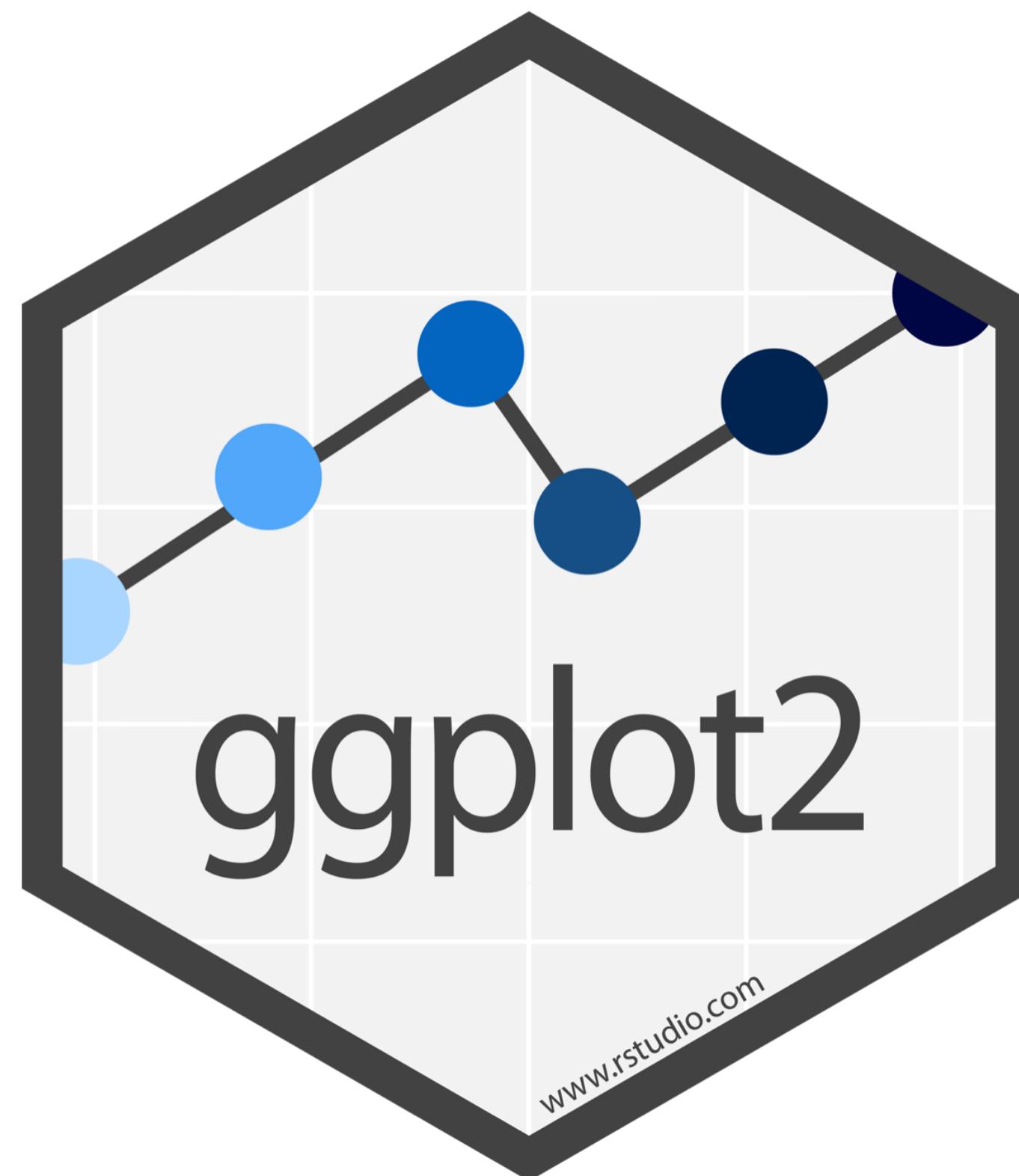


# Visualize Data with

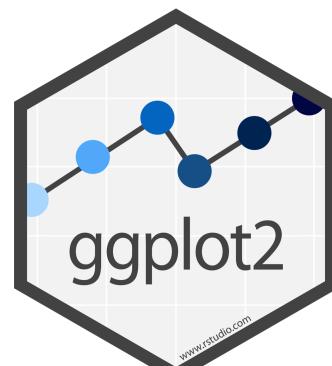
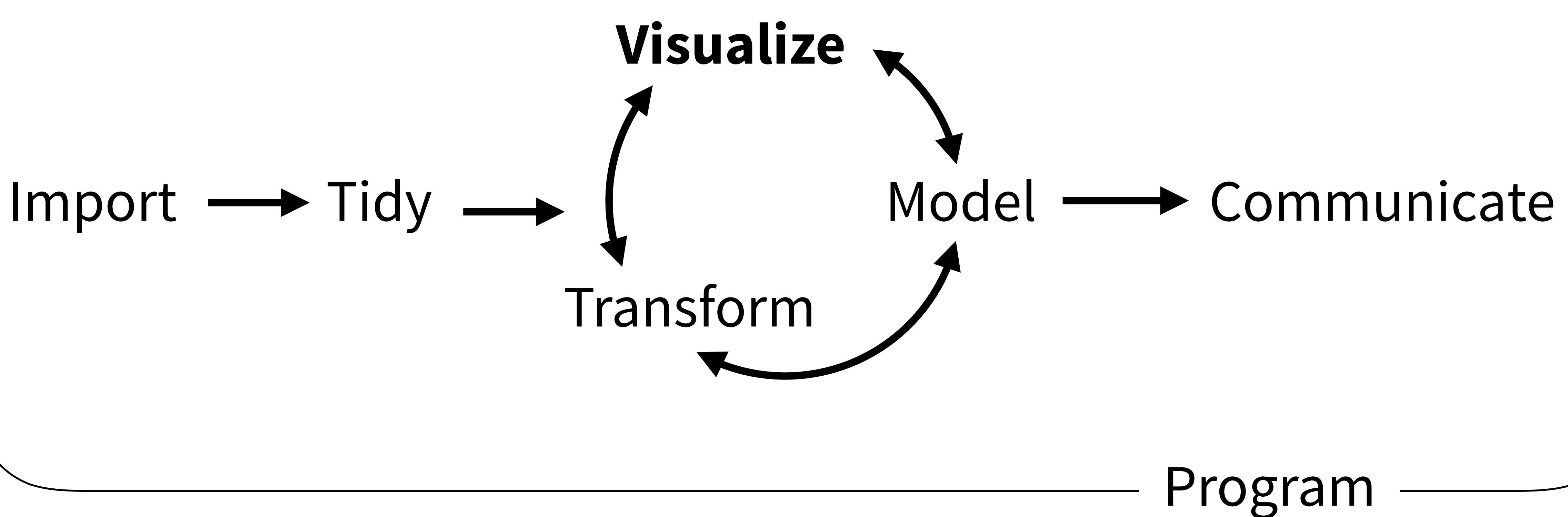


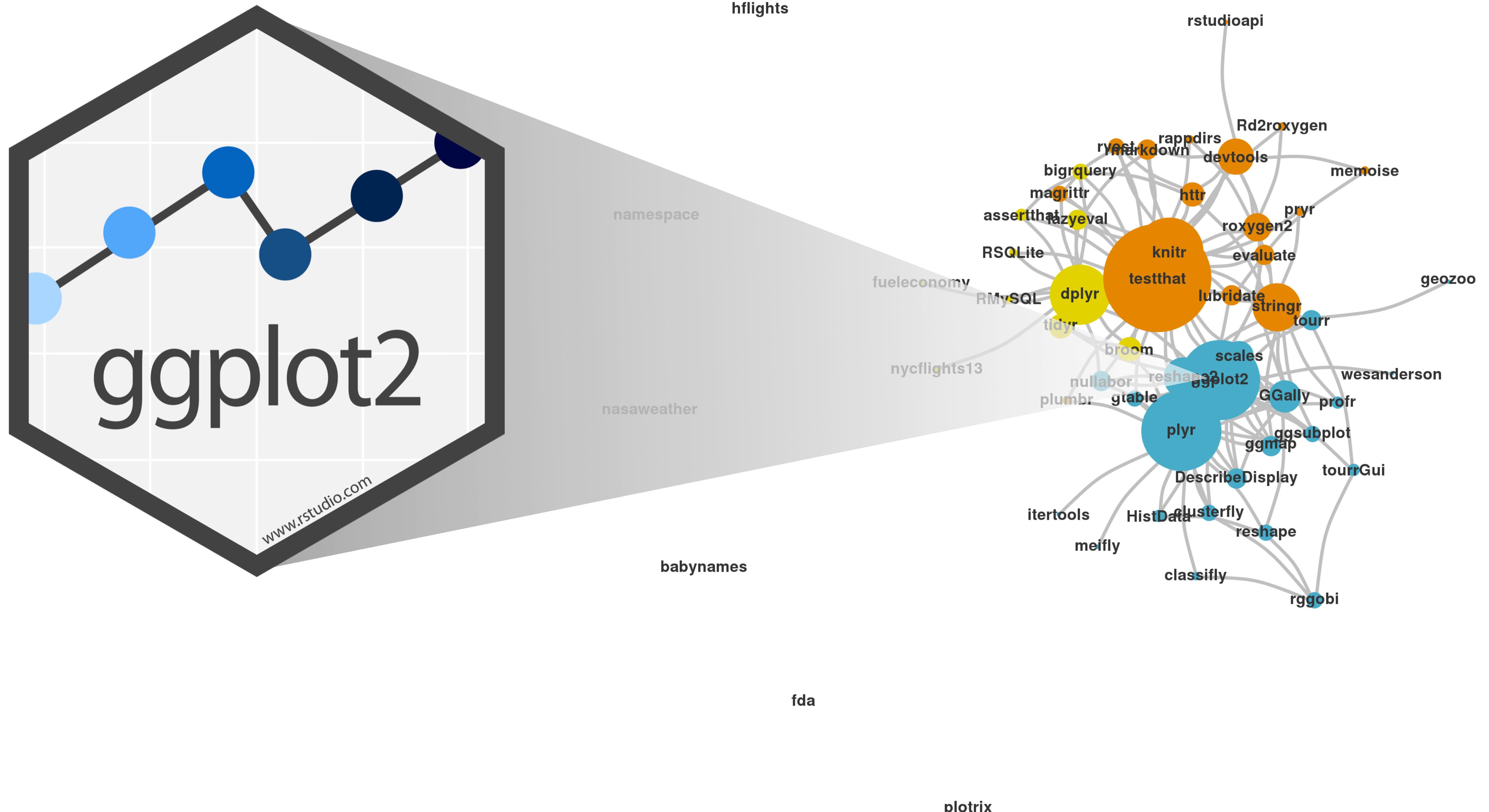
In R4DS  
Visualizing Data

"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

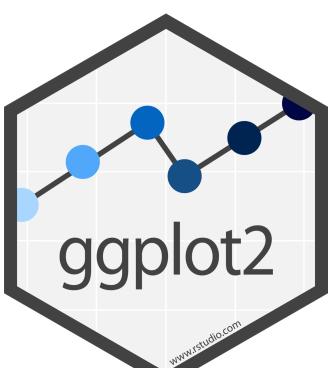
# Open the R Notebook 01-Visualize.Rmd

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

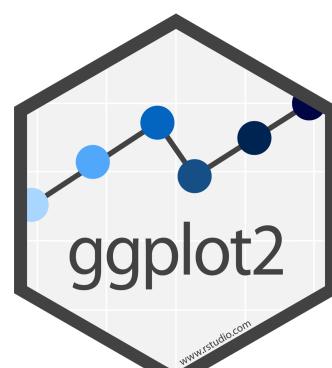
The setup chunk is always run once before anything else



```
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
```

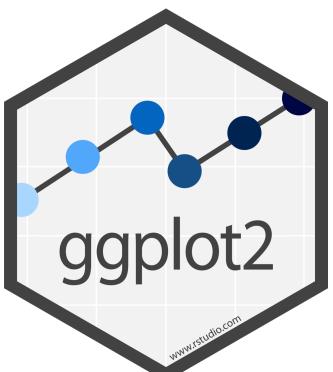


# bechdel

## 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)?

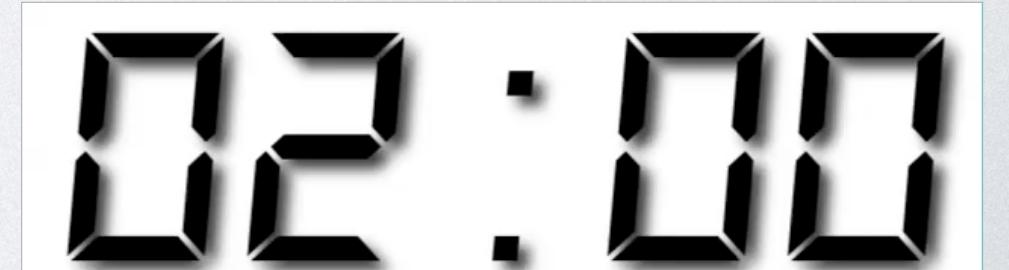


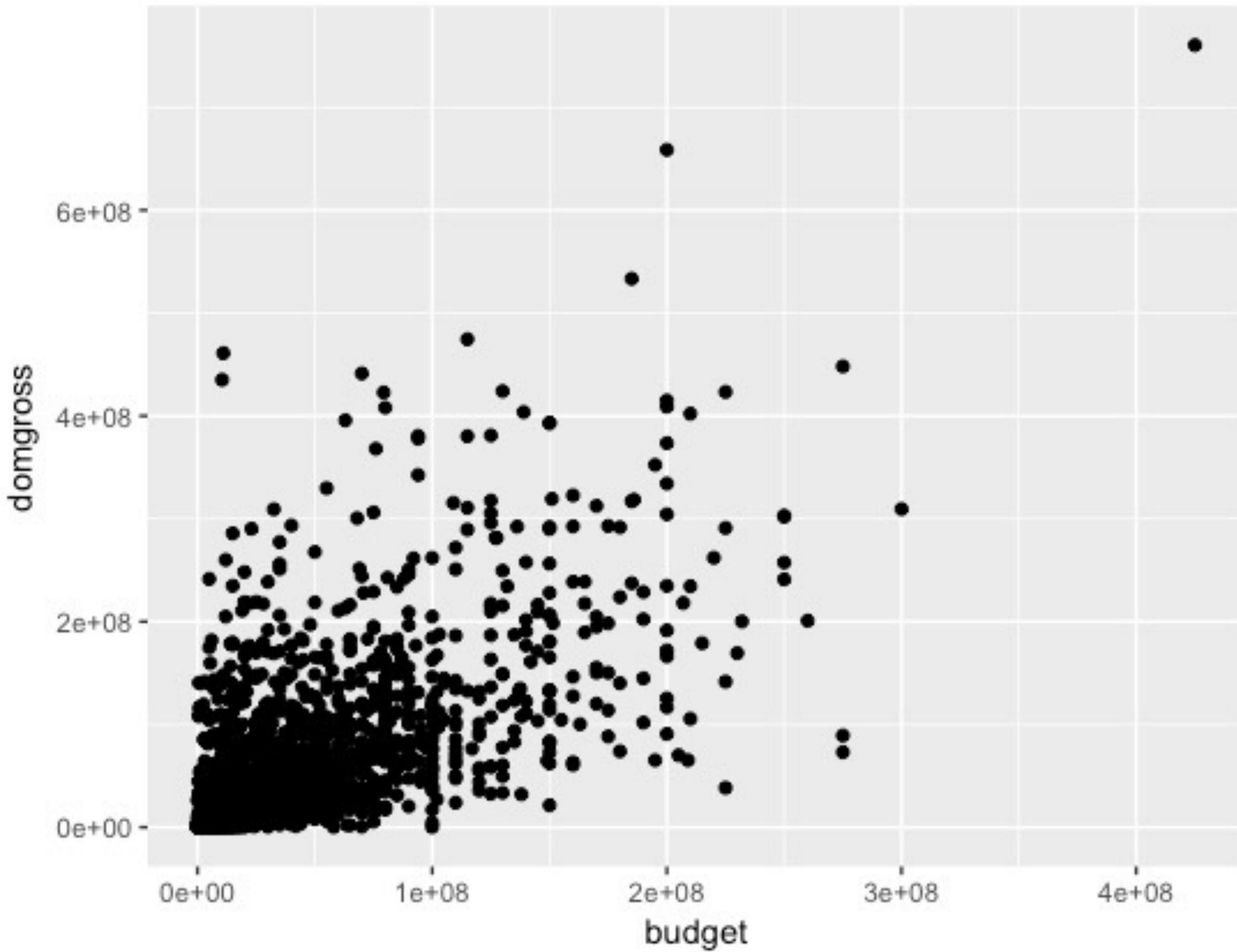
# Your Turn 1

Run this code in your notebook to make a graph.

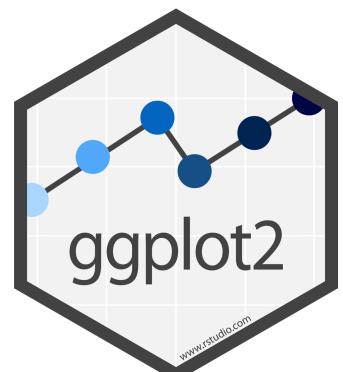
Pay strict attention to spelling, capitalization, and parentheses!

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

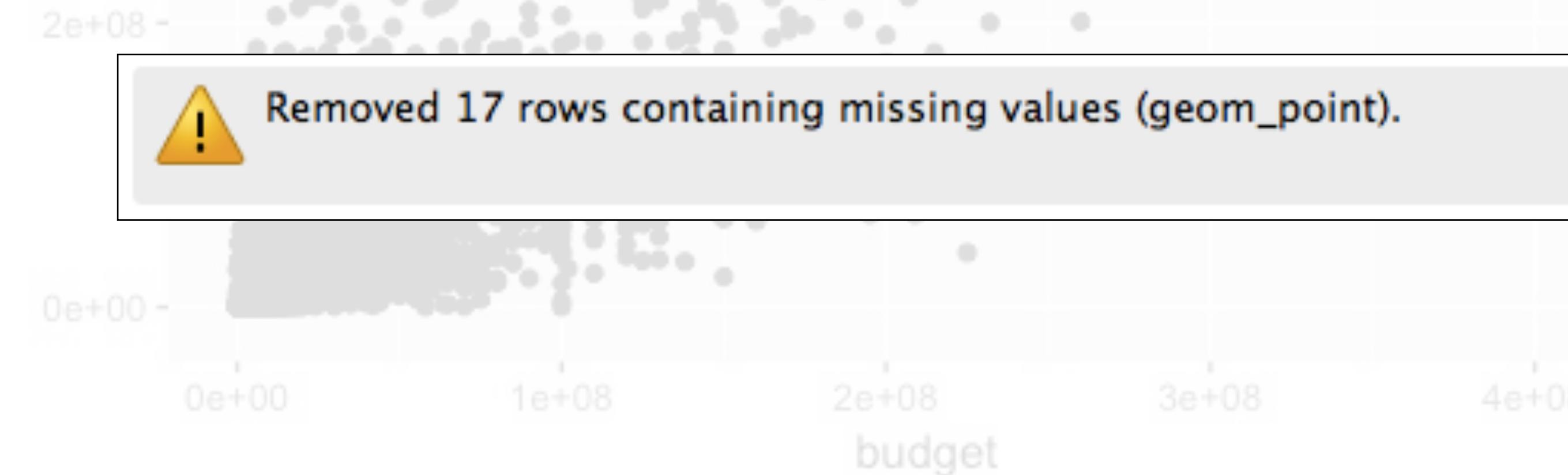




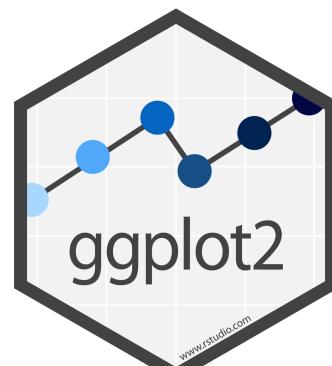
```
ggplot(data = bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross))
```



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.

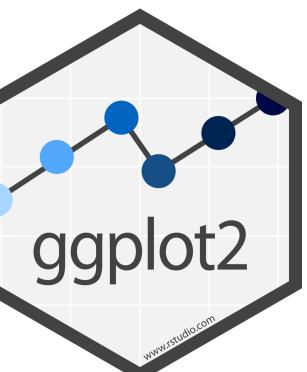


```
ggplot(data = bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross))
```



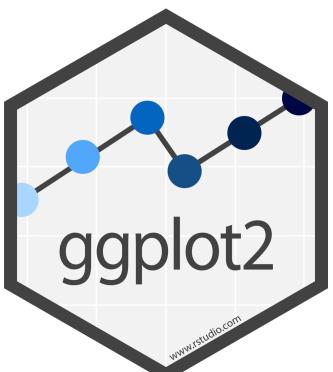
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

```
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

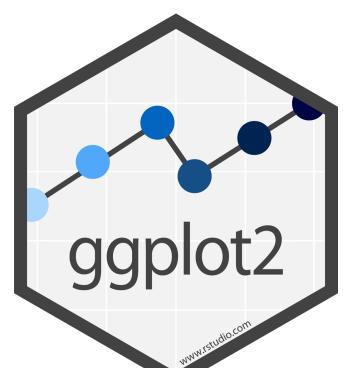
+ before new line

type of layer

aes()

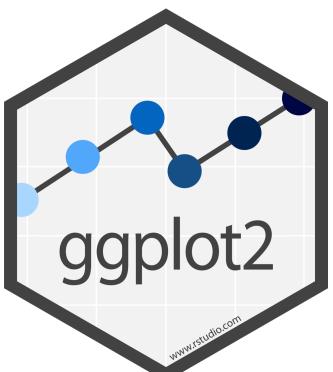
x variable

y variable



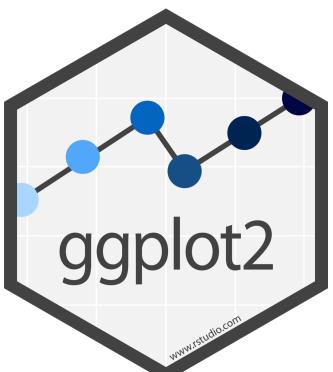
# A template

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



# A template

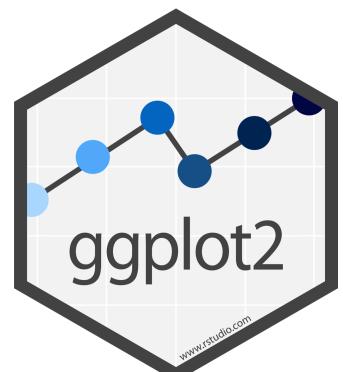
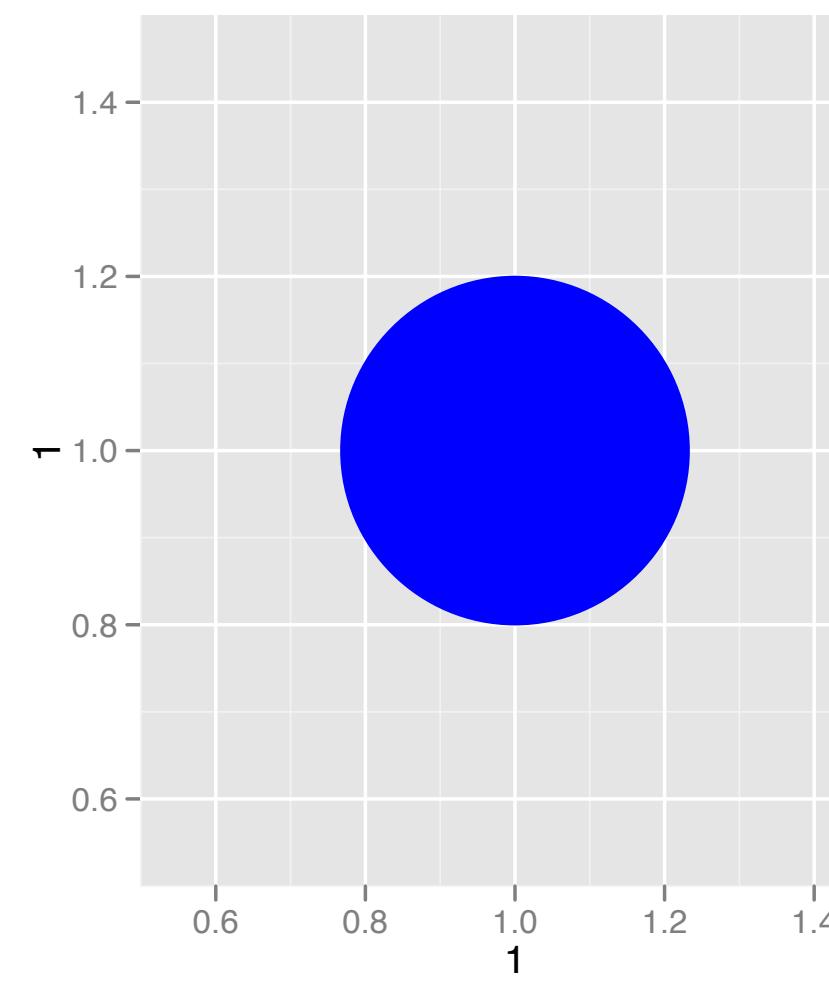
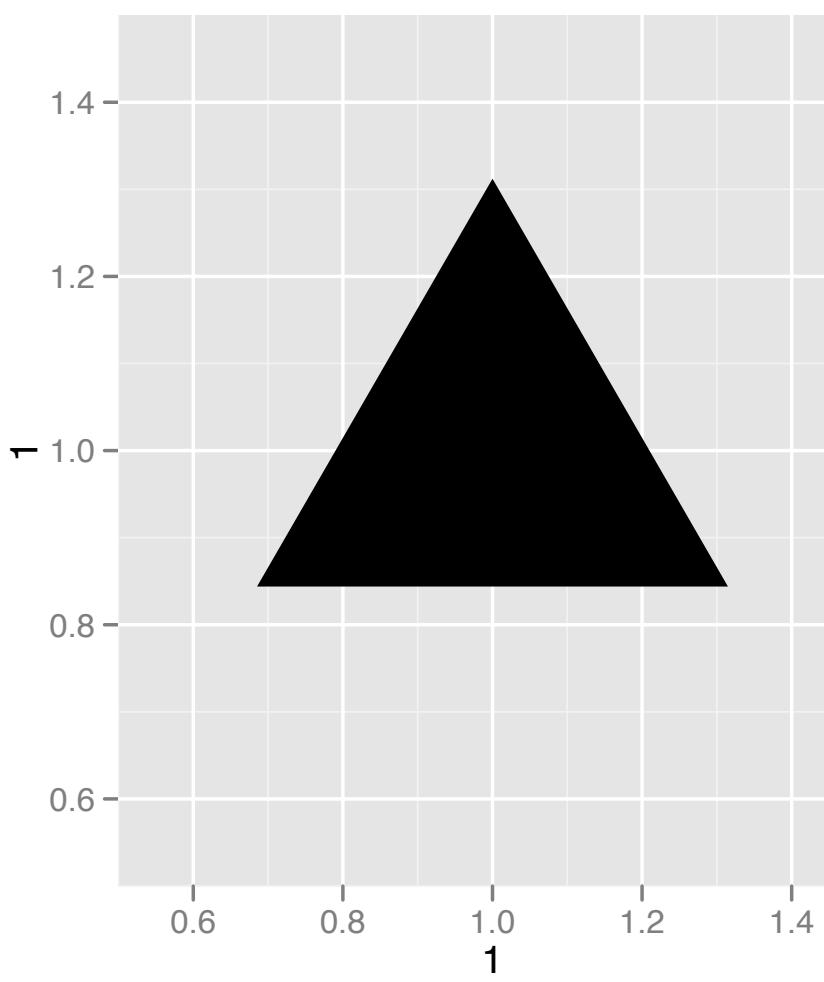
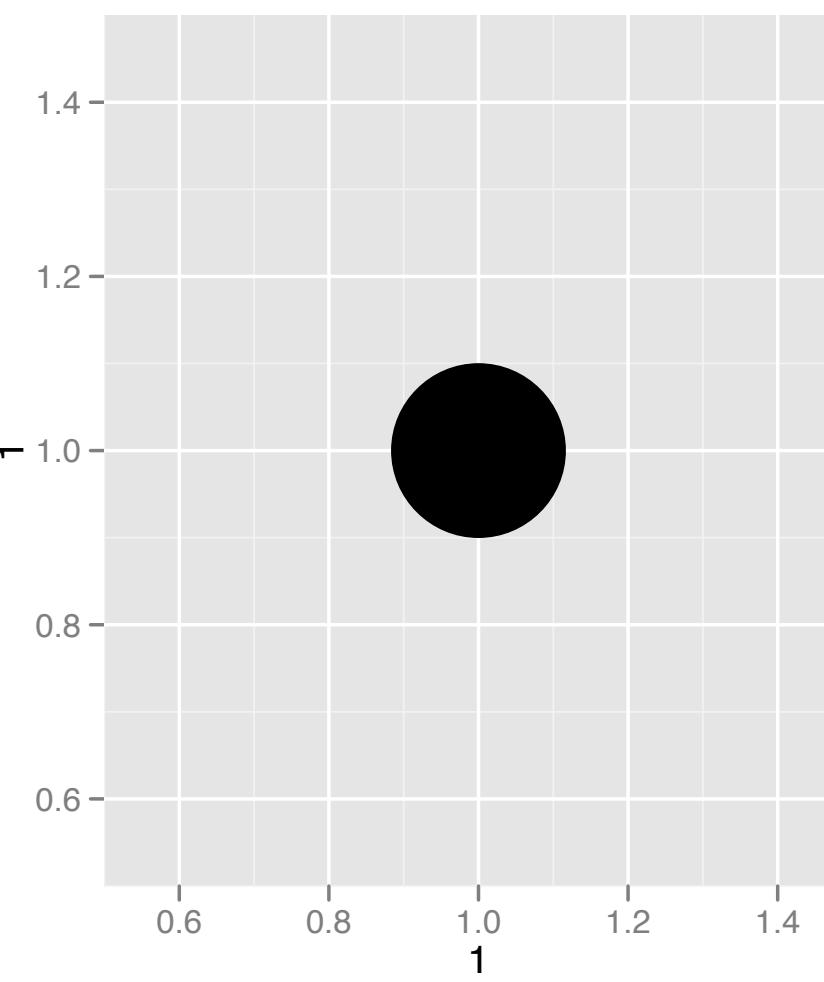
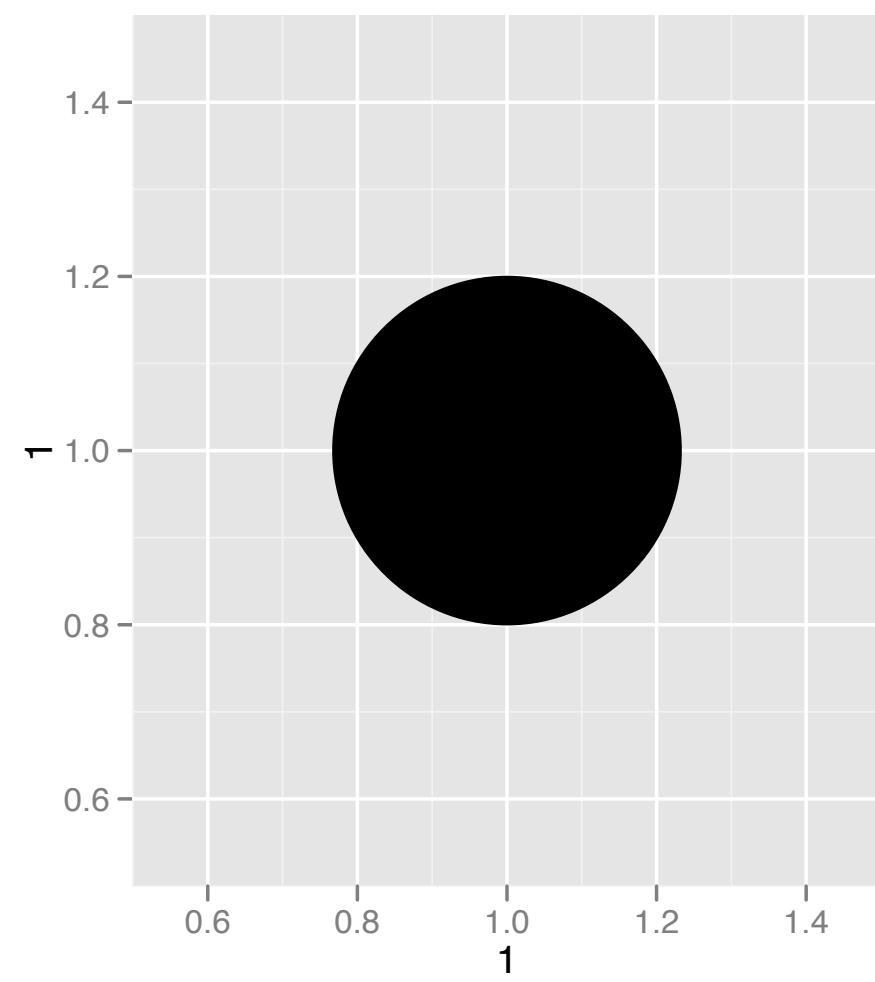
```
ggplot(data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```



