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

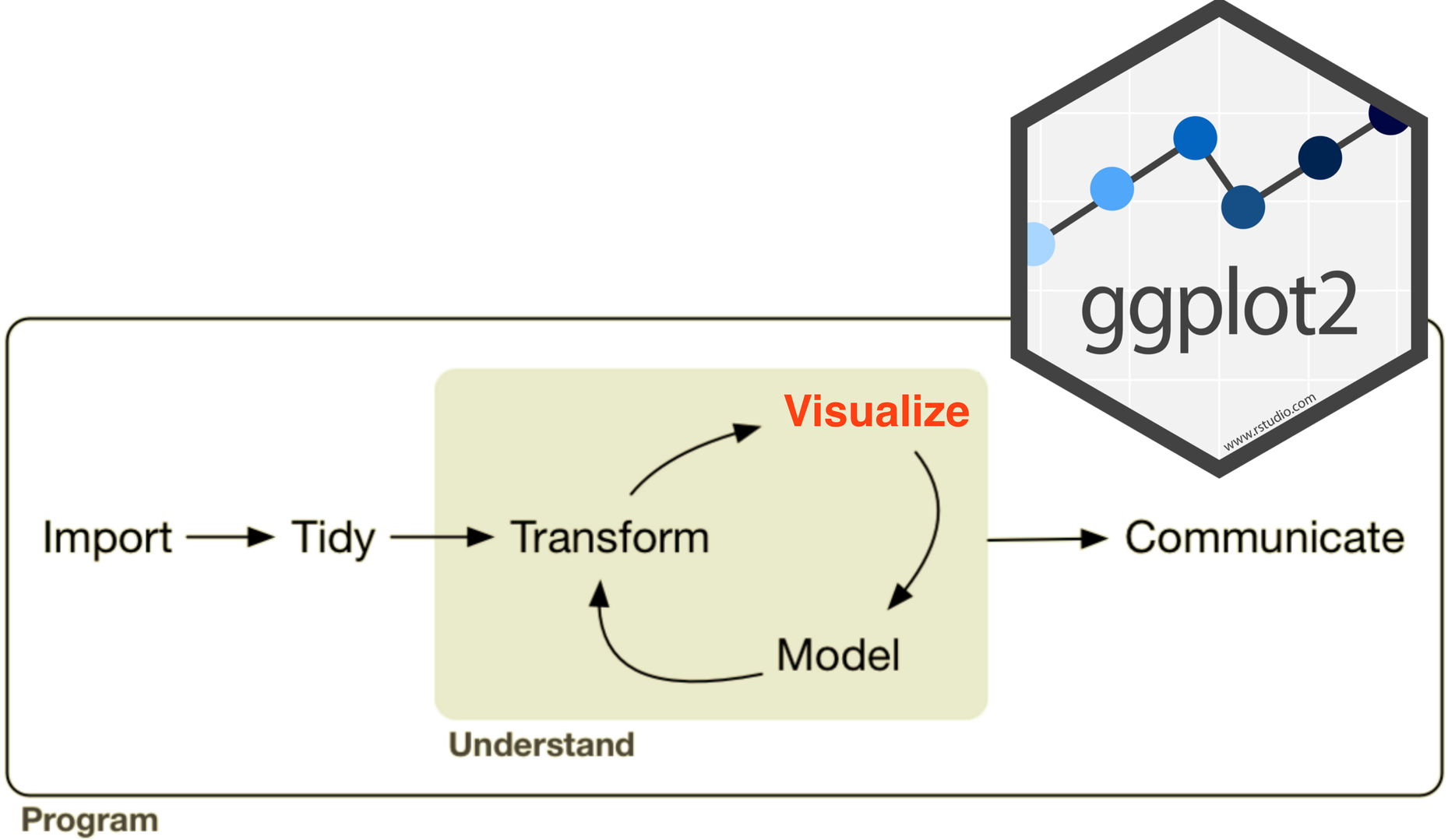
Visiting Assistant Professor of Statistical and Data Sciences

Smith College

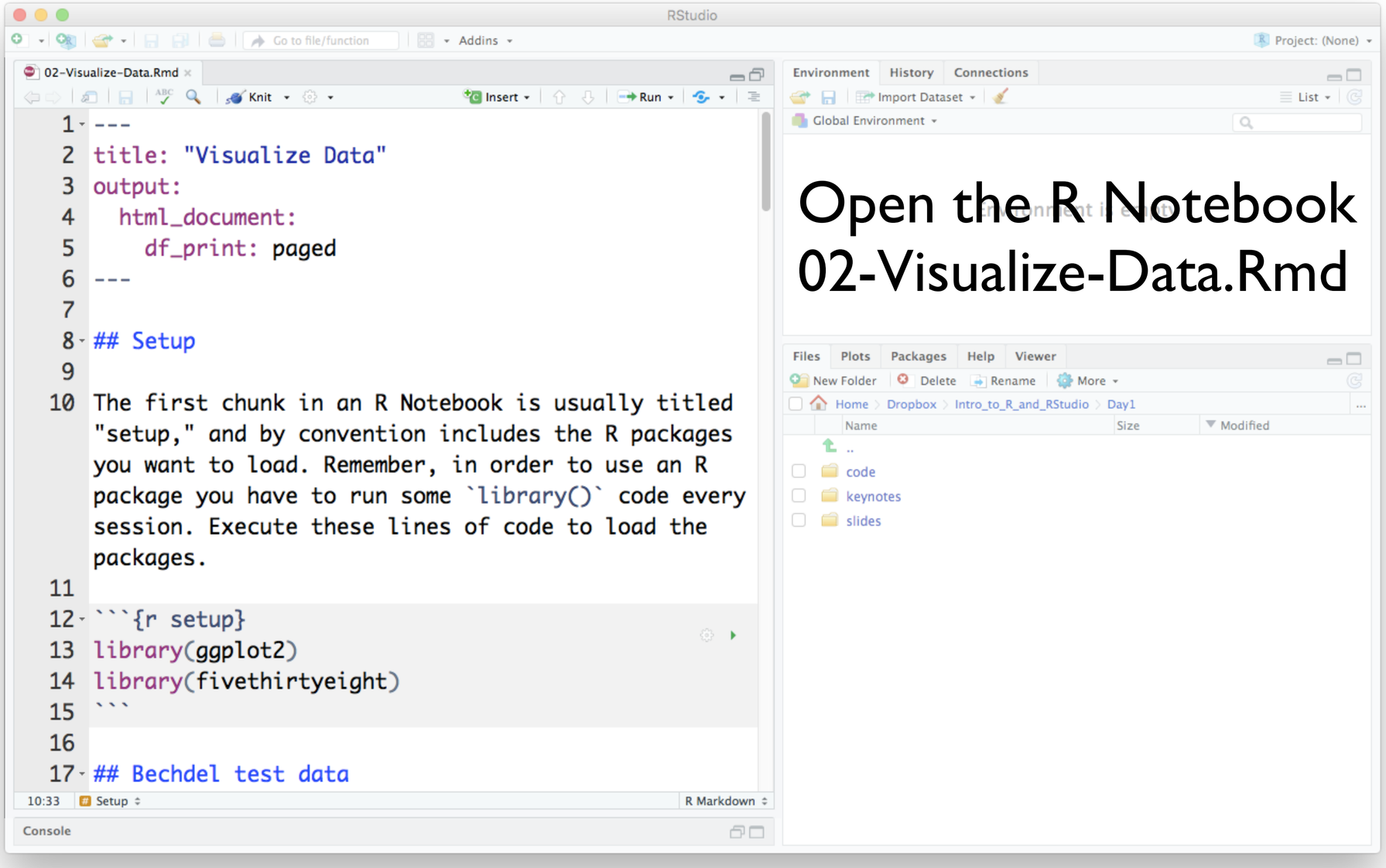
rstudio::conf 2018

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

Data visualization (good stuff first)



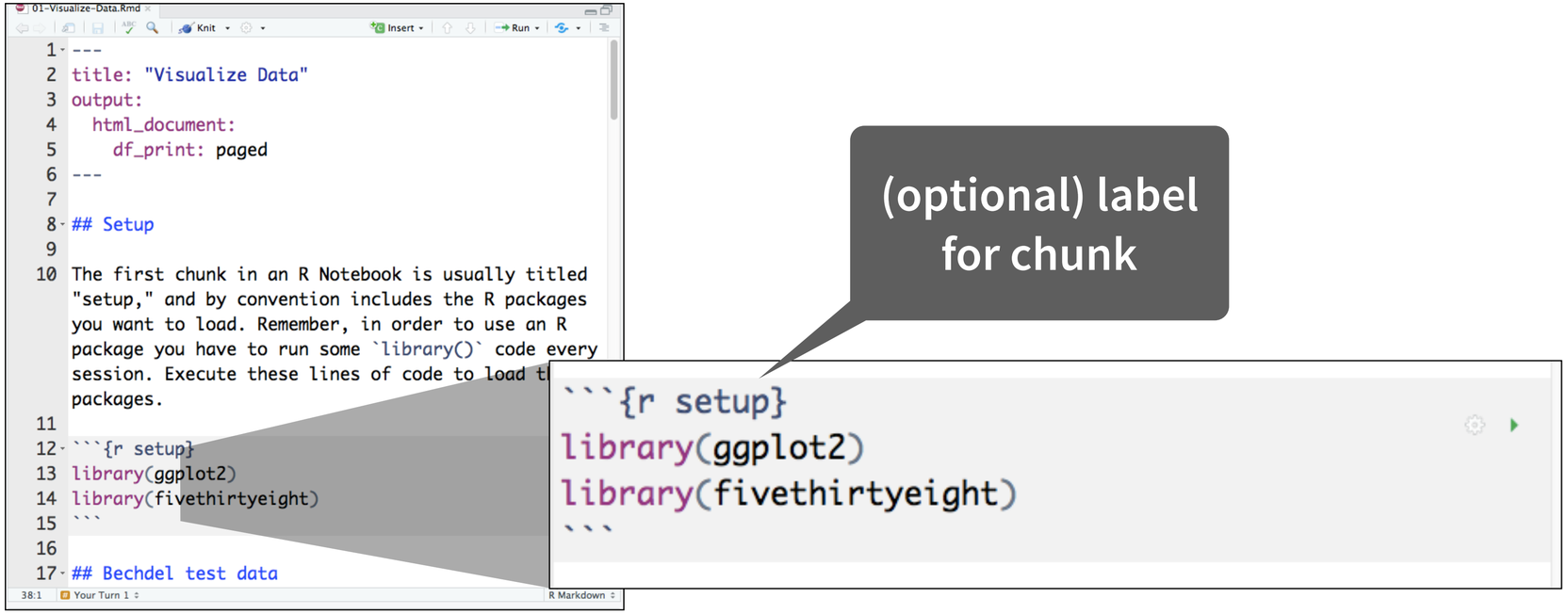
From *R for Data Science* by Hadley Wickham and Garrett Grolemund.


RStudio
Go to file/function
Addins
Project: (None)
02-Visualize-Data.Rmd
Knit
Insert
Run
---
title: "Visualize Data"
output:
html_document:
df_print: paged

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.
```{r setup}
library(ggplot2)
library(fivethirtyeight)
```
Bechdel test data
10:33
Setup
R Markdown
Console
Environment
History
Connections
Import Dataset
List
Global Environment
Open the R Notebook
02-Visualize-Data.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



bechdel

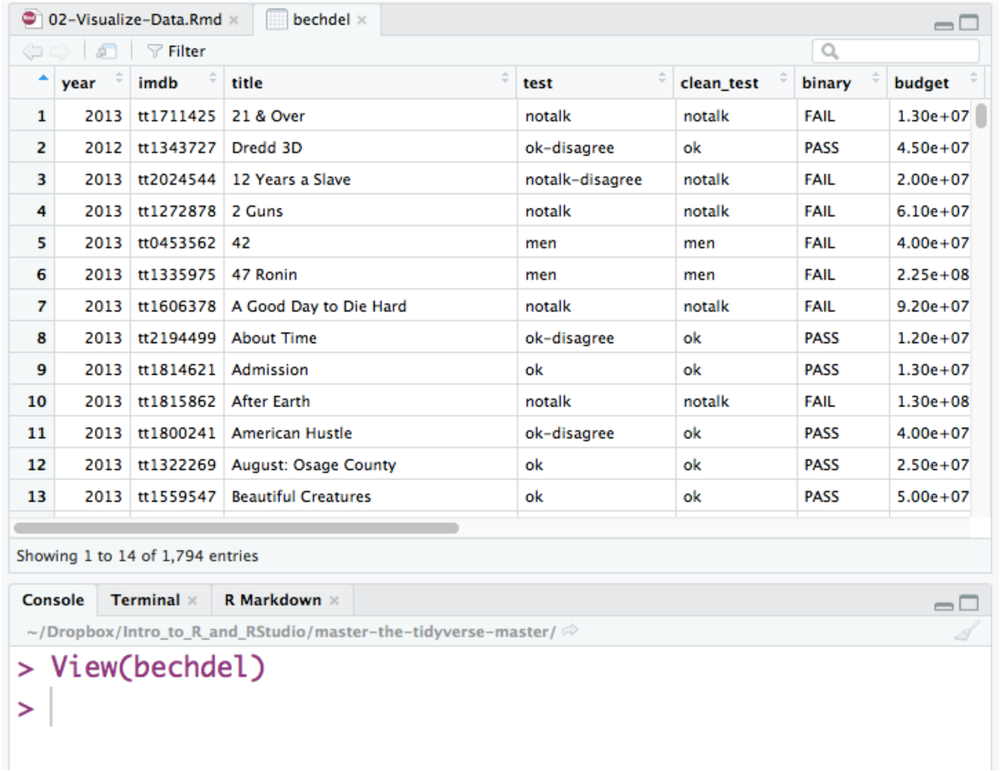
Data on movies and the Bechtel test

```
bechdel
```

bechdel

Data on movies and the Bechtel test

```
View(bechdel)
```



	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

Type this in your Console, NOT in your Notebook

Consider

Confer with the people around you.

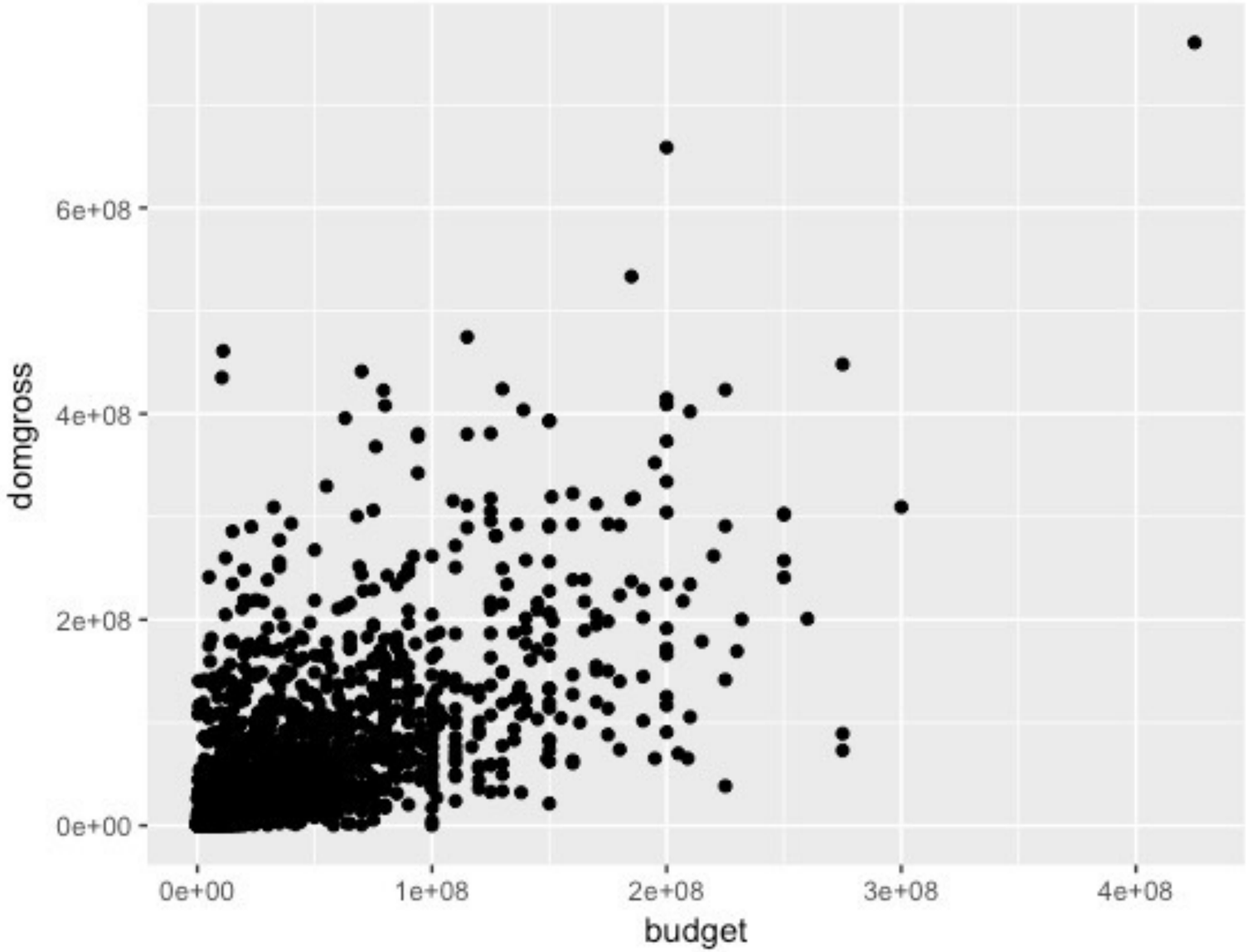
What relationship do you expect to see between movie budget (`budget`) and domestic gross(`domgross`)?

Your Turn 1

Run this code in your notebook to make a graph.

