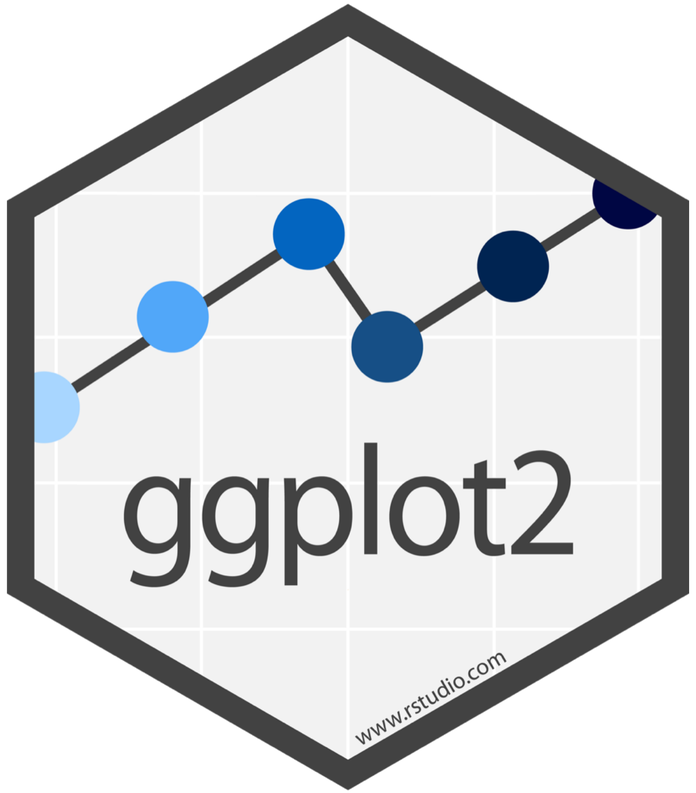
Visualize Data with



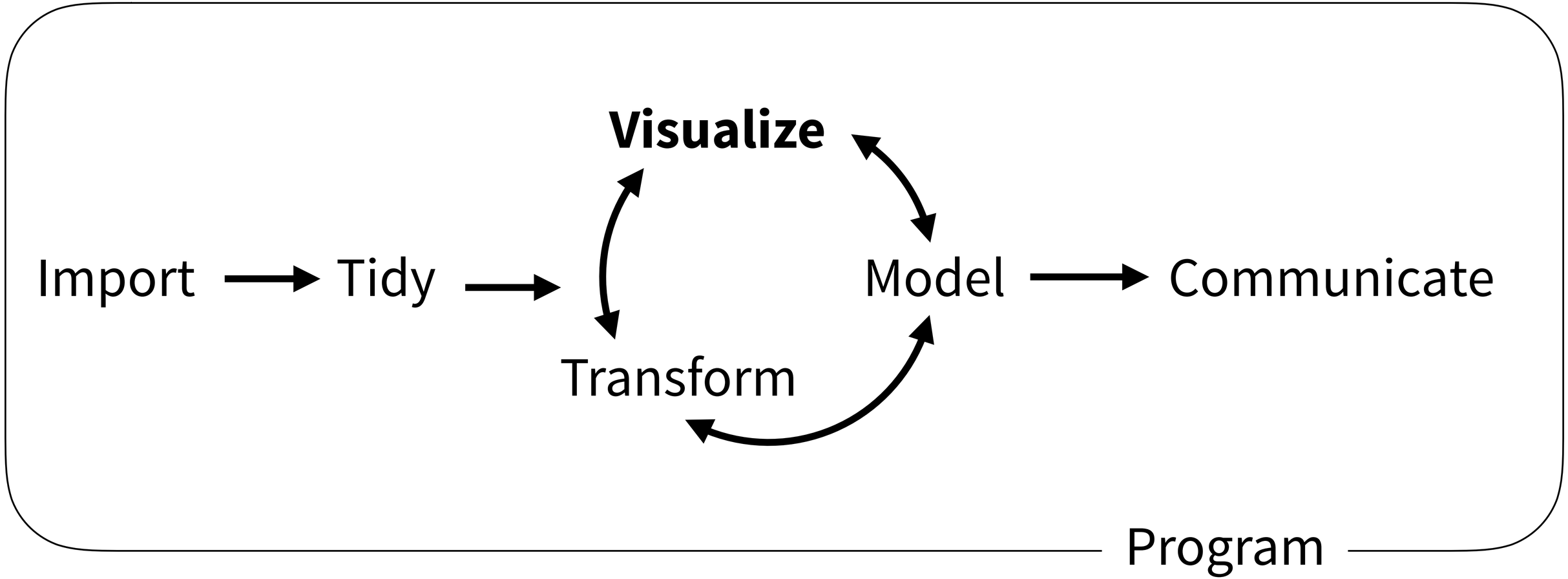
In R4DS

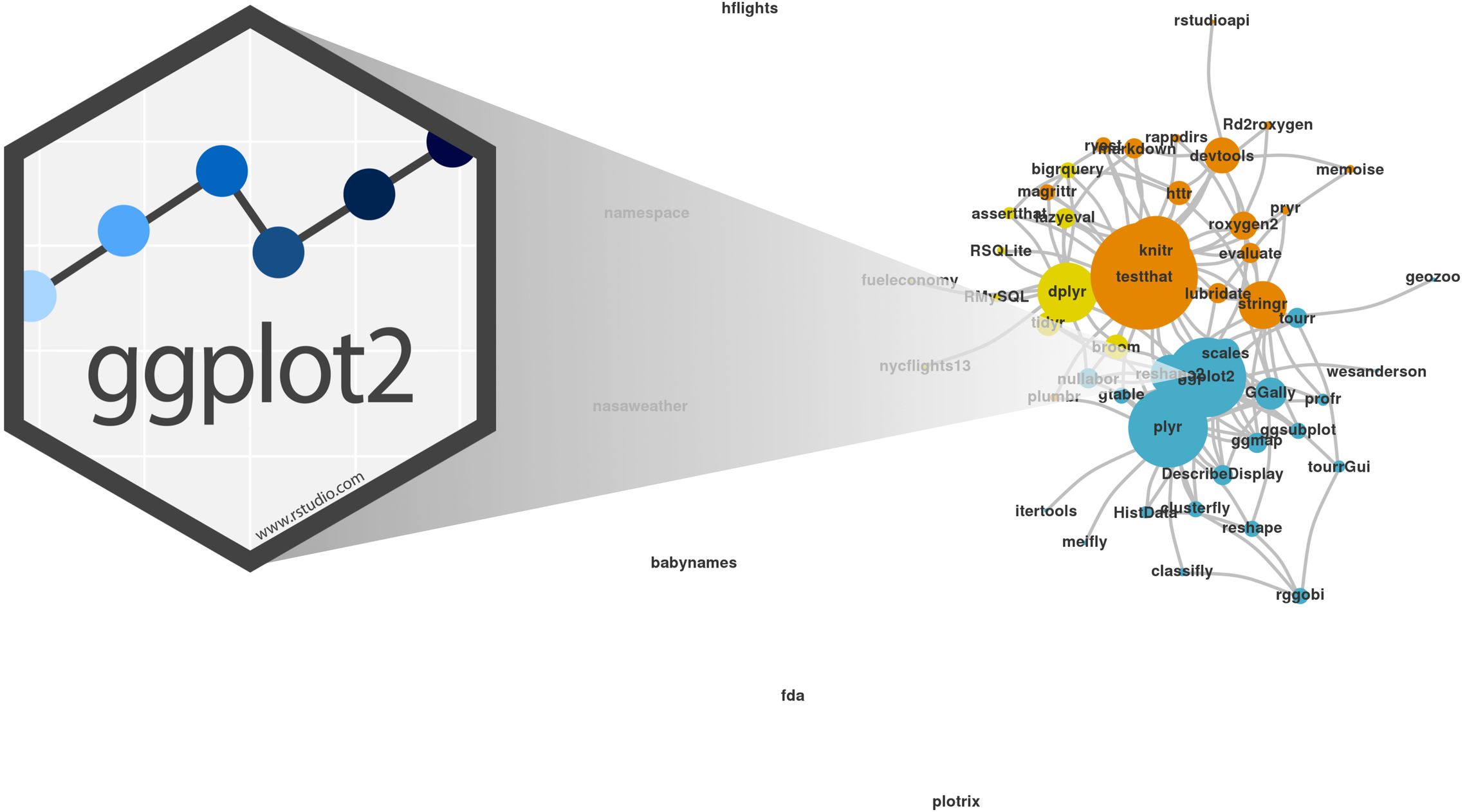
Visualizing Data

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

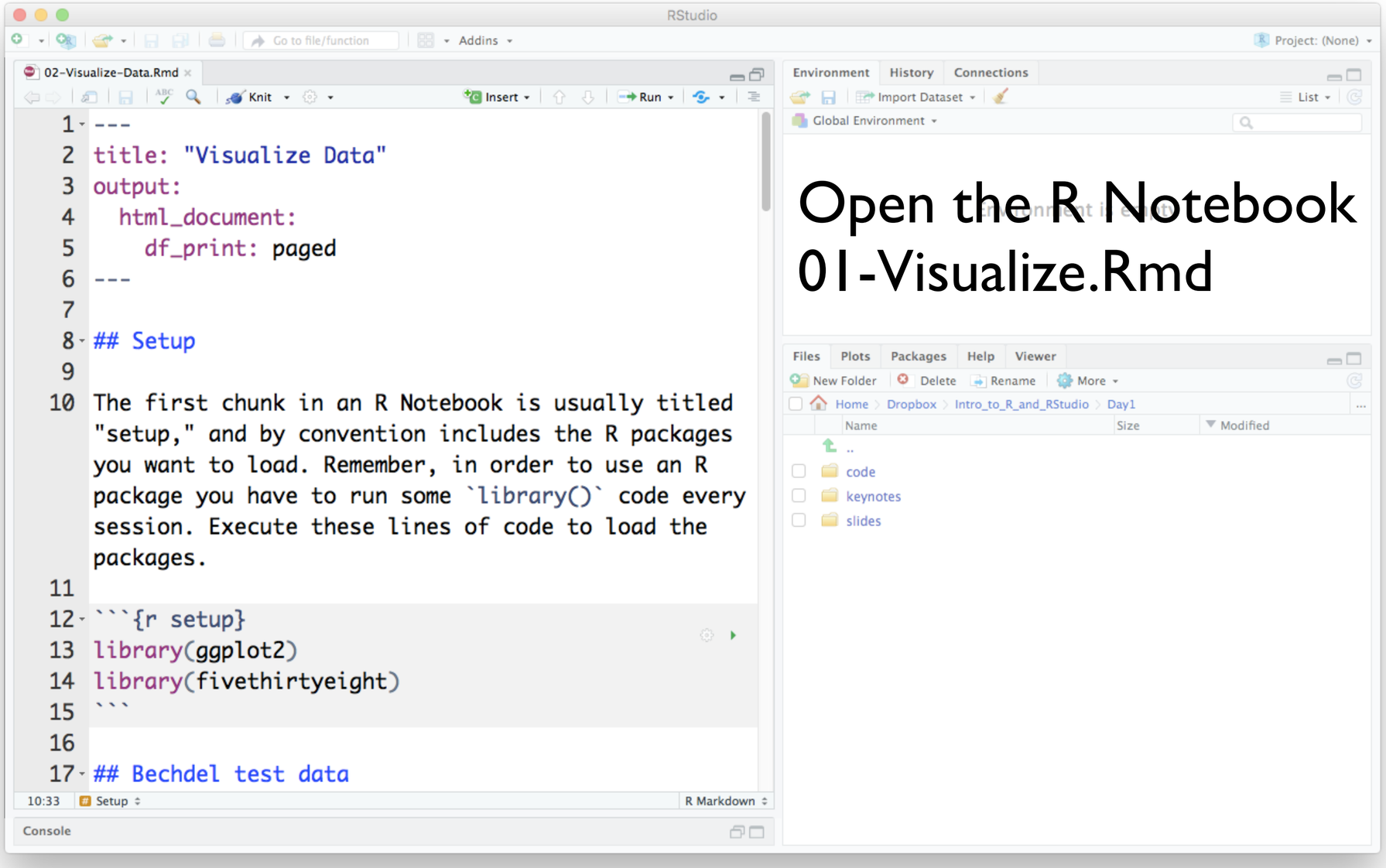
- John Tukey

(Applied) **Data Science**





Display by Adolfo Álvarez

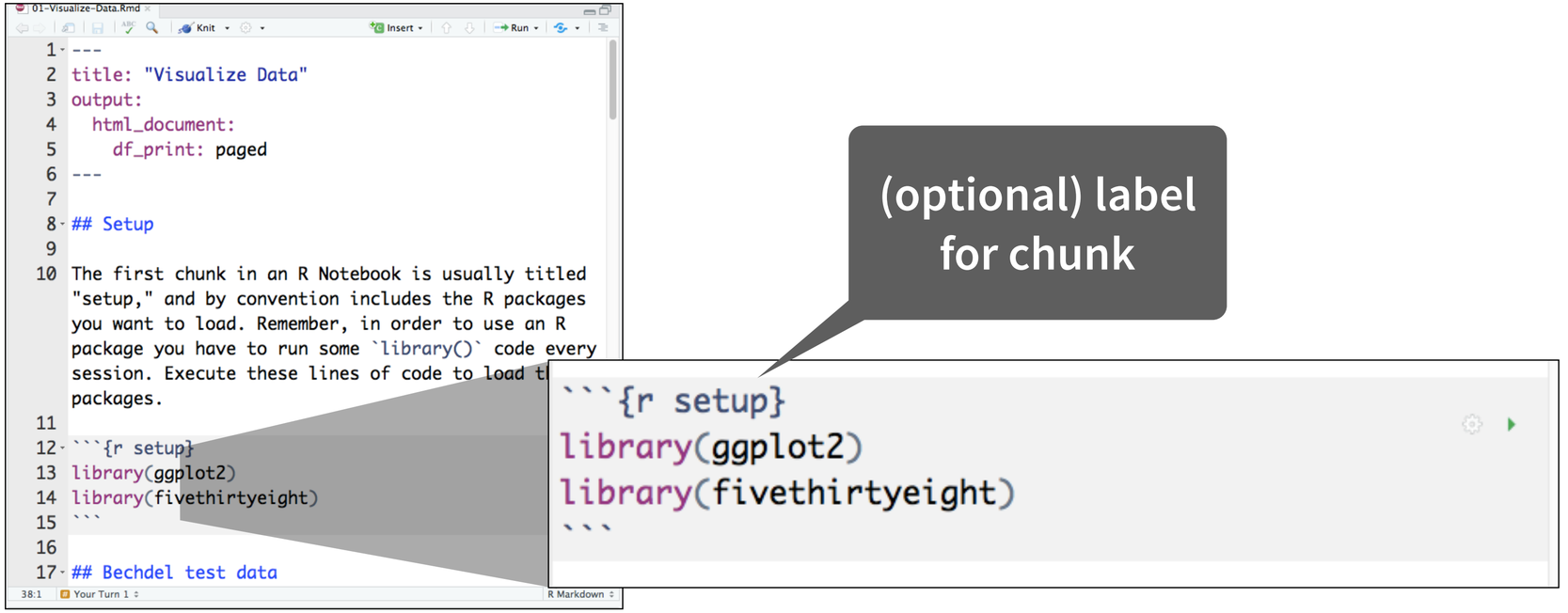

RStudio
Go to file/function
Addins
Project: (None)
02-Visualize-Data.Rmd
Knit
Insert
Run
---
title: "Visualize Data"
output:
html_document:
df_print: paged

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
01-Visualize.Rmd
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Dropbox
Intro_to_R_and_RStudio
Day1
Name
Size
Modified
code
keynotes
slides


ggplot2

Setup

The setup chunk is always run once before anything else



bechdel

Data on movies and the Bechtel test

```
bechdel
```

```
?bechdel
```

Consider

Confer with the people around you.