# Aesthetics

R

# Aesthetics



## Visual Space

color

Purple

Blue

Teal

Lime

Yellow

## Data Space

clean\_test

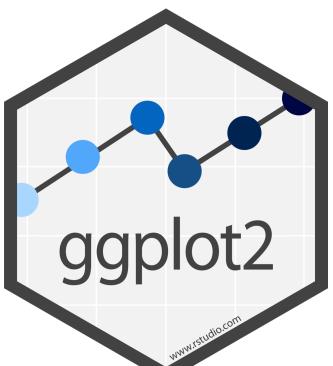
nowomen

notalk

men

dubious

ok



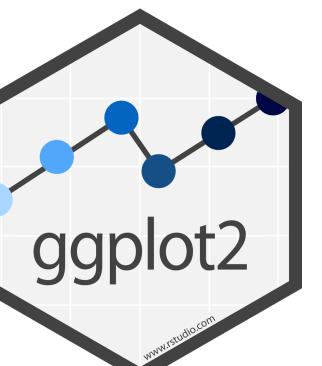
# Aesthetics

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

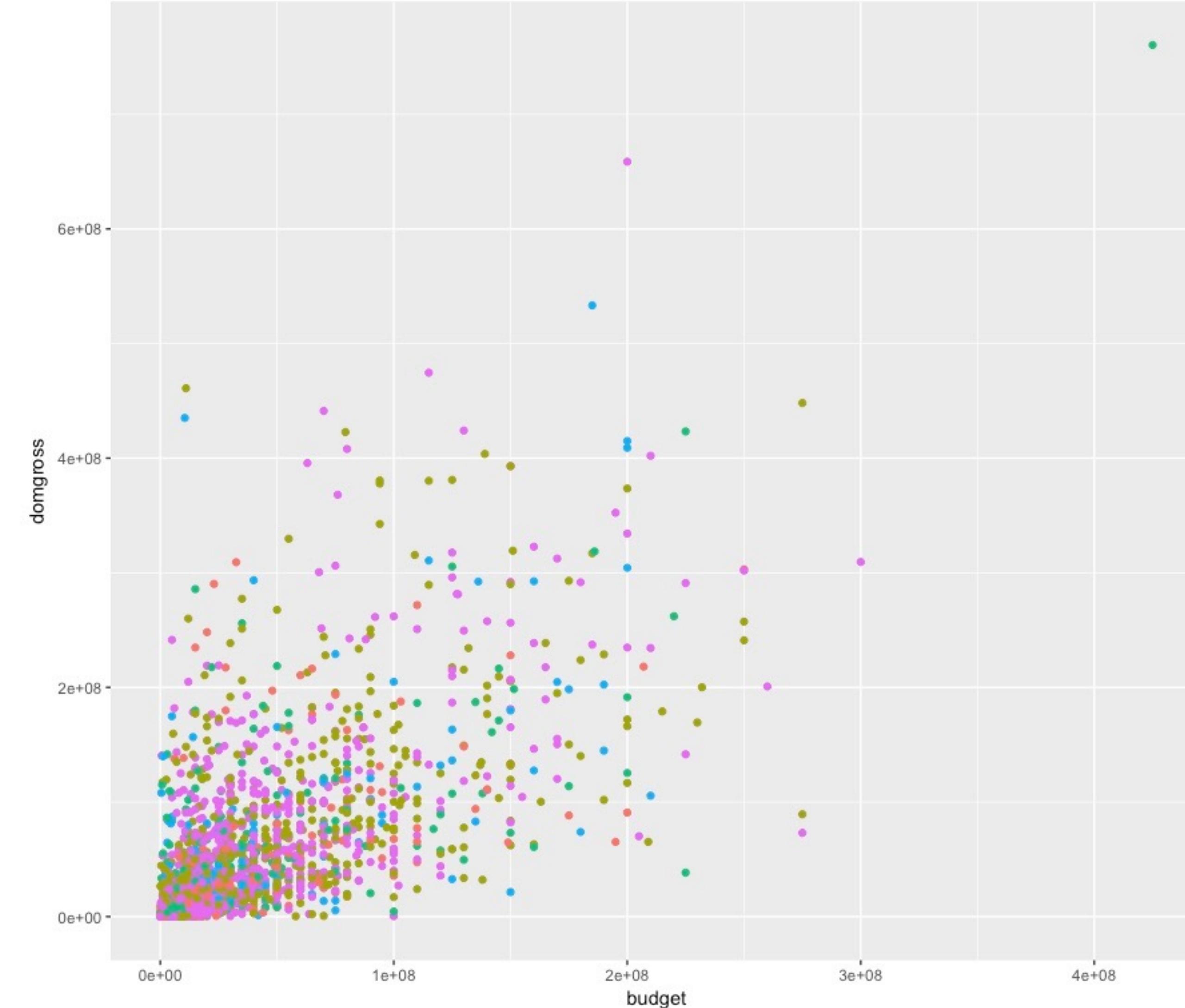
aesthetic  
property

Variable to  
map it to

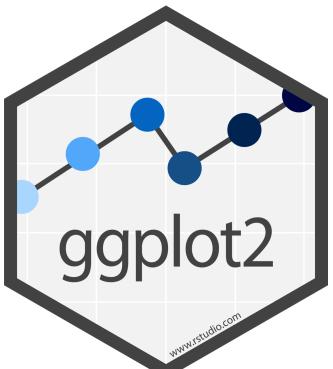
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, size = clean_test))
```



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



Legend added  
automatically



# Your Turn 2

recall

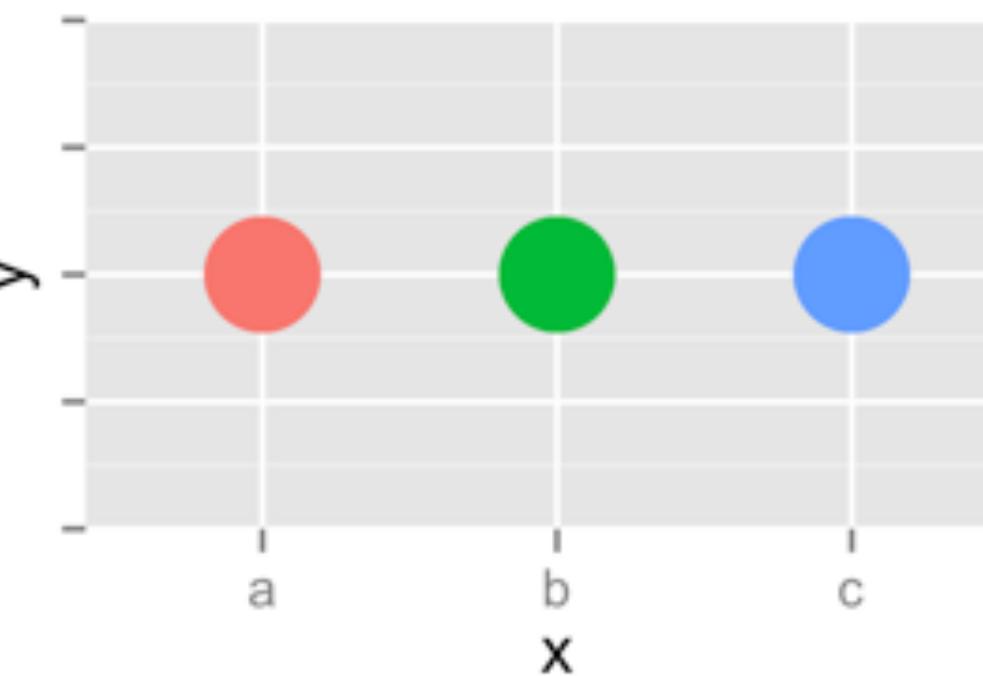
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color = clean_test))
```

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

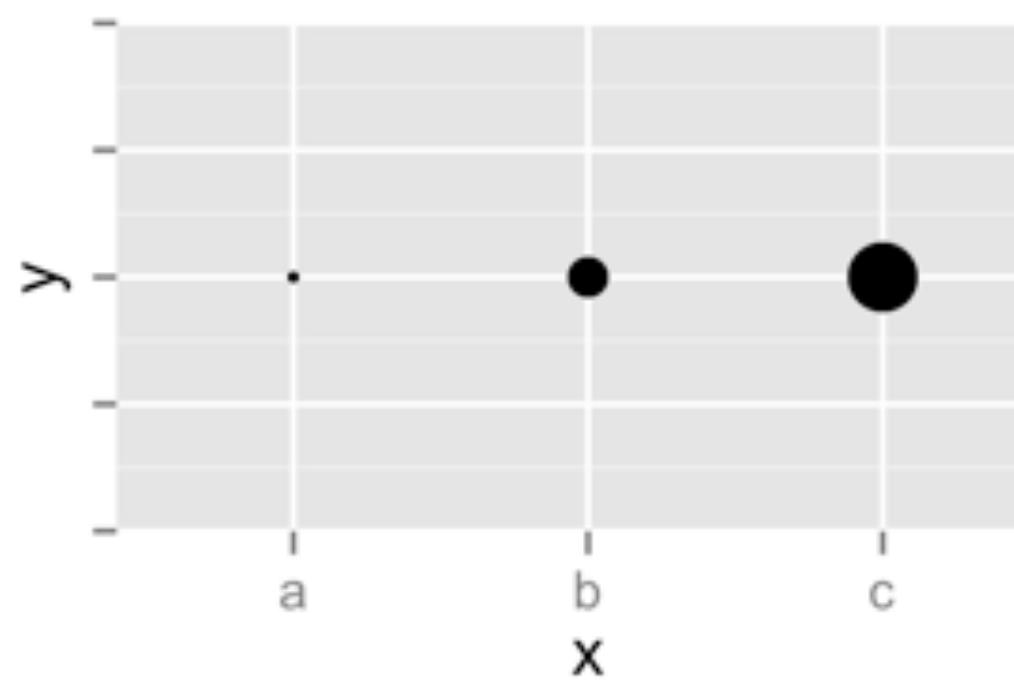
- Do different things happen when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?



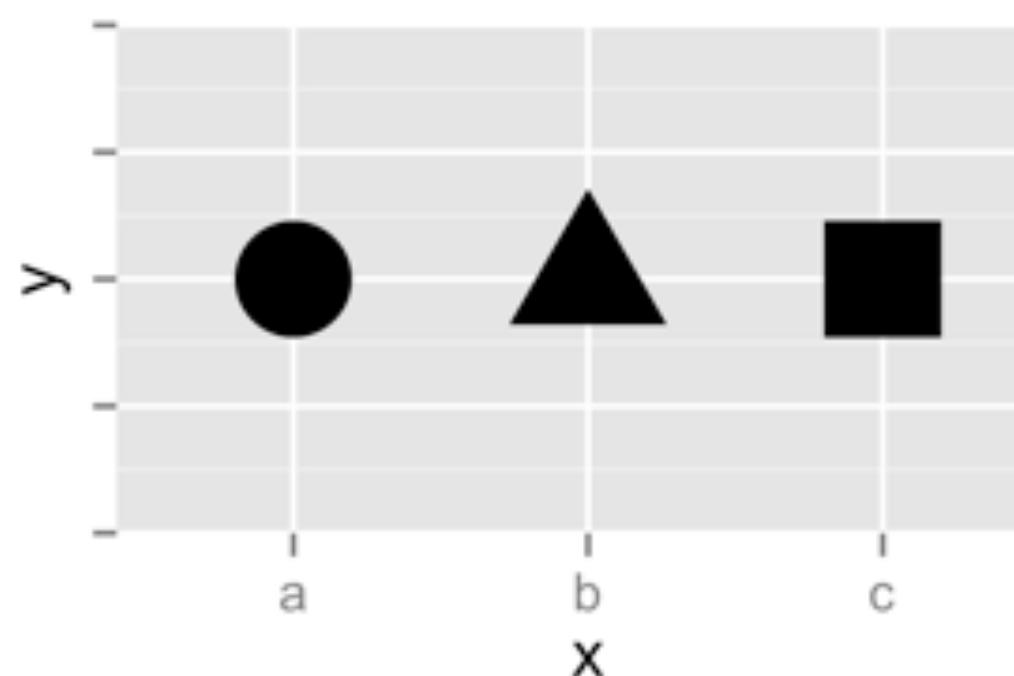
Color



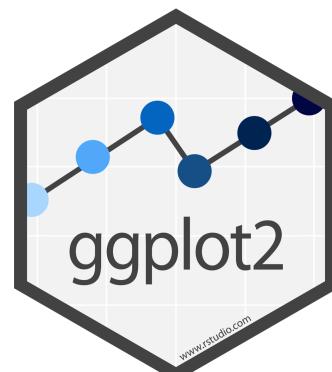
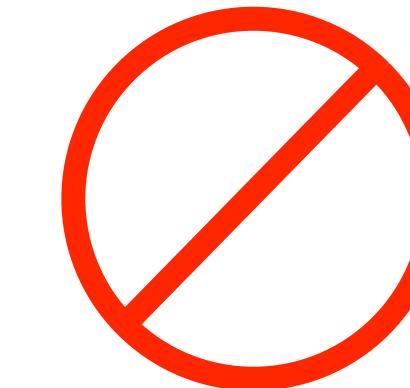
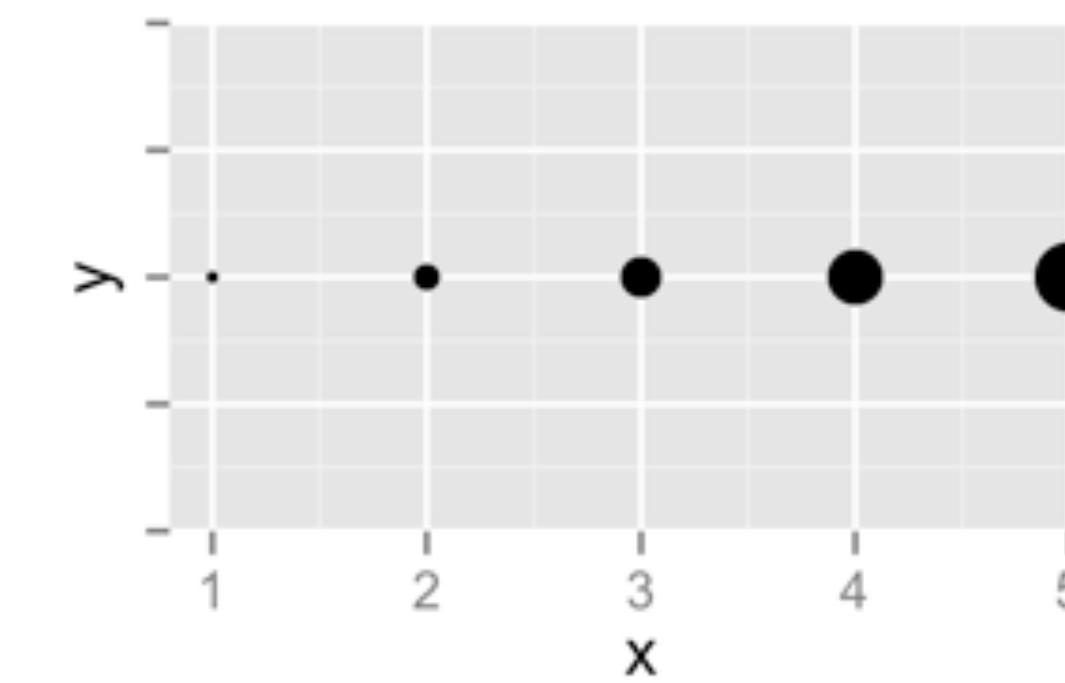
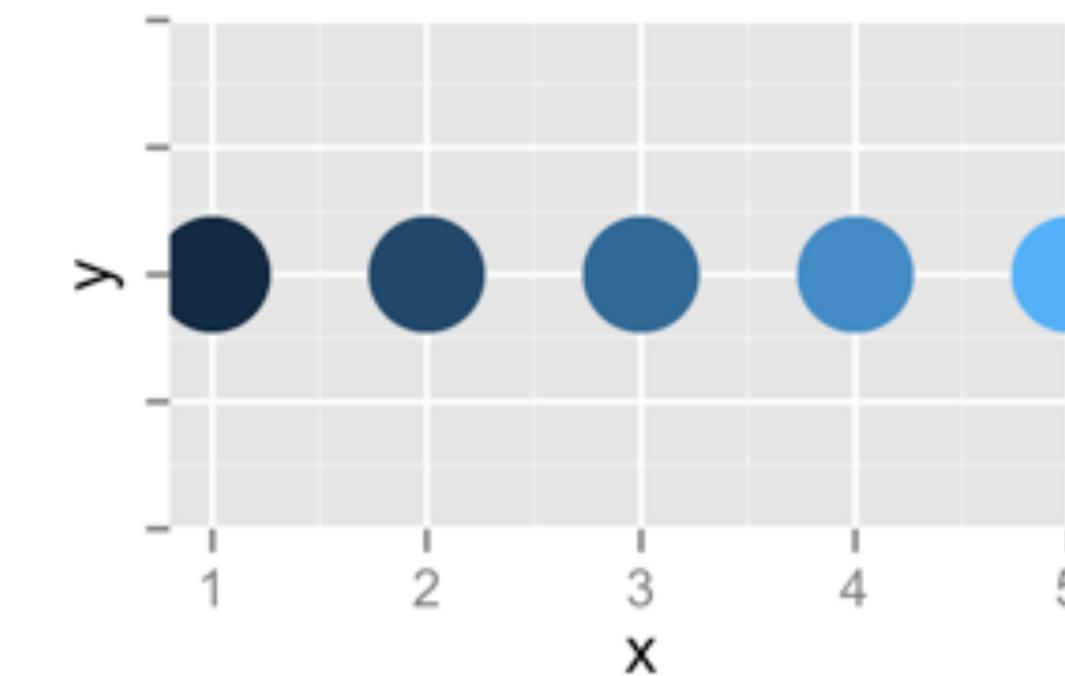
Size



Shape



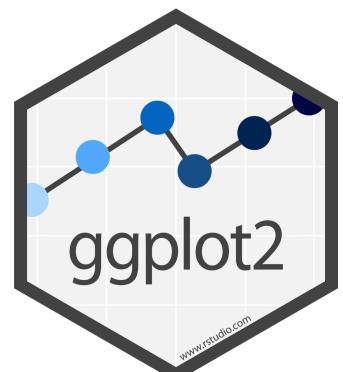
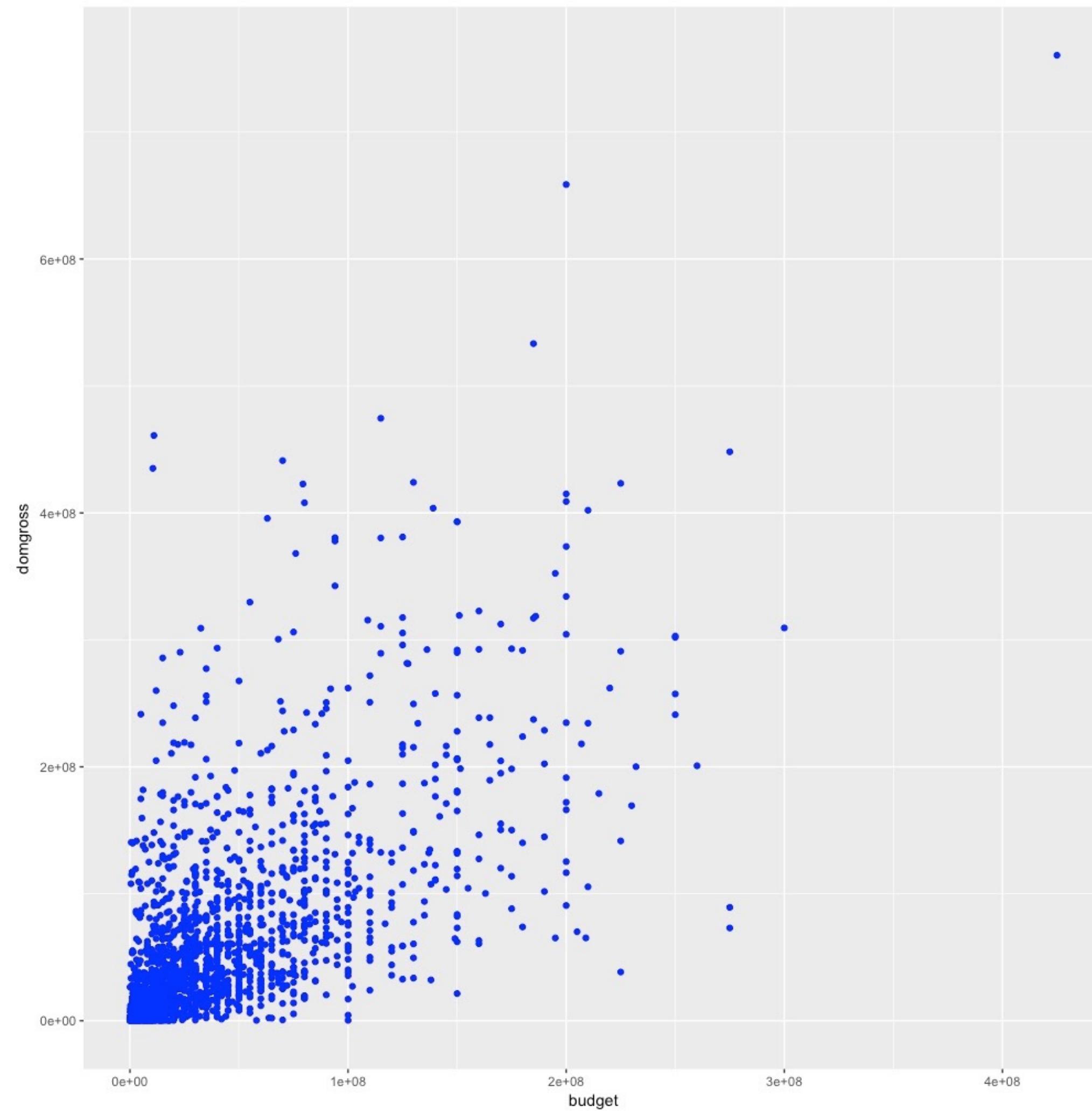
Continuous

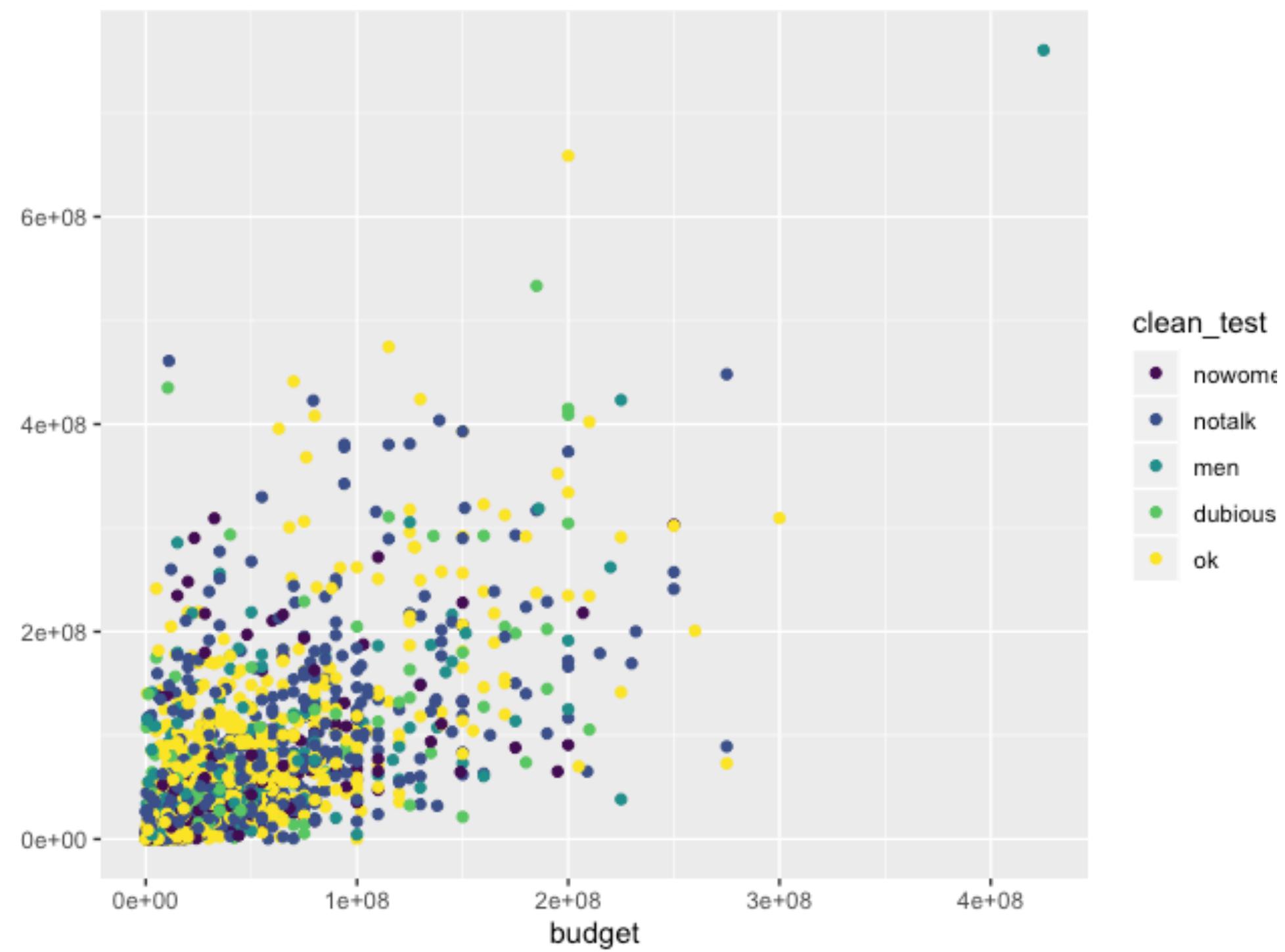


# set vs. map

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.

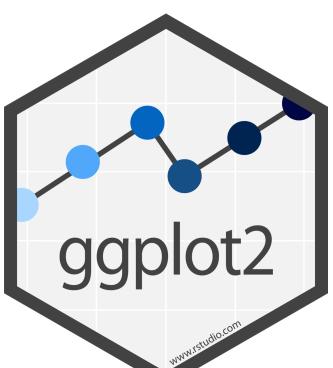
# How would you make this plot?