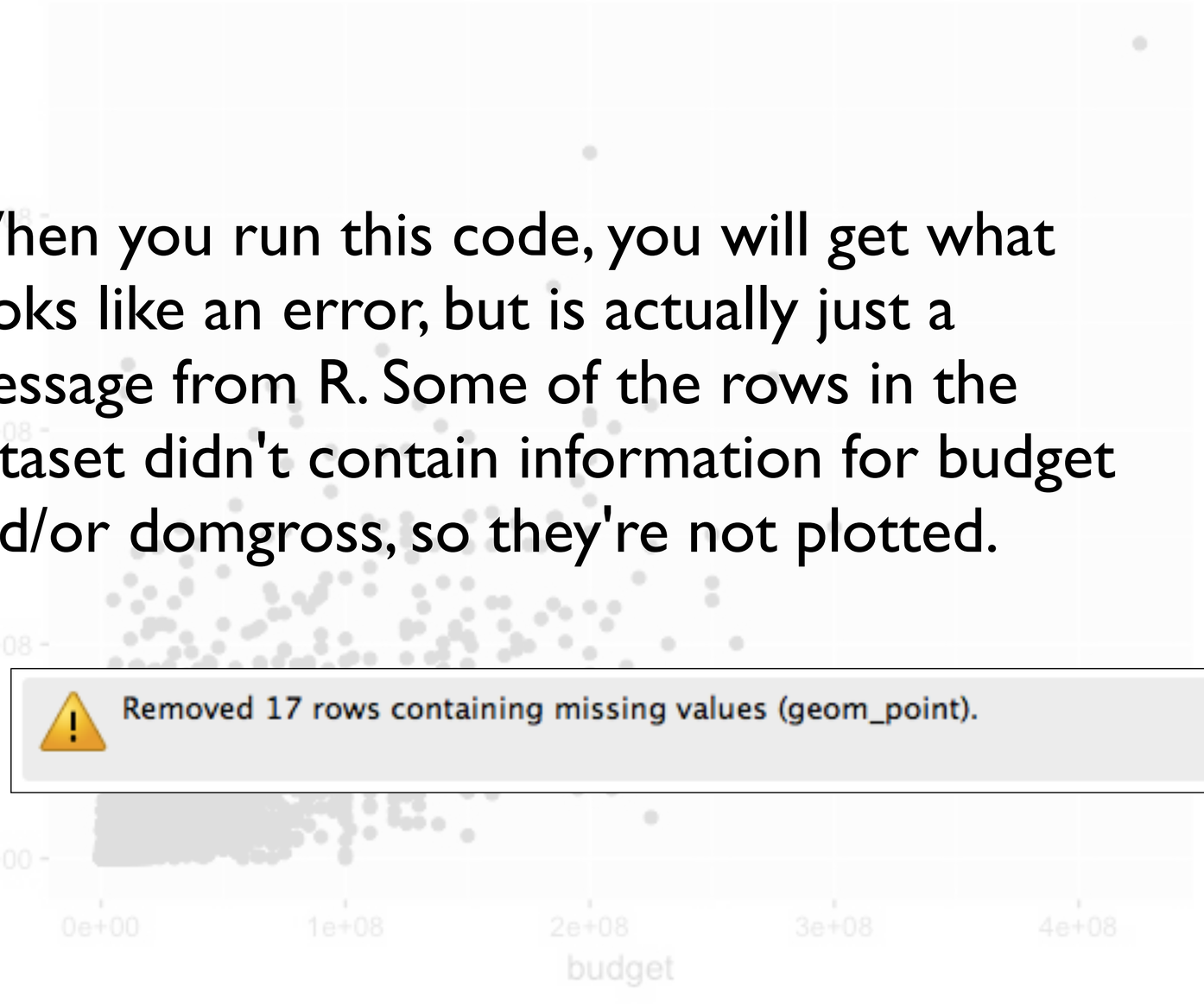
Pay strict attention to spelling, capitalization, and parentheses!

```
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.



⚠ Removed 17 rows containing missing values (geom_point).

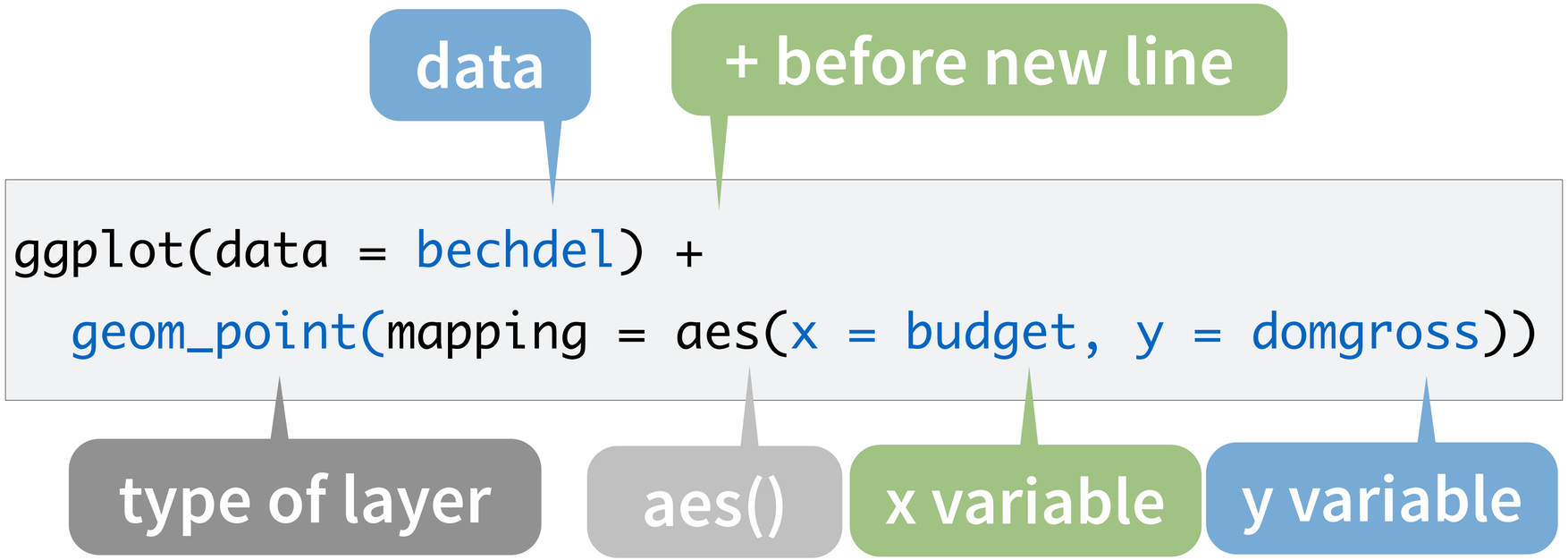
```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = budget, y = domgross))
```

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>))
```

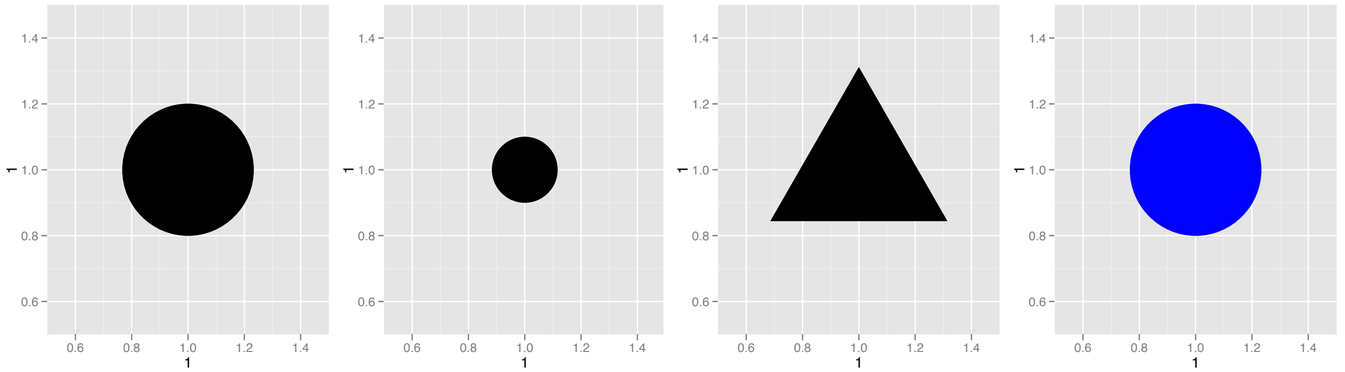
```
    geom_point(mapping = aes(x = budget, y = domgross))
```

