

These materials adapted by Amelia McNamara from  
the RStudio [CC BY-SA](#) materials Introduction to R  
(2014) and [Master the Tidyverse](#) (2017).

# Introduction to R & RStudio: deck 02: Visualization

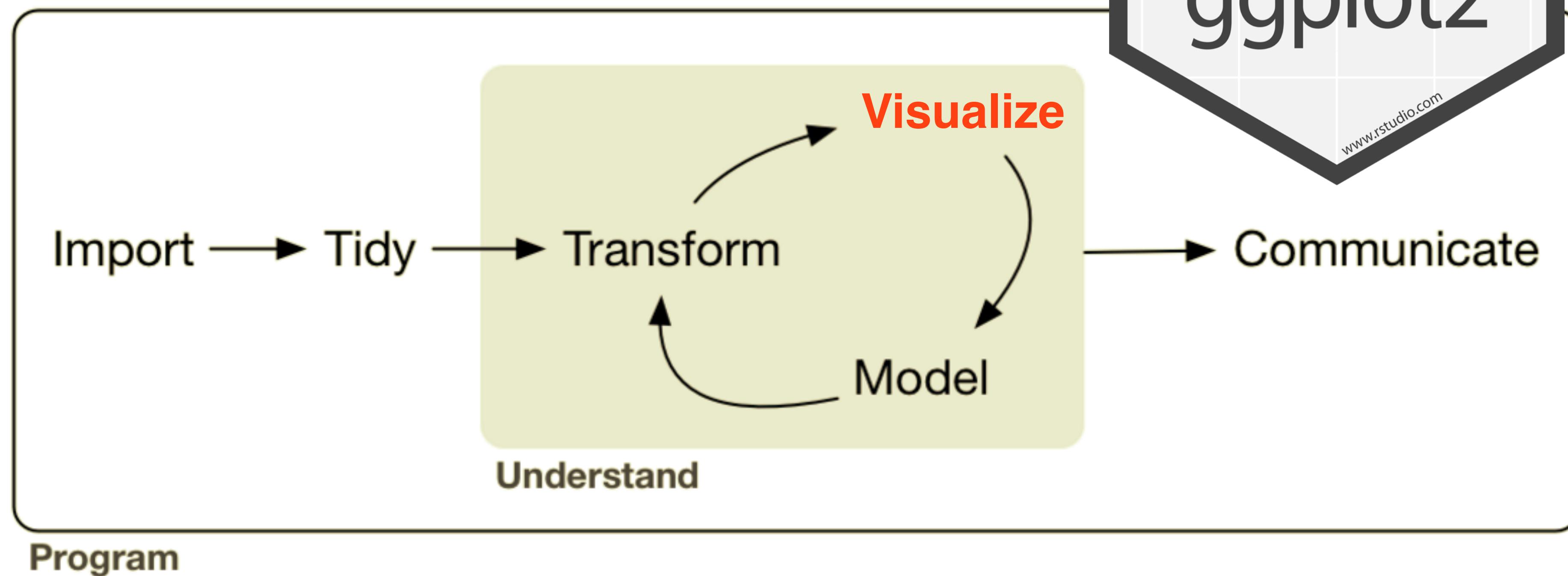
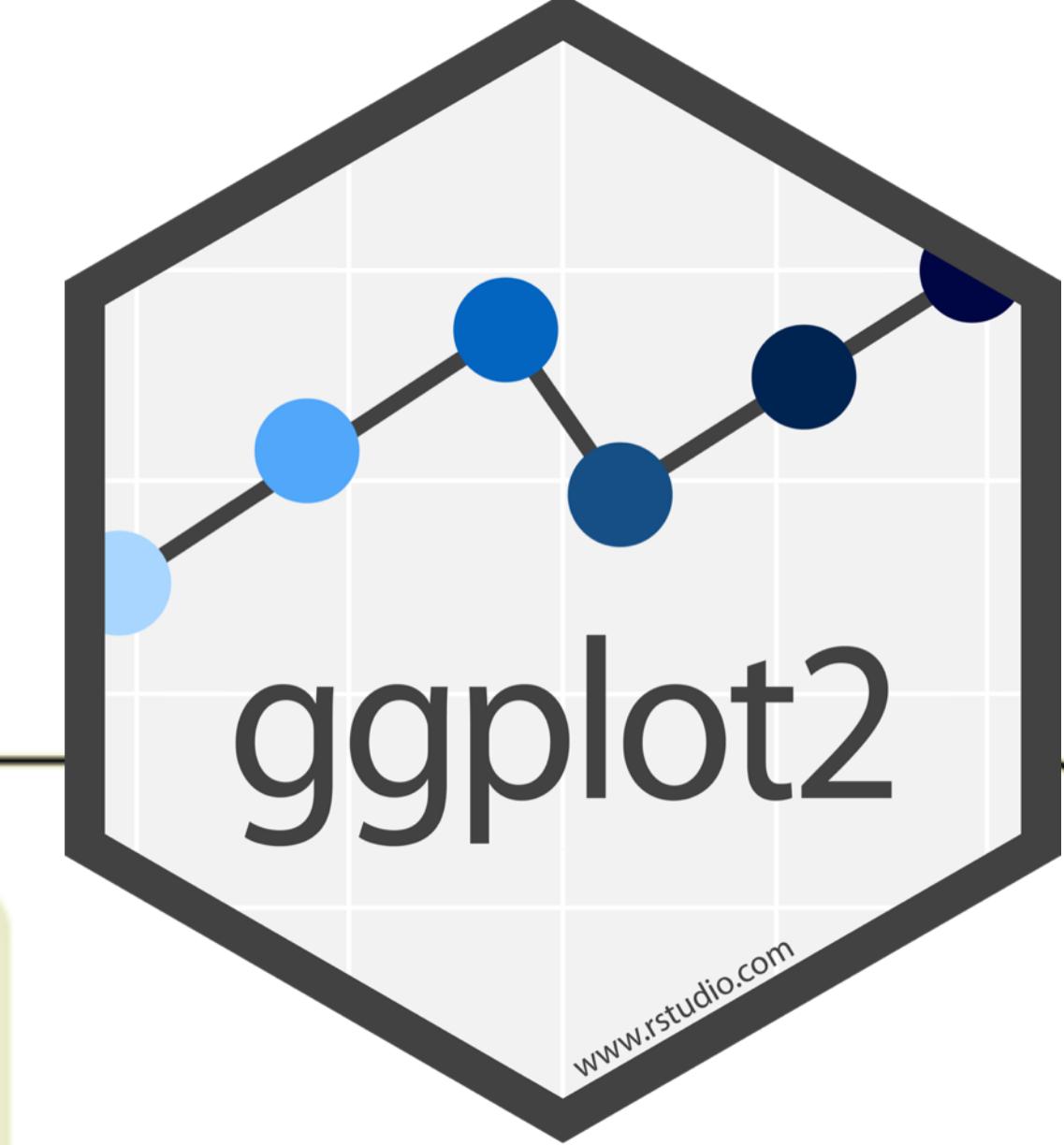
**Amelia McNamara**

Visiting Assistant Professor of Statistical and Data Sciences  
Smith College

**rstudio::conf 2018**

1. Data visualization
2. Aesthetics
3. set vs. map
4. geoms
5. Saving graphs
6. What else?

Data  
visualization  
(good stuff first)



From *R for Data Science* by Hadley Wickham and Garrett Grolemund.

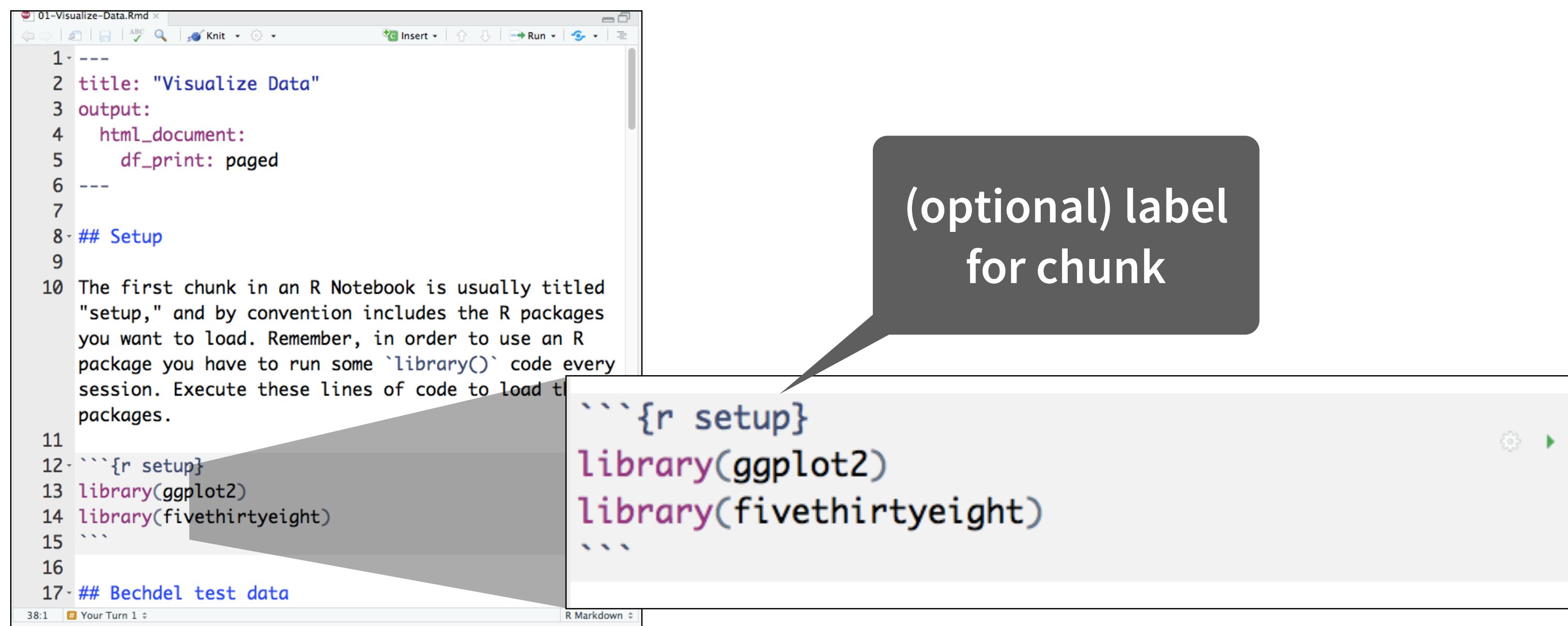
The screenshot shows the RStudio interface with the following components:

- Left Panel (Code Editor):** Displays the content of the file `02-Visualize-Data.Rmd`. The code includes YAML front matter, a setup chunk, and a Bechdel test data chunk.
- Top Bar:** Shows the RStudio logo and a "Project: (None)" dropdown.
- Environment Tab:** Shows the Global Environment pane.
- Files Tab:** Shows a file tree under "Dropbox > Intro\_to\_R\_and\_RStudio > Day1".
- Right Panel:** Displays the rendered content of the R Notebook, featuring a large title **Open the R Notebook 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
```

# Setup

The setup chunk is always run once before anything else



(optional) label  
for chunk

```
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 ```{r setup}  
18 library(ggplot2)  
19 library(fivethirtyeight)  
20 ````  
21  
22 ## Bechdel test data
```

# bechdel

## Data on movies and the Bechtel test

bechdel

# bechdel

## Data on movies and the Bechtel test

View(bechdel)

The screenshot shows the RStudio interface. At the top, there are two tabs: "02-Visualize-Data.Rmd" and "bechdel". Below the tabs is a search bar and a "Filter" button. The main area displays a data frame with 14 rows and 7 columns. The columns are: year, imdb, title, test, clean\_test, binary, and budget. The data includes various movie titles from 2012 and 2013, along with their Bechtel test results and budgets. A tooltip with a dark gray background and white text reads: "Type this in your Console, NOT in your Notebook". In the bottom left corner of the RStudio interface, there is a "Console" tab with the command "> View(bechdel)" and a cursor.

	year	imdb	title	test	clean_test	binary	budget
1	2013	tt1711425	21 & Over	notalk	notalk	FAIL	1.30e+07
2	2012	tt1343727	Dredd 3D	ok-disagree	ok	PASS	4.50e+07
3	2013	tt2024544	12 Years a Slave	notalk-disagree	notalk	FAIL	2.00e+07
4	2013	tt1272878	2 Guns	notalk	notalk	FAIL	6.10e+07
5	2013	tt0453562	42	men	men	FAIL	4.00e+07
6	2013	tt1335975	47 Ronin	men	men	FAIL	2.25e+08
7	2013	tt1606378	A Good Day to Die Hard	notalk	notalk	FAIL	9.20e+07
8	2013	tt2194499	About Time	ok-disagree	ok	PASS	1.20e+07
9	2013	tt1814621	Admission	ok	ok	PASS	1.30e+07
10	2013	tt1815862	After Earth	notalk	notalk	FAIL	1.30e+08
11	2013	tt1800241	American Hustle	ok-disagree	ok	PASS	4.00e+07
12	2013	tt1322269	August: Osage County	ok	ok	PASS	2.50e+07
13	2013	tt1559547	Beautiful Creatures	ok	ok	PASS	5.00e+07

Showing 1 to 14 of 1,794 entries

Console Terminal R Markdown

~/Dropbox/Intro\_to\_R\_and\_RStudio/master-the-tidyverse-master/ ↵

> View(bechdel)  
> |

# Consider

Confer with the people around you.  
What relationship do you expect to see  
between movie budget (budget) and  
domestic gross(domgross)?



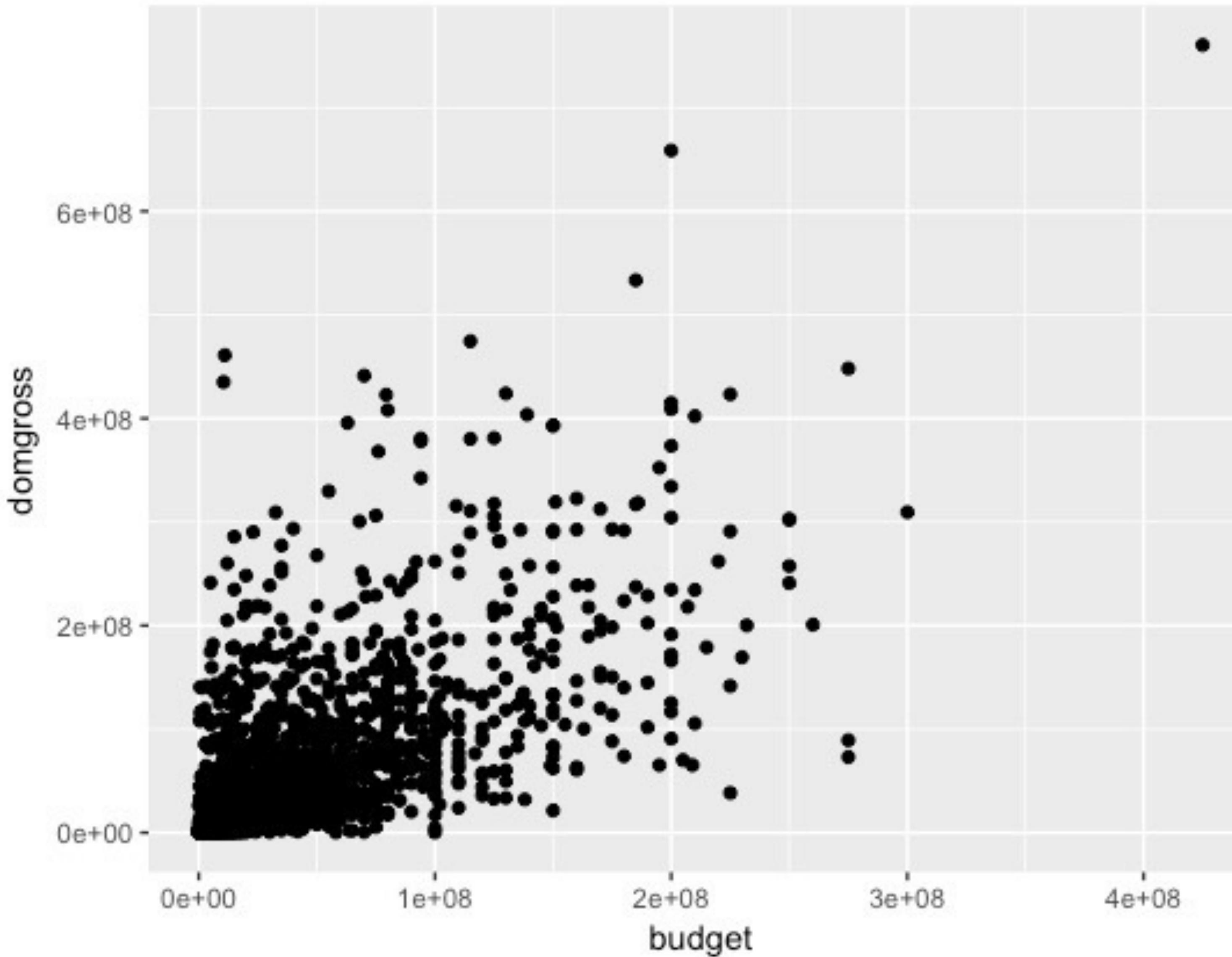
# Your Turn 1

Run this code in your notebook to make a graph.

Pay strict attention to spelling, capitalization, and parentheses!

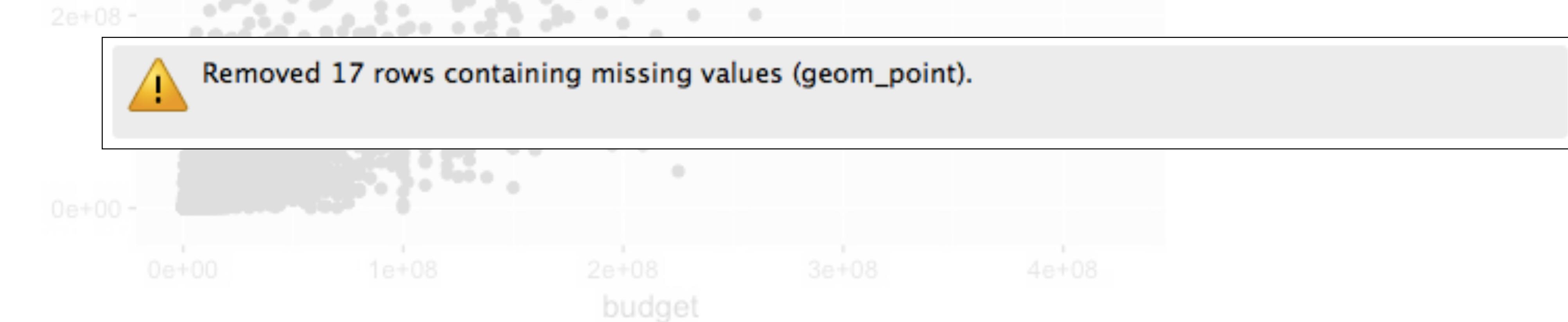
```
ggplot(data = bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross))
```





