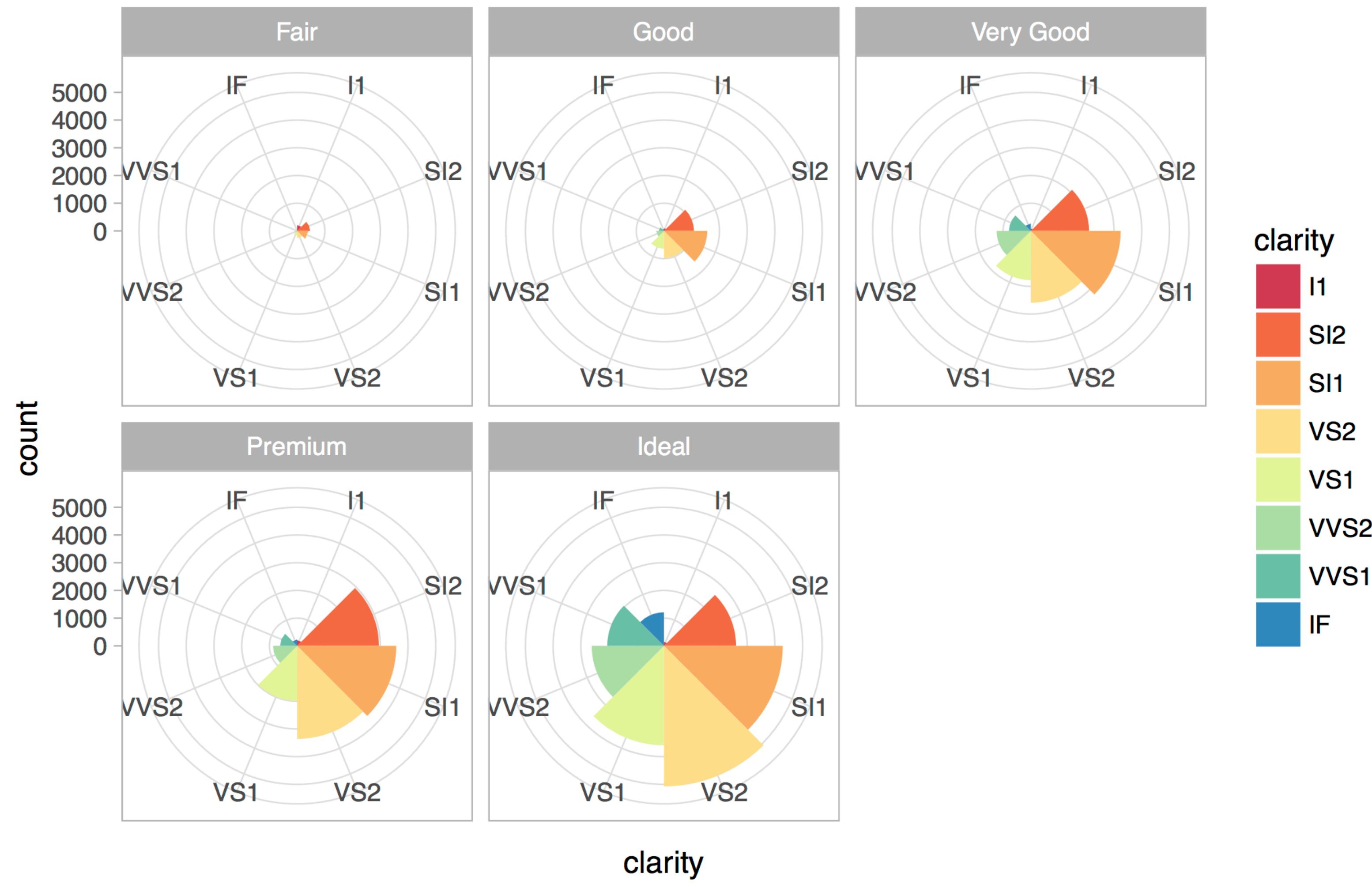
# Coordinate systems



# 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

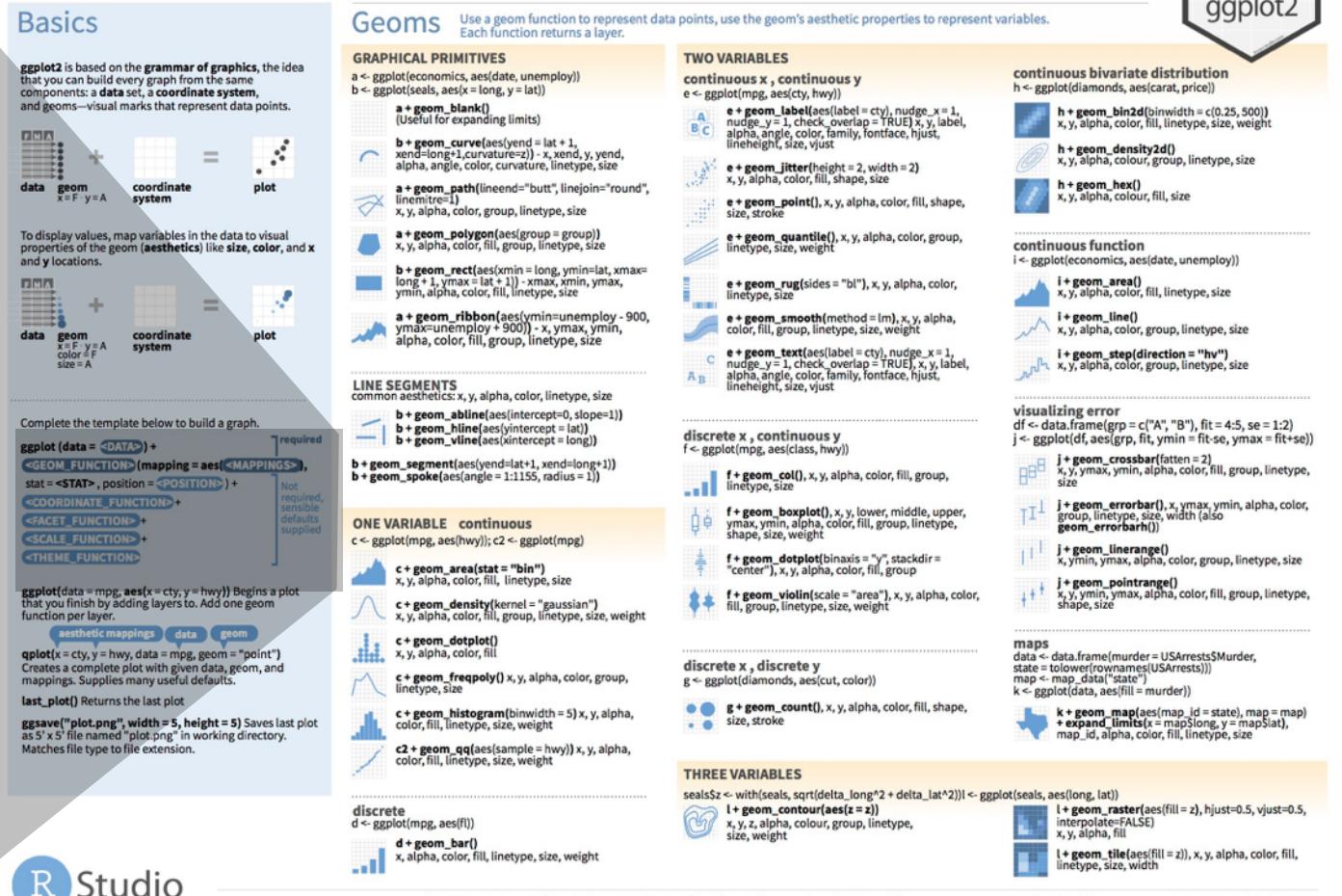
Complete the template below to build a graph.

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

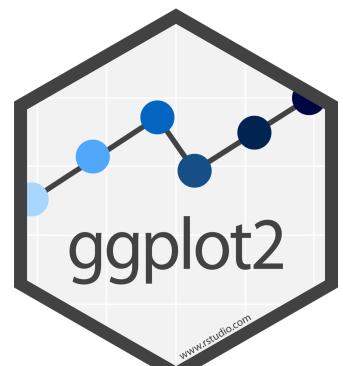
required

Not required,  
sensible  
defaults  
supplied

Data Visualization with ggplot2 :: CHEAT SHEET



R Studio



# ggplot2.tidyverse.org

The screenshot shows a web browser displaying the ggplot2.tidyverse.org website. The page title is "Create Elegant Data Visualisati" and the user is logged in as "Garrett". The address bar shows the URL "ggplot2.tidyverse.org". The main content area features the ggplot2 logo and the text "part of the tidyverse". A large heading "Usage" is followed by a paragraph explaining the philosophy of ggplot2 and how to use it. Below this is a code block showing R code to create a scatter plot. To the right, there are sections for "Links", "License", and "Developers", along with a ggplot2 logo at the bottom right.

## 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) on the y-axis versus engine displacement (displ) on the x-axis. The plot includes a legend for car classes, with a specific entry for "2seater" marked with a purple dot.

### 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

ggplot2

# Visualize Data with