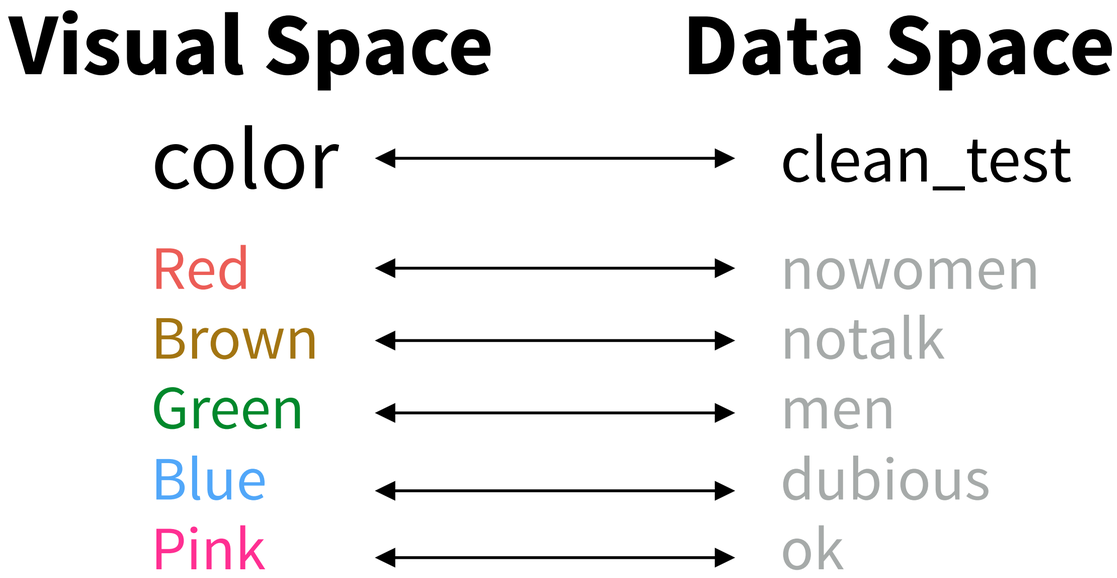
A template

```
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

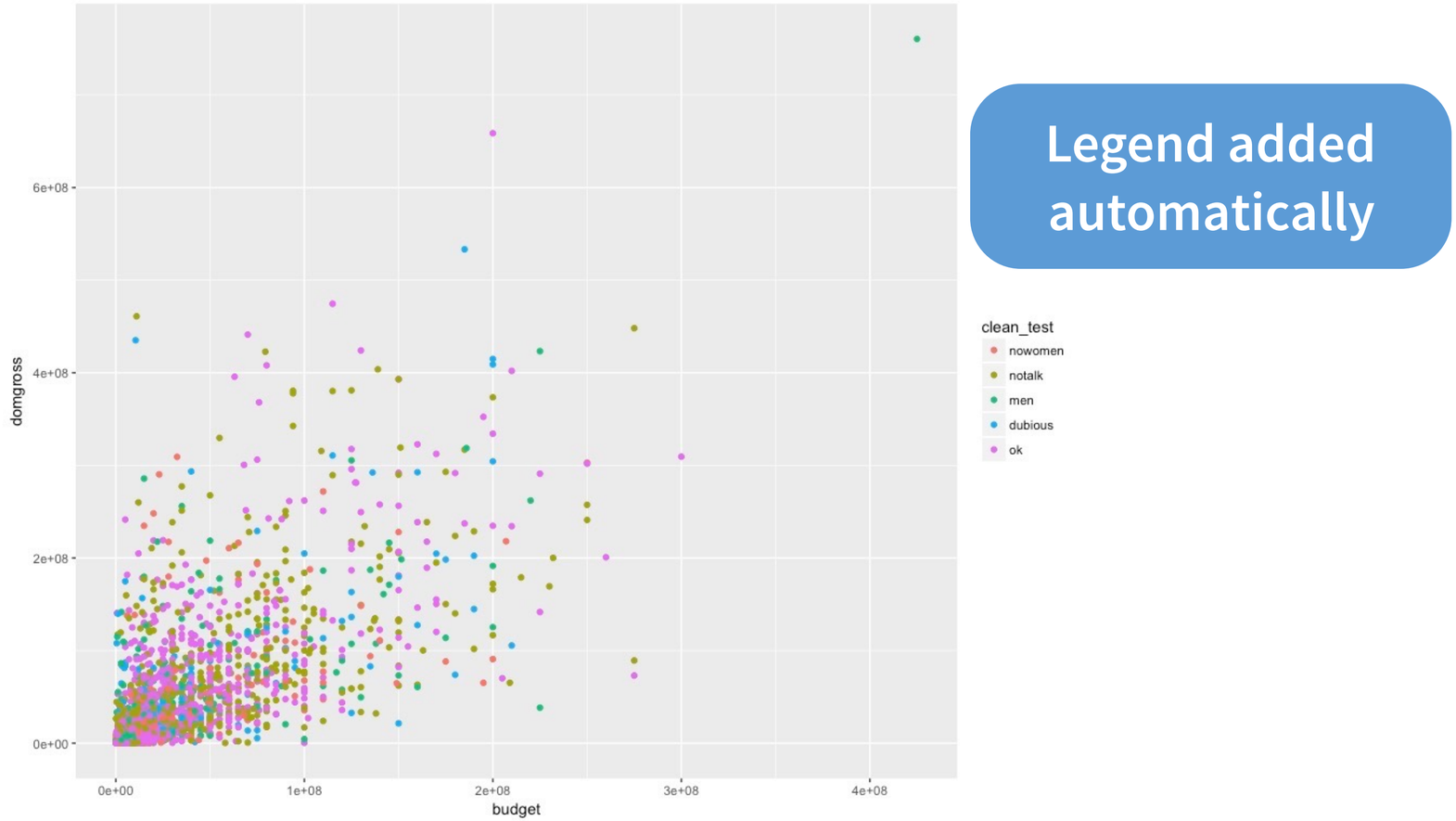
Aesthetics

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

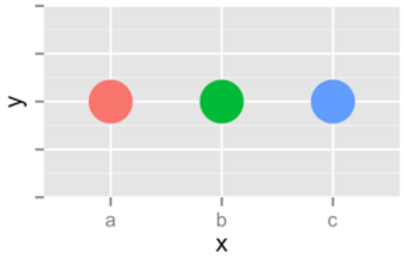
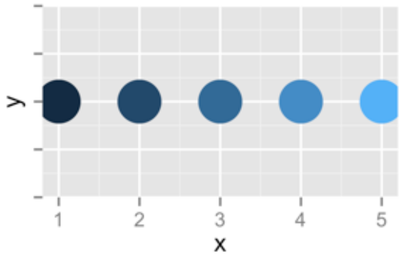
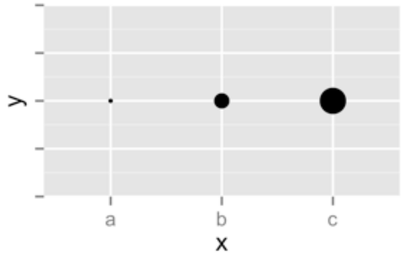
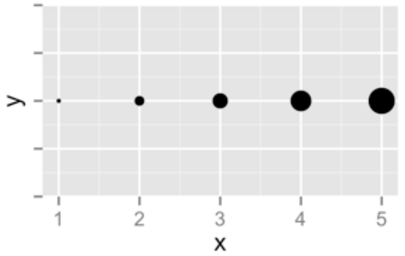
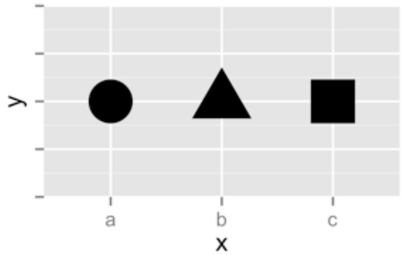
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

Your Turn 2

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

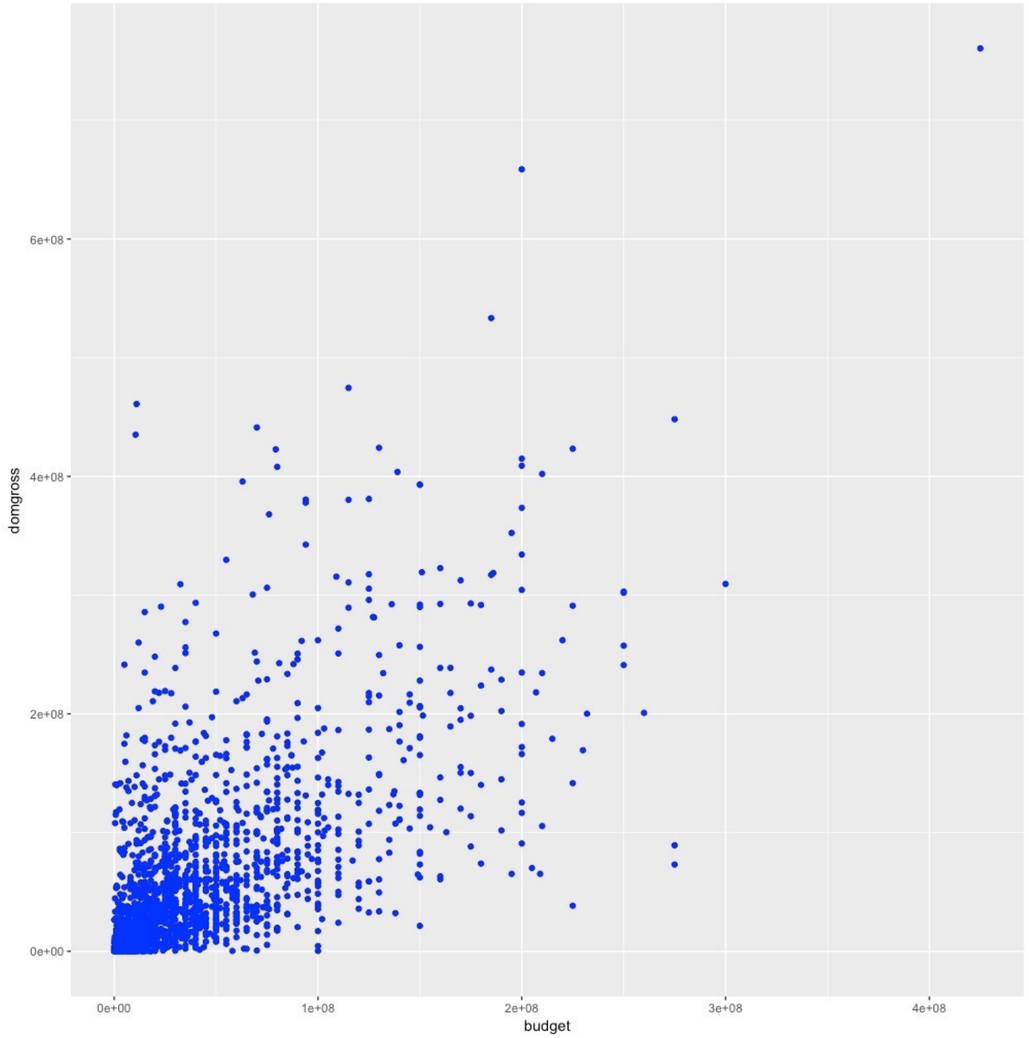
Do different things happen when you map aesthetics to discrete and continuous variables?

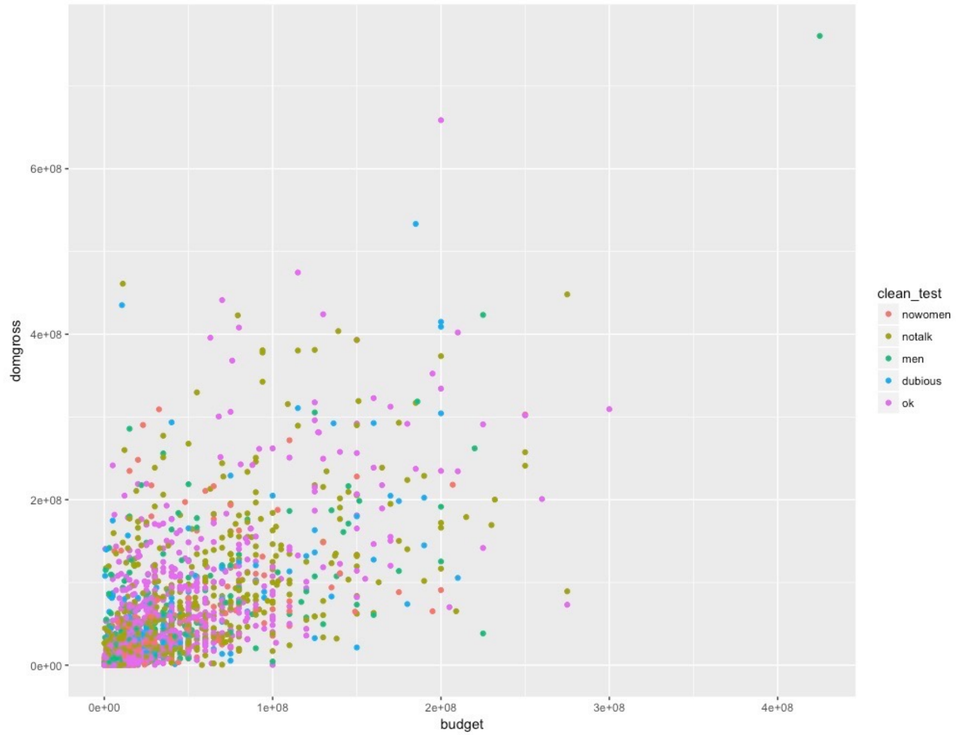
What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

set vs. map

How would you make this plot?

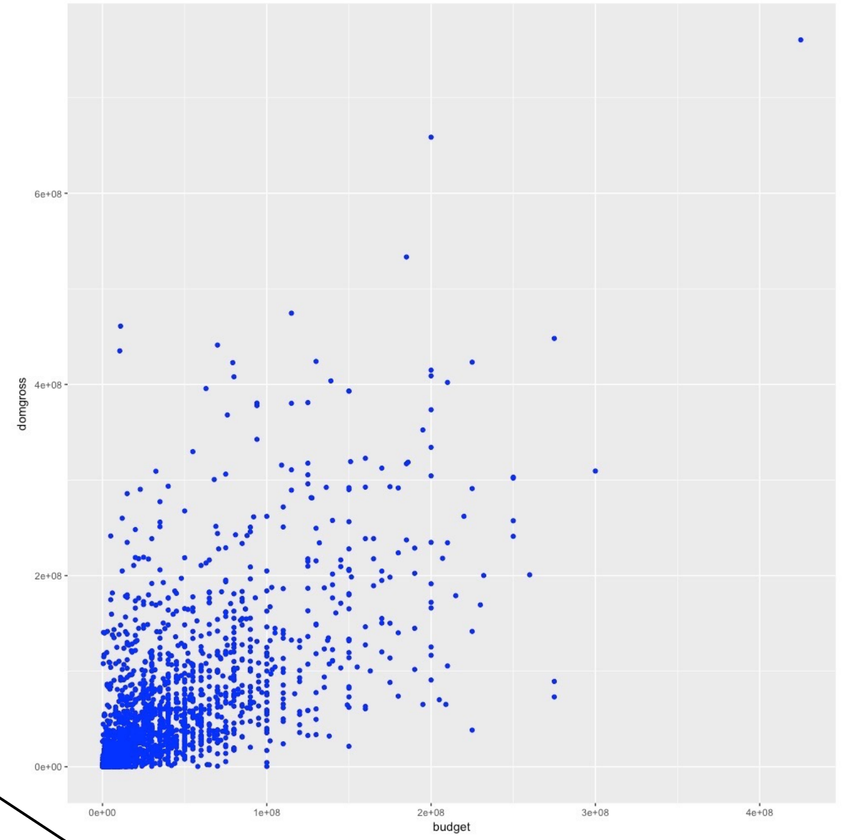




Inside of aes(): maps an aesthetic to a variable

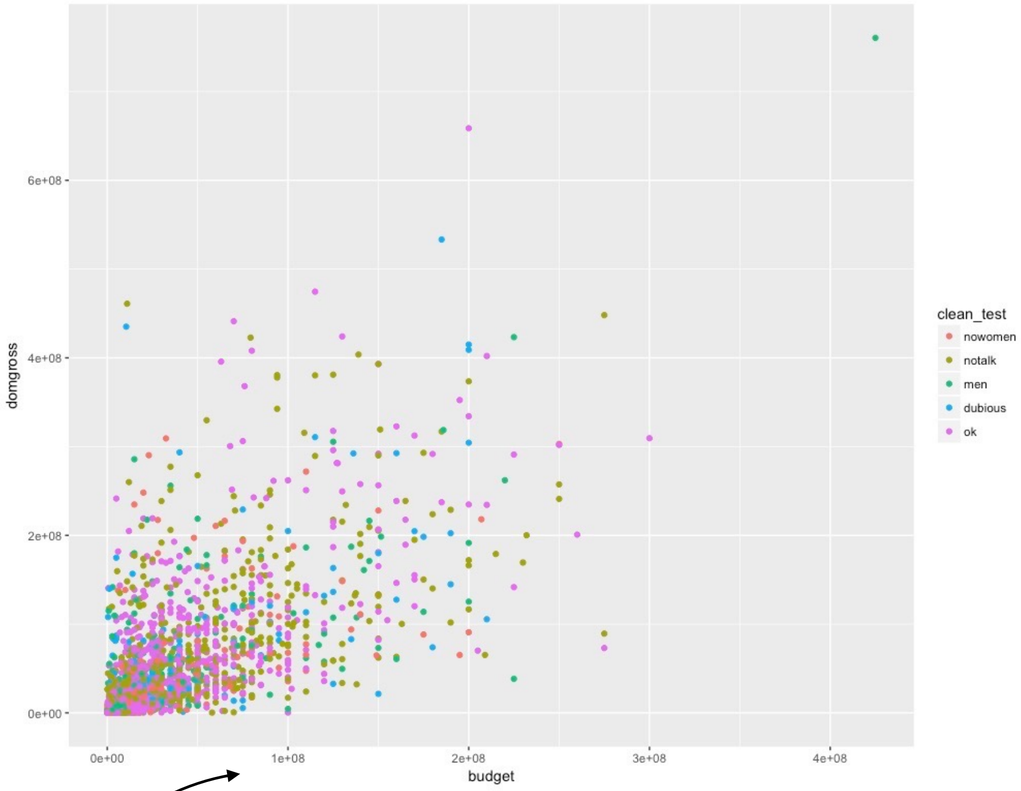
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

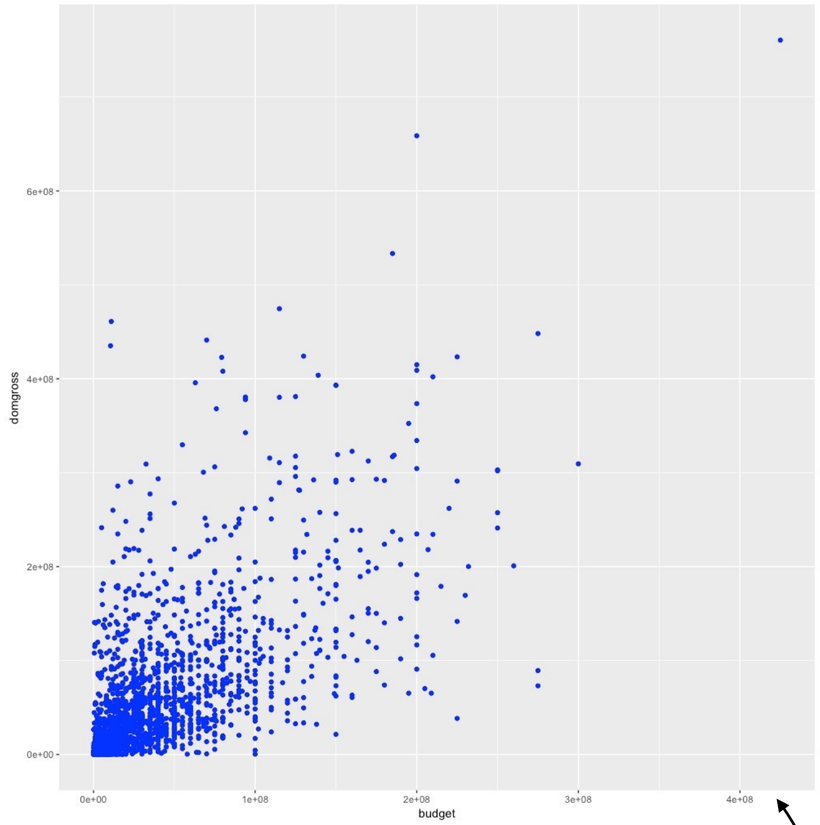
Outside of aes(): sets an aesthetic to a value




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

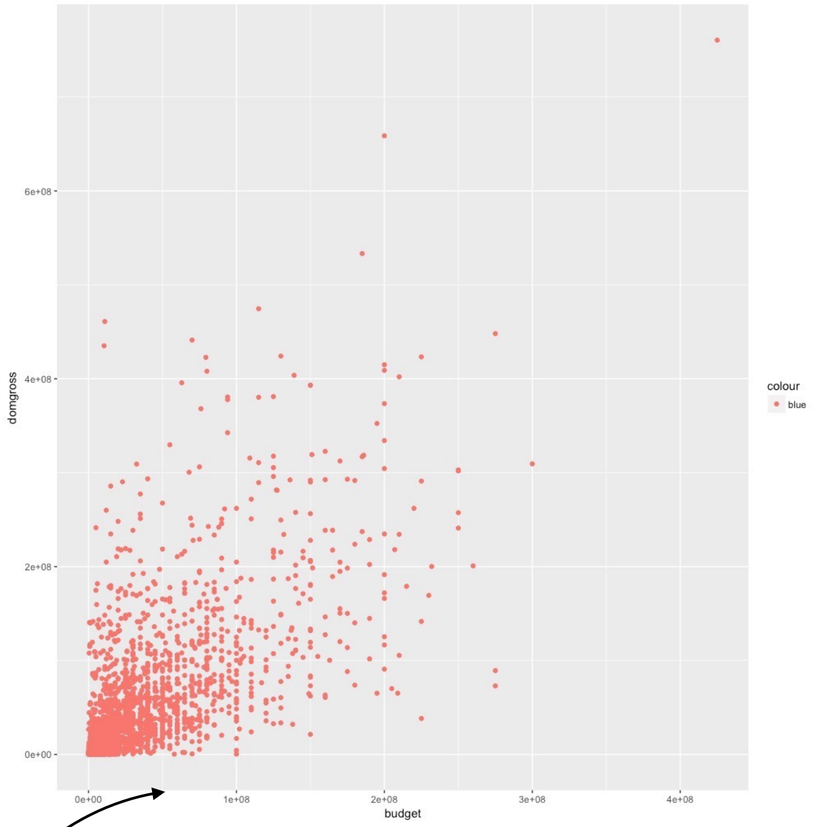
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

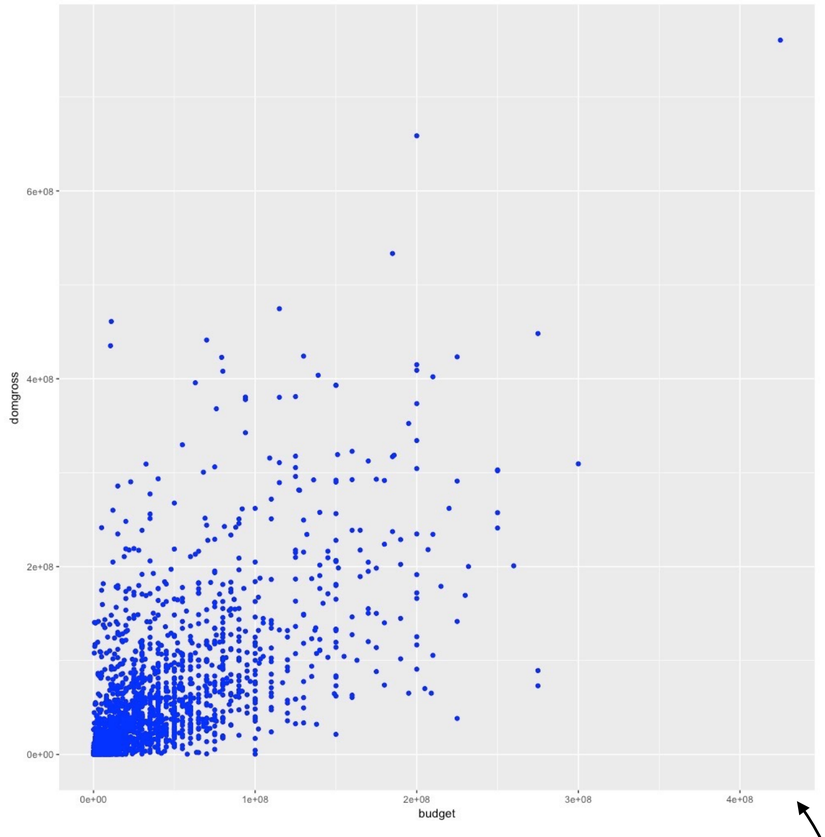





