

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

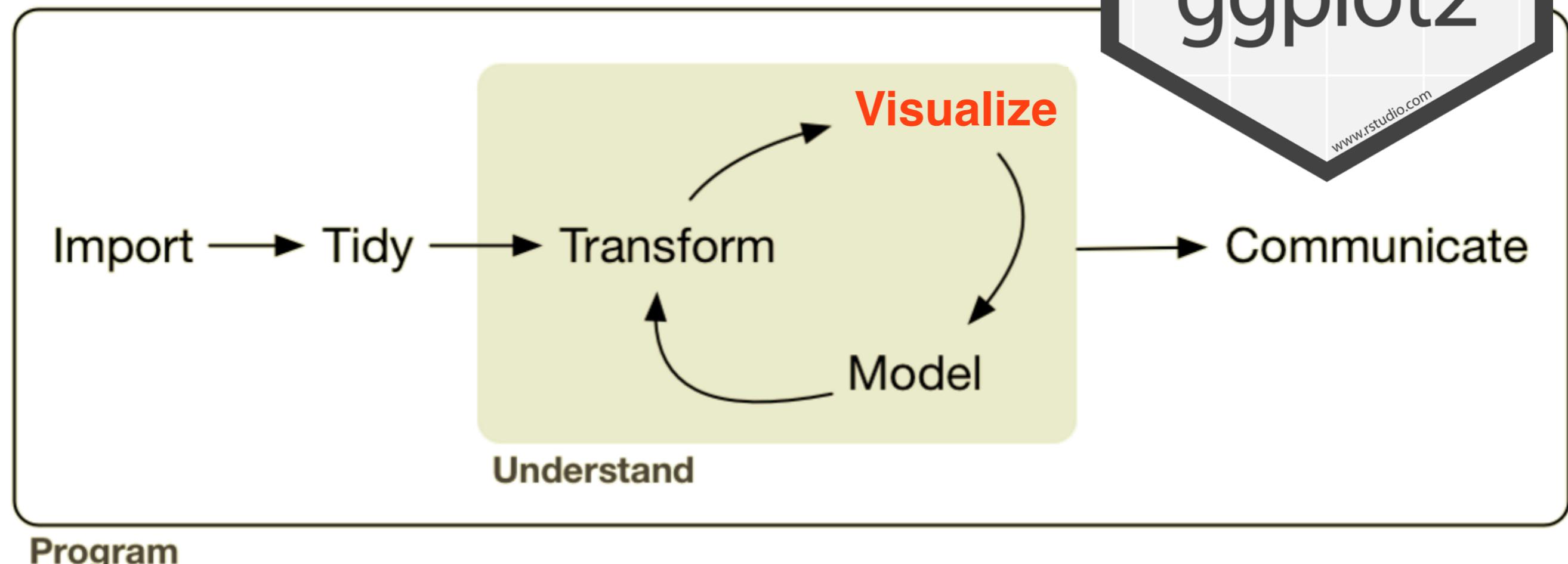
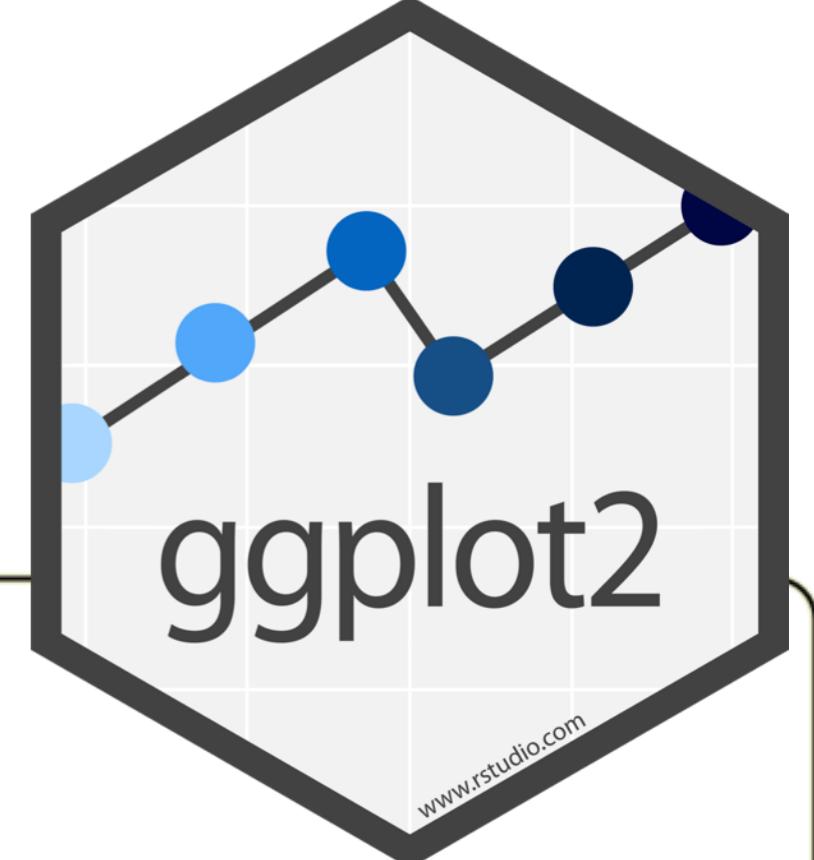
**Amelia McNamara**

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:

- Top Bar:** Shows the RStudio logo and menu items like "File", "Edit", "View", "Insert", "Run", "Knit", "Addins", and "Project: (None)".
- Left Panel:** Displays the content of the R Notebook "02-Visualize-Data.Rmd". The code includes:

  - YAML header:

```
1 ---  
2 title: "Visualize Data"  
3 output:  
4   html_document:  
5     df_print: paged  
6 ---  
7
```

  - A section titled "# Setup" followed by explanatory text:

```
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.
```

  - Code chunks:

```
11  
12 ```{r setup}  
13 library(ggplot2)  
14 library(fivethirtyeight)  
15 ```  
16  
17 ## Bechdel test data
```


- Bottom Left:** A "Console" tab is visible at the bottom left.
- Right Panel:** The "Global Environment" panel is open, showing the "Environment" tab with a list of objects. It also includes tabs for "History" and "Connections".
- Bottom Right:** A large text overlay reads "Open the R Notebook 02-Visualize-Data.Rmd".

# Setup

The setup chunk is always run once before anything else

A screenshot of the RStudio interface showing an R Markdown file named "01-Visualize-Data.Rmd". The code editor pane displays the following content:

```
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
```

A gray callout bubble with a black border and white text points from the bottom left towards the line of code starting with "```{r setup}". The text inside the bubble reads "(optional) label for chunk".

# 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. In the top-left corner, there are two tabs: "02-Visualize-Data.Rmd" and "bechdel". Below the tabs is a toolbar with icons for back, forward, and search. The main area displays a data frame titled "View(bechdel)". The columns are labeled: year, imdb, title, test, clean\_test, binary, and budget. The data consists of 14 rows of movie information. At the bottom of the data frame, it says "Showing 1 to 14 of 1,794 entries". In the bottom-left corner of the RStudio window, there is a console tab with the command "> View(bechdel)" entered.

	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)

Type this in your  
Console, NOT in your  
Notebook

# Consider

Confer with the people around you.

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



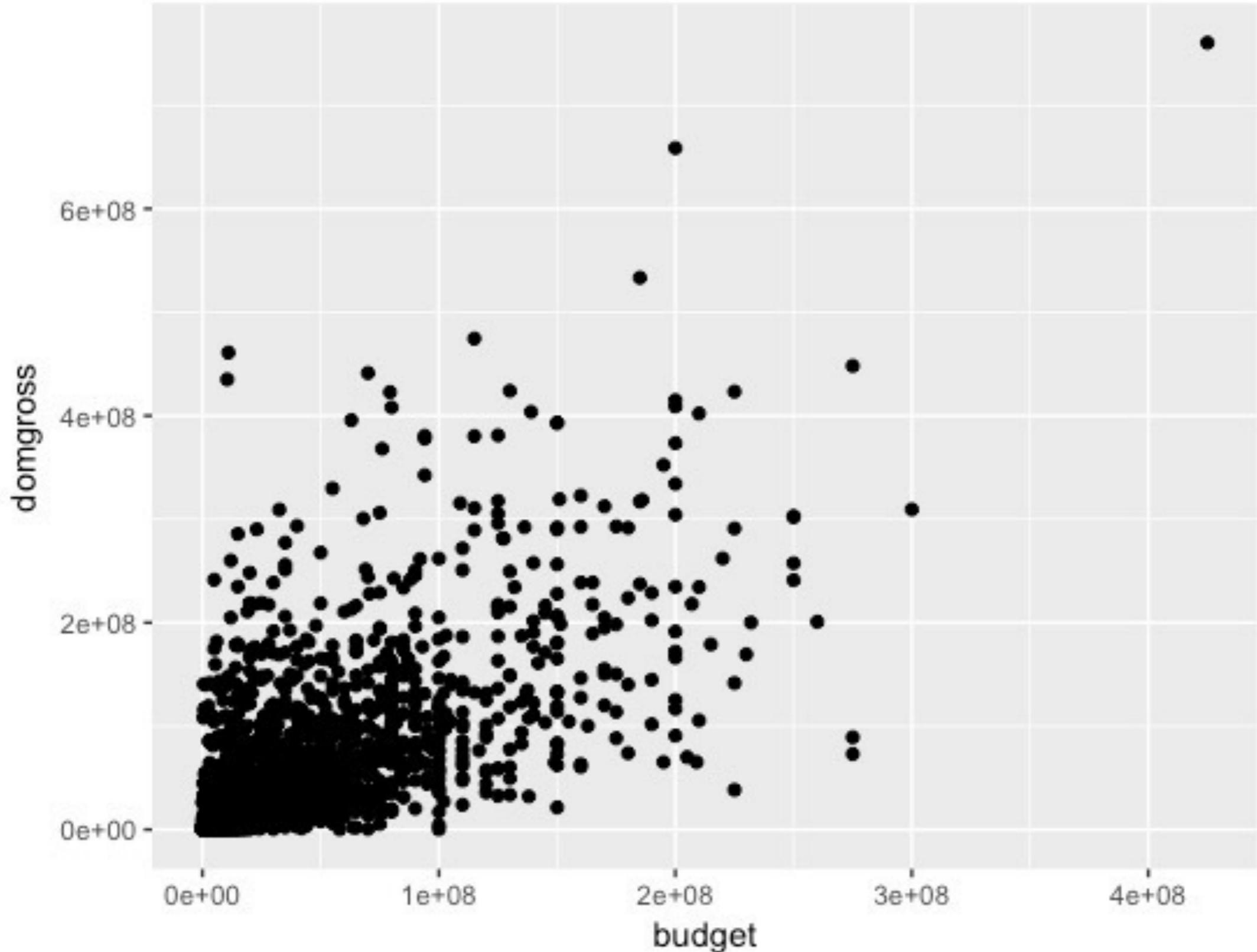
# Your Turn 1

Run this code in your notebook to make a graph.

Pay strict attention to spelling, capitalization, and parentheses!

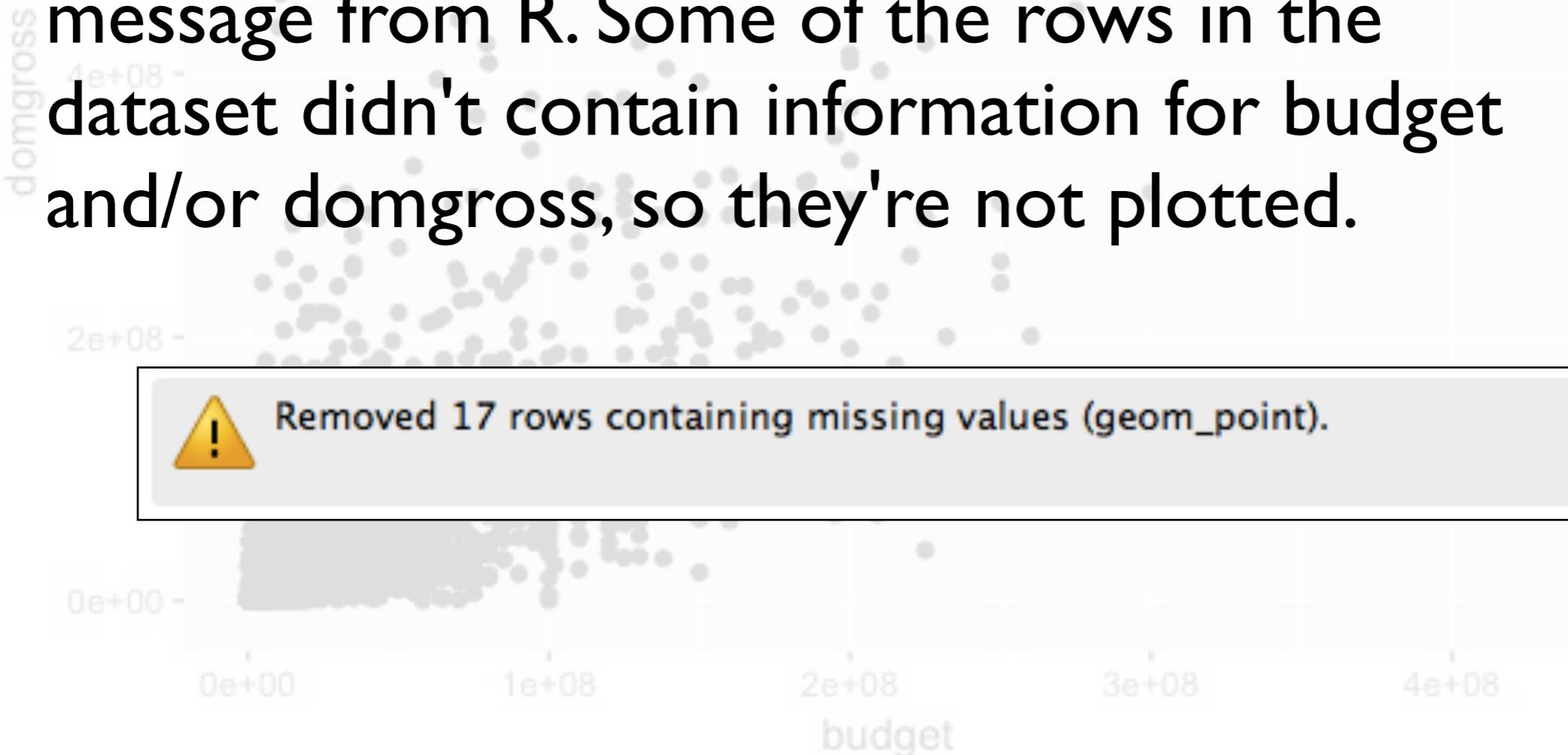
```
ggplot(data = bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross))
```





```
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

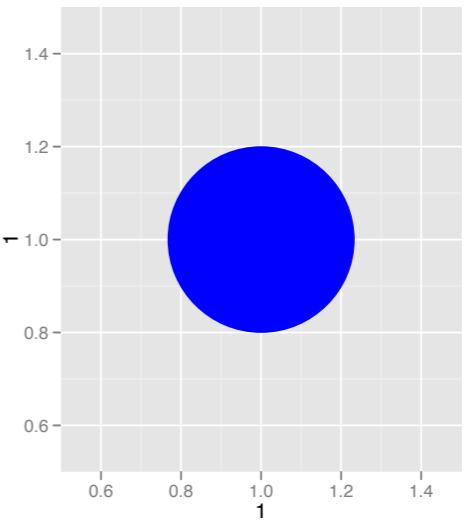
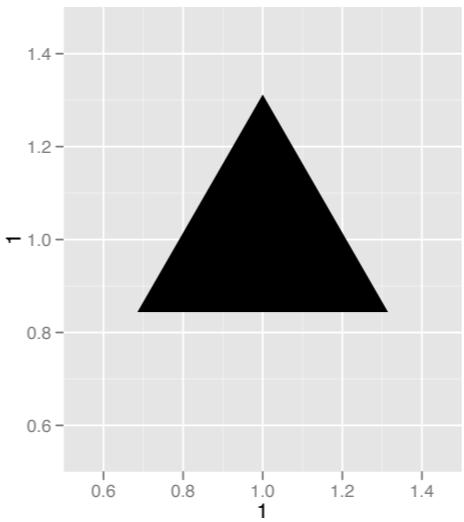
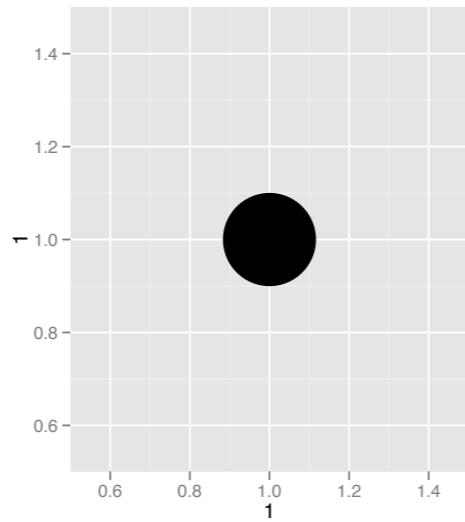
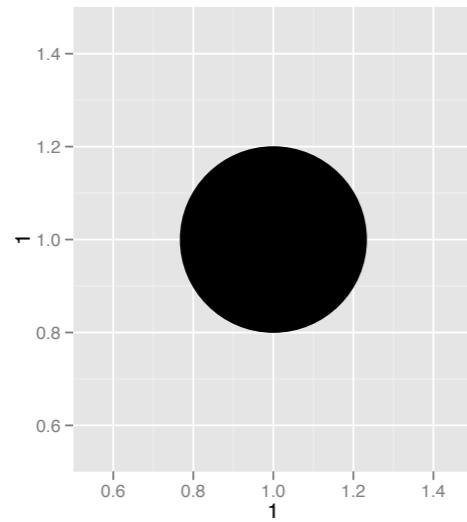
```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))  
    geom_point(mapping = aes(x = budget, y = domgross))
```

# A template

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

# Aesthetics

# Aesthetics



# Visual Space

color

# Data Space

class

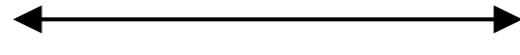
Red

Brown

Green

Blue

Pink



nowomen

notalk

men

dubious

ok

# Aesthetics

aesthetic  
property

Variable to  
map it to

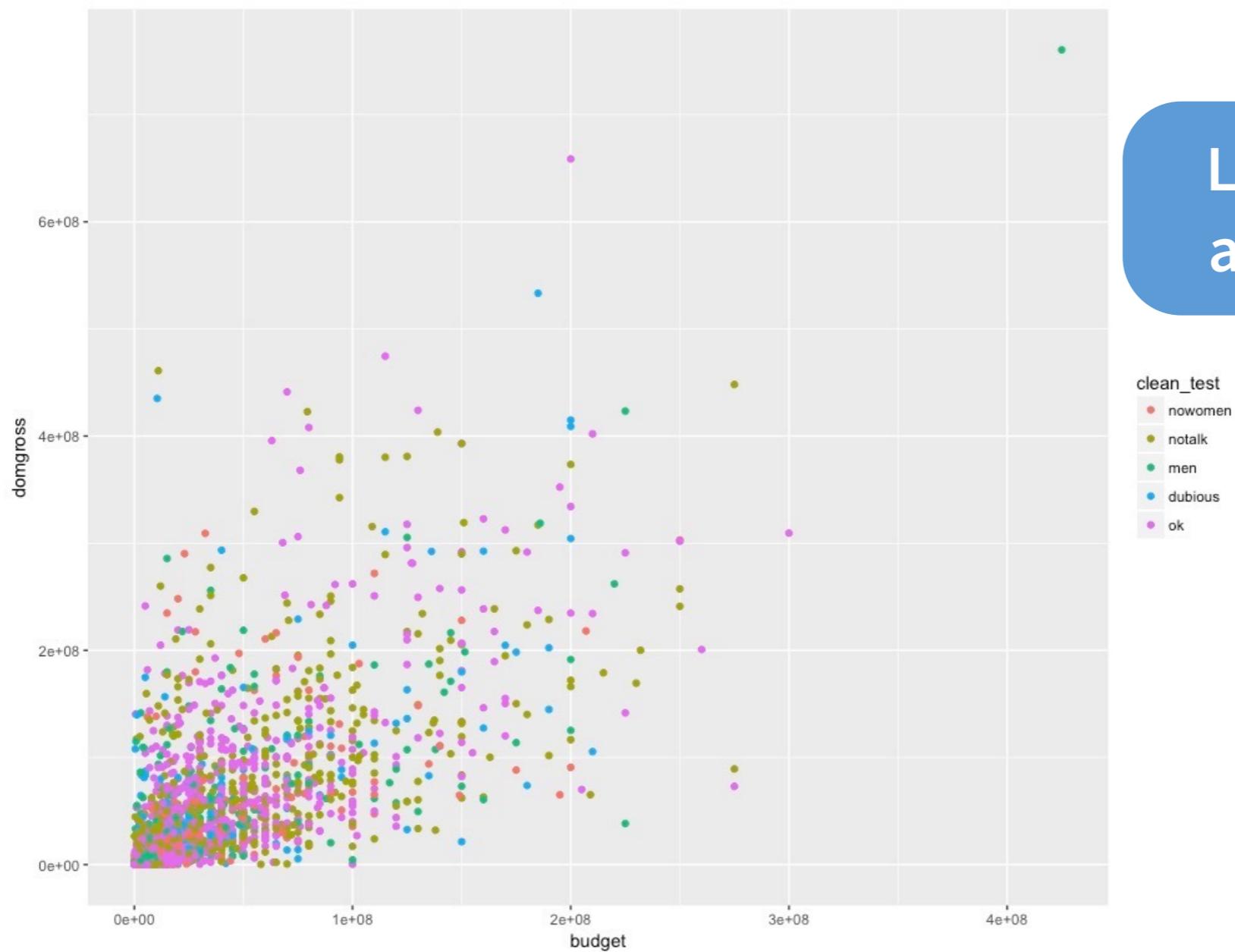
```
ggplot(bechdel) + geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
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.