```
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))
```

data

+ before new line

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

type of layer

aes()

x variable

y variable

# A template

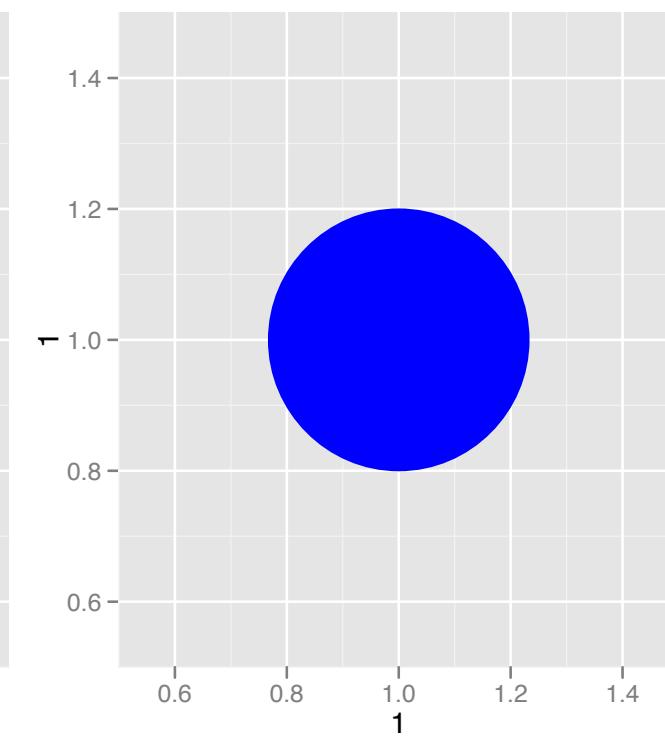
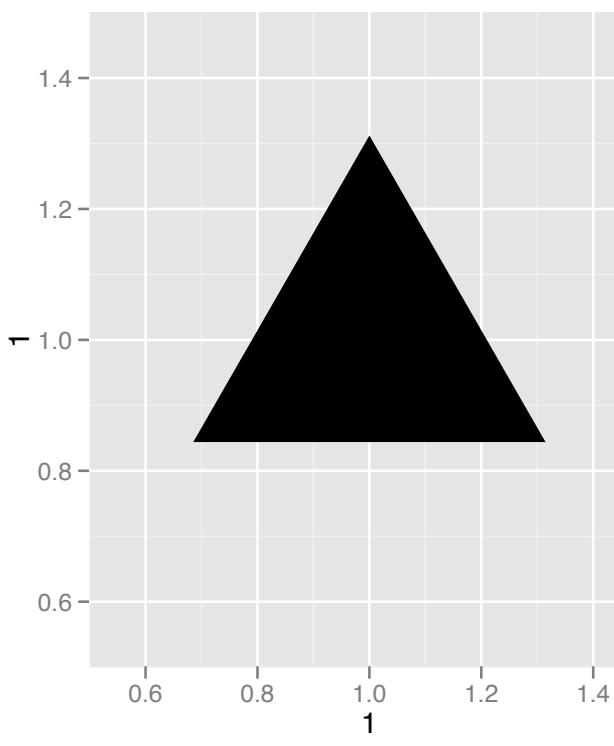
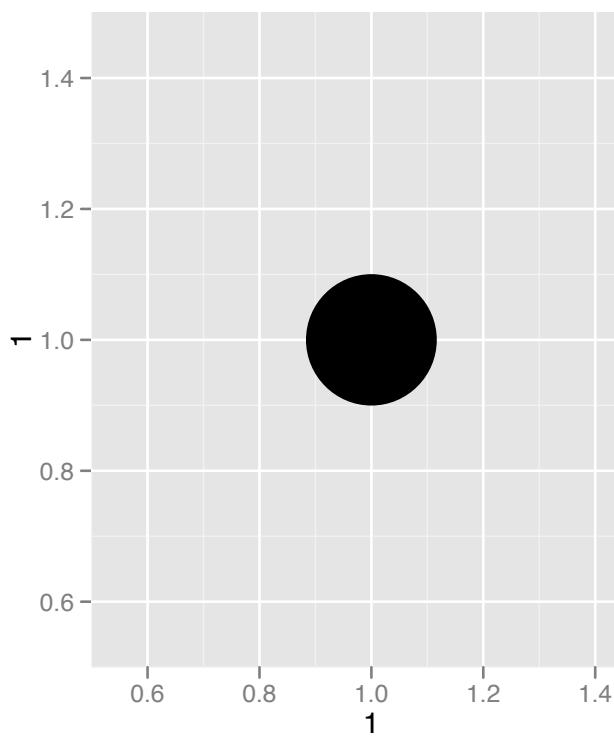
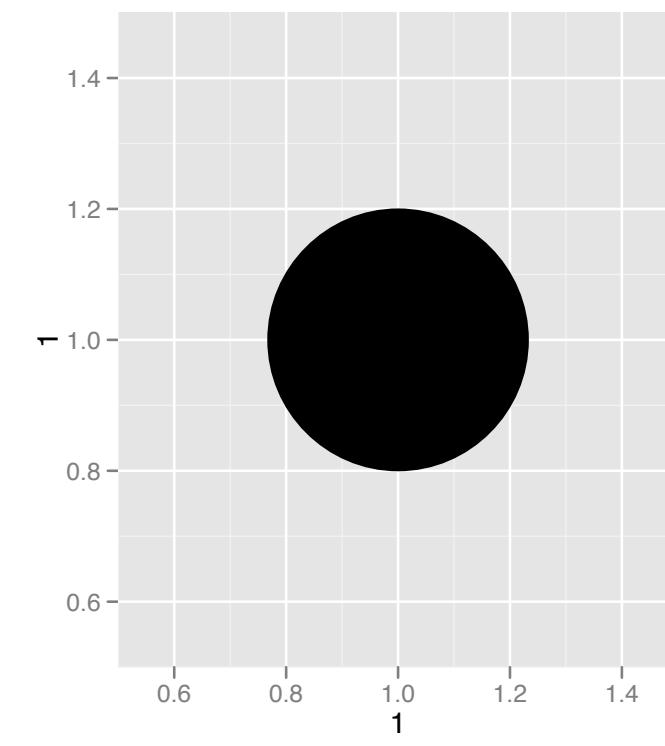
```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))  
    geom_point(mapping = aes(x = budget, y = domgross))
```

# A template

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

# Aesthetics

# Aesthetics



## Visual Space      Data Space

color ←→ clean\_test

Red ←→ nowomen

Brown ←→ notalk

Green ←→ men

Blue ←→ dubious

Pink ←→ ok

# Aesthetics

aesthetic  
property

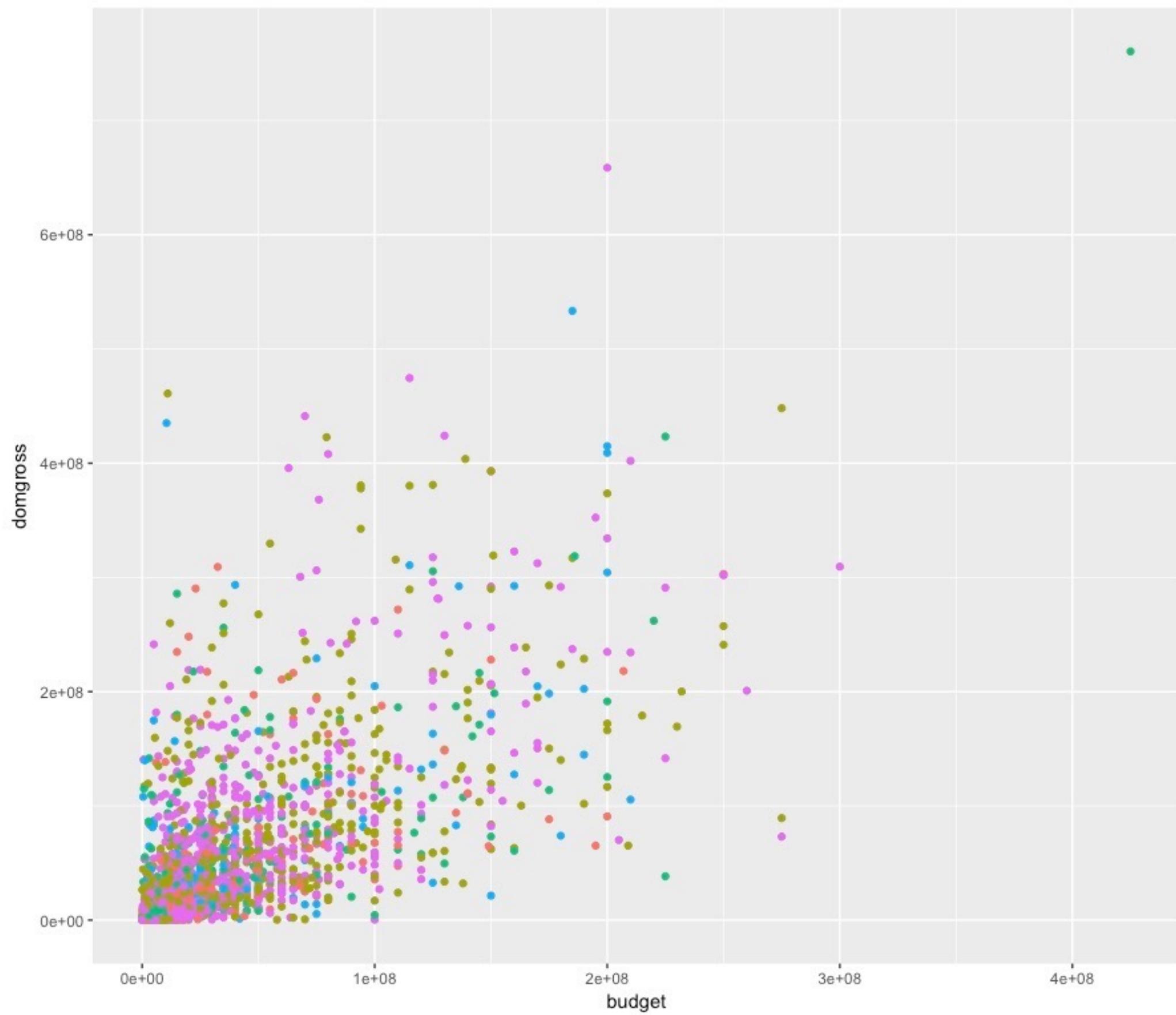
Variable to  
map it to

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

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

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

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



Legend added  
automatically

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

# Your Turn 2

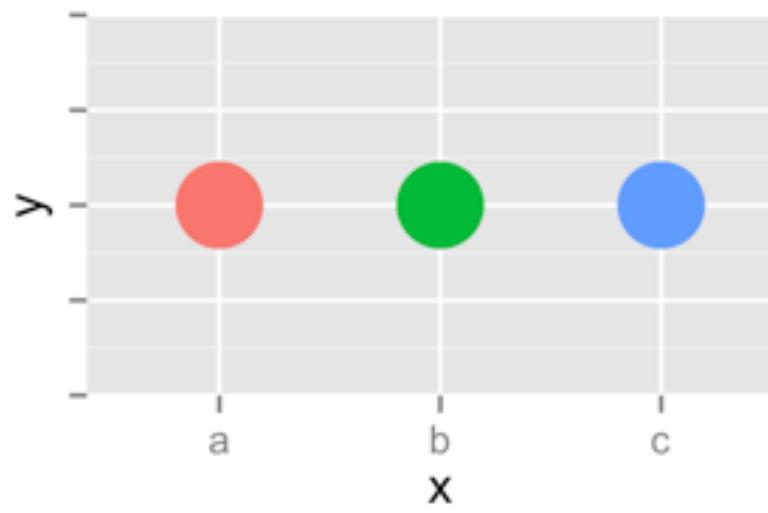
In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

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

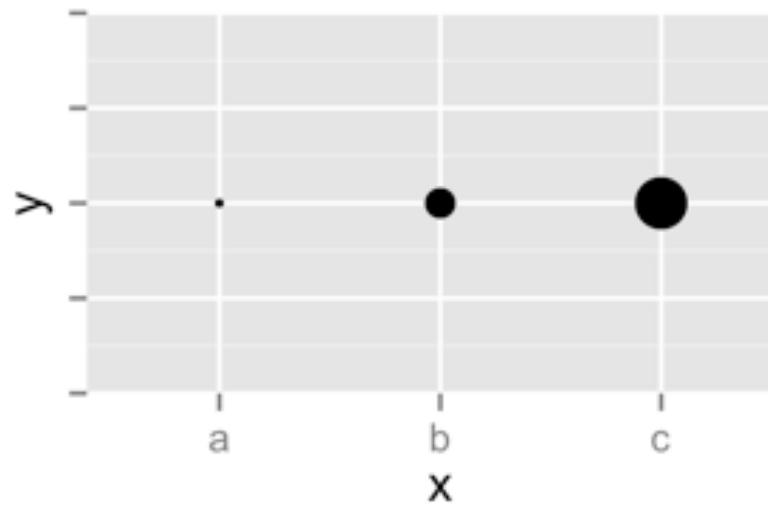
What happens when you use more than one aesthetic?



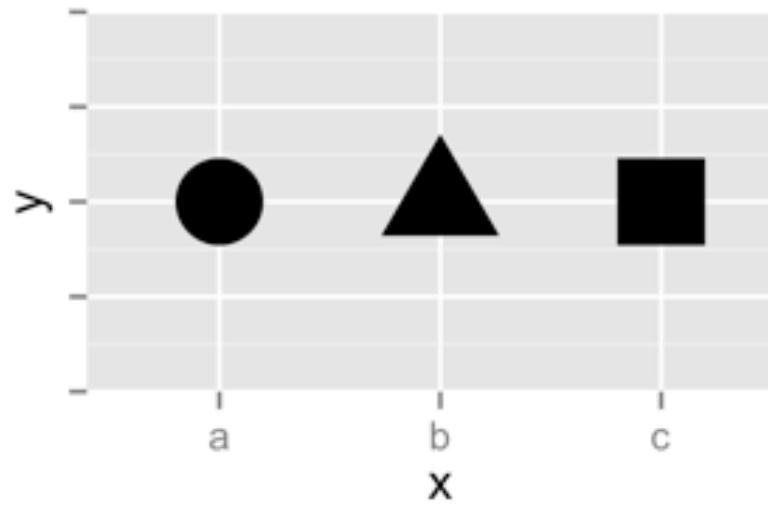
Color



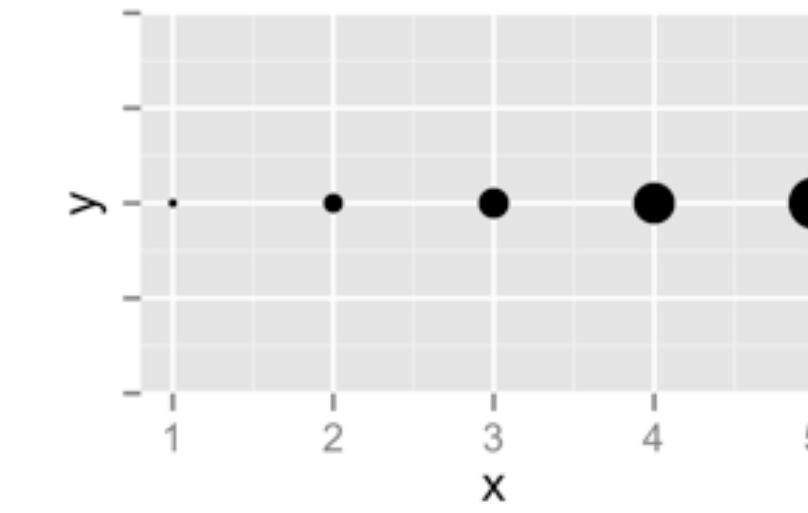
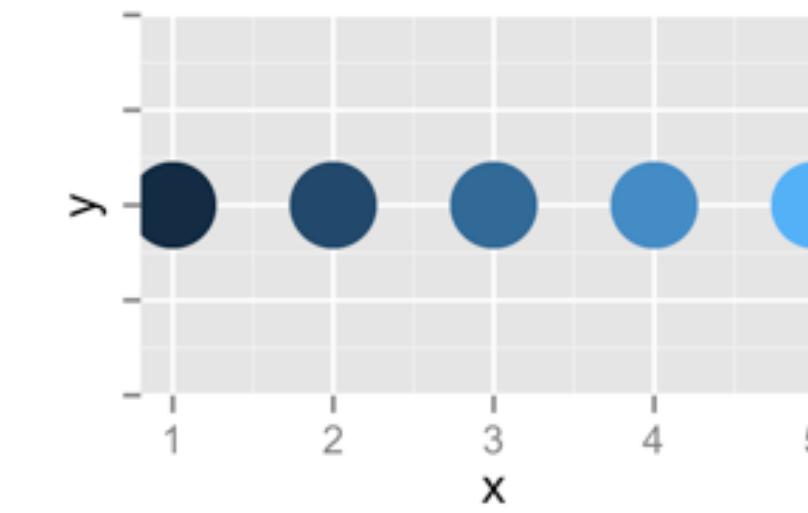
Size



Shape

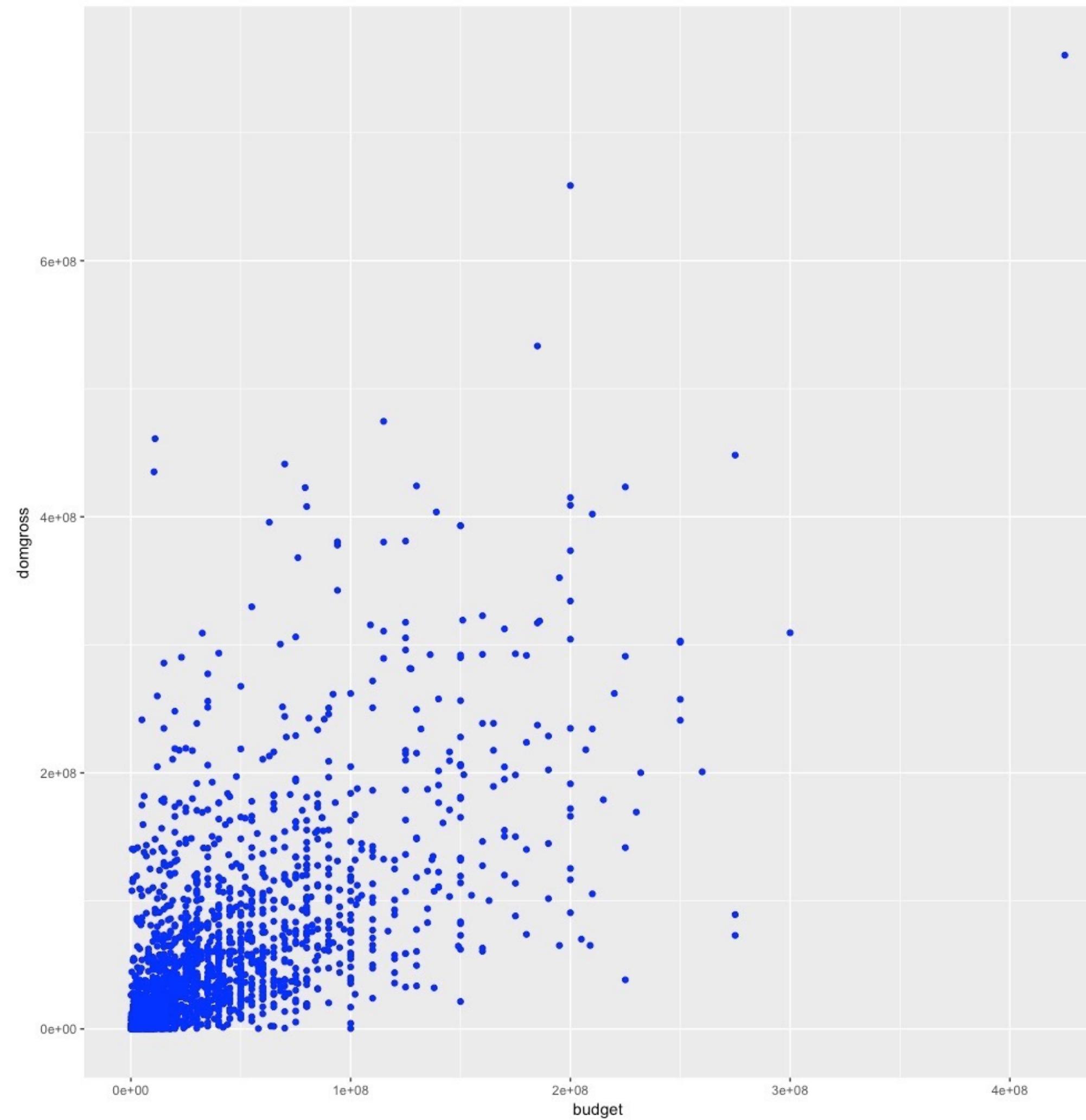


Continuous



# set vs. map

# How would you make this plot?

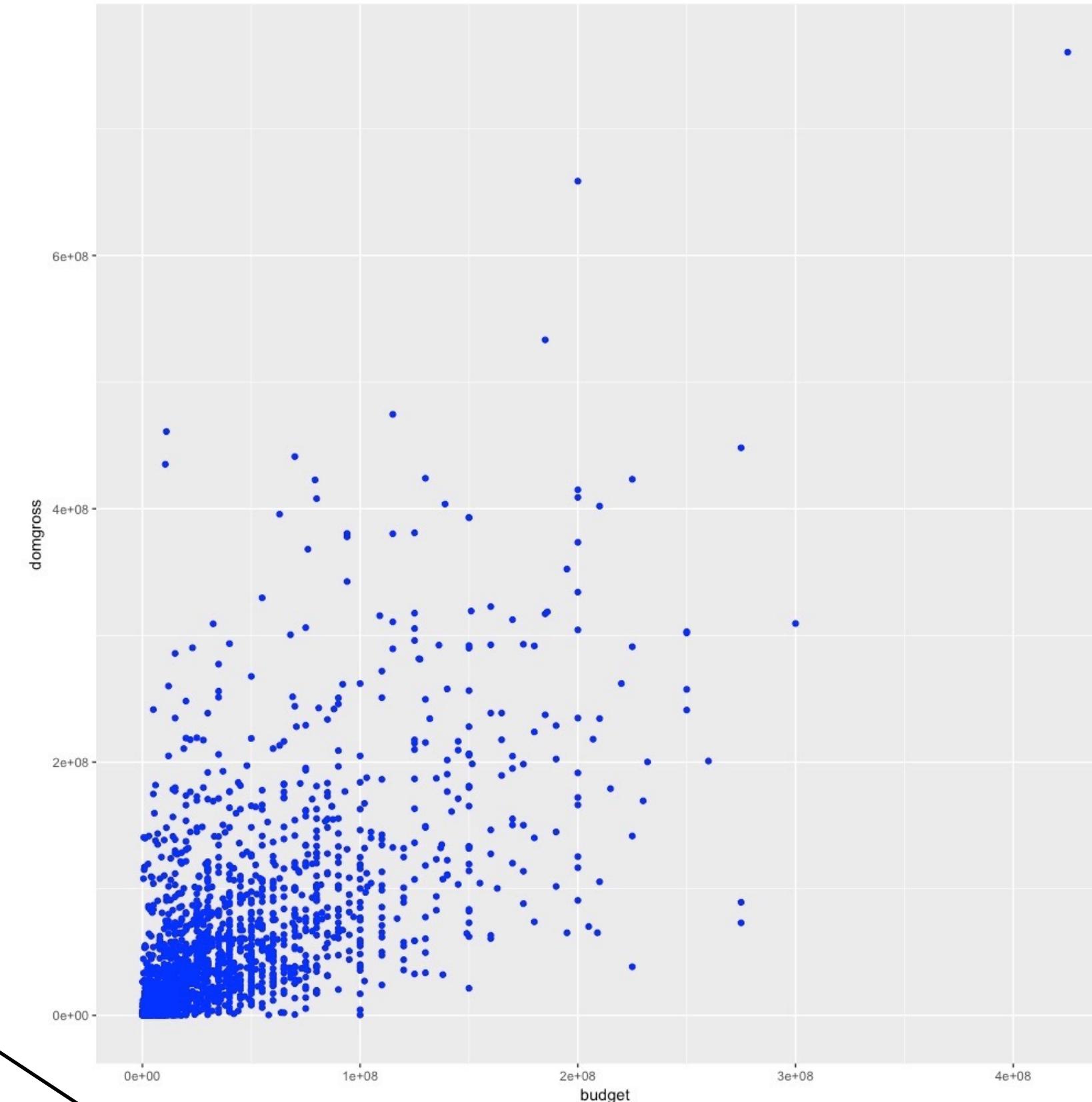




**Inside of aes():** maps an aesthetic to a variable

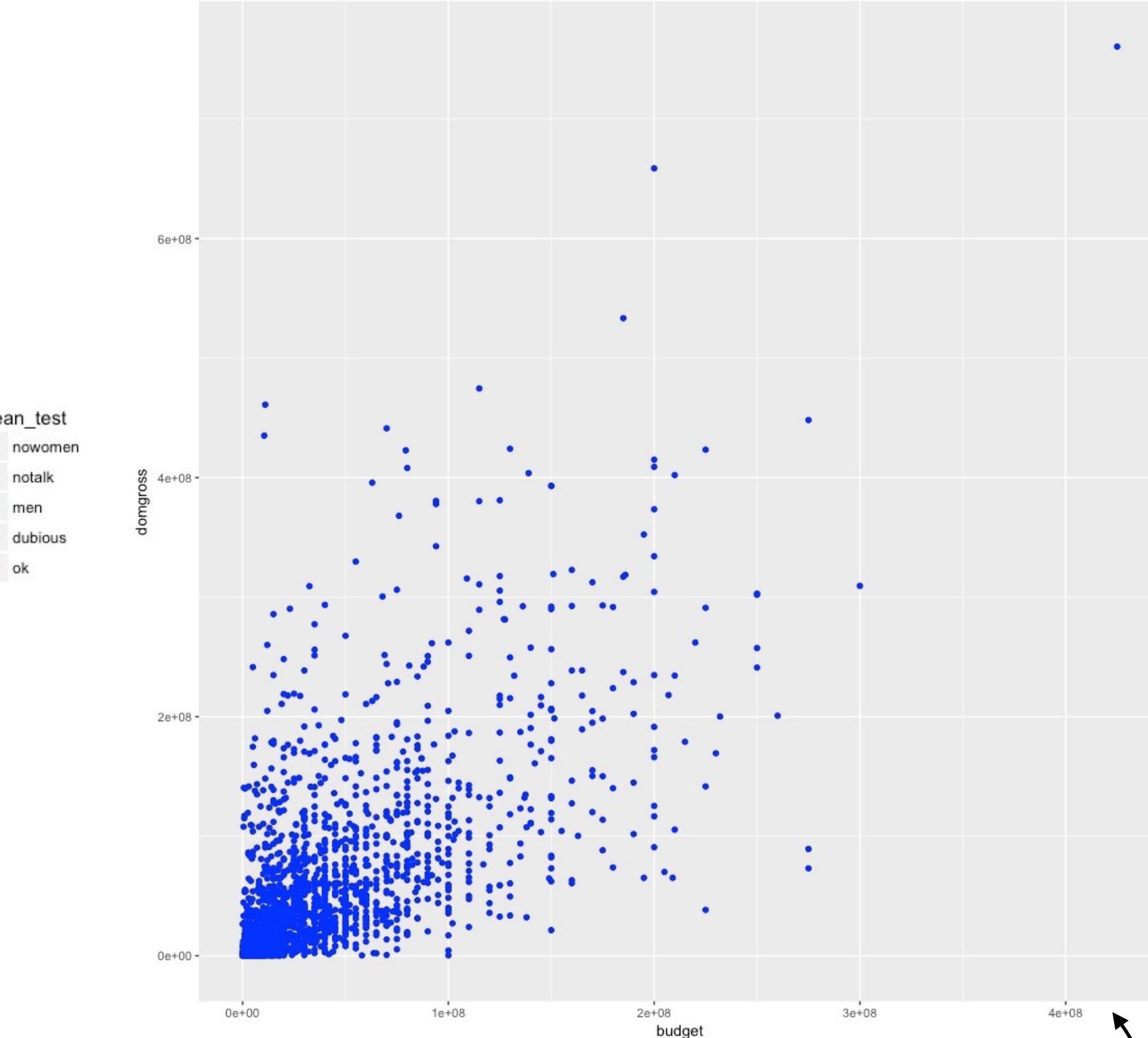
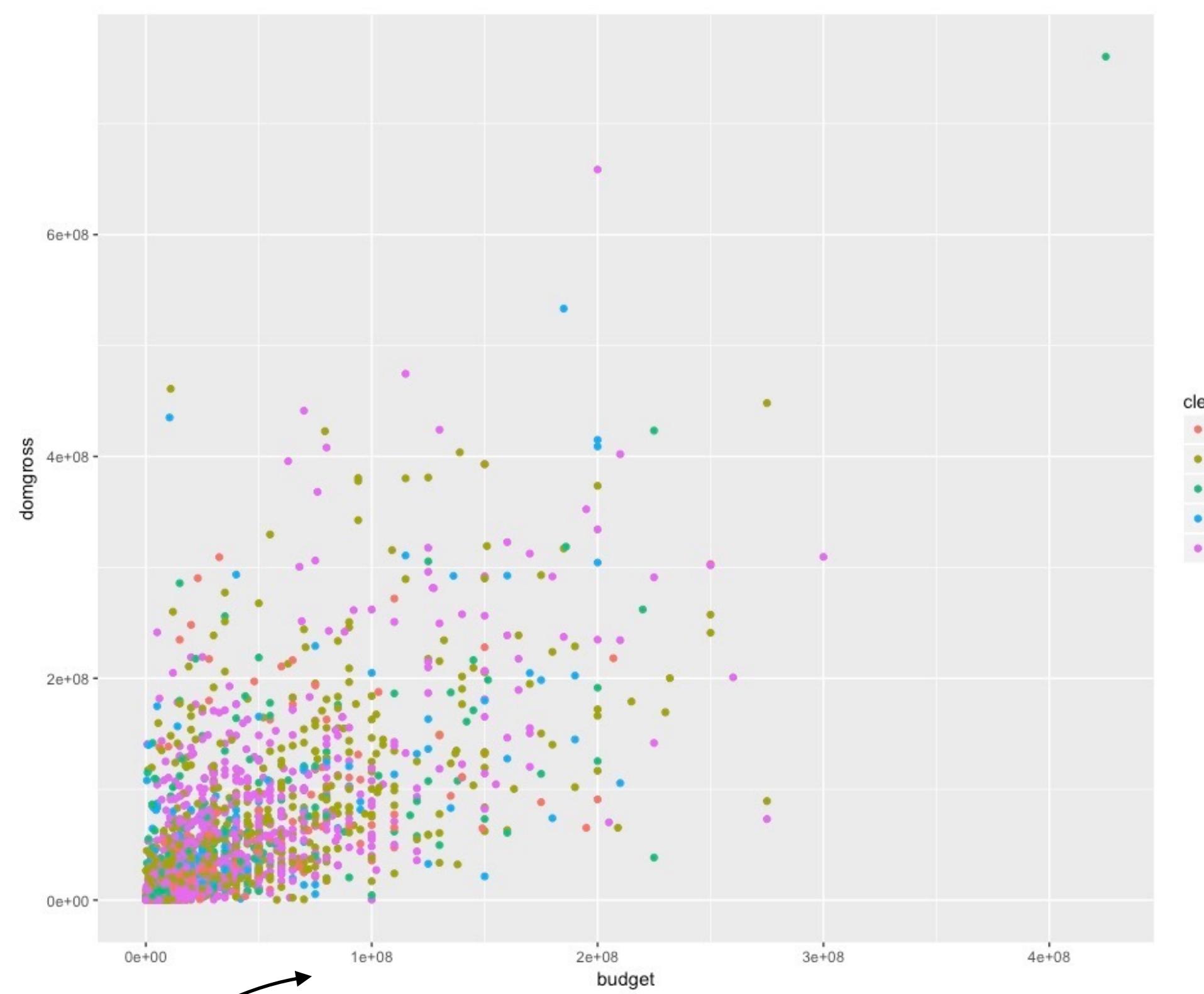
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

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



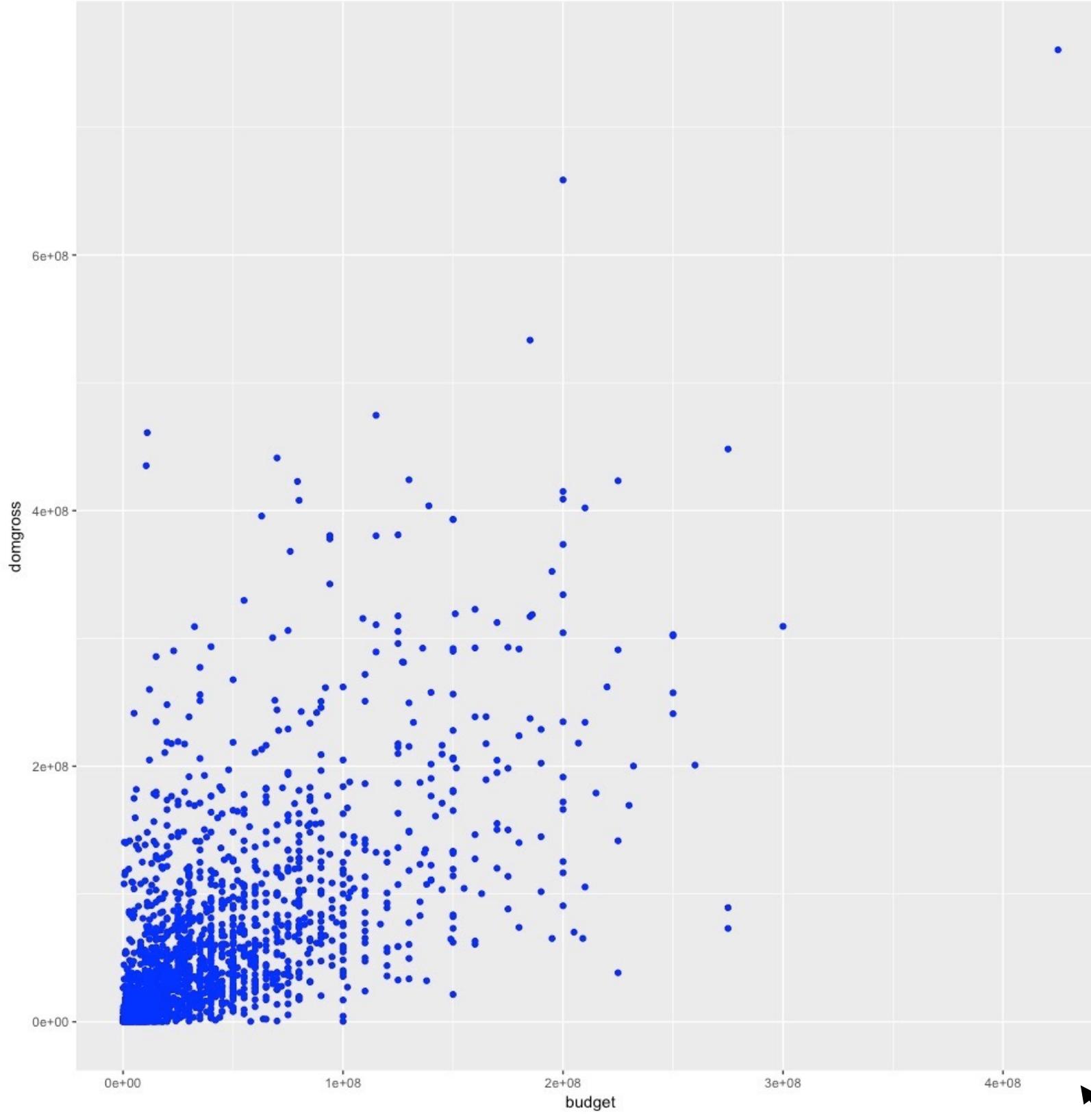
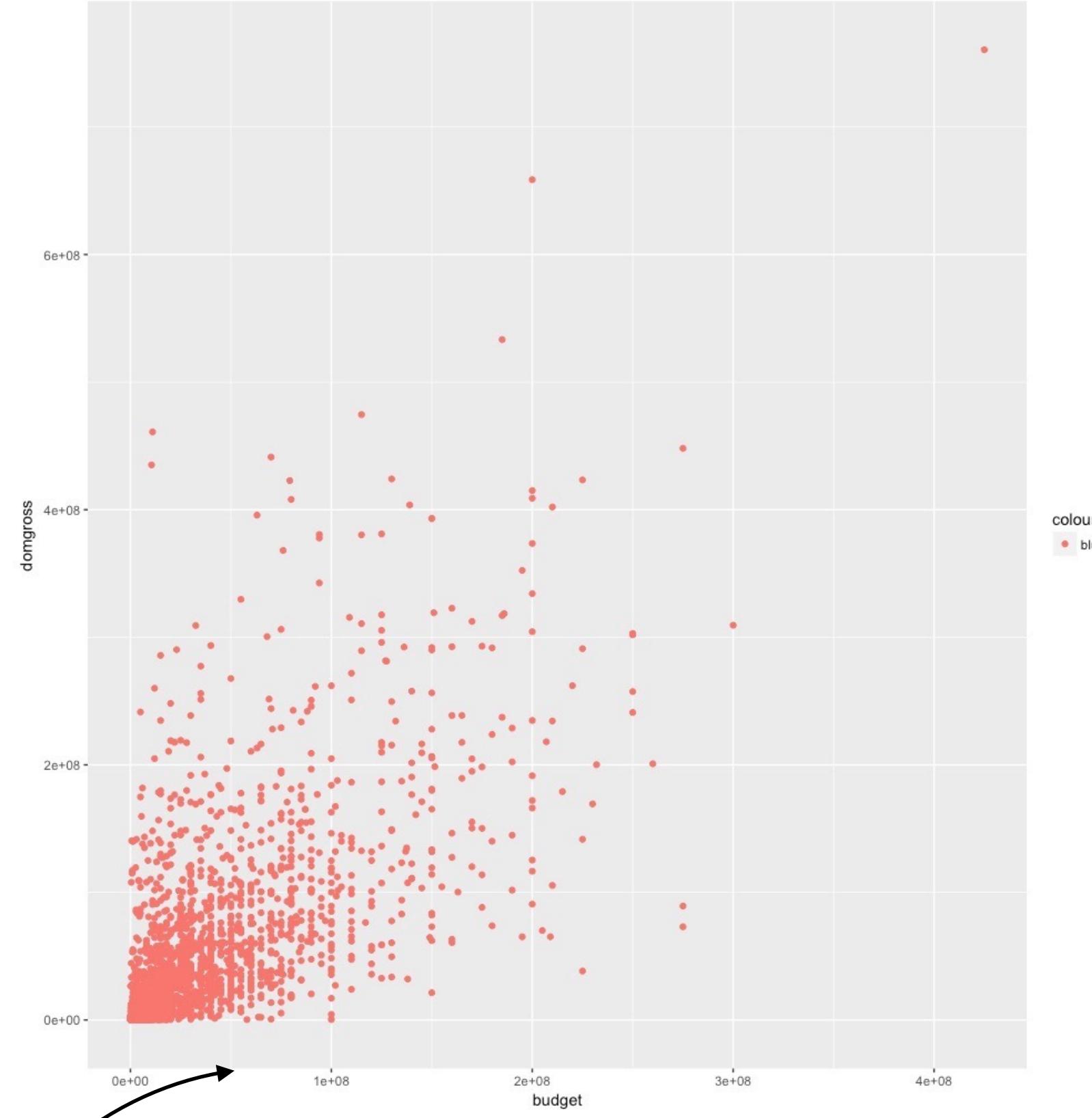
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
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")
```