```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

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





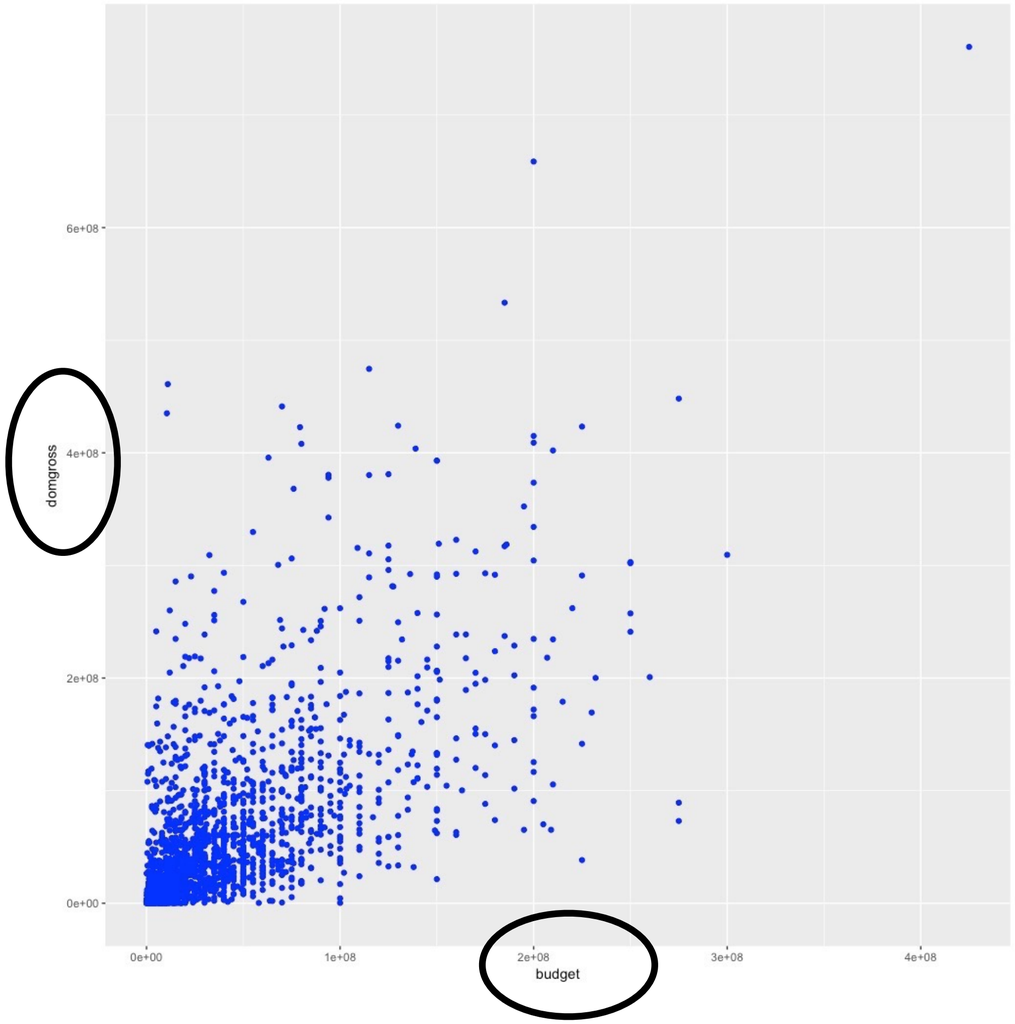

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color="blue"))
```

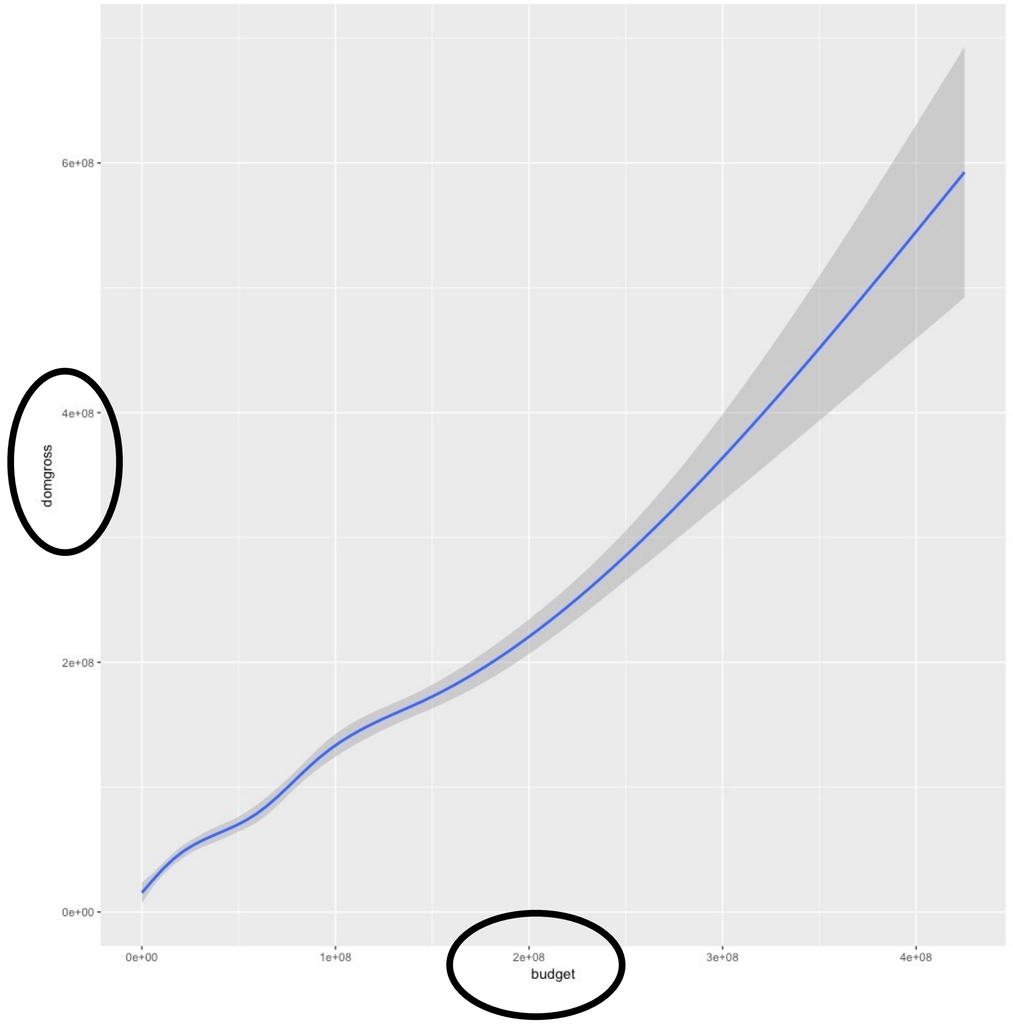
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

geoms

How are these plots similar?

Same: x var, y var, data

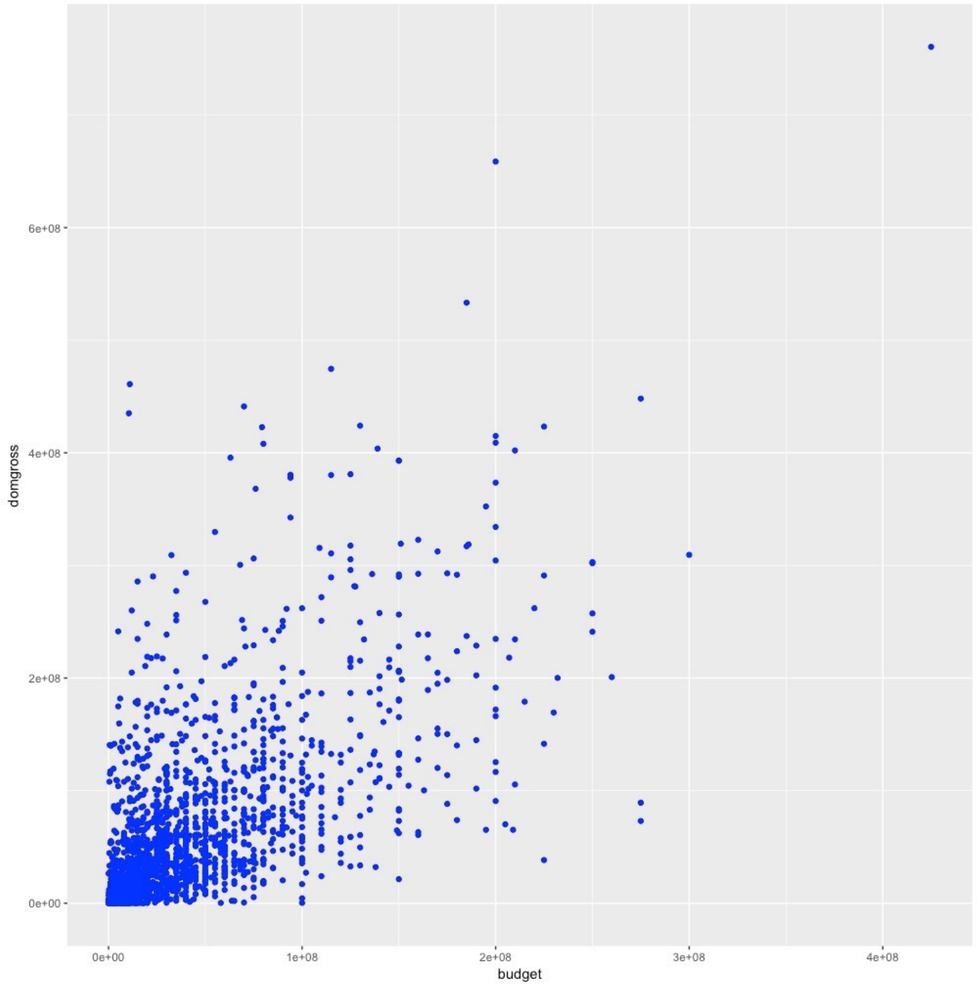


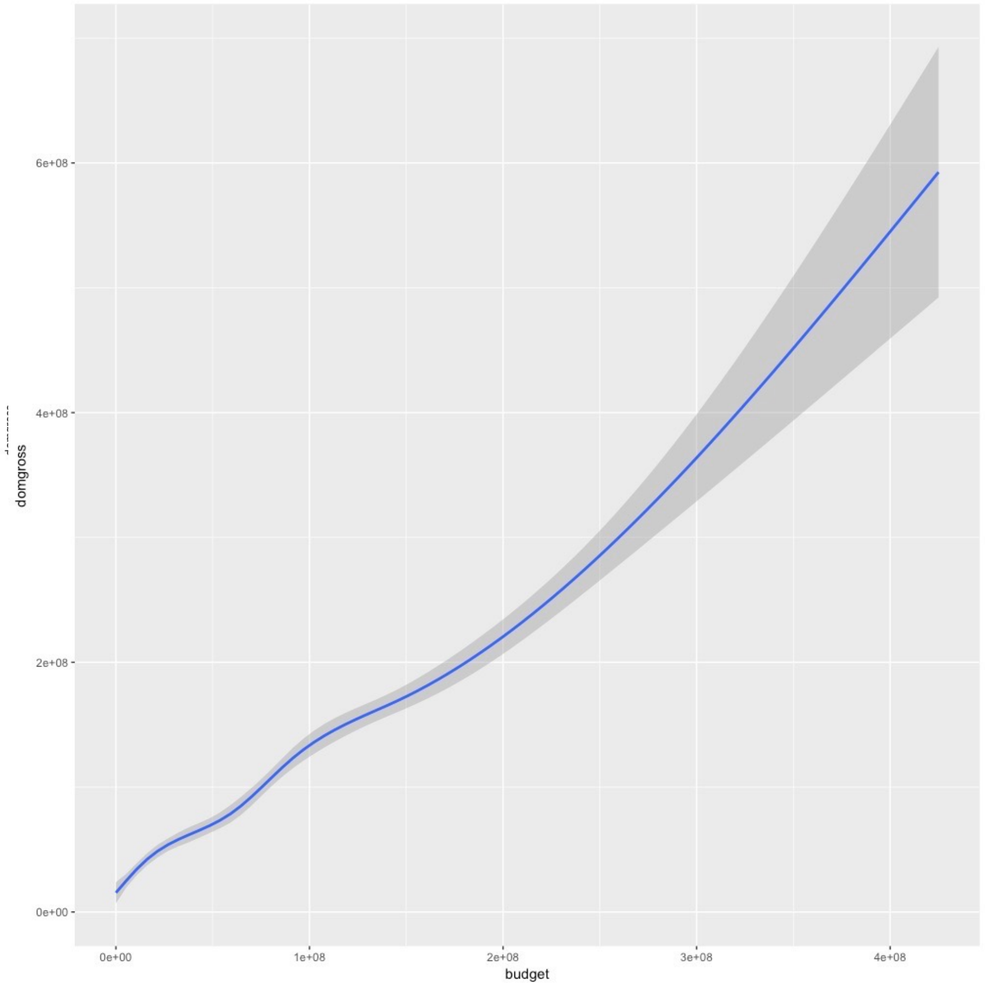


How are these plots different?

Different: geometric object (geom),

i.e. the visual object used to represent the data



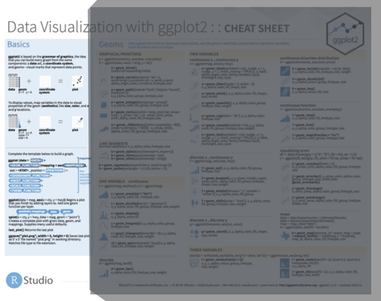


geoms

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

geom_ functions

Each requires a mapping argument.



Geoms

Use a geom 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, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=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

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 = 1:1155, 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

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

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

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

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, width (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

THREE VARIABLES

seals$z <- 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), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

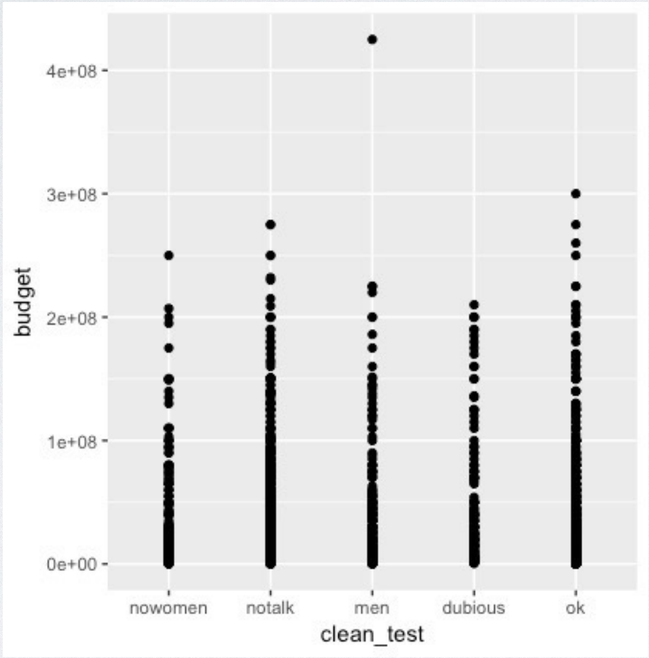
l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

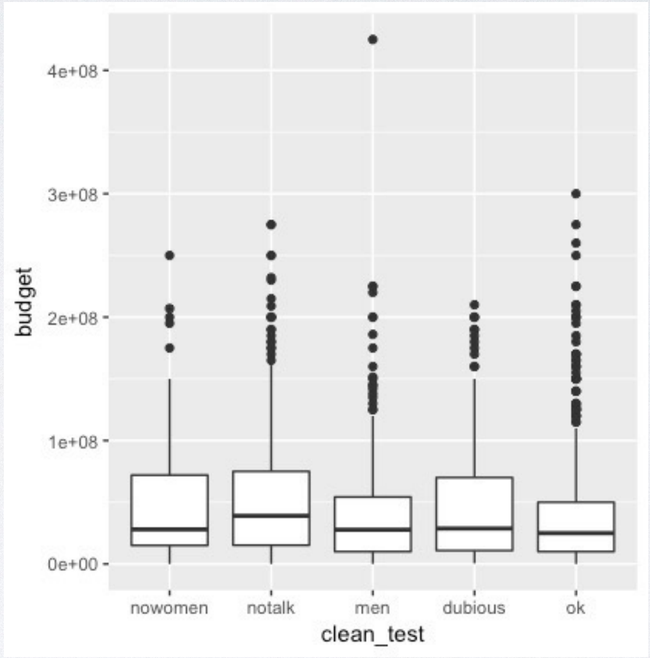
Your Turn

Pair up.

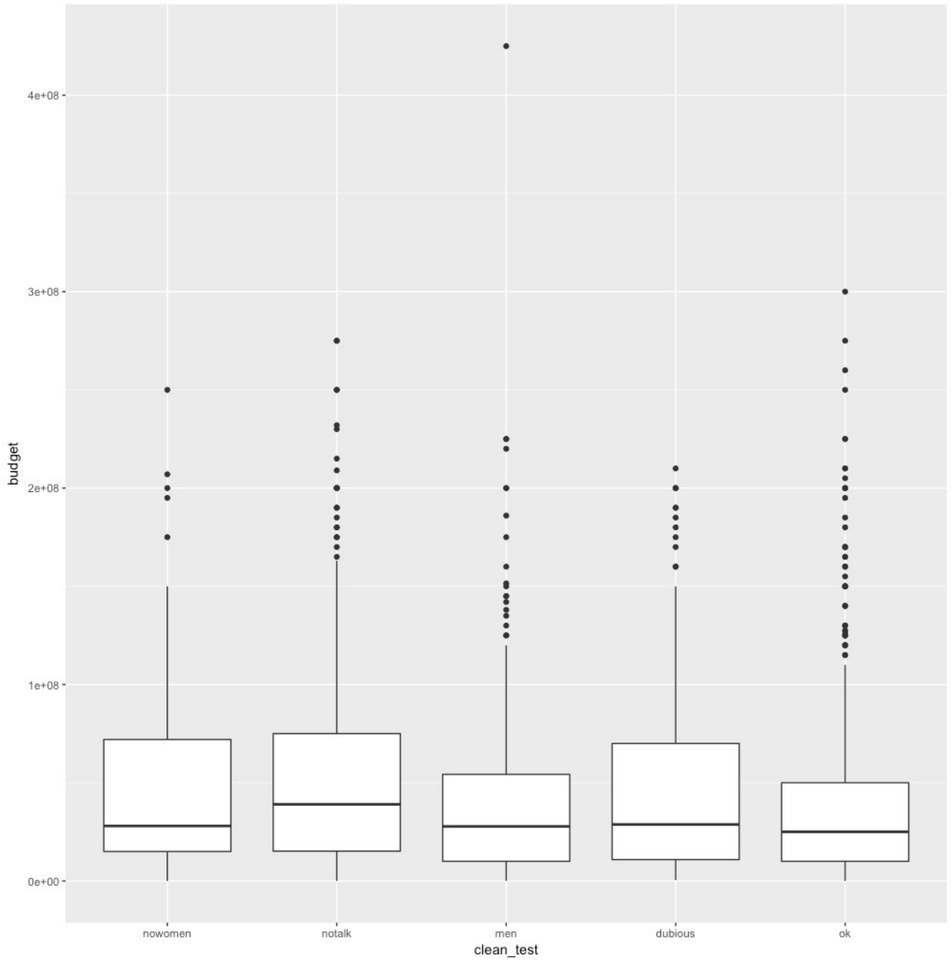
Your Turn 3

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





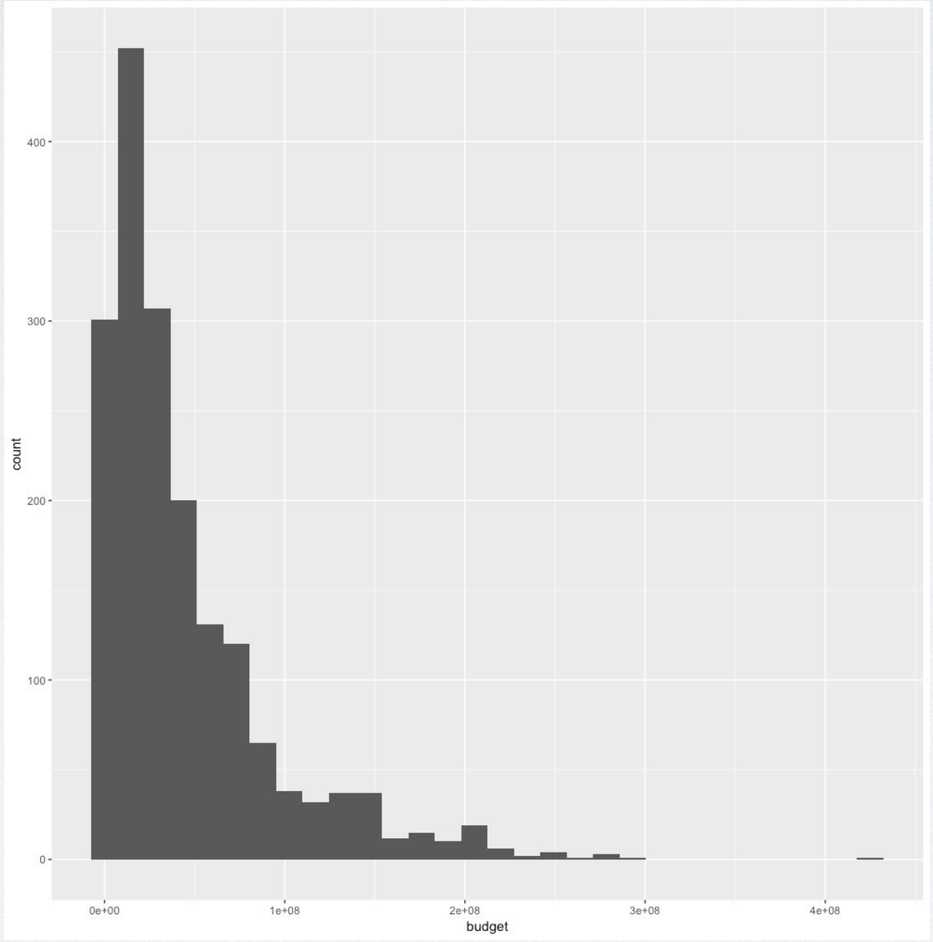