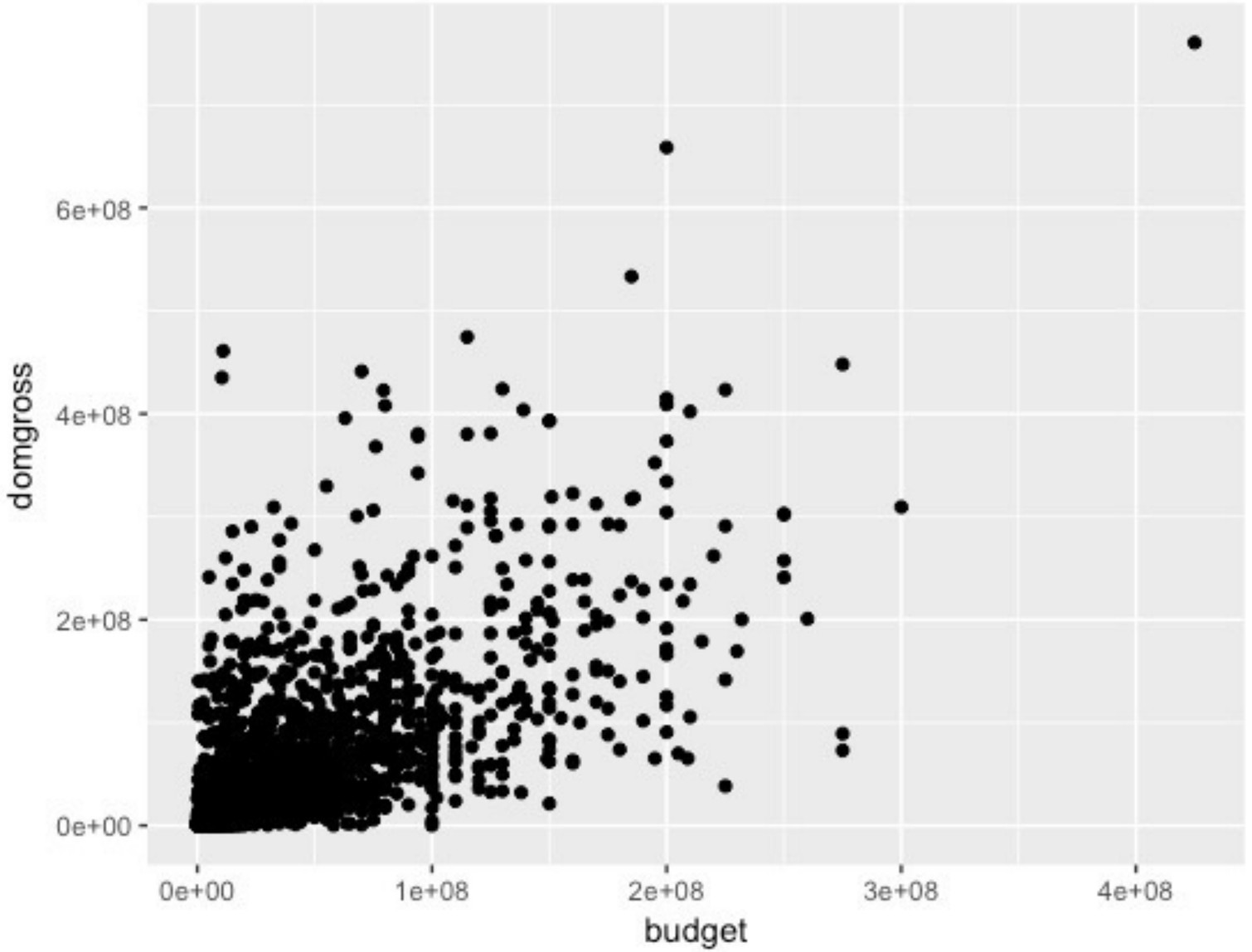
What relationship do you expect to see between movie budget (`budget`) and domestic gross(`domgross`)?

Your Turn 1

Run this code in your notebook to make a graph.

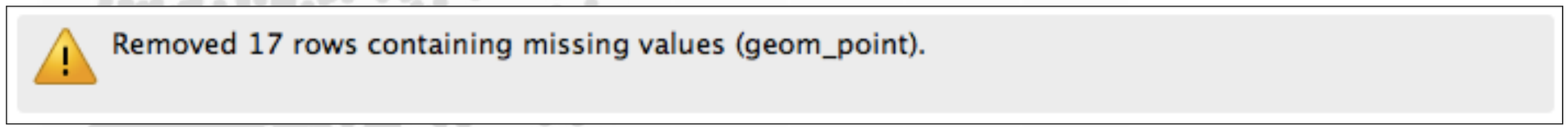
Pay strict attention to spelling, capitalization, and parentheses!

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




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

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




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

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

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

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

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

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>))
```

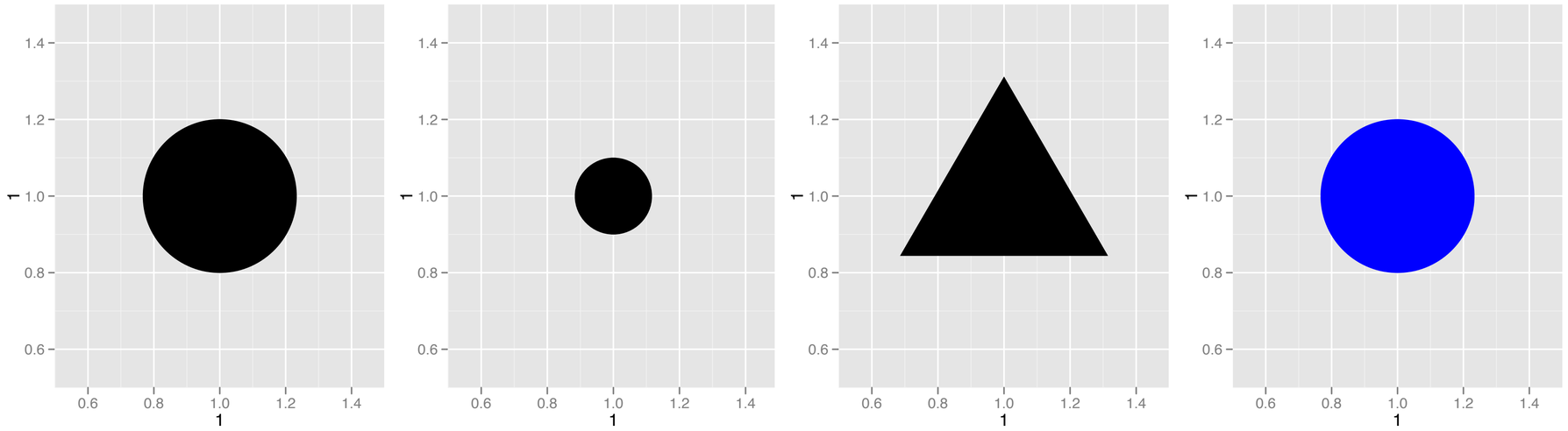
```
    geom_point(mapping = aes(x = budget, y = domgross))
```

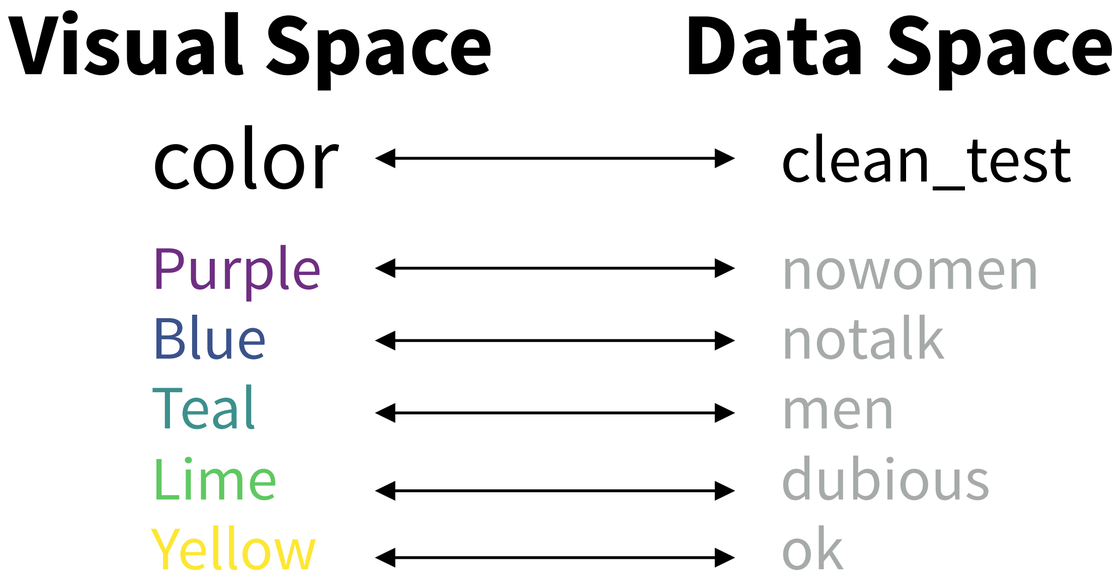
A template

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

Aesthetics

Aesthetics




Visual Space
Data Space
color
clean_test
Purple
nowomen
Blue
notalk
Teal
men
Lime
dubious
Yellow
ok

Aesthetics

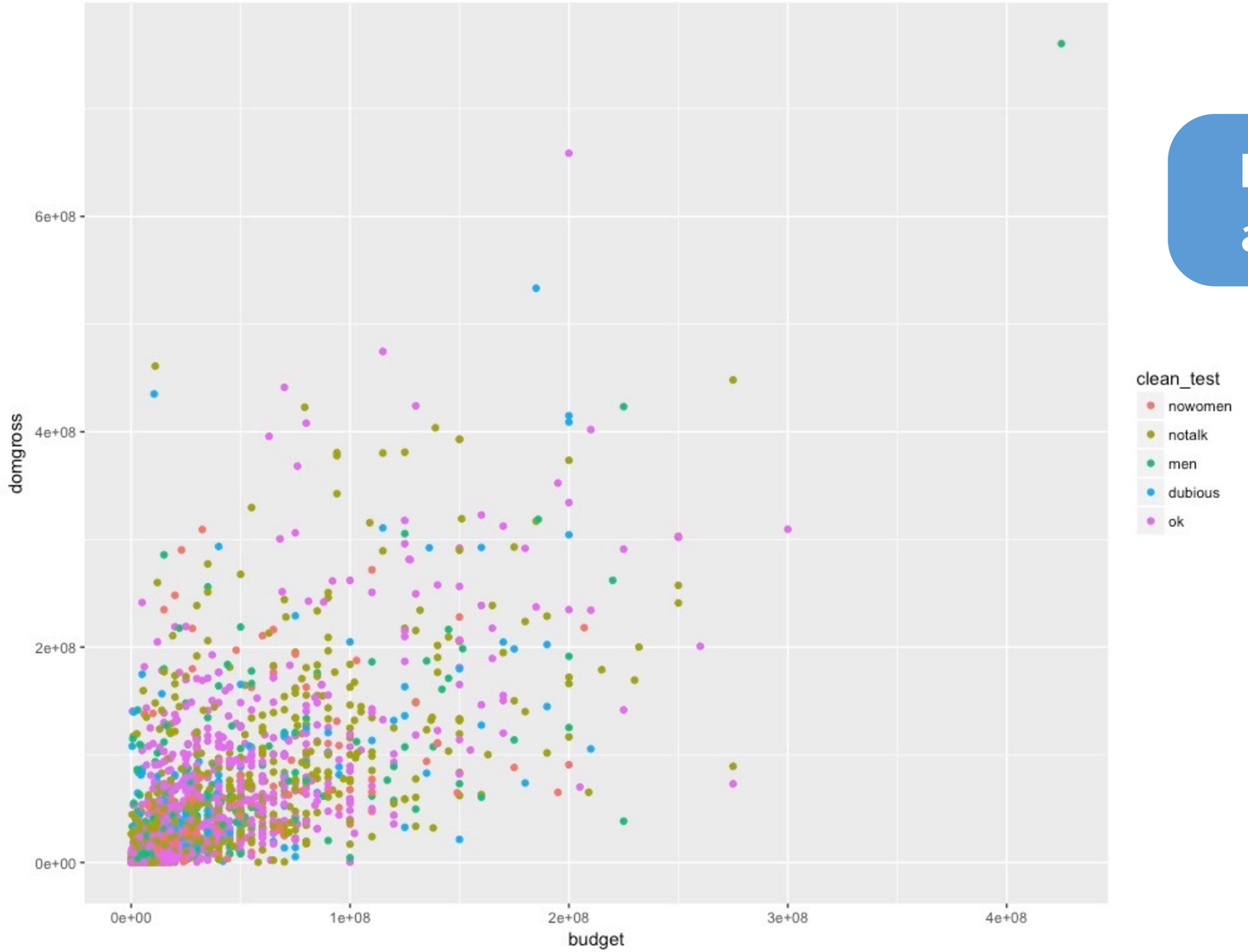
aesthetic property

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, color=clean_test))
```

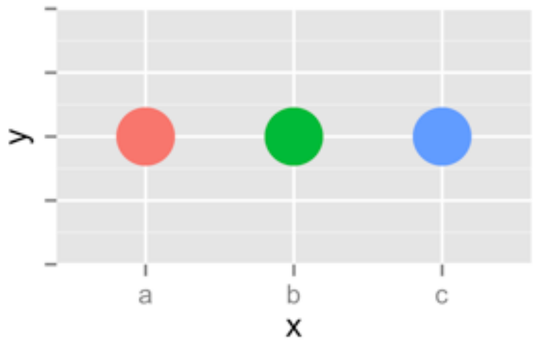
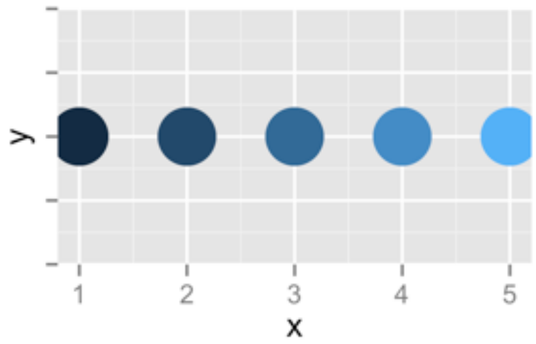
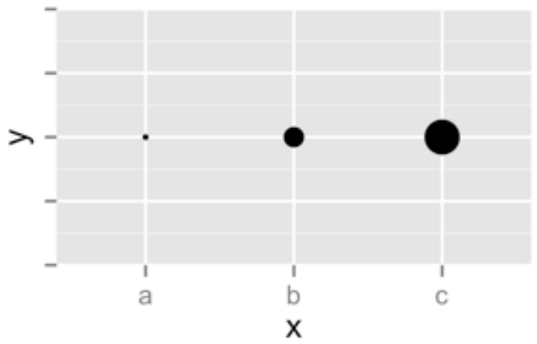
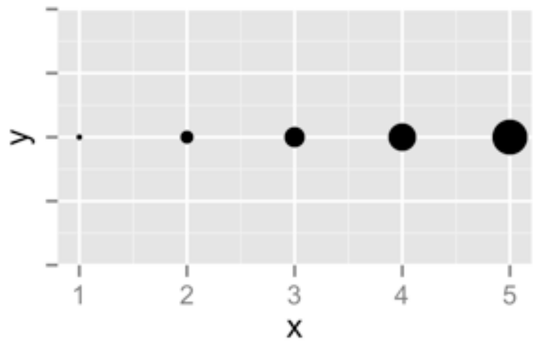
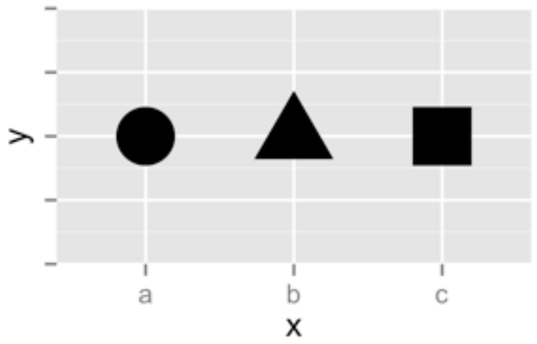


Legend added automatically

Your Turn 2

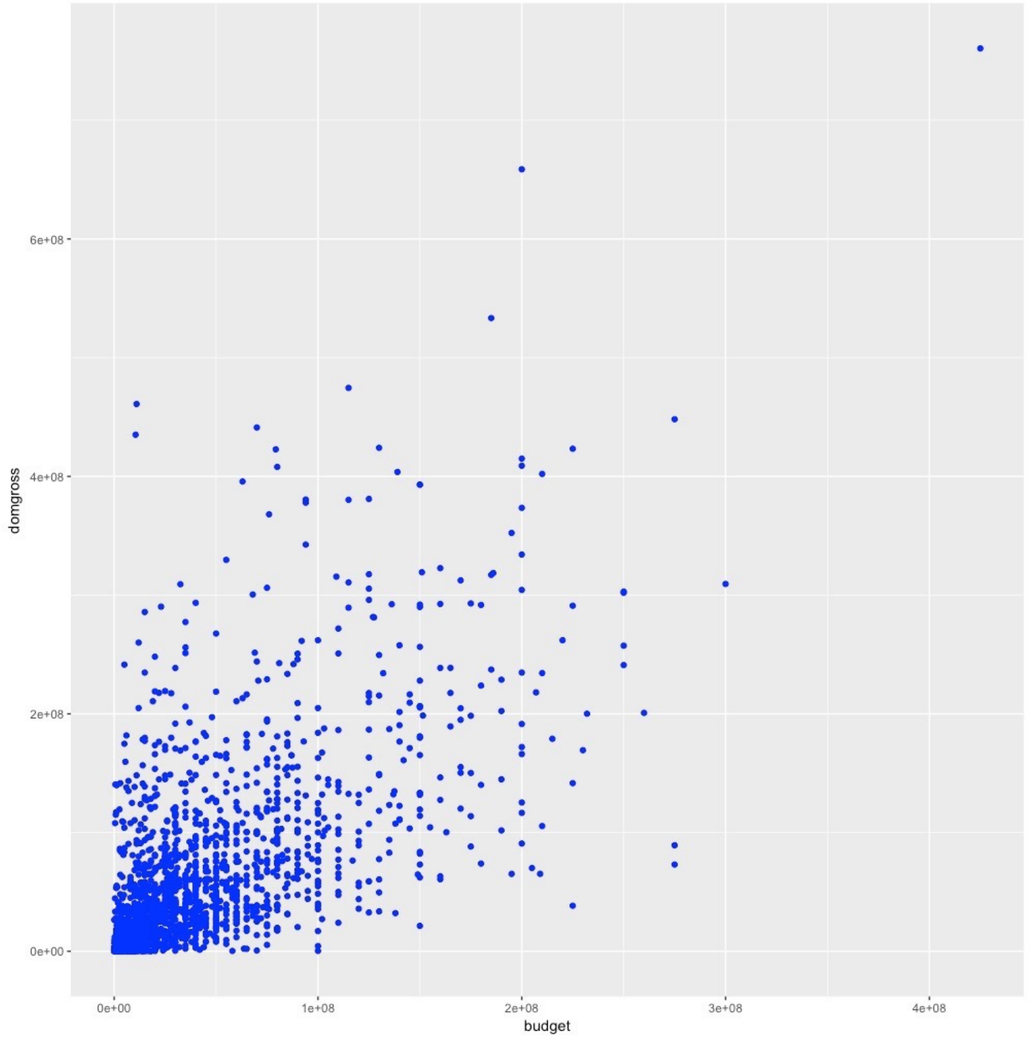
In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