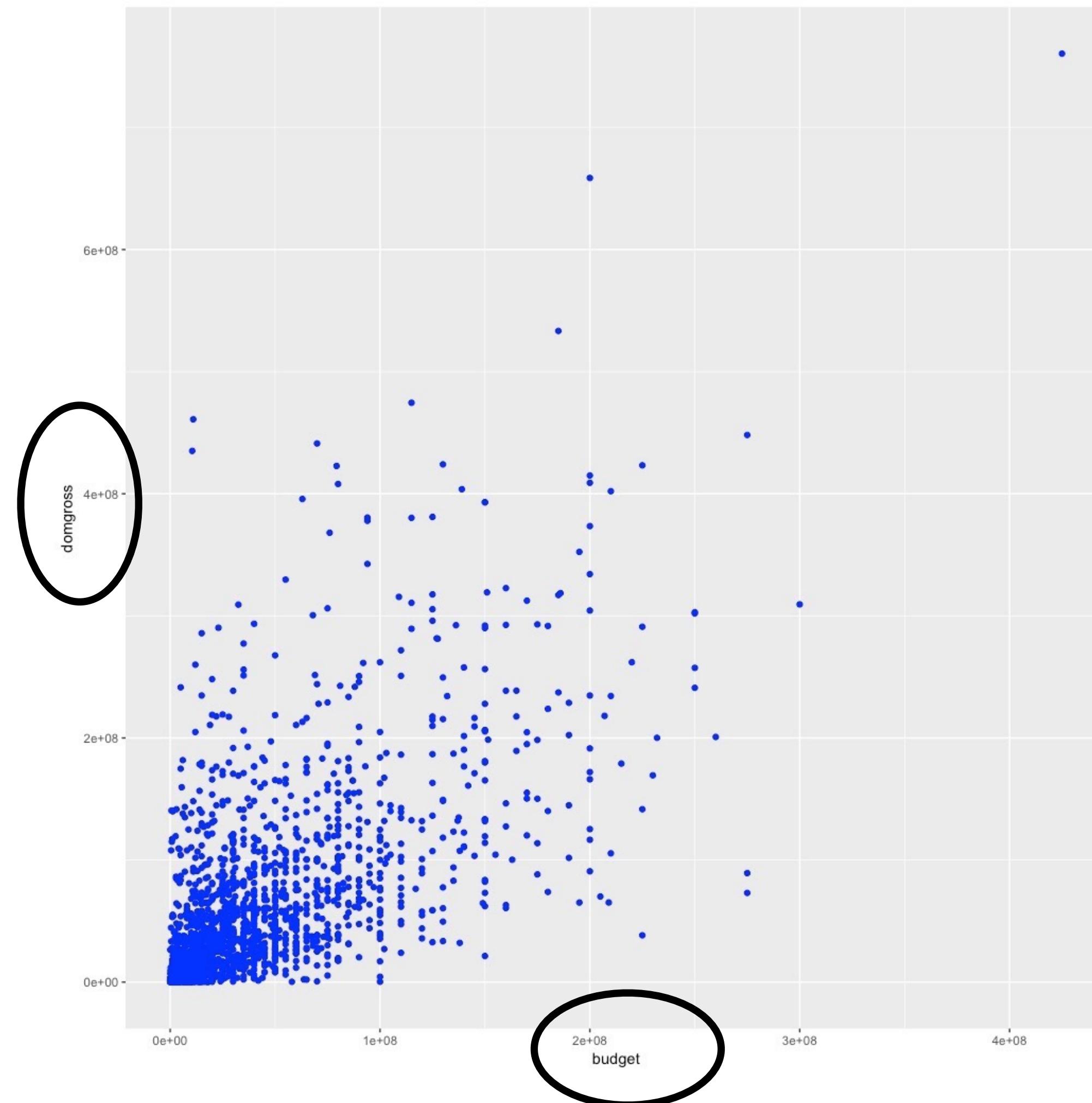


```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color="blue"))
```