```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = clean_test, y = budget))
```



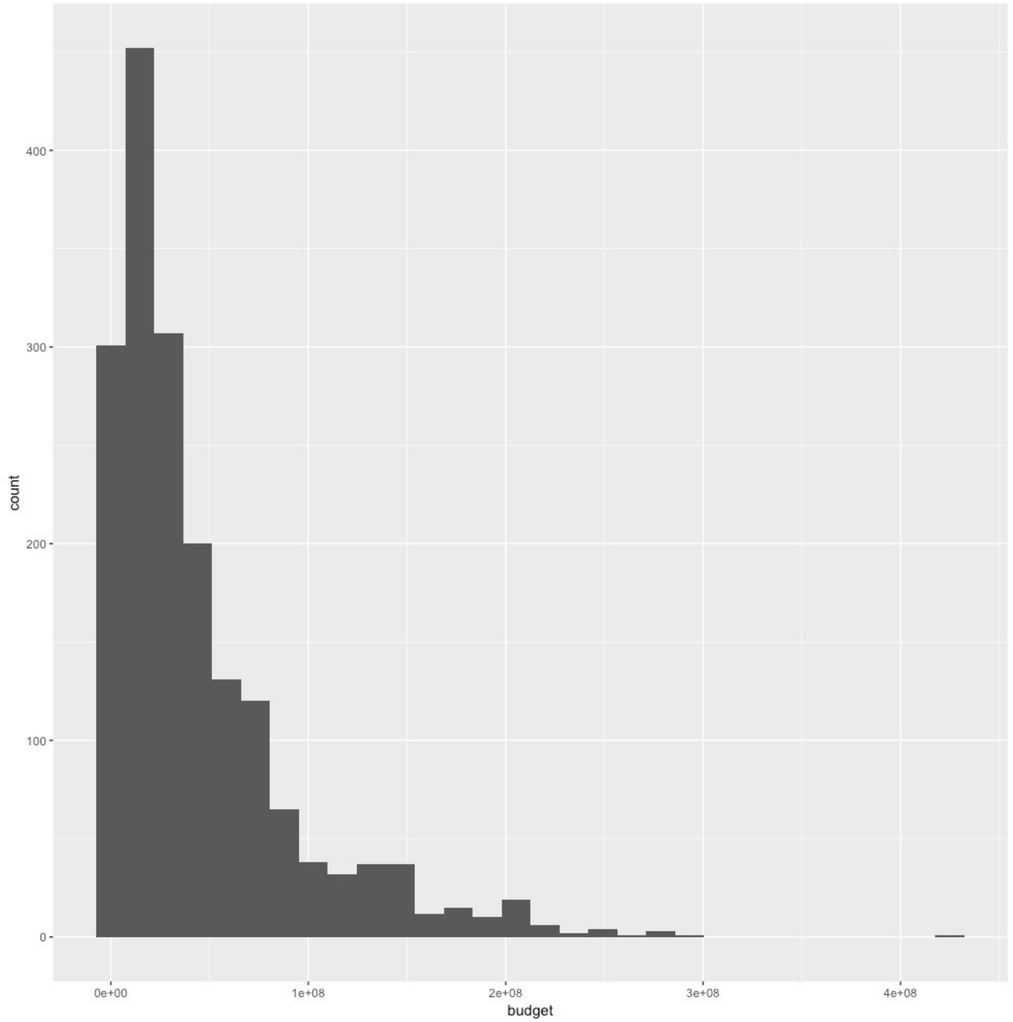

```
ggplot(data = bechdel) +
  geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

Your Turn 4

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

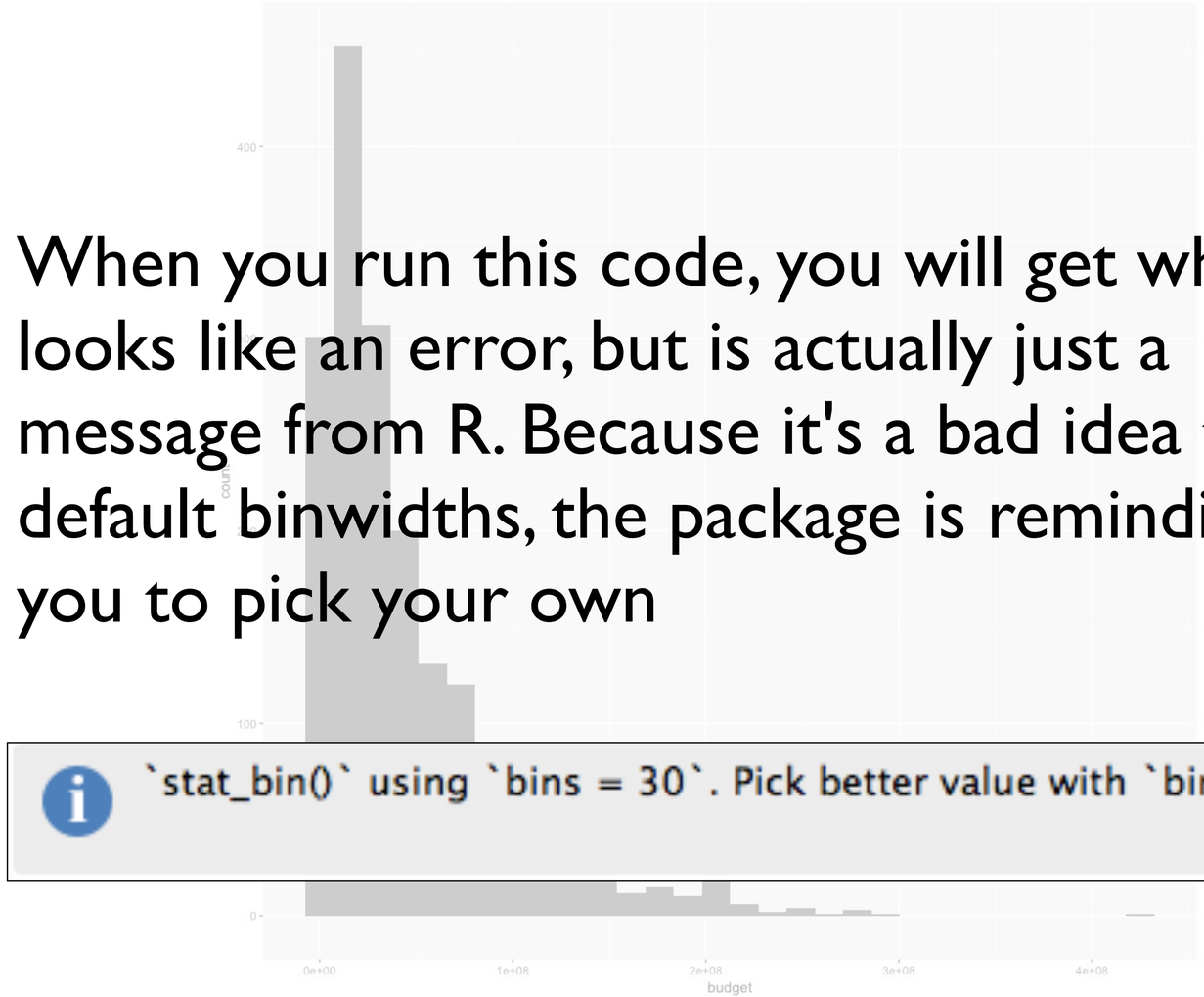


02:00




```
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



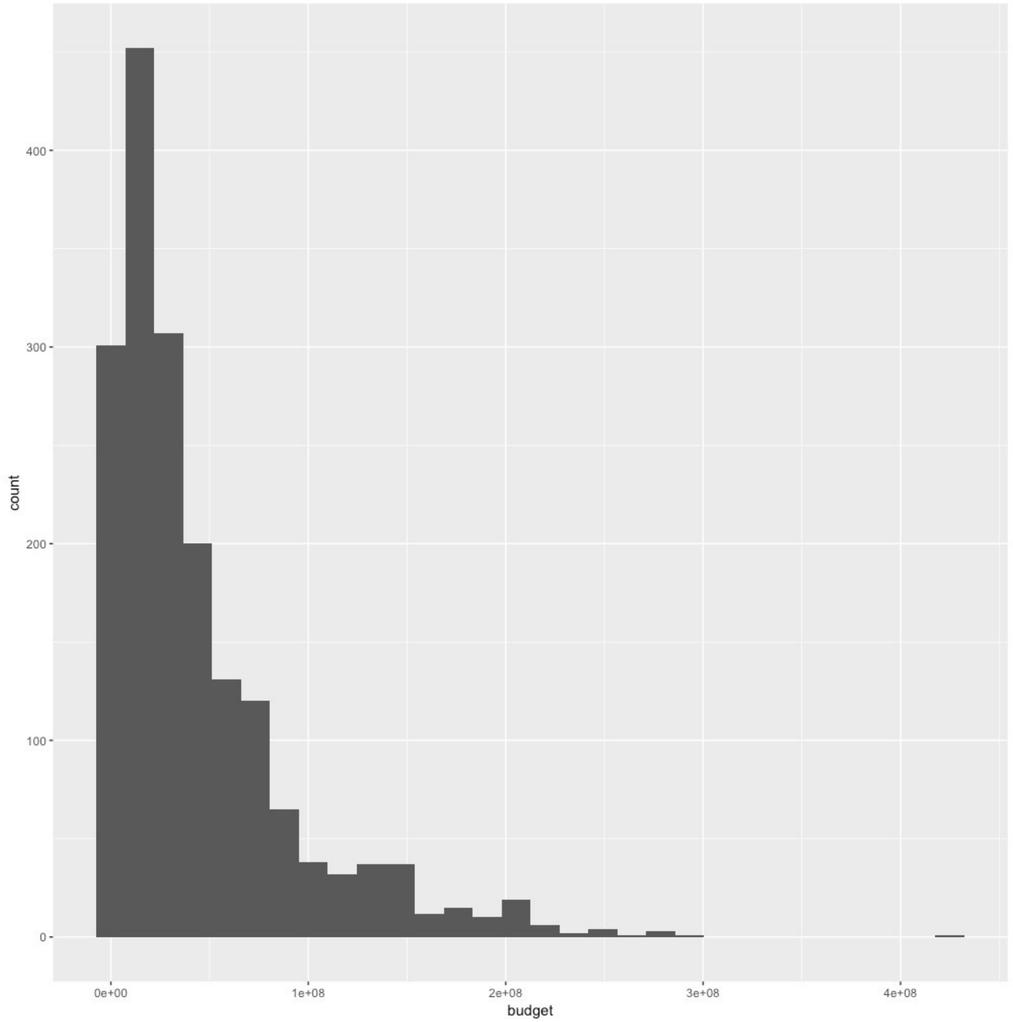
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

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

Your Turn 5

What would be a reasonable `binwidth` for `budget`?

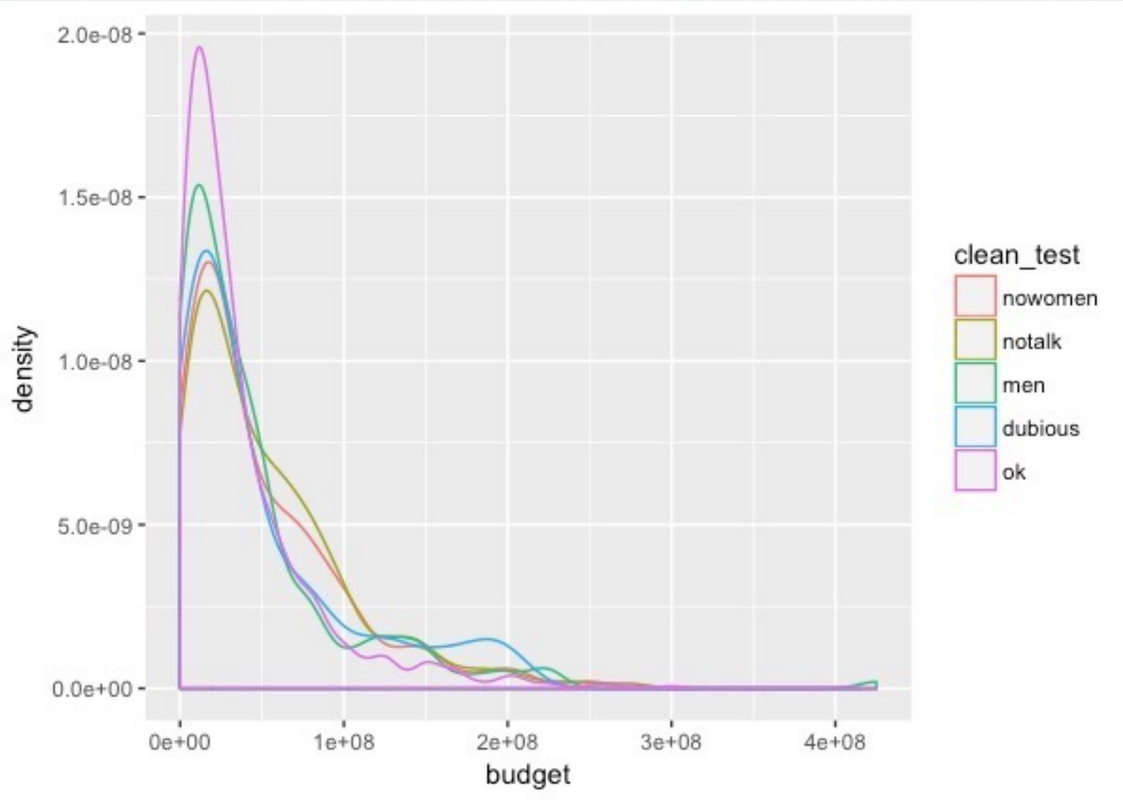
Try it out



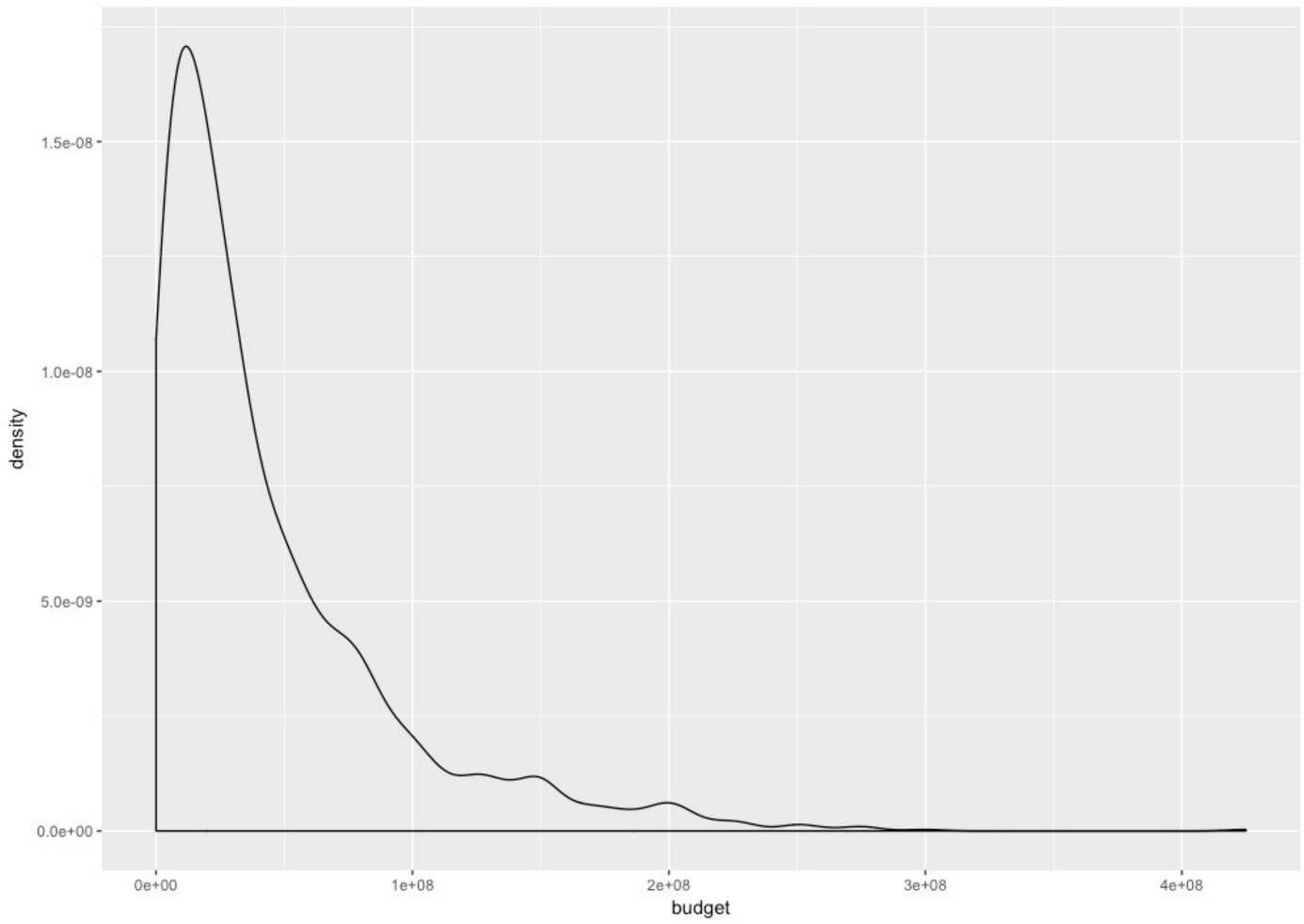

```
ggplot(data = bechdel) +
  geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```