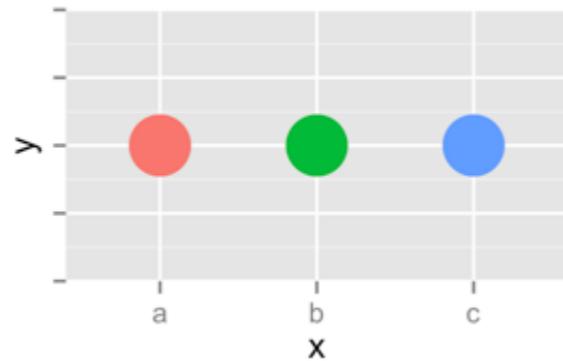
Do different things happen when you map aesthetics to discrete and continuous variables?

What happens when you use more than one aesthetic?



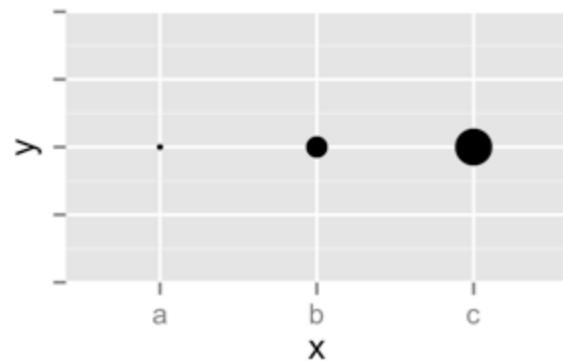
Color

Discrete

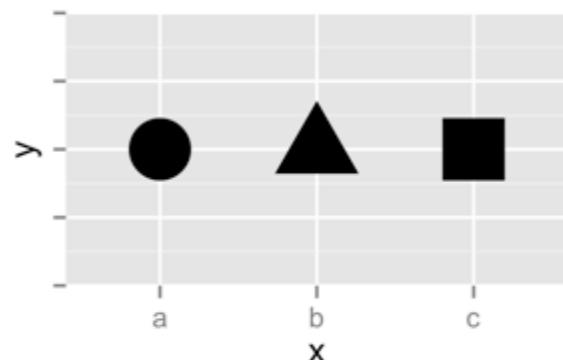


Size

Continuous

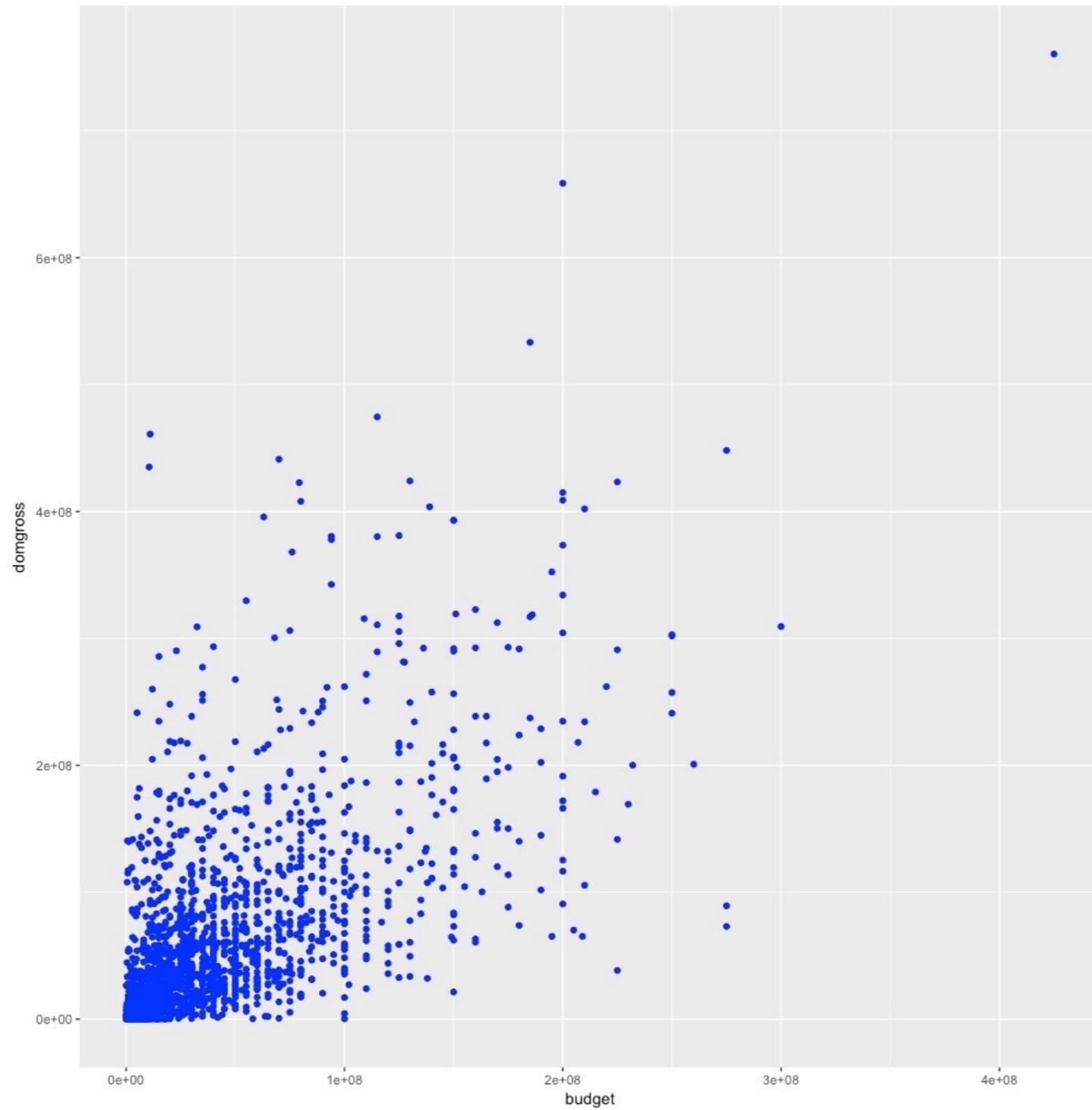


Shape



set vs. map

# How would you make this plot?

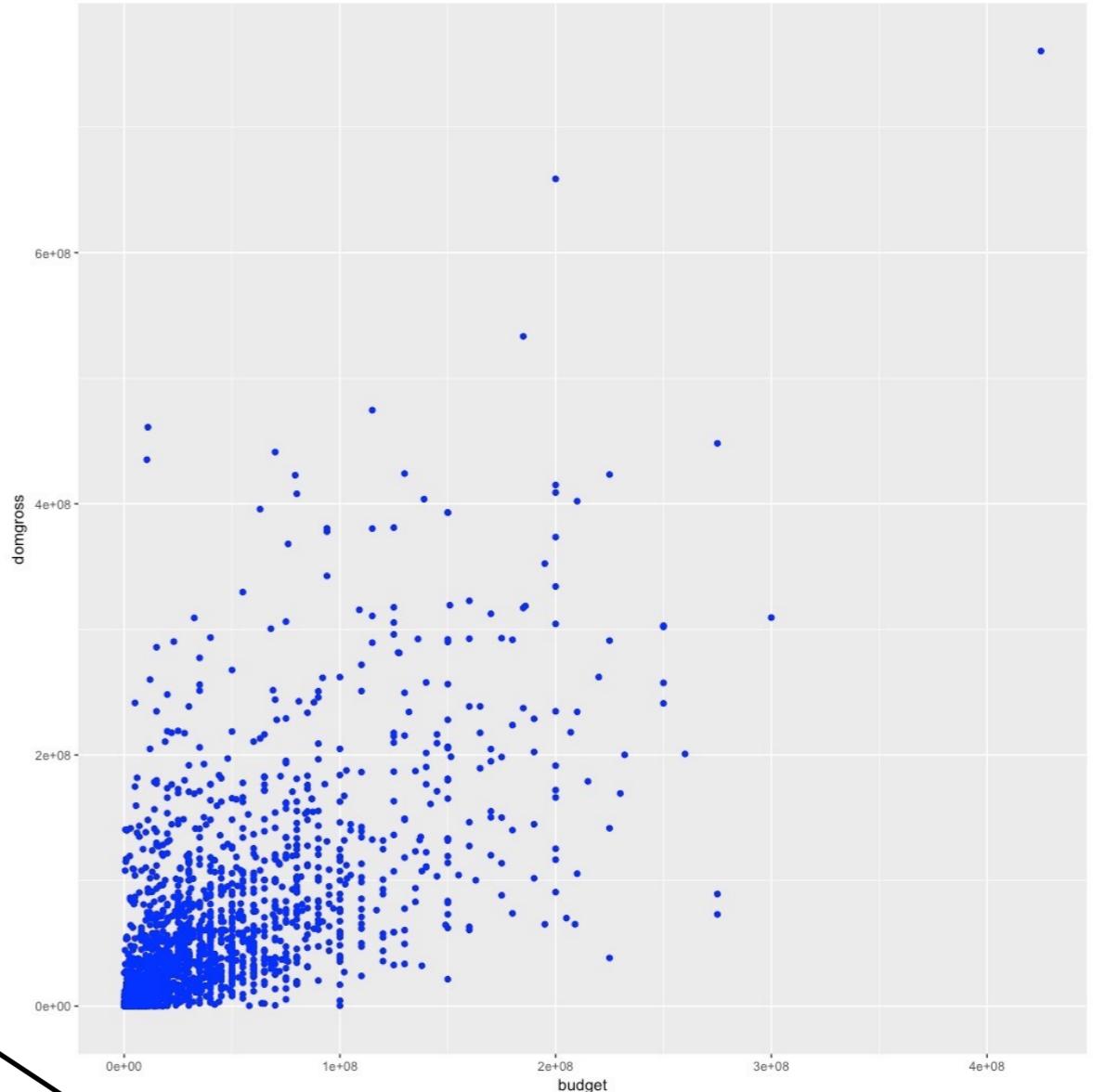




**Inside of aes():** maps an aesthetic to a variable

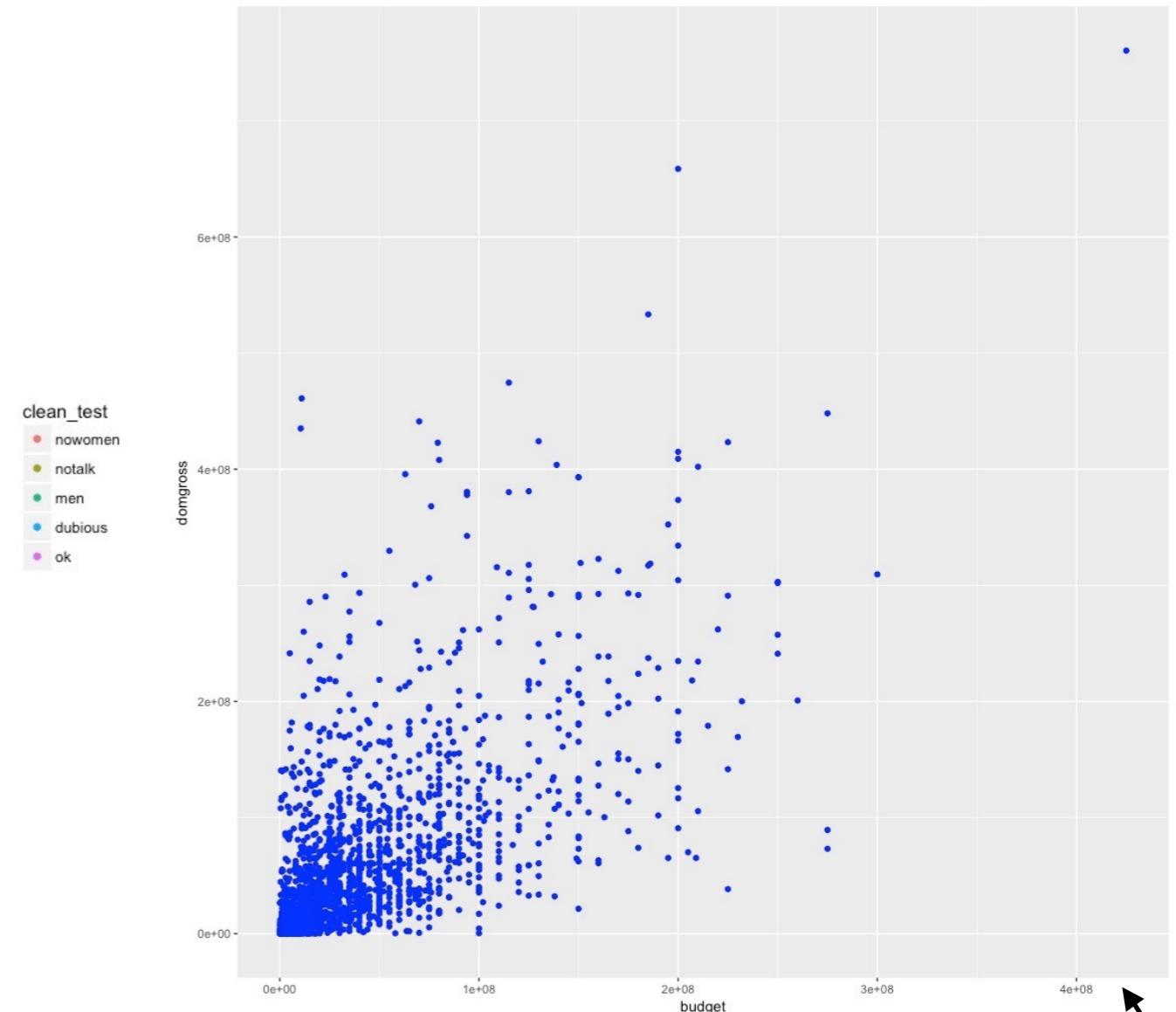
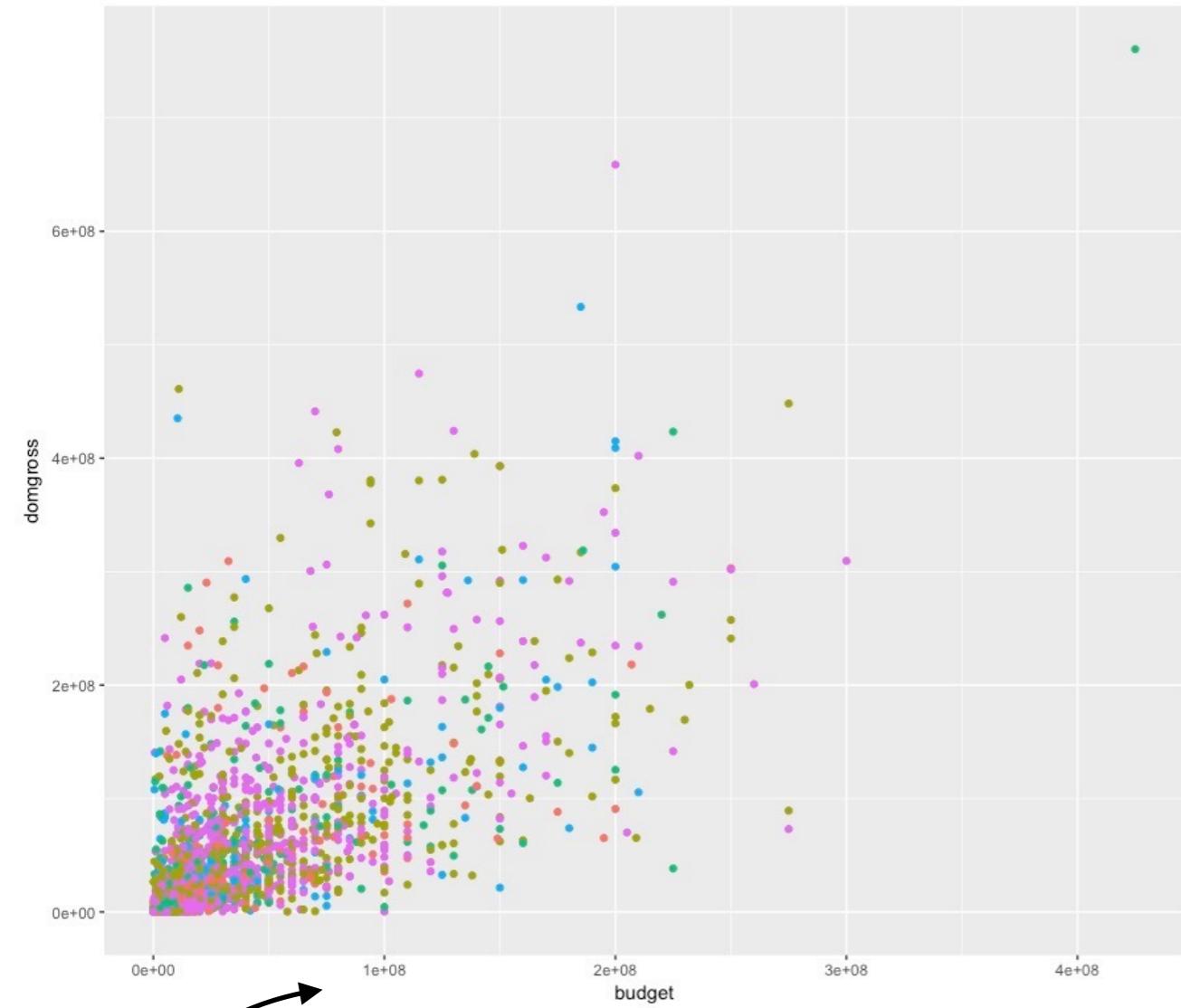
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