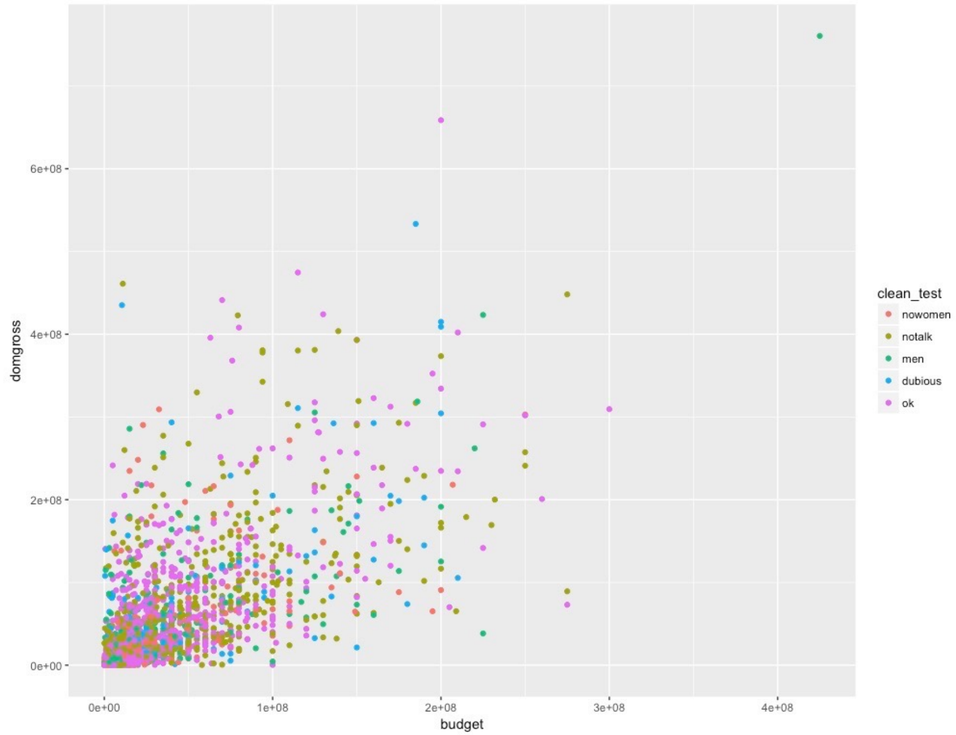
- Do different things happen when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

set vs. map

How would you make this plot?

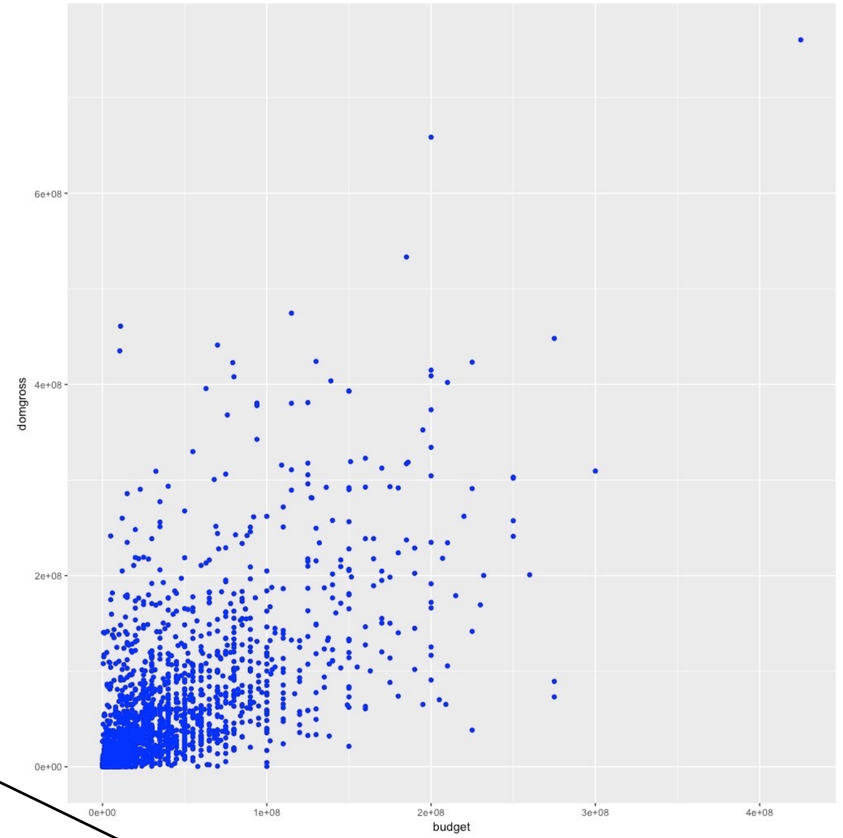




Inside of aes(): maps an aesthetic to a variable

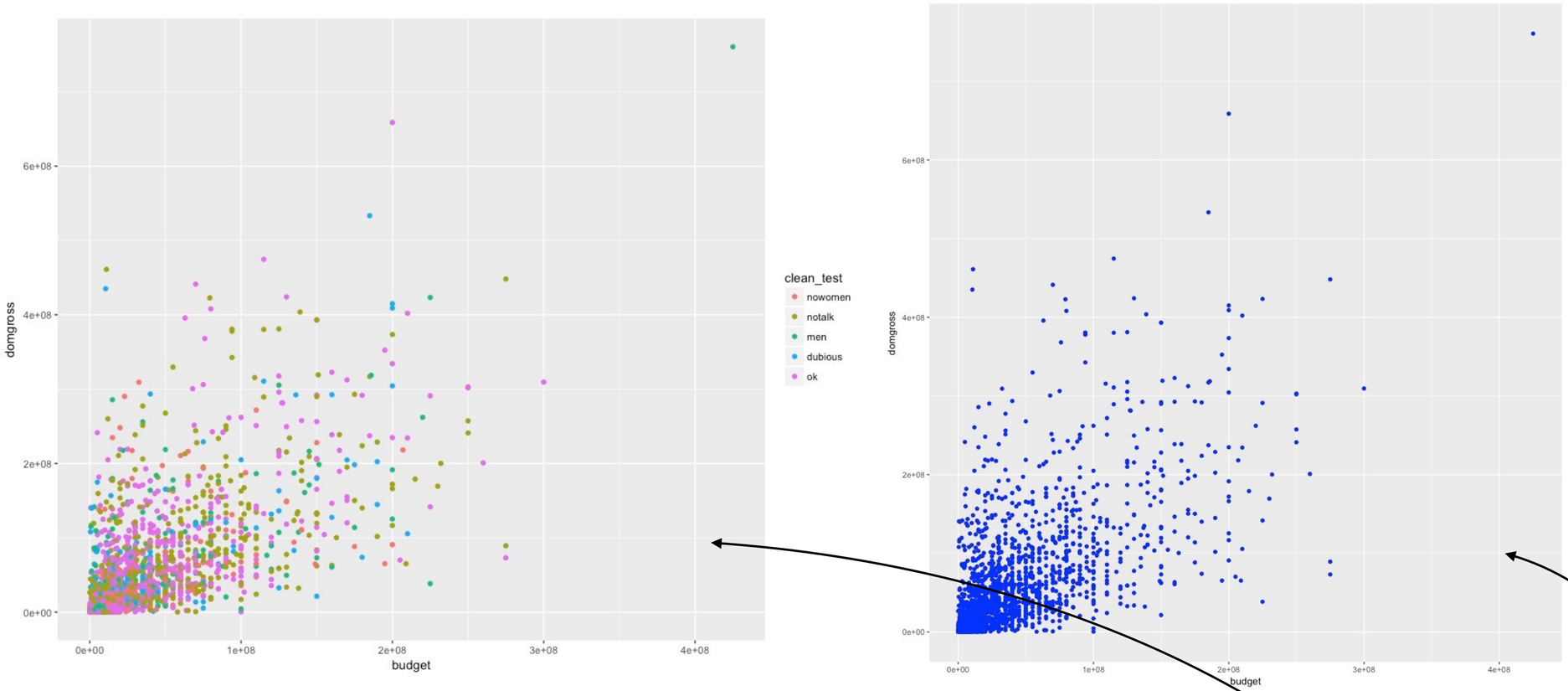
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

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



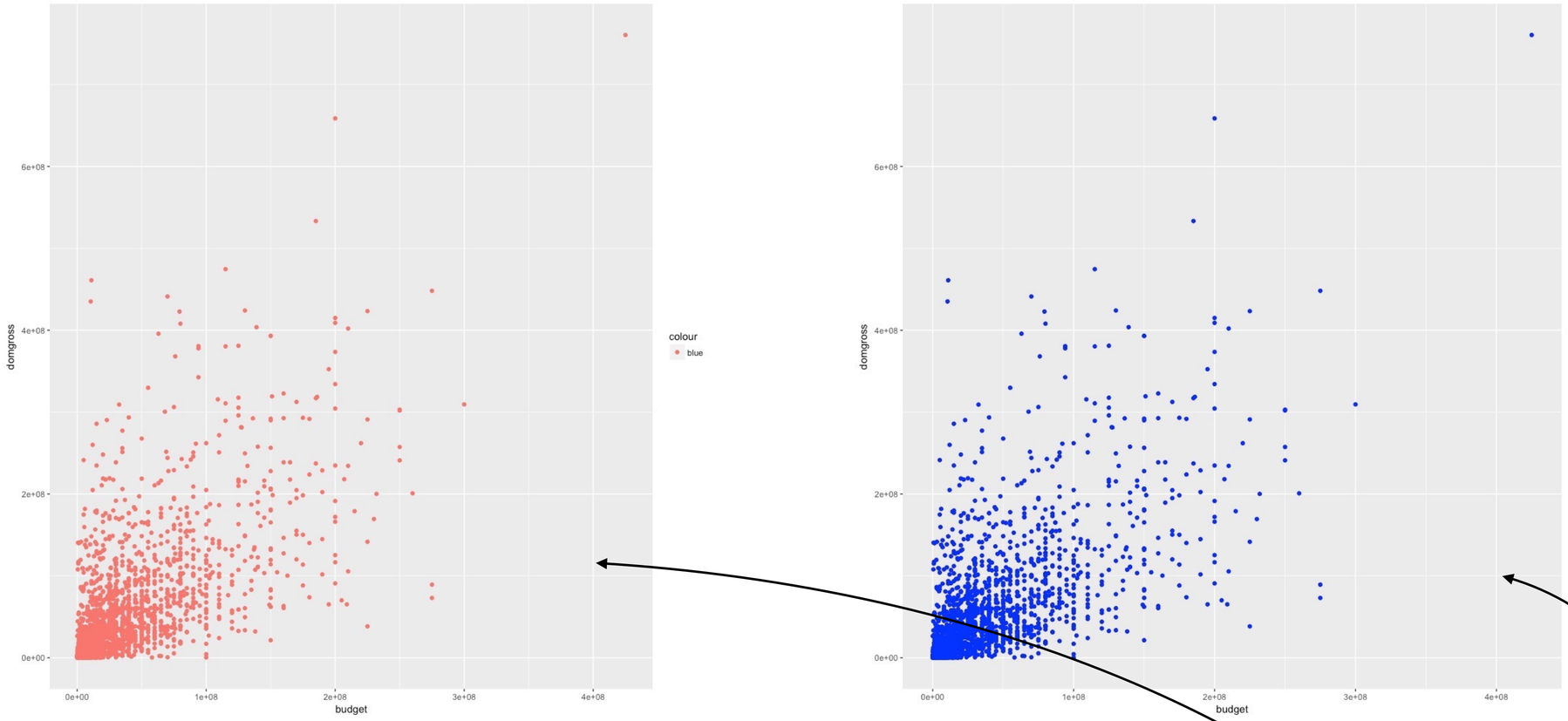

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test)
```

```
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")
```



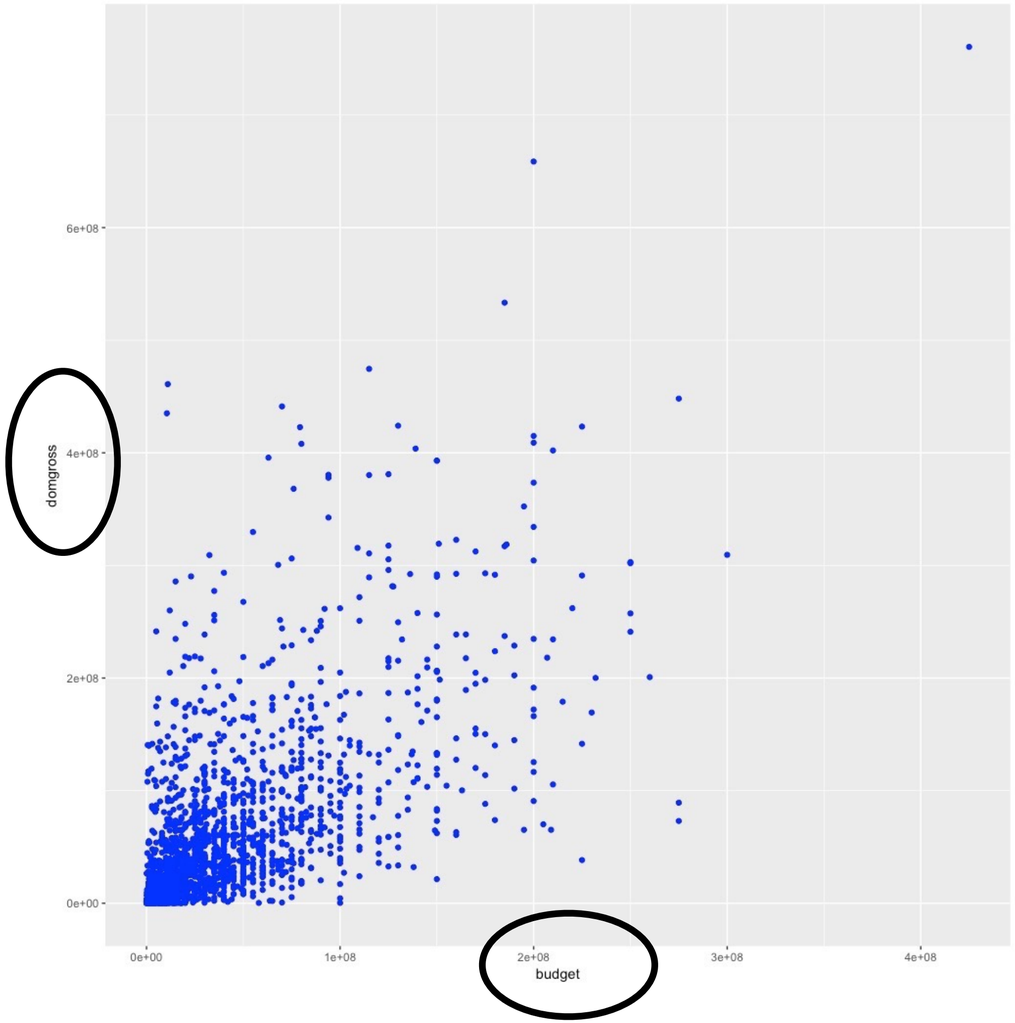

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color="blue"))
```