Your Turn 6

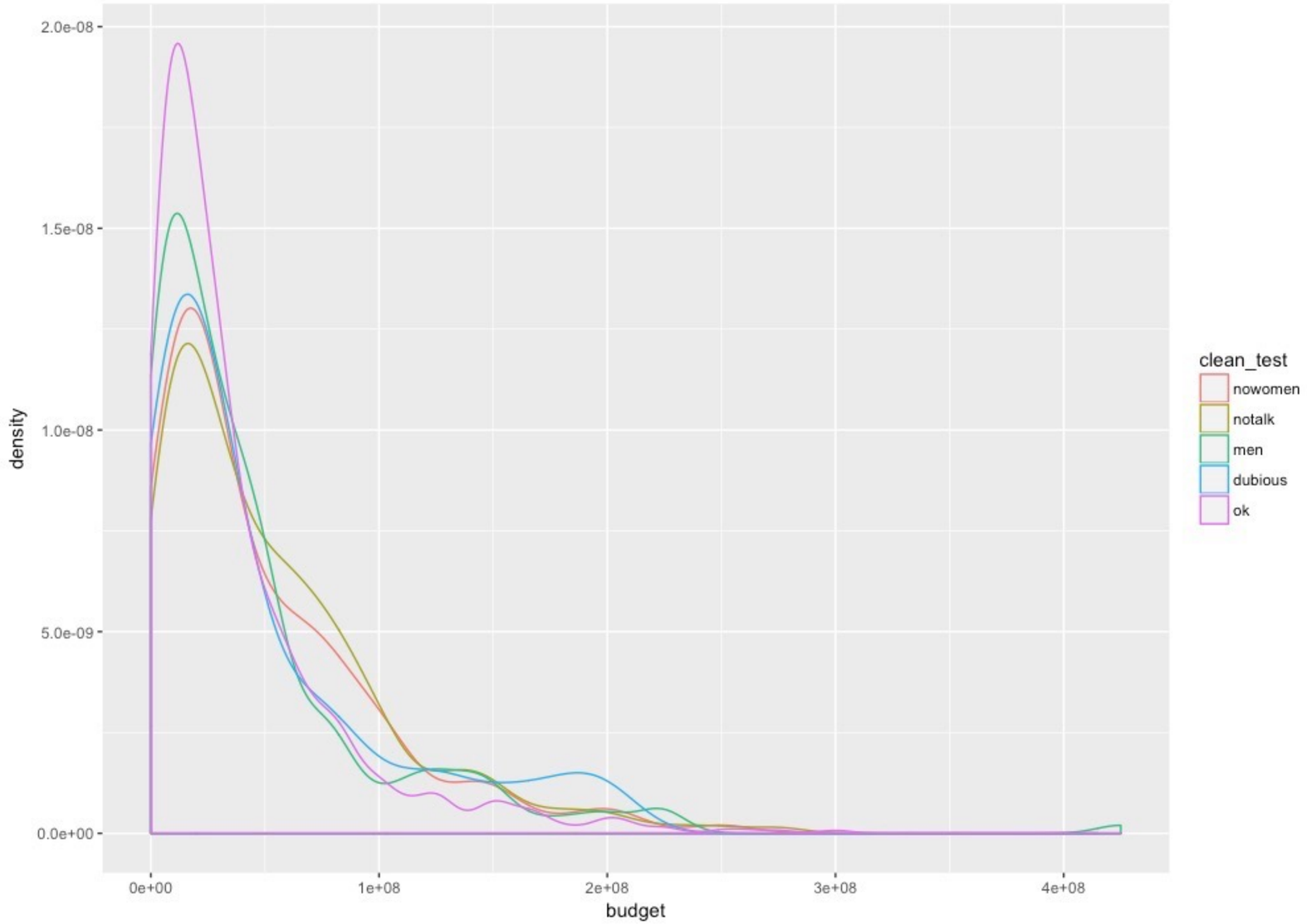
With your partner, make the density plot of **budget** colored by **clean_test** below. Use the cheatsheet. Try your best guess.



02:00




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



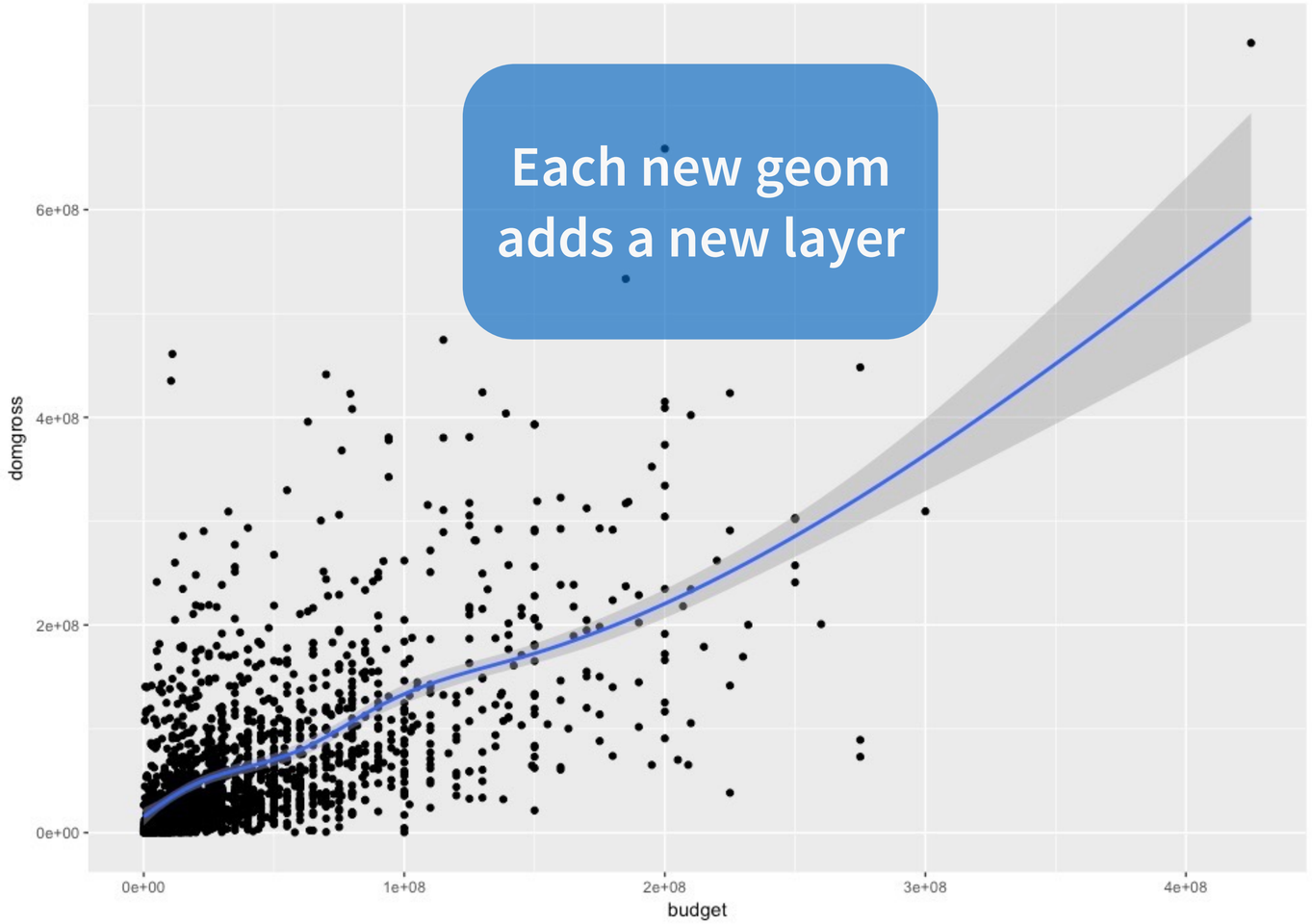

```
ggplot(data = bechdel) +
  geom_density(mapping = aes(x = budget, color=clean_test))
```

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))
```

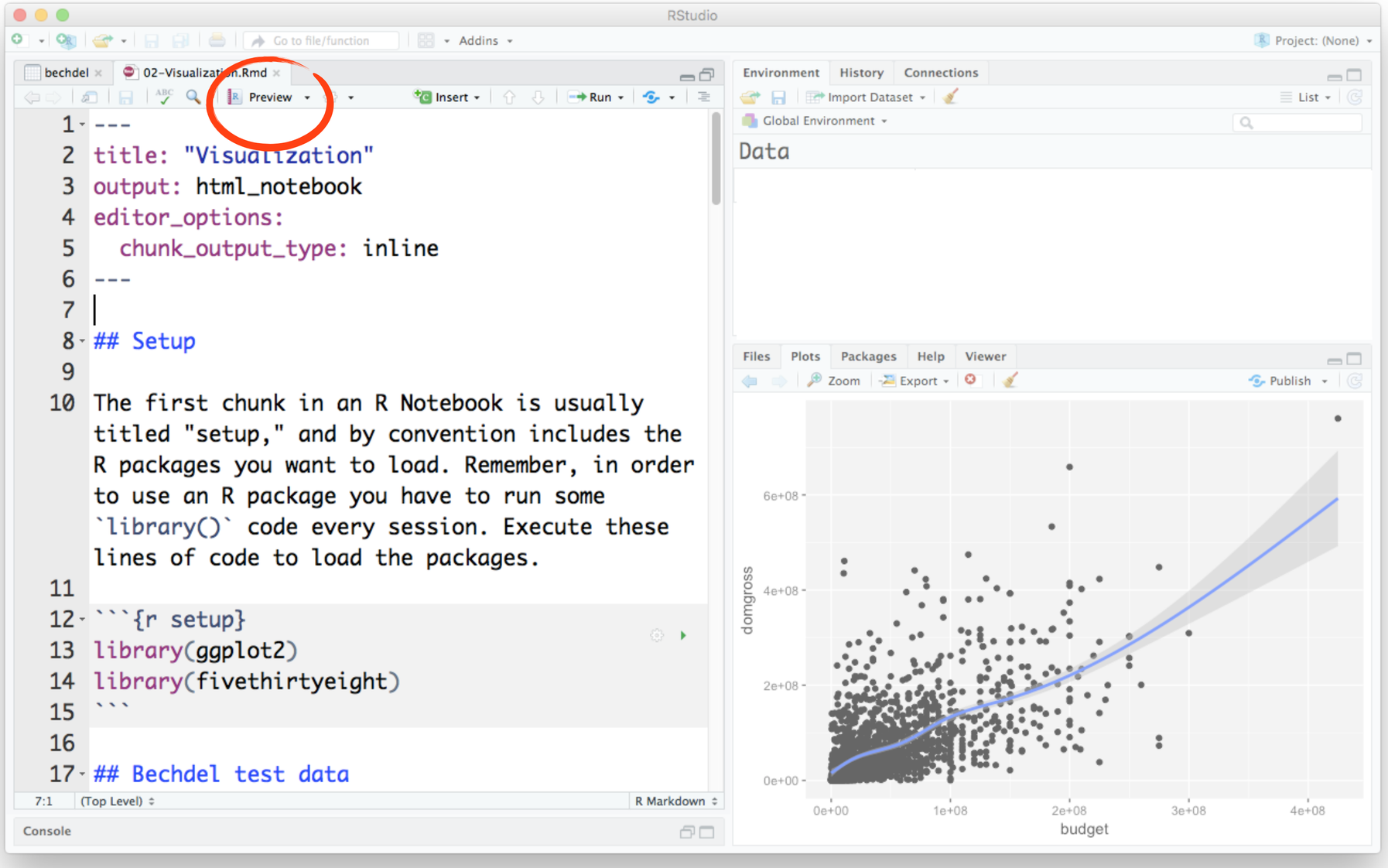