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



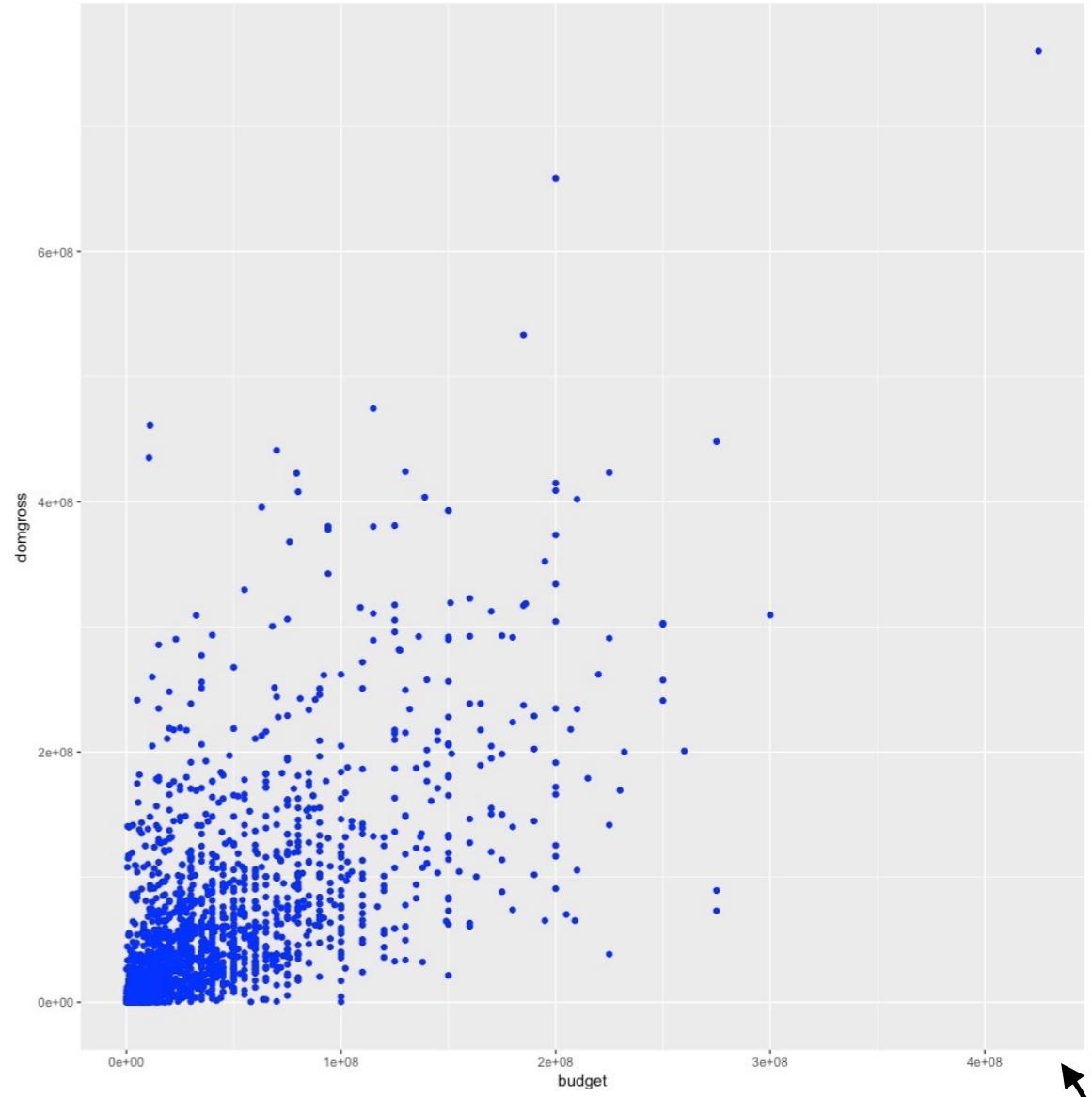
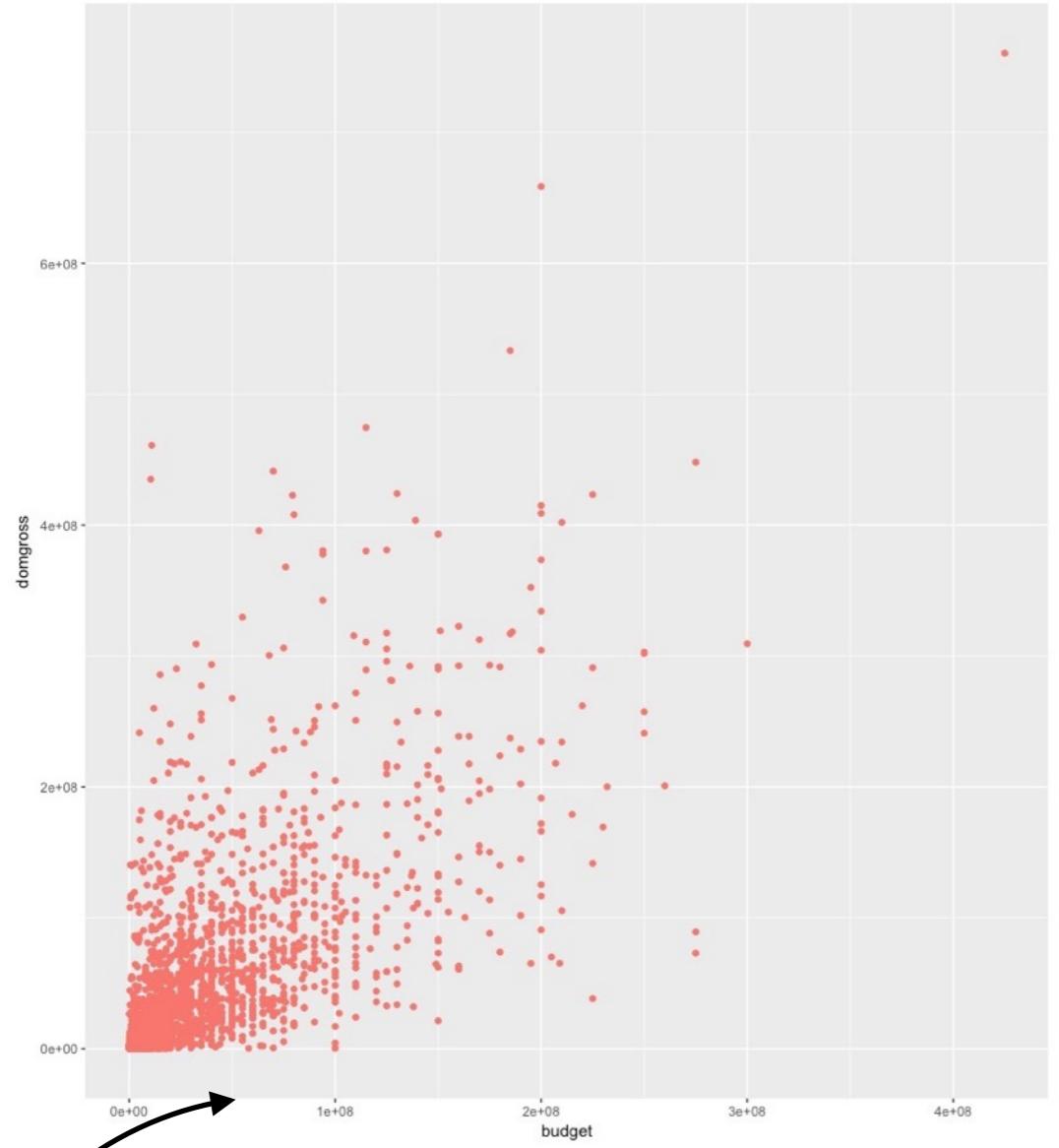
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
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")
```

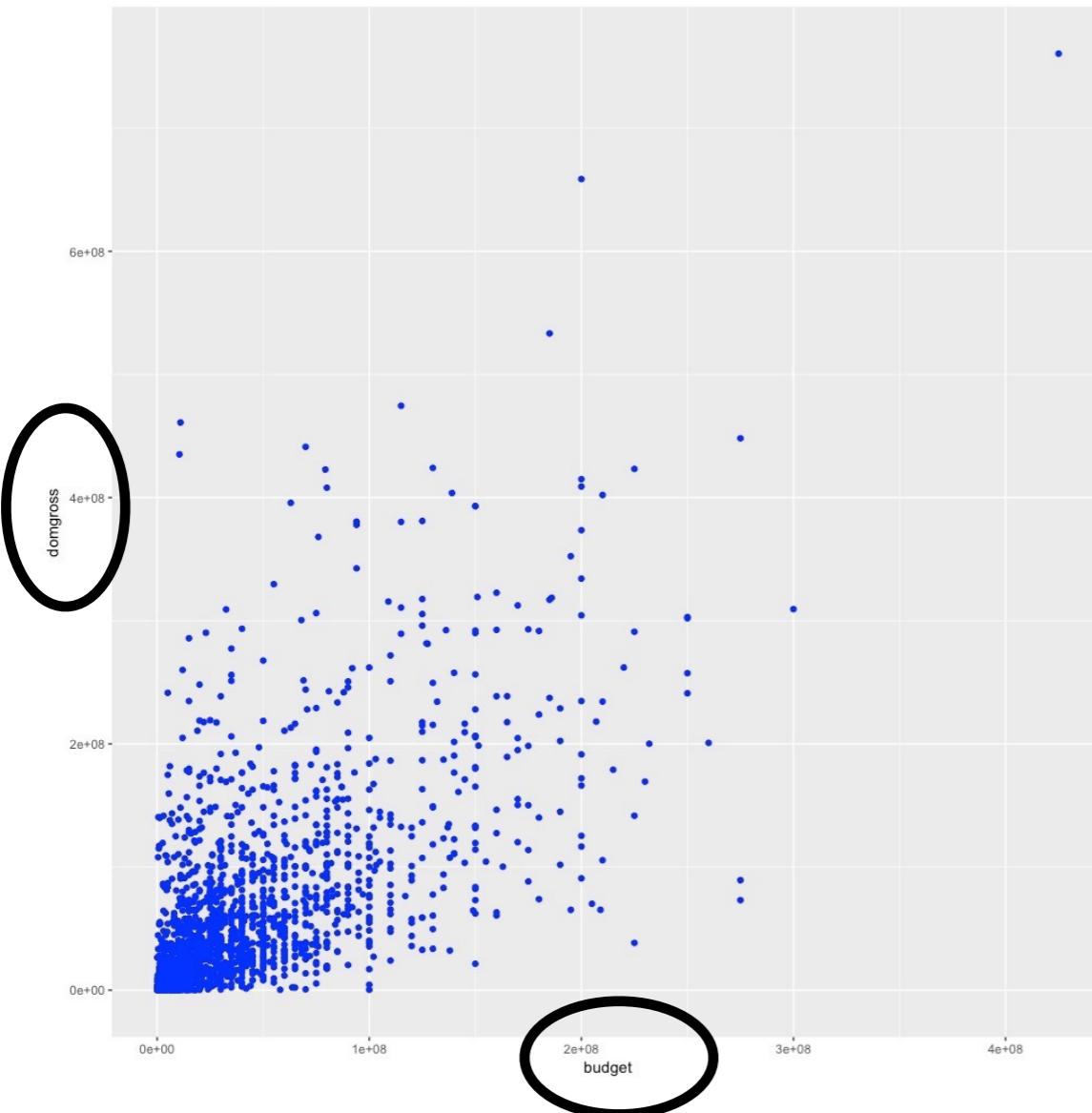


```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross, color="blue"))
```

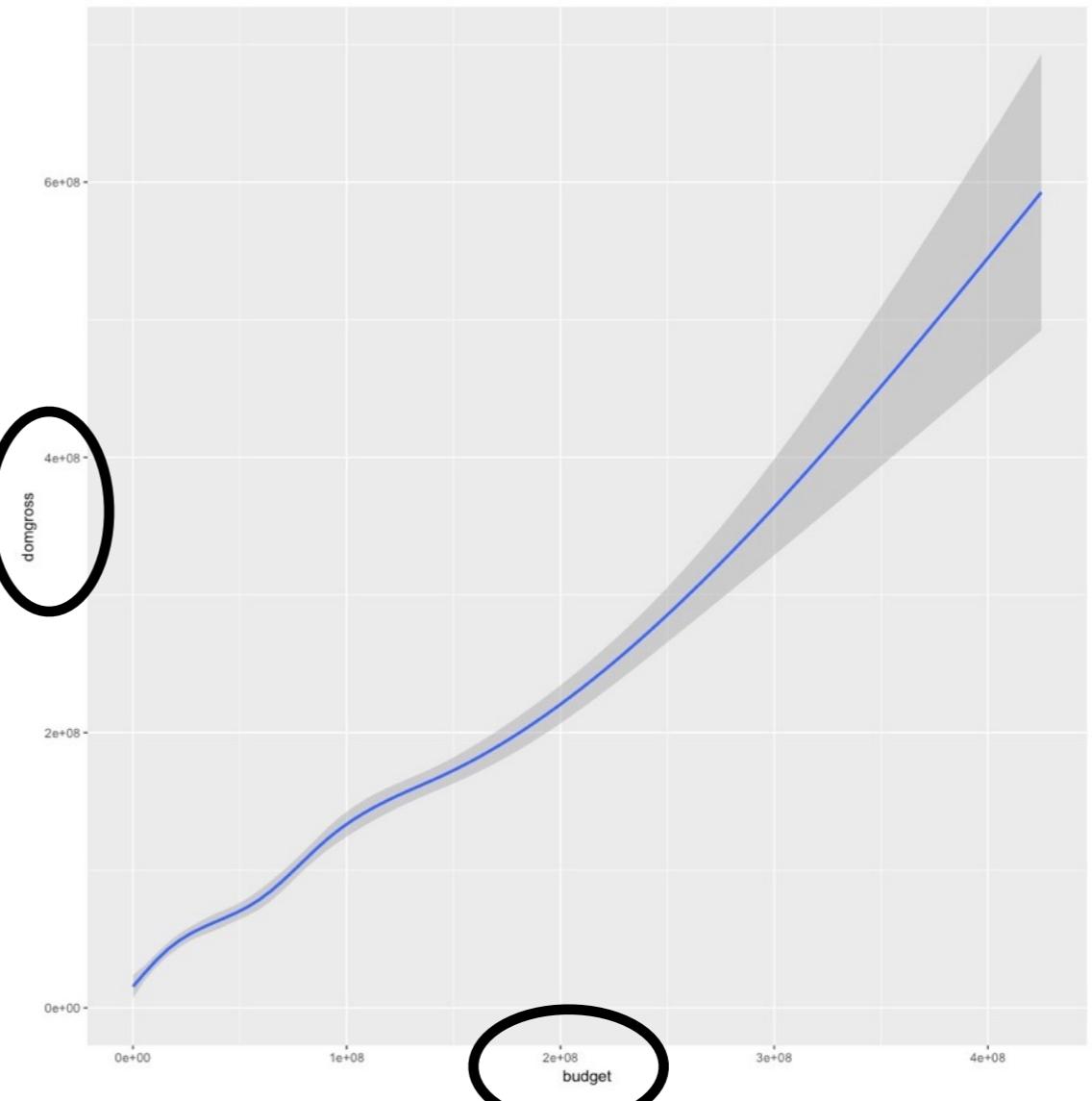
```
ggplot(bechdel) +  
  geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

geoms

How are these plots similar?