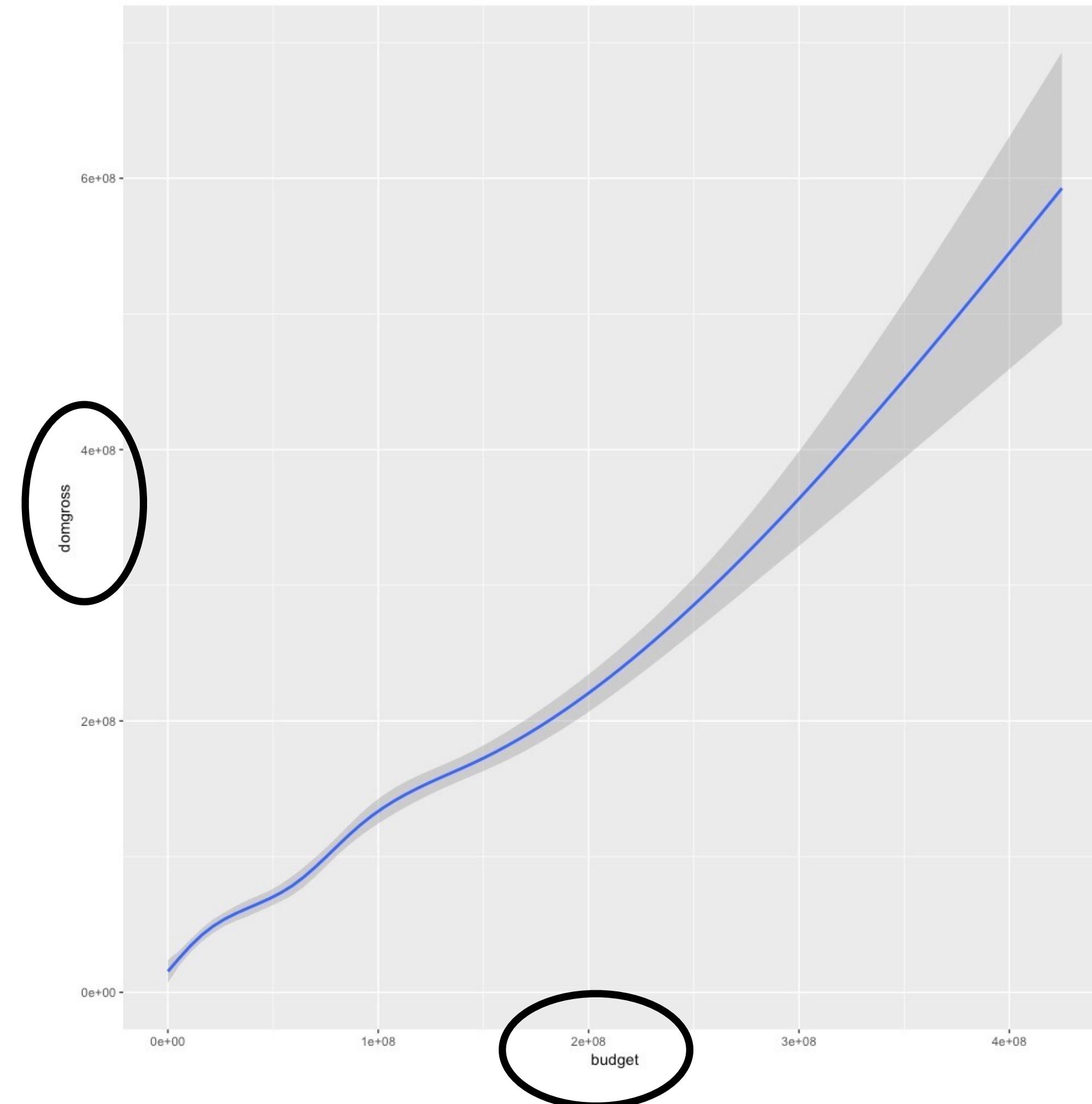
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

geoms

How are these plots similar?

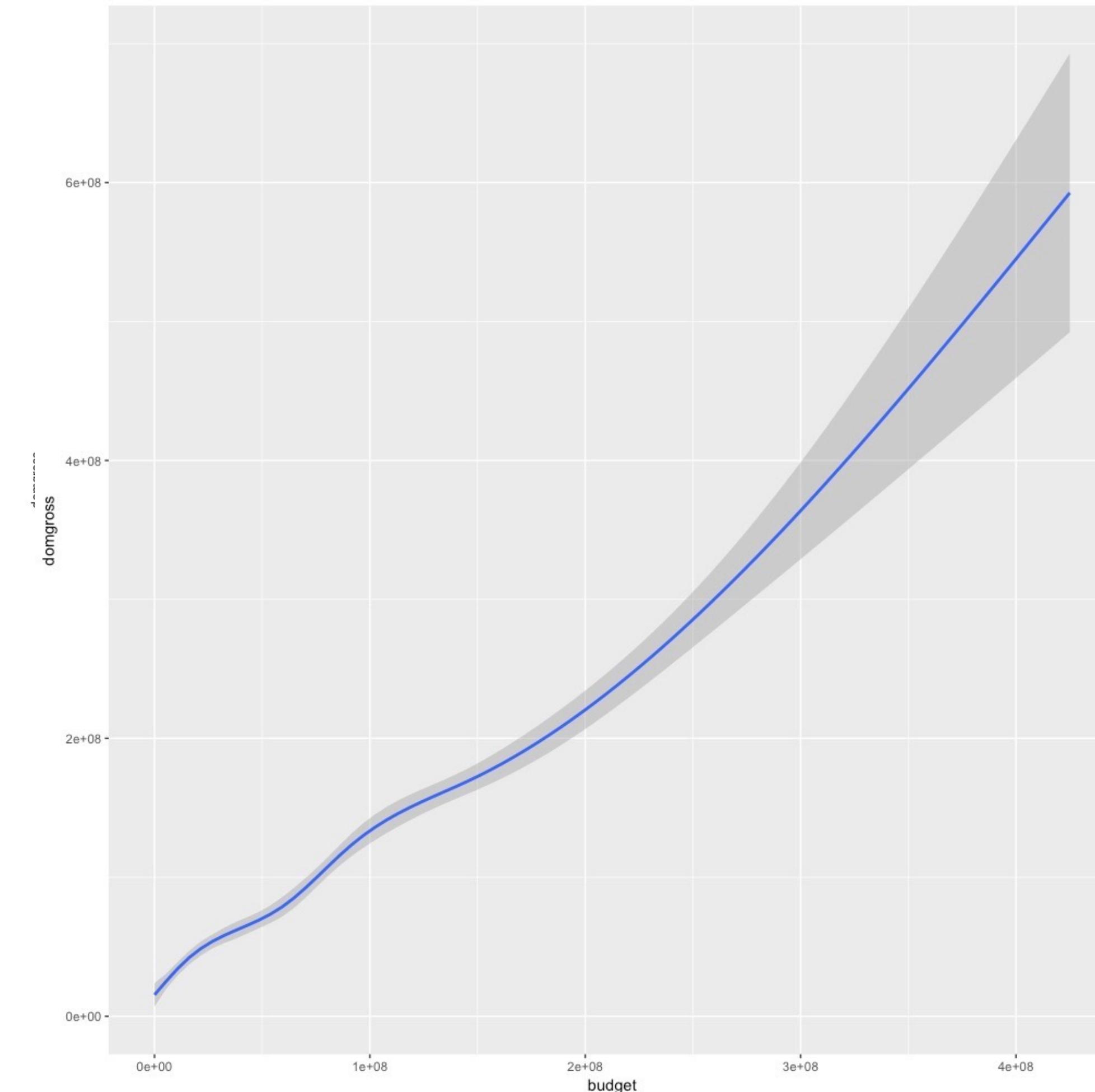
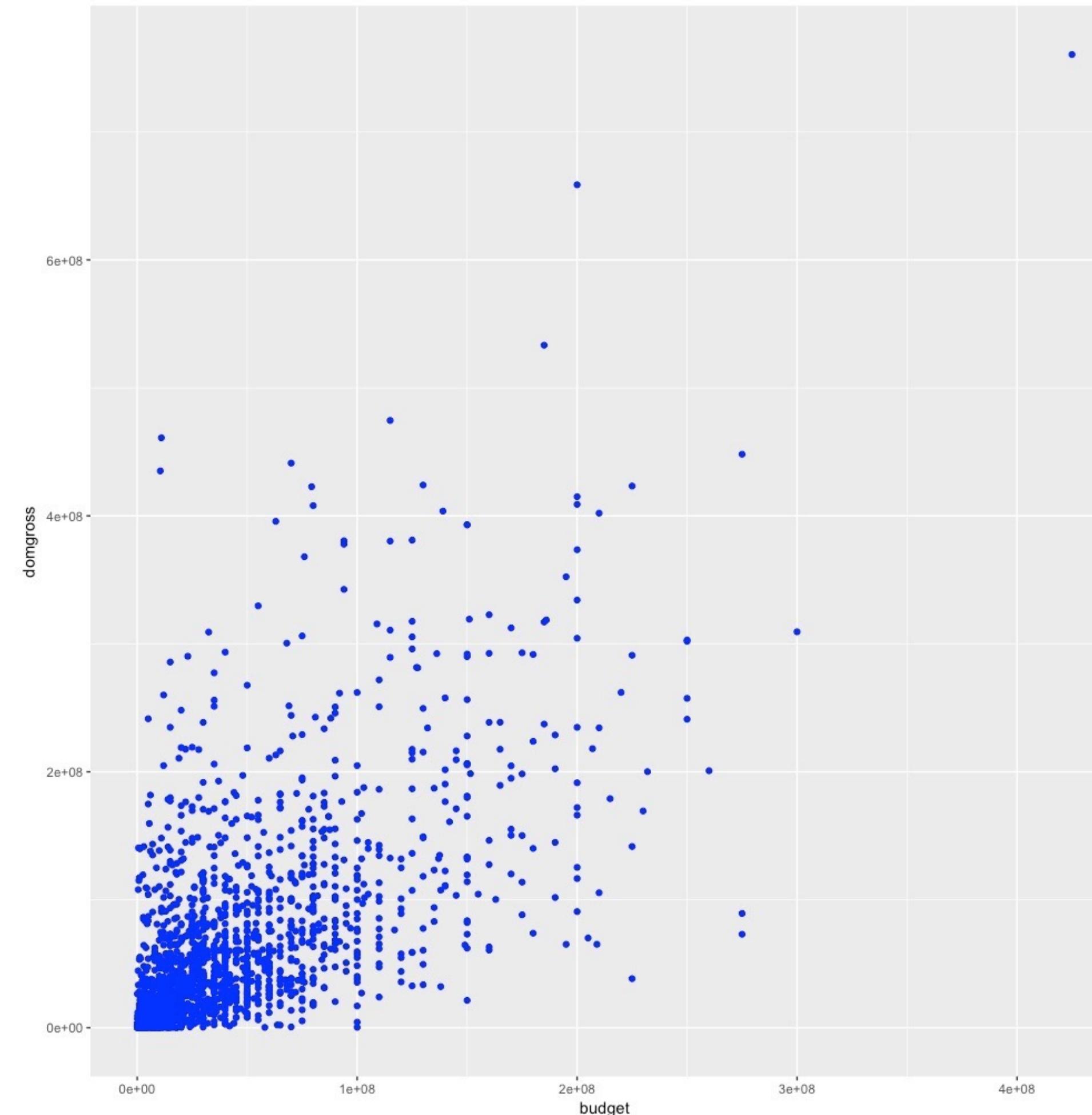


Same: x var, y var, data



How are these plots  
different?

Different: geometric object (geom),  
i.e. 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
- 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

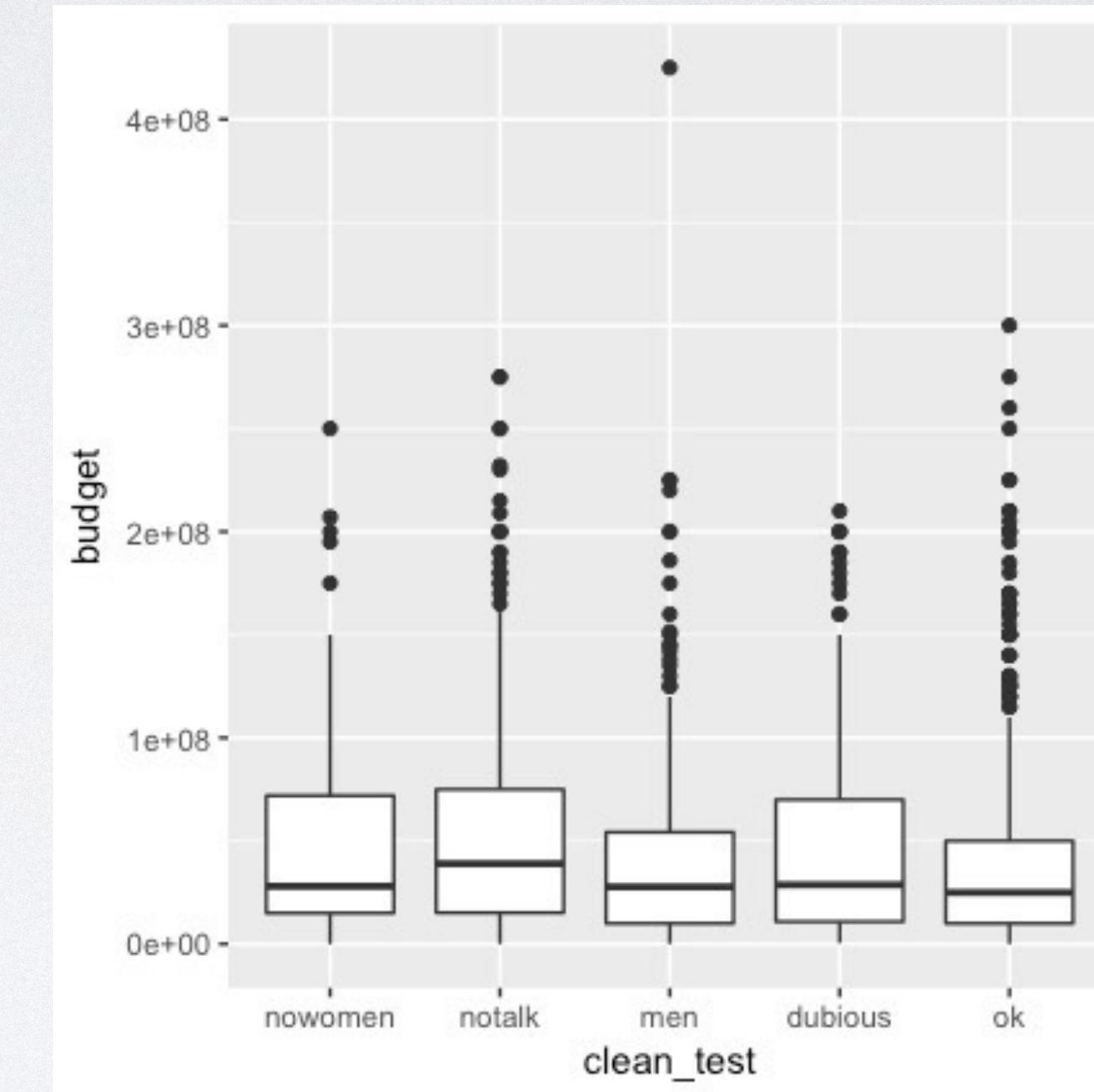
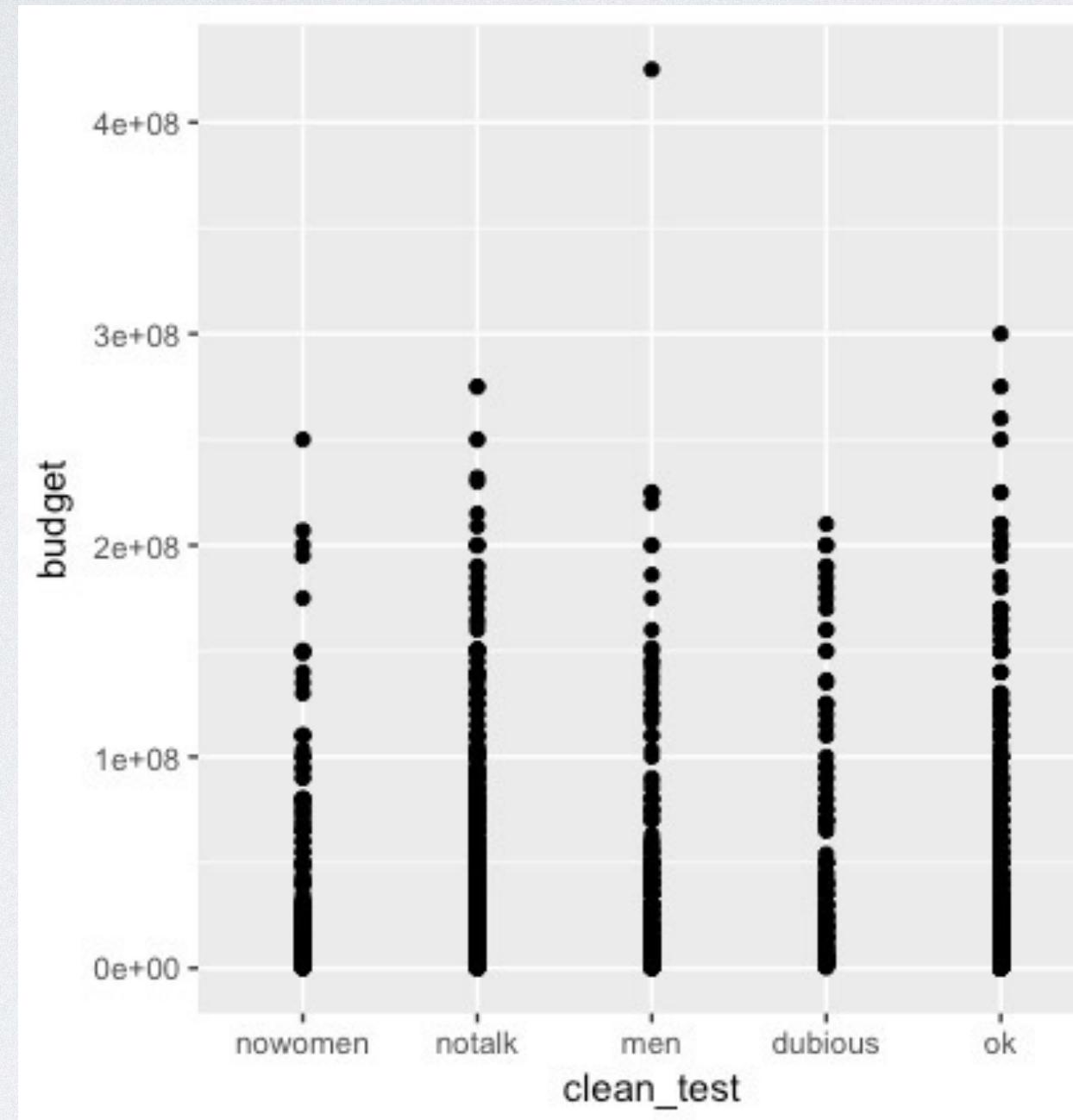
# Your Turn

Pair up.