```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = budget, y = domgross)) +
  geom_smooth(mapping = aes(x = budget, y = domgross))
```

Saving graphs

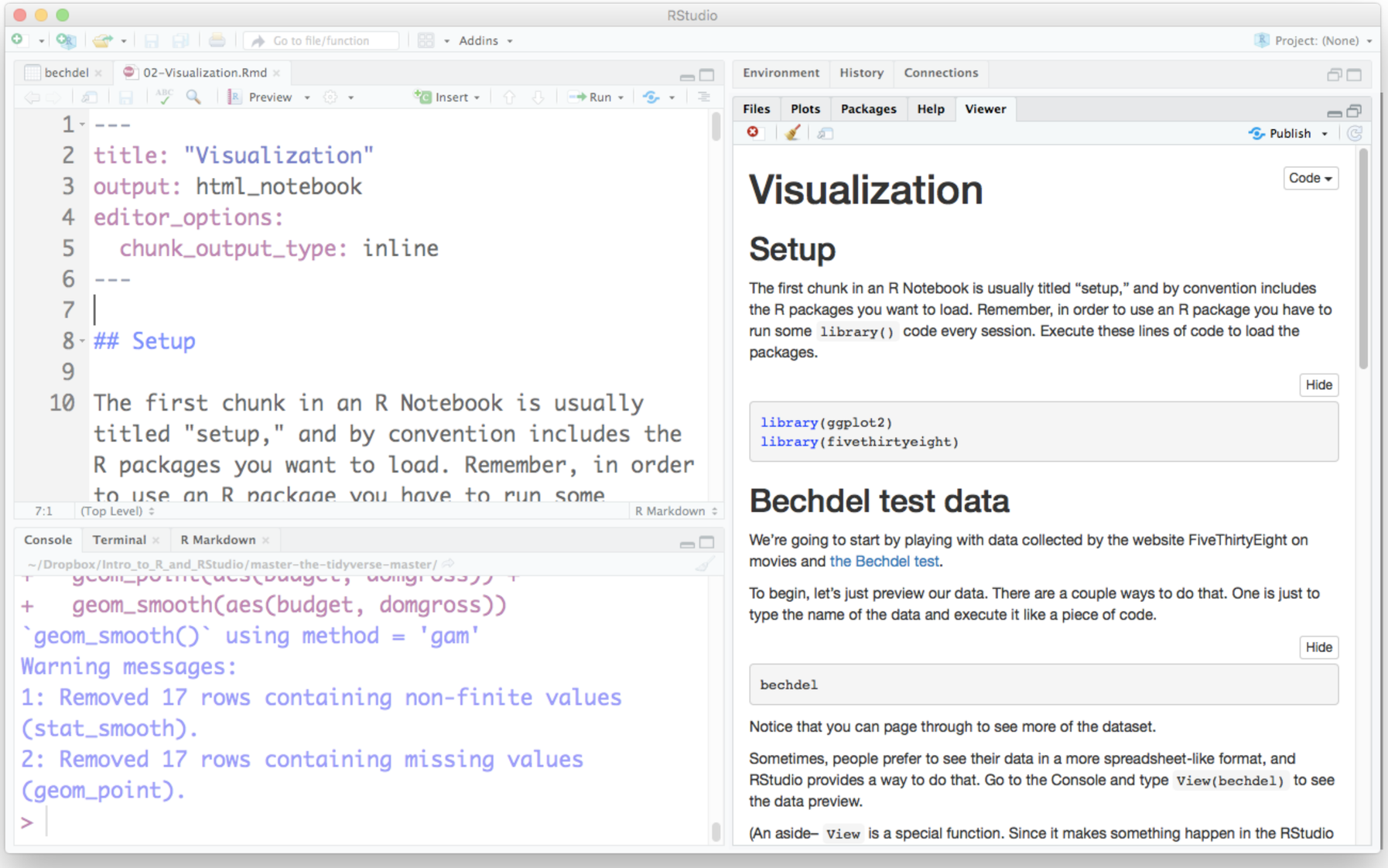
R Notebook Preview

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"



R Notebook Preview

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



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

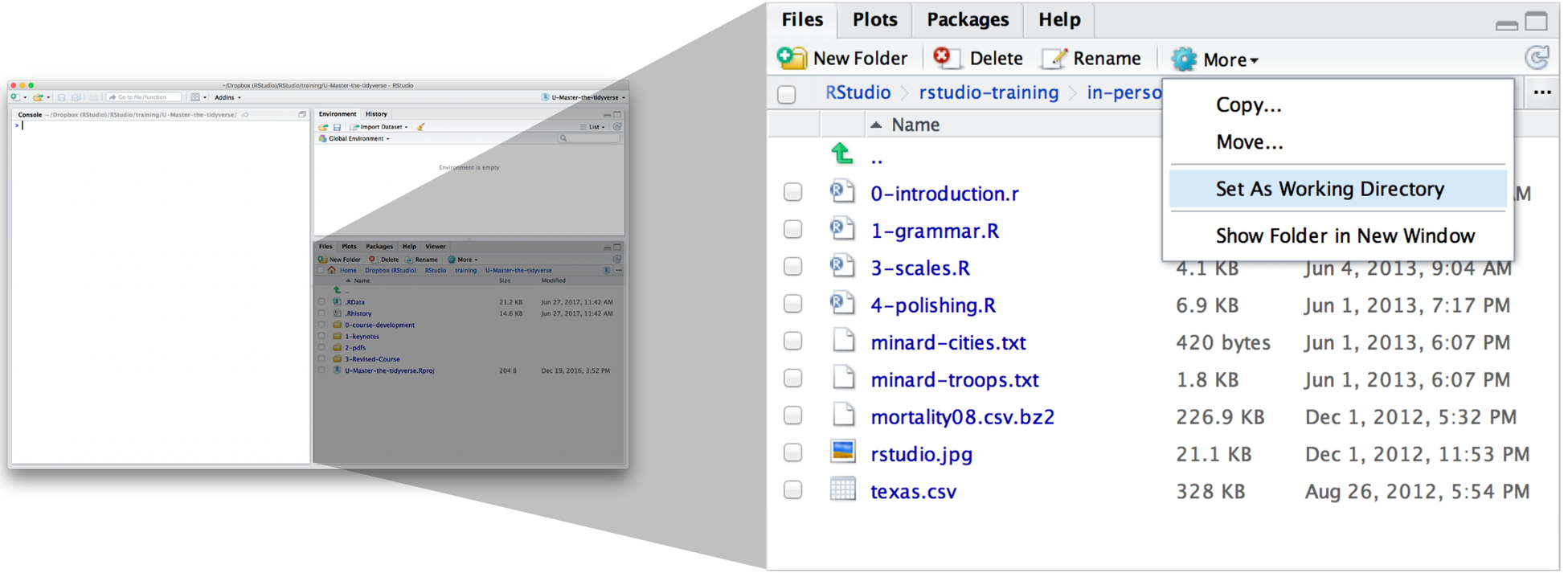
R associates itself with a folder (i.e. directory) on your computer.

- 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()`

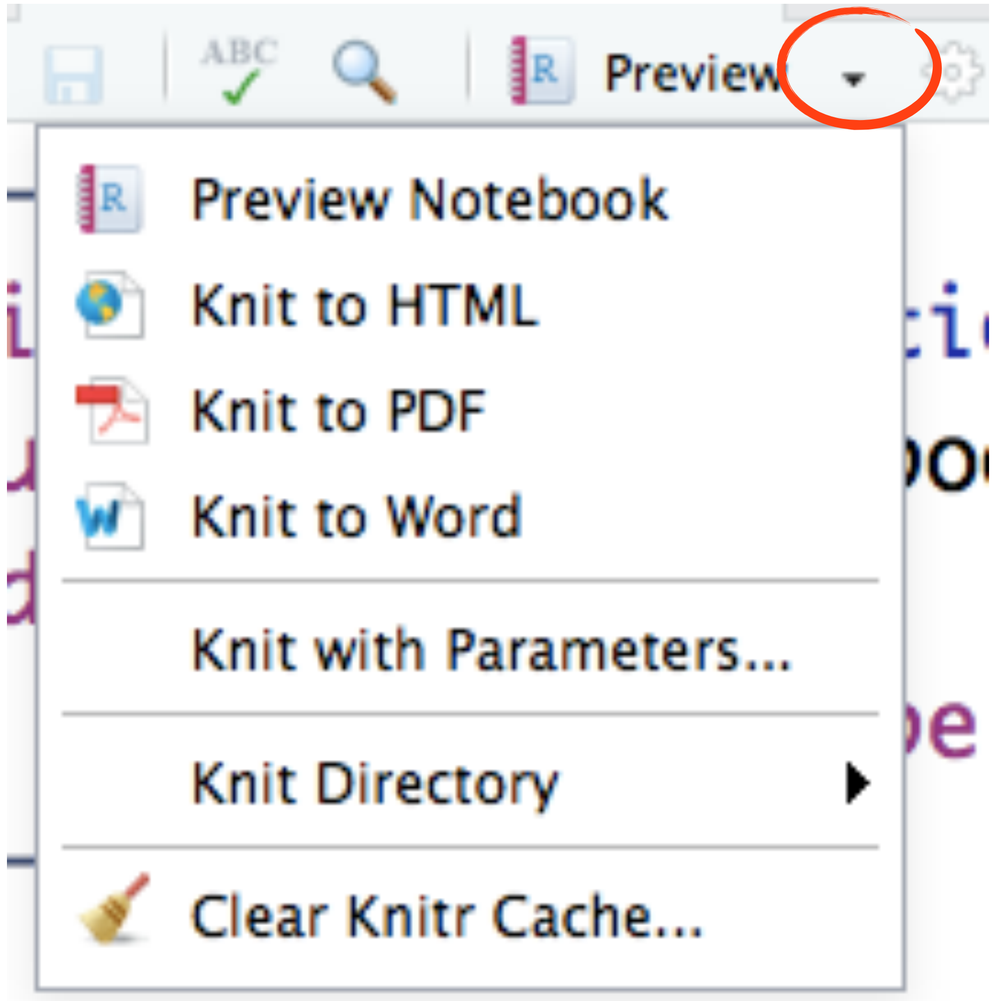
Changing the Working directory

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



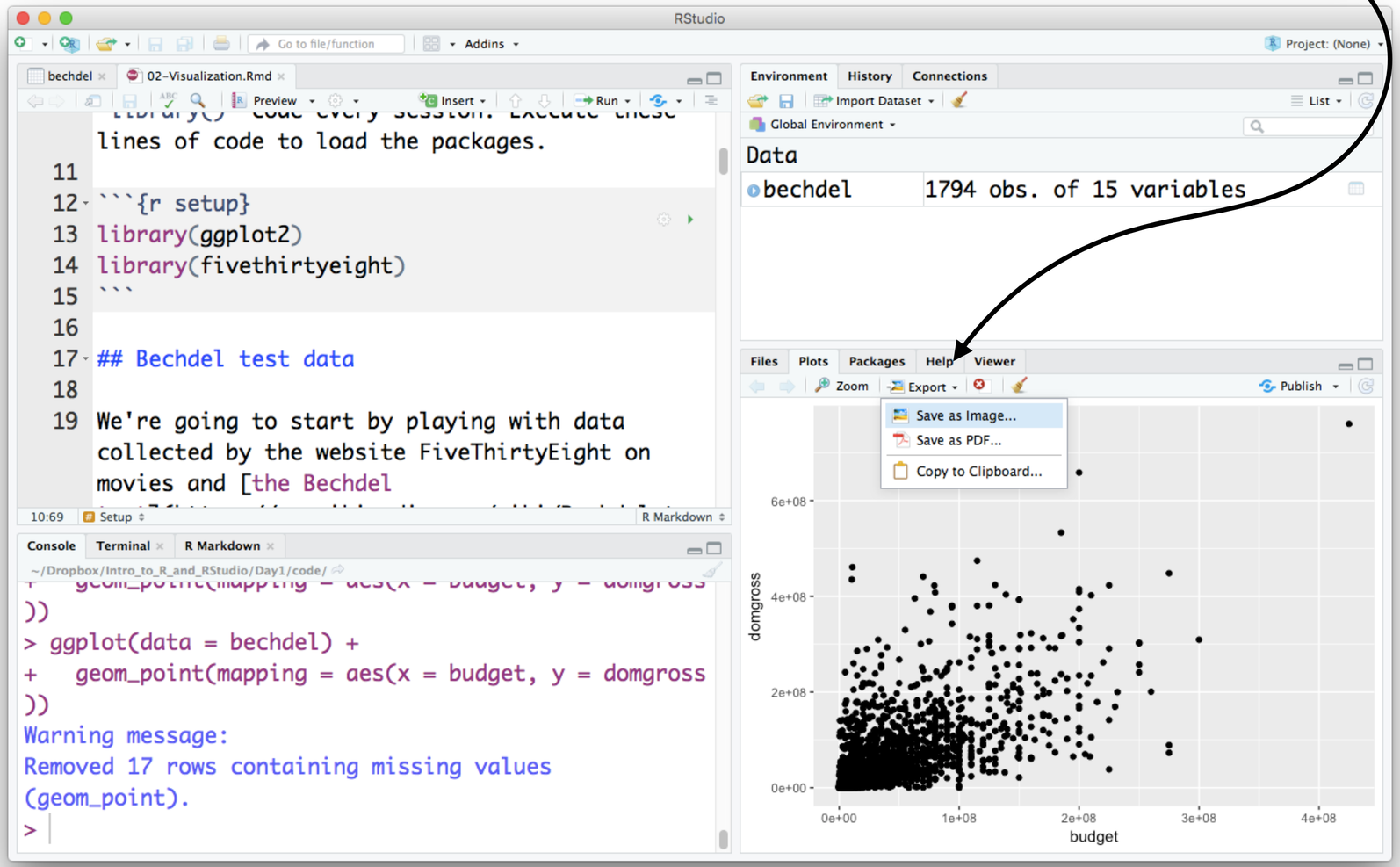
"knitting" a document

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:

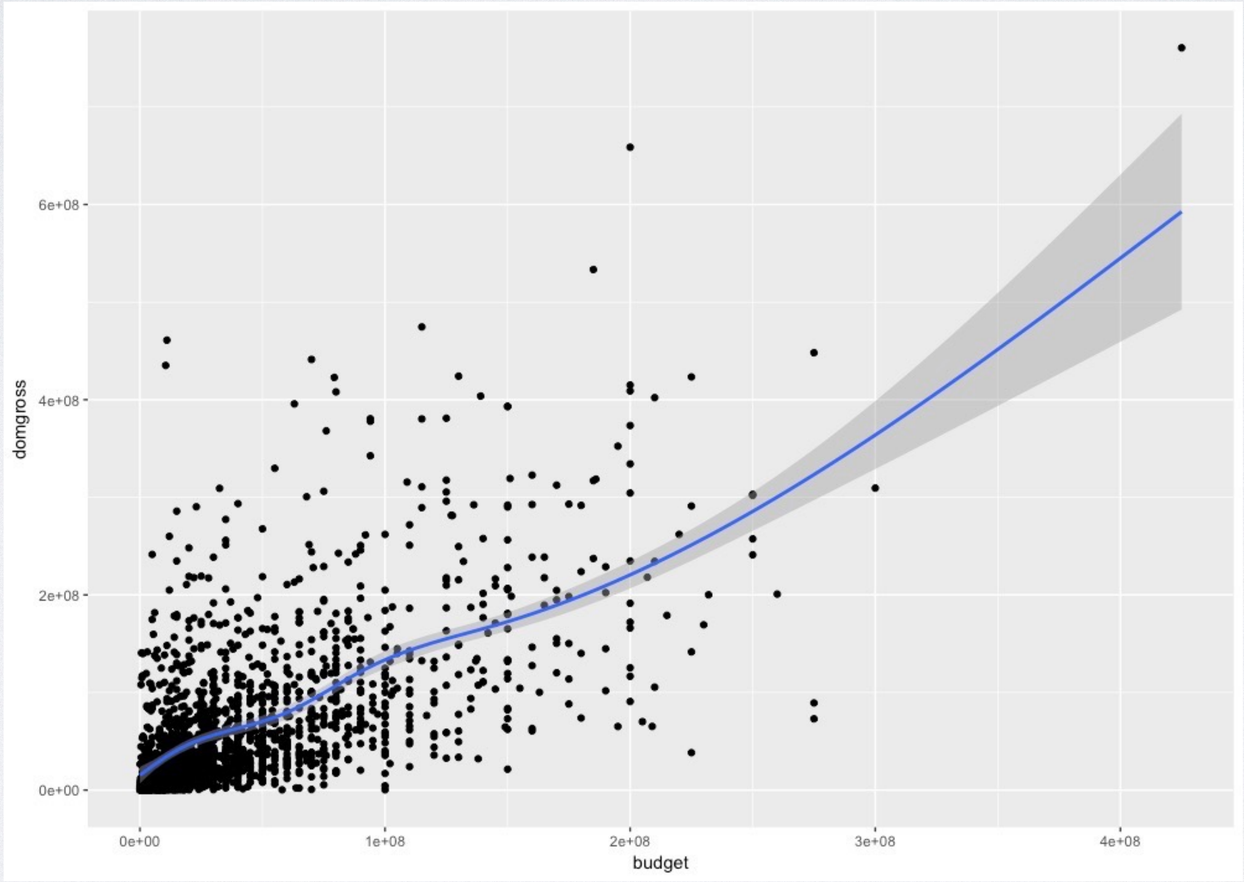
```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

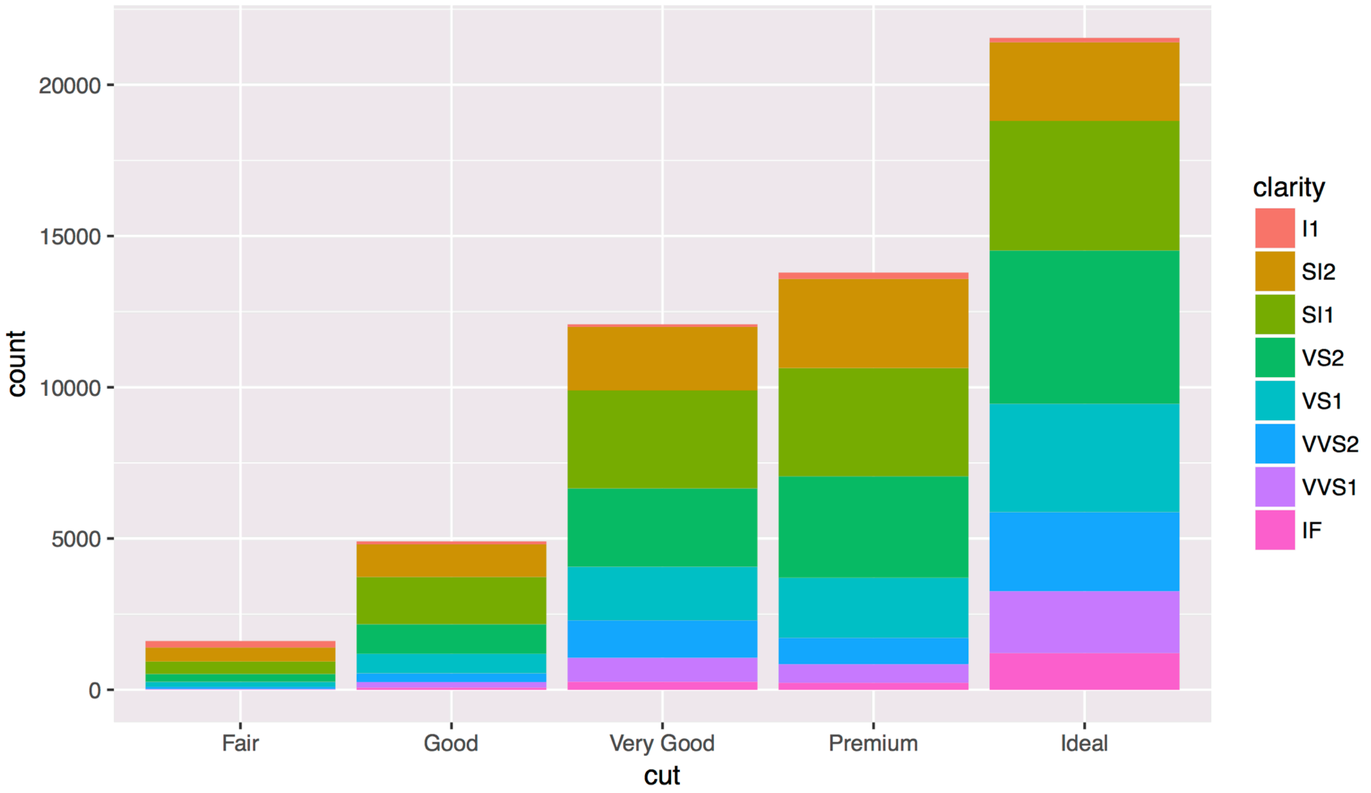
```
ggsave("my-plot.pdf", width = 6, height = 6)
```

Your Turn 9

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

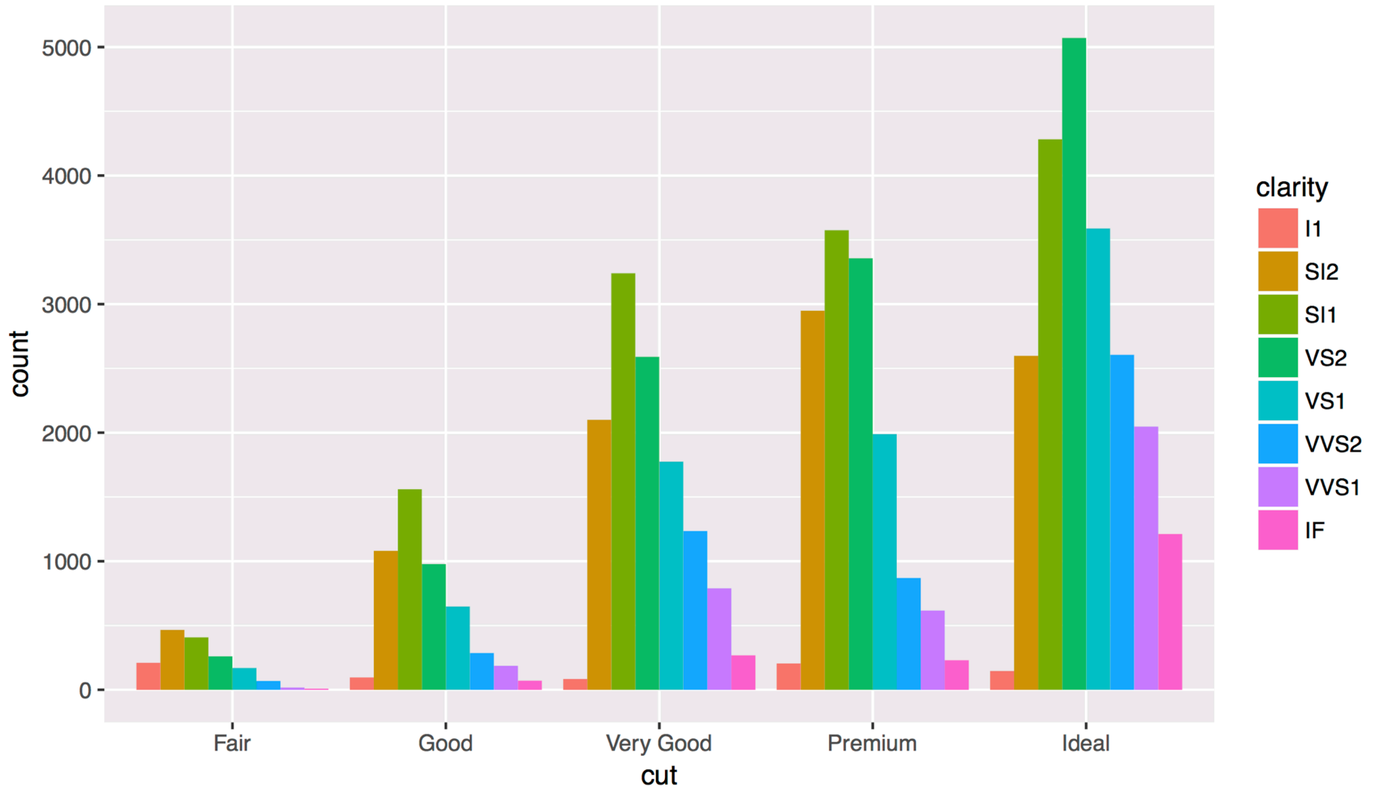


What else?