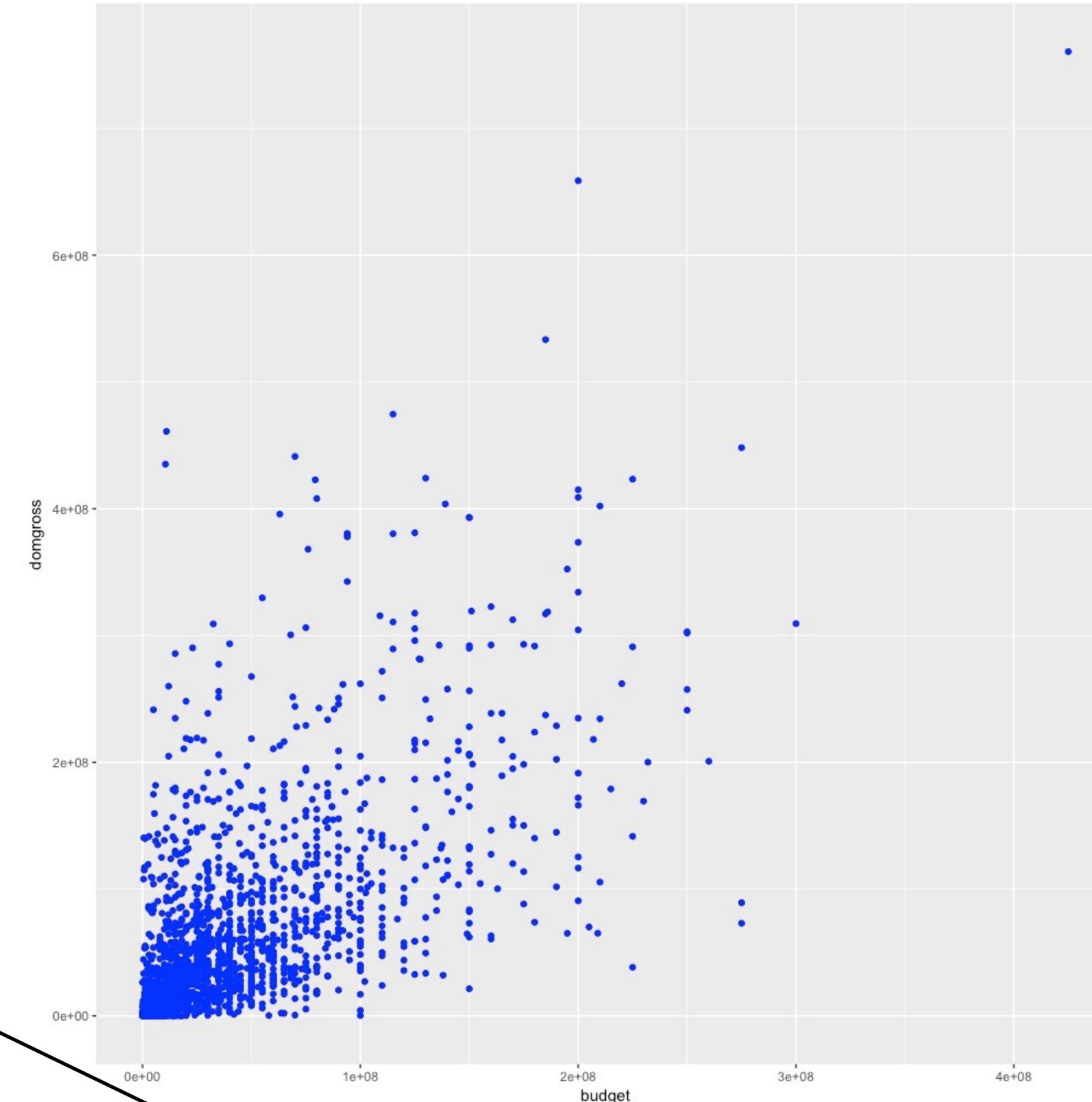


```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

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

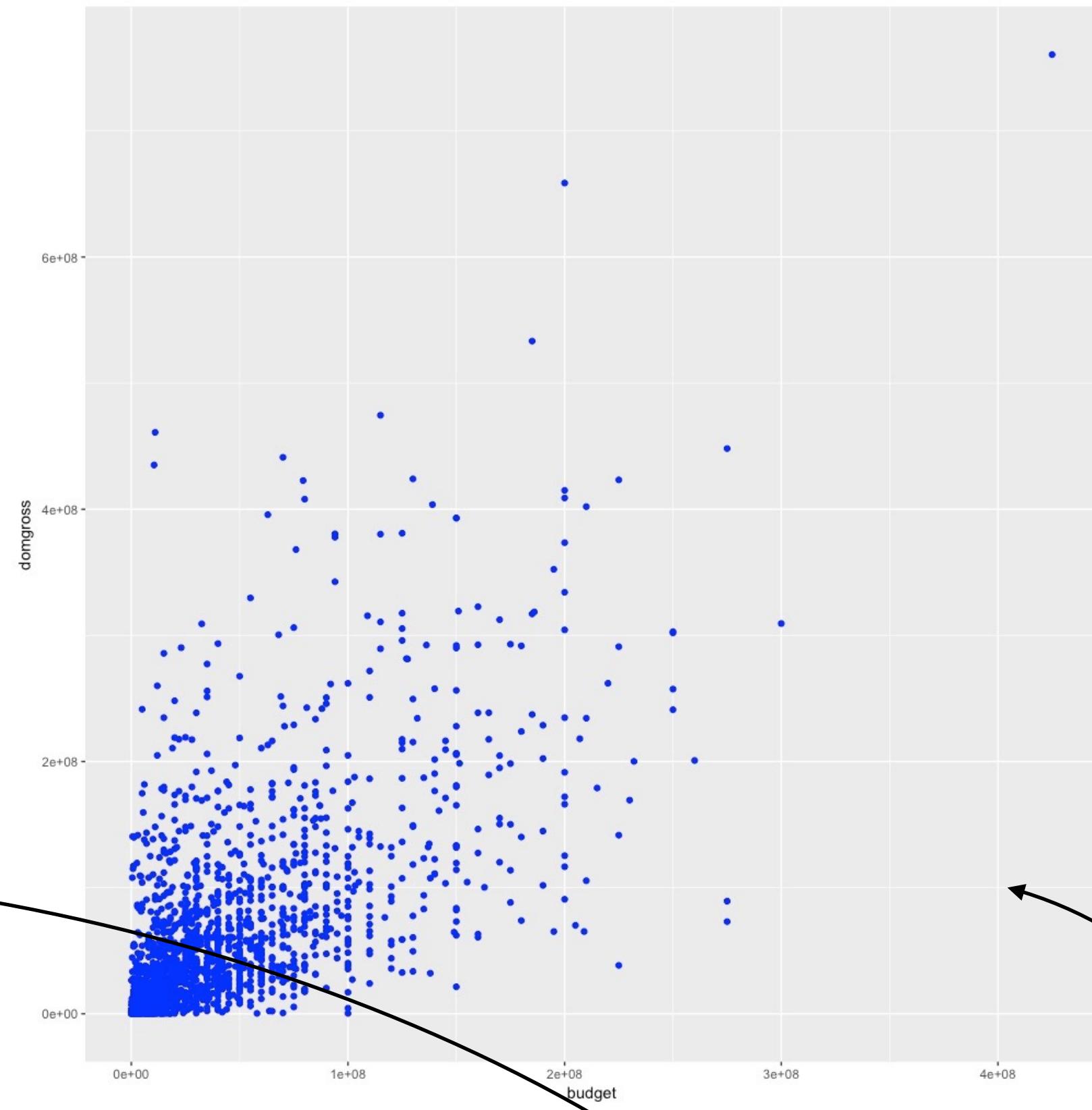
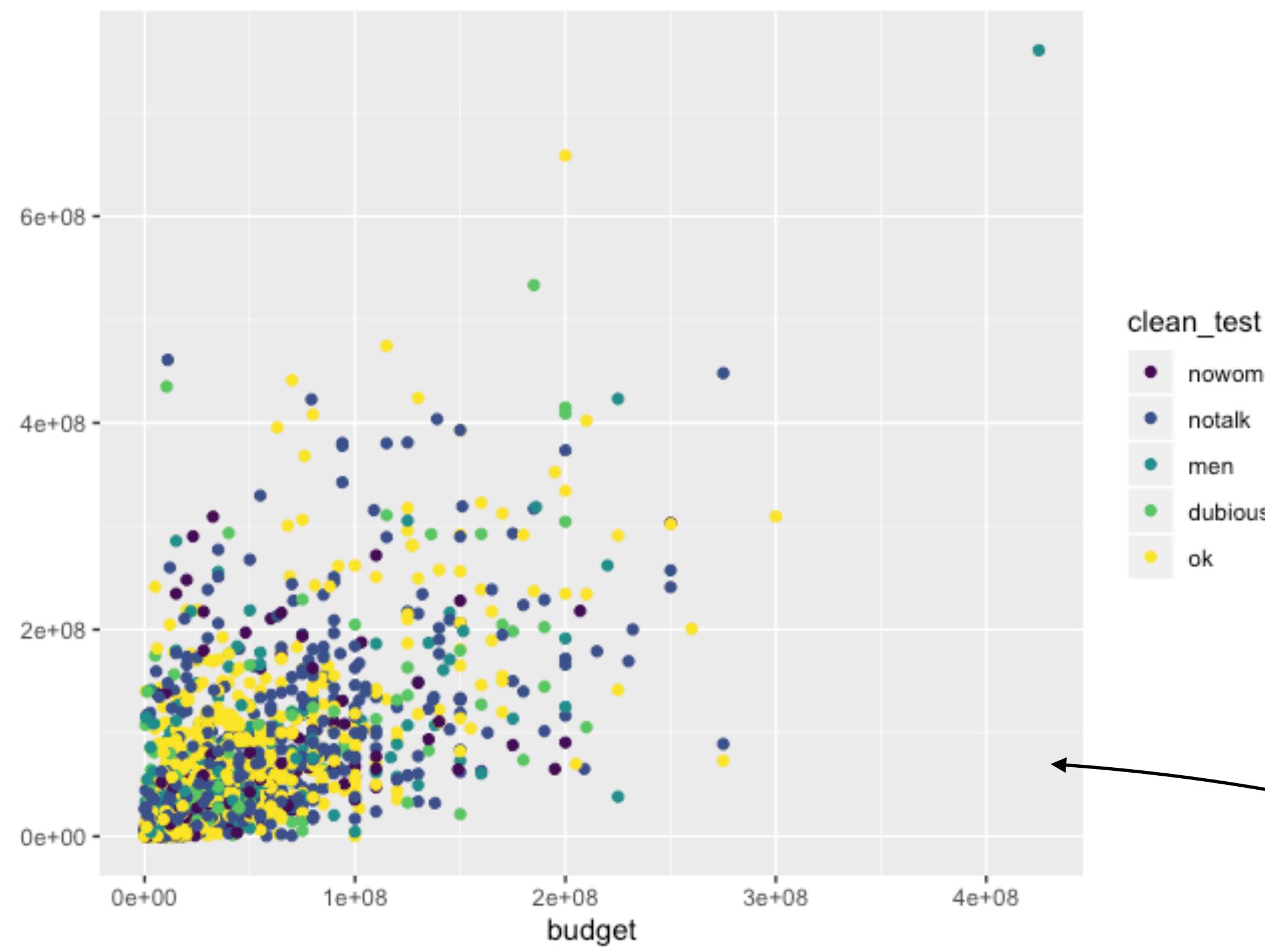


**Outside of aes():** sets  
an aesthetic to a value



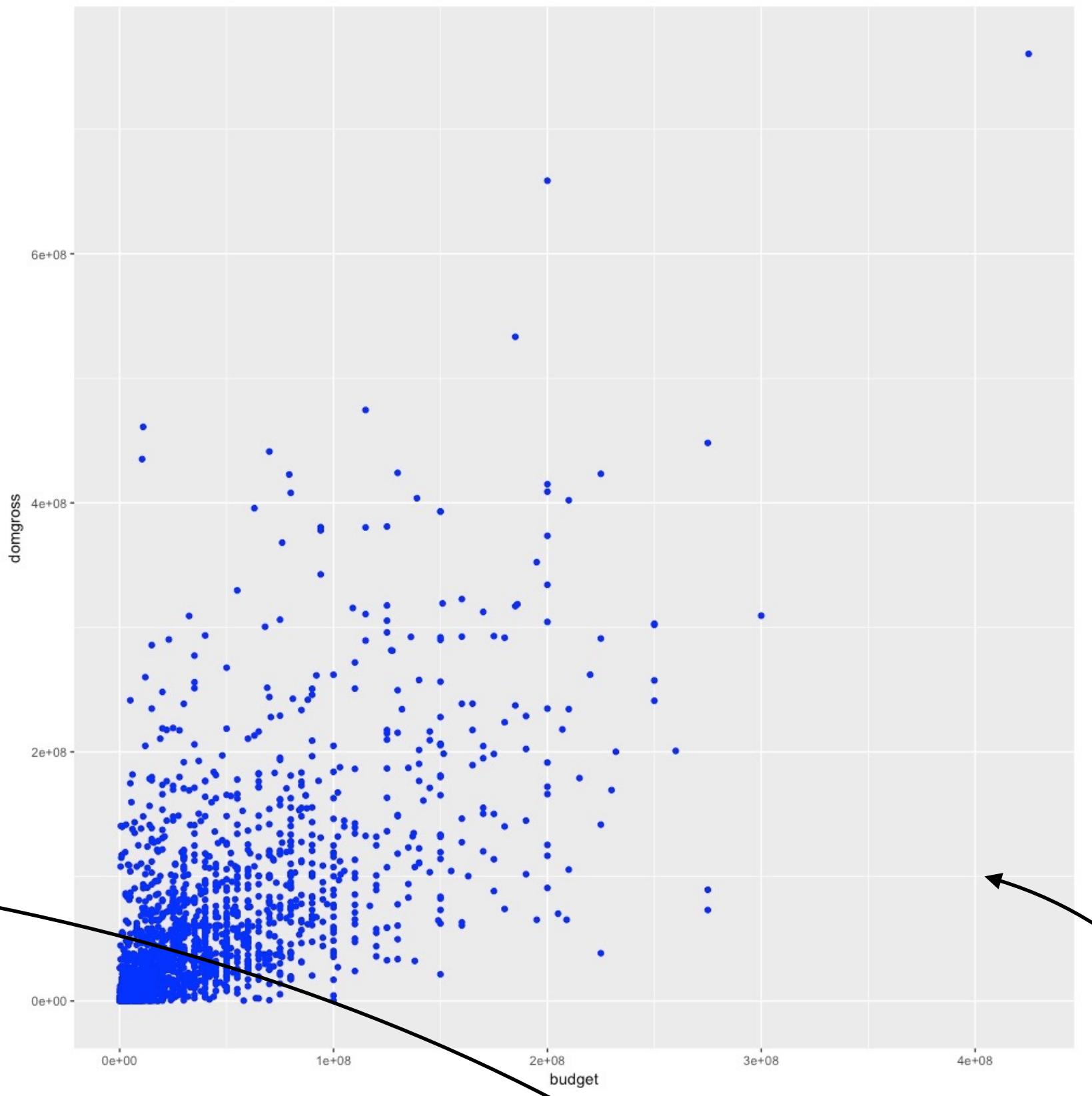
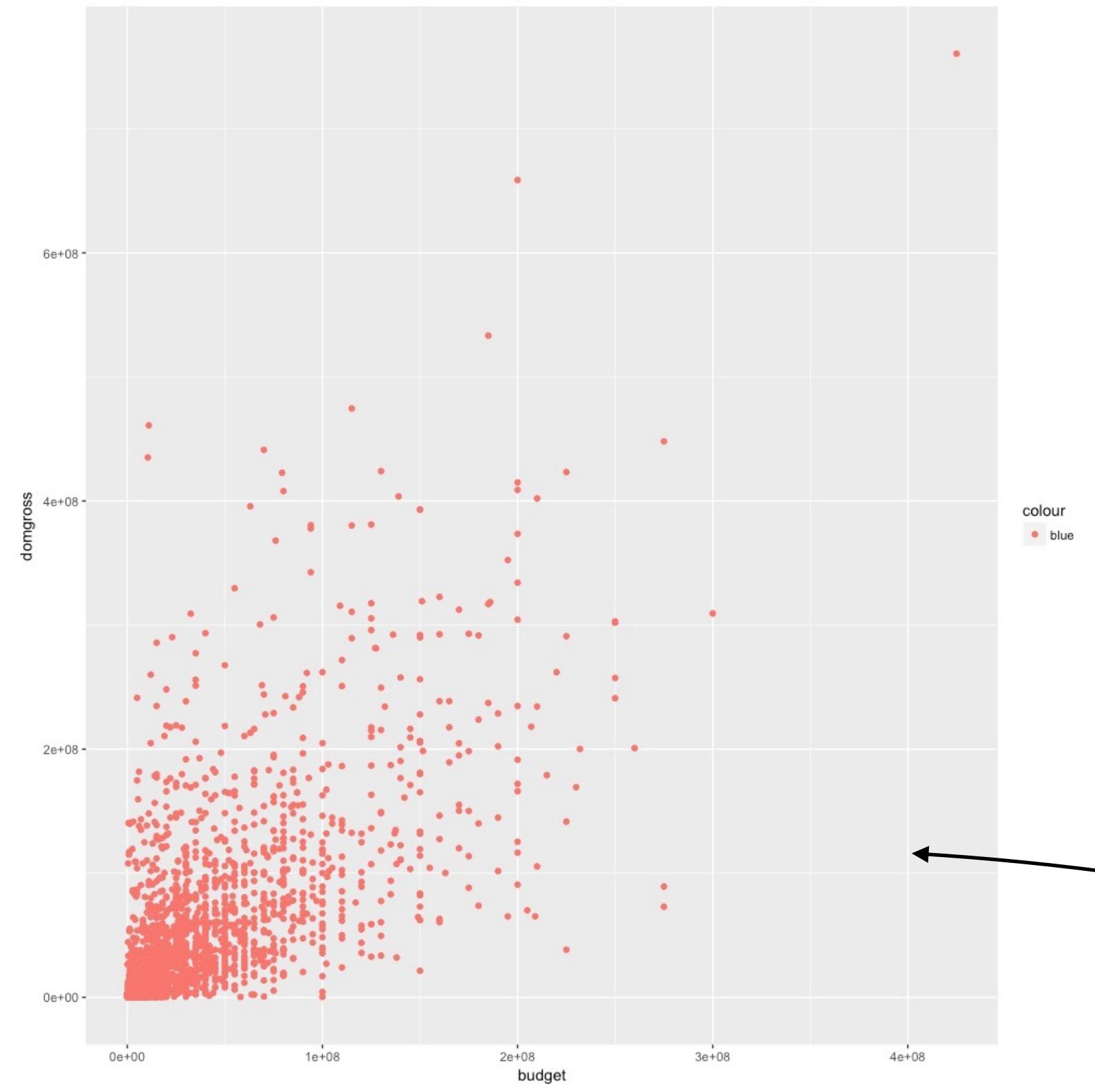
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
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"))
```

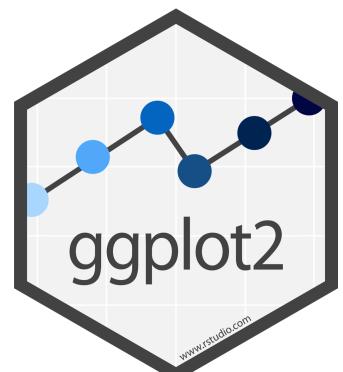
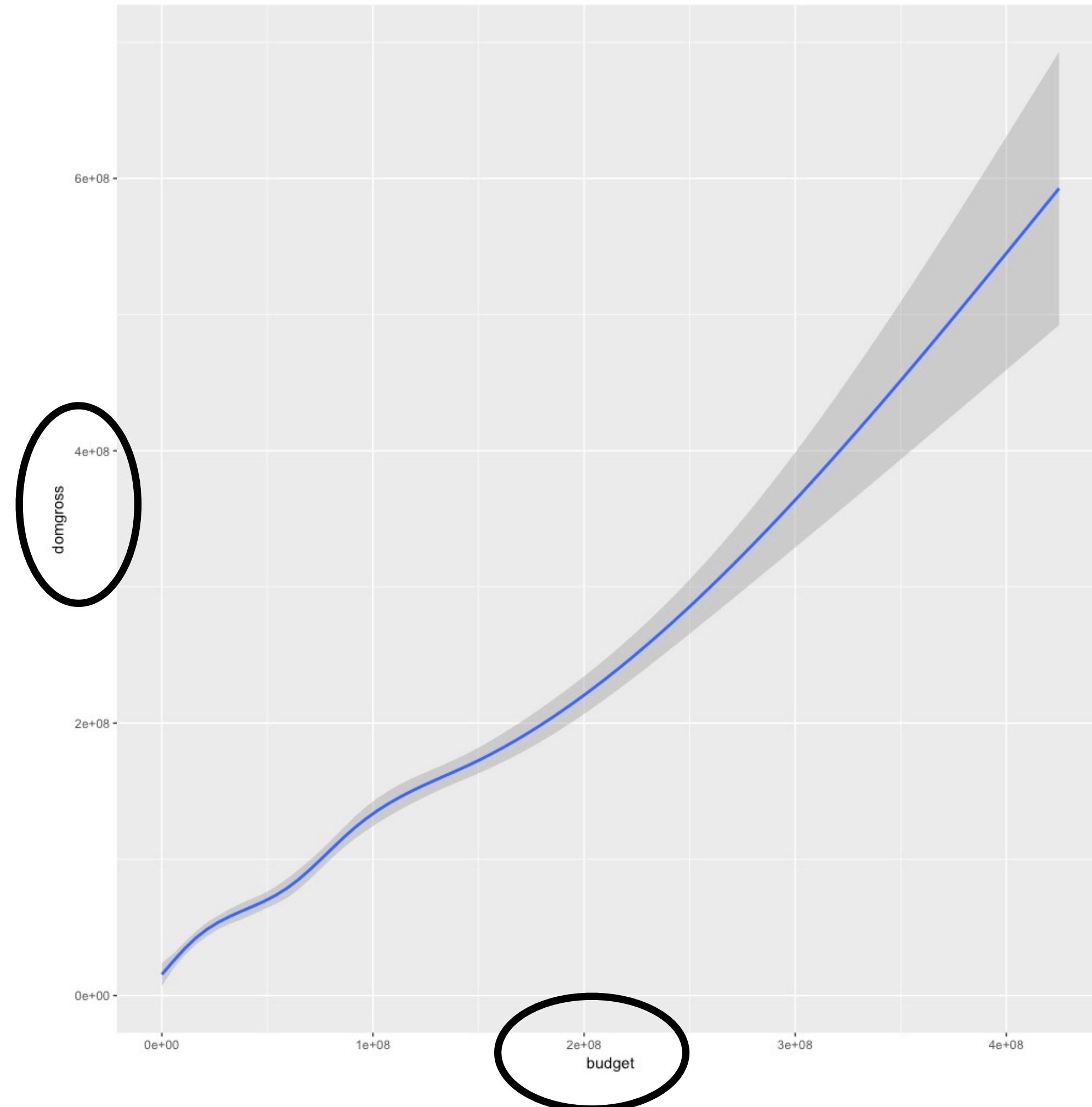
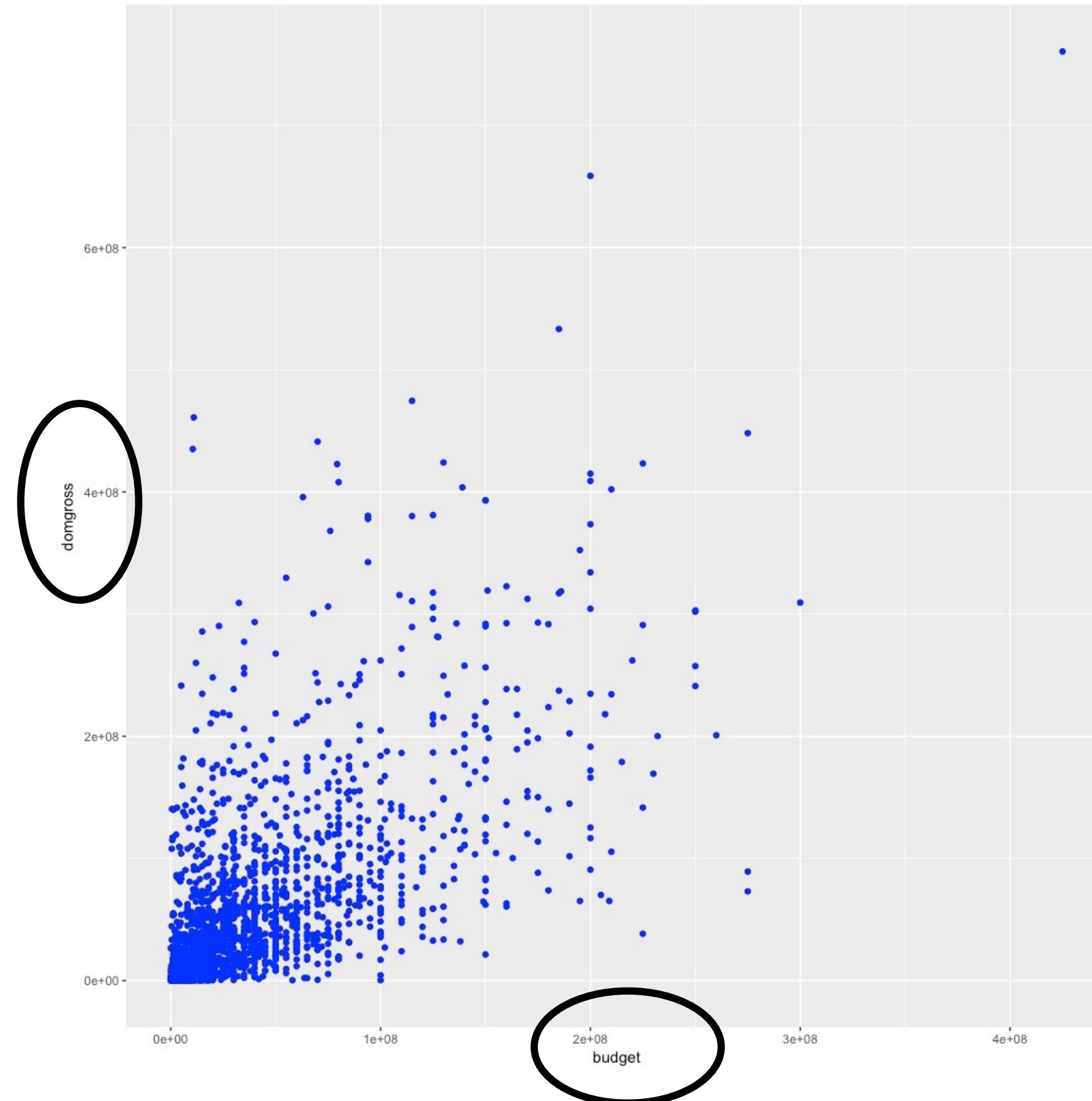
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

# geoms



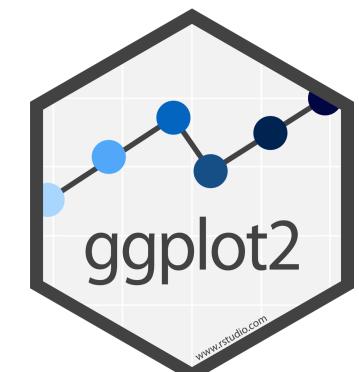
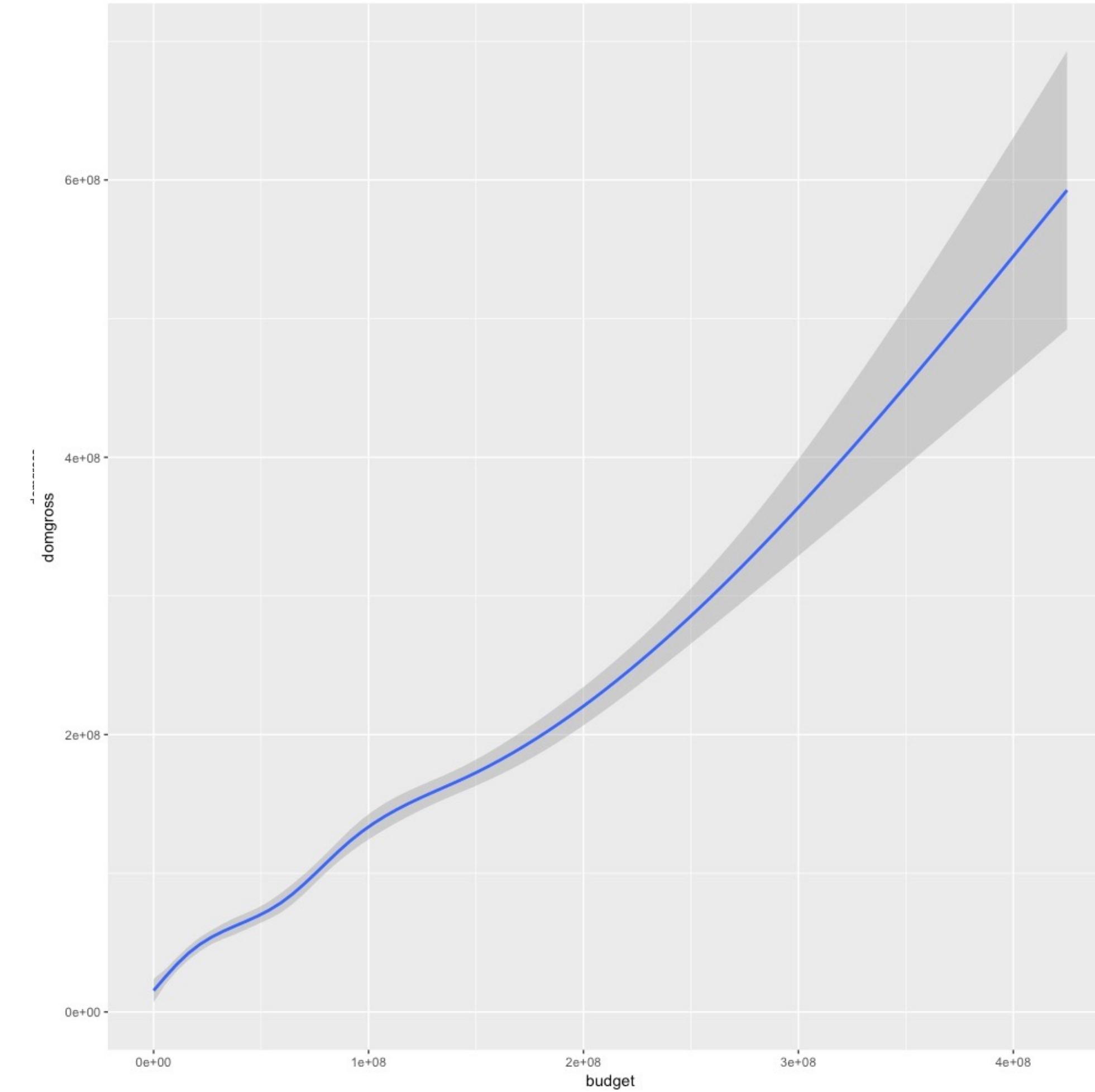
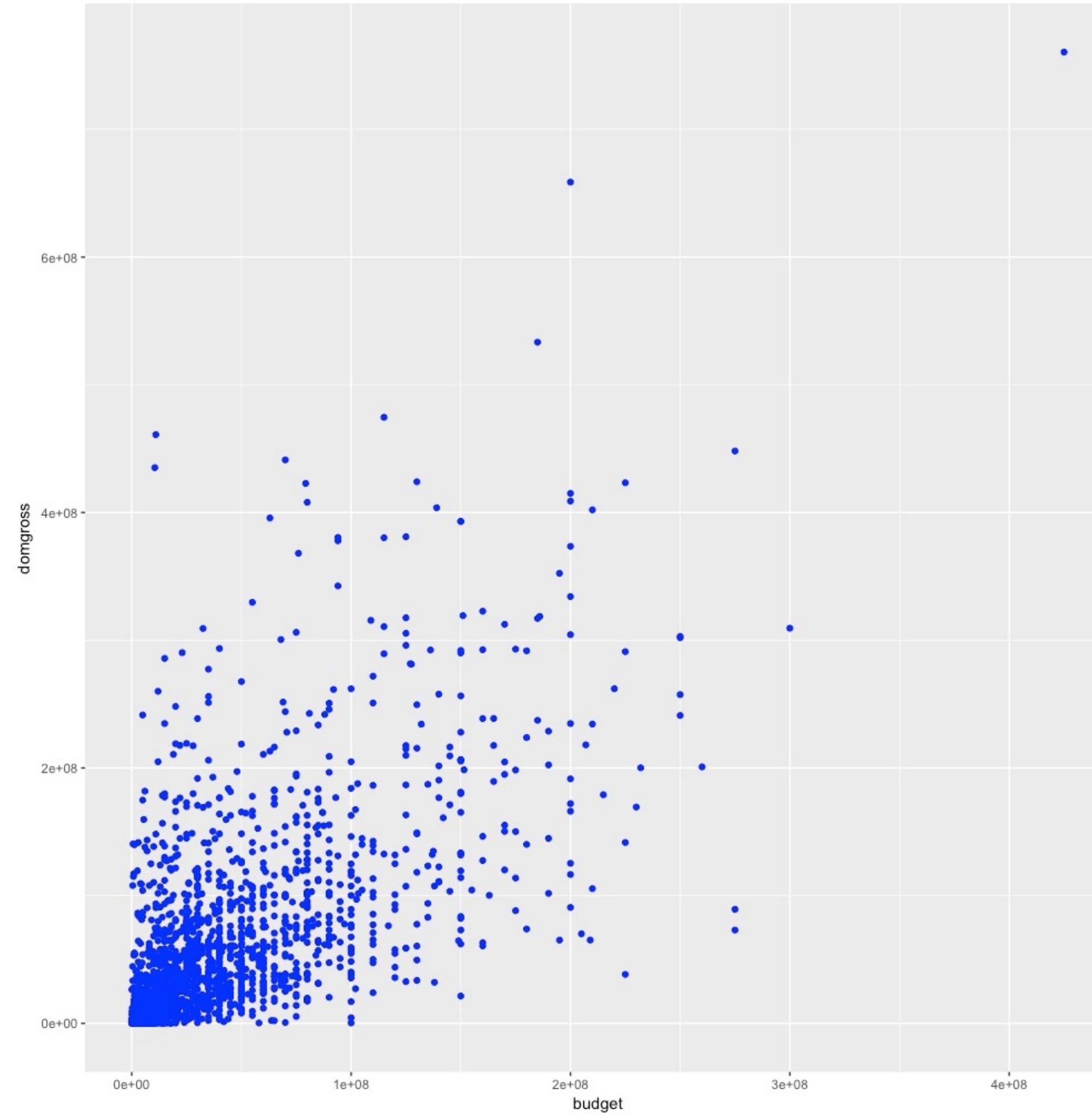
How are these plots similar?

Same: x var , y var , data



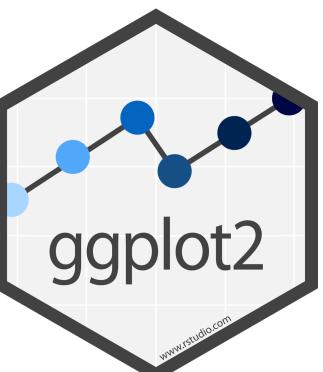
How are these plots different?

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(lineend = "butt", linejoin = "round", linemetre = 1)** x, y, alpha, color, group, linetype, size
- a + geom\_polygon(aes(group = group))** x, y, alpha, color, fill, group, linetype, size
- b + geom\_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + geom\_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))** - x, ymax, 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, ymax, ymin, alpha, color, 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)

**LINE SEGMENTS**

common aesthetics: x, y, alpha, color, linetype, size

```
b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:155, radius = 1))
```

**ONE VARIABLE continuous**

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

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

**discrete x , continuous y**

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

- f + geom\_col()** x, y, alpha, color, fill, group, linetype, size
- f + geom\_boxplot()**, x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- f + geom\_dotplot(binaxis = "y", stackdir = "center")** x, y, alpha, color, fill, group
- f + geom\_violin(scale = "area")** x, y, alpha, color, fill, group, linetype, size, weight

**discrete x , discrete y**

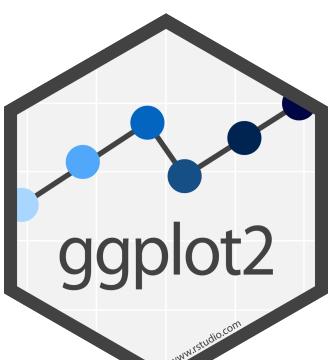
```
g <- ggplot(diamonds, aes(cut, color))
```

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

**THREE VARIABLES**

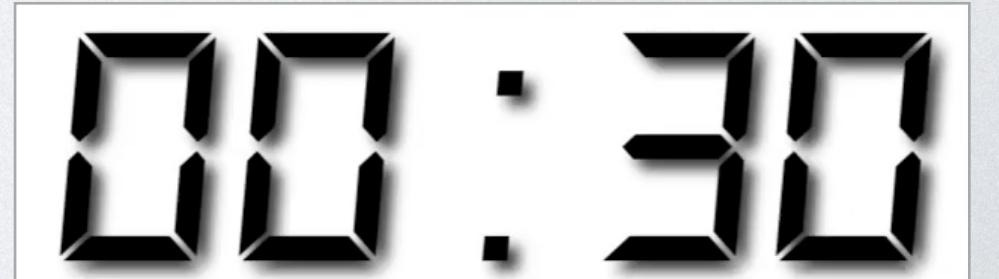
```
sealsSz <- with(seals, sqrt(delta_long^2 + delta_lat^2))
```

- l <- ggplot(seals, aes(long, lat))**
- l + geom\_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.