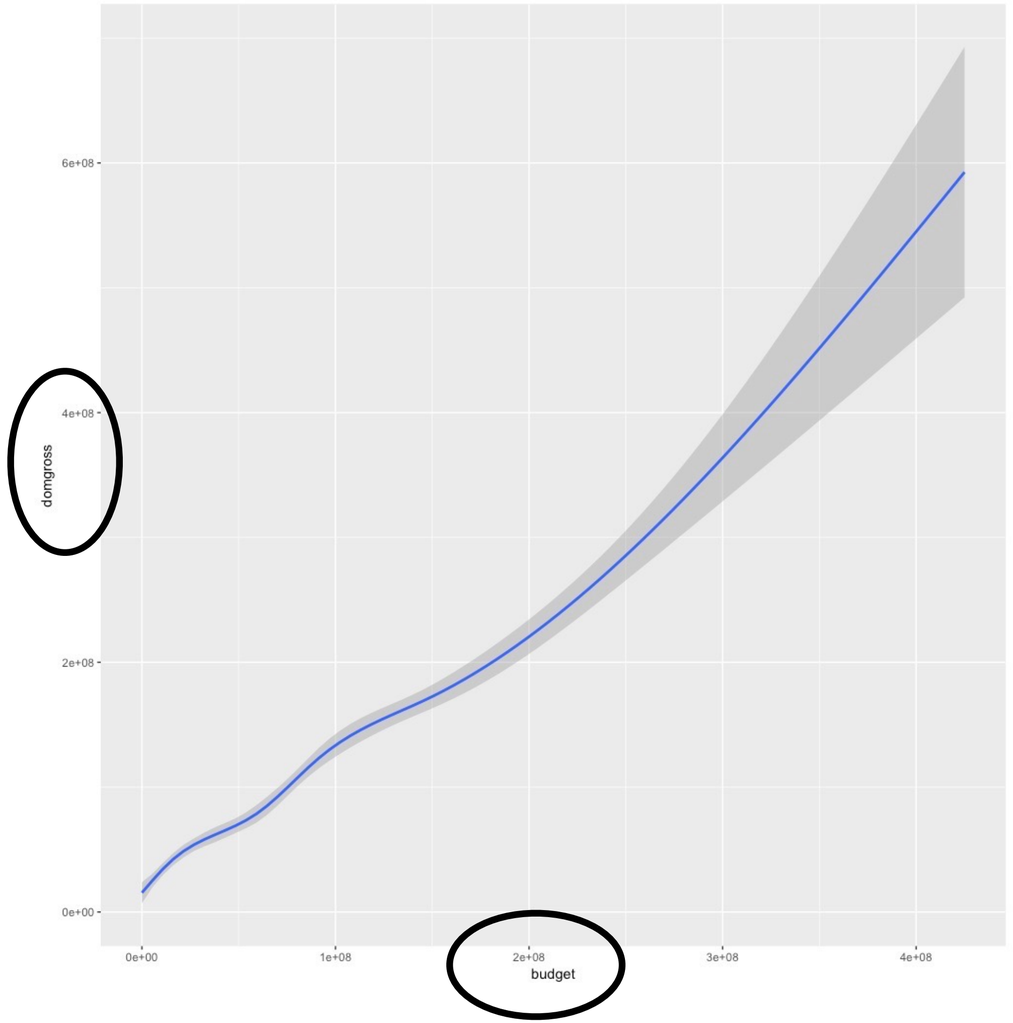
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

Geoms

How are these plots similar?

Same: x var , y var , data



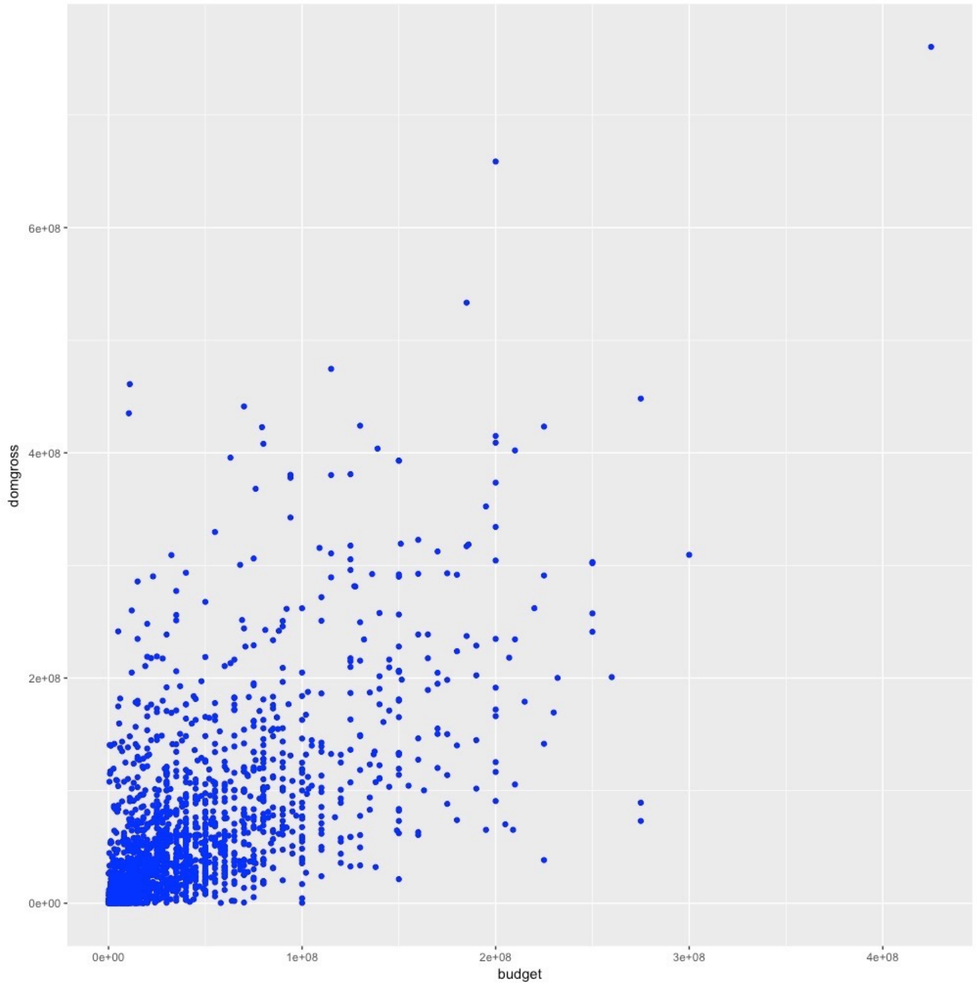


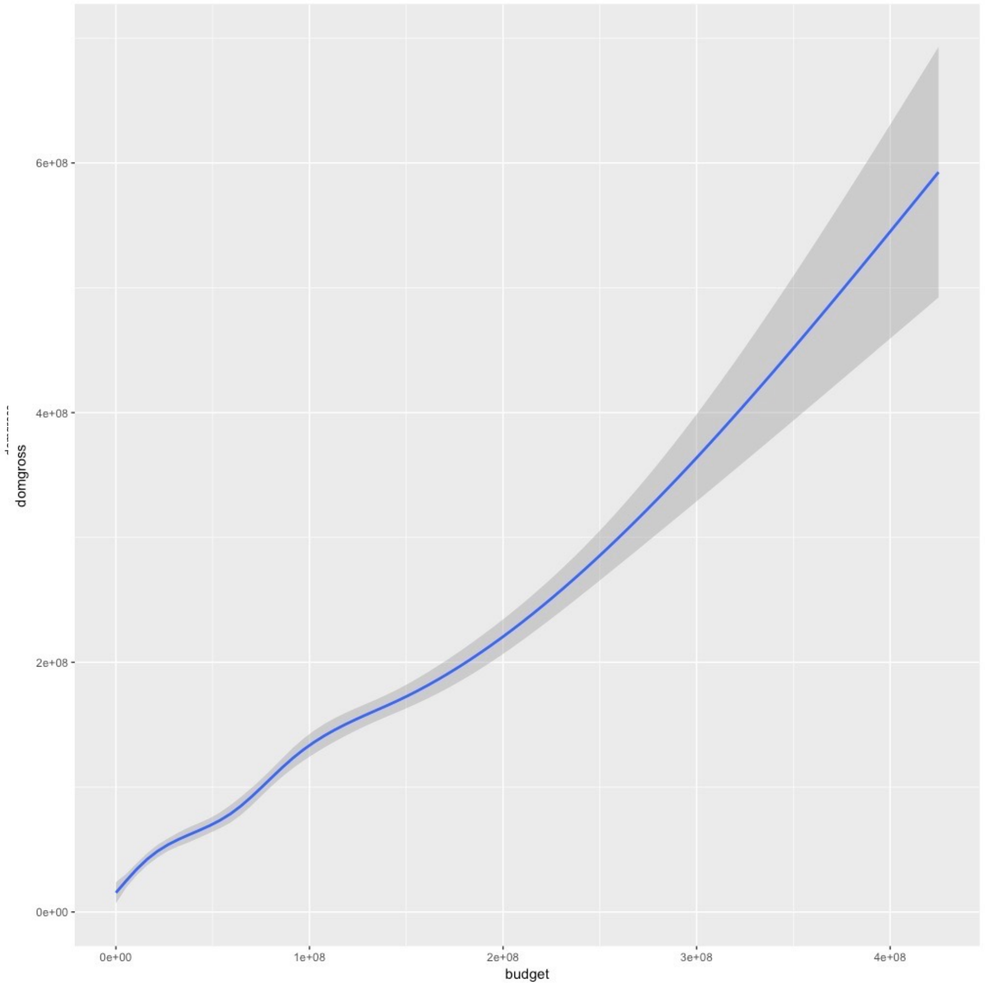
ggplot2

How are these plots different?

Different: geometric object (geom),

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



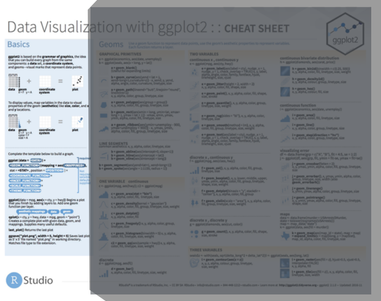


geoms

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

geom_ functions

Each requires a mapping argument.



Geoms

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

ggplot2

GRAPHICAL PRIMITIVES

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

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, 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

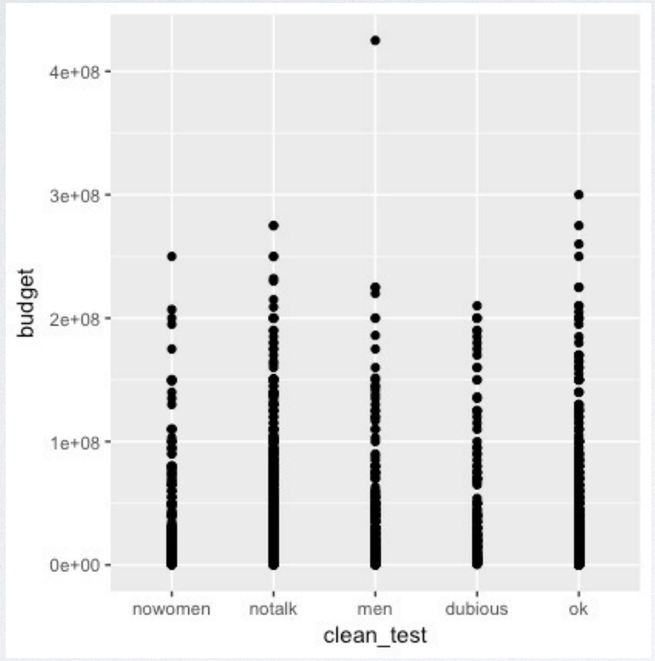
ggplot2

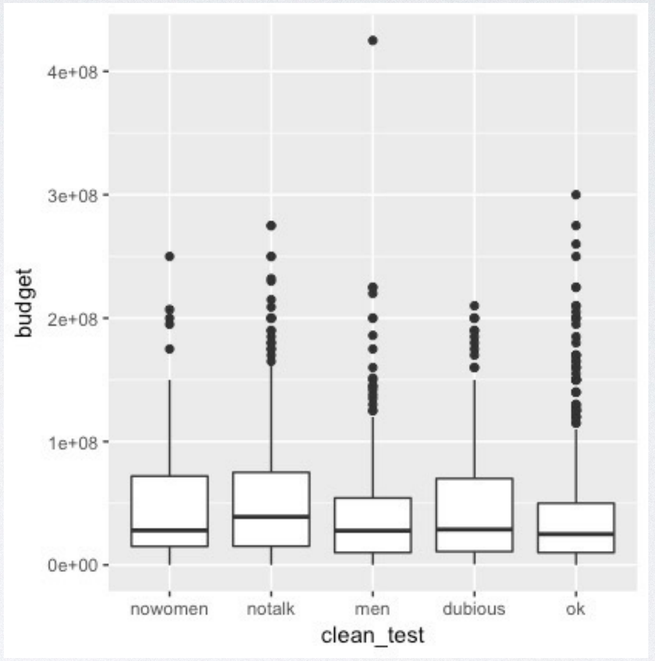
Your Turn

Pair up.

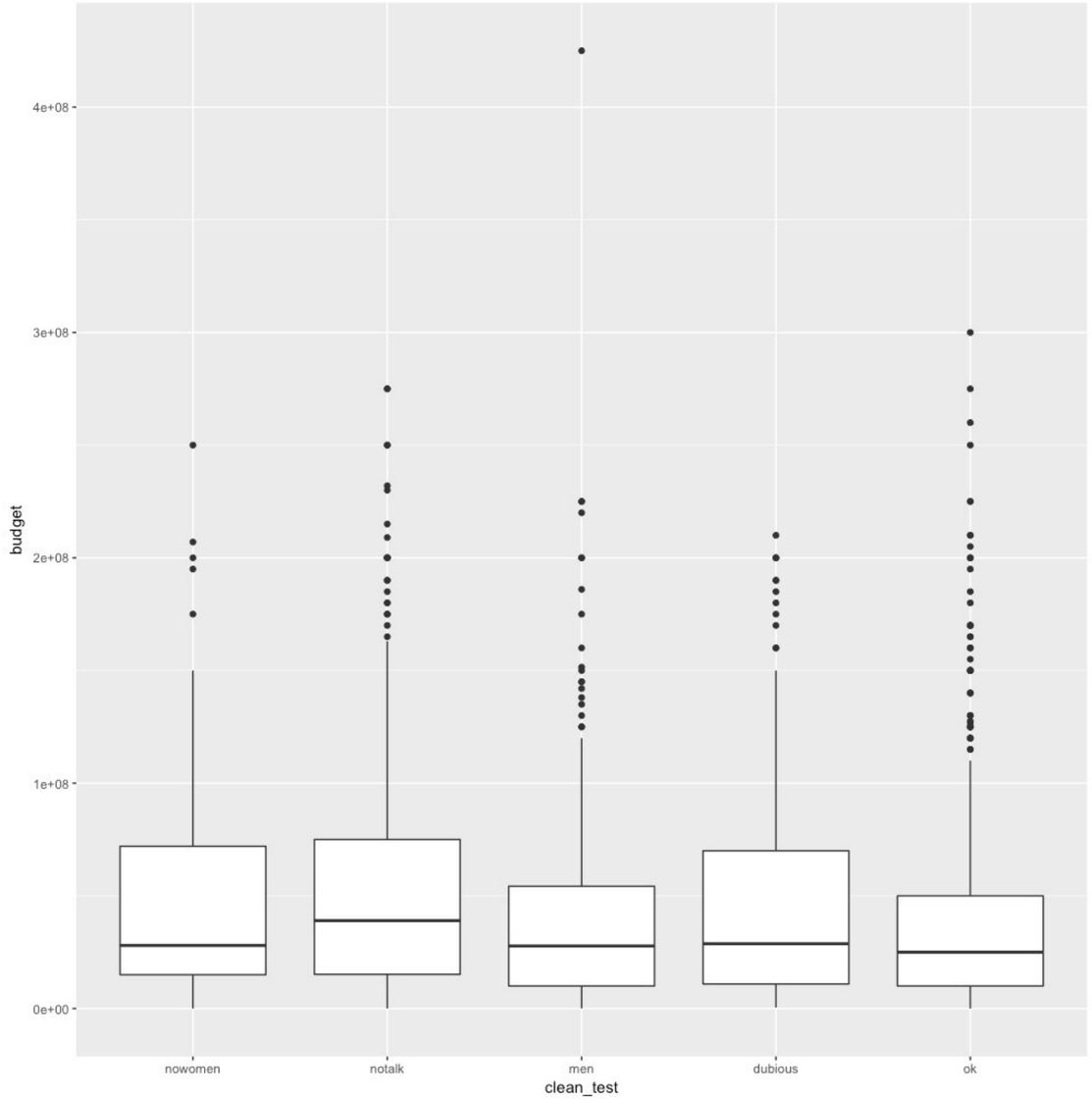
Your Turn 3

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





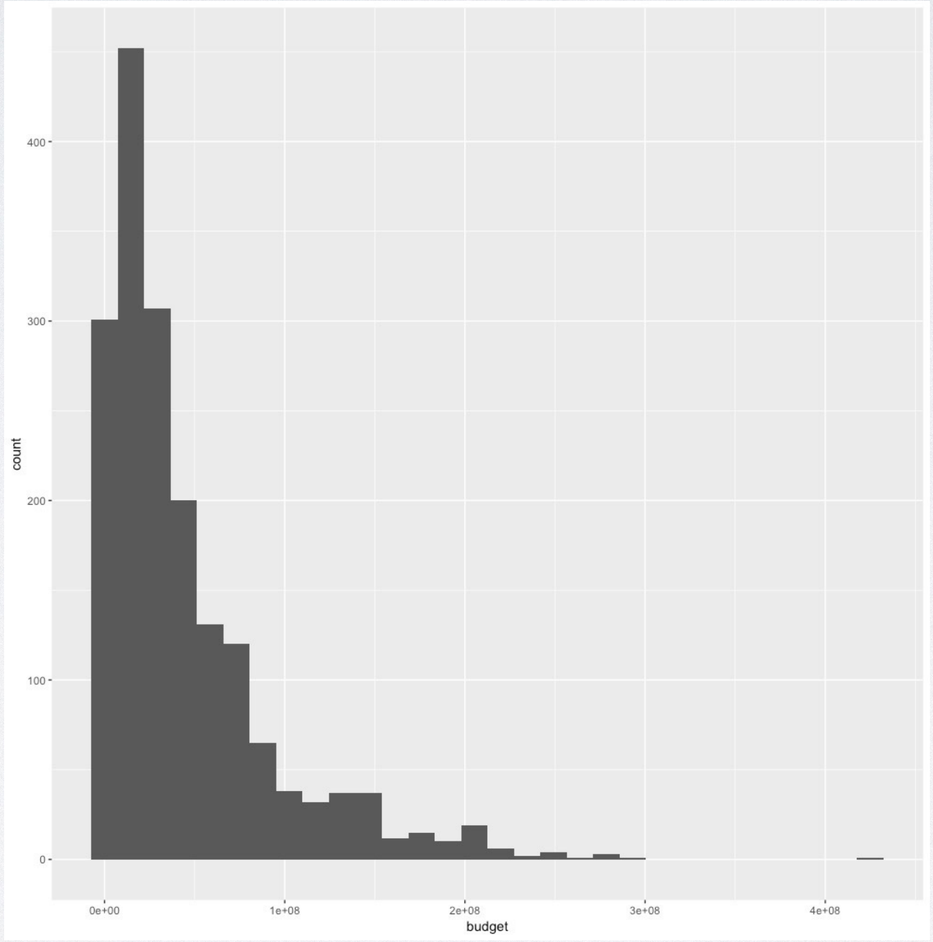

```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = clean_test, y = budget))
```