20000
15000
10000
5000
0
count
Fair
Good
Very Good
Premium
Ideal
cut
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

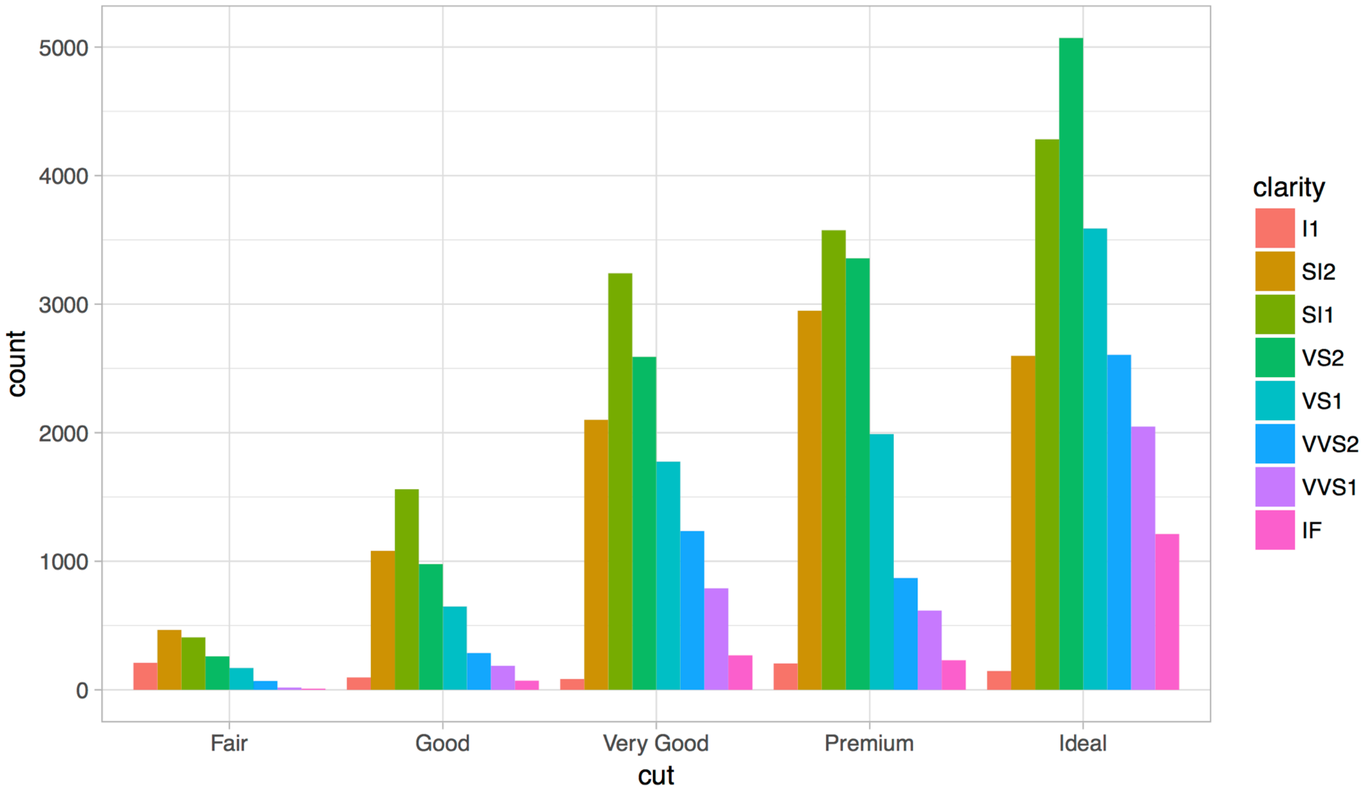
Position Adjustments

How overlapping objects are arranged



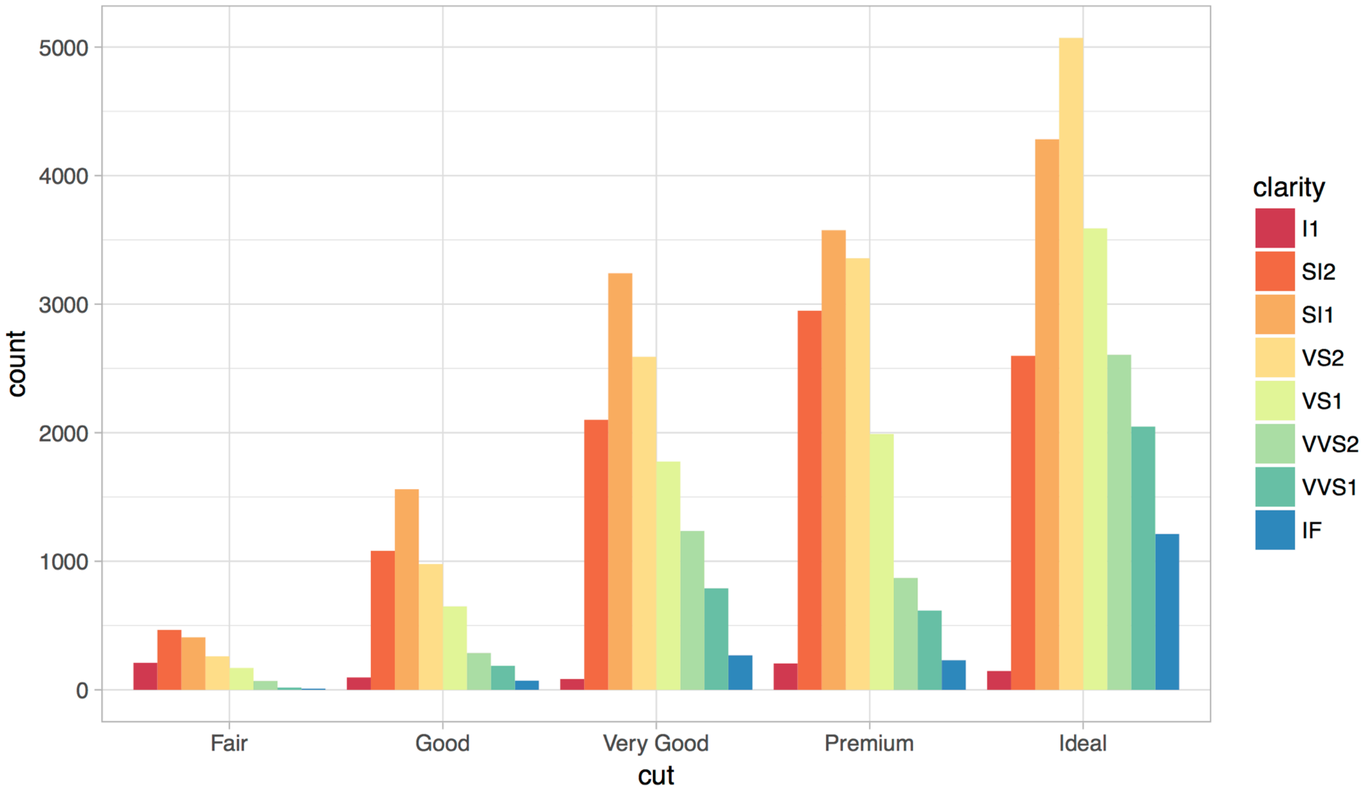
Themes

Visual appearance of non-data elements



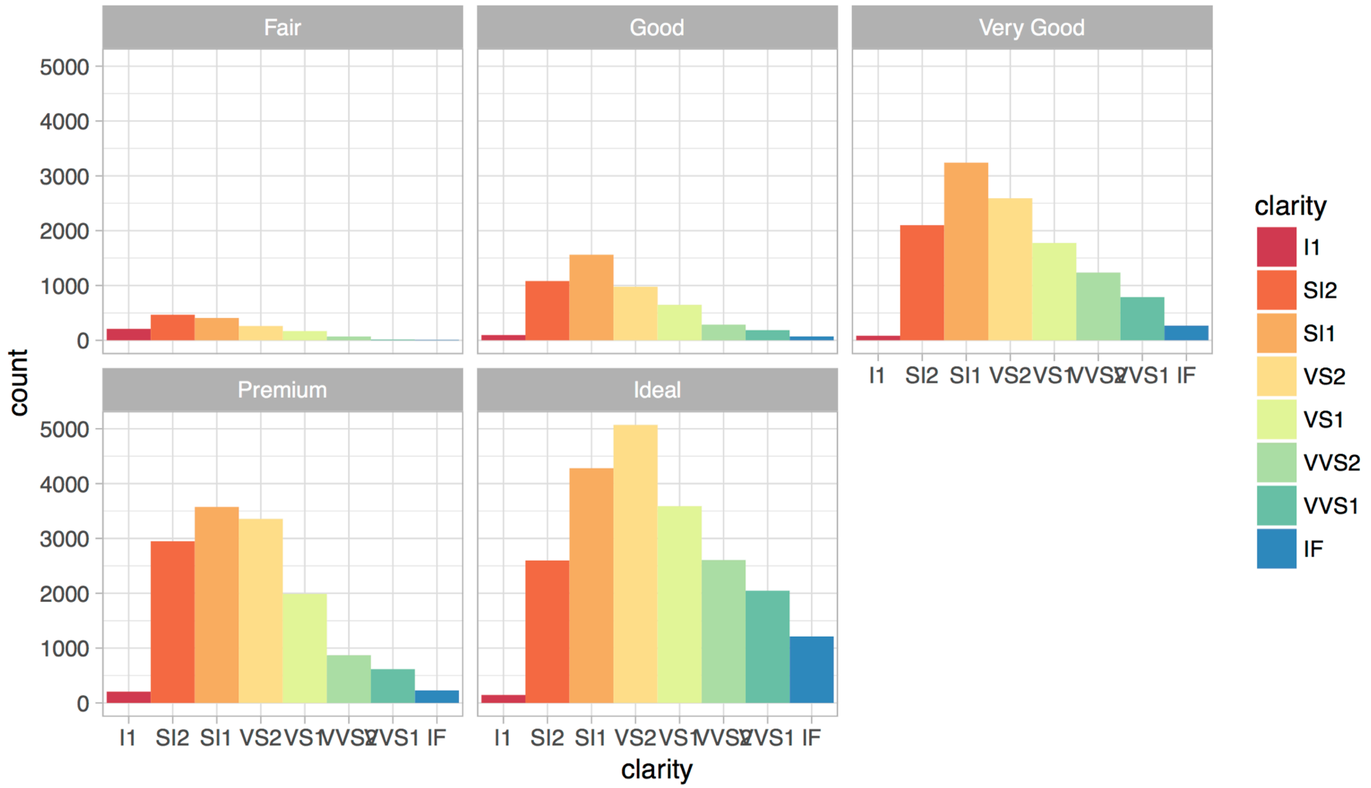
Scales

Customize color scales, other mappings

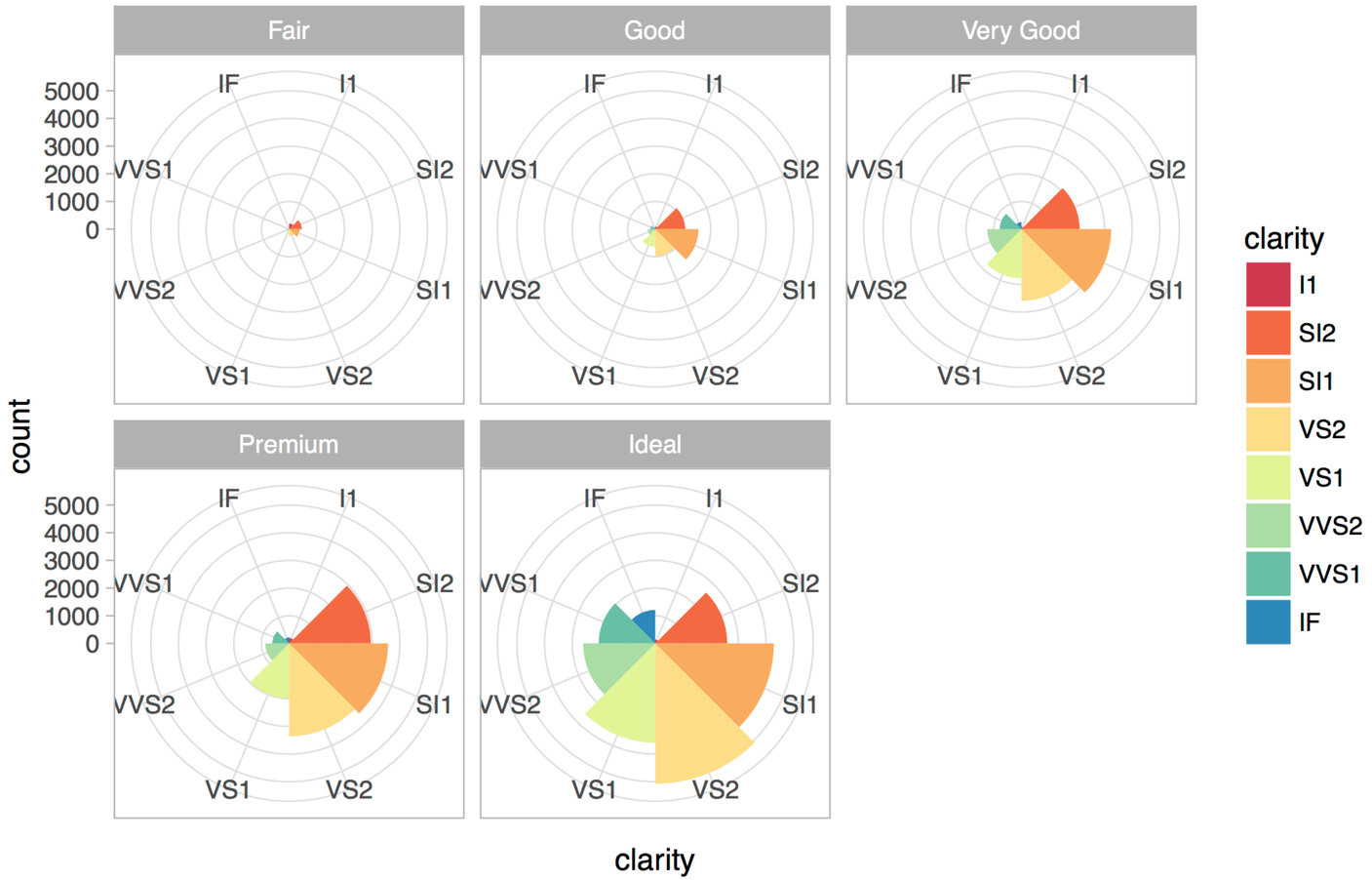


Facets

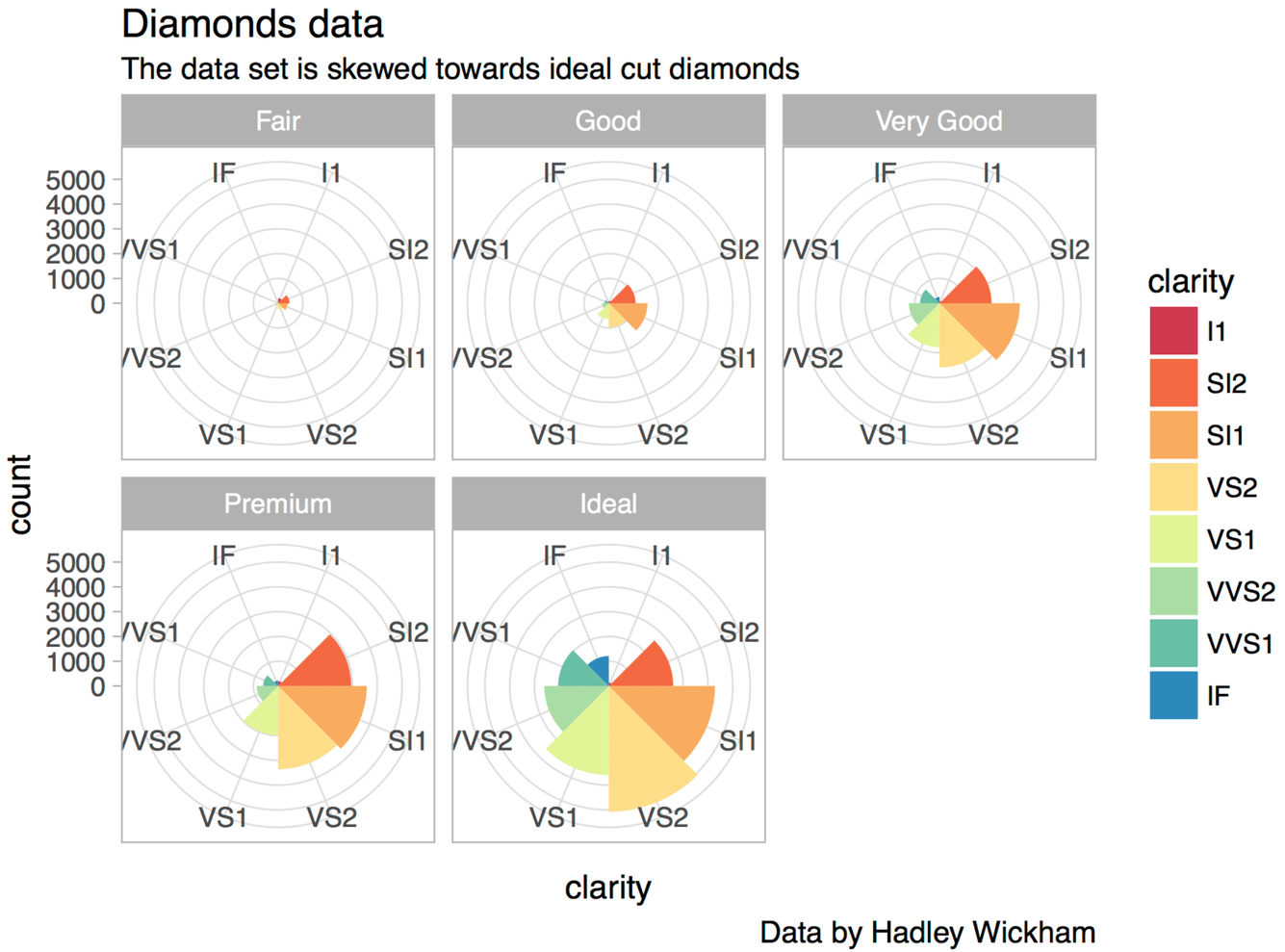
Subplots that display subsets of the data.



Coordinate systems



Titles and captions



A ggplot2 template

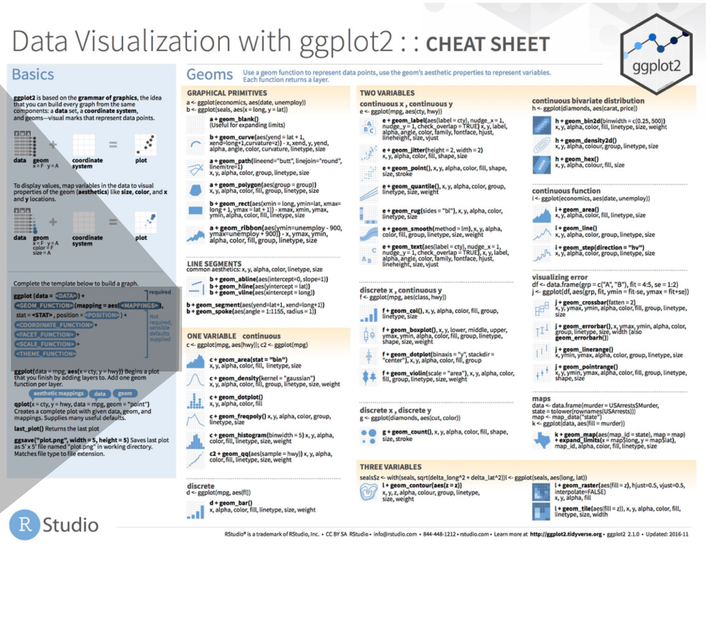
Make any plot by filling in the parameters of this template

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



ggplot2.tidyverse.org