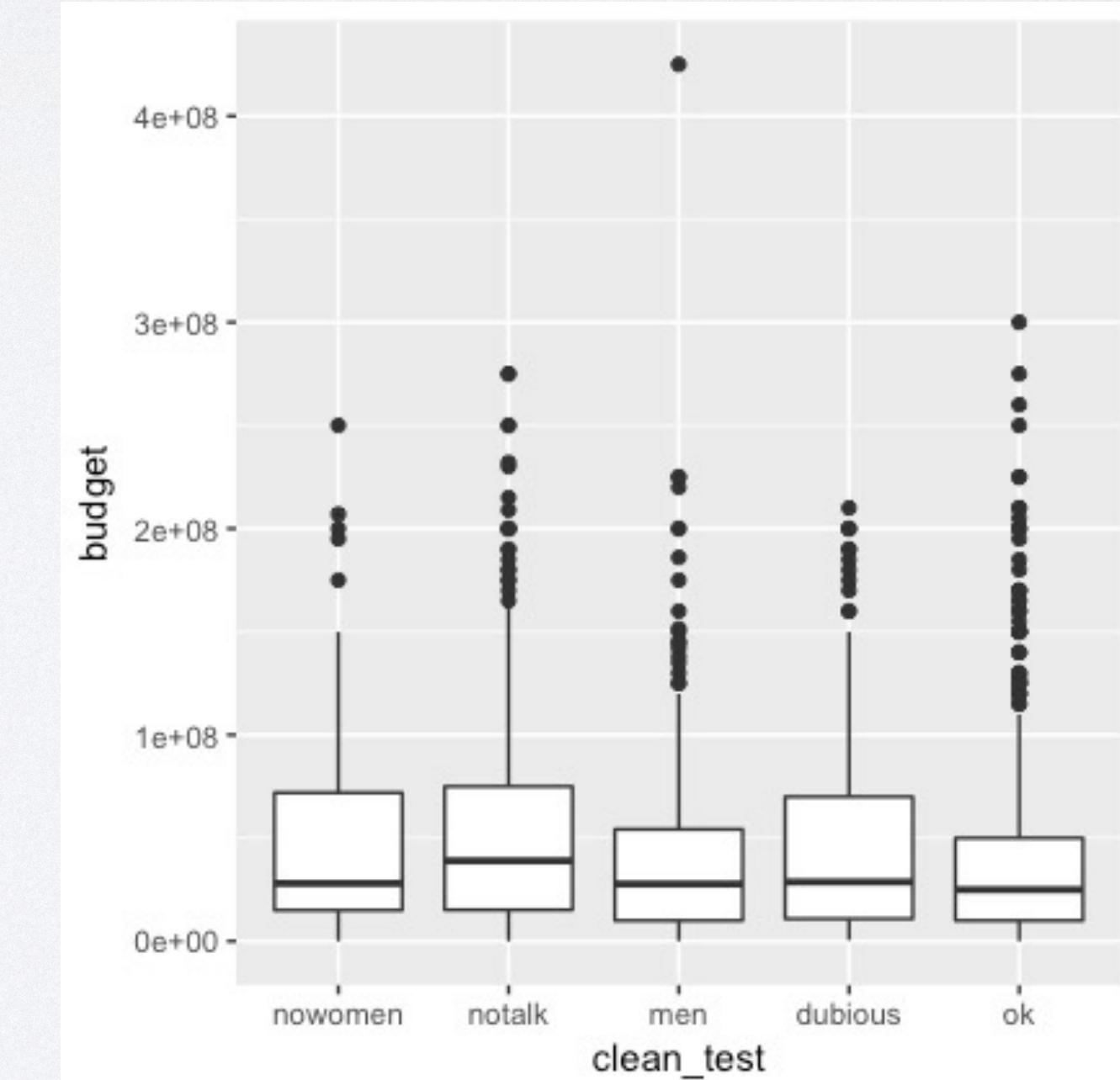
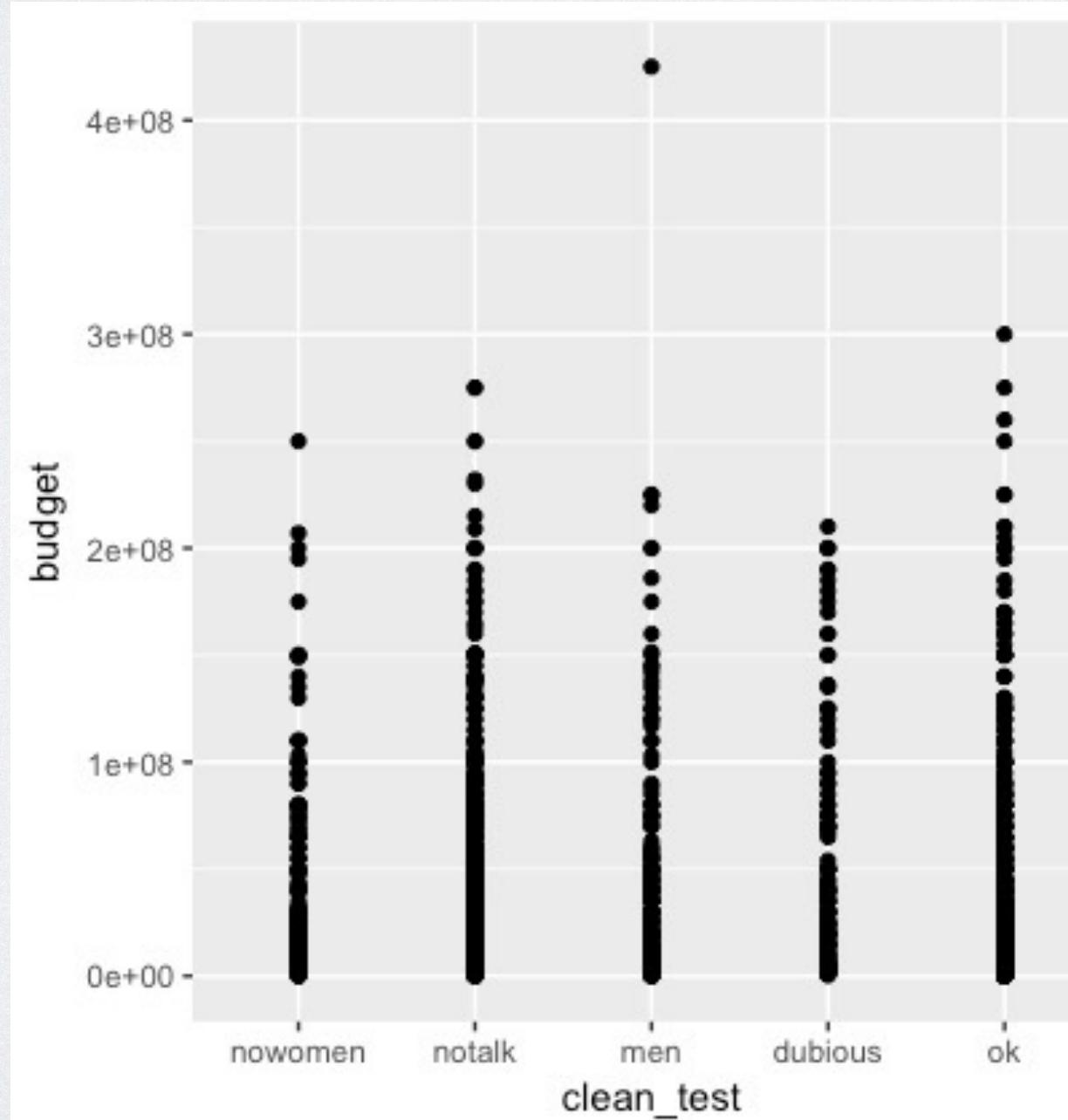


# Your Turn 3

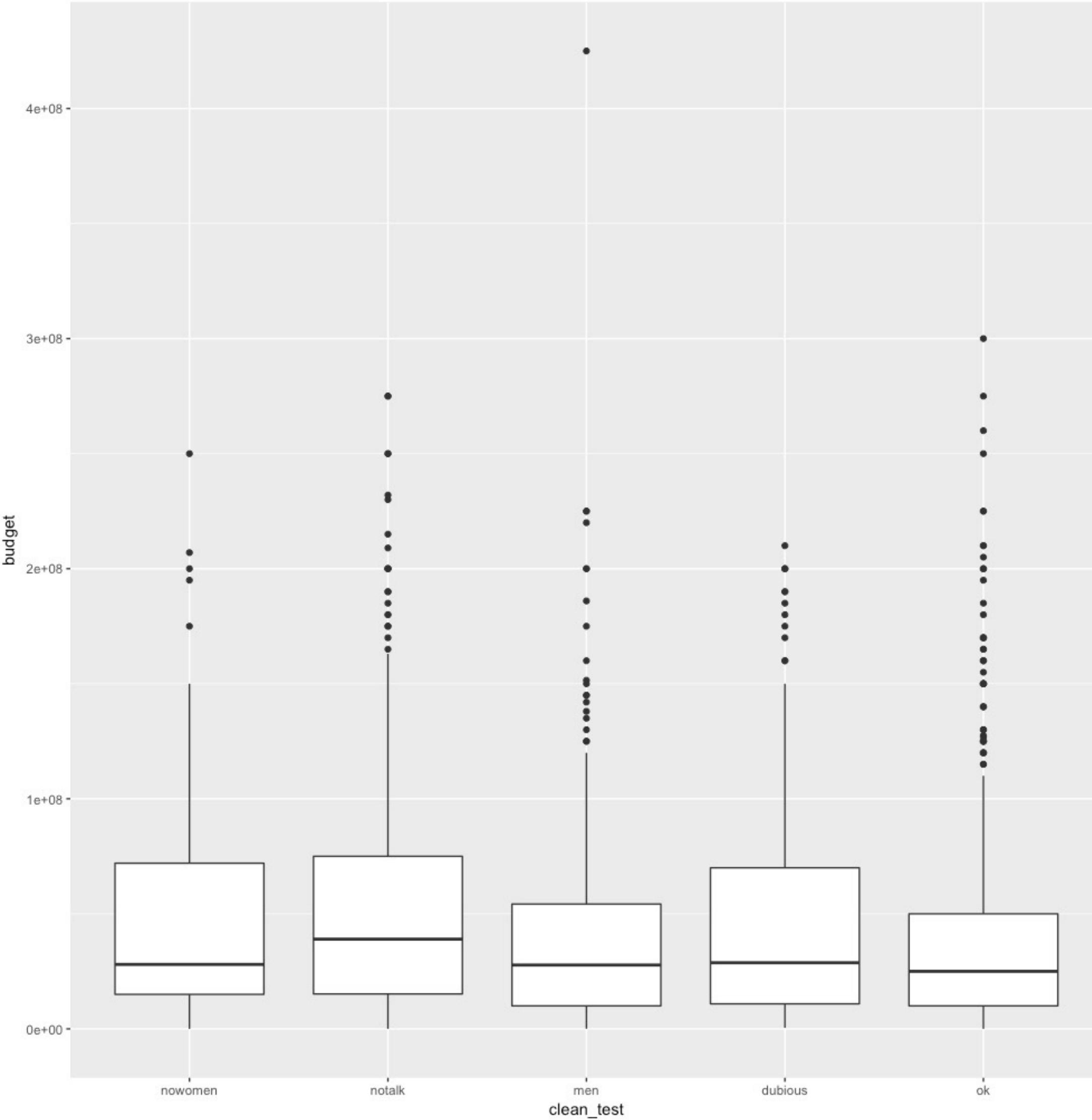
**recall**

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

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



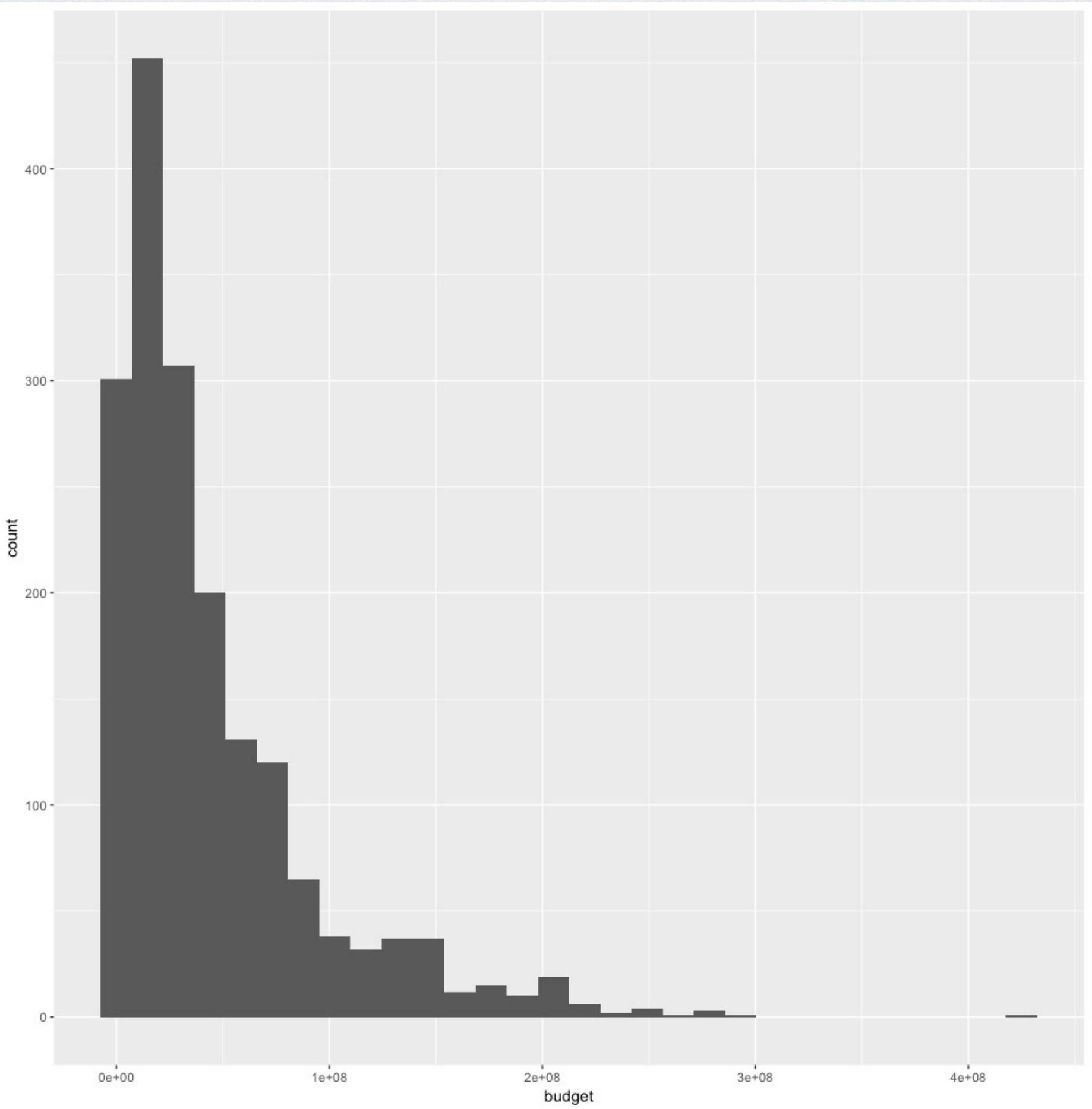
02 : 00

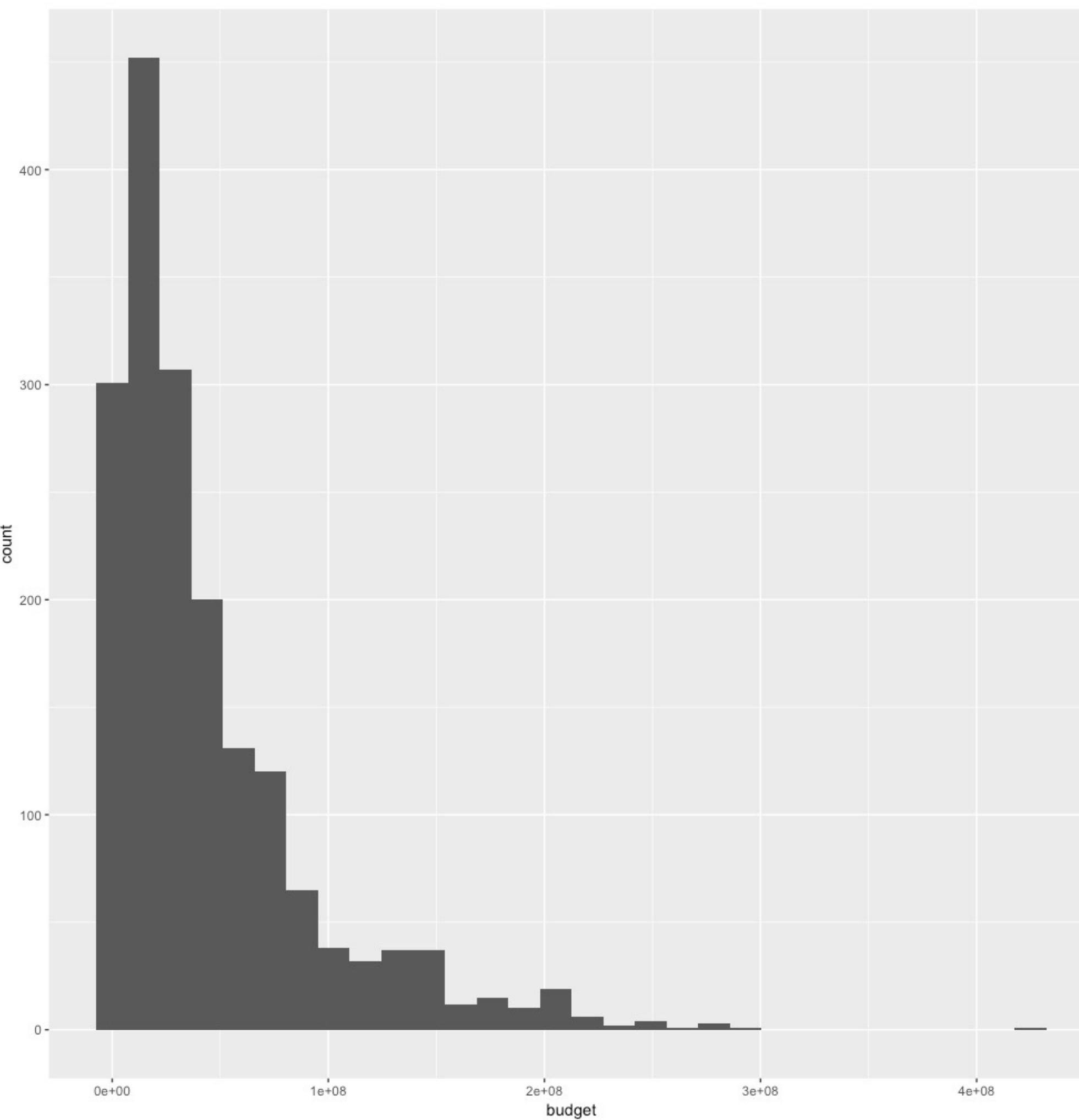


```
ggplot(data = bechdel) +  
  geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

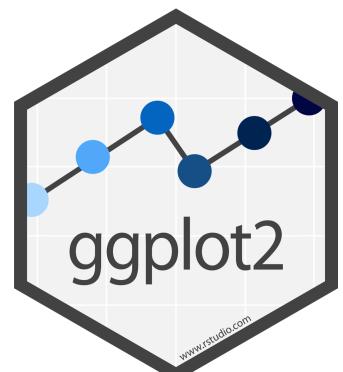
# Your Turn 4

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

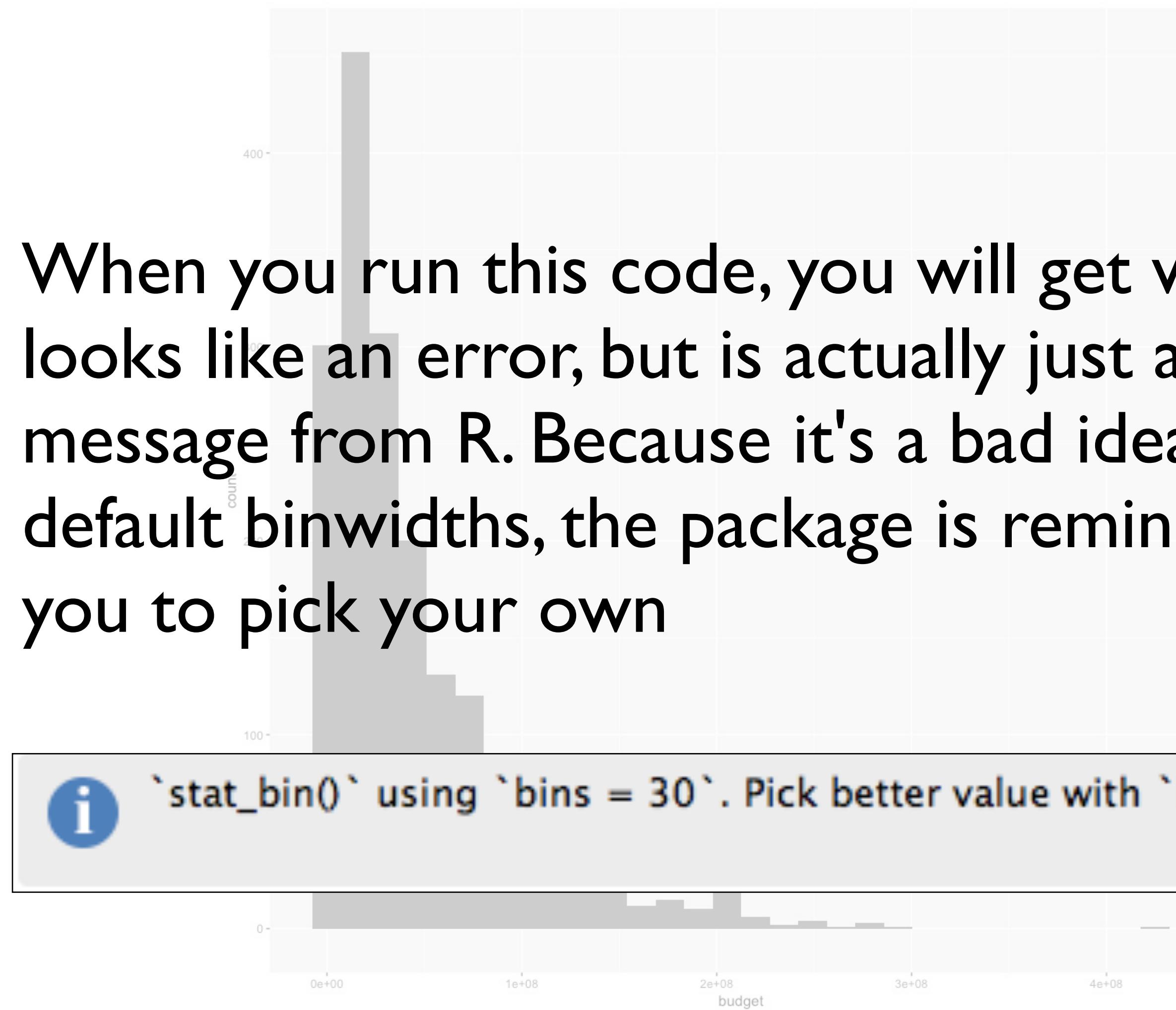




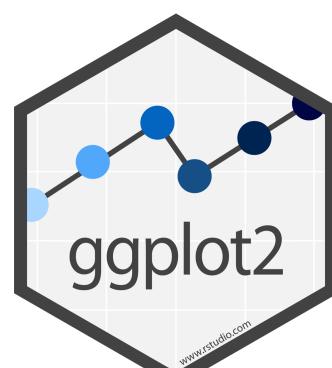
```
ggplot(data = bechdel) +  
  geom_histogram(mapping = aes(x = budget))
```



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))
```

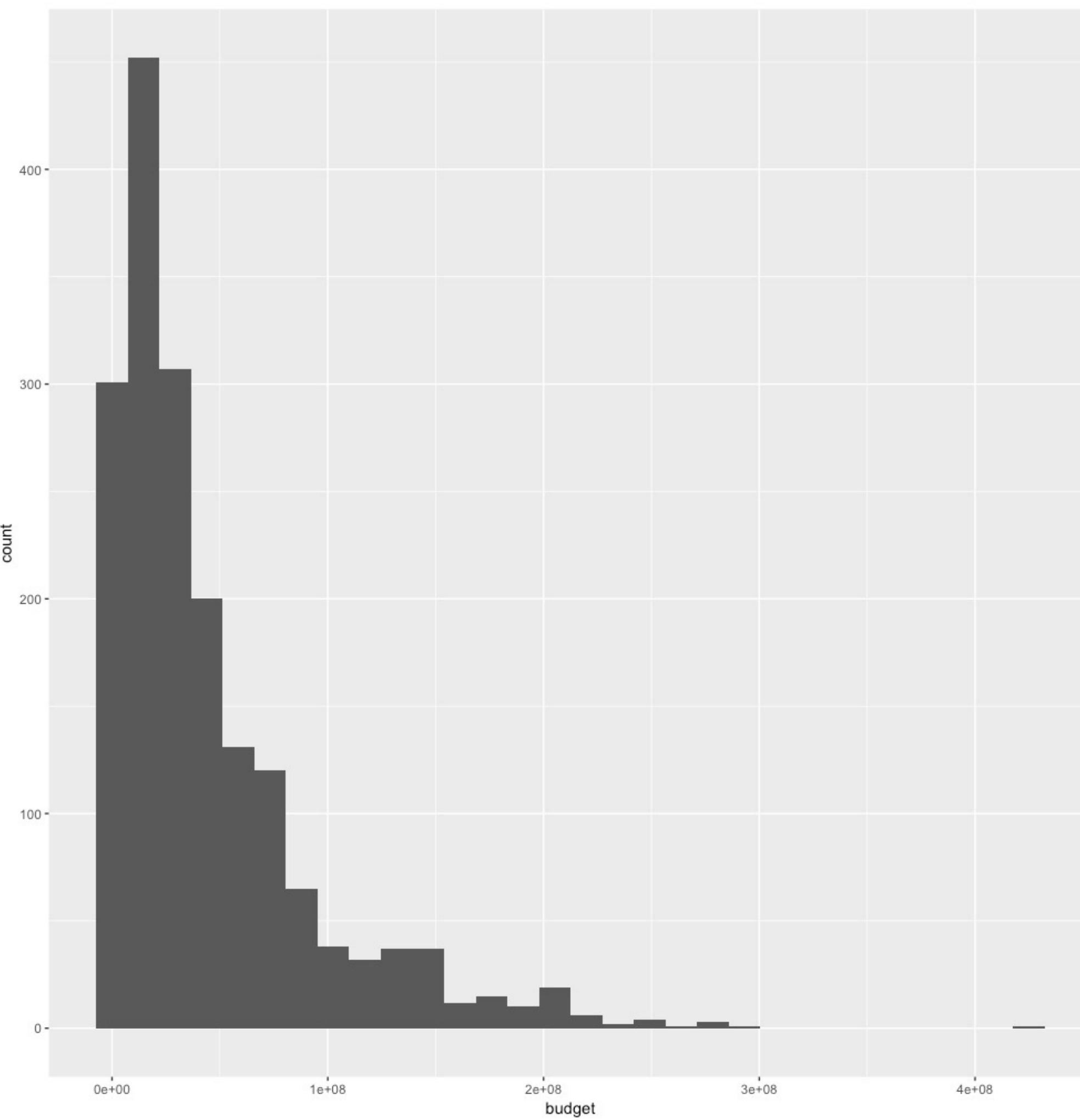


# Your Turn 5

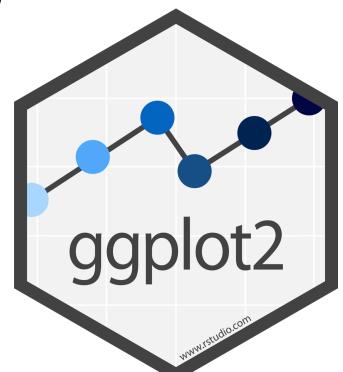
What would be a reasonable  
binwidth for budget?

Try it out



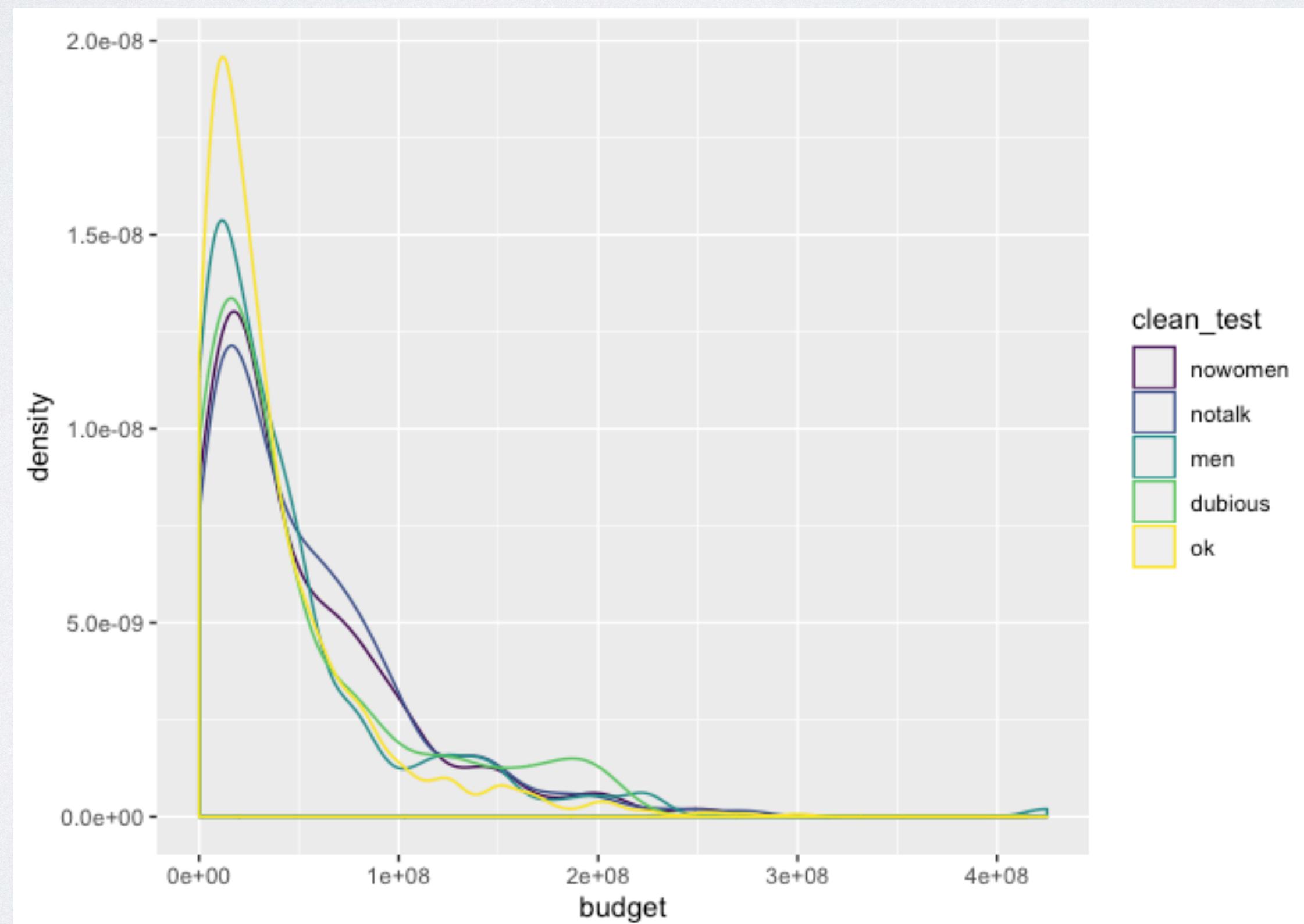


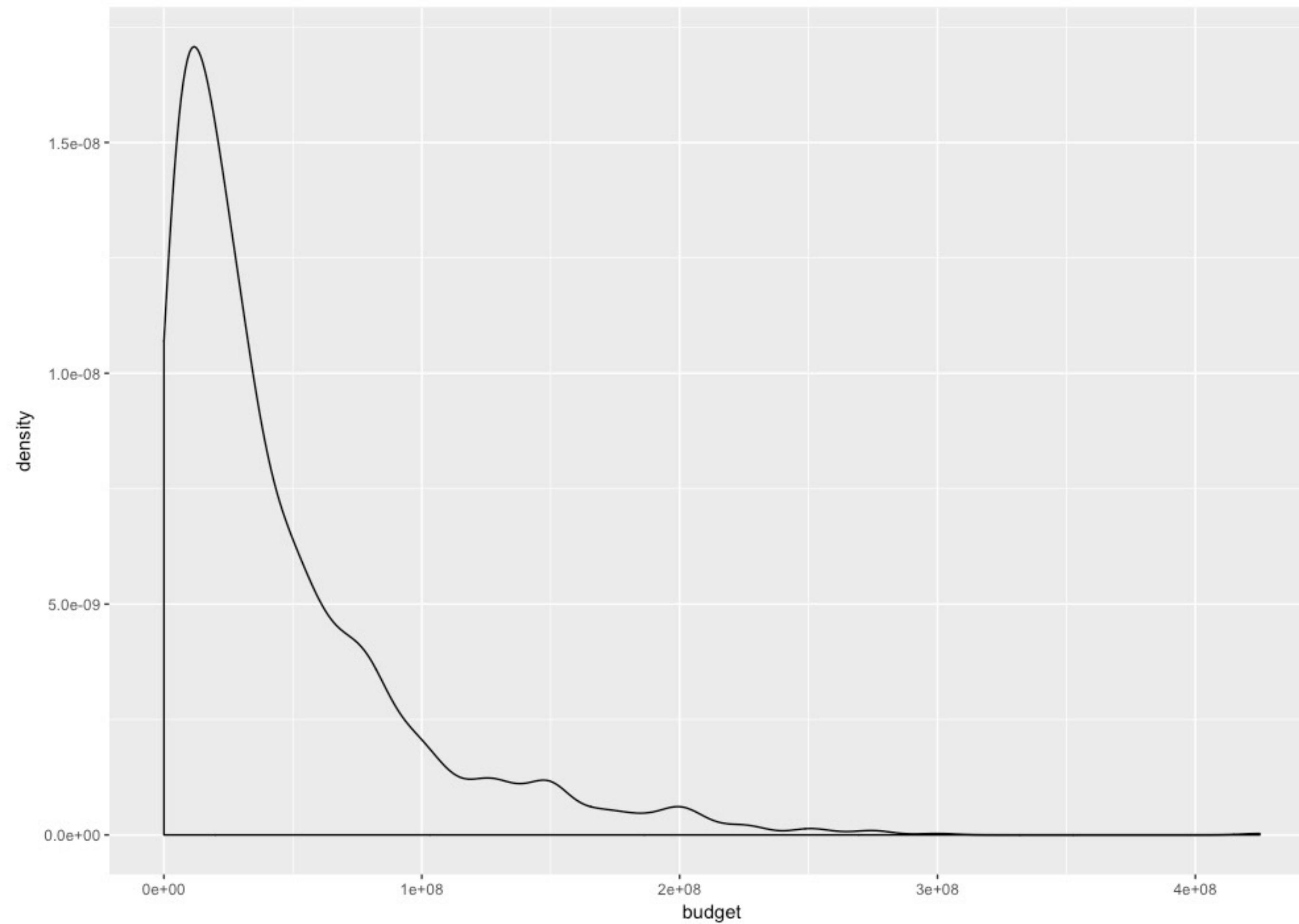
```
ggplot(data = bechdel) +  
  geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```



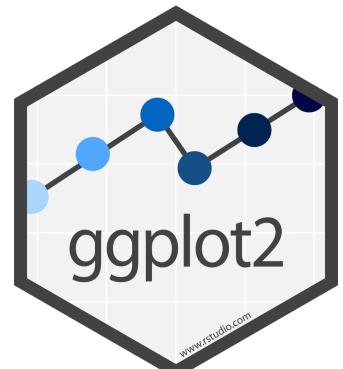
# Your Turn 6

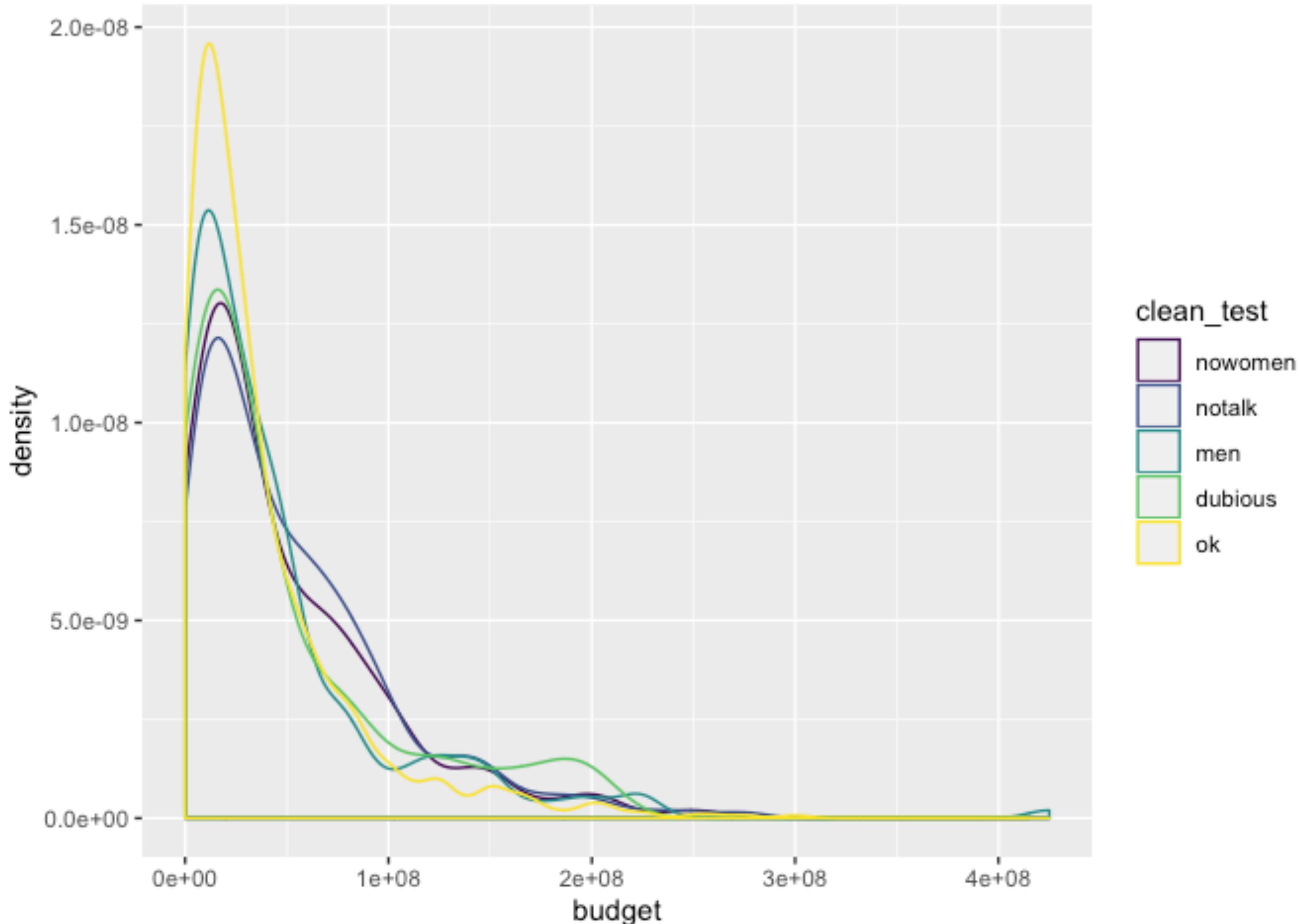
With your partner, make the density plot of **budget** colored by **clean\_test** below. Use the cheatsheet. Try your best guess.



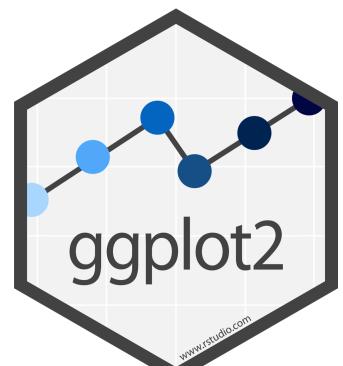