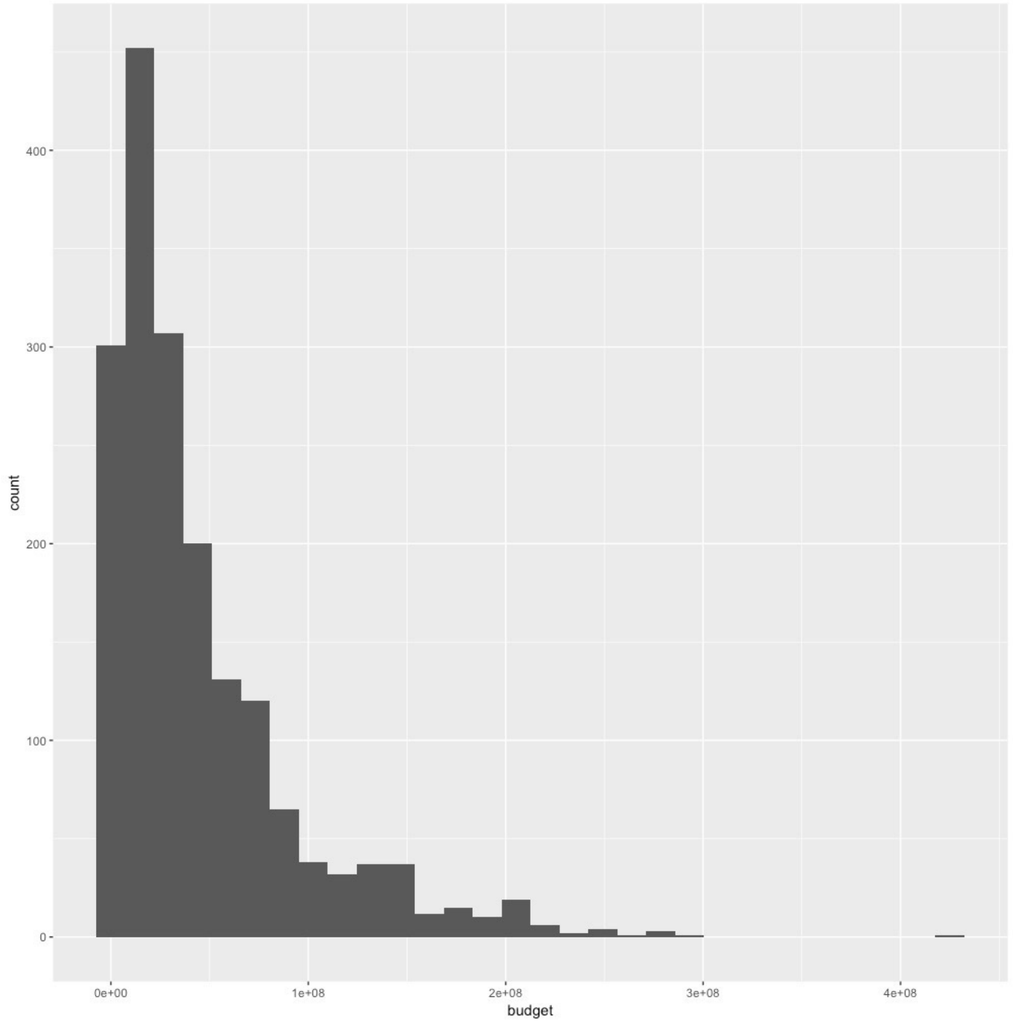

```
ggplot(data = bechdel) +
  geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

Your Turn 4

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

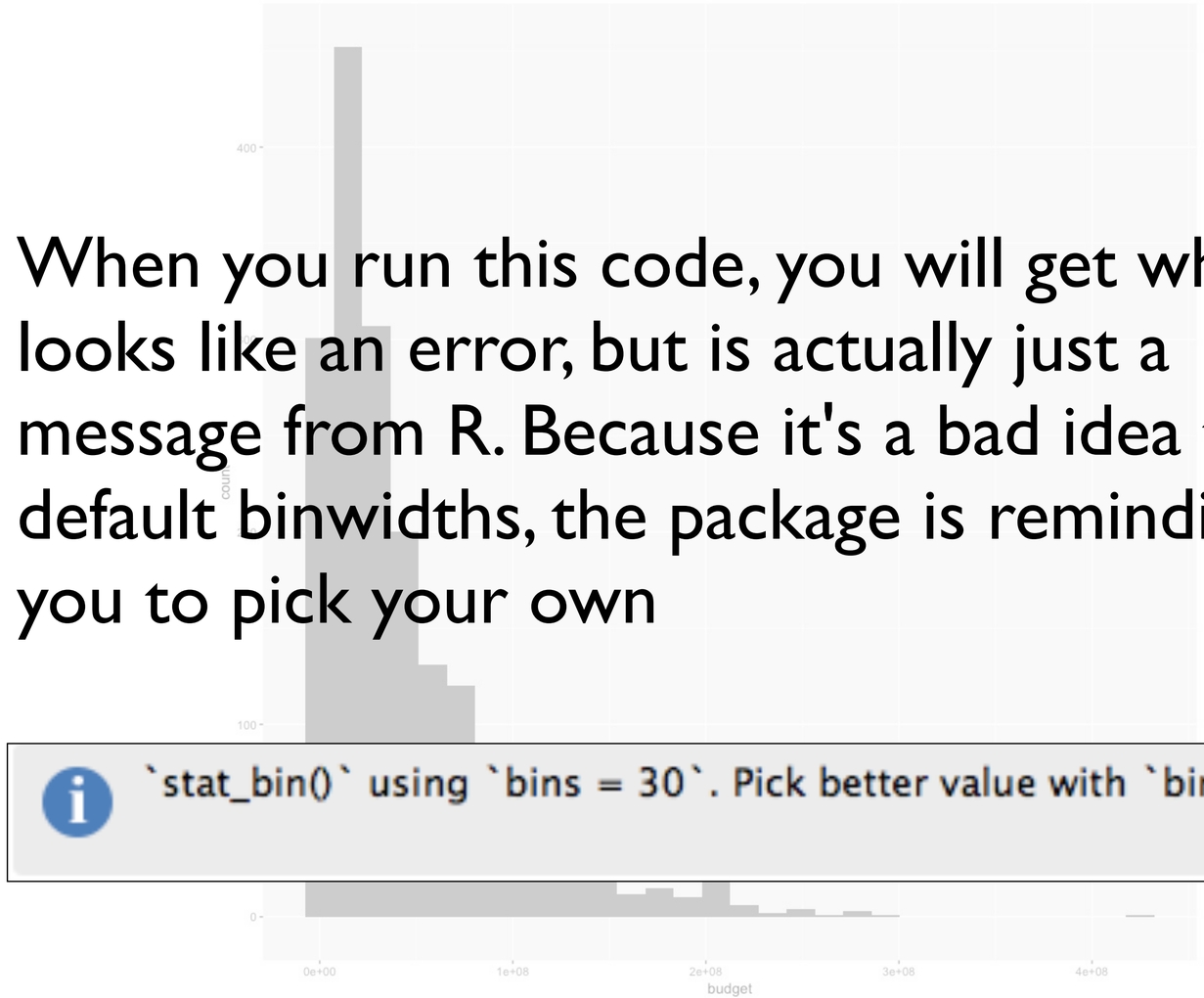


02:00