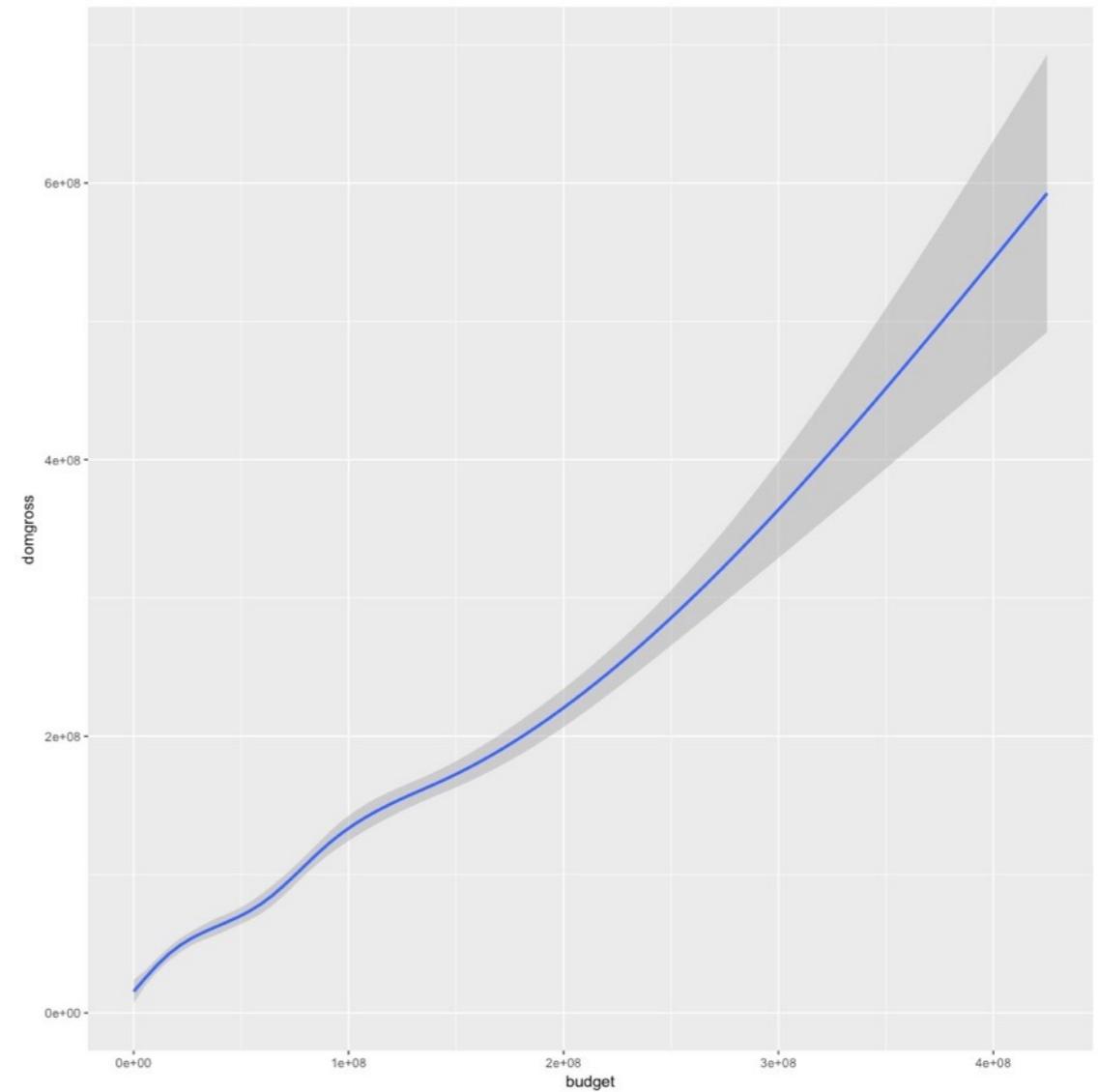
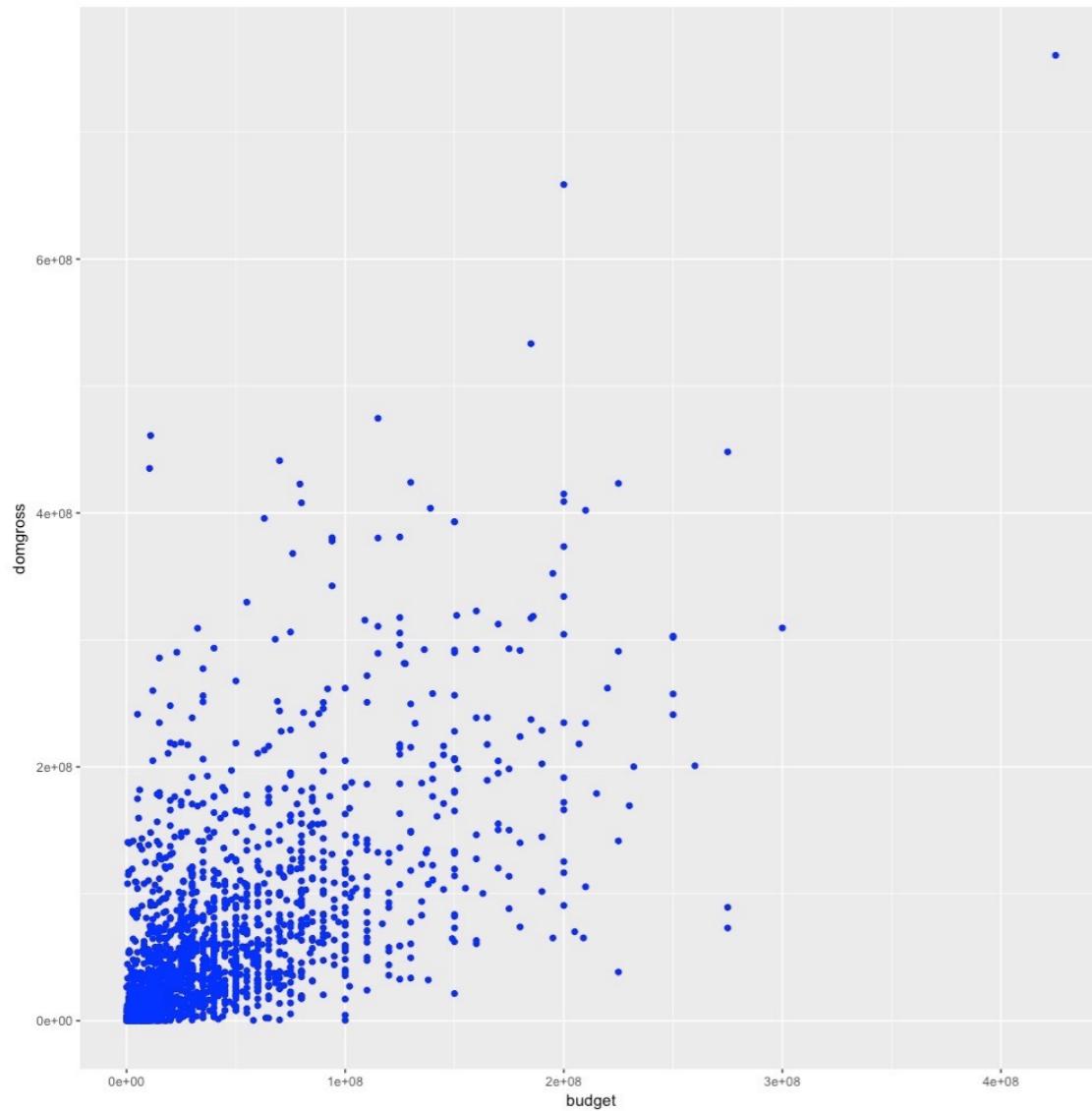


Same: x var, y var, data



## How are these plots different?

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

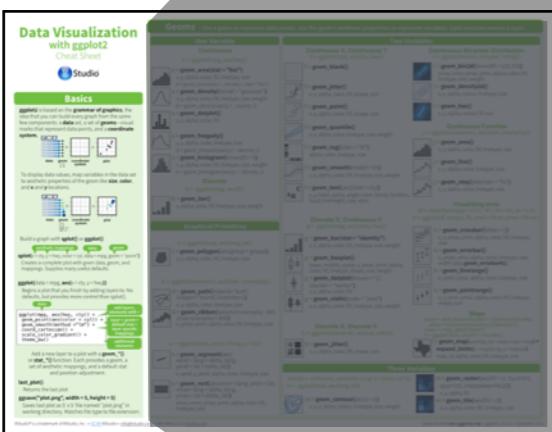


# geoms

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

# geom\_ functions

Each requires a mapping argument.



Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.		
One Variable	Two Variables	Three Variables
<b>Continuous</b> <ul style="list-style-type: none"> <li>a + <code>geom_area(stat = "bin")</code> x, y, alpha, color, fill, linetype, size</li> <li>a + <code>geom_density(kernel = "gaussian")</code> x, y, alpha, color, fill, linetype, size, weight</li> <li>a + <code>geom_dotplot()</code> x, y, alpha, color, fill</li> <li>a + <code>geom_freqpoly()</code> x, y, alpha, color, linetype, size b + <code>geom_freqpoly(aes(y = ..density..))</code></li> <li>a + <code>geom_histogram(binwidth = 5)</code> x, y, alpha, color, fill, linetype, size, weight b + <code>geom_histogram(aes(y = ..density..))</code></li> </ul>	<b>Continuous X, Continuous Y</b> <ul style="list-style-type: none"> <li>f + <code>geom_blank()</code> (Useful for expanding limits)</li> <li>f + <code>geom_jitter()</code> x, y, alpha, color, fill, shape, size</li> <li>f + <code>geom_point()</code> x, y, alpha, color, fill, shape, size</li> <li>f + <code>geom_quantile()</code> x, y, alpha, color, linetype, size, weight</li> <li>f + <code>geom_rug(sides = "bl")</code> alpha, color, linetype, size</li> <li>f + <code>geom_smooth(method = lm)</code> x, y, alpha, color, fill, linetype, size, weight</li> <li>f + <code>geom_text(aes(label = cty))</code> x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust</li> </ul>	<b>Continuous Bivariate Distribution</b> <ul style="list-style-type: none"> <li>i + <code>geom_bin2d(binwidth = c(5, 0.5))</code> xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight</li> <li>i + <code>geom_density2d()</code> x, y, alpha, colour, linetype, size</li> <li>i + <code>geom_hex()</code> x, y, alpha, colour, fill size</li> </ul>
<b>Discrete</b> <ul style="list-style-type: none"> <li>b + <code>geom_bar()</code> x, alpha, color, fill, linetype, size, weight</li> </ul>	<b>Continuous Function</b> <ul style="list-style-type: none"> <li>j + <code>geom_area()</code> x, y, alpha, color, fill, linetype, size</li> <li>j + <code>geom_line()</code> x, y, alpha, color, linetype, size</li> <li>j + <code>geom_step(direction = "hv")</code> x, y, alpha, color, linetype, size</li> </ul>	<b>Visualizing error</b> <ul style="list-style-type: none"> <li>df &lt;- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2) k &lt;- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))</li> </ul>
<b>Graphical Primitives</b> <ul style="list-style-type: none"> <li>c + <code>geom_polygon(aes(group = group))</code> x, y, alpha, color, fill, linetype, size</li> </ul>	<b>Discrete X, Continuous Y</b> <ul style="list-style-type: none"> <li>g + <code>geom_bar(stat = "identity")</code> x, y, alpha, color, fill, linetype, size, weight</li> <li>g + <code>geom_boxplot()</code> lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight</li> <li>g + <code>geom_dotplot(binaxis = "y", stackdir = "center")</code> x, y, alpha, color, fill</li> <li>g + <code>geom_violin(scale = "area")</code> x, y, alpha, color, fill, linetype, size, weight</li> </ul>	<b>Maps</b> <ul style="list-style-type: none"> <li>k + <code>geom_crossbar(fatten = 2)</code> x, y, ymax, ymin, alpha, color, fill, linetype, size</li> <li>k + <code>geom_errorbar()</code> x, ymax, ymin, alpha, color, linetype, size, width (also <code>geom_errorbarh()</code>)</li> <li>k + <code>geom_linerange()</code> x, ymin, ymax, alpha, color, linetype, size</li> <li>k + <code>geom_pointrange()</code> x, y, ymin, ymax, alpha, color, fill, linetype, shape, size</li> </ul>
	<b>Discrete X, Discrete Y</b> <ul style="list-style-type: none"> <li>h &lt;- ggplot(diamonds, aes(cut, color))</li> <li>h + <code>geom_jitter()</code> x, y, alpha, color, fill, shape, size</li> </ul>	<ul style="list-style-type: none"> <li>data &lt;- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests))) map &lt;- map_data("state") l &lt;- ggplot(data, aes(fill = murder))</li> <li>l + <code>geom_map(aes(map_id = state), map = map) + expand_limits(x = map\$long, y = map\$lat)</code> map_id, alpha, color, fill, linetype, size</li> </ul>
		<b>Three Variables</b> <ul style="list-style-type: none"> <li>seals\$z &lt;- with(seals, sqrt(delta_long^2 + delta_lat^2)) m &lt;- ggplot(seals, aes(long, lat))</li> <li>m + <code>geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)</code> x, y, alpha, fill (<b>fast</b>)</li> <li>m + <code>geom_contour(aes(z = z))</code> x, y, z, alpha, colour, linetype, size, weight</li> <li>m + <code>geom_tile(aes(fill = z))</code> x, y, alpha, color, fill, linetype, size (<b>slow</b>)</li> </ul>

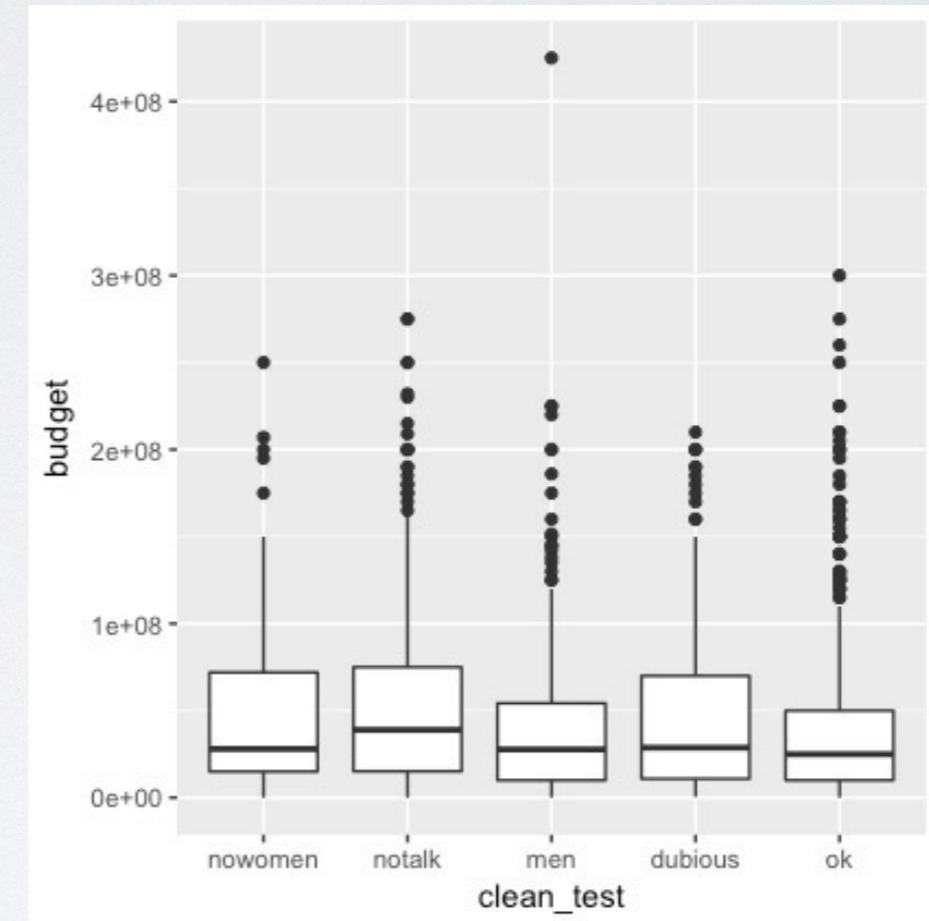
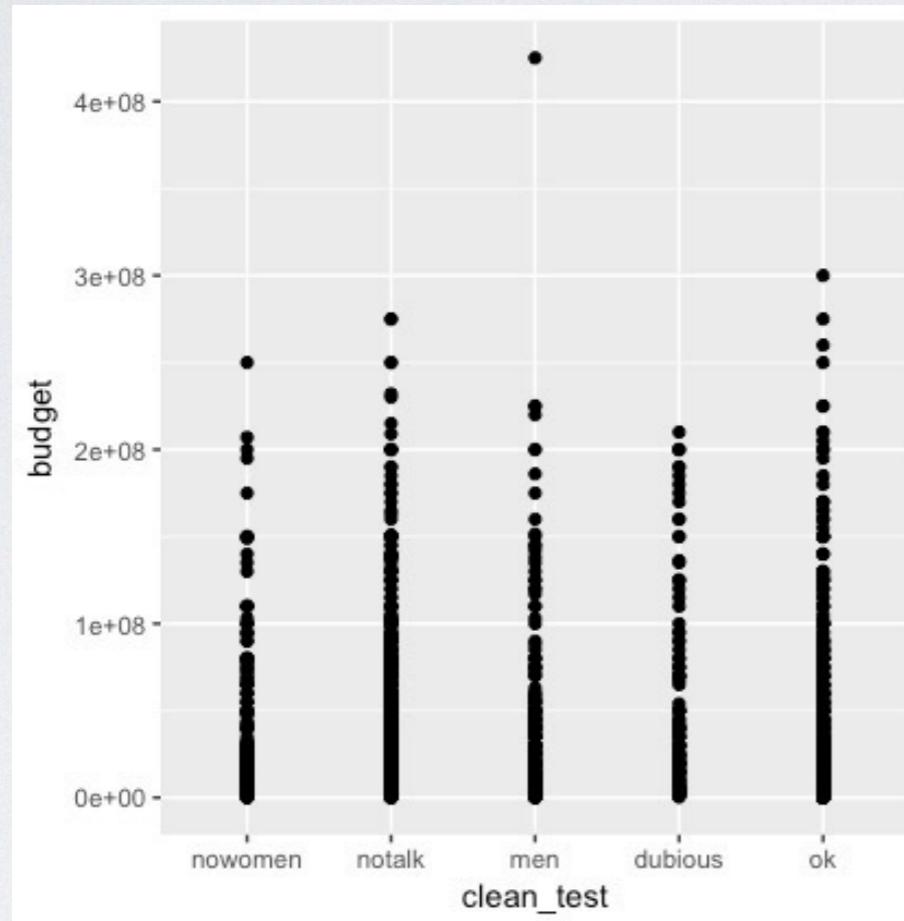
# Your Turn

Pair up.