```
ggplot(data = bechdel) +  
  geom_density(mapping = aes(x = budget))
```



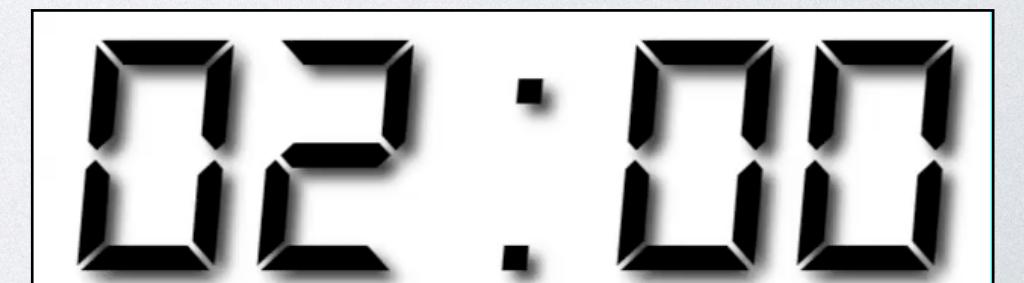
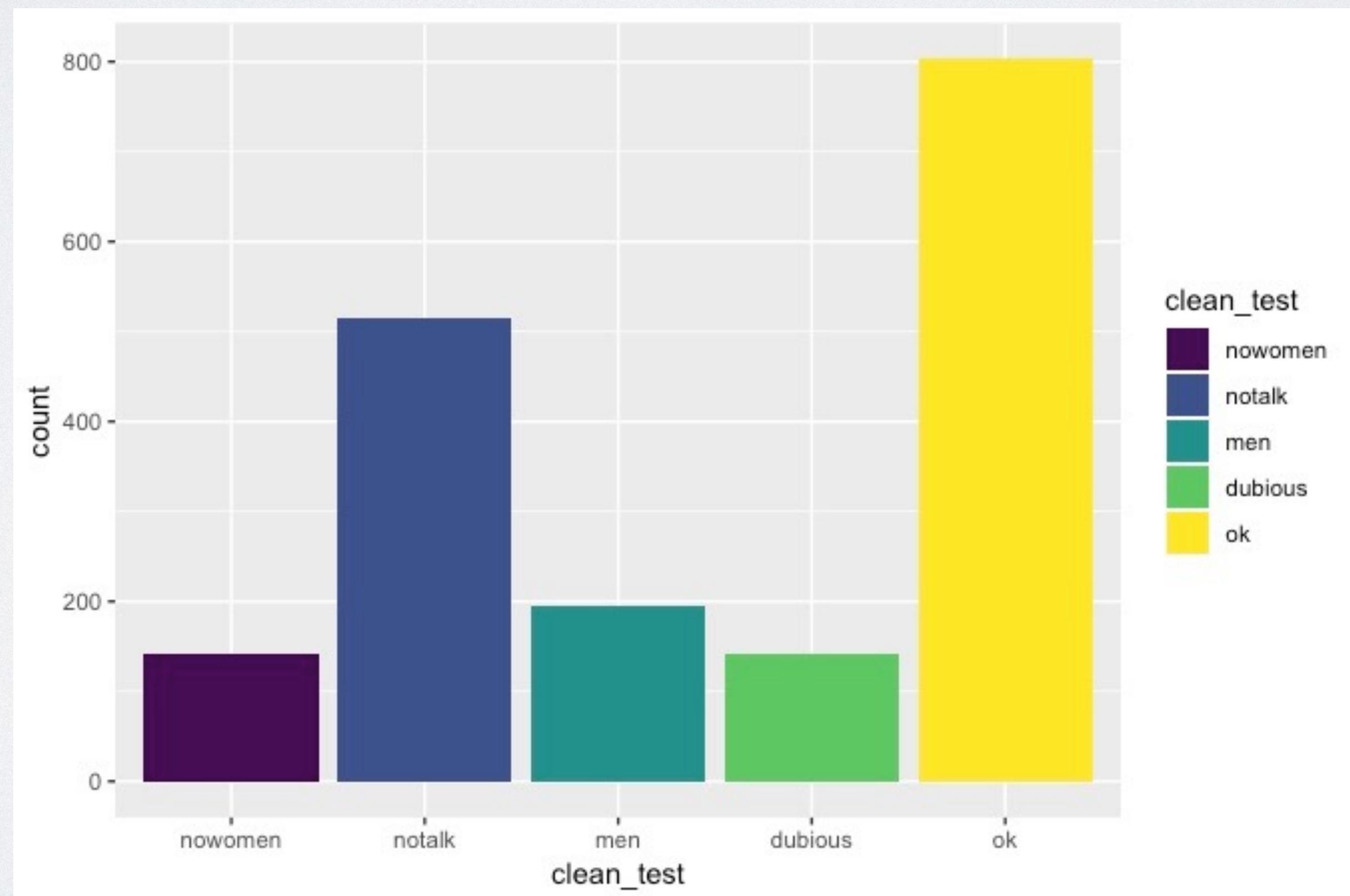


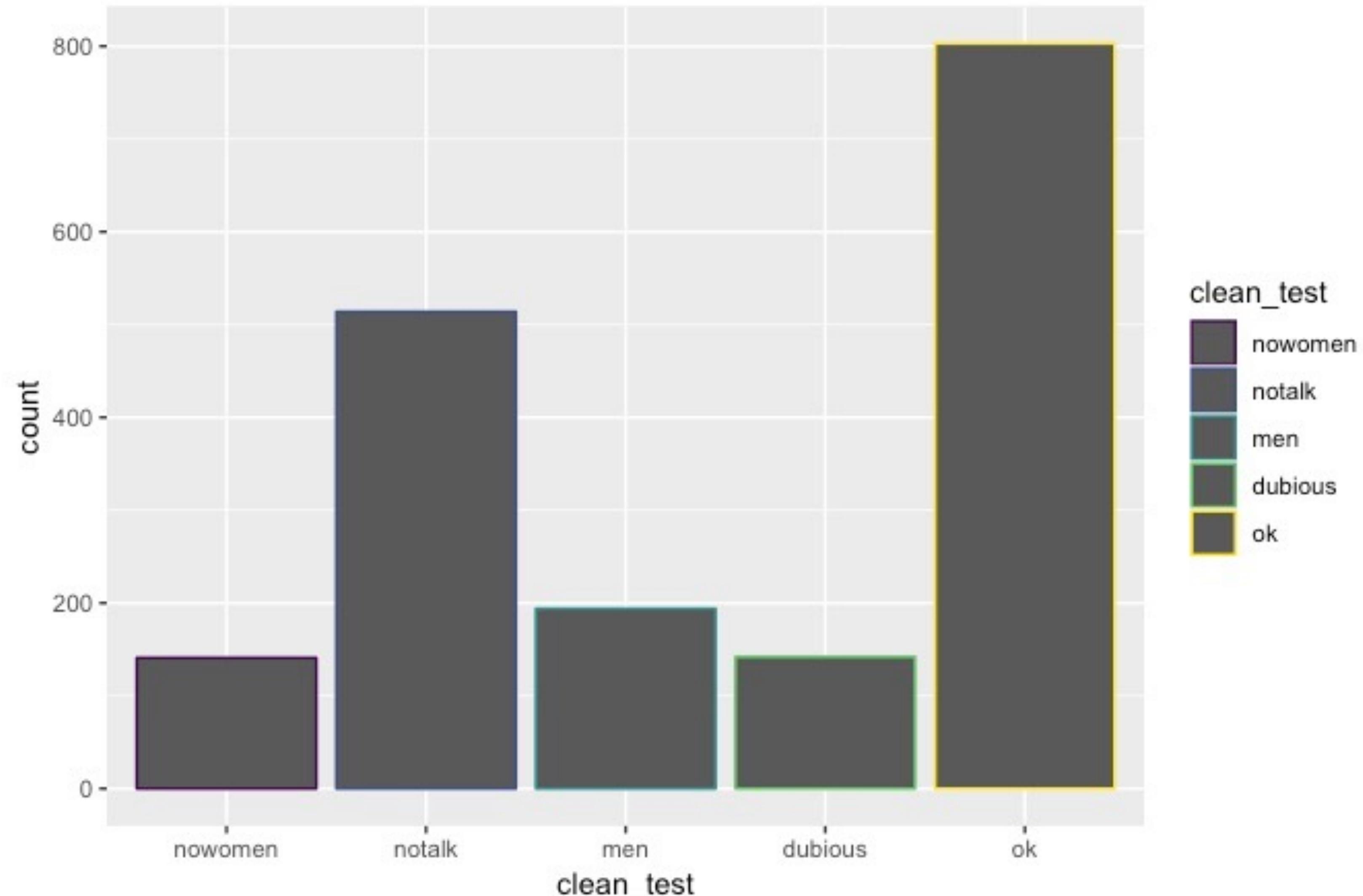
```
ggplot(data = bechdel) +  
  geom_density(mapping = aes(x = budget, color=clean_test))
```



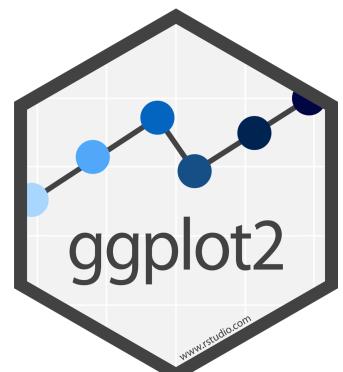
# Your Turn 7

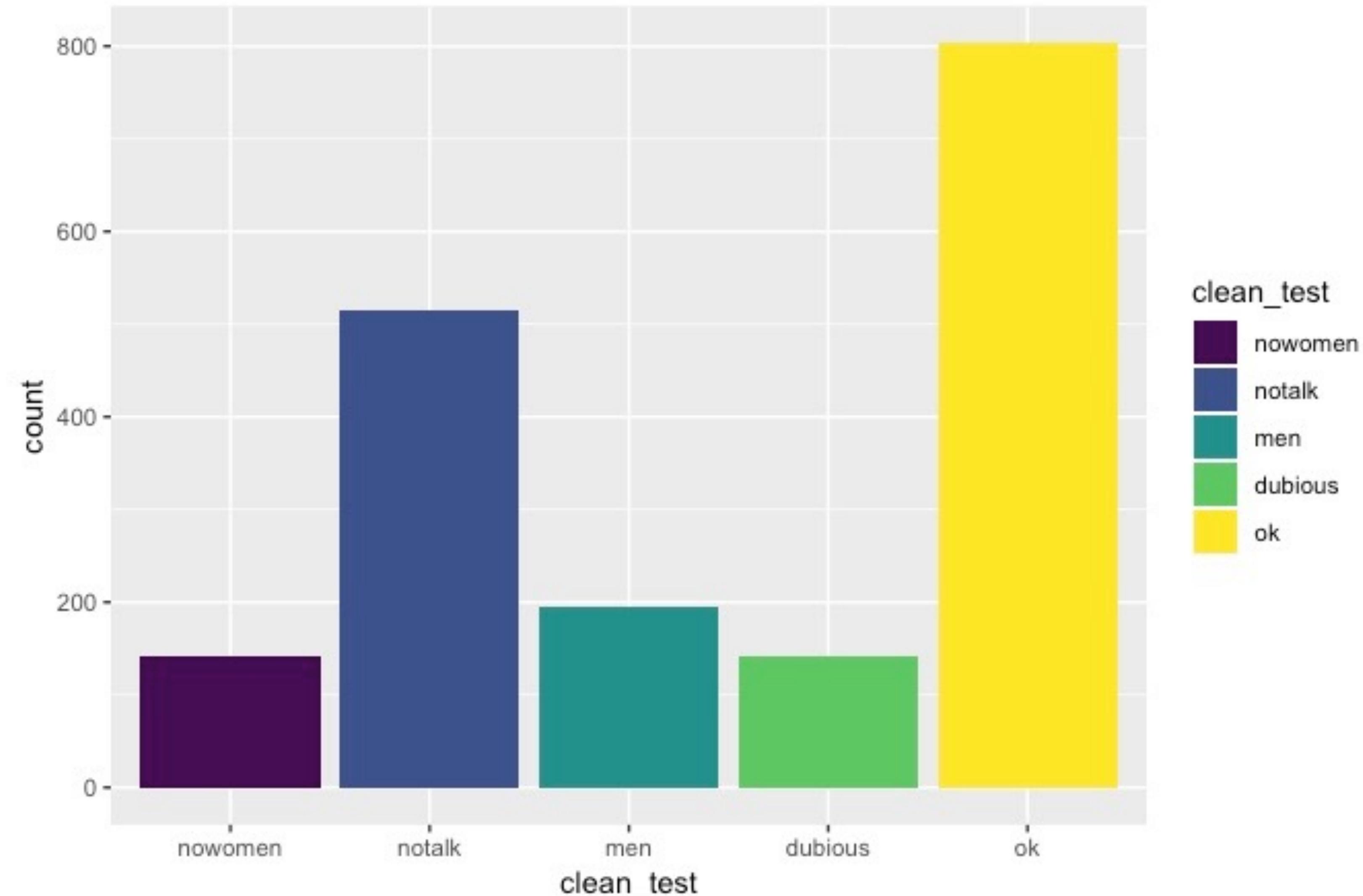
With your partner, make the bar chart of **clean\_test** colored by **clean\_test** below. Use the cheatsheet. Try your best guess.



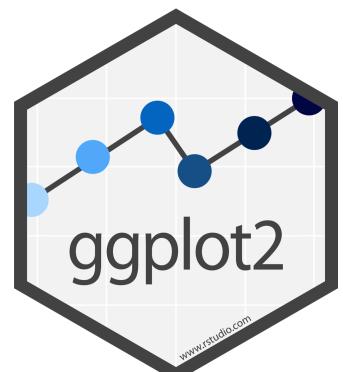


```
ggplot(data=bechdel) +  
  geom_bar(aes(x=clean_test, color=clean_test))
```





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

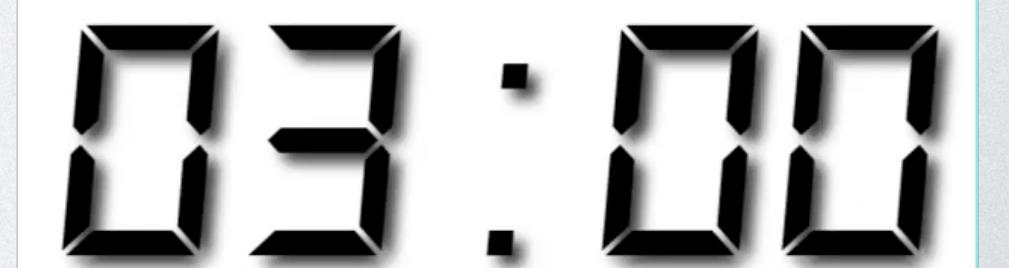


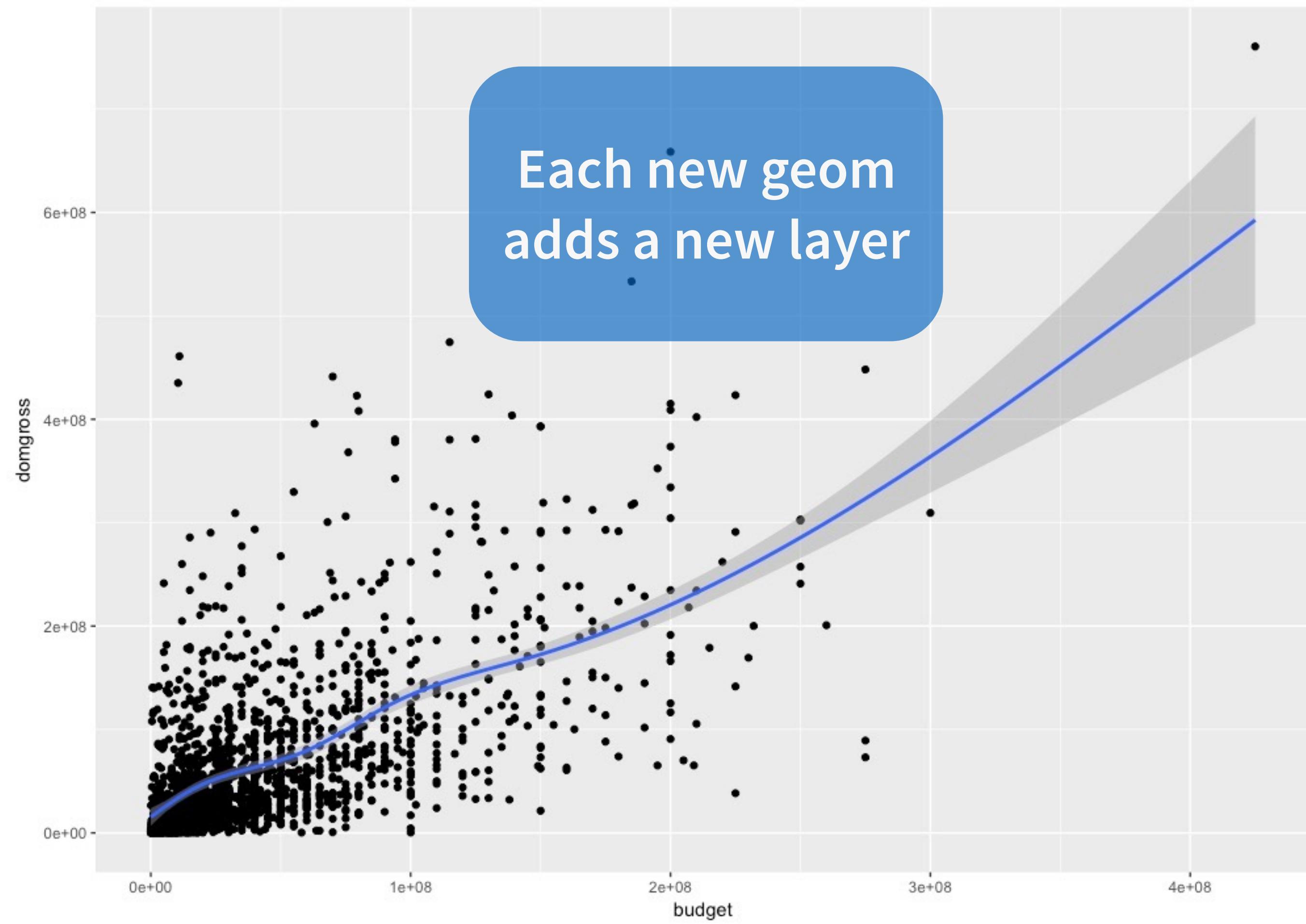
# Your Turn 8

With a partner, predict what this code will do.

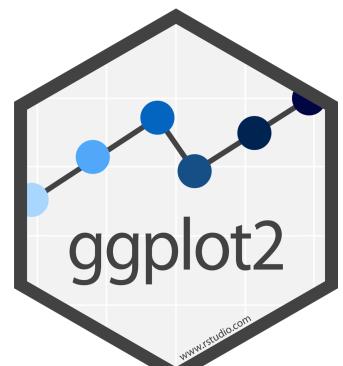
Then run it.

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



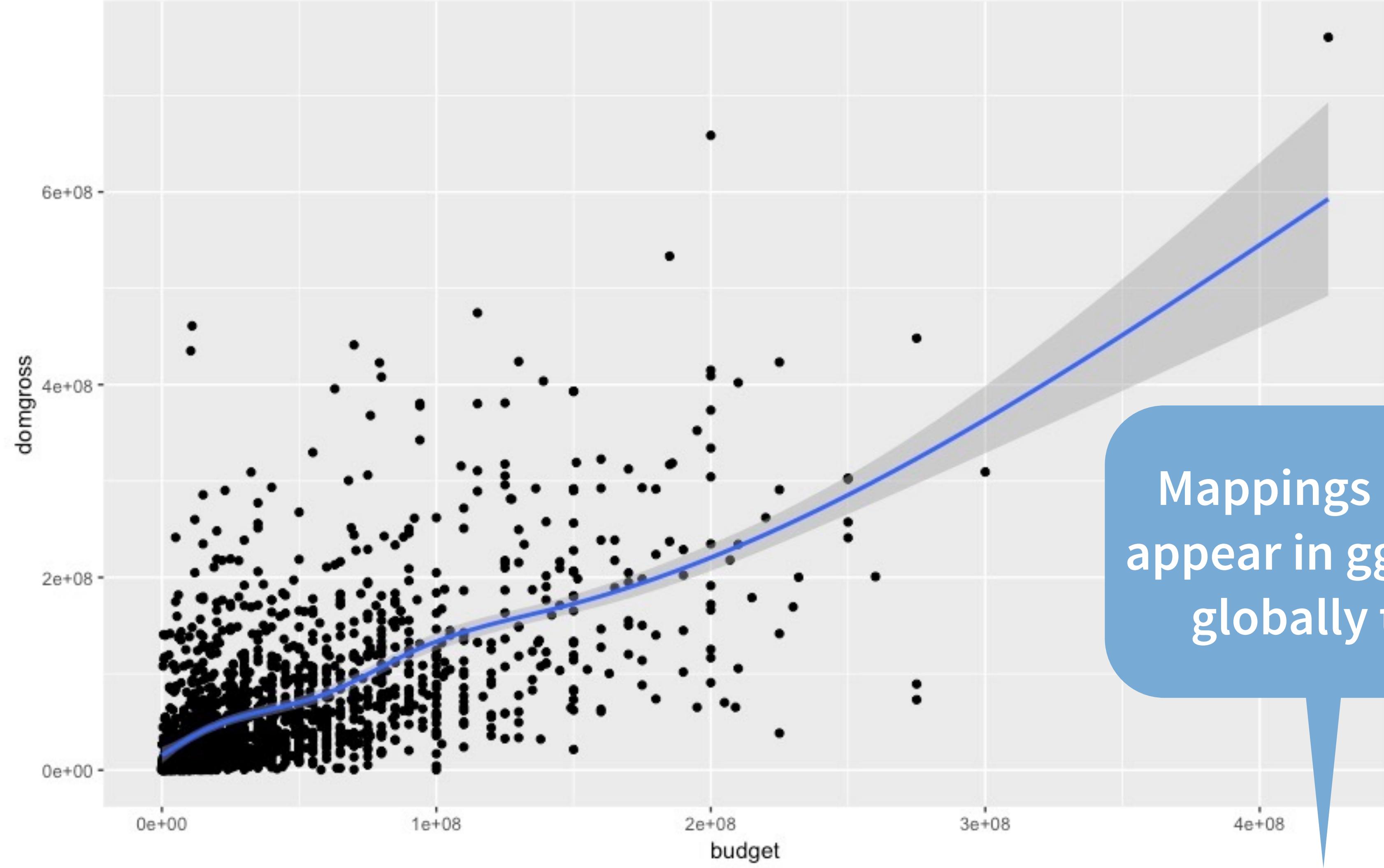


```
ggplot(data = bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross)) +  
  geom_smooth(mapping = aes(x = budget, y = domgross))
```

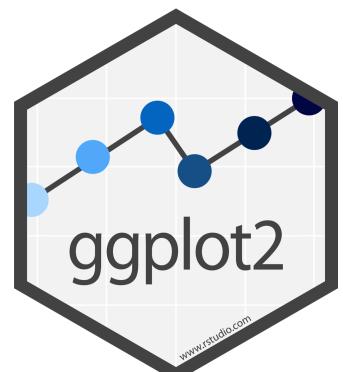


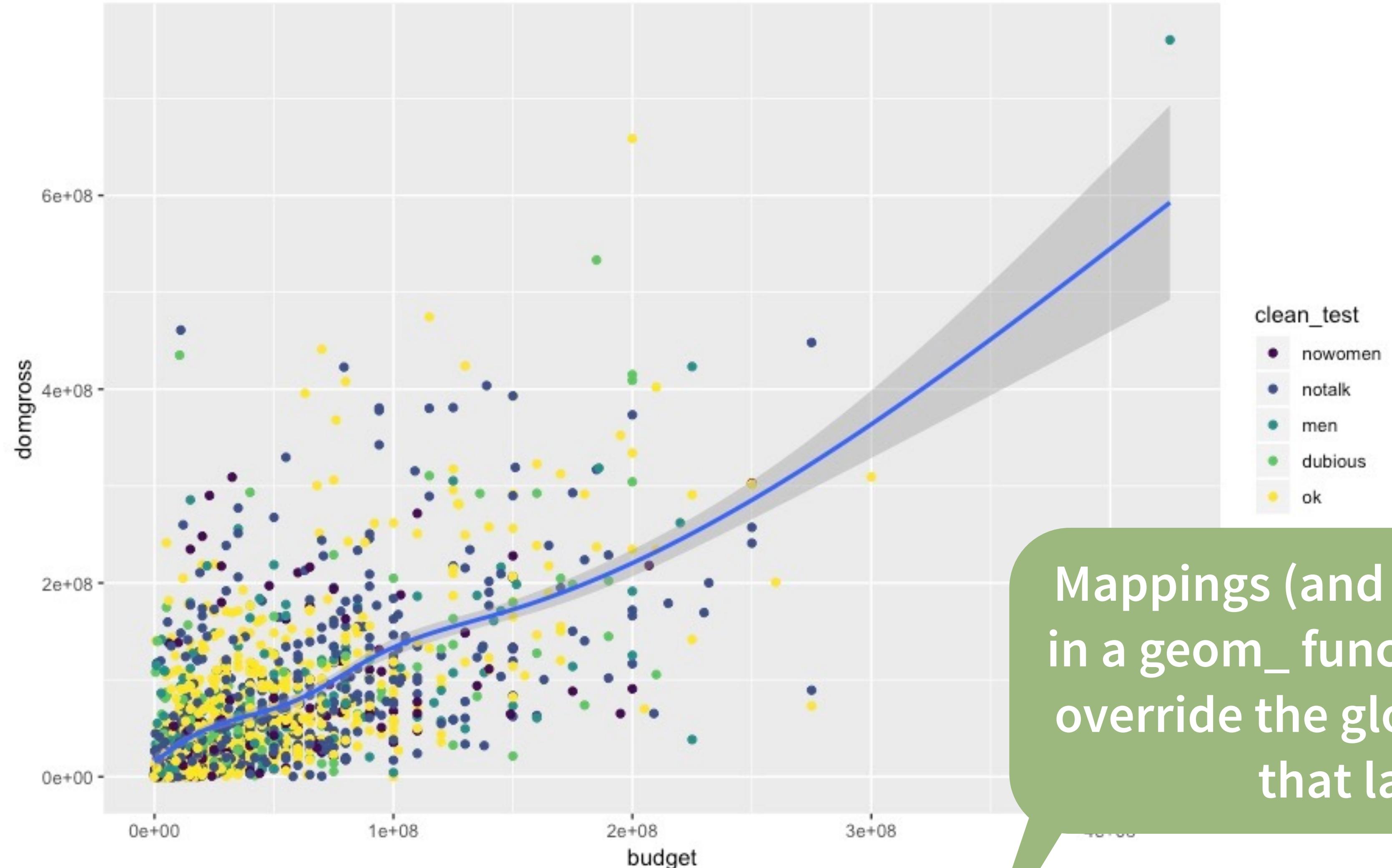
# global vs. local

R

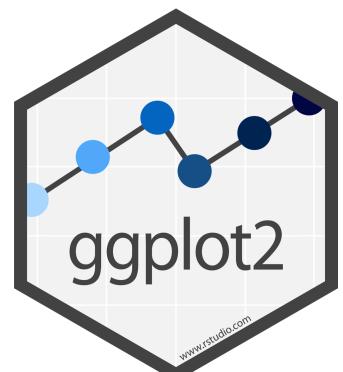


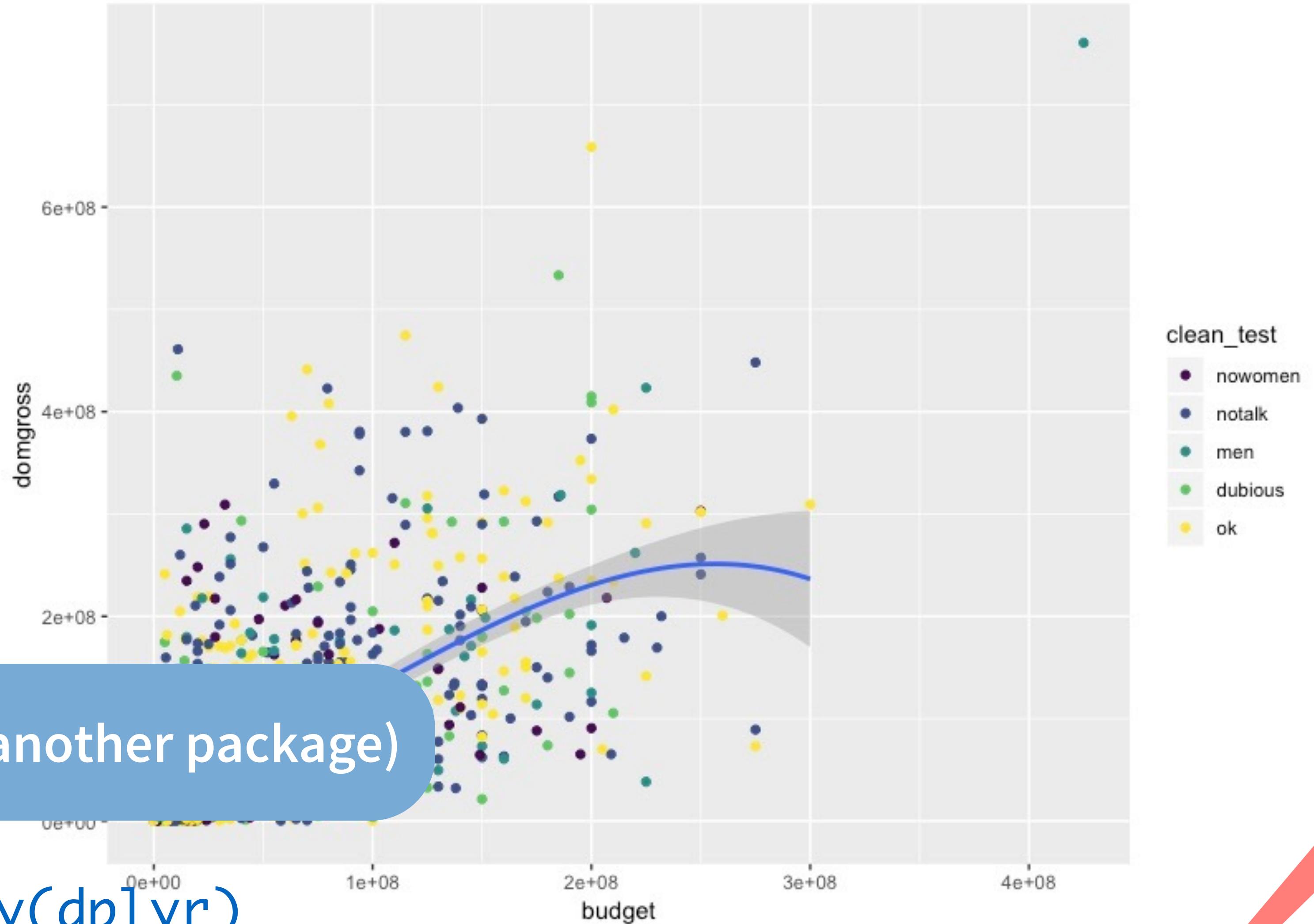
```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +  
  geom_point() +  
  geom_smooth()
```





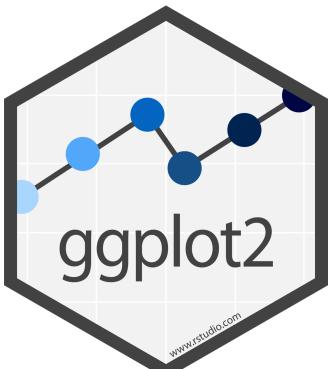
```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +  
  geom_point(mapping = aes(color = clean_test)) +  
  geom_smooth()
```