00 : 30

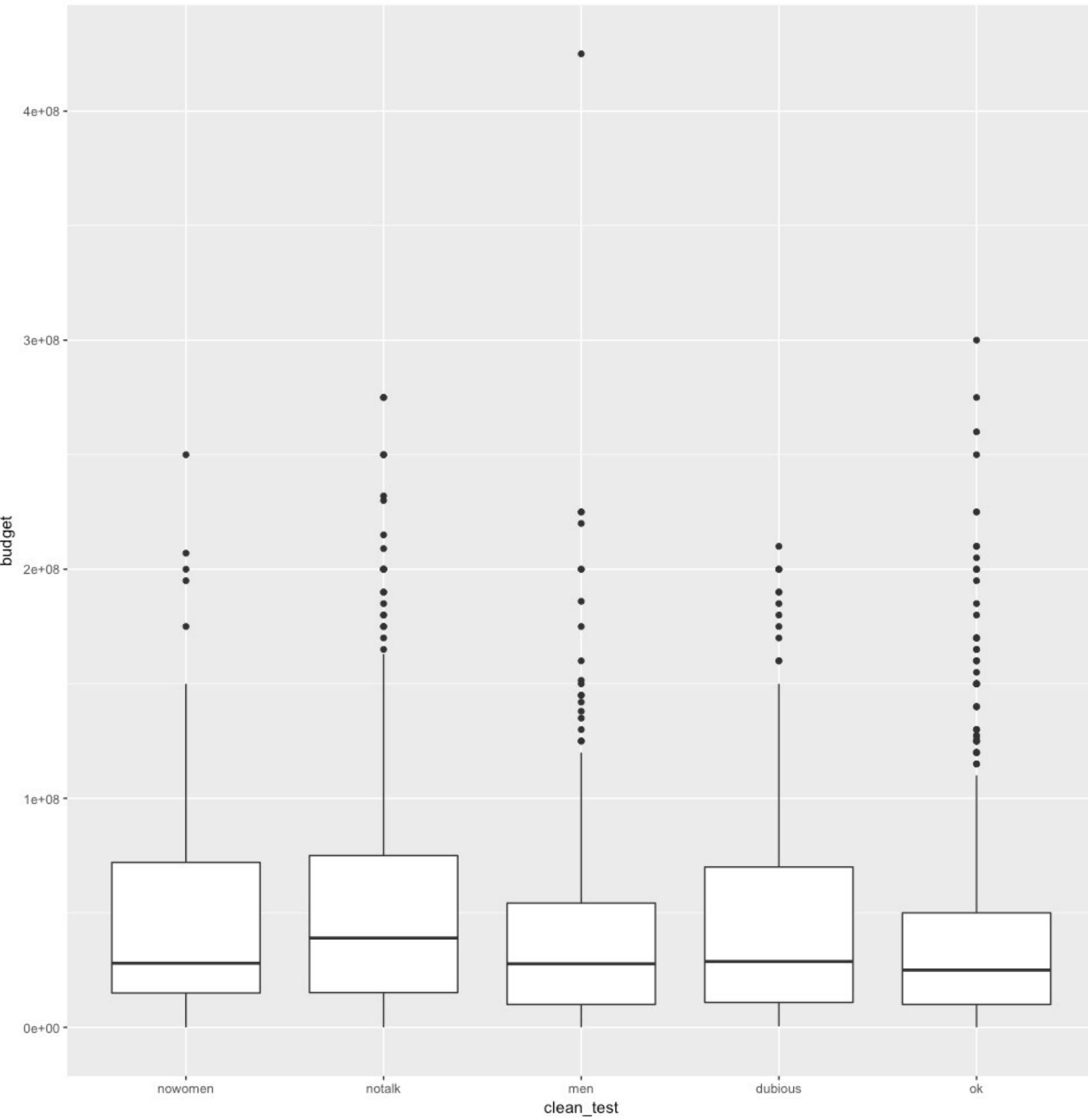
# Your Turn 3

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))
```

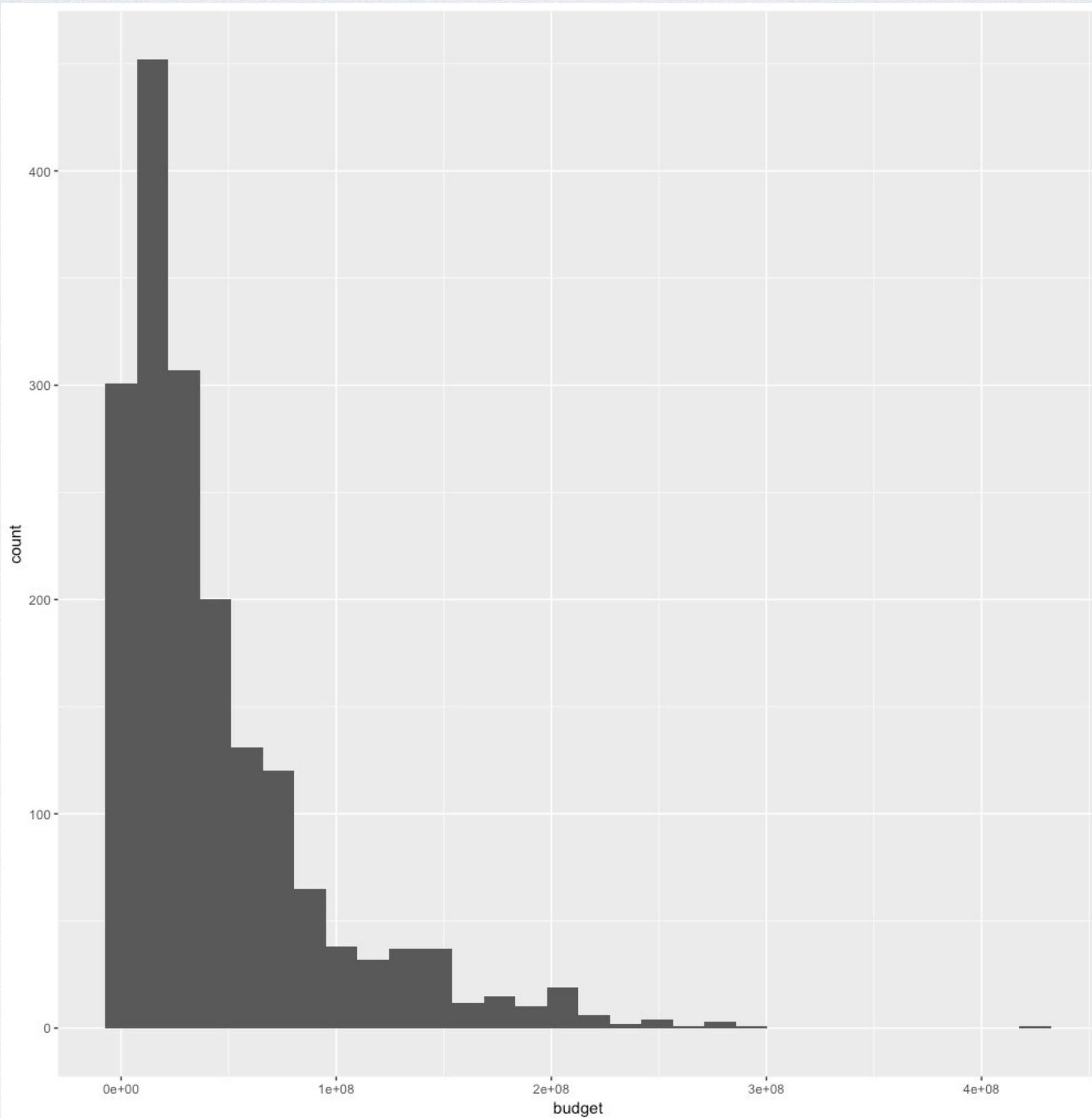




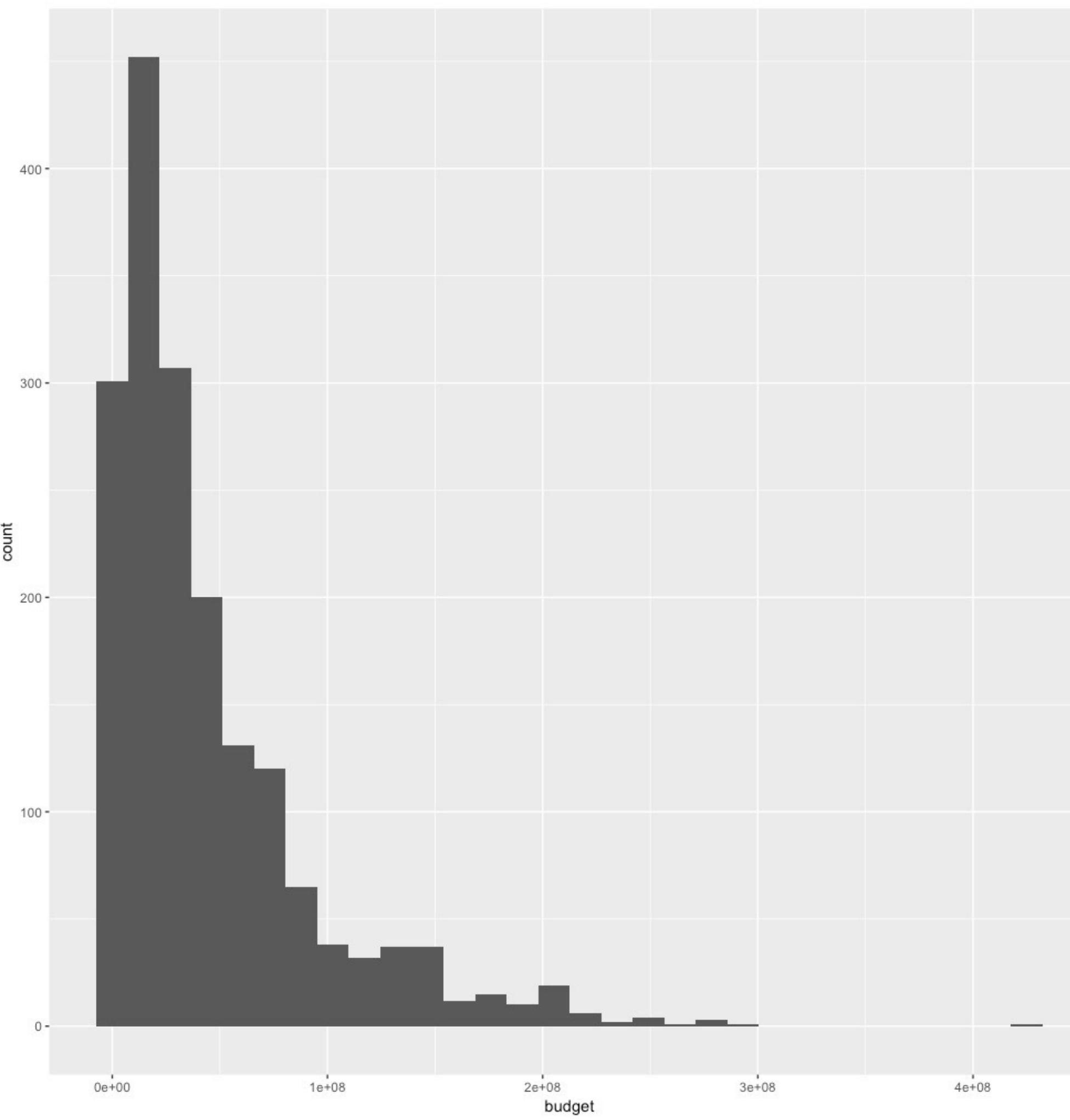
```
ggplot(data = bechdel) +  
  geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