```
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



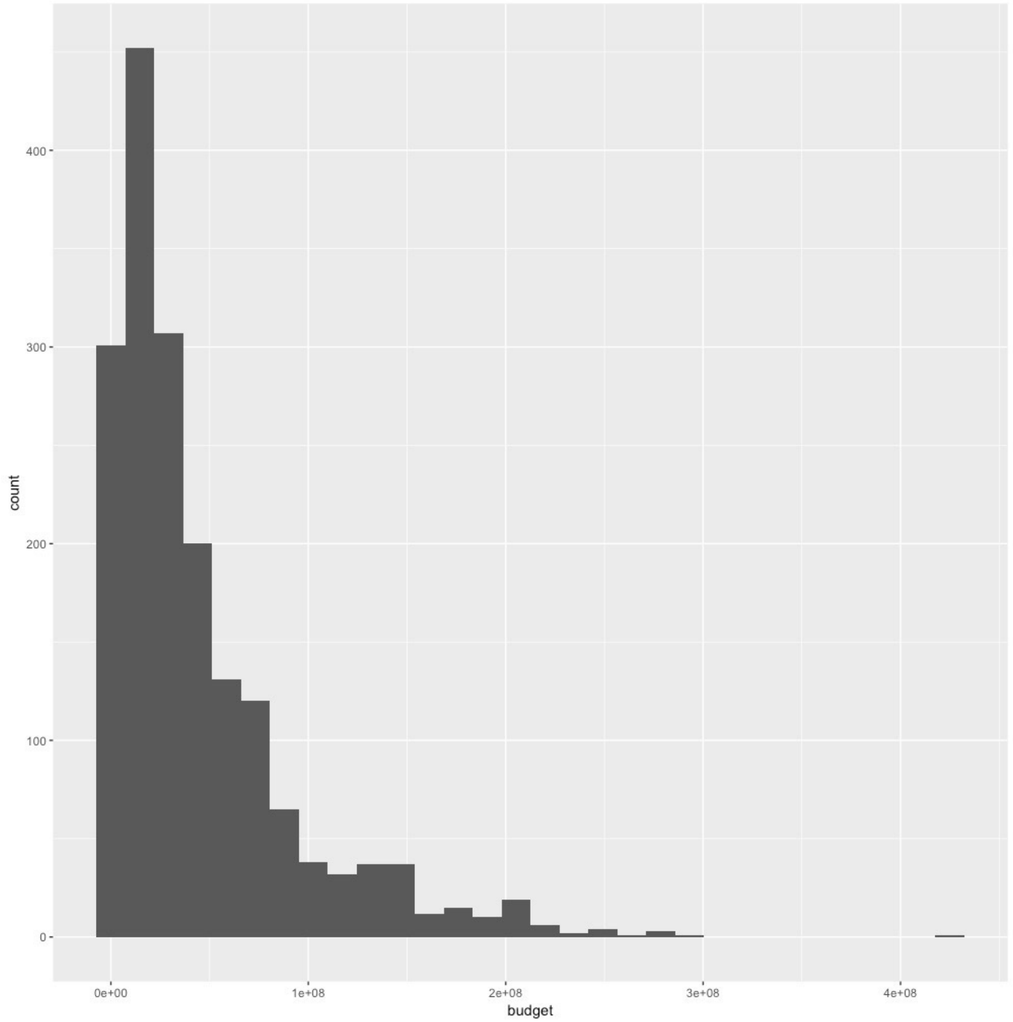

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

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

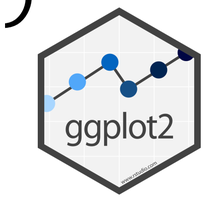
Your Turn 5

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

Try it out

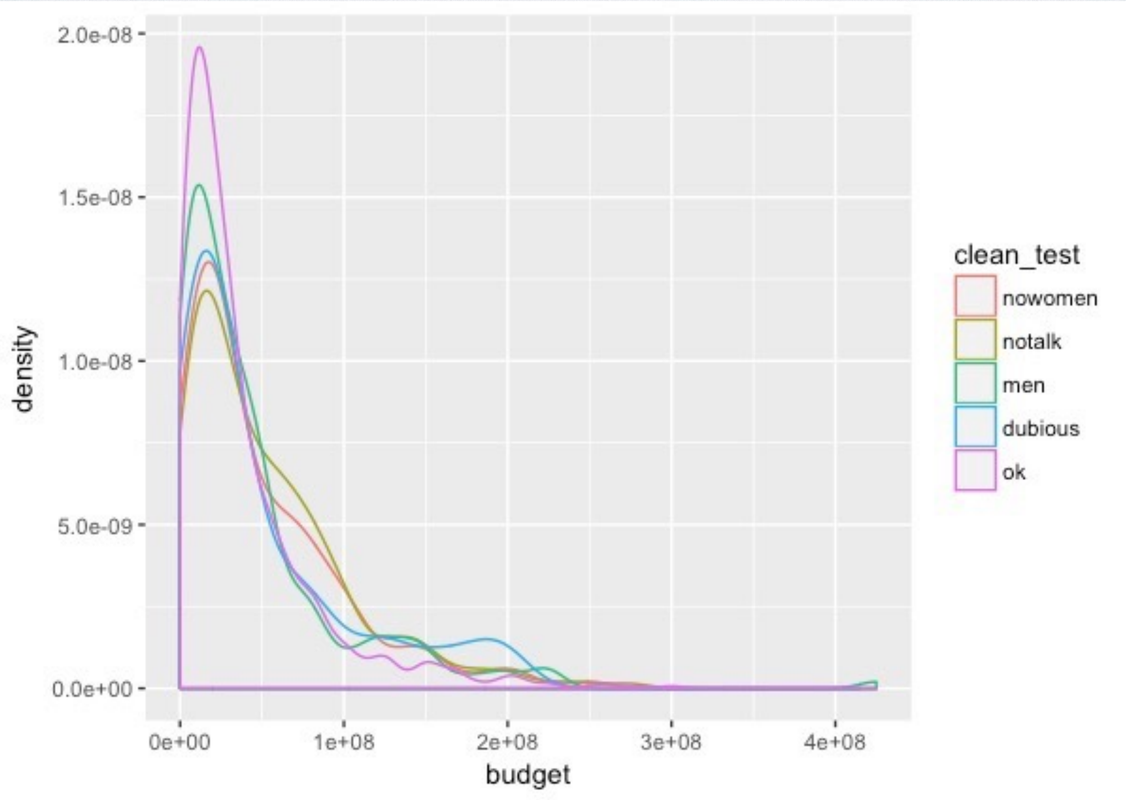



```
ggplot(data = bechdel) +
  geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```

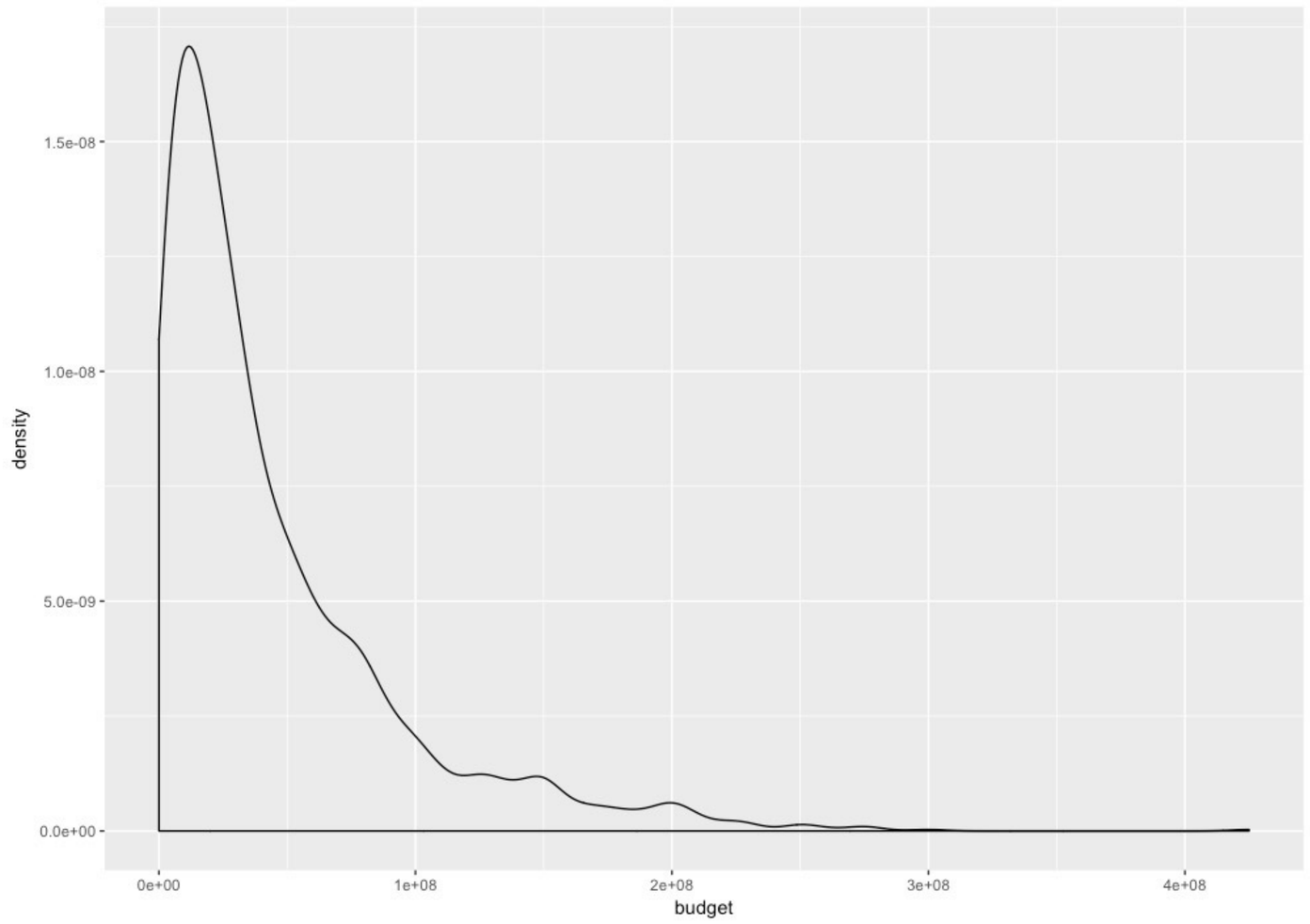


Your Turn 6

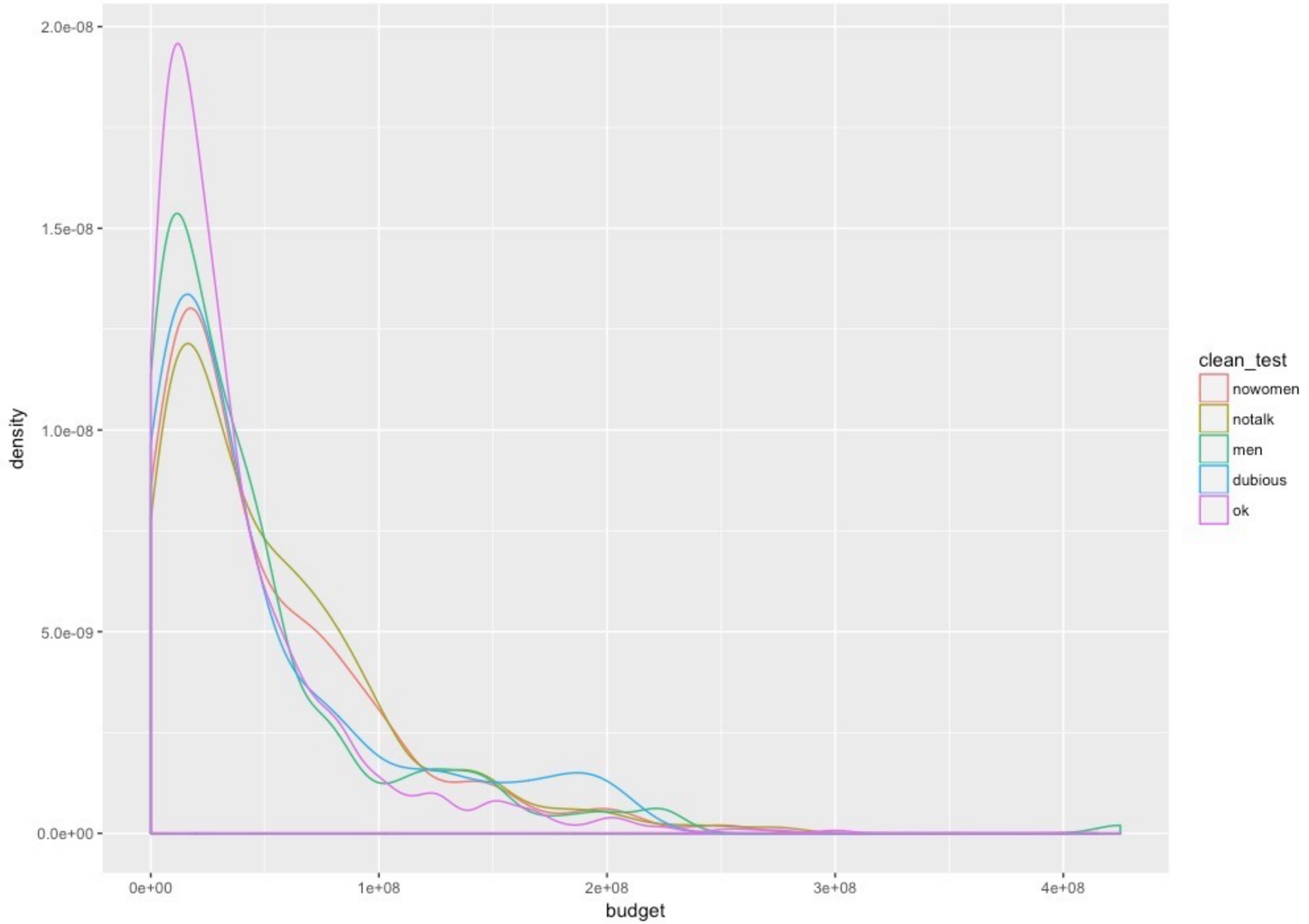
With your partner, make the density plot of **budget** colored by **clean_test** below. Use the cheatsheet. Try your best guess.



02:00



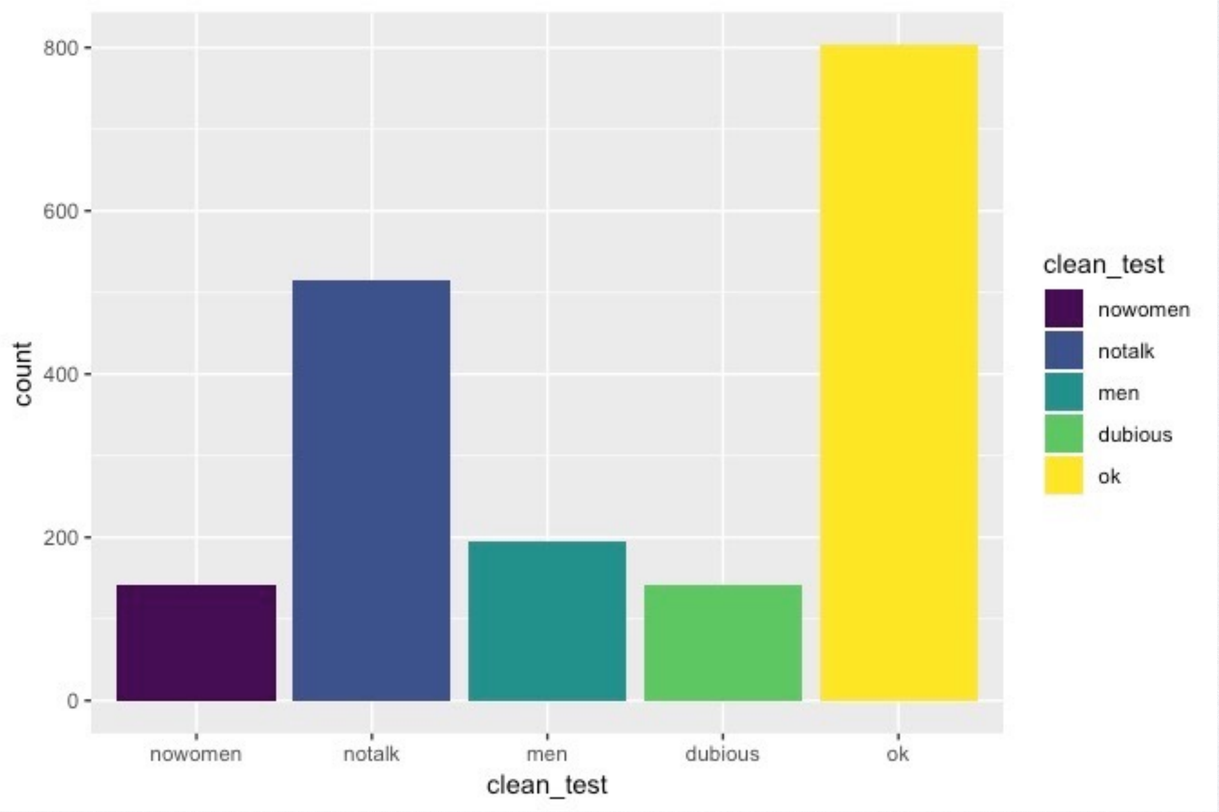

```
ggplot(data = bechdel) +
  geom_density(mapping = aes(x = budget))
```



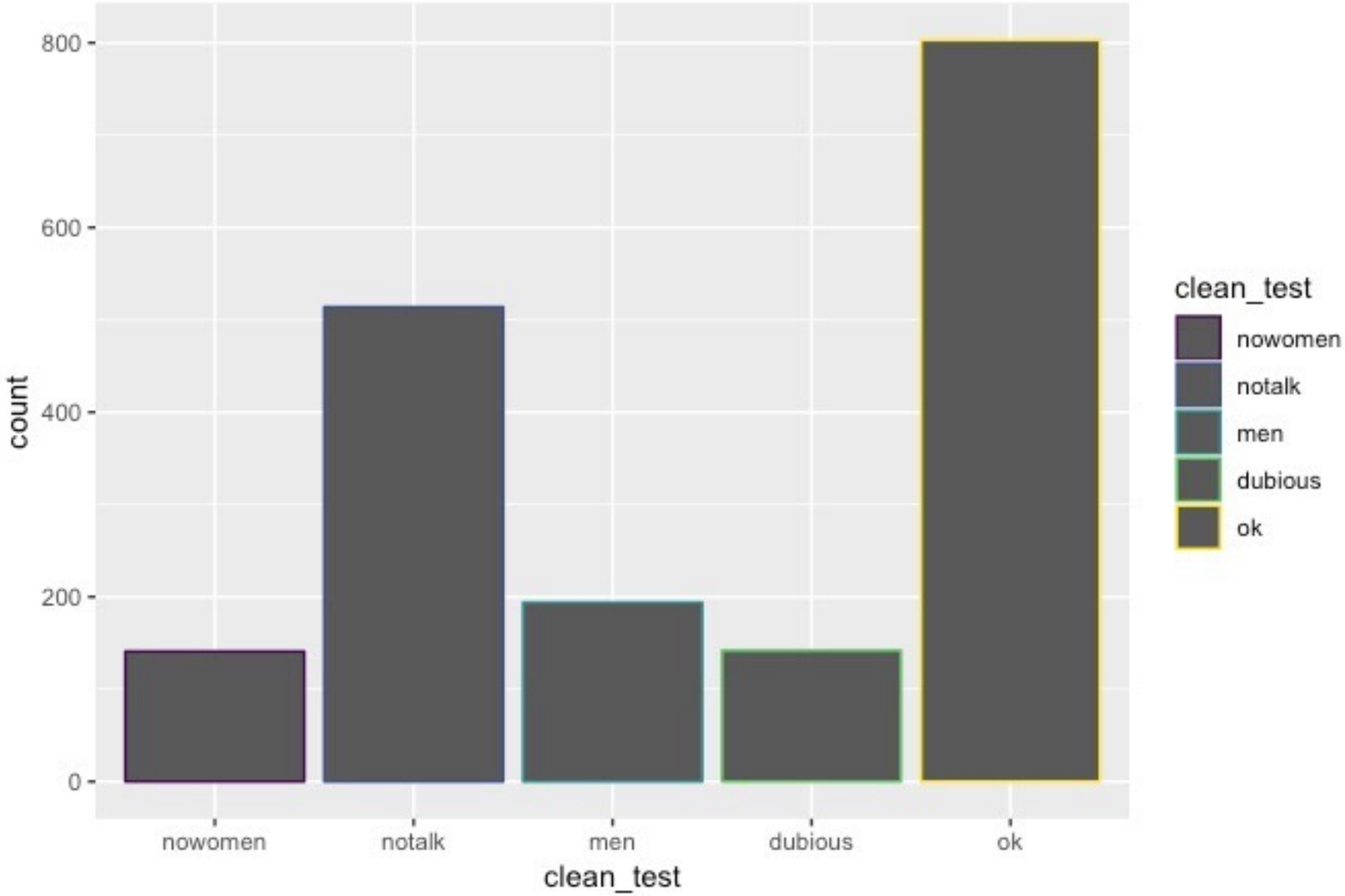