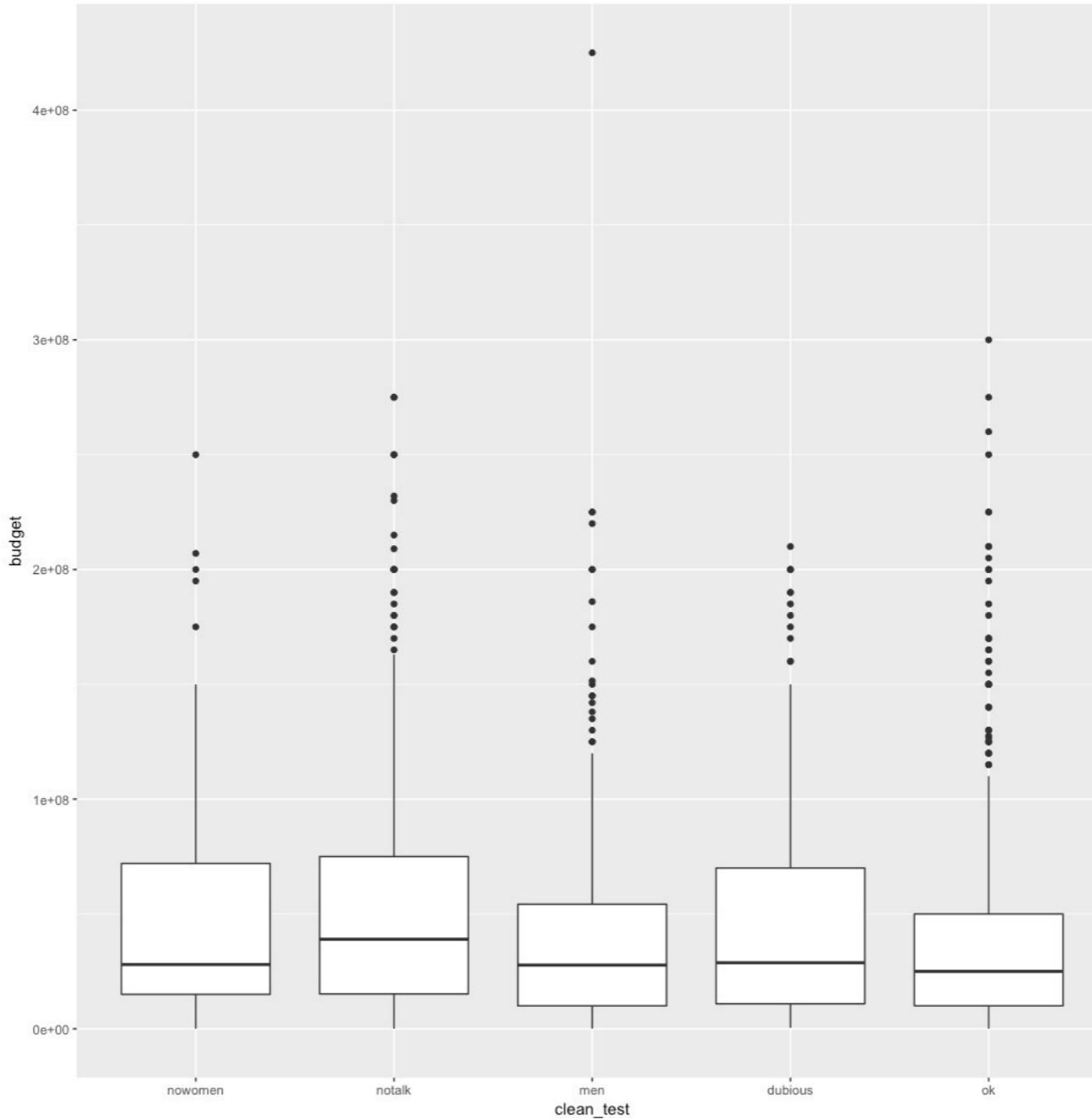
# Your Turn 3

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



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

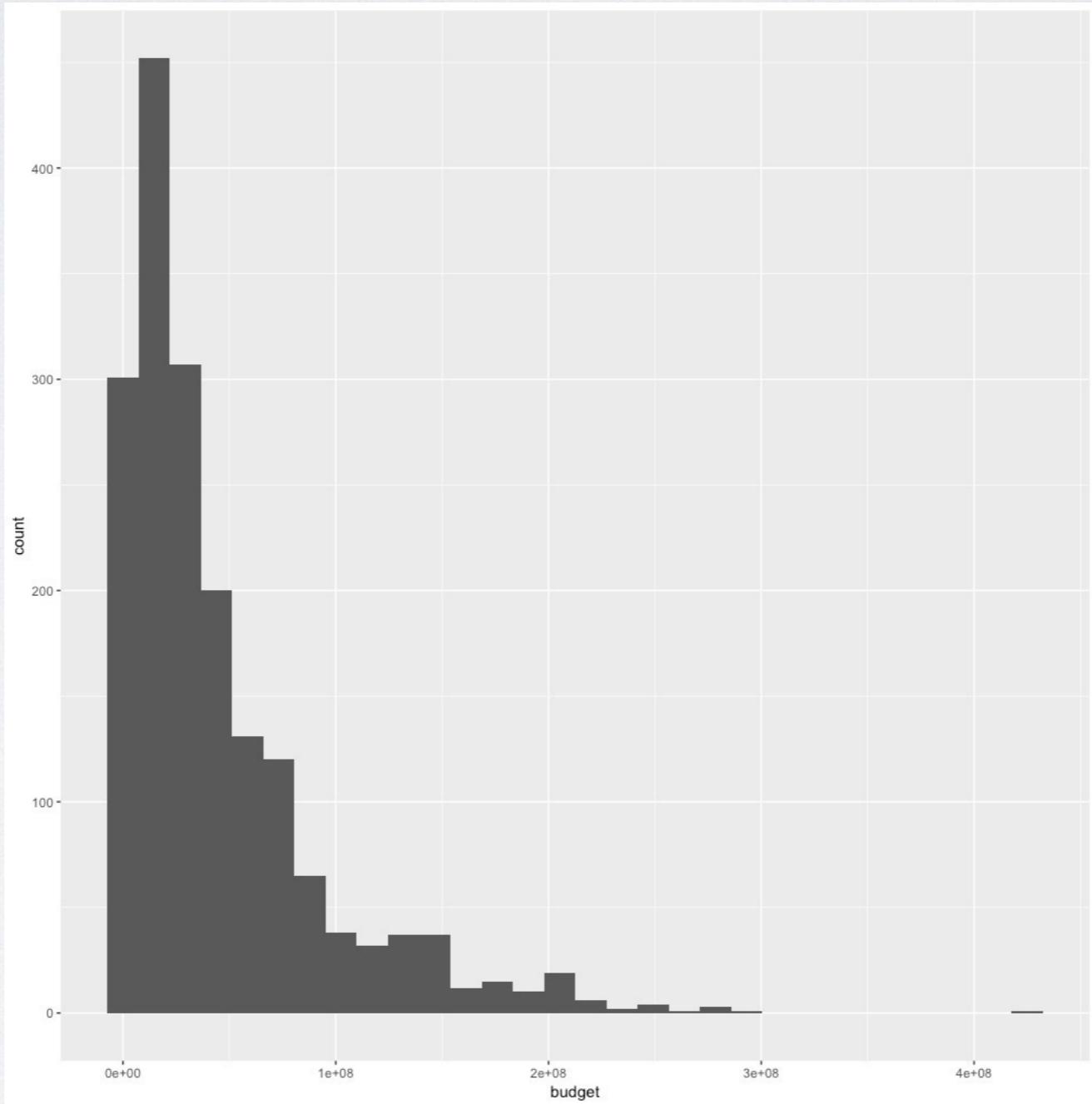


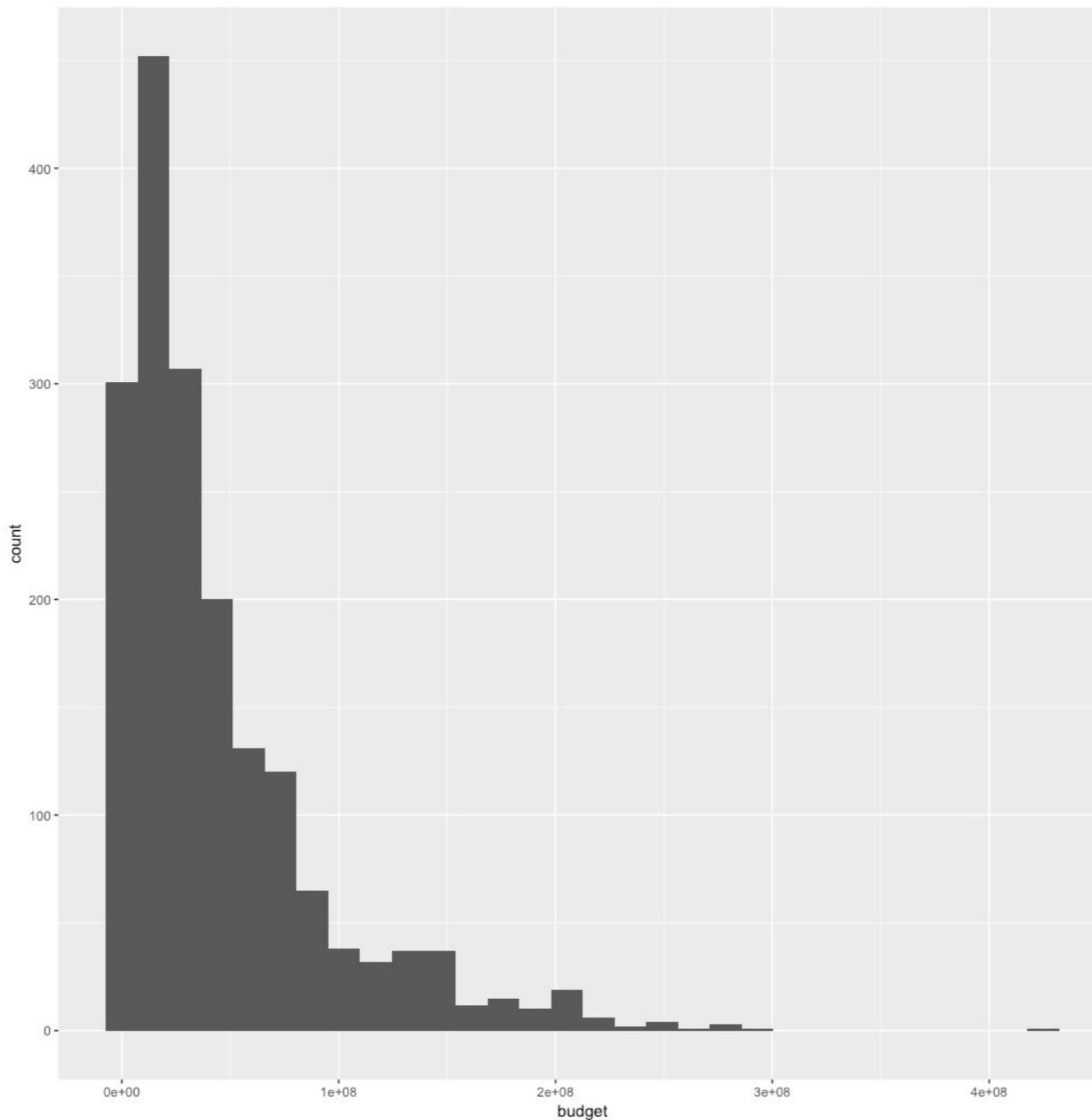


```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

# 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



`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

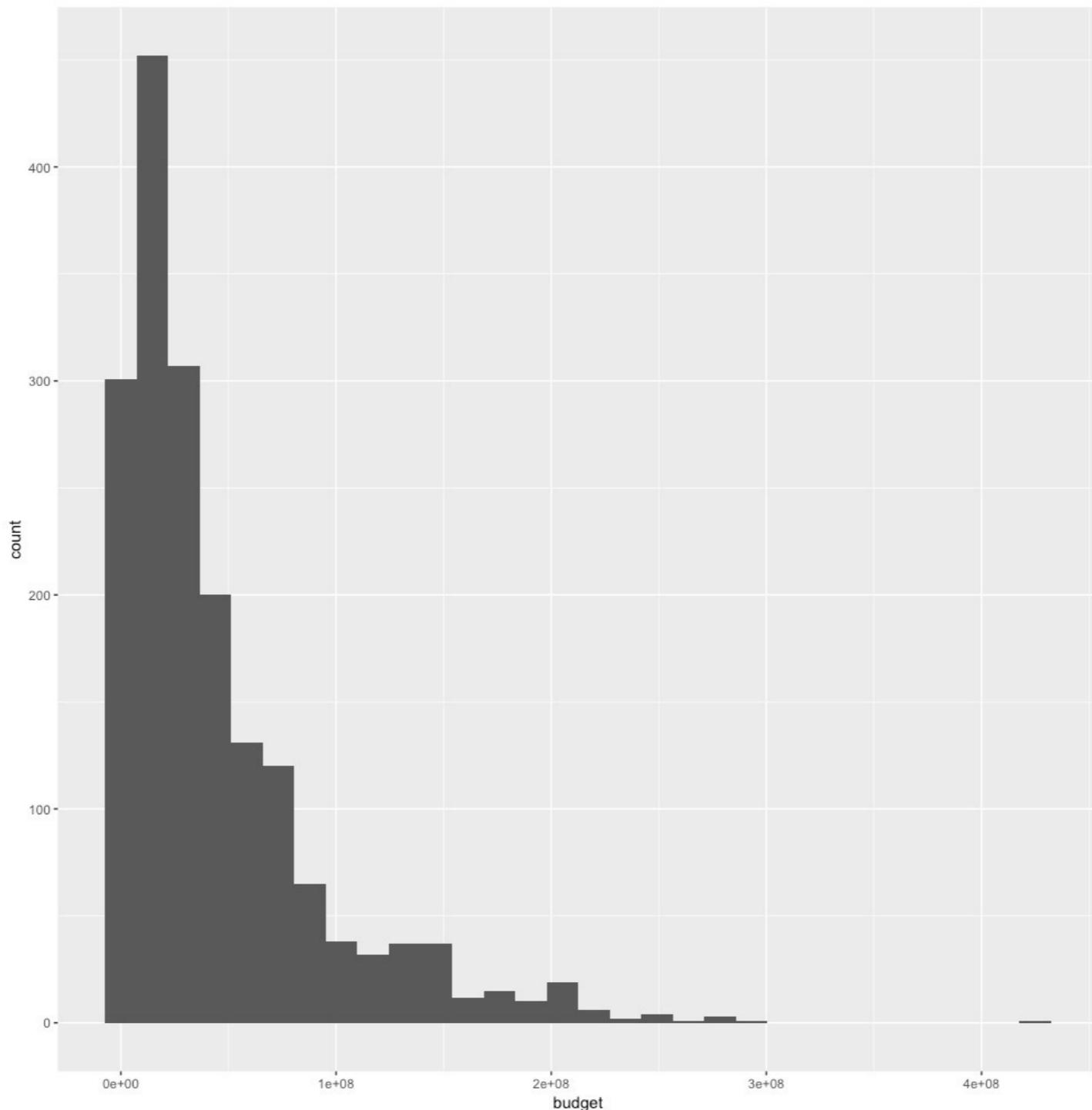
```
ggplot(data = bechdel) +  
  geom_histogram(mapping = aes(x = budget))
```

# Your Turn 5

What would be a  
reasonable bin width for  
**budget?**

Try it out

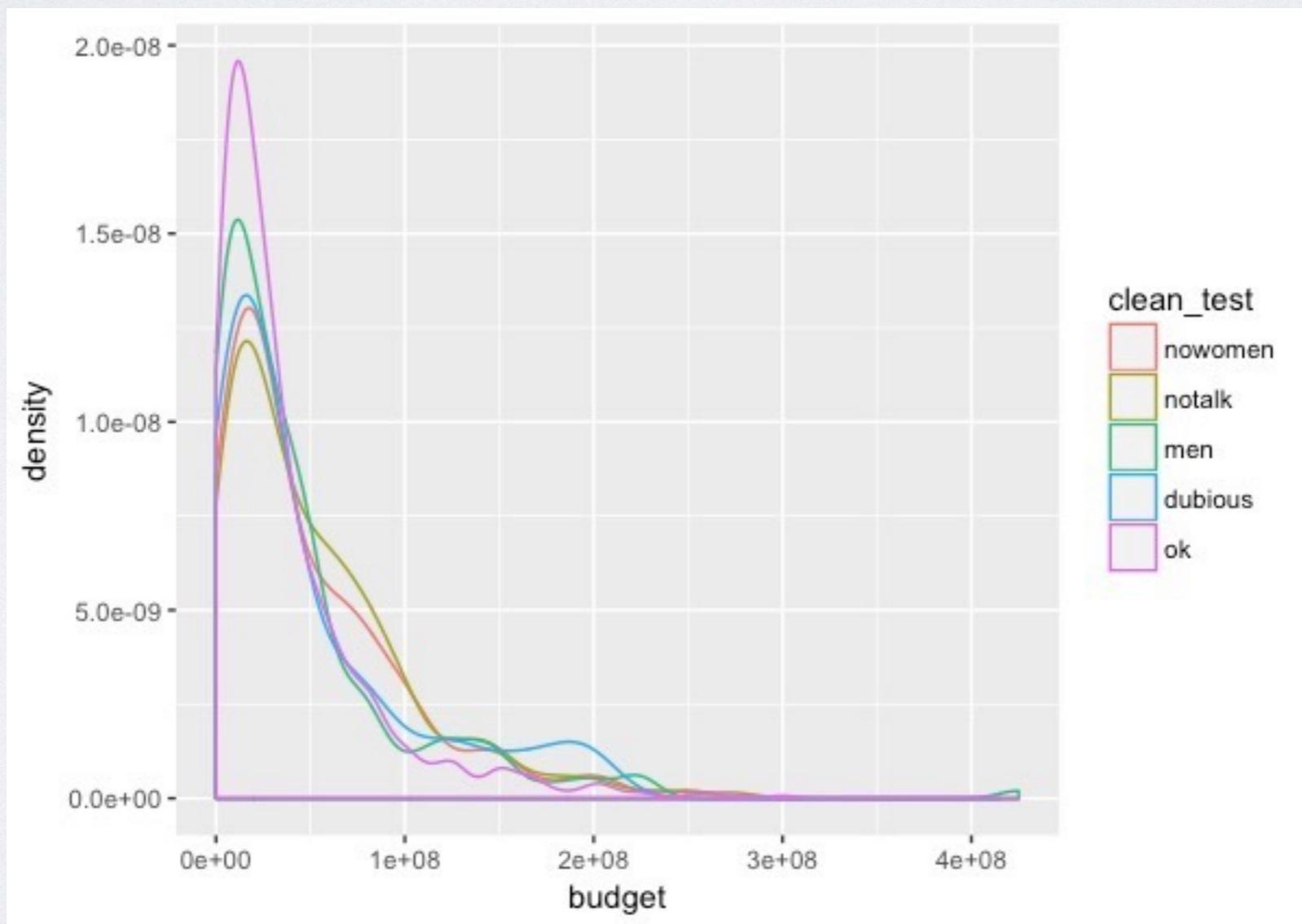


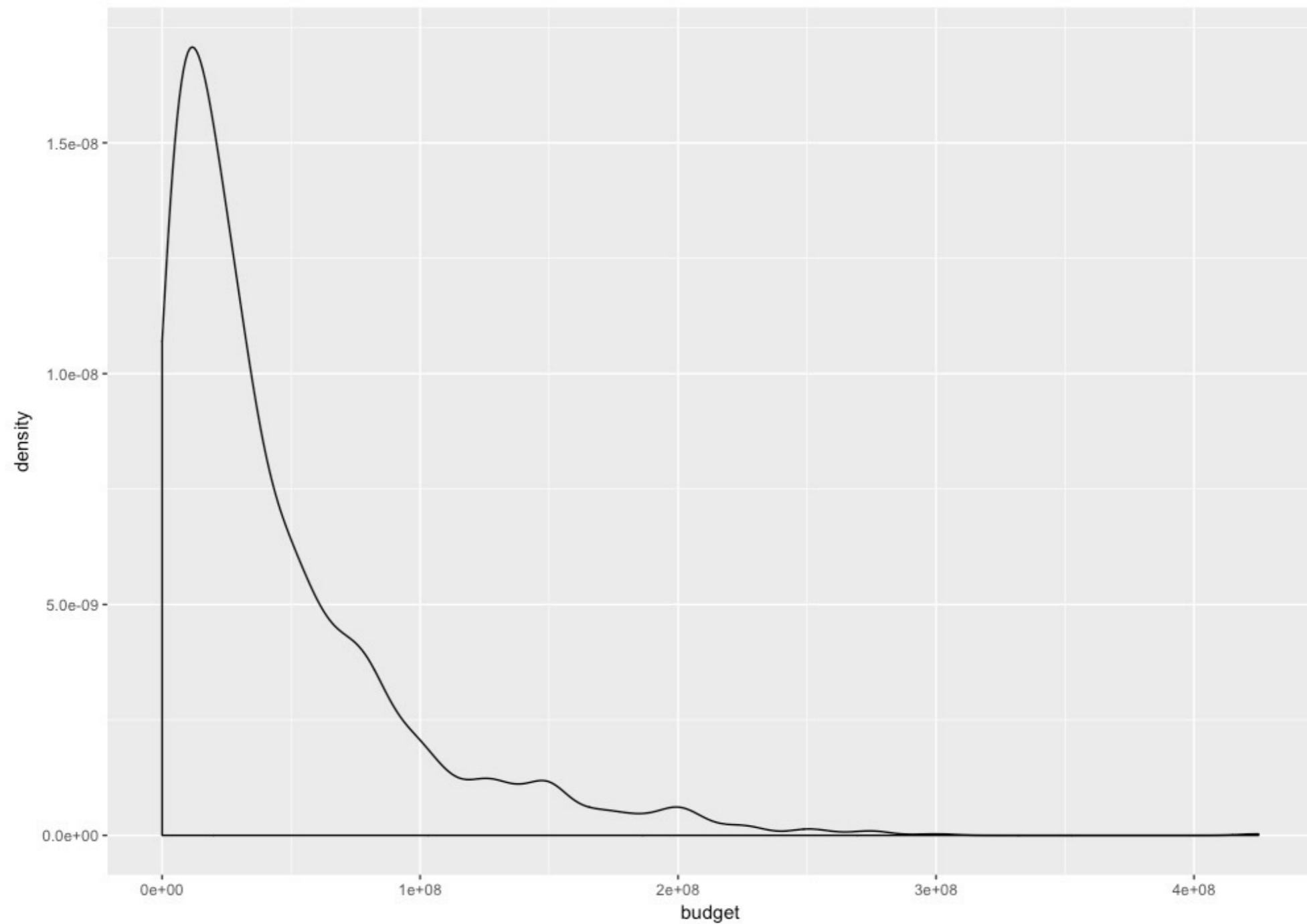