# Your Turn 4

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

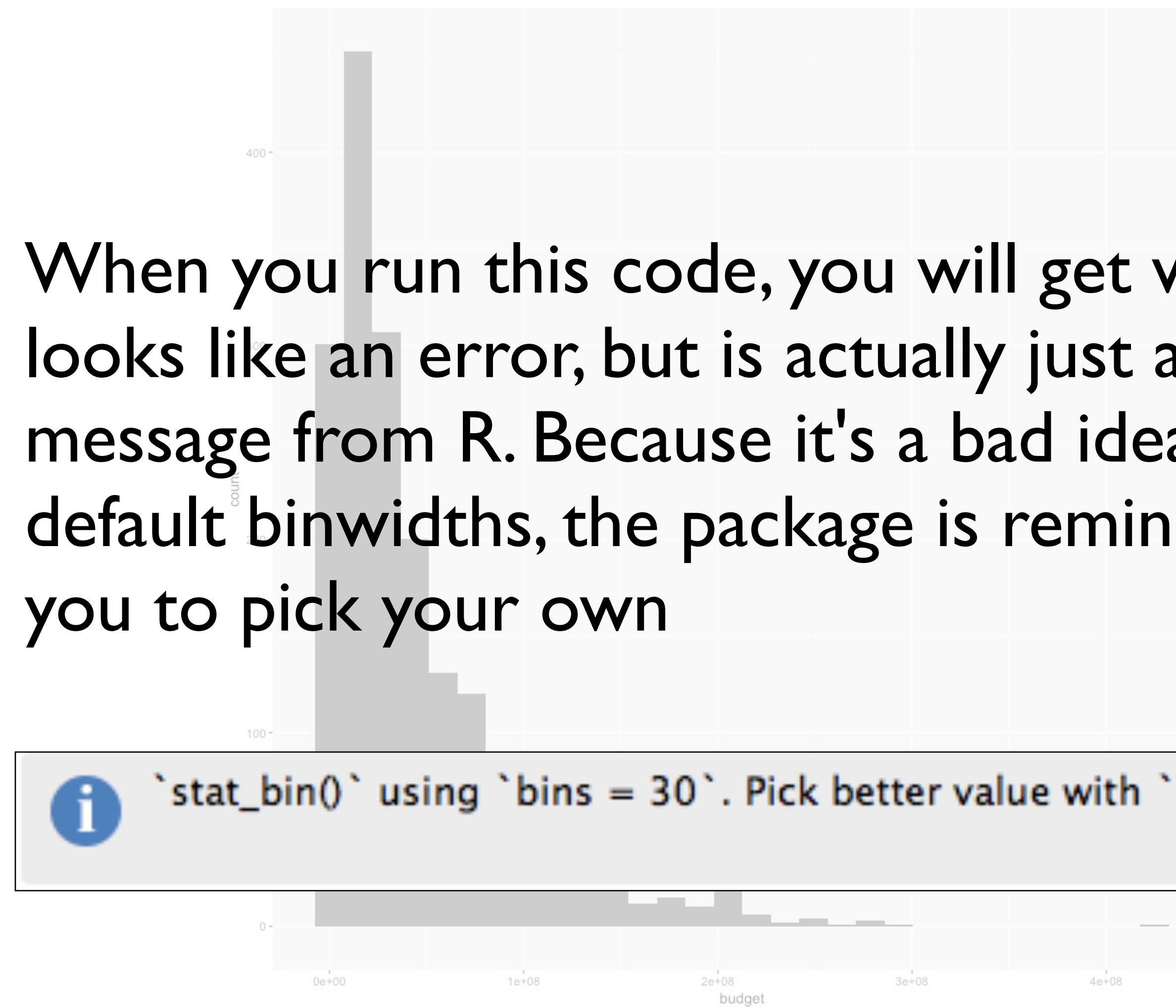


02 : 00



```
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



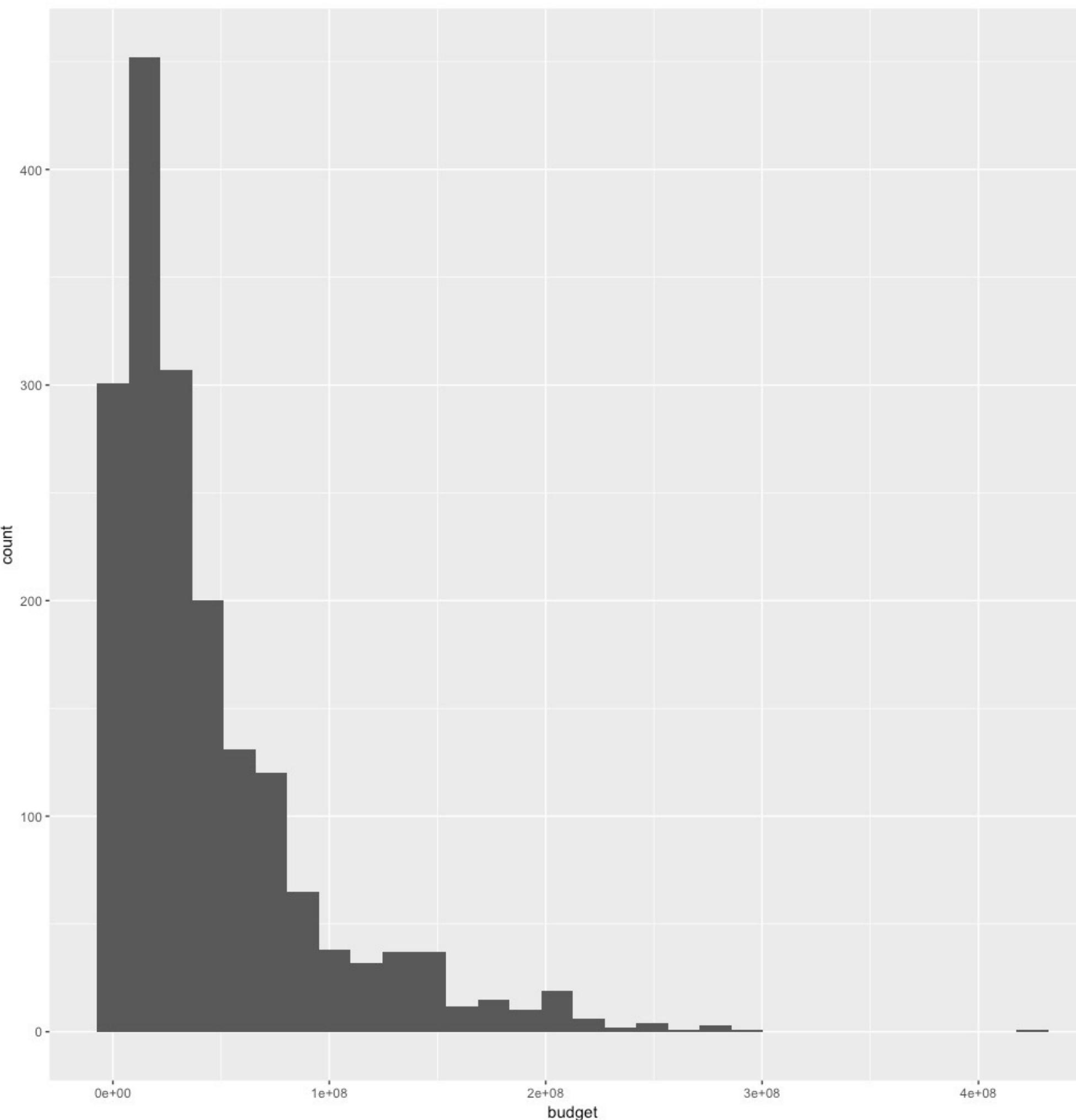
```
ggplot(data = bechdel) +  
  geom_histogram(mapping = aes(x = budget))
```

# Your Turn 5

What would be a reasonable  
binwidth for budget?

Try it out

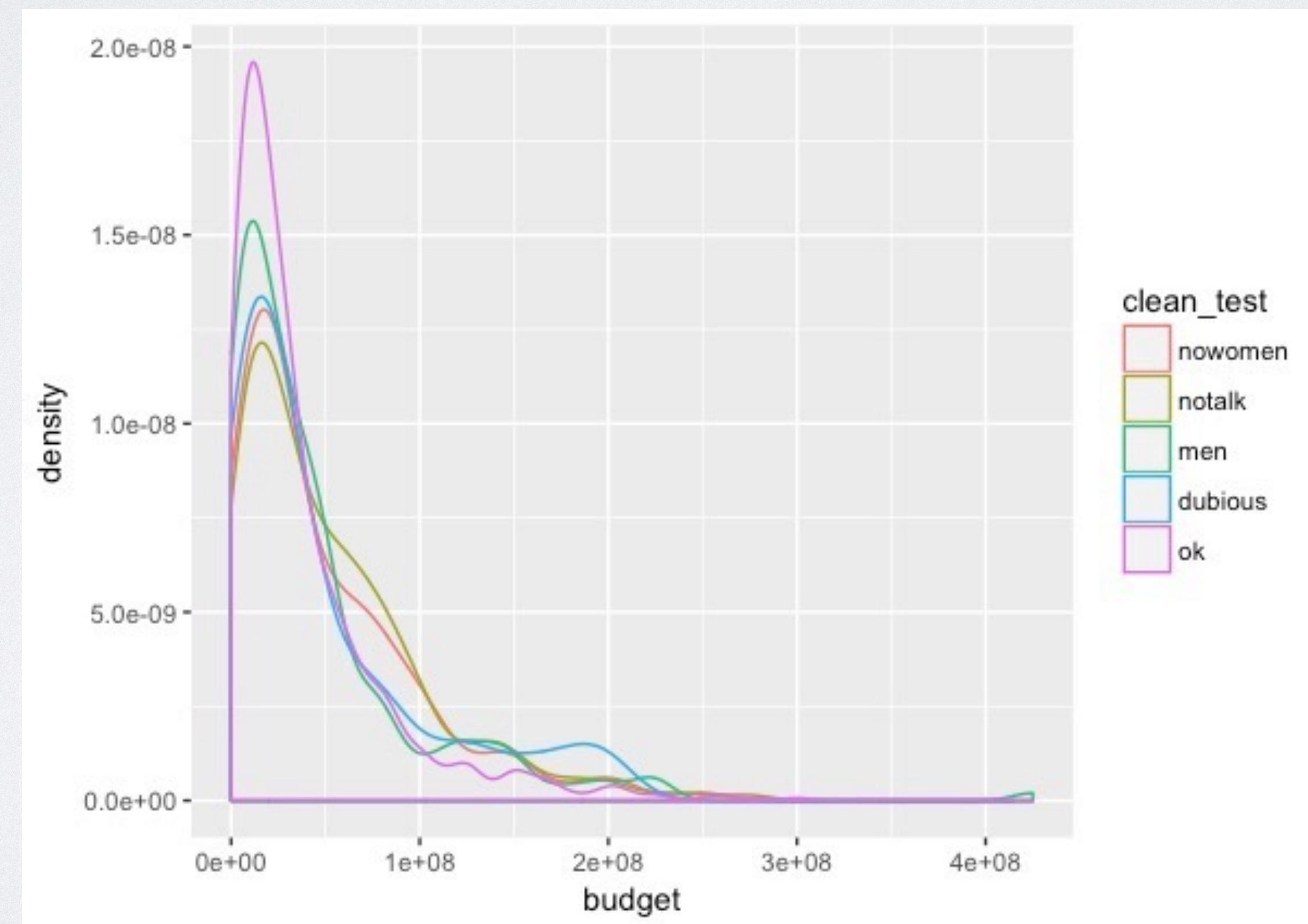
00 : 30



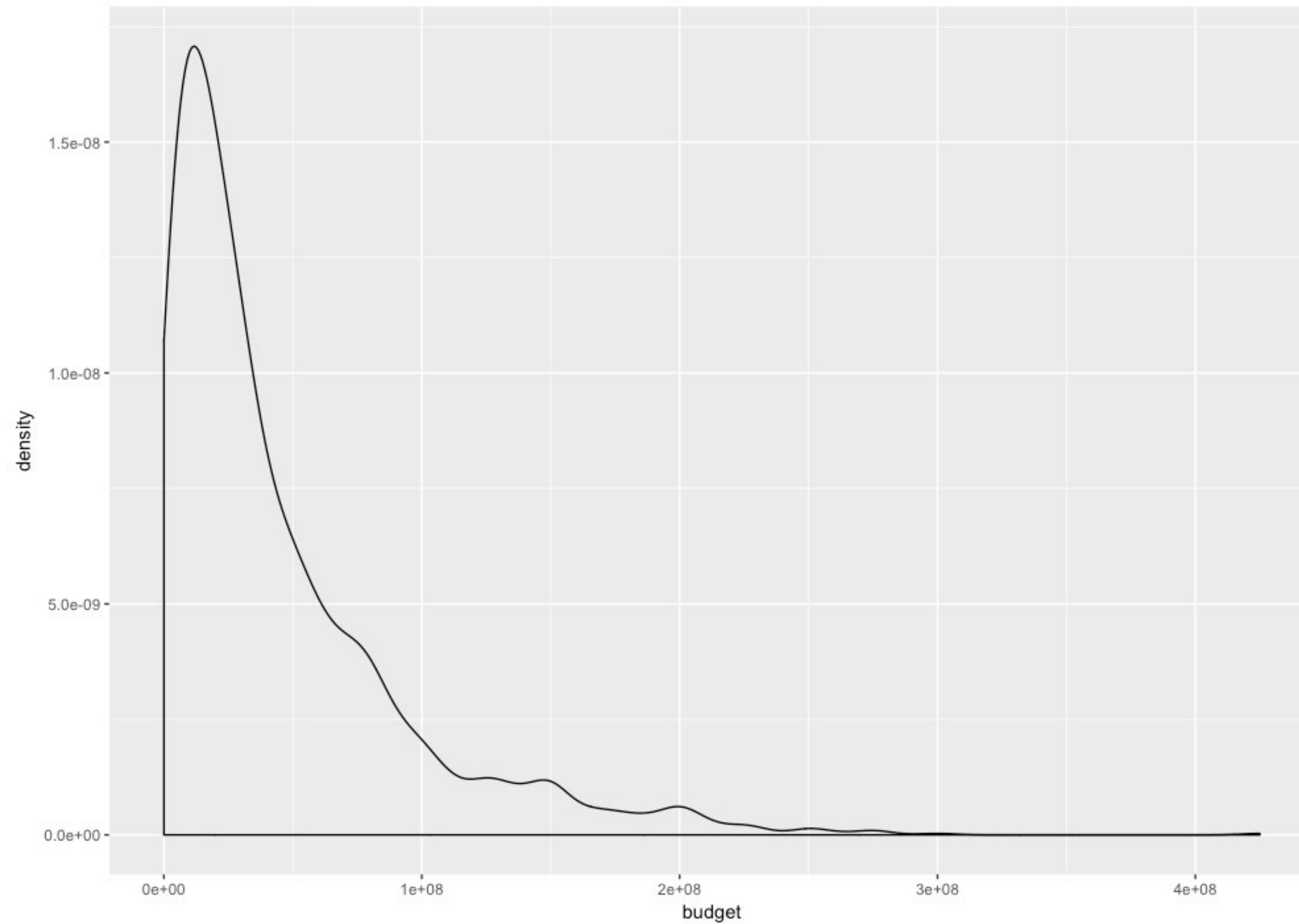
```
ggplot(data = bechdel) +  
  geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```

# Your Turn 6

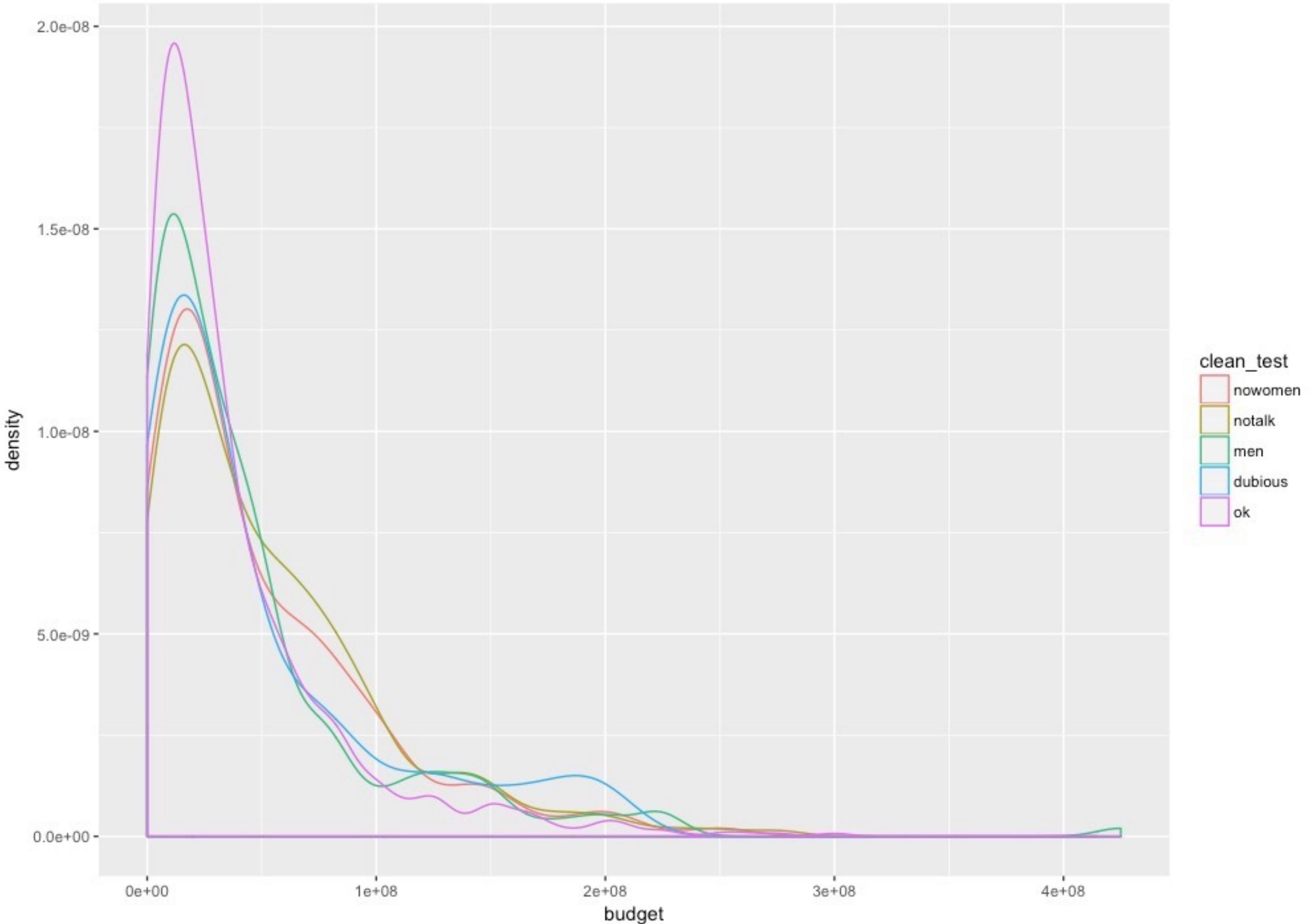
With your partner, make the density plot of **budget** colored by **clean\_test** below. Use the cheatsheet. Try your best guess.



02 : 00



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



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

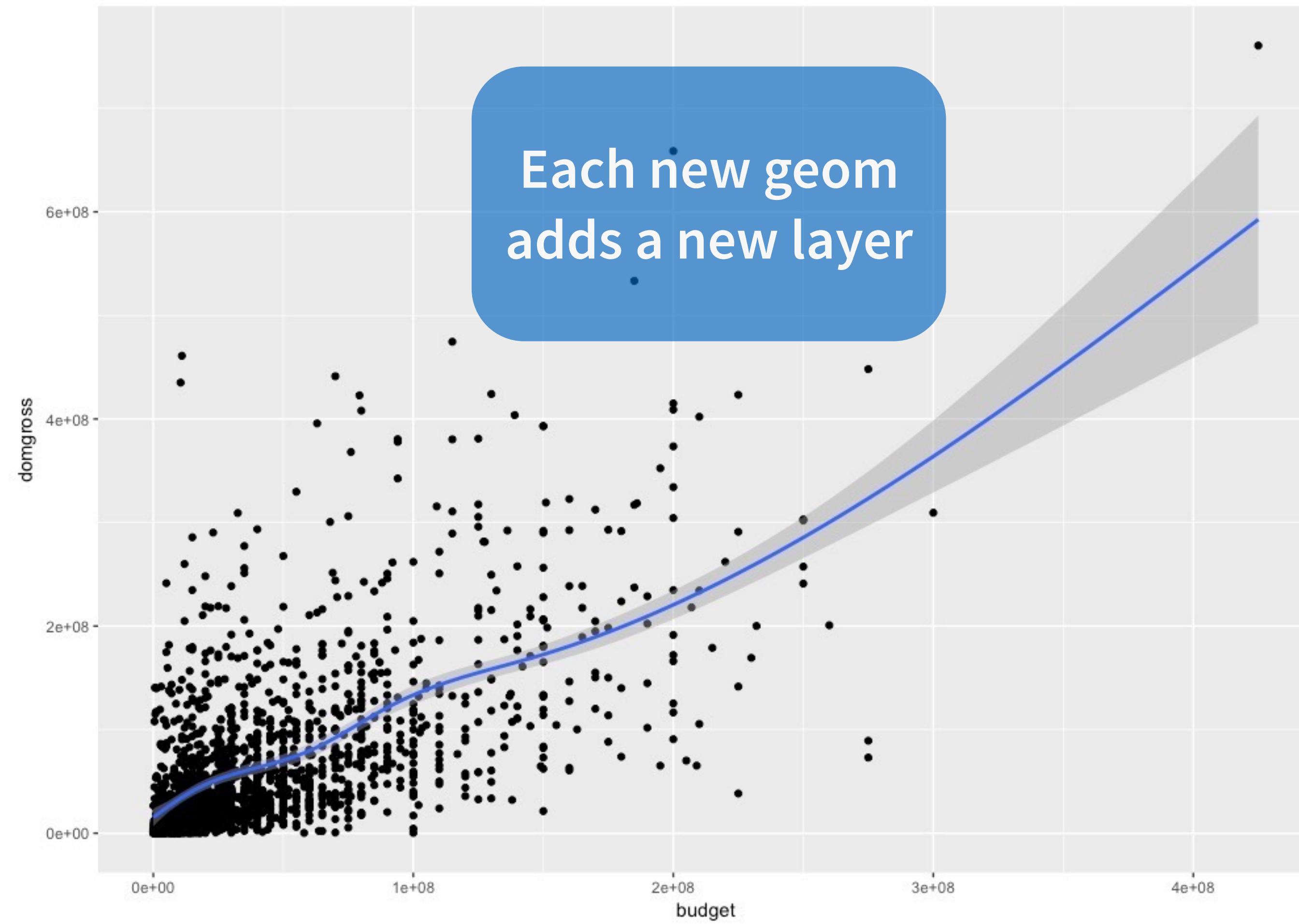
# Your Turn 7

With your partner, 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))
```

# 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 open. The left pane displays the R Markdown 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 ```{r setup}  
18 library(ggplot2)  
19 library(fivethirtyeight)  
20 ```  
21  
22 ## Bechdel test data
```