```
library(dplyr)
```

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



# Saving graphs

R

# “knitting” an R Markdown document

The easiest way to save all your work (including graphs) is to include it in an R Markdown document. While you're working, you can “knit” your document by clicking “Knit.”

A screenshot of the RStudio interface. On the left, the code editor shows an R Markdown file named "01-Visualize.Rmd". The "Knit" button in the toolbar is circled in red. The main pane displays the R Markdown code:

```
1 ---  
2 title: "Visualization"  
3 output: html_document  
4 ---  
5  
6 ## Setup  
7  
8 The first chunk in an R Notebook is usually titled "setup," and by convention  
9 includes the R packages you want to load. Remember, in order to use an R package  
10 you have to run some `library()` code every session. Execute these lines of code to  
11 load the packages.  
12  
13 ```{r setup}  
14 library(ggplot2)  
15 library(fivethirtyeight)  
16 ```  
17  
18 ## Bechdel test data  
19  
20 We're going to start by playing with data collected by the website FiveThirtyEight  
21 on movies and [the Bechdel test](https://en.wikipedia.org/wiki/Bechdel\_test).  
22  
23 To begin, let's just preview our data. There are a couple ways to do that. One is  
24 just to type the name of the data and execute it like a piece of code.
```

The right side of the interface shows the "Environment" tab with the "bechdel" dataset selected, showing 1794 observations and 15 variables. Below it is a "Plots" tab displaying a scatter plot of movie budget versus 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 gray shaded confidence interval is overlaid on a dense cloud of black data points.

# HTML preview

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

The screenshot shows the RStudio interface with an R Notebook on the left and its HTML preview on the right.

**Left Panel (R Notebook):**

- File menu: File, New, Open, Save, Save As, Import, Export, Print, Go to file/function.
- Addins menu: Addins.
- Code editor:

```
1 ---  
2 title: "Visualization"  
3 output: html_document  
4 ---  
5  
6 ## Setup  
7  
8 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.  
9  
10 `r setup}  
11 library(ggplot2)  
12 library(fivethirtyeight)  
13 `r  
14  
15 ## Bechdel test data  
16  
17 We're going to start by playing with data collected by the website FiveThirtyEight on movies and [the Bechdel test](https://en.wikipedia.org/w/index.php?title=Bechdel_test&oldid=91111111).  
18  
19 To begin, let's just preview our data. There's just one step: just type the name of the data and execute it.
```

- Console: 108:1 C Chunk 11 ↴

**Right Panel (HTML Preview):**

- Title: 01-Visualize.html | Open in Browser | Find
- Header: Visualization
- Section: Setup
- Text: 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)
```

- Section: Bechdel test data
- Text: We're going to start by playing with data collected by the website FiveThirtyEight on movies and [the Bechdel test](https://en.wikipedia.org/w/index.php?title=Bechdel\_test&oldid=91111111). To begin, let's just preview our data. There's just one step: just type the name of the data and execute it like a piece of code.

```
bechdel
```

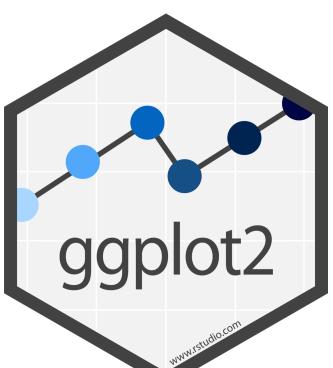
## # A tibble: 1,794 x 15
## #   year    imdb    title    test    clean_test    binary    budget    domgross    intgross    code
## #   <int> <chr> <chr> <chr> <ord> <chr> <int> <dbl> <dbl> <chr>
## 1 2013 tt17... 21 &... nota... notalk FAIL 1.30e7 25682380 4.22e7 2013...
## 2 2012 tt13... Dred... ok-d... ok PASS 4.50e7 13414714 4.09e7 2012...
## 3 2013 tt20... 12 Y... nota... notalk FAIL 2.00e7 53107035 1.59e8 2013...
## 4 2013 tt12... 2 Gu... nota... notalk FAIL 6.10e7 75612460 1.32e8 2013...
## 5 2013 tt04... 42 men men FAIL 4.00e7 95020213 9.50e7 2013...
## 6 2013 tt13... 47 R... men men FAIL 2.25e8 38362475 1.46e8 2013...
## 7 2013 tt16... A Go... nota... notalk FAIL 9.20e7 67349198 3.04e8 2013...
## 8 2013 tt21... Abou... ok-d... ok PASS 1.20e7 15323921 8.73e7 2013...
## 9 2013 tt18... Admi... ok ok PASS 1.30e7 18007317 1.80e7 2013...
## 10 2013 tt18... Afte... nota... notalk FAIL 1.30e8 60522097 2.44e8 2013...
## # ... with 1,784 more rows, and 5 more variables: budget_2013 <int>,
## #   domgross_2013 <dbl>, intgross_2013 <dbl>, period_code <int>,
## #   decade_code <int>

- Text: Notice that you can page through to see more of the dataset.
- Text: 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.
- Text: (An aside— `view` is a special function. Since it makes something happen in the RStudio interface, it doesn't work properly in R Notebooks. Most R functions have names that start with lowercase letters, so the uppercase "V" is there to remind you of its special status.)
- Section: Consider
- Text: What relationship do you expect to see between movie budget (budget) and domestic gross(domgross)?

# Sharing your work

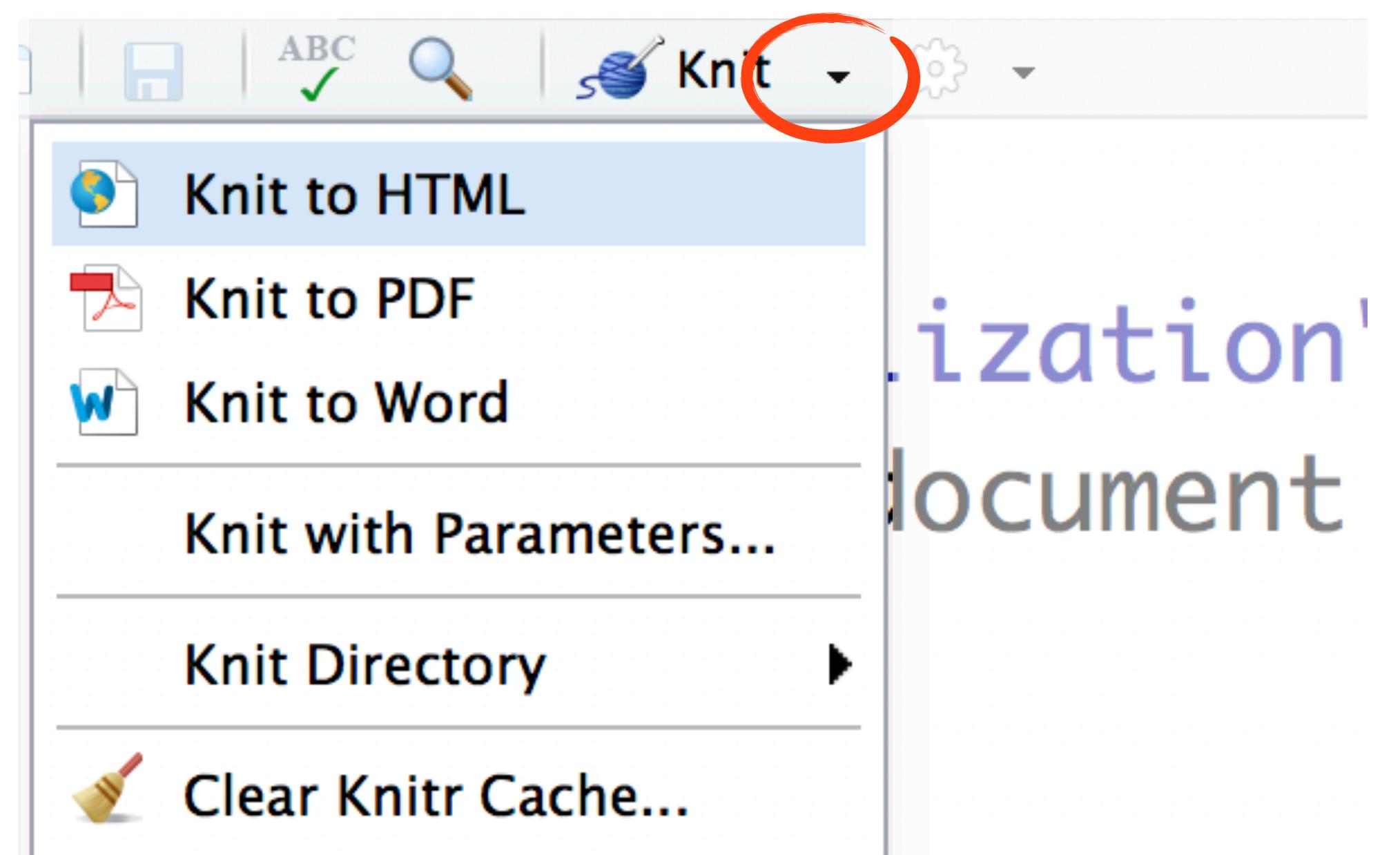
The preview is something that pops up for you to look at as you work, but RStudio automatically creates a file you can share with others when you knit an RMarkdown document. By default, it creates an HTML file, and the file's name corresponds to the name of your RMarkdown document.

You can email this HTML document to anyone you'd like to share your work with!



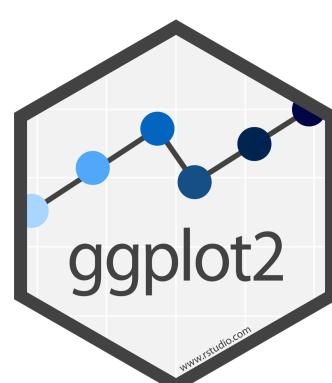
# Other formats

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



`## Setup`

The first chunk in an  
includes the R package



# Your Turn

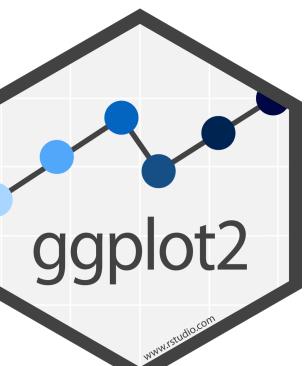
Locate the 01-Visualize.html file in your Files pane. It will be located in your **working directory**



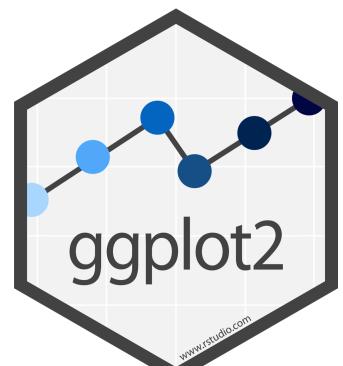
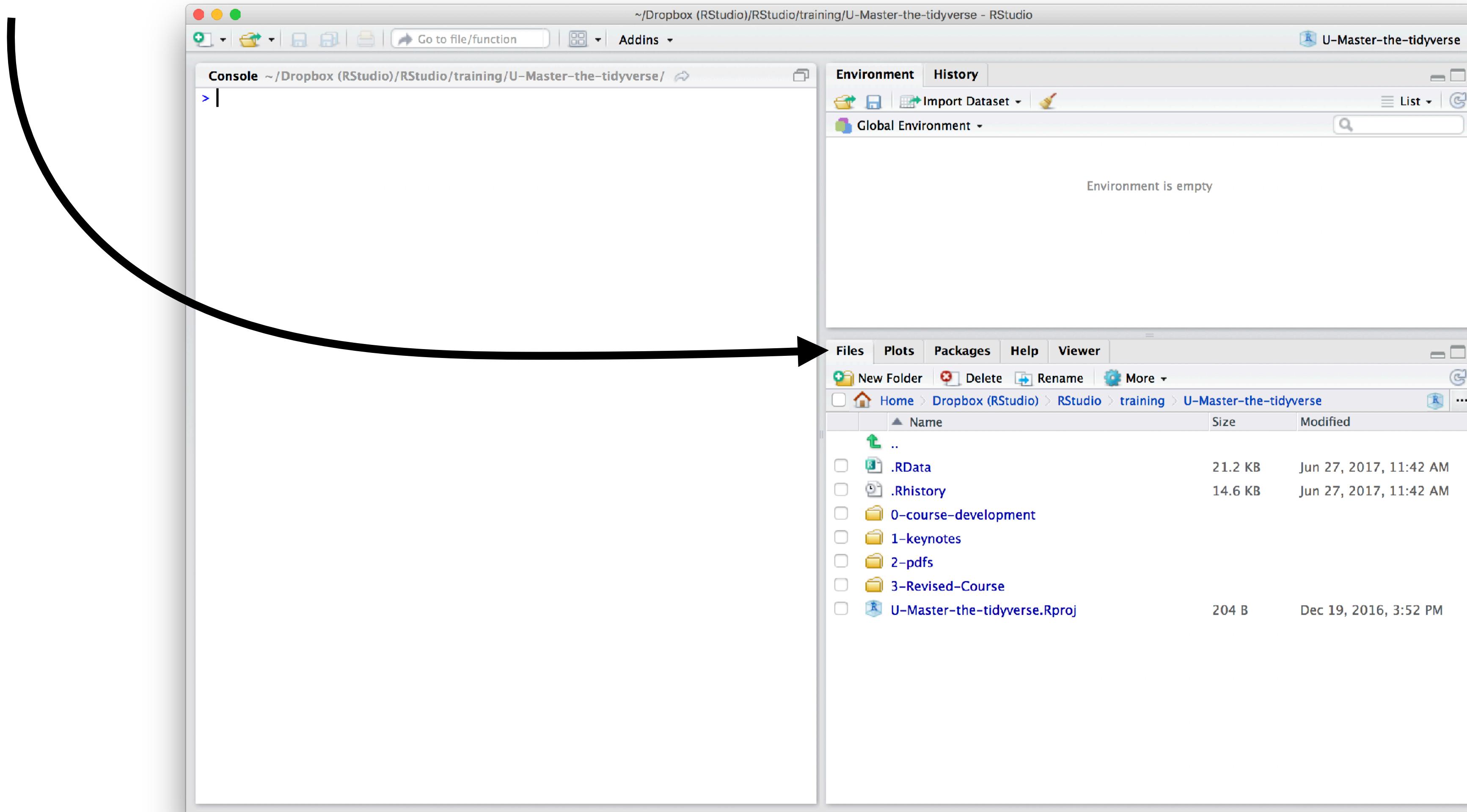
# Working directory

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

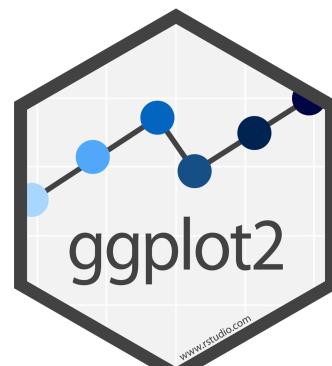
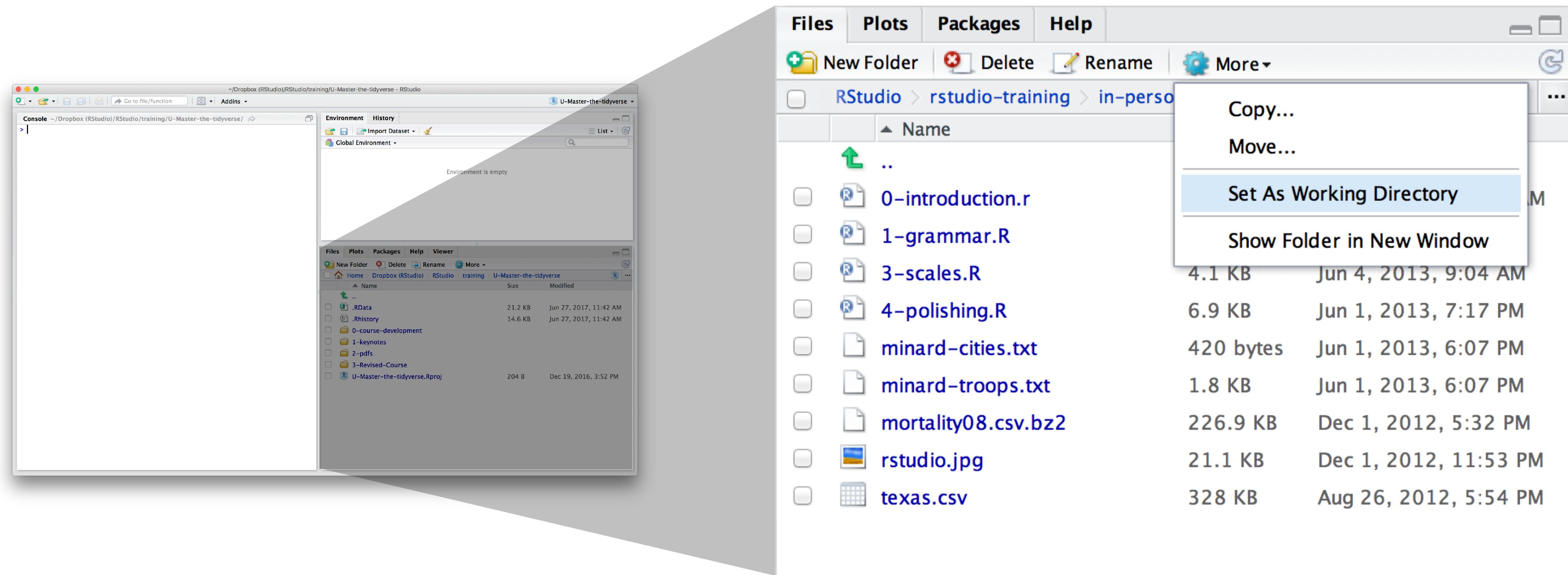


# The files pane of the IDE displays the contents of your working directory



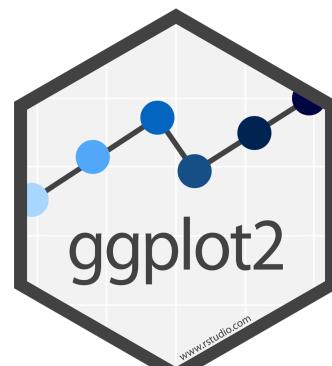
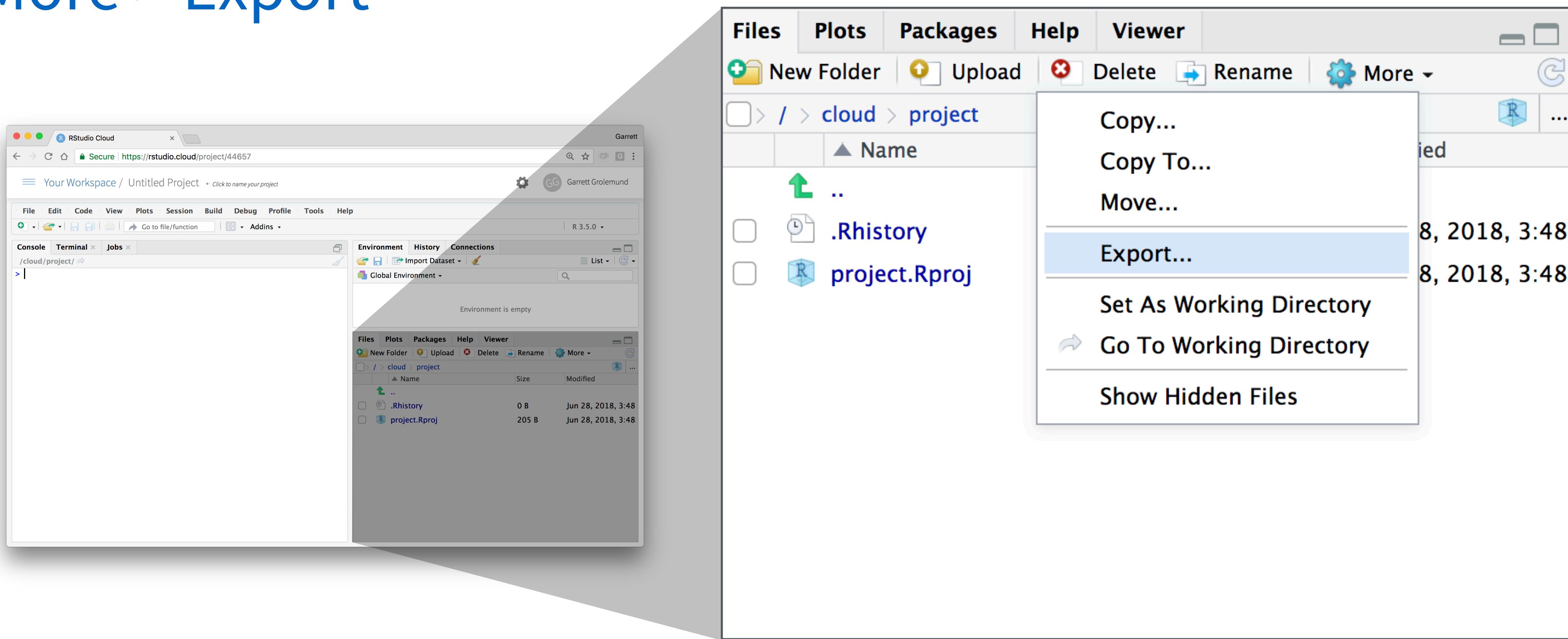
# Changing the Working directory

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



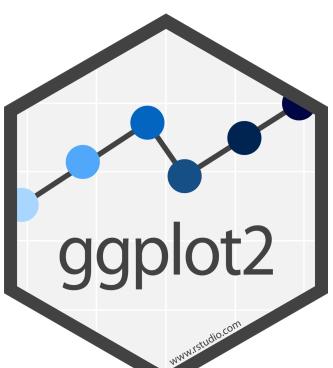
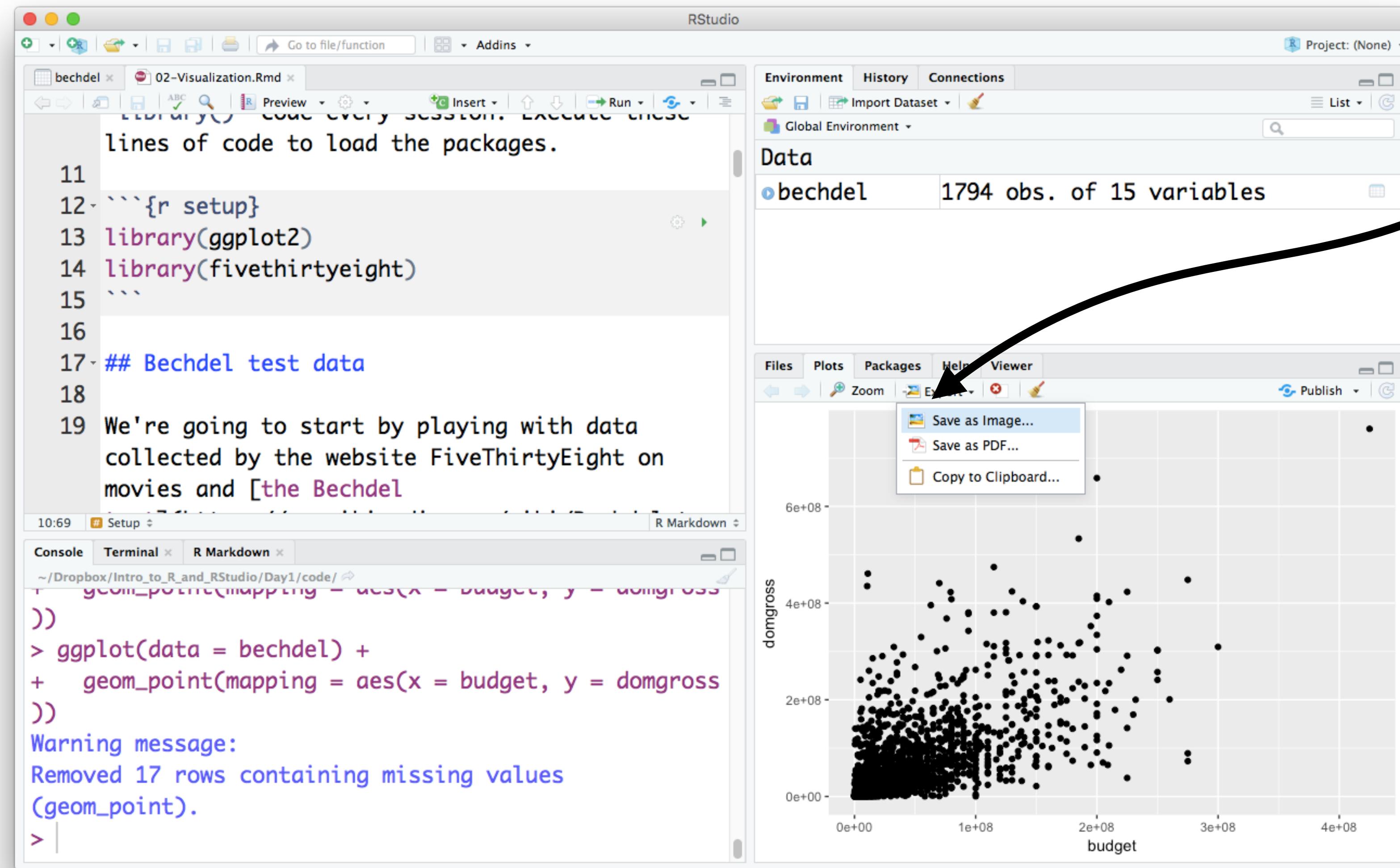
# Download files

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

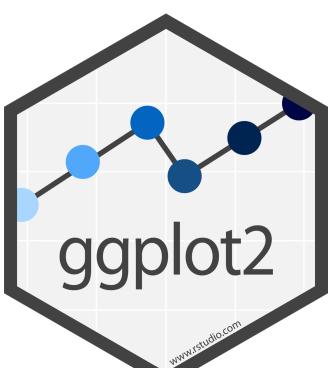
**ggsave()** saves the last plot.

Uses size on screen:

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

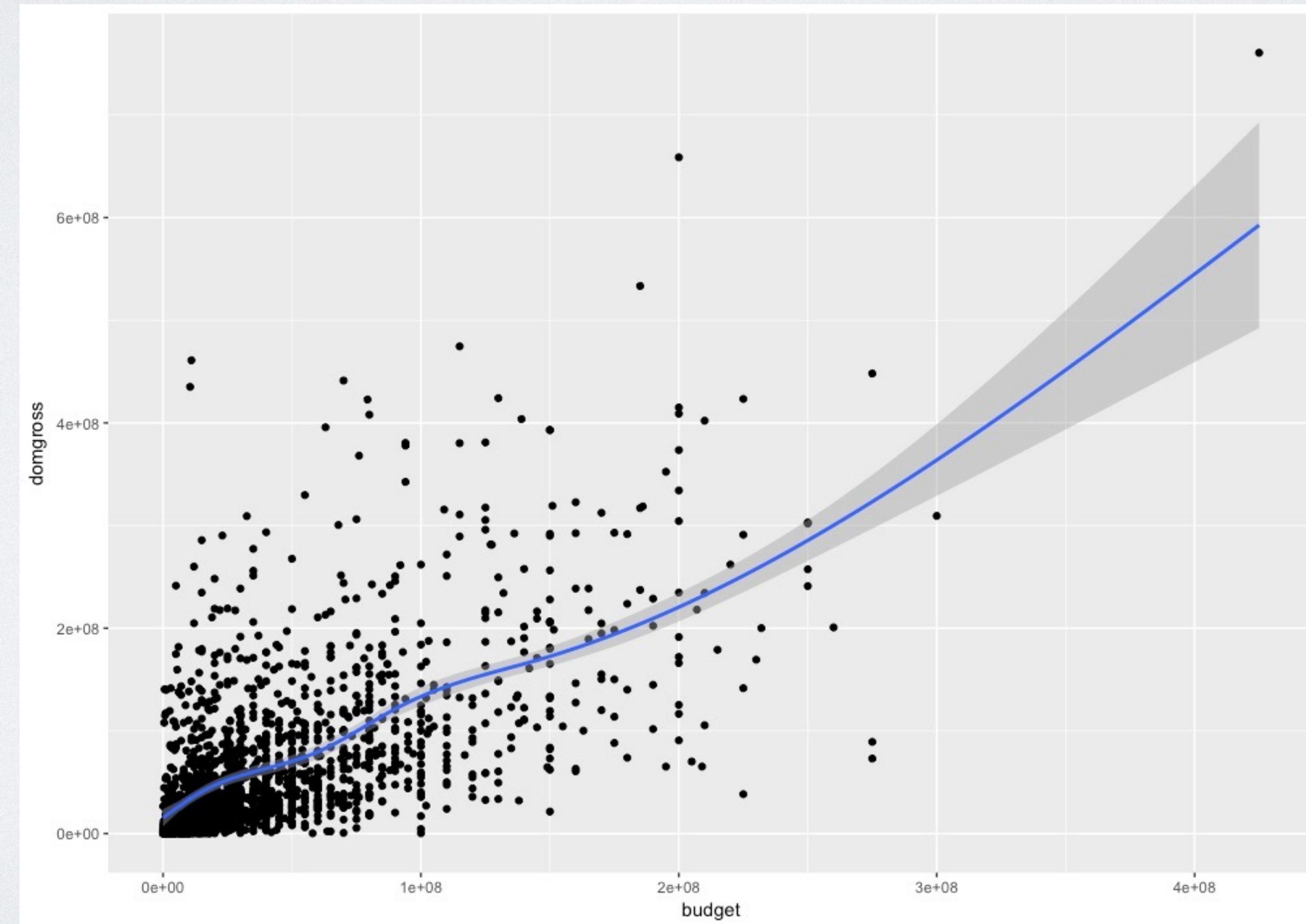
Specify size in inches

```
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).



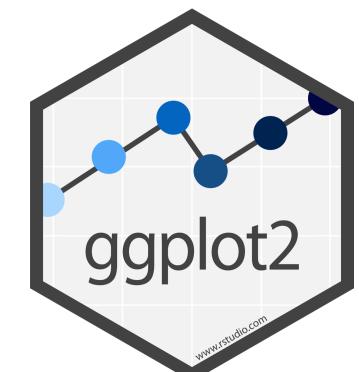
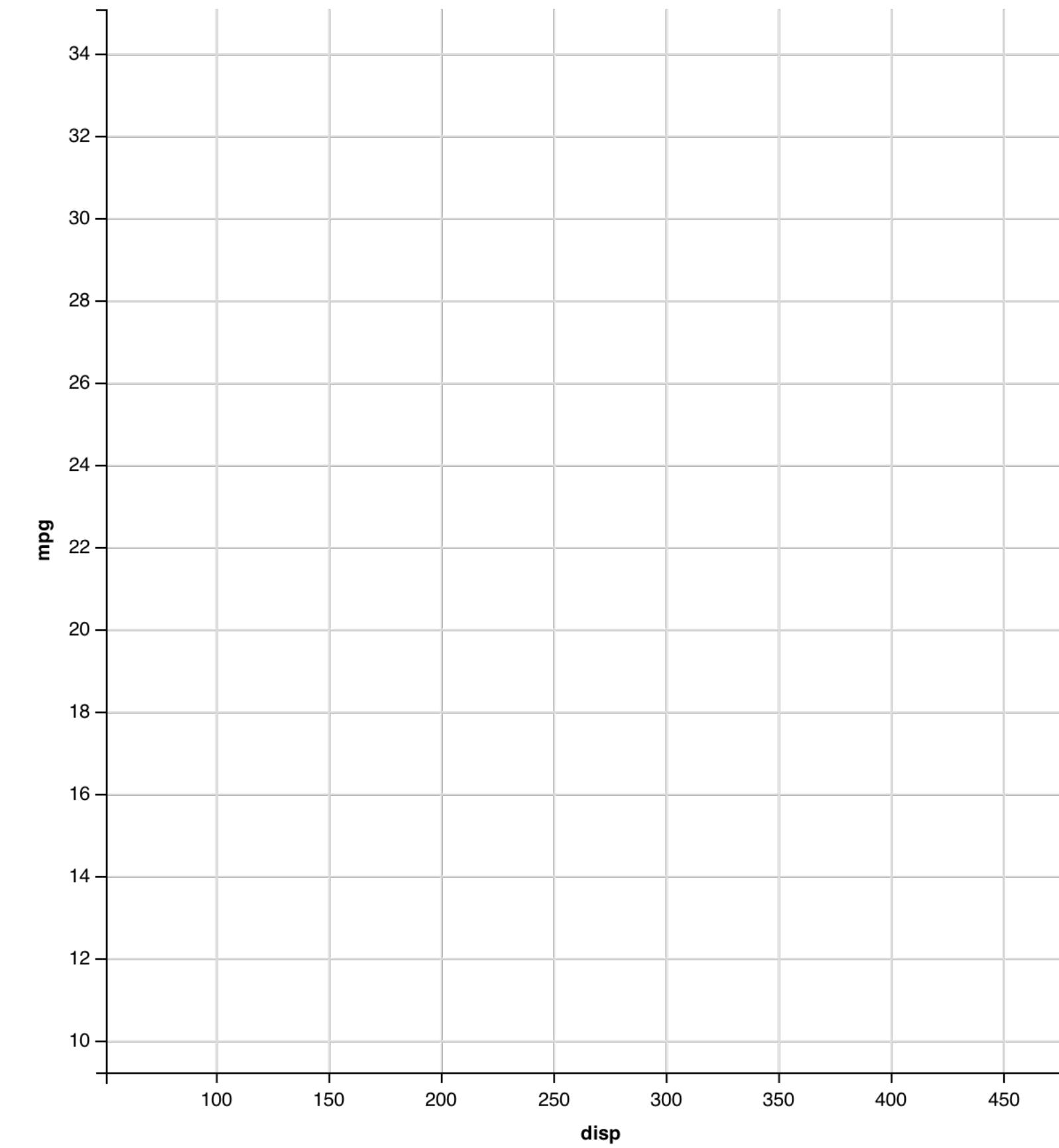
# Grammar of Graphics



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

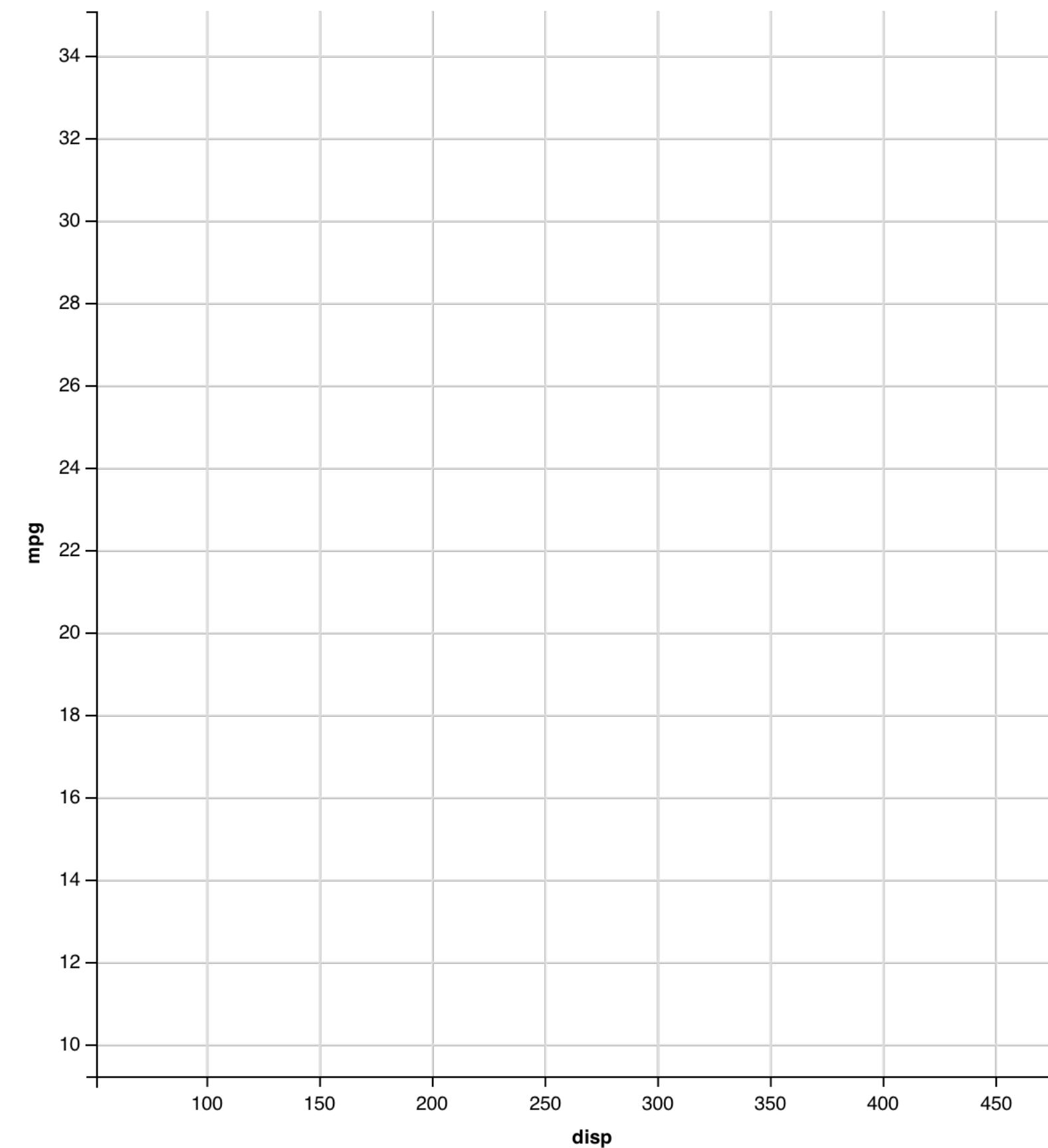
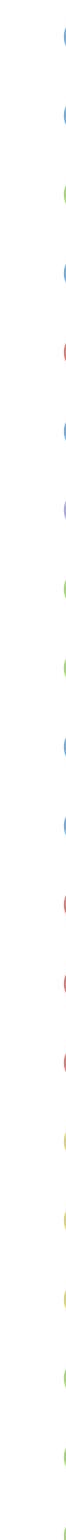
geom



# mappings

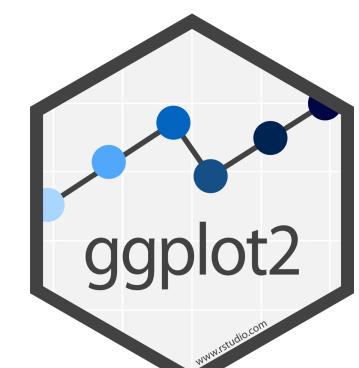
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

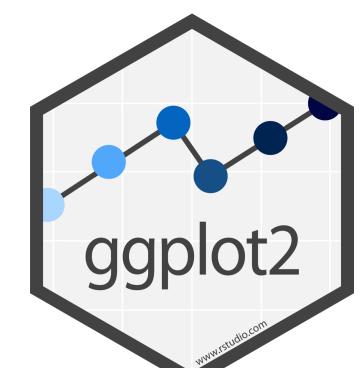
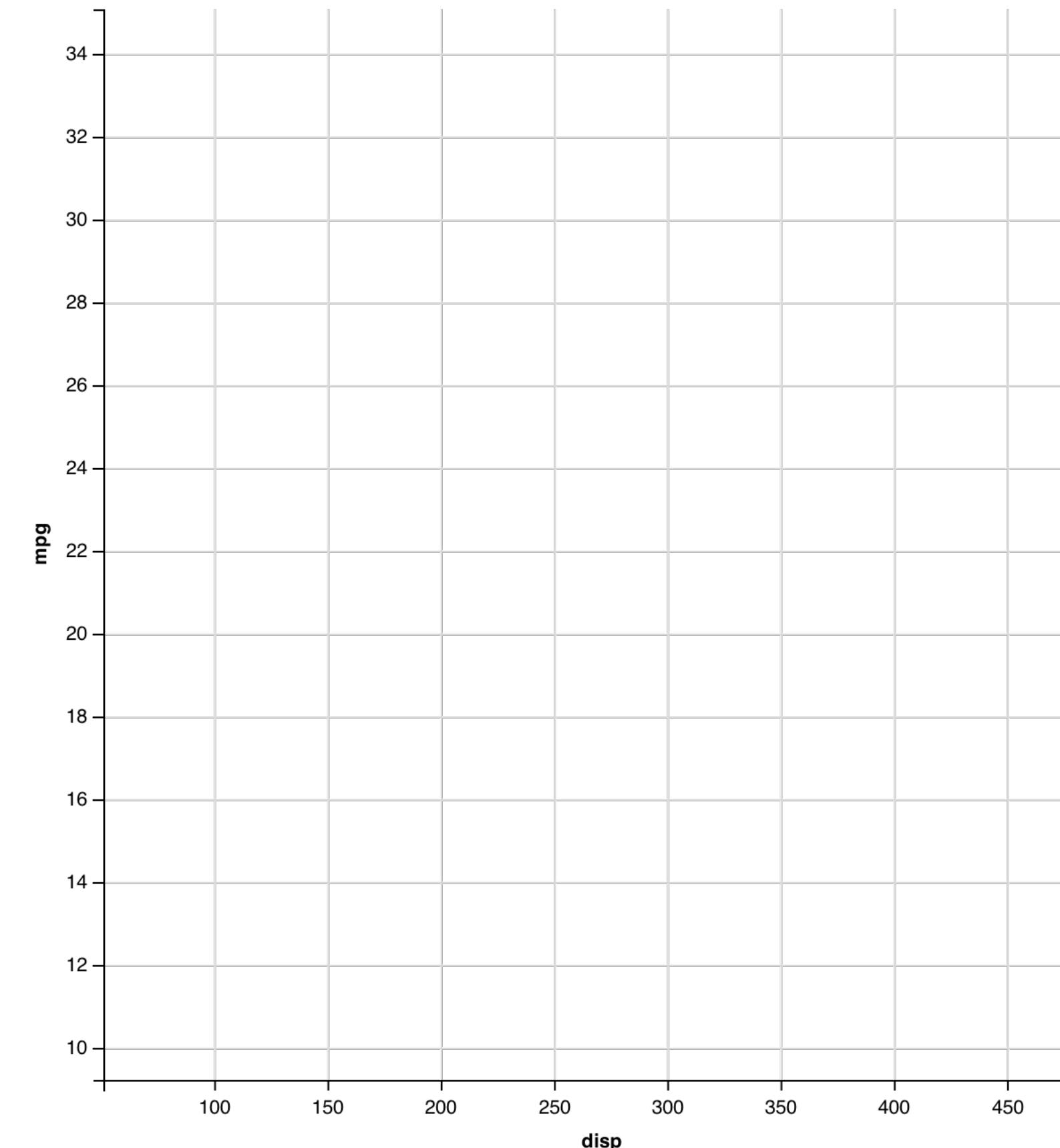


# mappings

	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

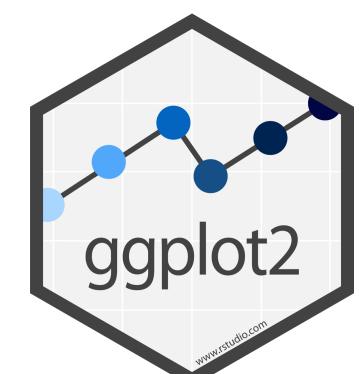
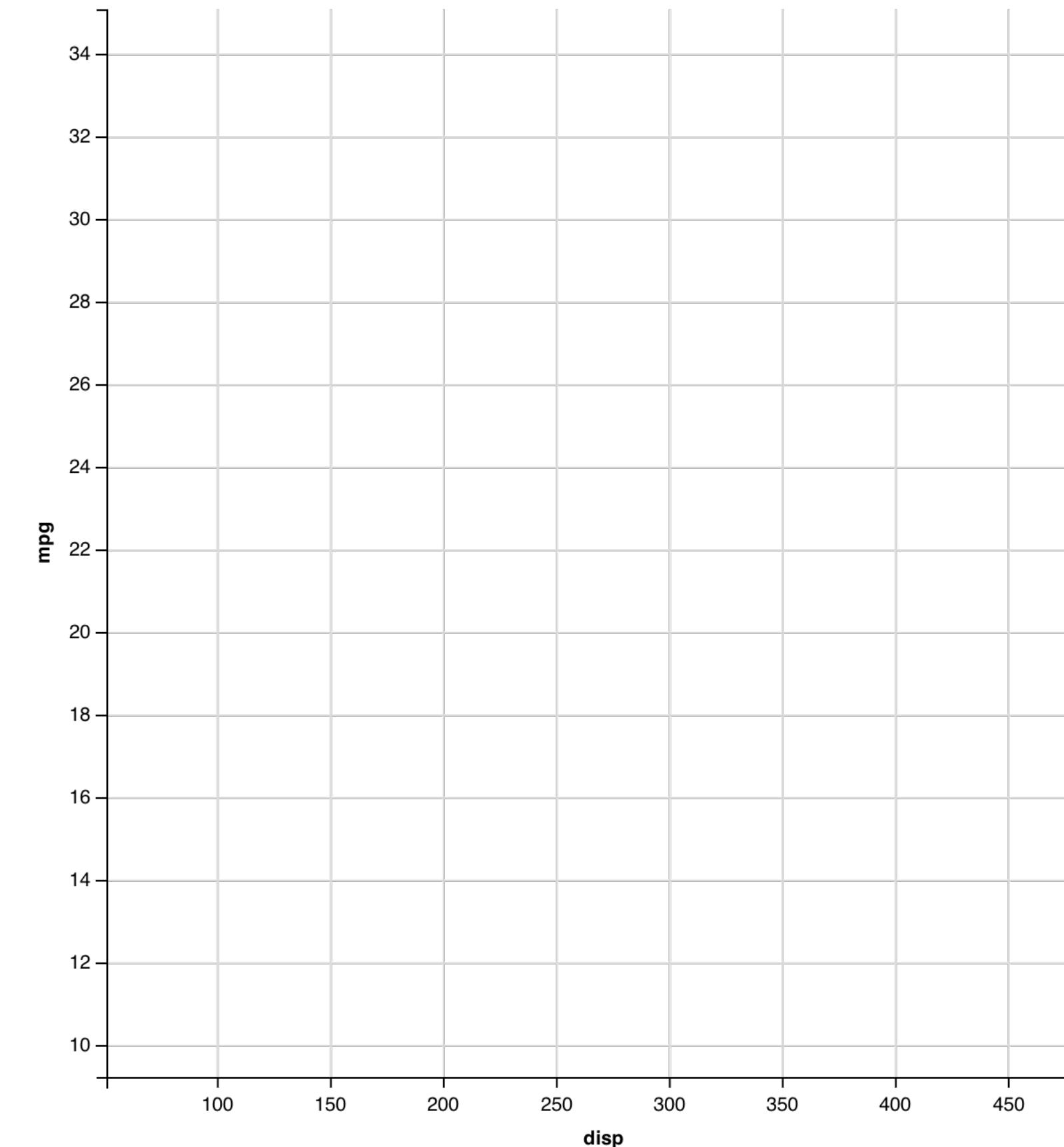


# mappings

	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

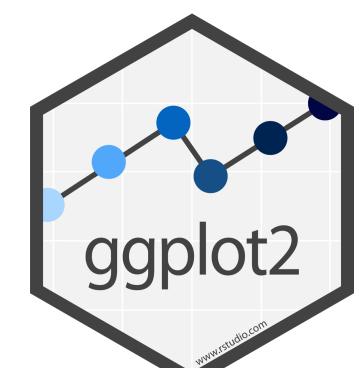
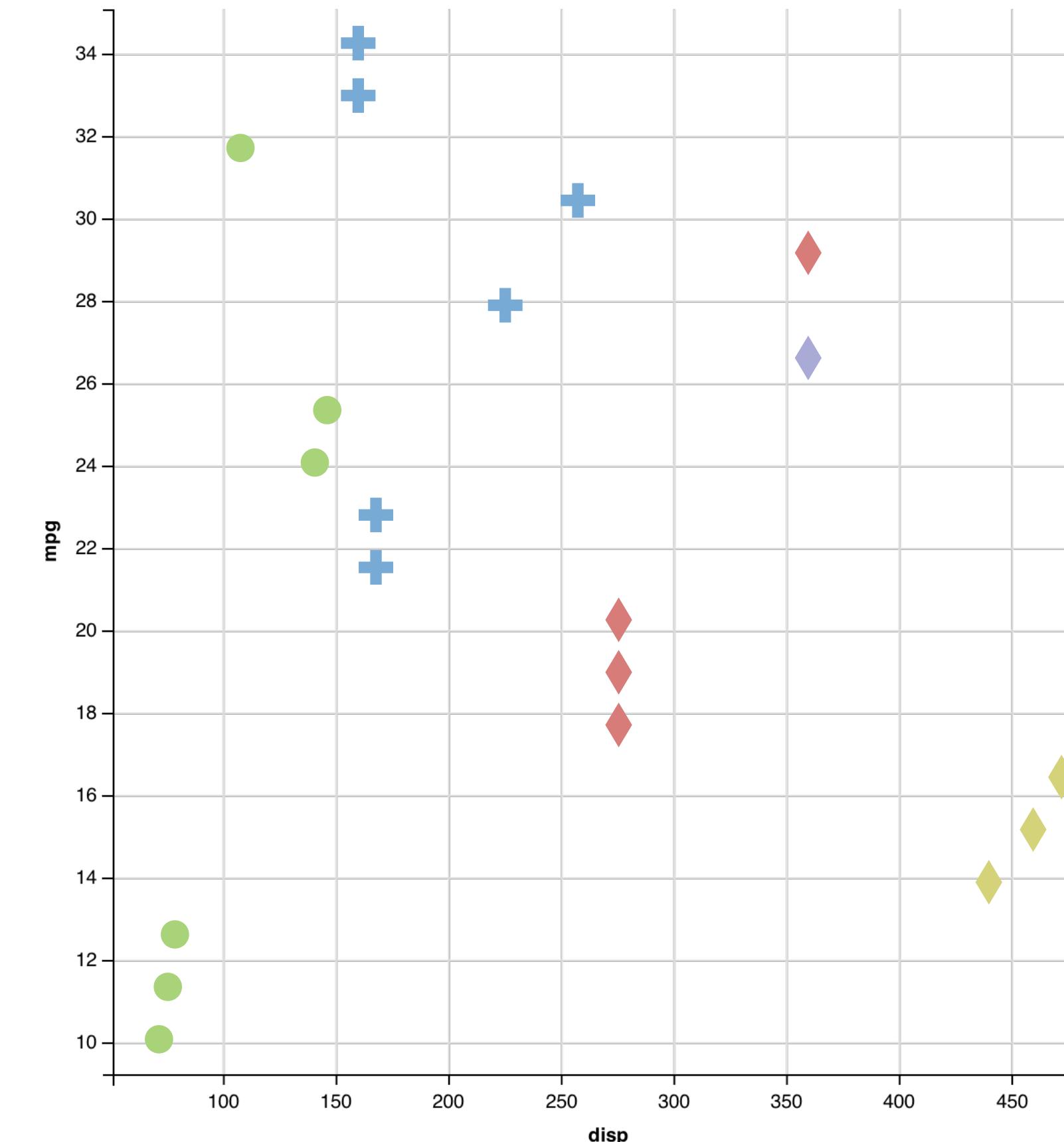


# mappings

	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

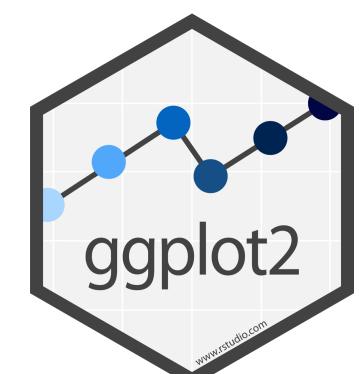
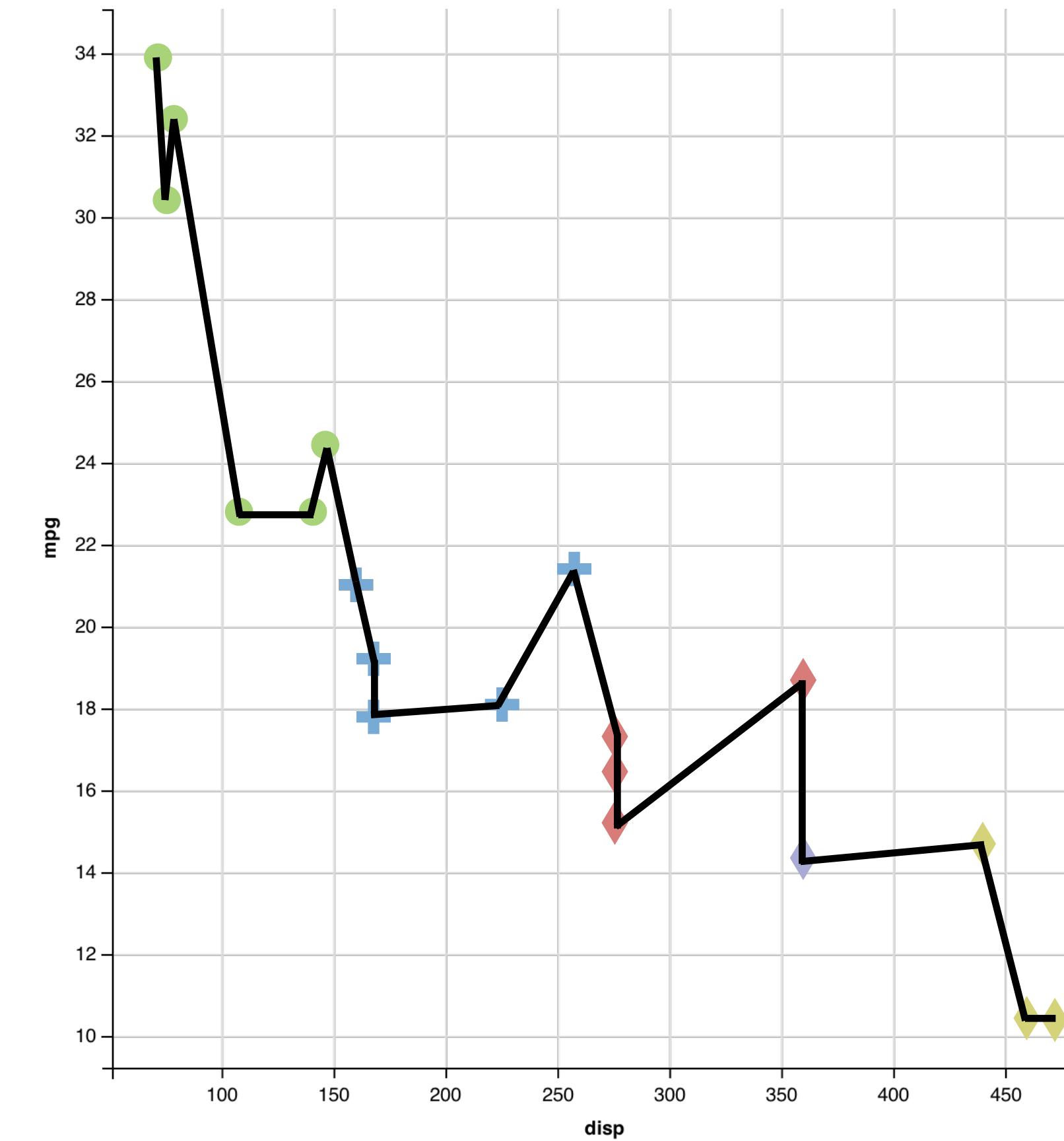


# mappings

	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

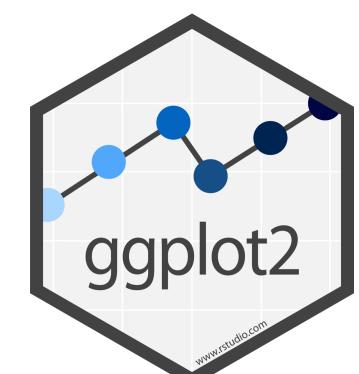
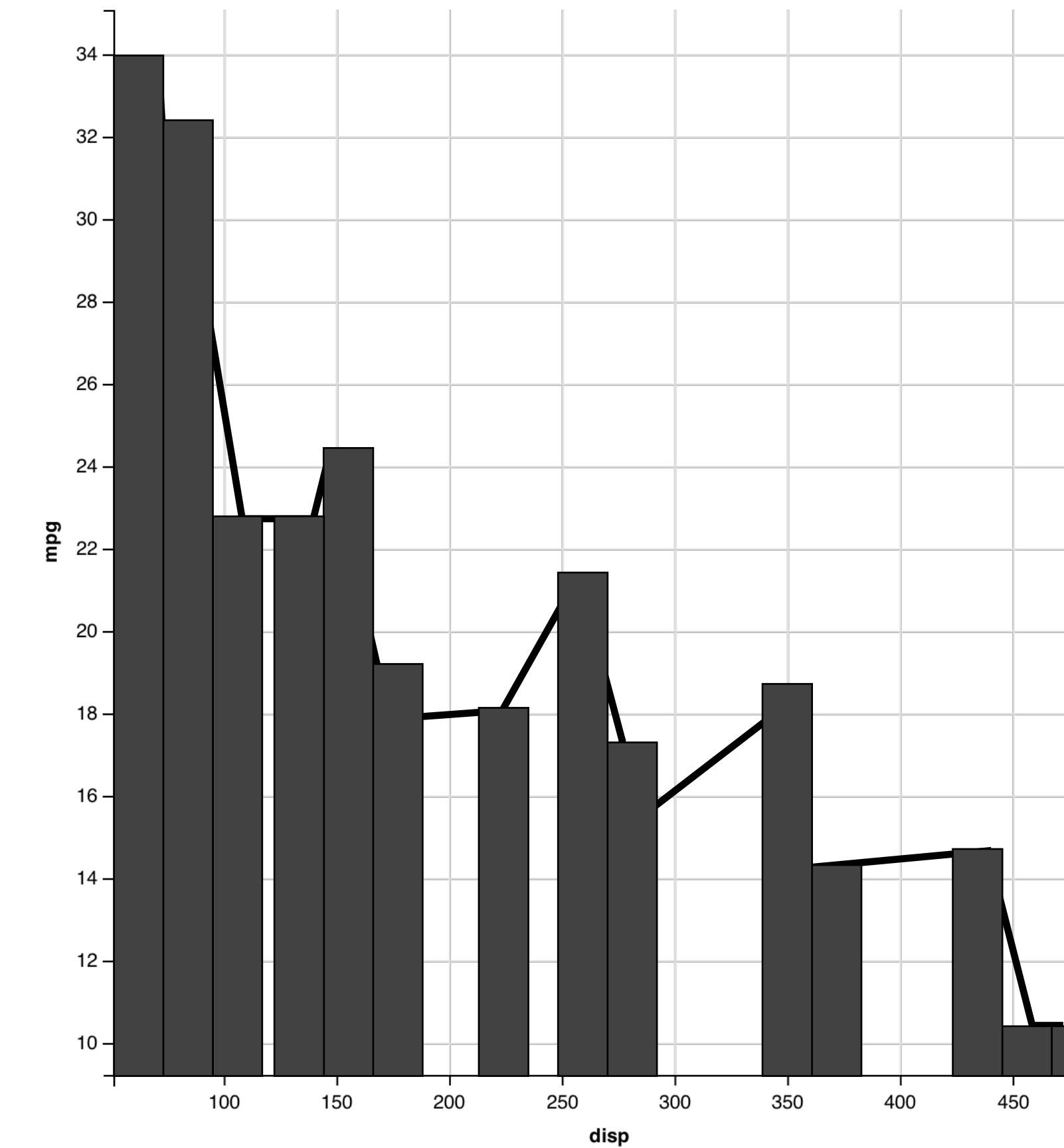


# mappings

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

data

geom  
points  
lines  
bars

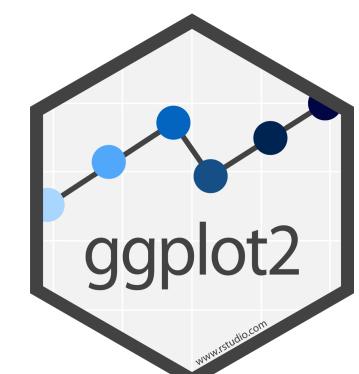
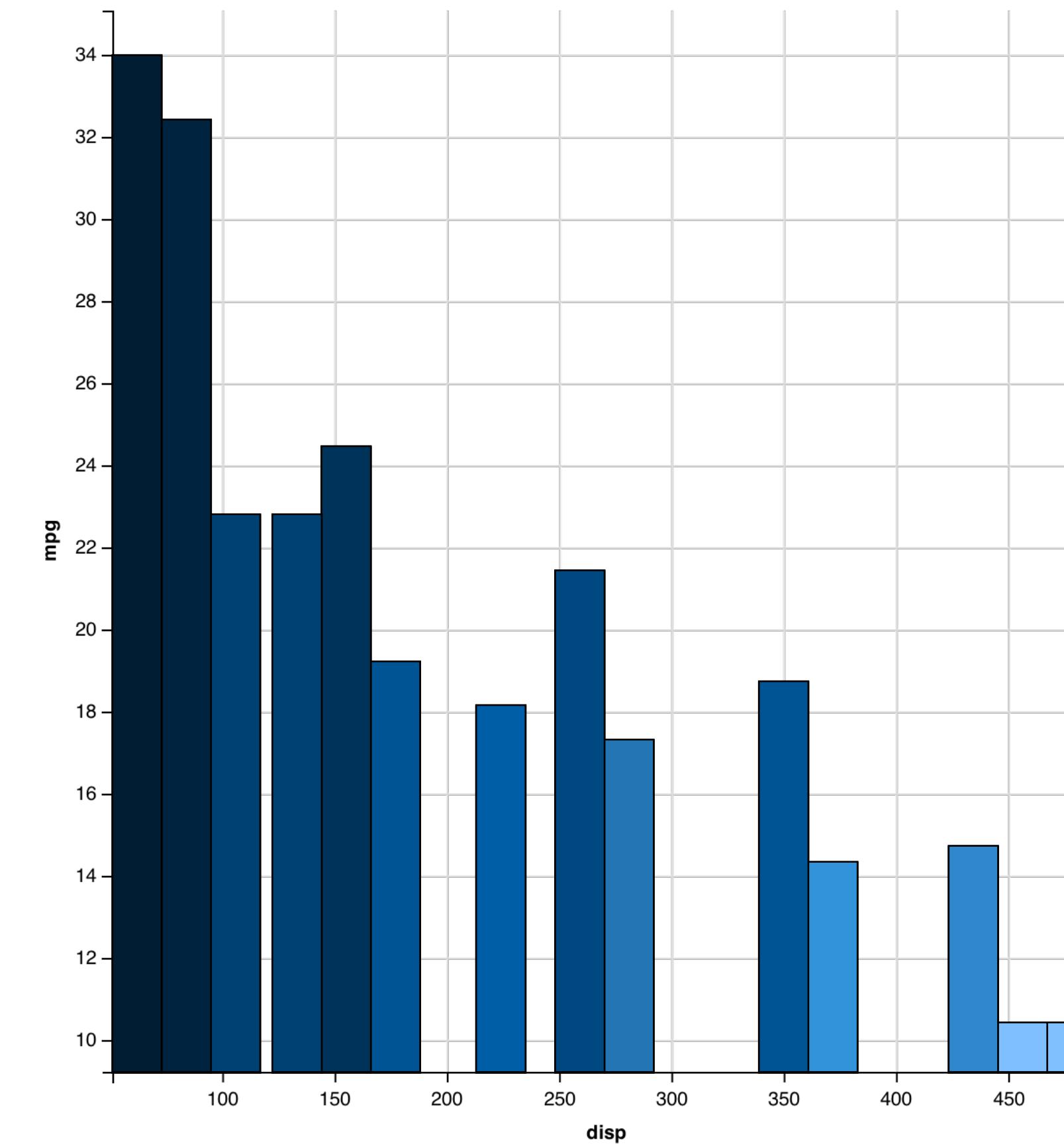


# mappings

	y		fill	
	mpg	cyl	disp	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

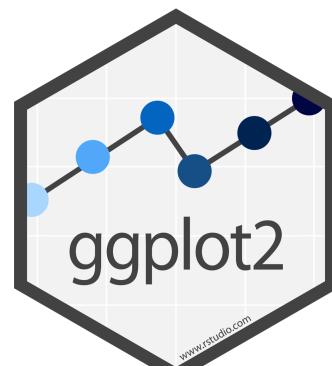
geom  
points  
lines  
bars



# To make a graph

[template]

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



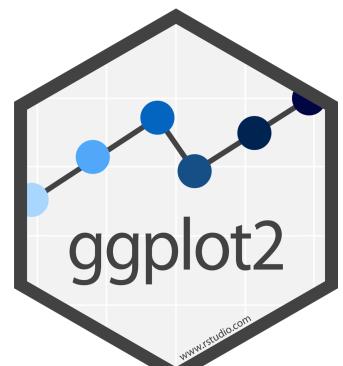
# To make a graph

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

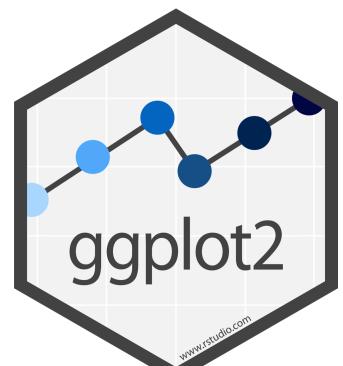
data

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	fill
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

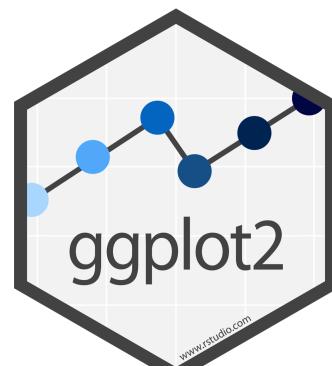
geom

1. Pick a **data** set

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

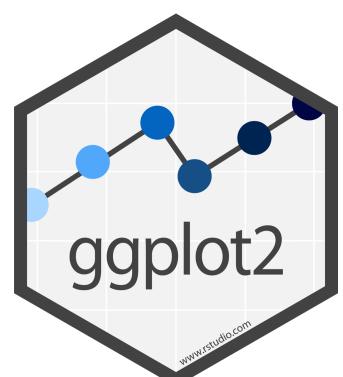
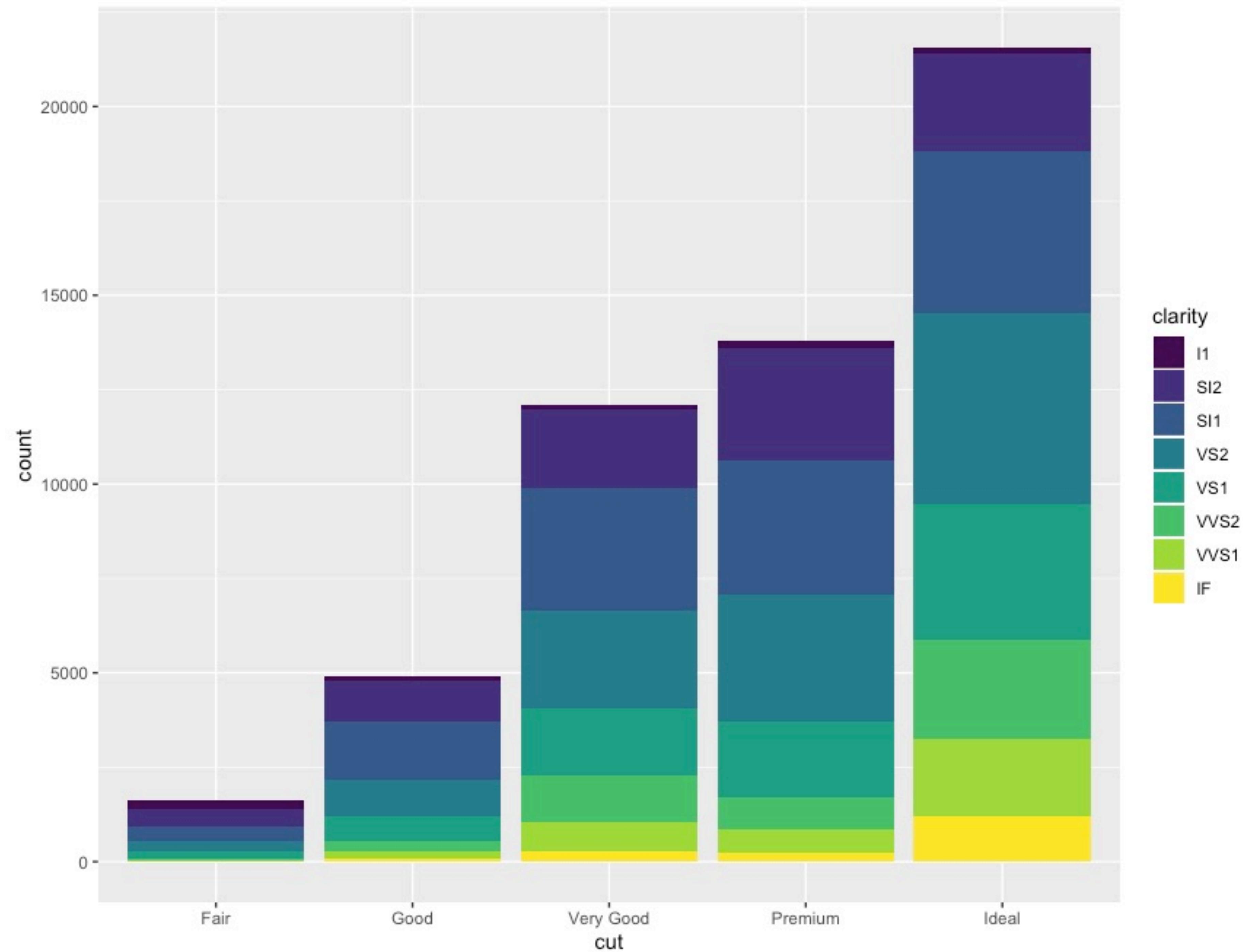
2. Choose a **geom**  
to display cases

3. **Map** aesthetic  
properties to  
variables



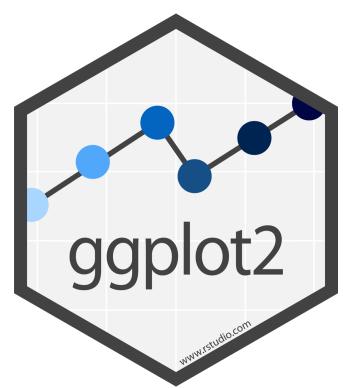
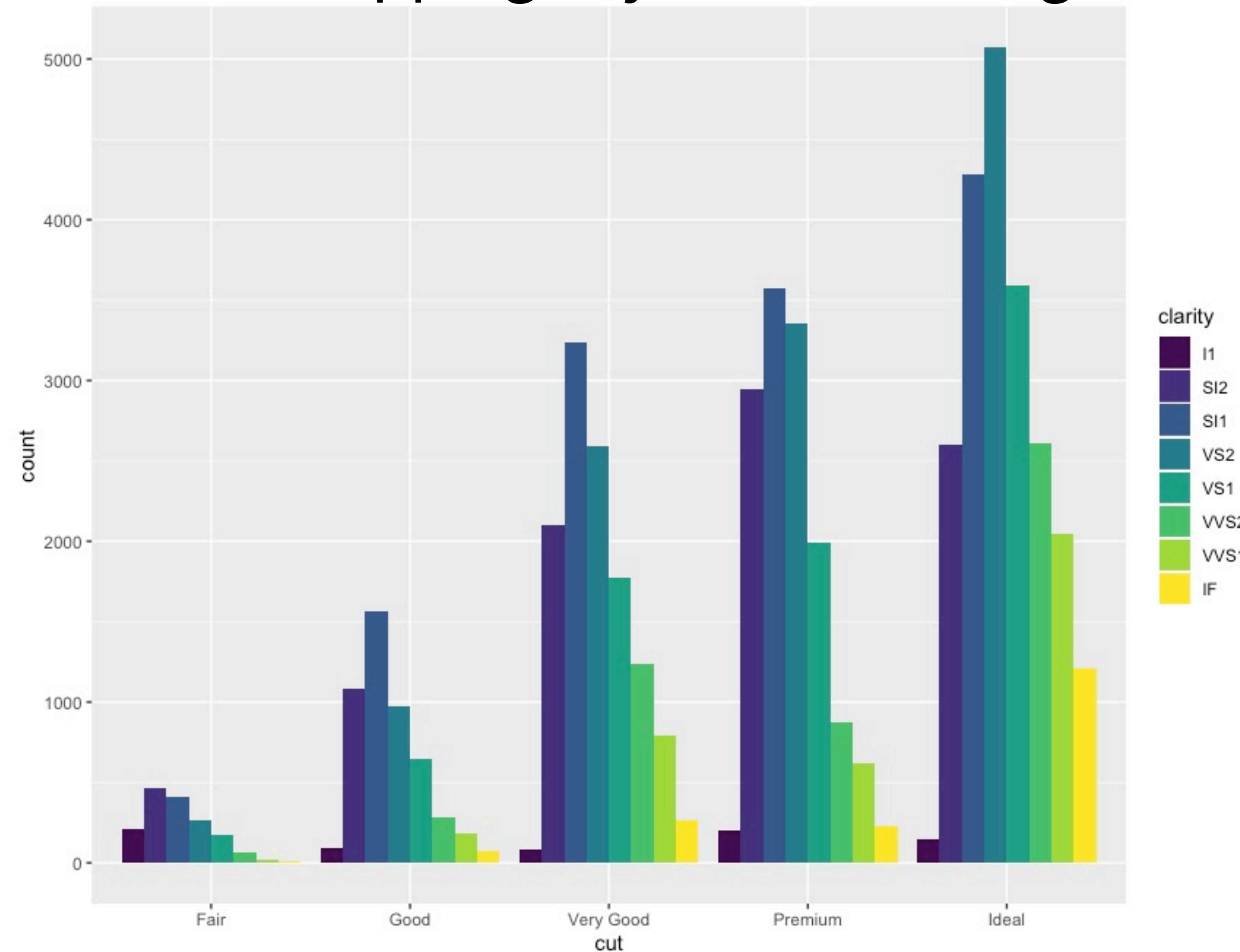