```
ggplot(data = bechdel) +
  geom_density(mapping = aes(x = budget, color=clean_test))
```

Your Turn 7

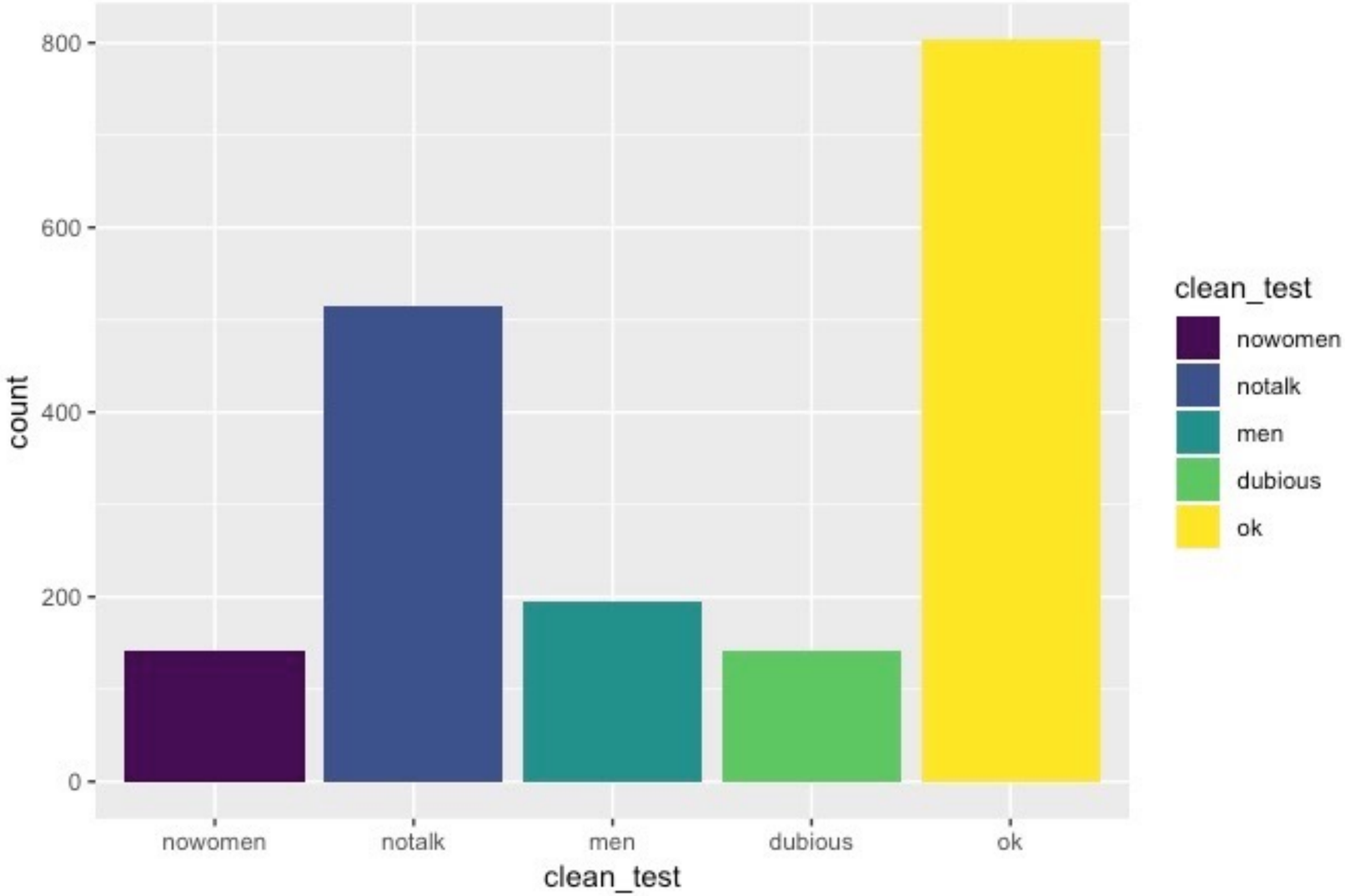
With your partner, make the bar chart of **clean_test** colored by **clean_test** below. Use the cheatsheet. Try your best guess.



02:00




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



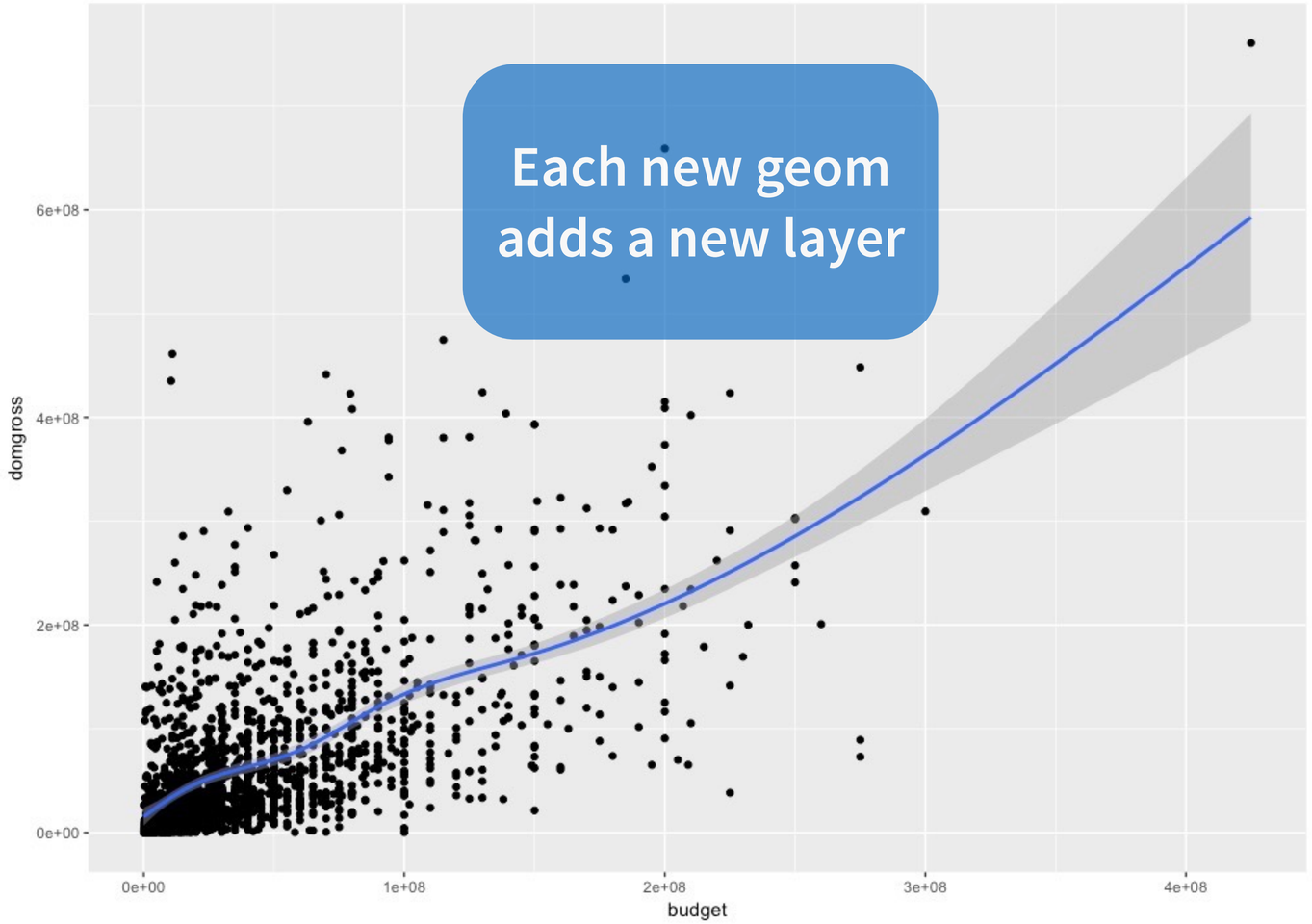

```
ggplot(data=bechdel) +
    geom_bar(aes(x=clean_test, fill=clean_test))
```

Your Turn 8

With a partner, predict what this code will do.

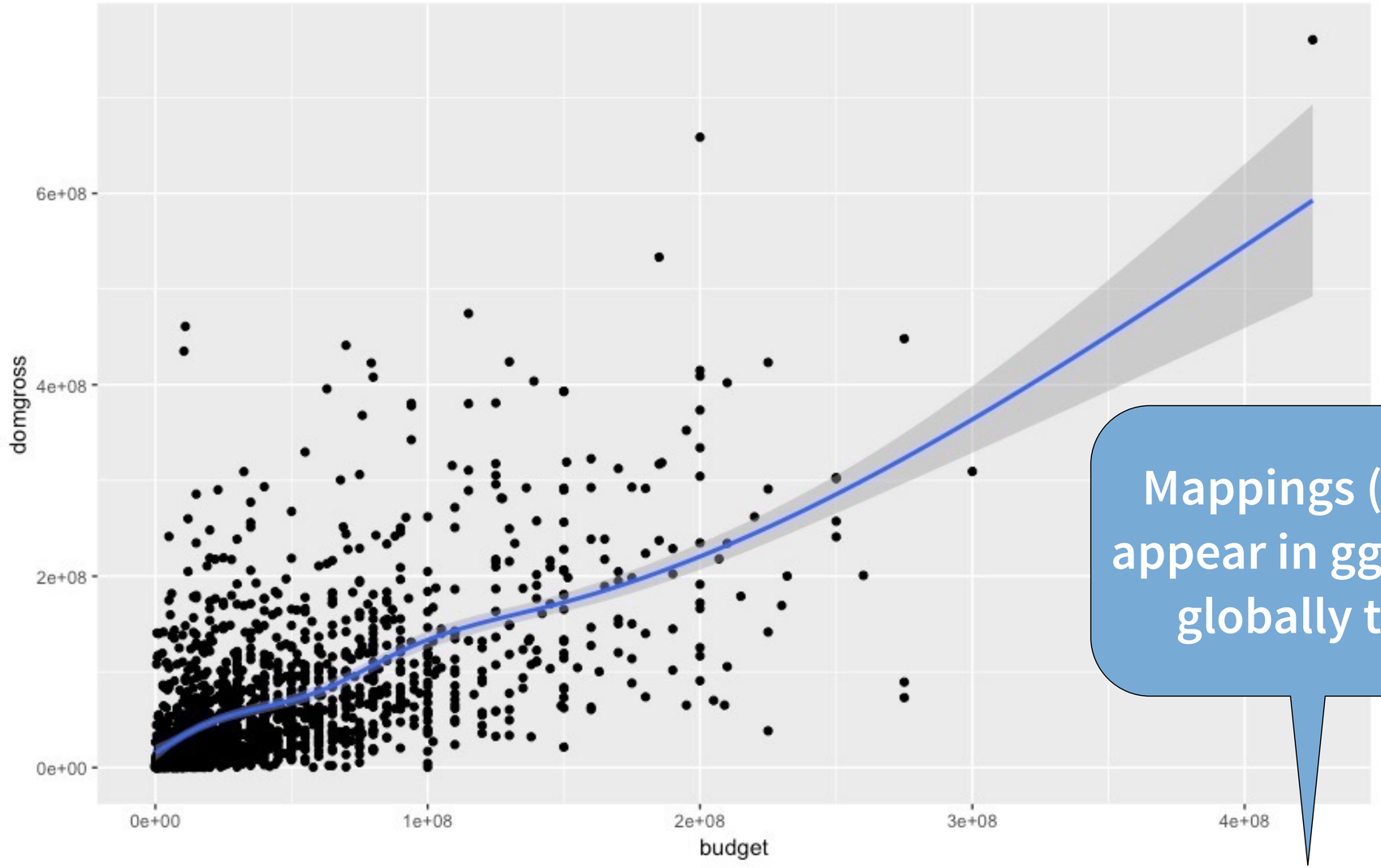
Then run it.

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



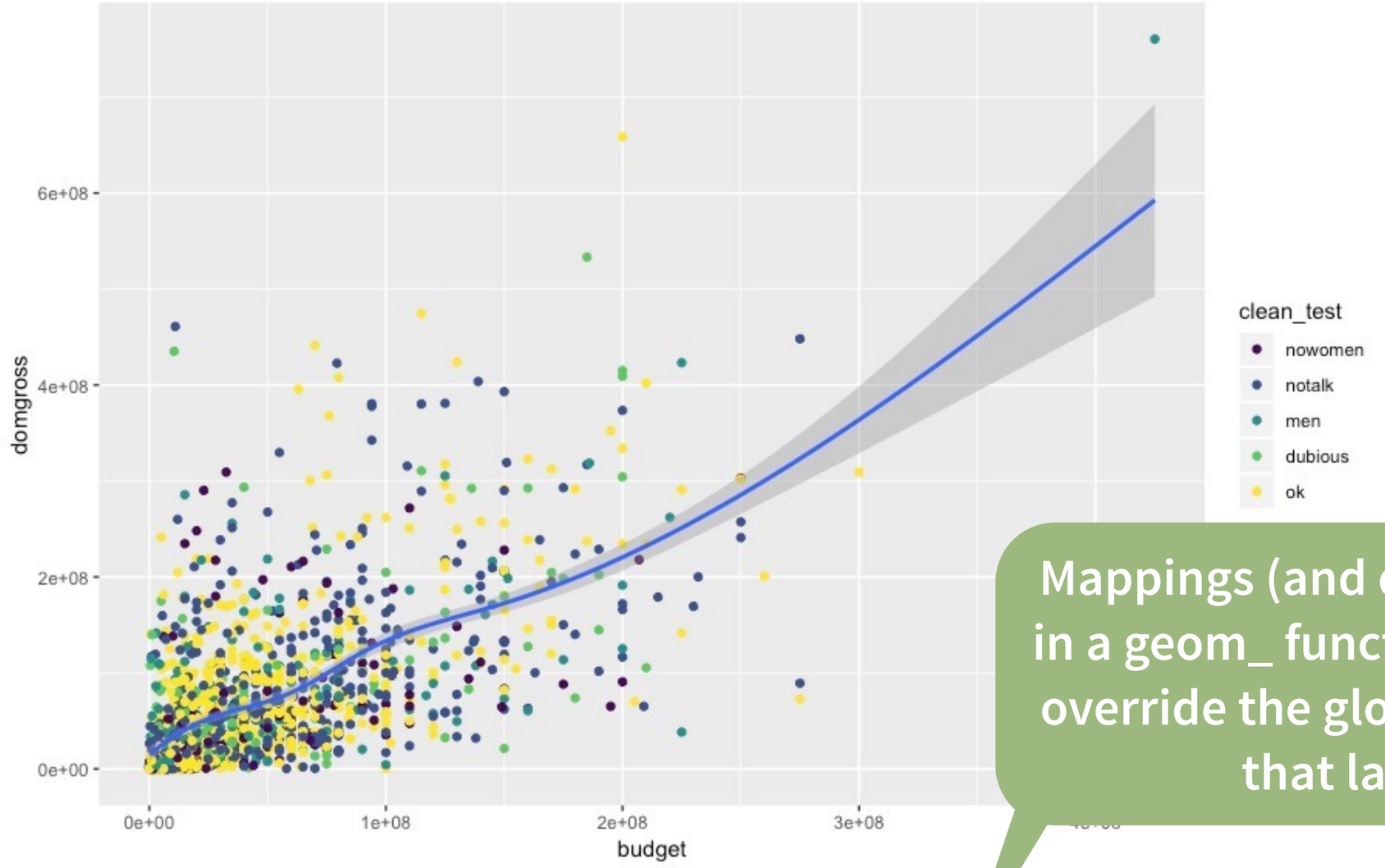