A red circle highlights the "Preview" button in the toolbar above the code editor. The right pane shows a scatter plot of movie budgets 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 Editor (Left):**

```
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):**

## 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

# Your Turn 8

Locate the 02-Visualization.nb.html file in  
your Files pane. It will be located in your  
**working directory**



# Working directory

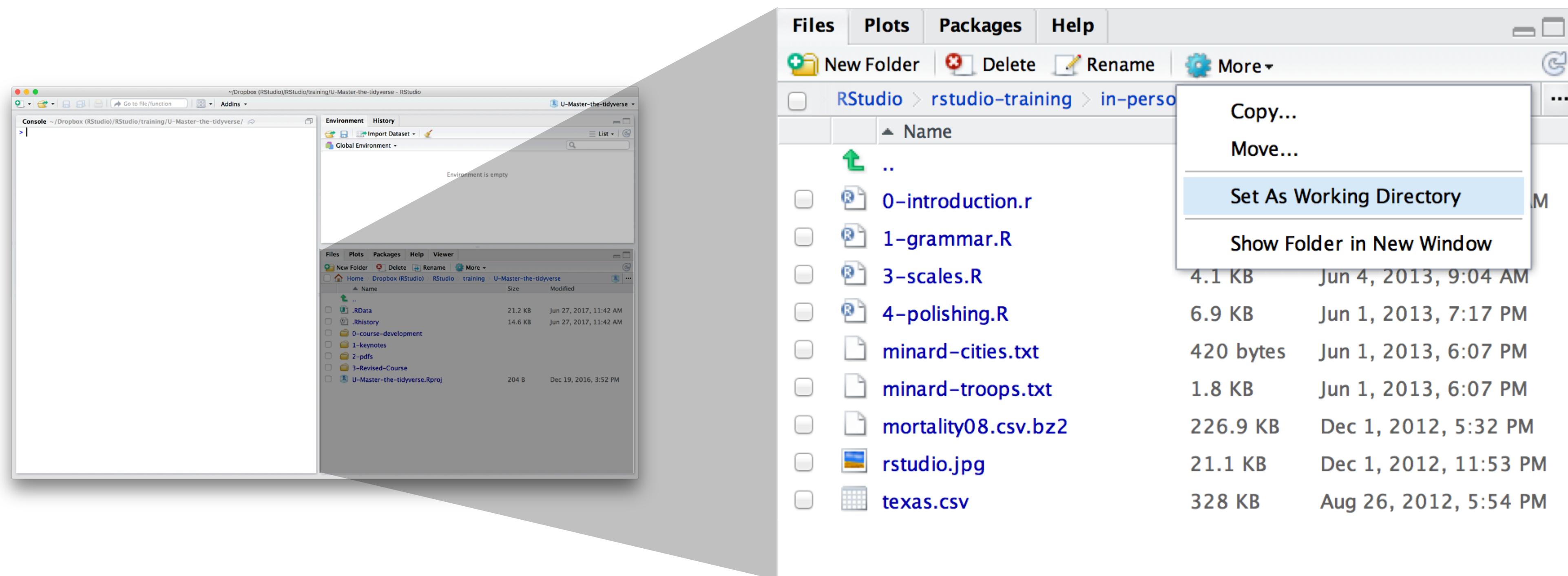
R associates itself with a folder (i.e. directory) on your computer.

- 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

You can check your working directory by running `getwd()`

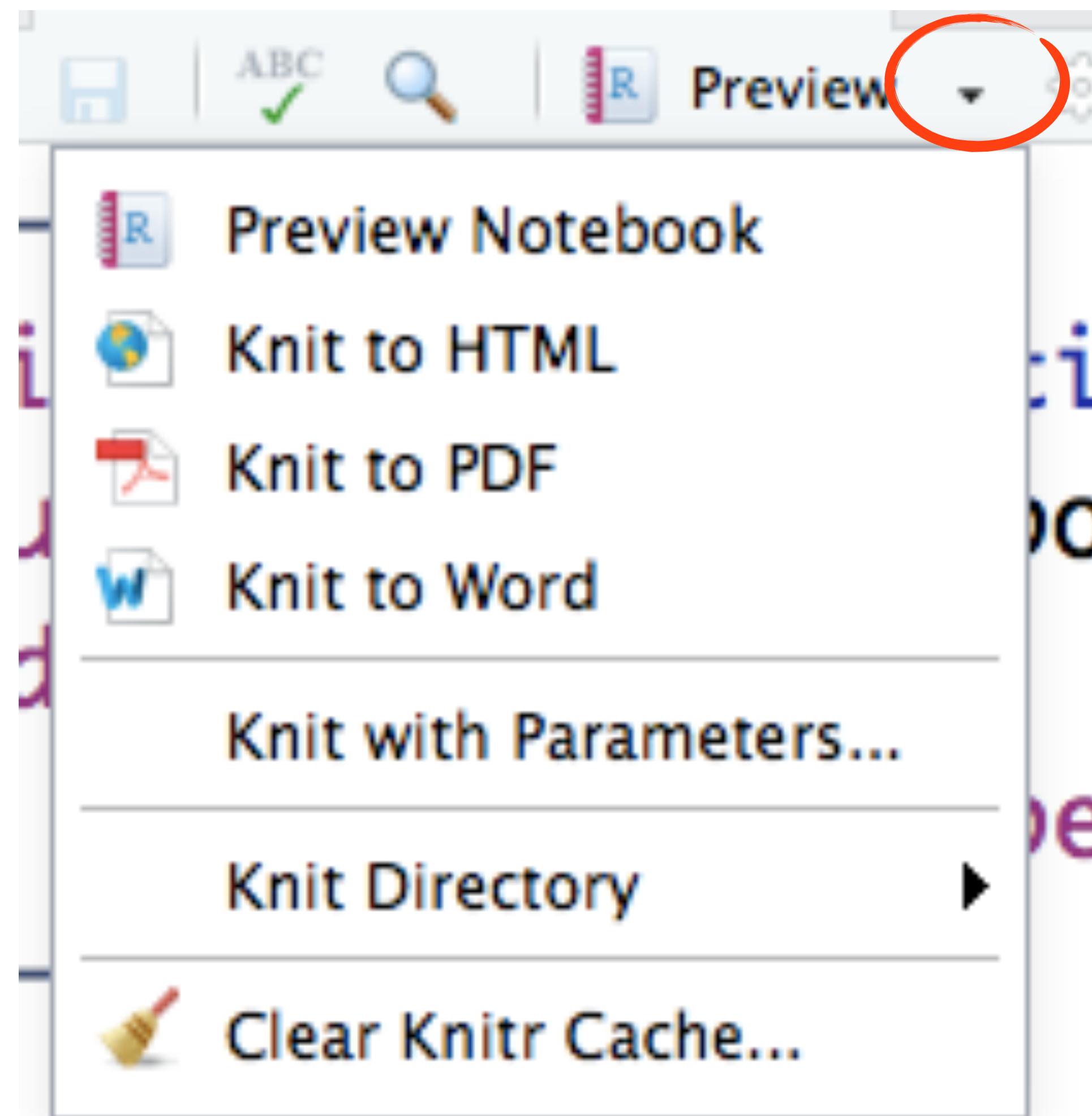
# Changing the Working directory

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



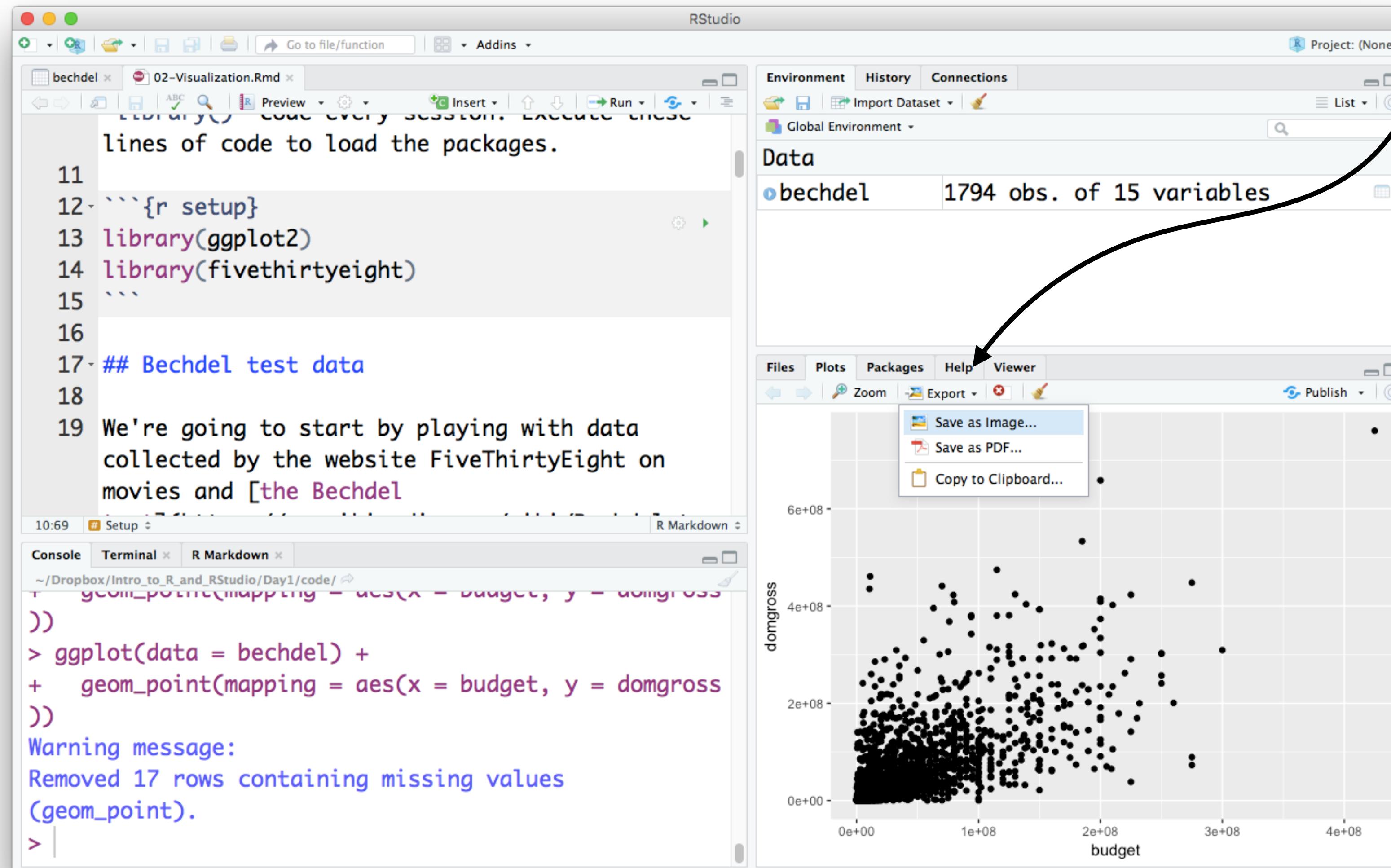
# "knitting" a document

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.



# Manually saving plots

If you just want to save a plot, you can do so for any plot in your Plot pane using the export menu



# Saving plots

`ggsave()` saves the last plot.

Uses size on screen:

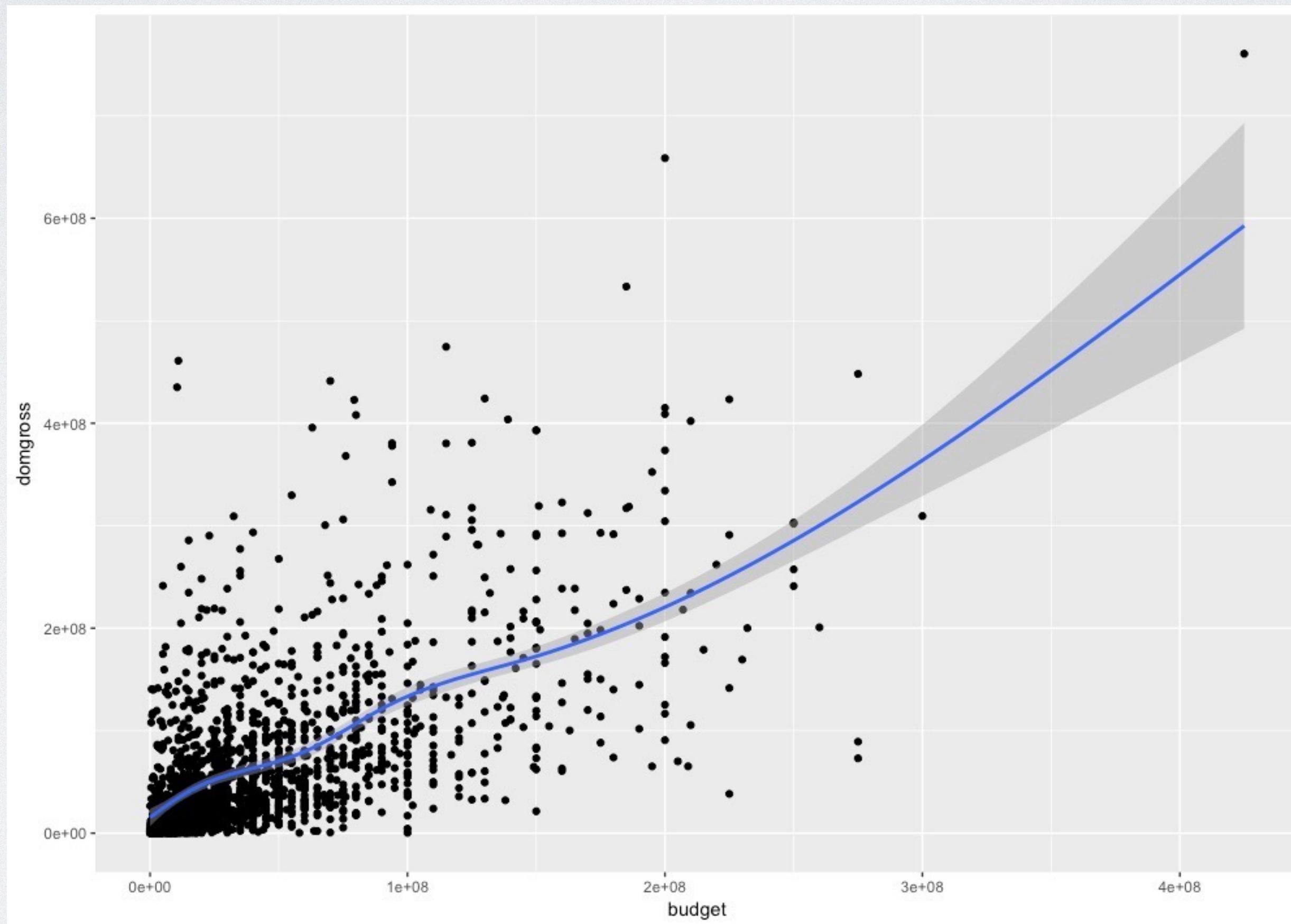
```
ggsave("my-plot.pdf")  
ggsave("my-plot.png")
```

Specify size in inches

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

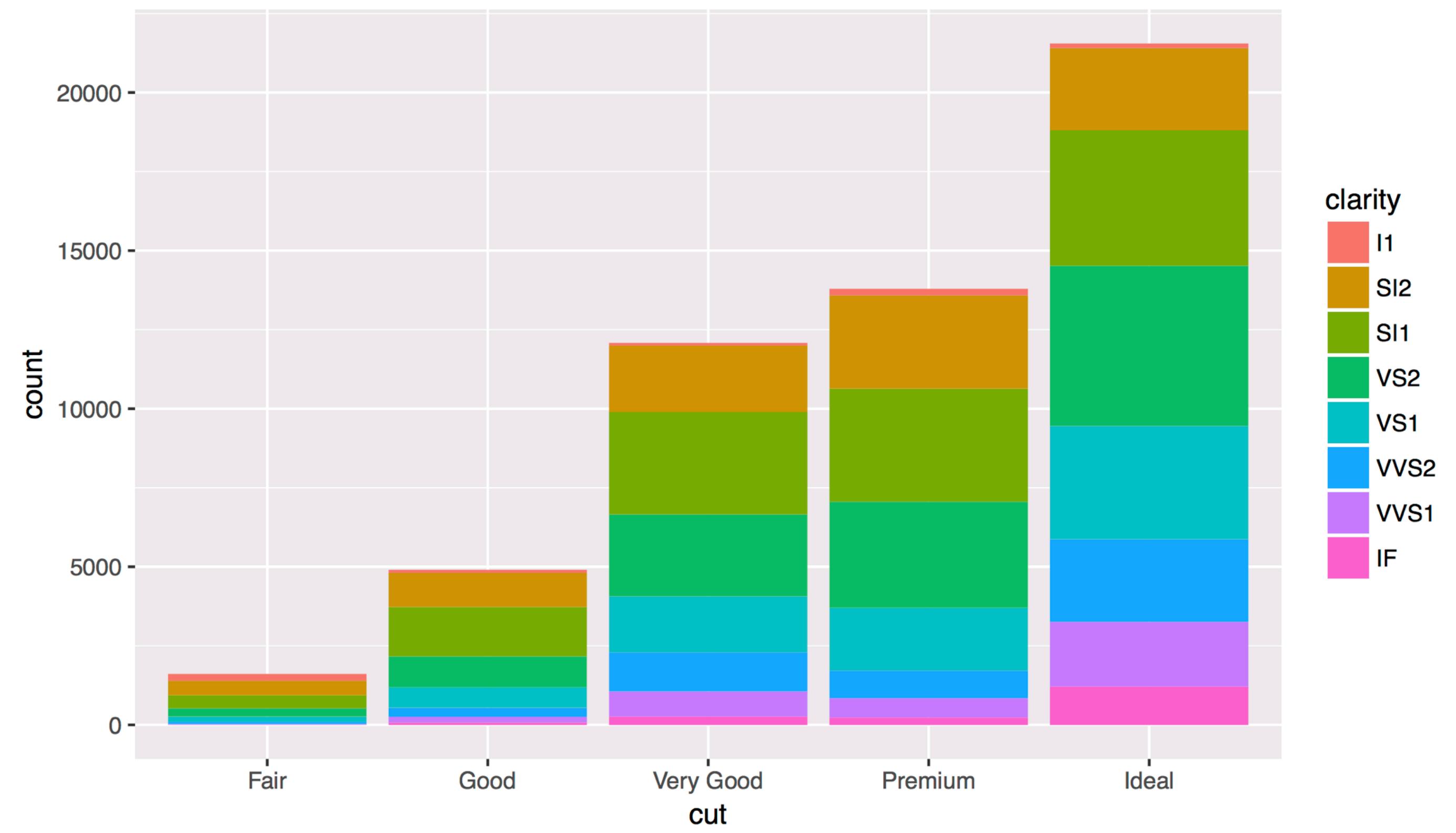
# Your Turn 9

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



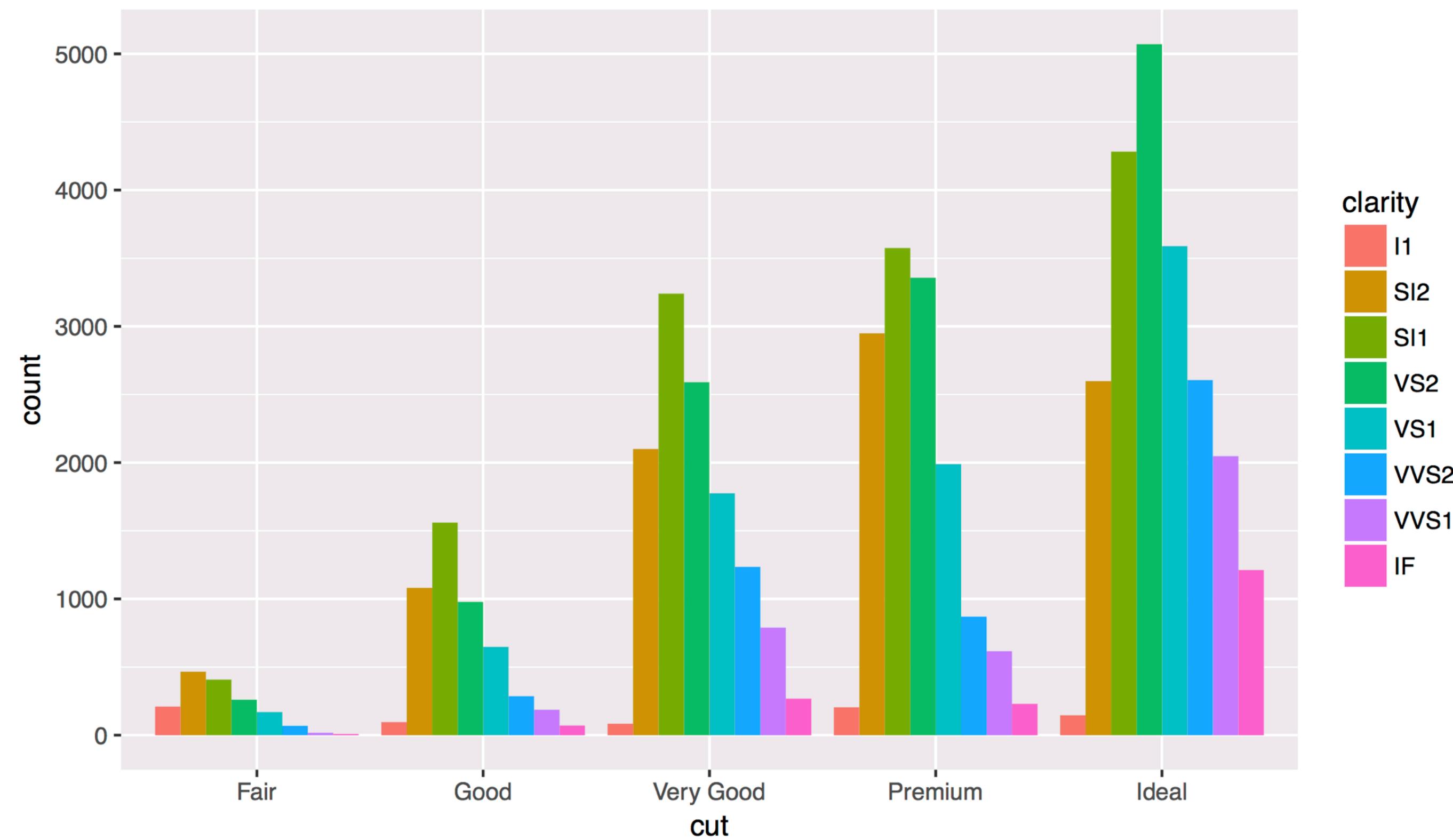
01 : 00

what else?



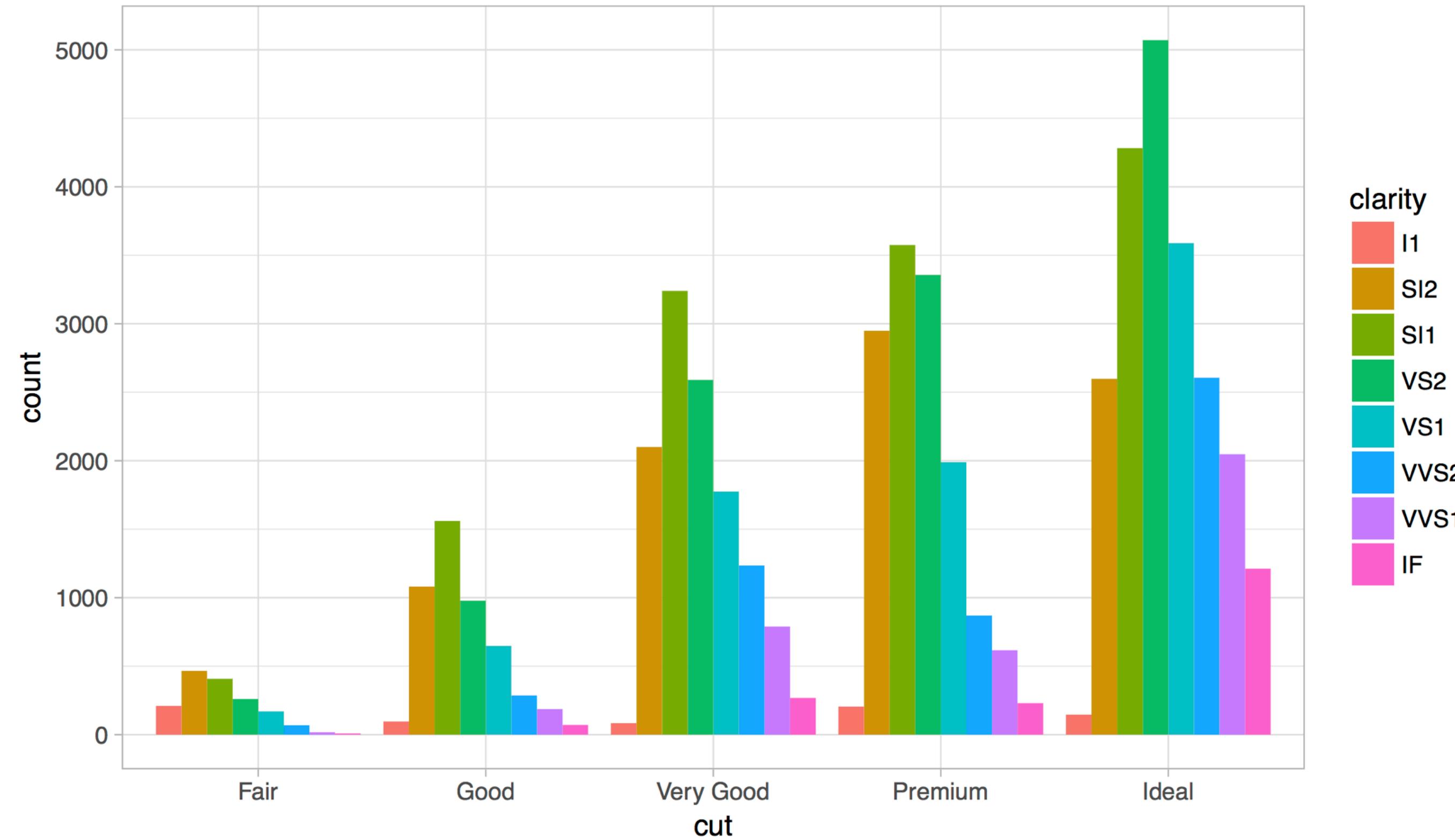
# Position Adjustments

## How overlapping objects are arranged



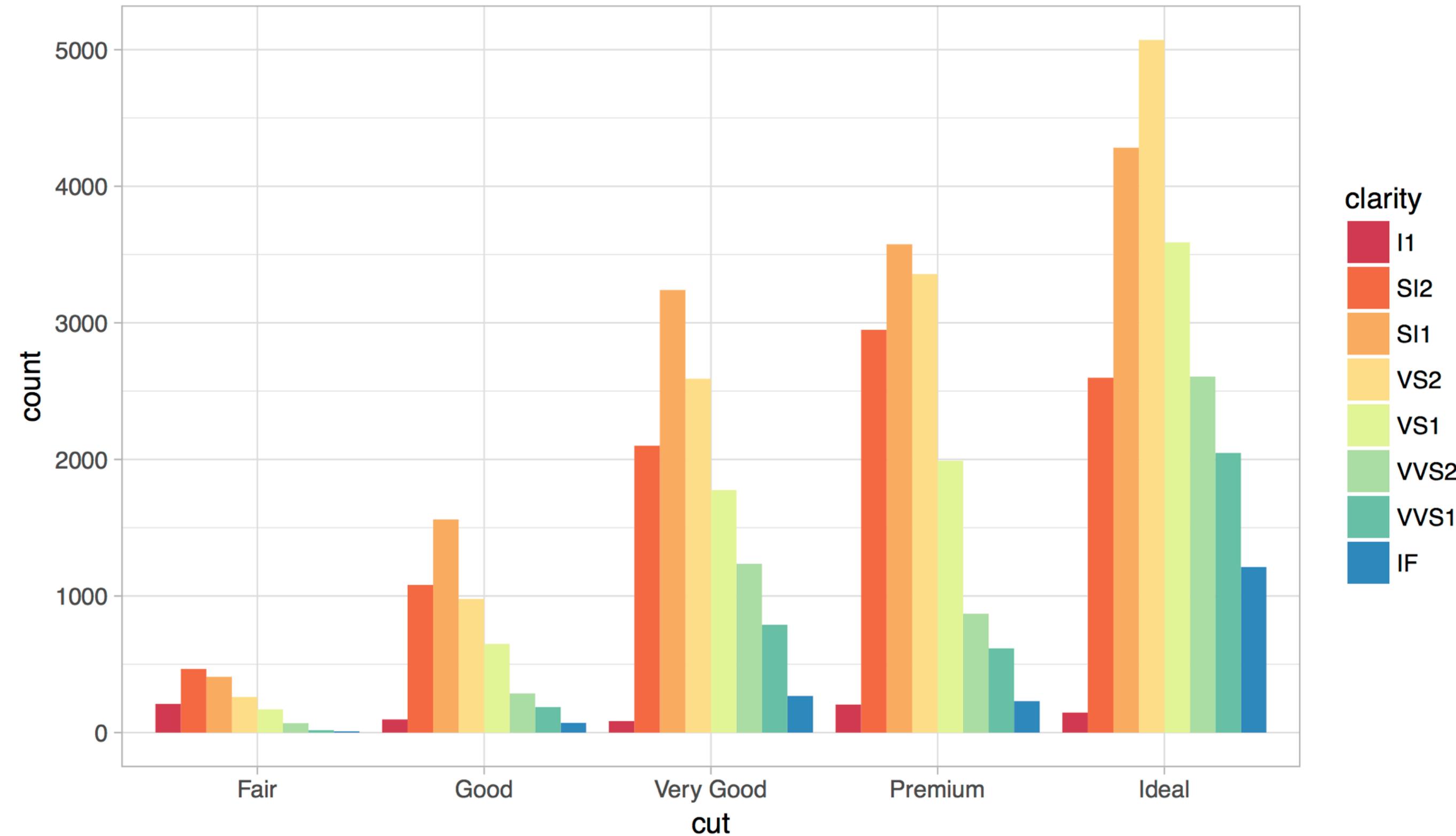
# Themes

## Visual appearance of non-data elements



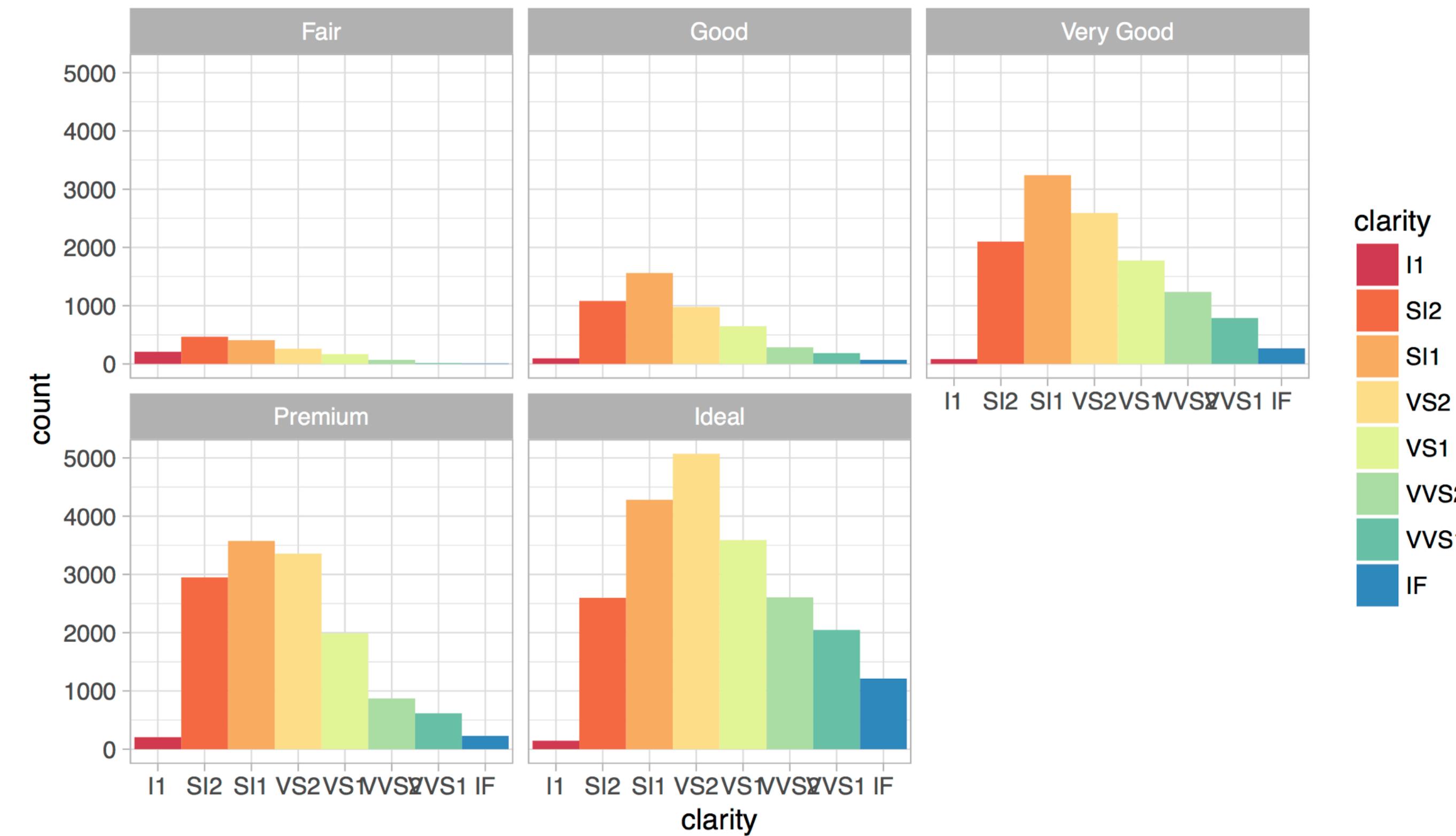
# Scales

## Customize color scales, other mappings

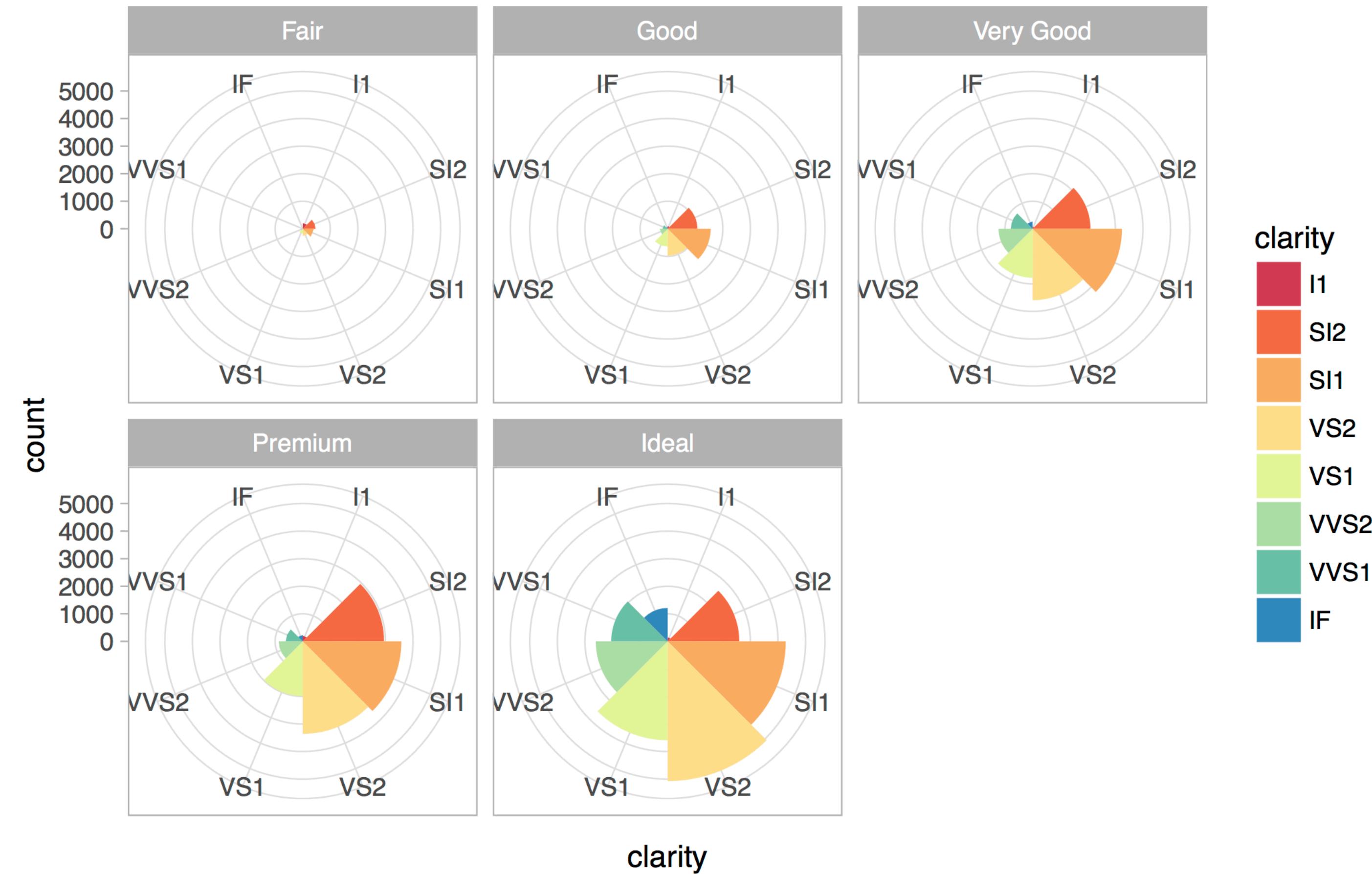