```
ggplot(data = bechdel) +  
  geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```

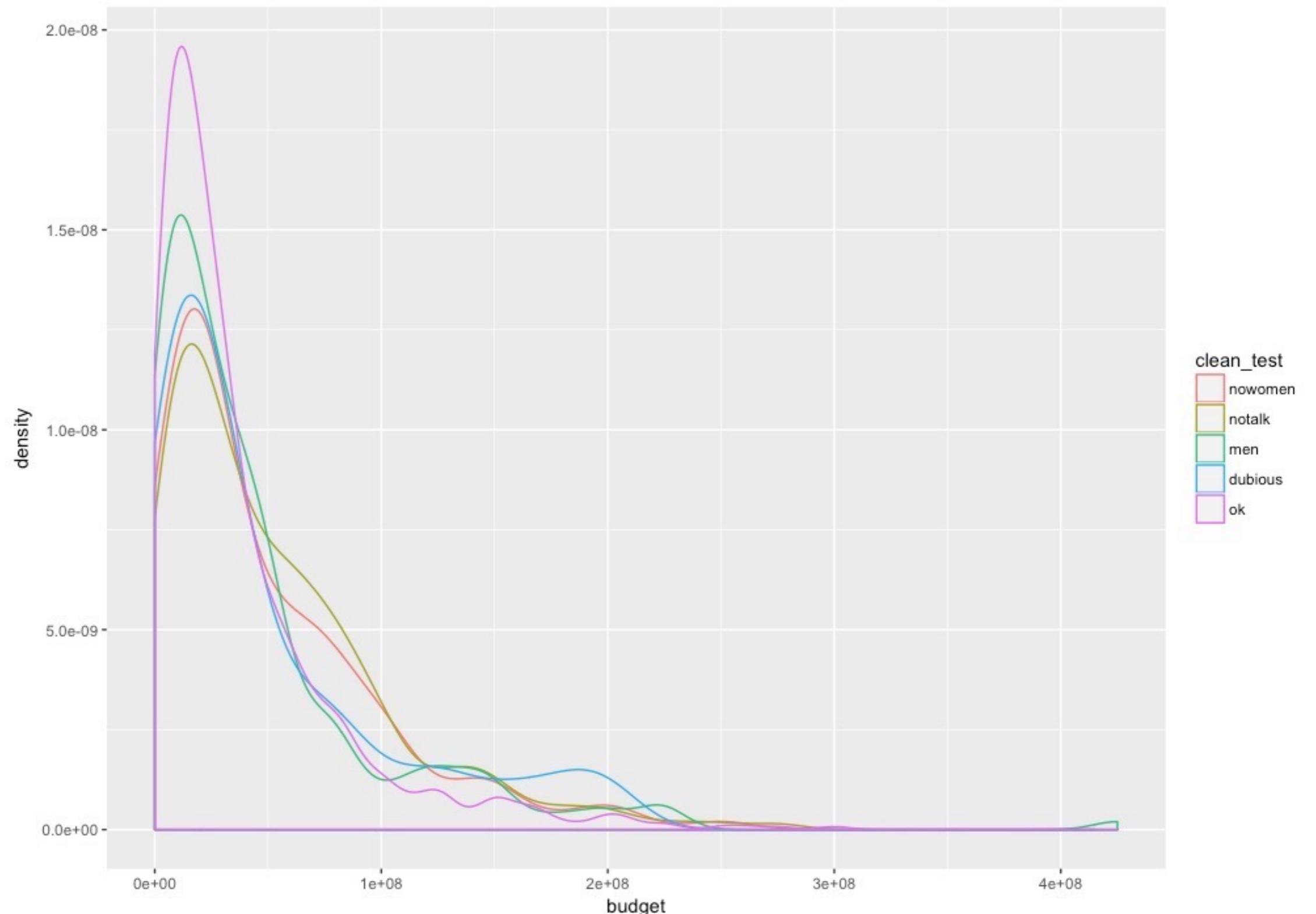
# Your Turn 6

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))
```



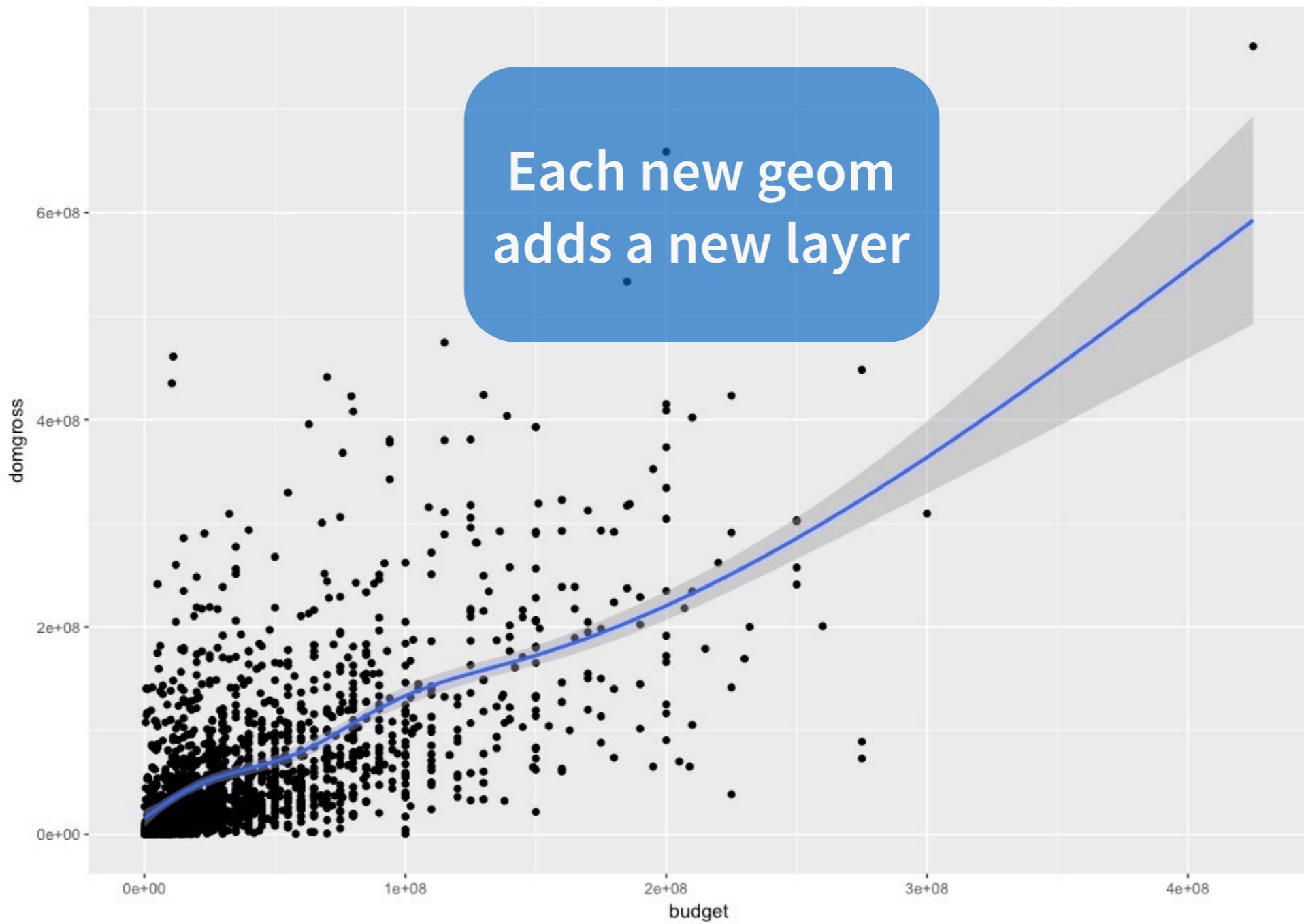
```
ggplot(data = bechdel) +  
  geom_density(mapping = aes(x = budget, color=clean_test))
```

# Your Turn 7

With your partner, 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. On the left, the code editor displays an Rmd file named '02-Visualization.Rmd'. A red circle highlights the 'Preview' button in the toolbar above the code editor. The main workspace on the right 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 scatter plot.

```
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
```

# 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 View (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  
7:1 (Top Level) R Markdown
```

**Console View (Bottom Left):**

```
~/Dropbox/Intro_to_R_and_RStudio/master-the-tidyverse-master/  
+ geom_point(aes(budget, domgross))  
+ geom_smooth(aes(budget, domgross))  
`geom_smooth()` using method = 'gam'  
Warning messages:  
1: Removed 17 rows containing non-finite values  
(stat_smooth).  
2: Removed 17 rows containing missing values  
(geom_point).  
>
```

**Preview View (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

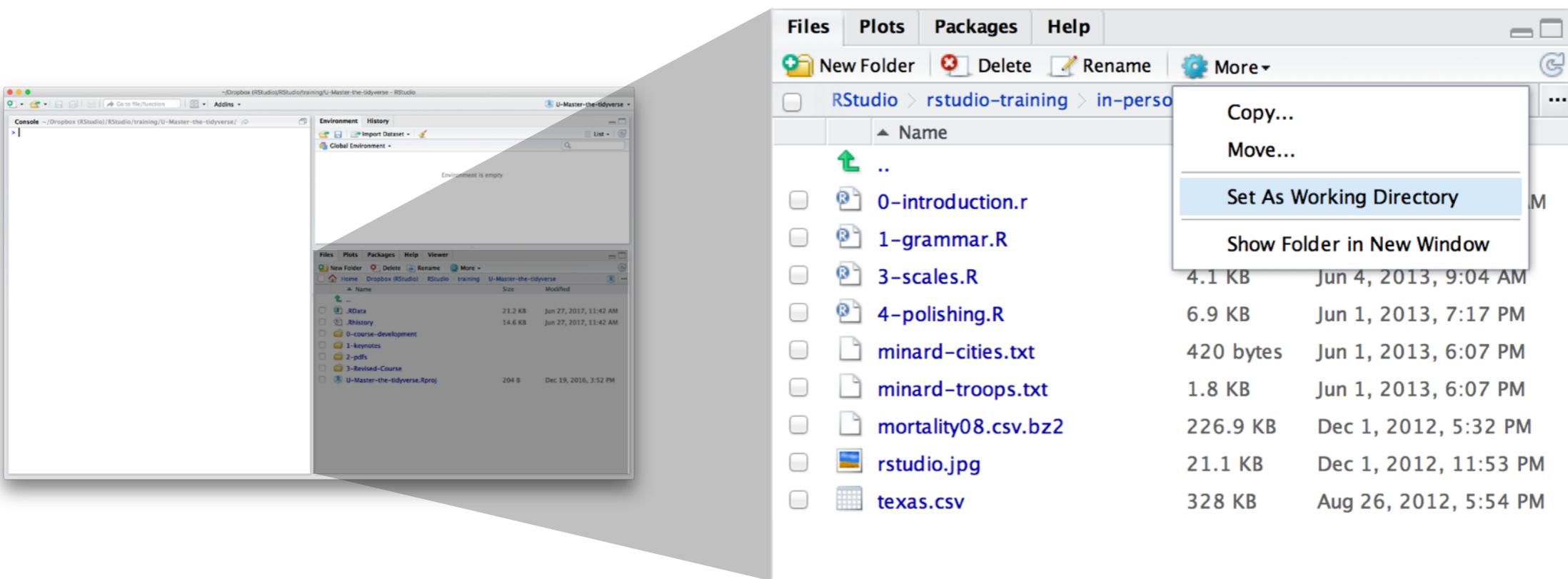
R associates itself with a folder (i.e. directory) on your computer.