# what else?

R



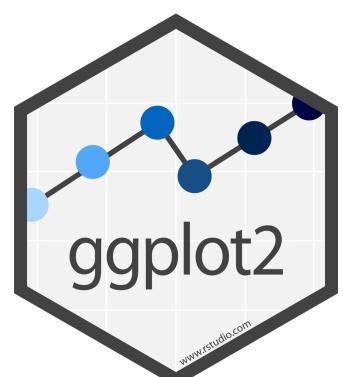
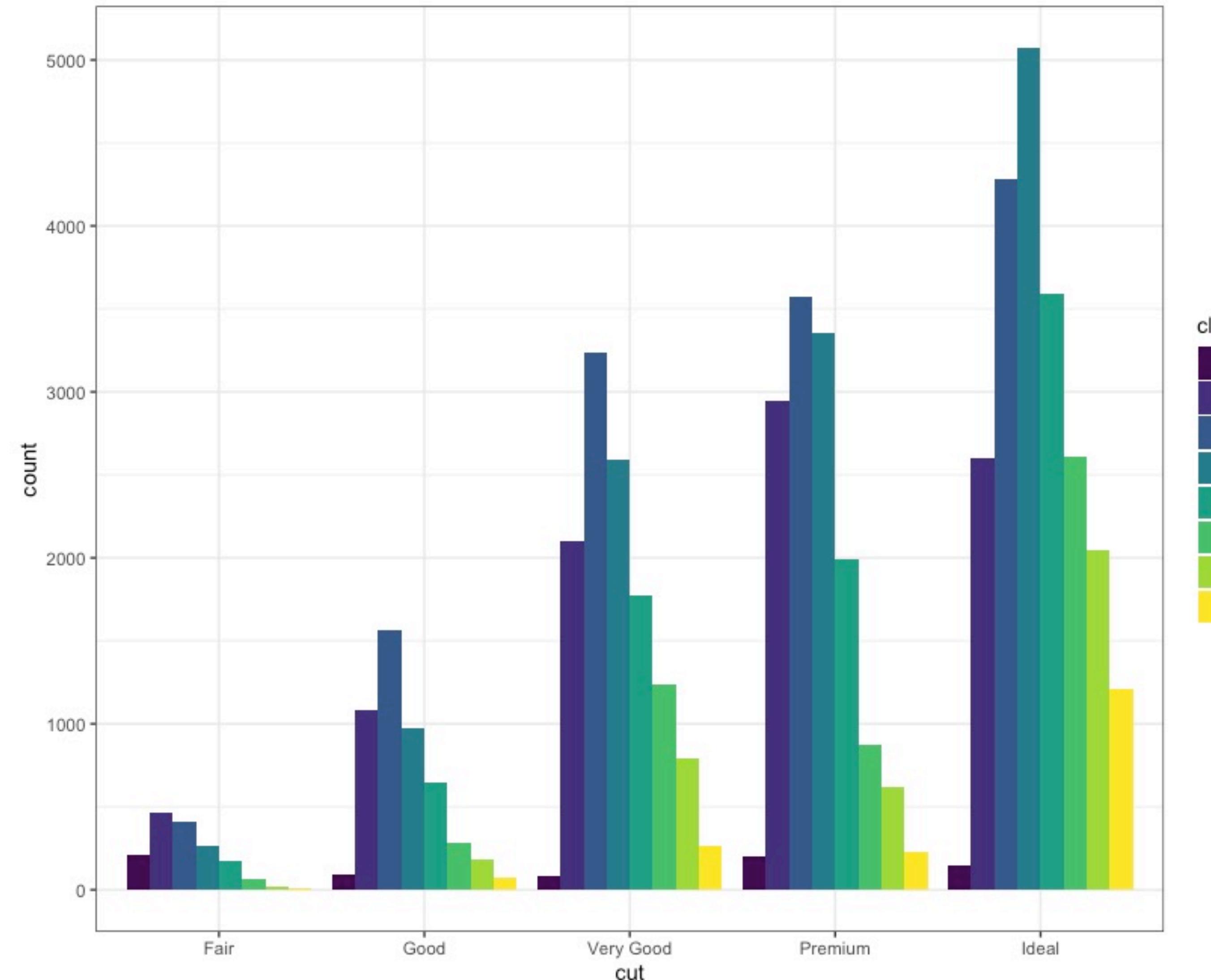
# Position Adjustments

How overlapping objects are arranged



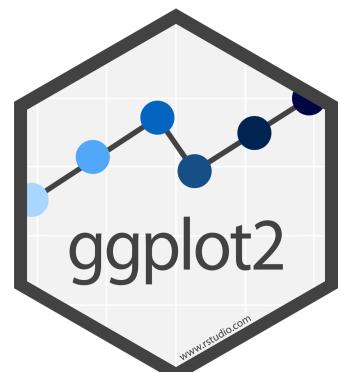
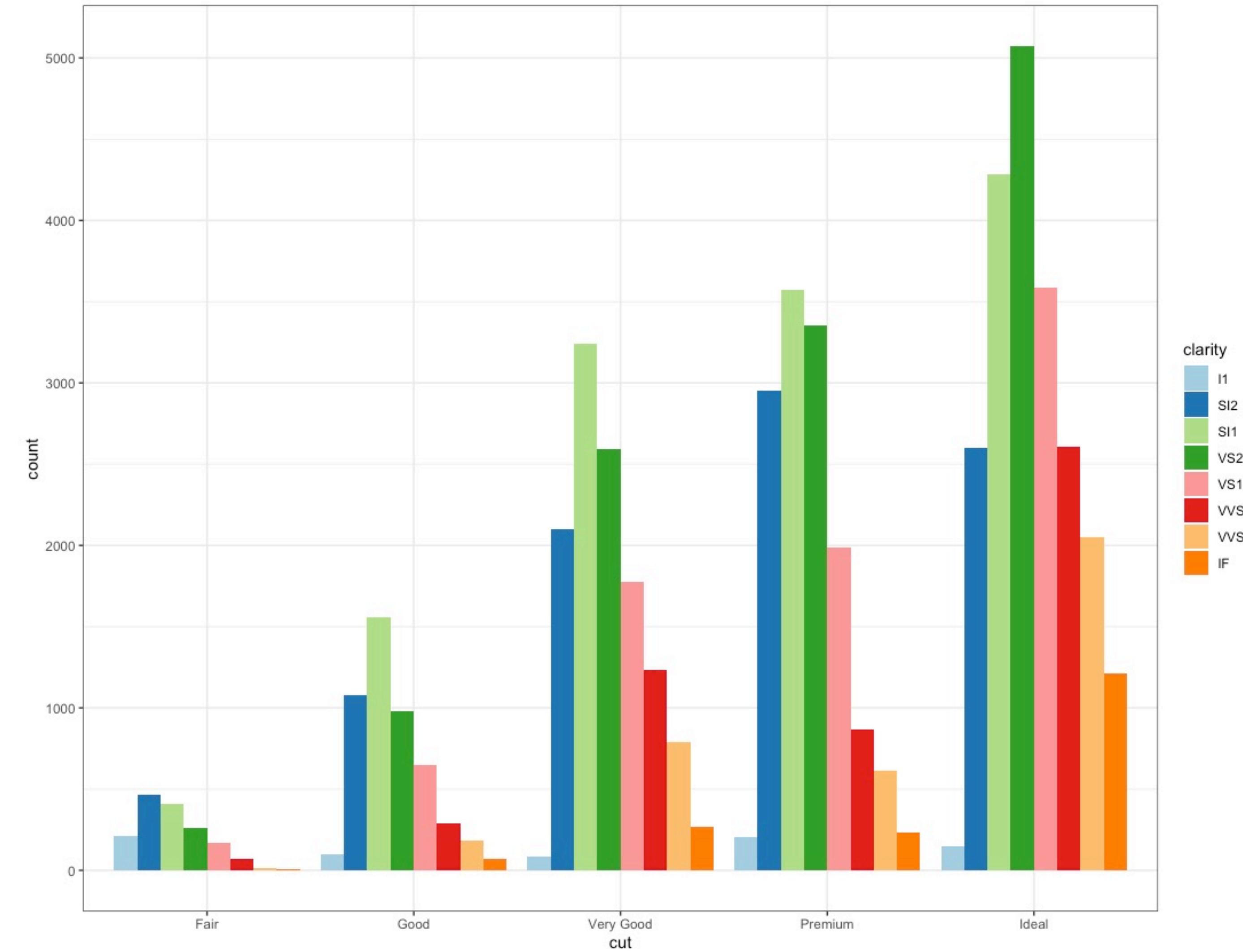
# Themes

## Visual appearance of non-data elements



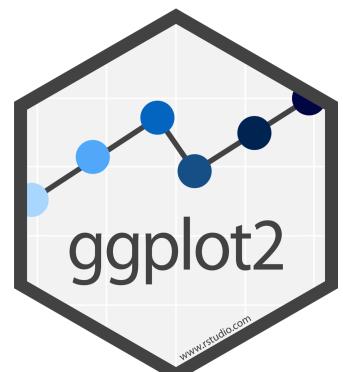
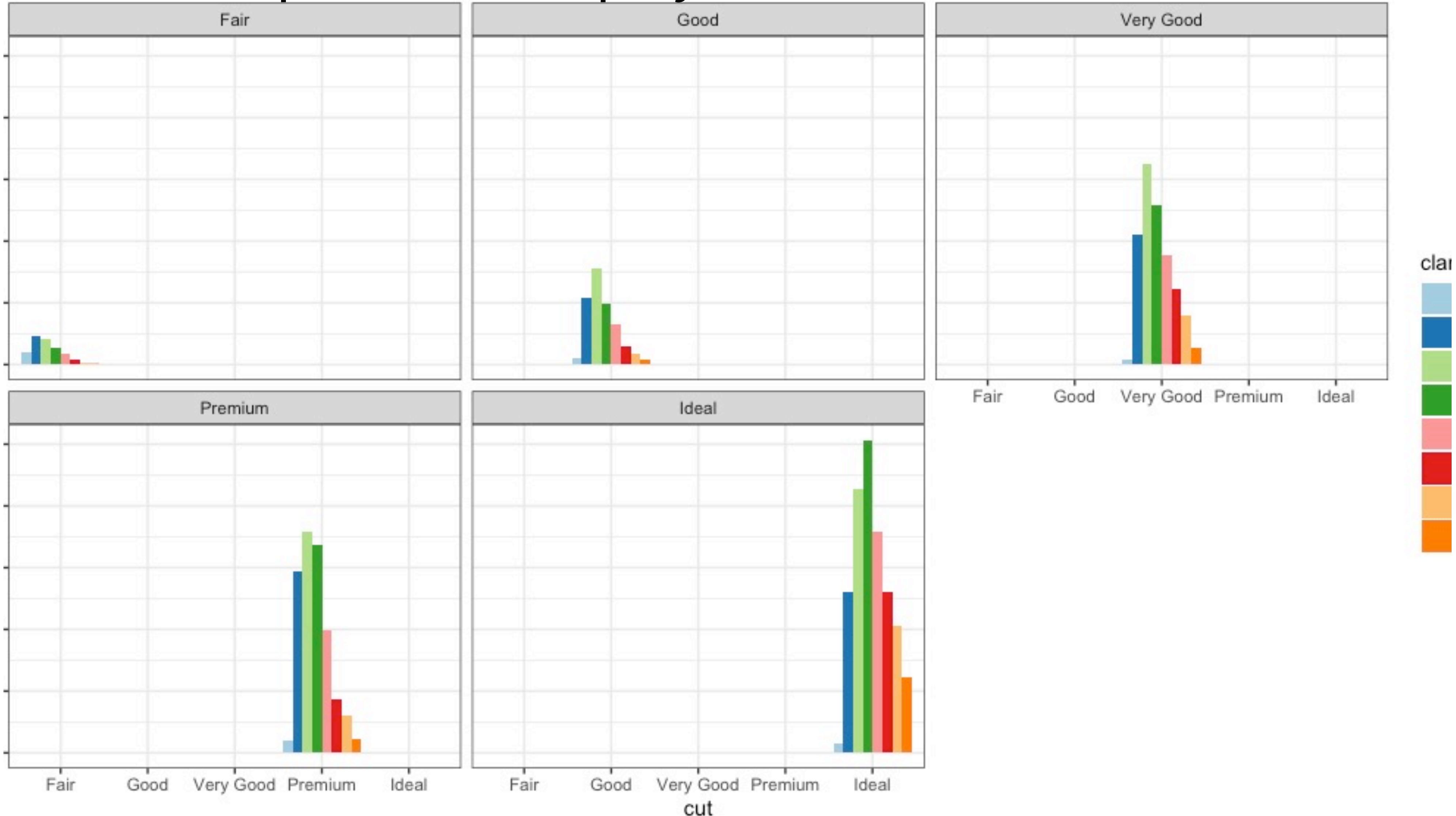
# Scales

Customize color scales, other mappings

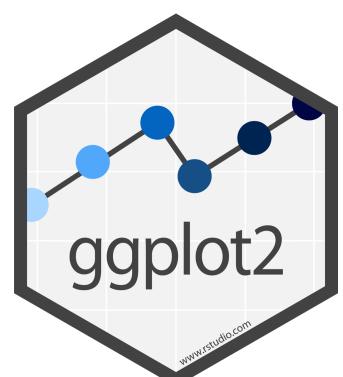
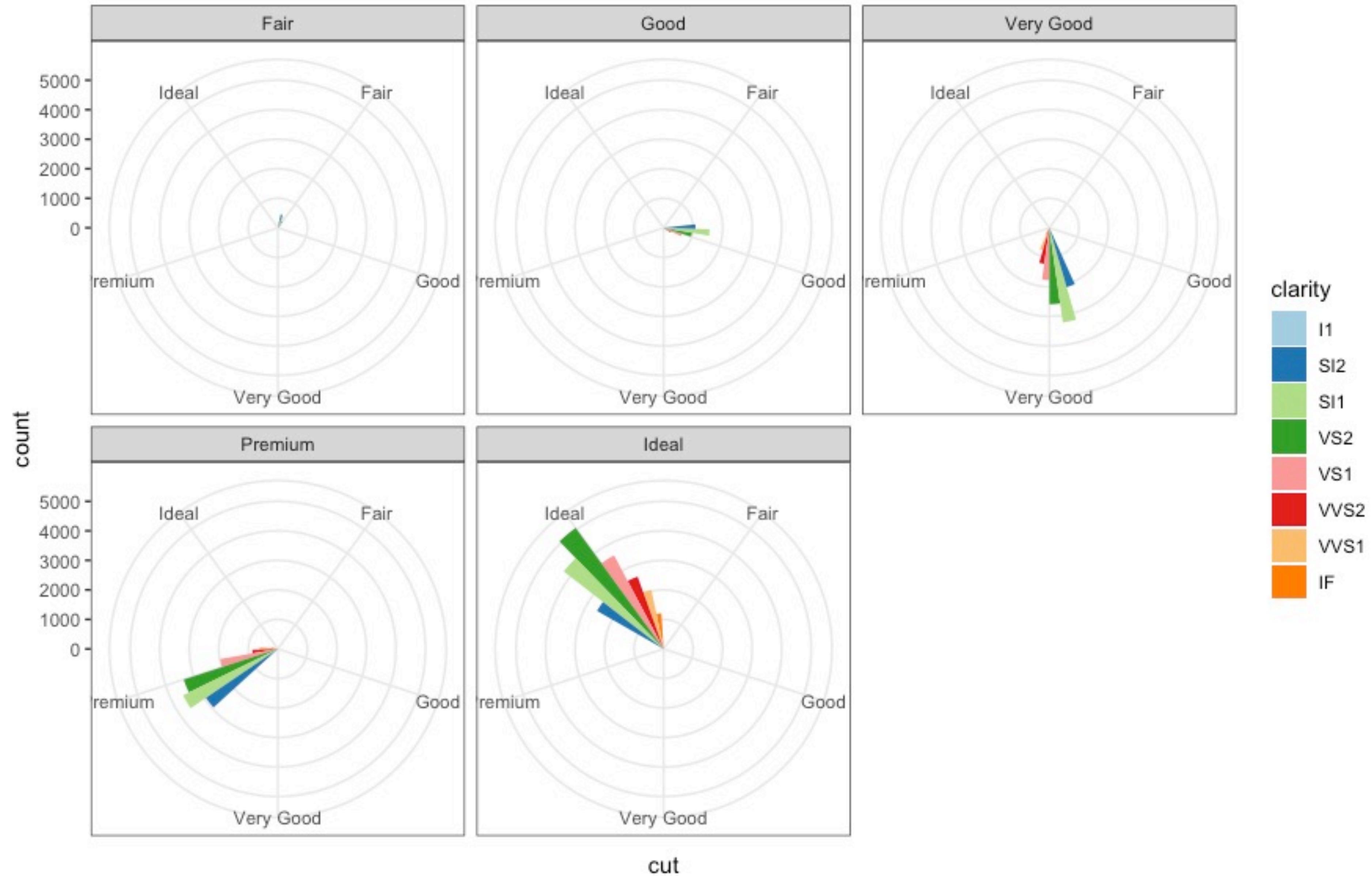


# Facets

Subplots that display subsets of the data.



# Coordinate systems

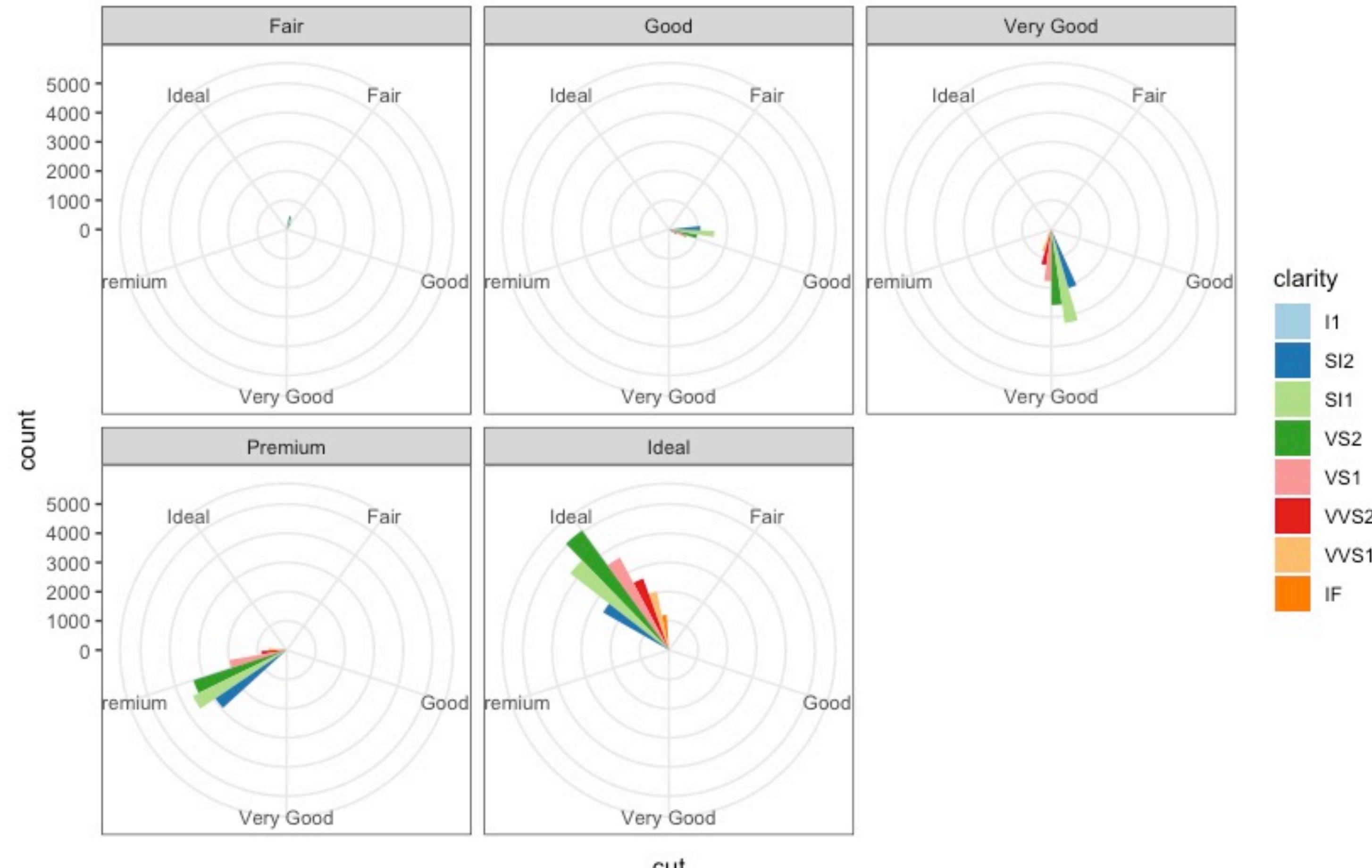


# Titles and captions

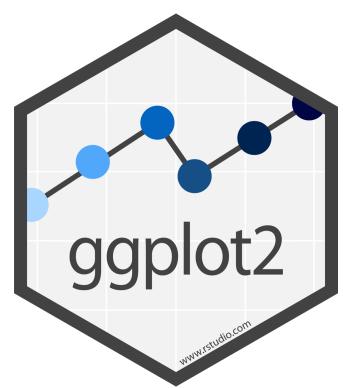
## Diamonds data

The data set is skewed towards ideal cut diamonds

The data is skewed toward ideal cut diamonds



Data by Hadley Wickham

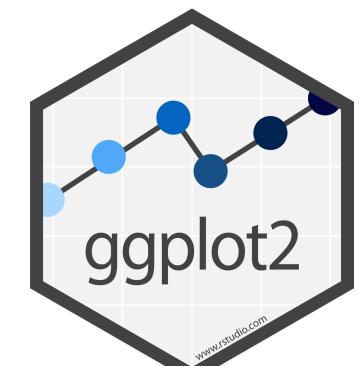
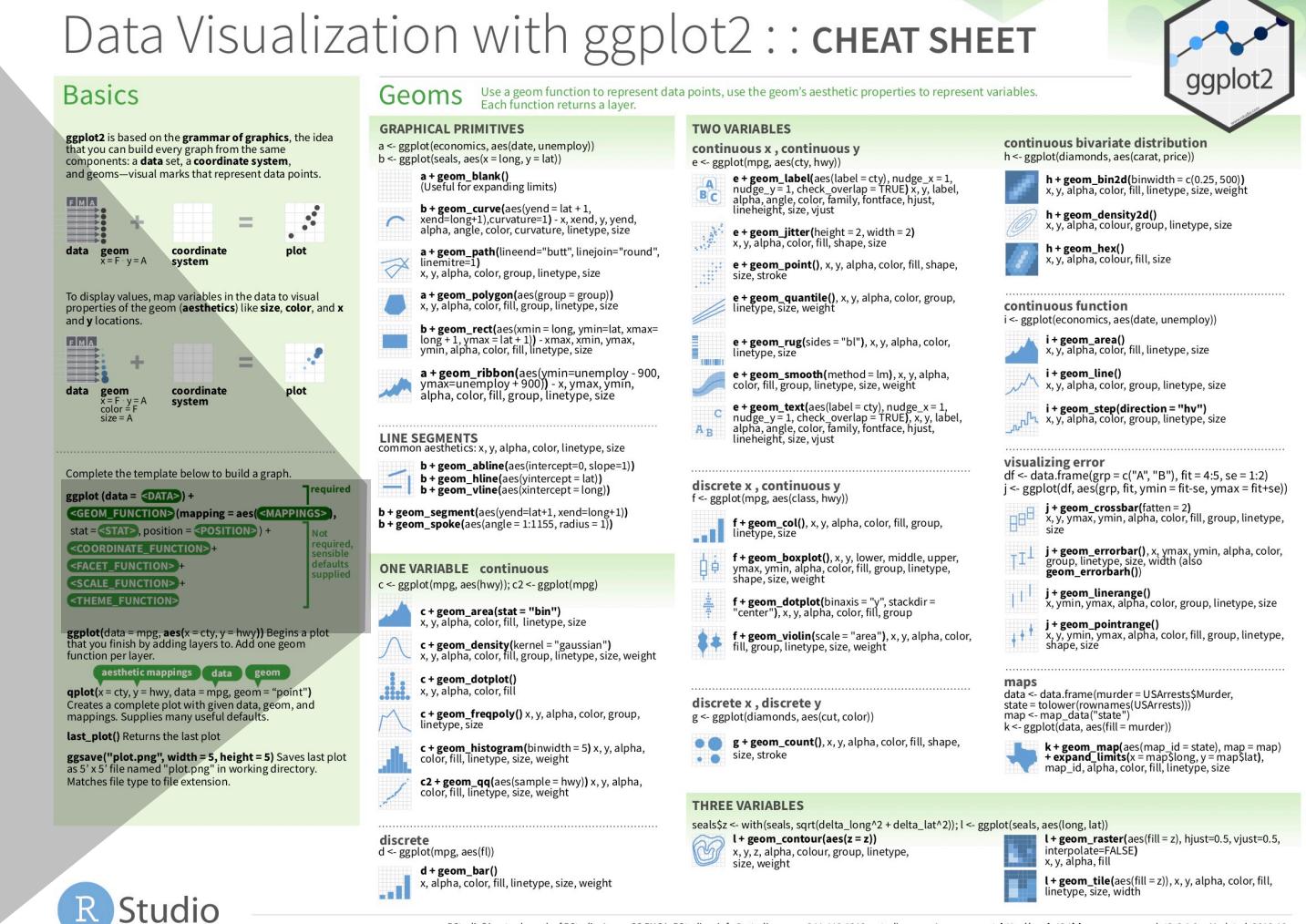


# A ggplot2 template

Make any plot by filling in the parameters of this template

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

↑ required  
↓ Not required, sensible defaults supplied



# ggplot2.tidyverse.org

The screenshot shows a web browser window with the URL <https://ggplot2.tidyverse.org>. The page title is "Create Elegant Data Visualisation". The main content area features the **ggplot2** logo and navigation links for Reference, Articles, News, Extensions, and Developers. A sidebar on the left contains the "Usage" section and a code snippet. The right side lists the developers: Hadley Wickham, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and All authors... At the bottom, there is a small ggplot2 logo.

## 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) and vehicle class (class) on the x-axis. The plot uses a color scale to represent vehicle classes. The y-axis ranges from approximately 10 to 50 mpg, with a break between 40 and 45. The x-axis shows categories like compact, subcompact, minivan, etc. A few data points are visible at the bottom left.

Developers

- [Hadley Wickham](#)  
Author, maintainer
- [Winston Chang](#)  
Author
- [Lionel Henry](#)  
Author
- [Thomas Lin Pedersen](#)  
Author
- [Kohske Takahashi](#)  
Author
- [Claus Wilke](#)  
Author
- [Kara Woo](#)  
Author
- [Hiroaki Yutani](#)  
Author
- All authors...

# Visualize Data with