# Facets

Subplots that display subsets of the data.



# Coordinate systems



# Titles and captions

## Diamonds data

The data set is skewed towards ideal cut diamonds



Data by Hadley Wickham

# A ggplot2 template

Make any plot by filling in the parameters of this template

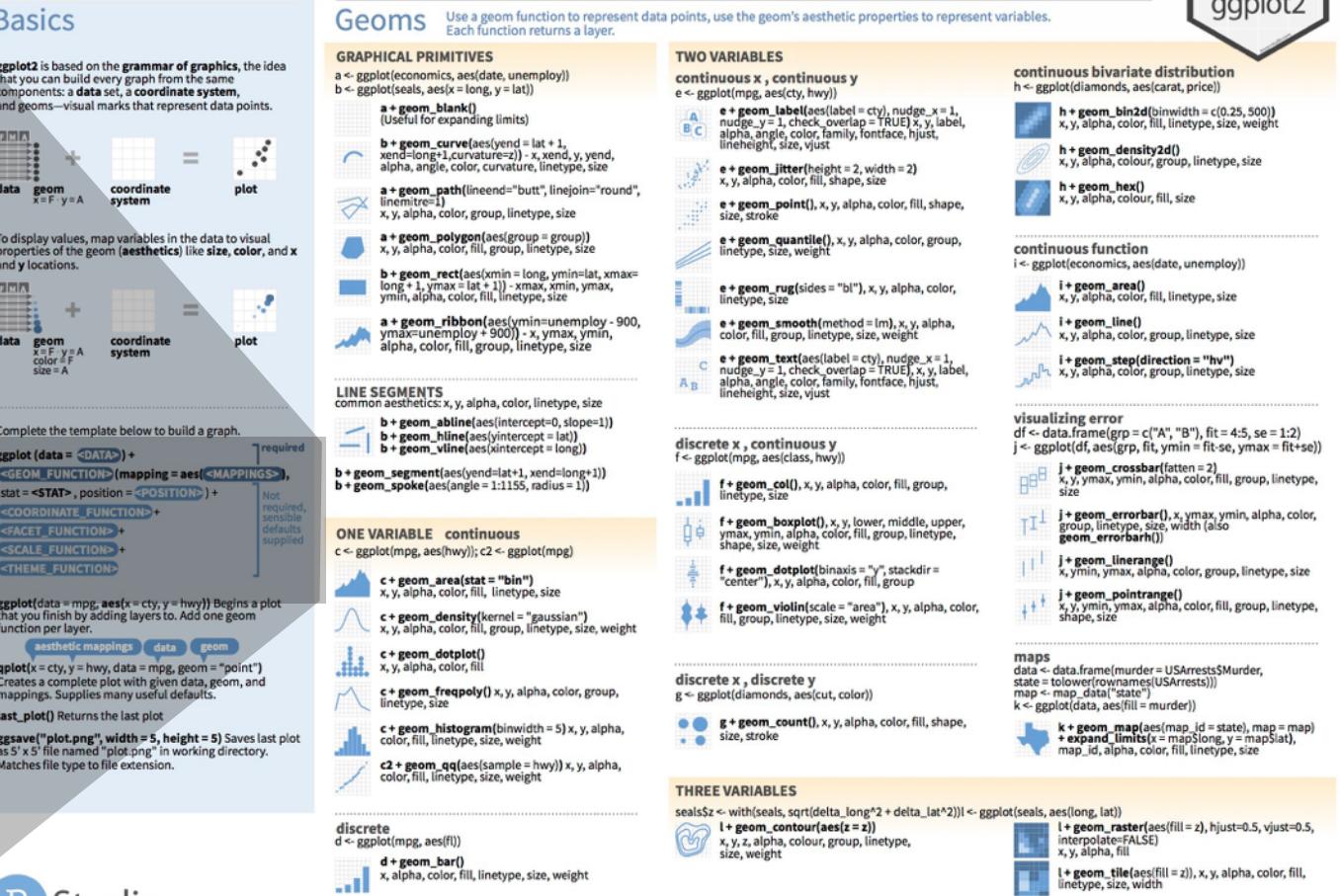
Complete the template below to build a graph.

**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

RStudio is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at <http://ggplot2.tidyverse.org> • ggplot2 2.1.0 • Updated: 2016-11

# ggplot2.tidyverse.org

The screenshot shows a web browser window displaying the ggplot2.tidyverse.org website. The page has a clean, modern design with a white background and light gray header and sidebar elements.

**Header:** The title bar says "Create Elegant Data Visualisation" and "ggplot2.tidyverse.org". The address bar also shows "ggplot2.tidyverse.org". The top right corner has a user name "Amelia".

**Left Sidebar:** A sidebar on the left contains sections for "Overview", "Installation", and "Usage".

- Overview:** A brief introduction to ggplot2, mentioning it's based on The Grammar of Graphics and provides a declarative interface for creating graphics.
- Installation:** Instructions for installing ggplot2, including options to install the whole tidyverse or just ggplot2, and a link to the development version on GitHub.
- Usage:** A paragraph explaining the philosophy of ggplot2 and how it works.

**Right Sidebar:** A sidebar on the right provides links to various resources and information about the package.

- Reference:** A dropdown menu with options like "Aesthetic specifications" and "Extending ggplot2".
- 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
  - Winston Chang**: Author
  - [All authors...](#)
- Dev status:** [build passing](#)