- 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()`

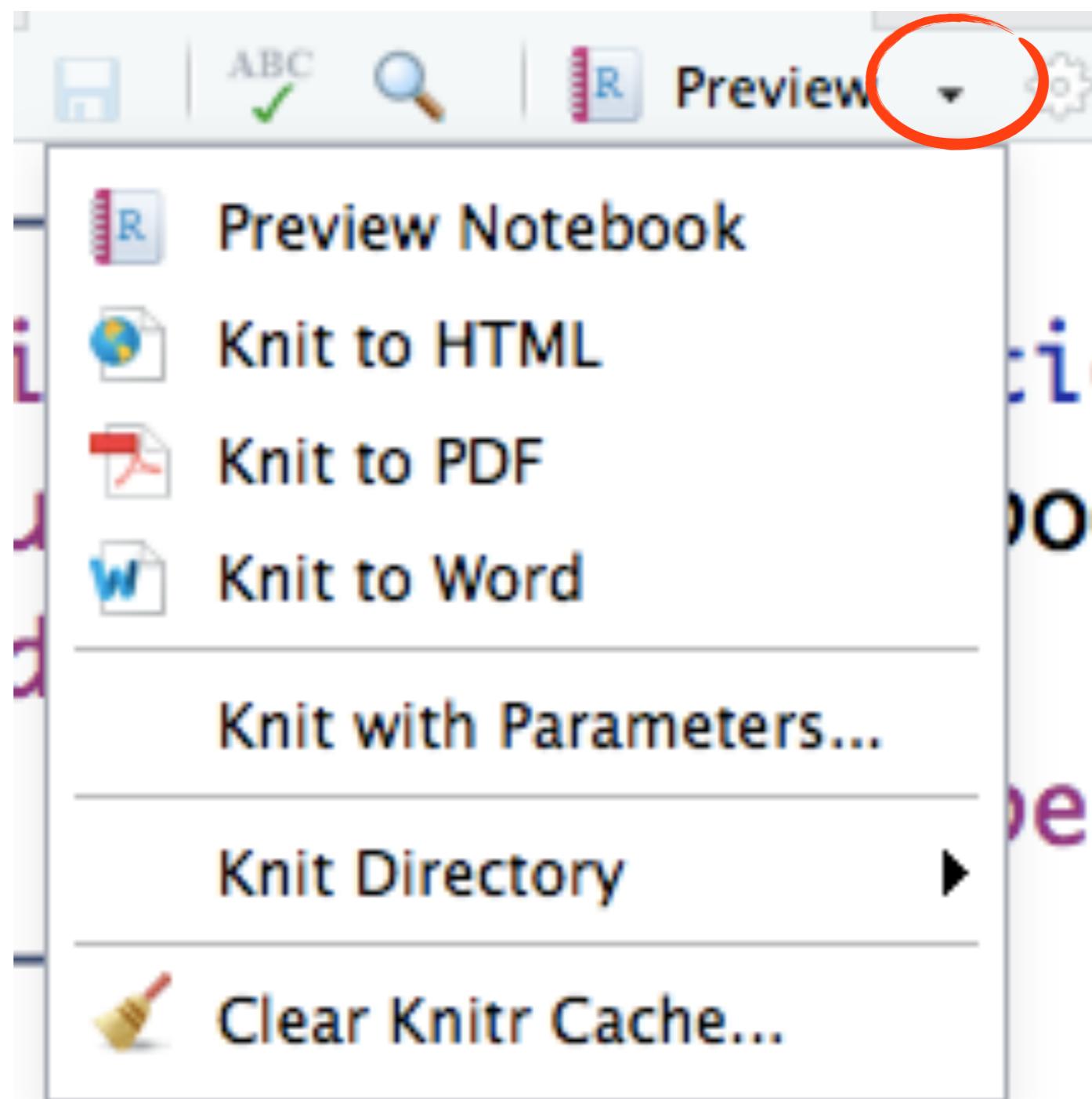
# Changing the Working directory

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



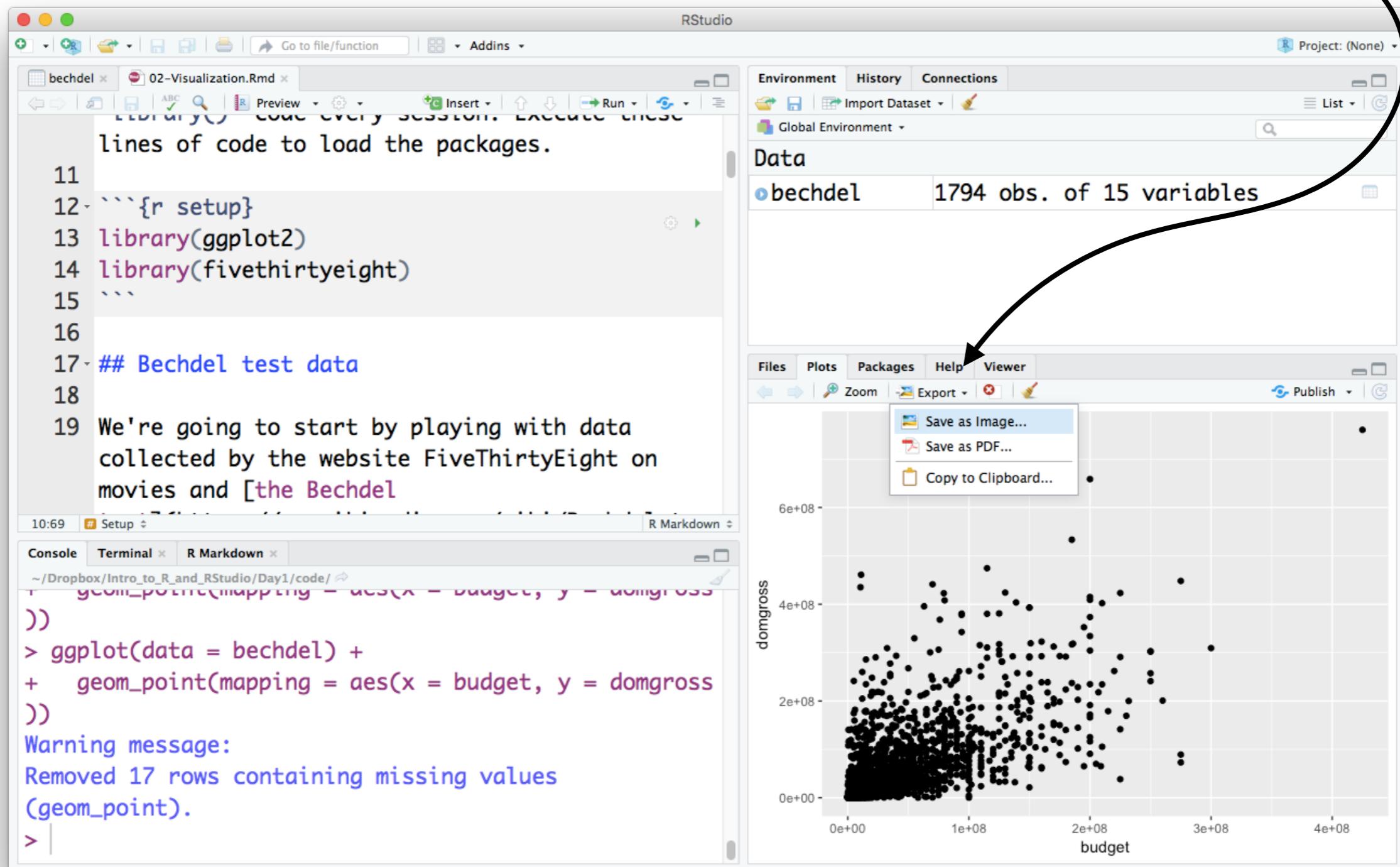
# "knitting" a document

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")
```

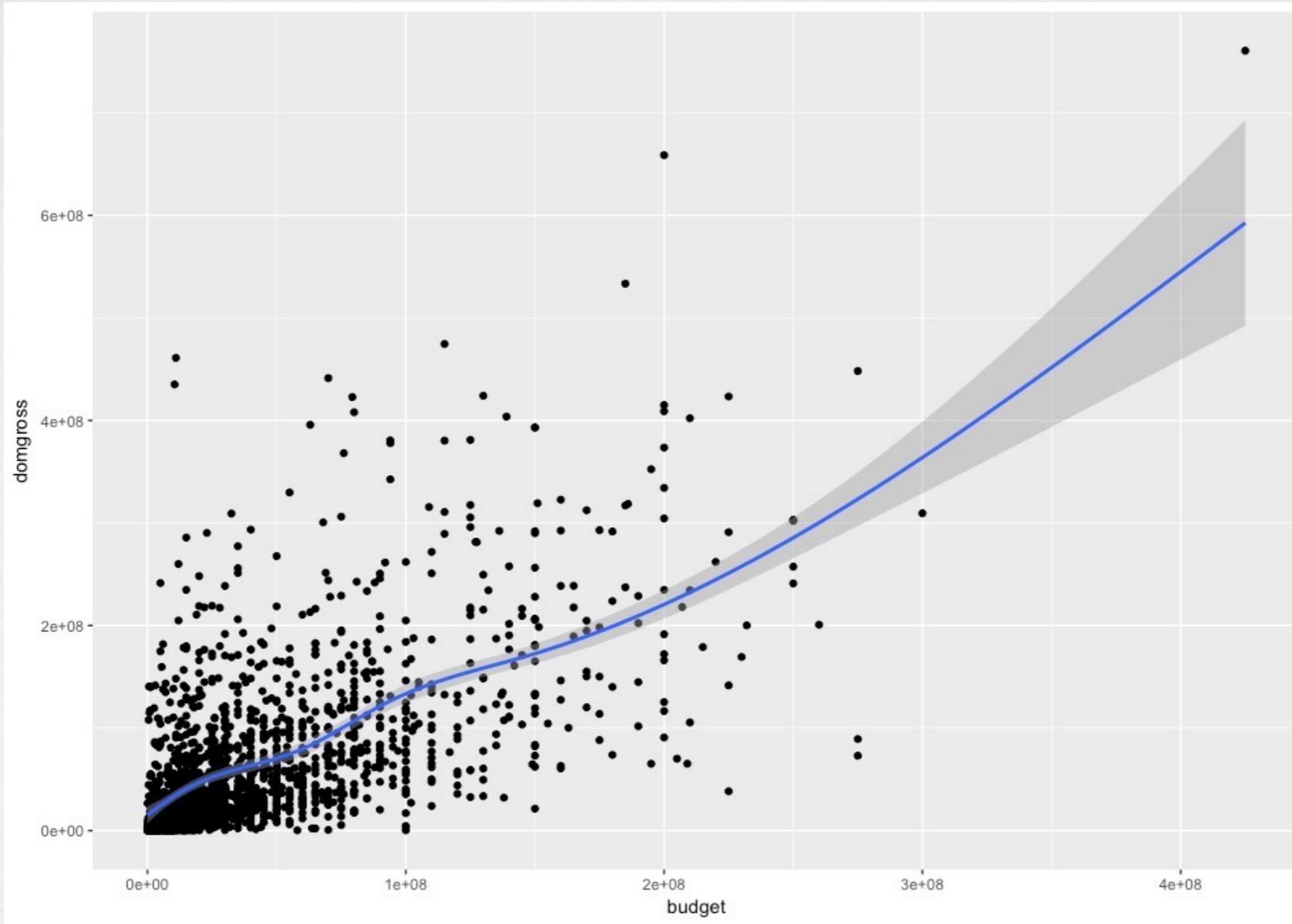
```
ggsave("my-plot.png")
```

Specify size in inches

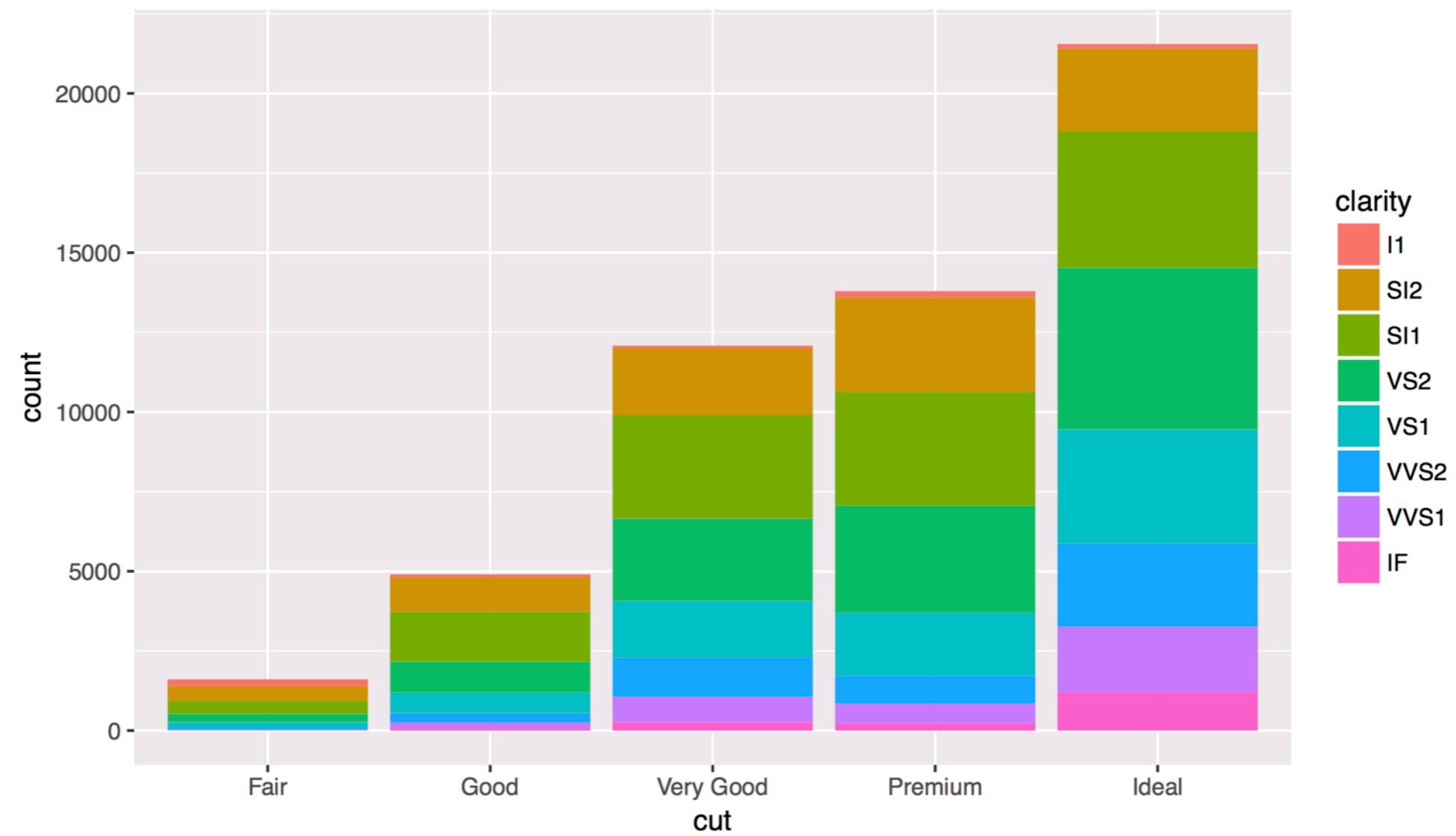
```
ggsave("my-plot.pdf", width = 6, height = 6)
```

# Your Turn 9

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

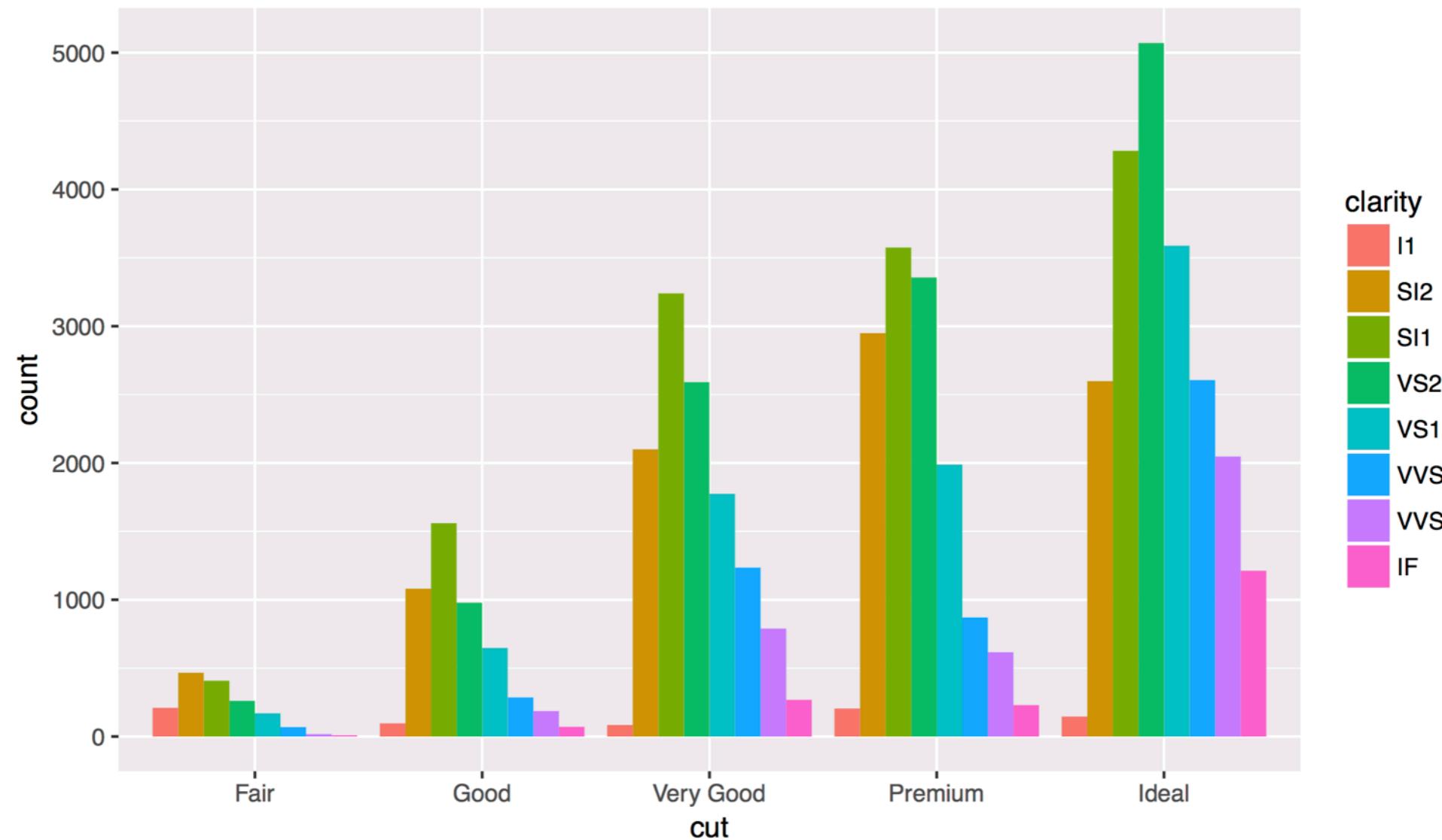


What else?



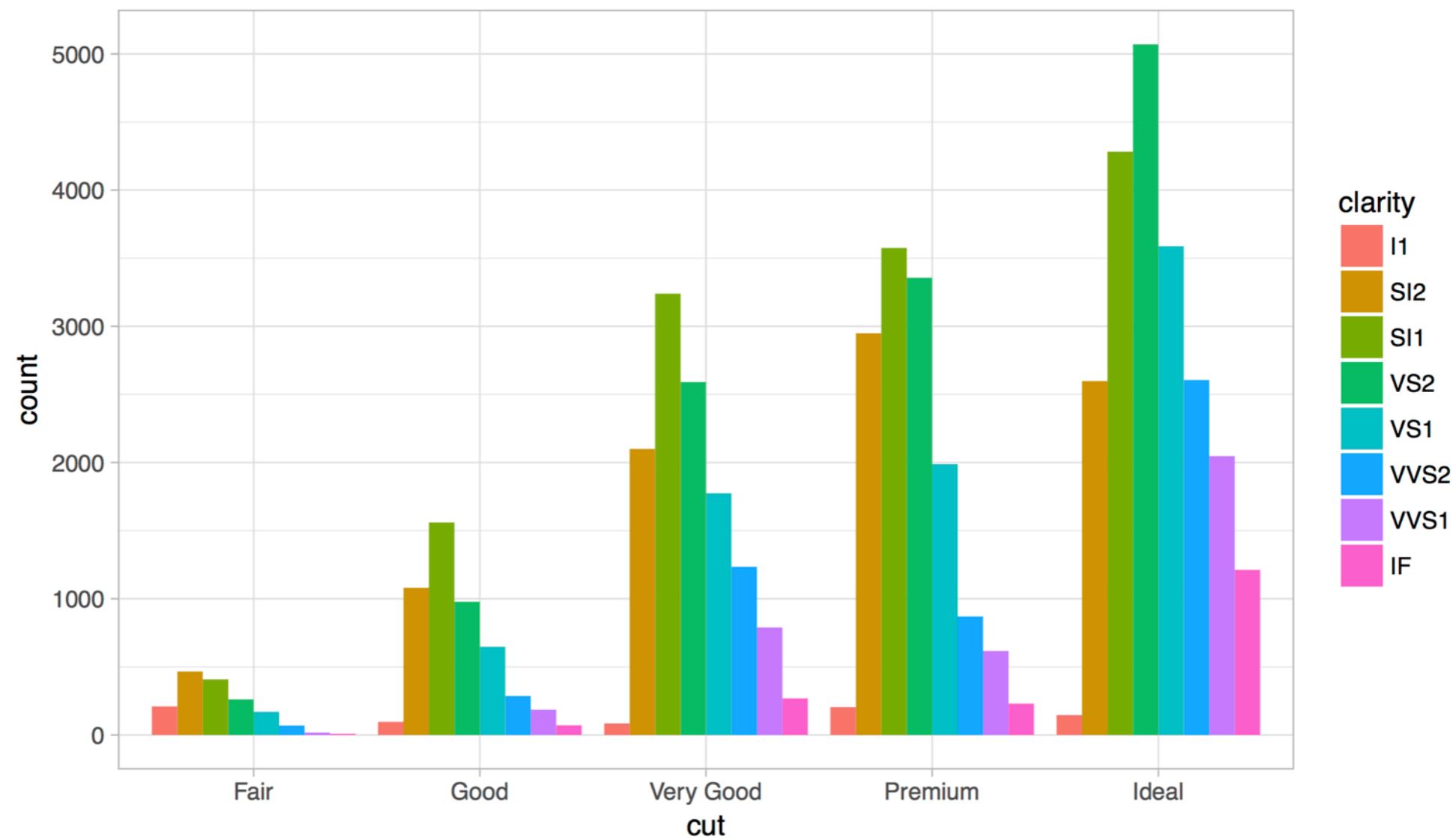
# Position Adjustments

## How overlapping objects are arranged



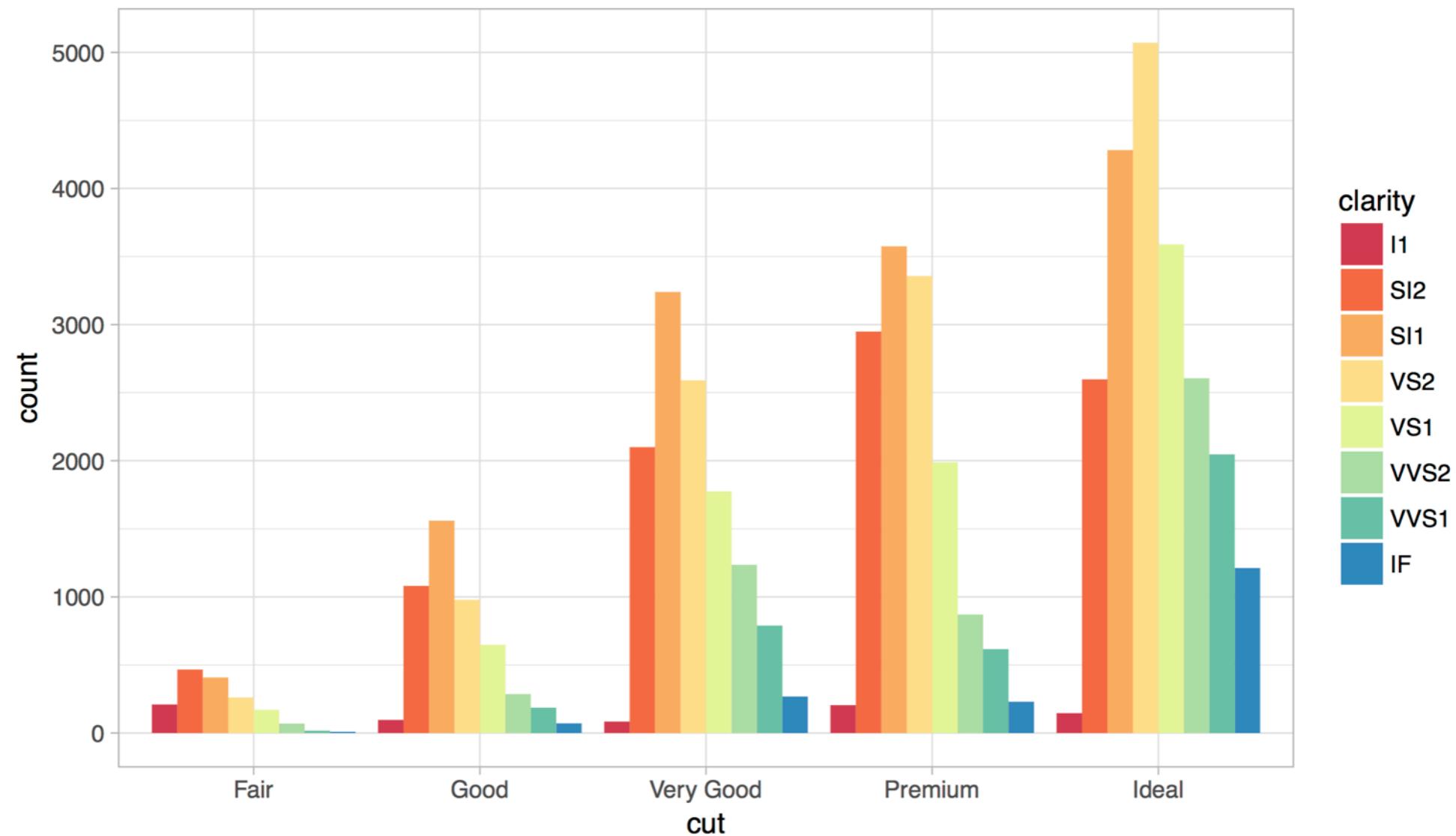
# Themes

## Visual appearance of non-data elements



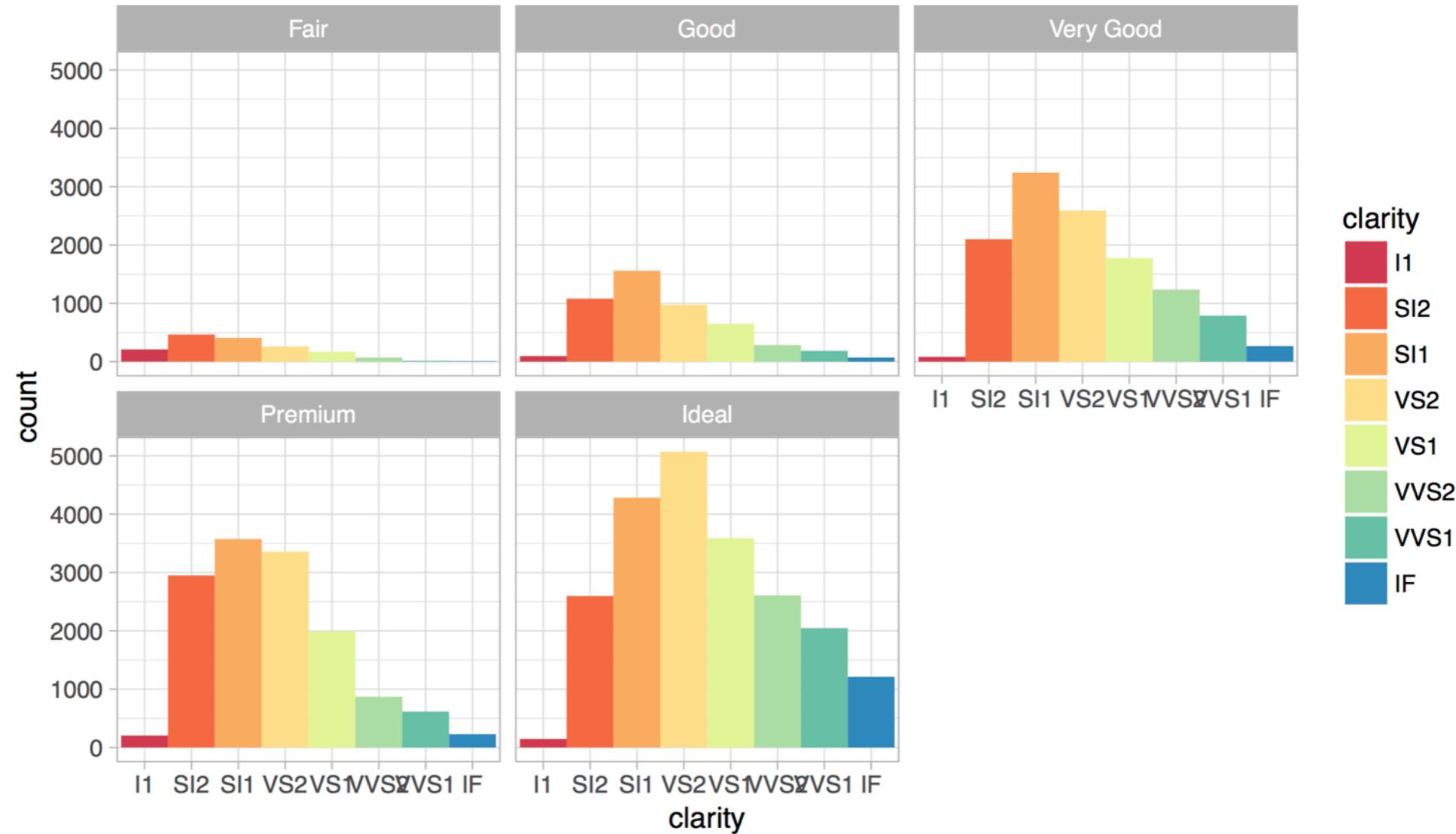
# Scales

## Customize color scales, other mappings

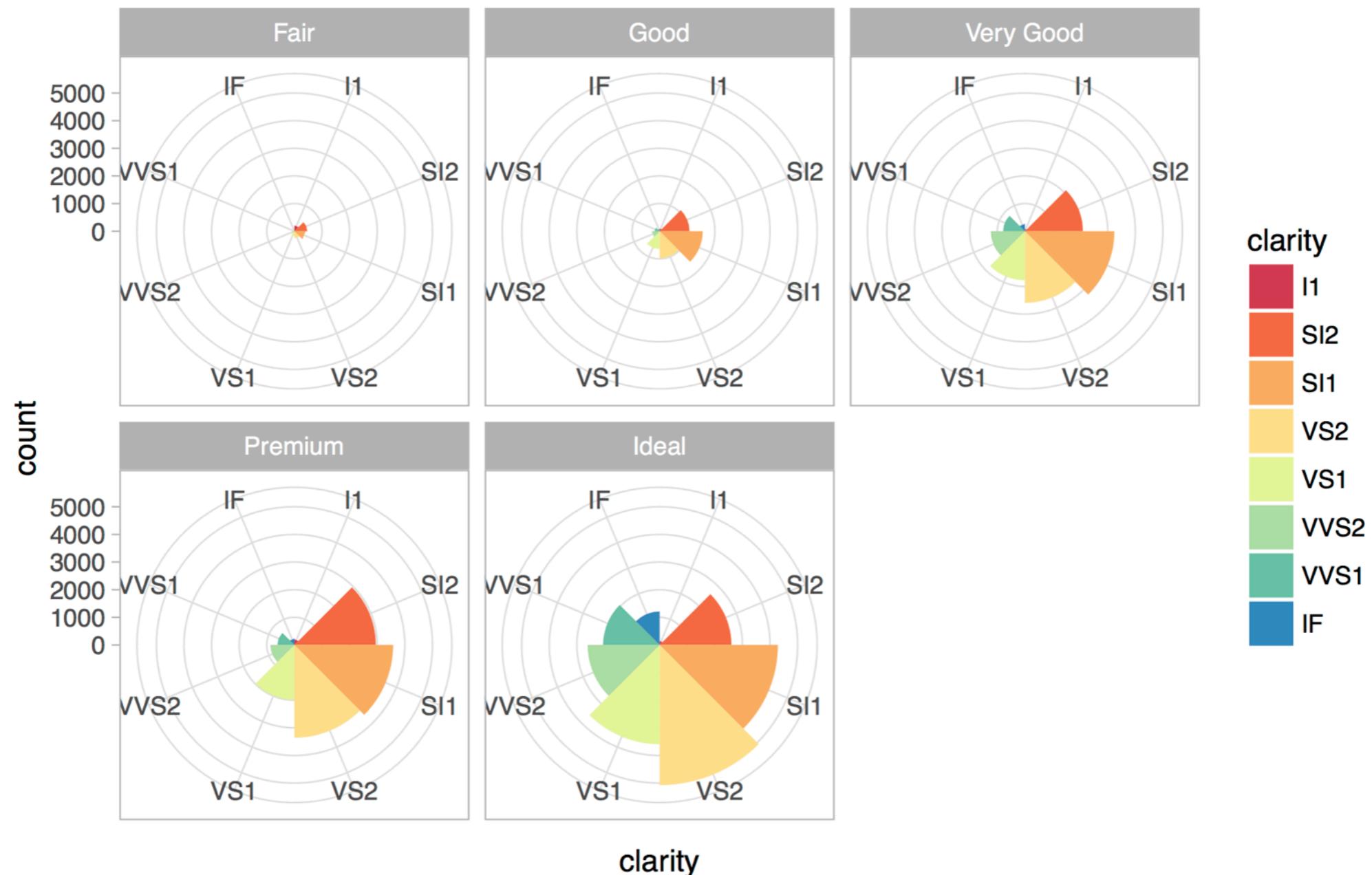


# Facets

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

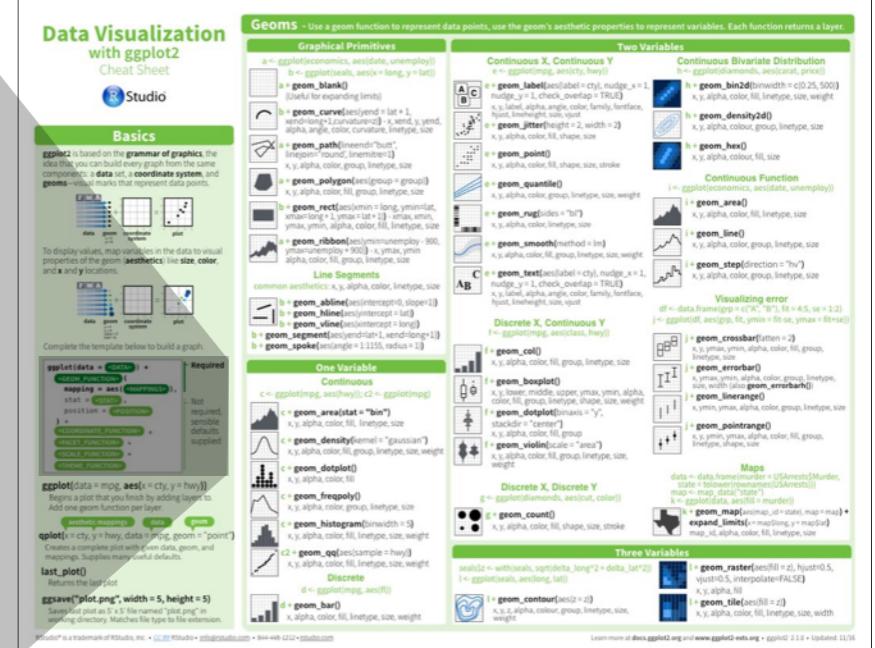
# A ggplot2 template

## Make any plot by filling in the parameters of this template

```
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

ggplot2 part of the tidyverse

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()
```

Links

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

License

GPL-2 | file [LICENSE](#)

Developers

[Hadley Wickham](#)  
Author, maintainer

[Winston Chang](#)  
Author

[All authors...](#)

Dev status