```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = budget, y = domgross)) +
  geom_smooth(mapping = aes(x = budget, y = domgross))
```

global vs. local



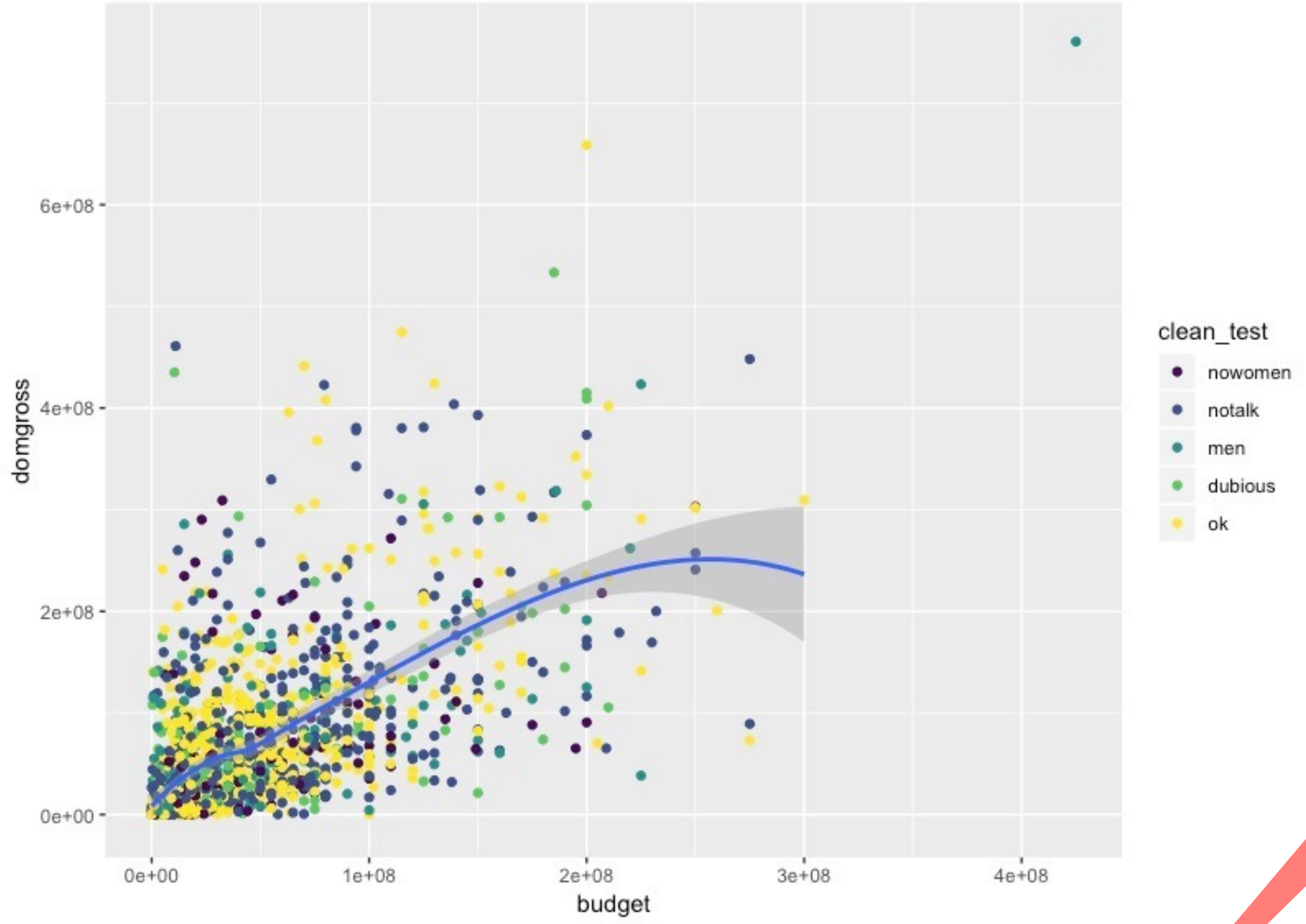
Mappings (and data) that appear in ggplot() will apply globally to every layer

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



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

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



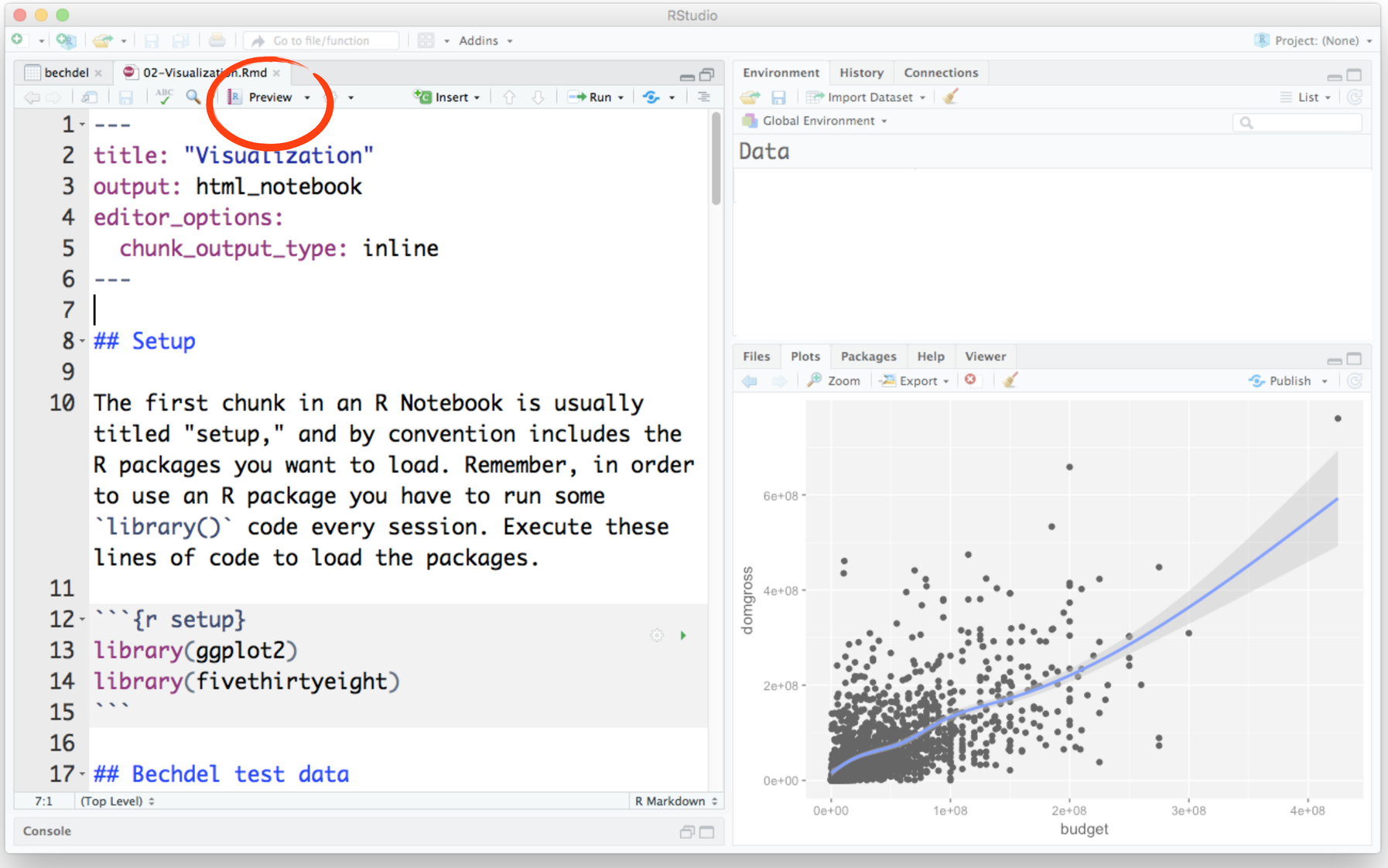
data can also be set locally or globally

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

Saving graphs

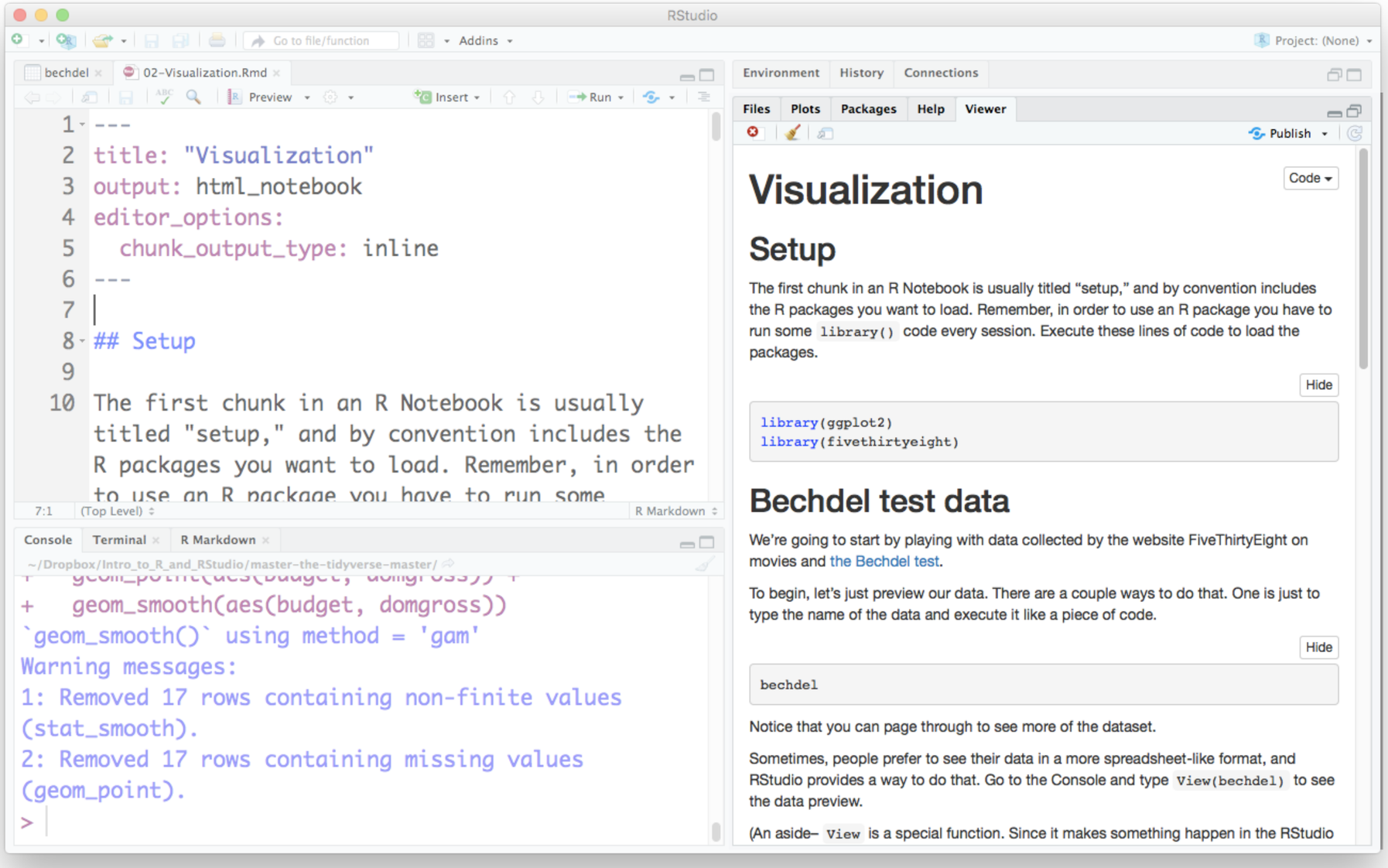
R Notebook Preview

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!

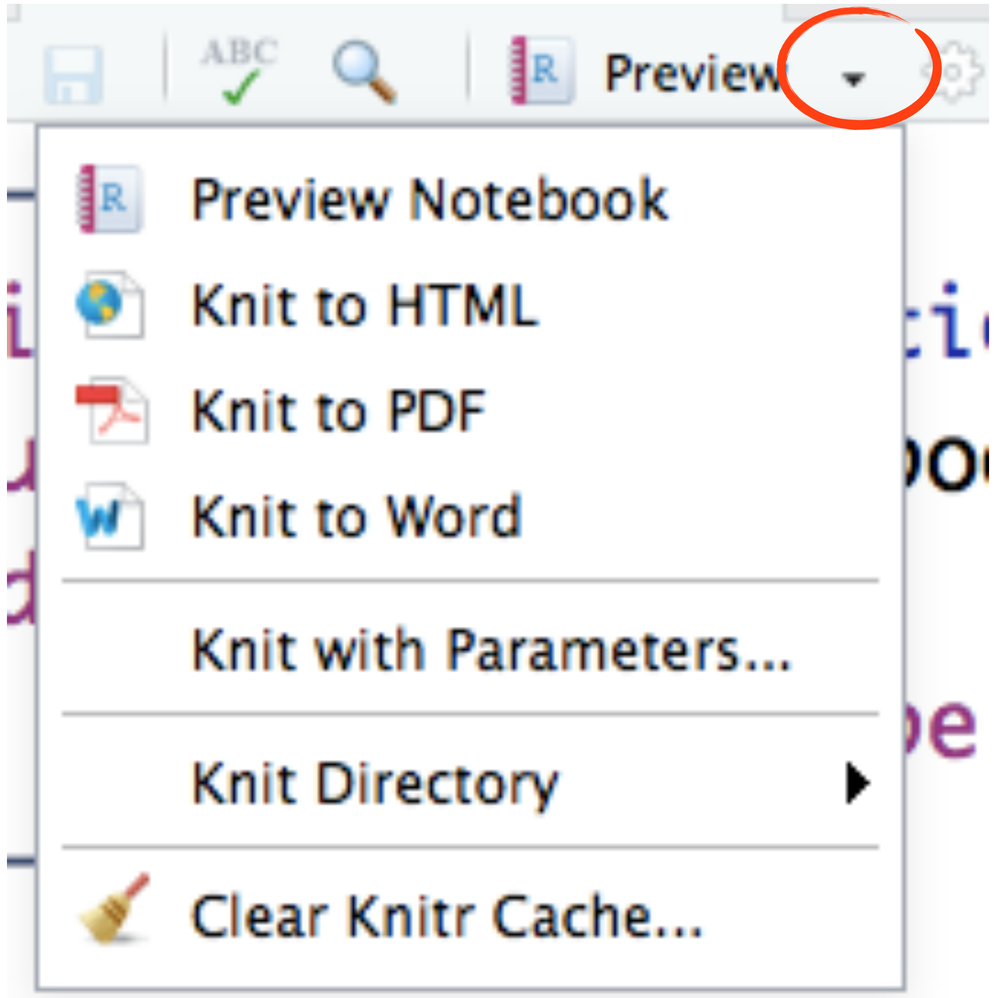


Sharing your work

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

"knitting" a document

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



Your Turn

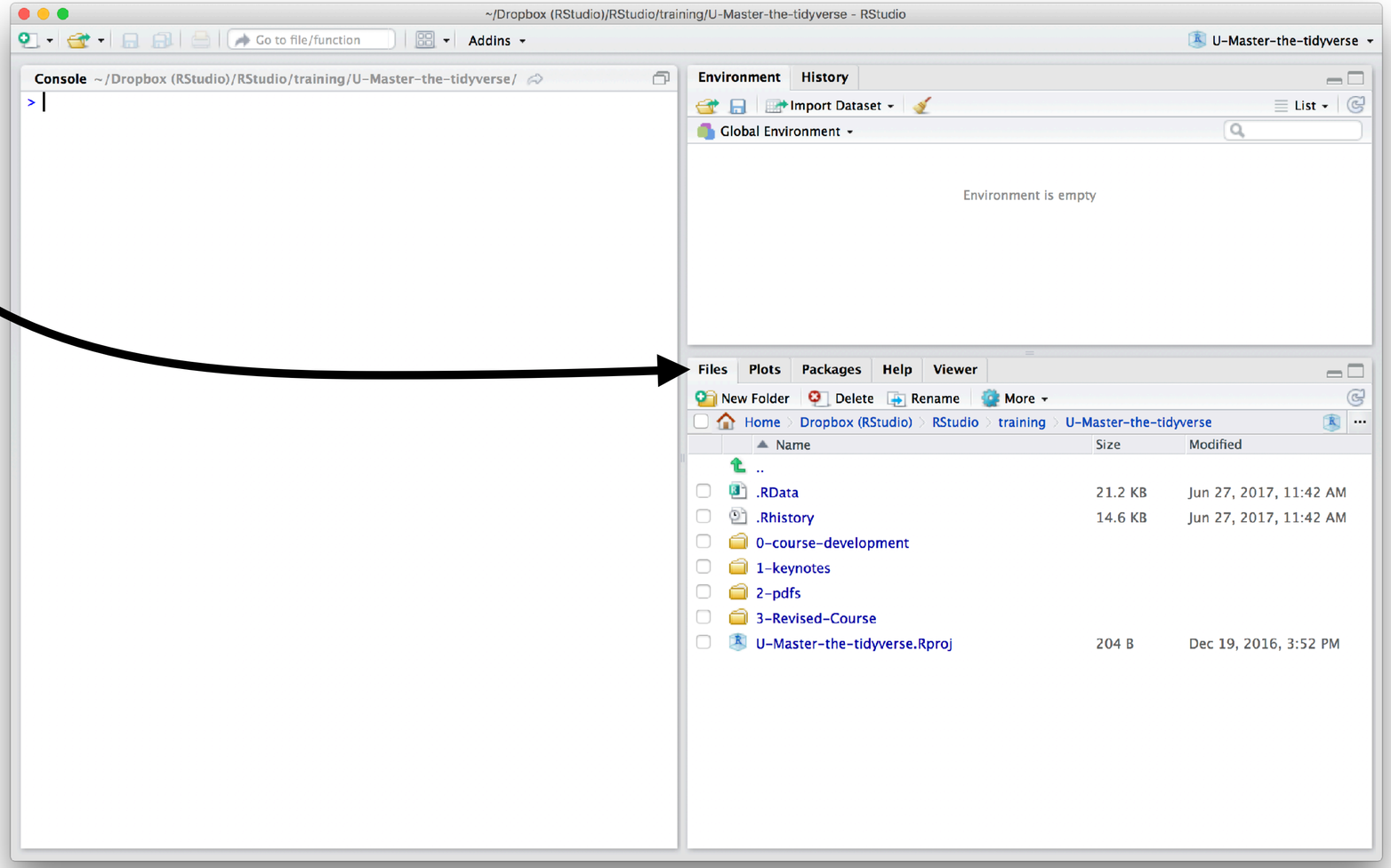
Locate the 01-Visualize.nb.html file in your Files pane. It will be located in your **working directory**

Working directory

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

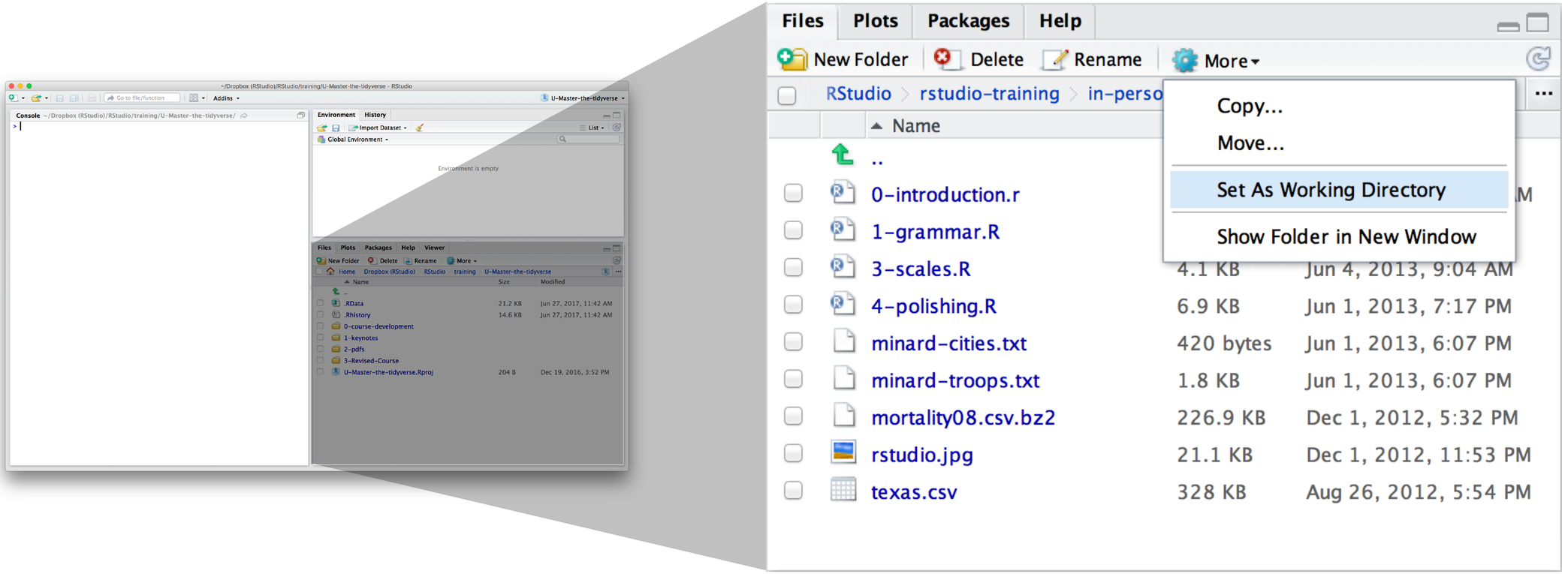
- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here

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



Changing the Working directory

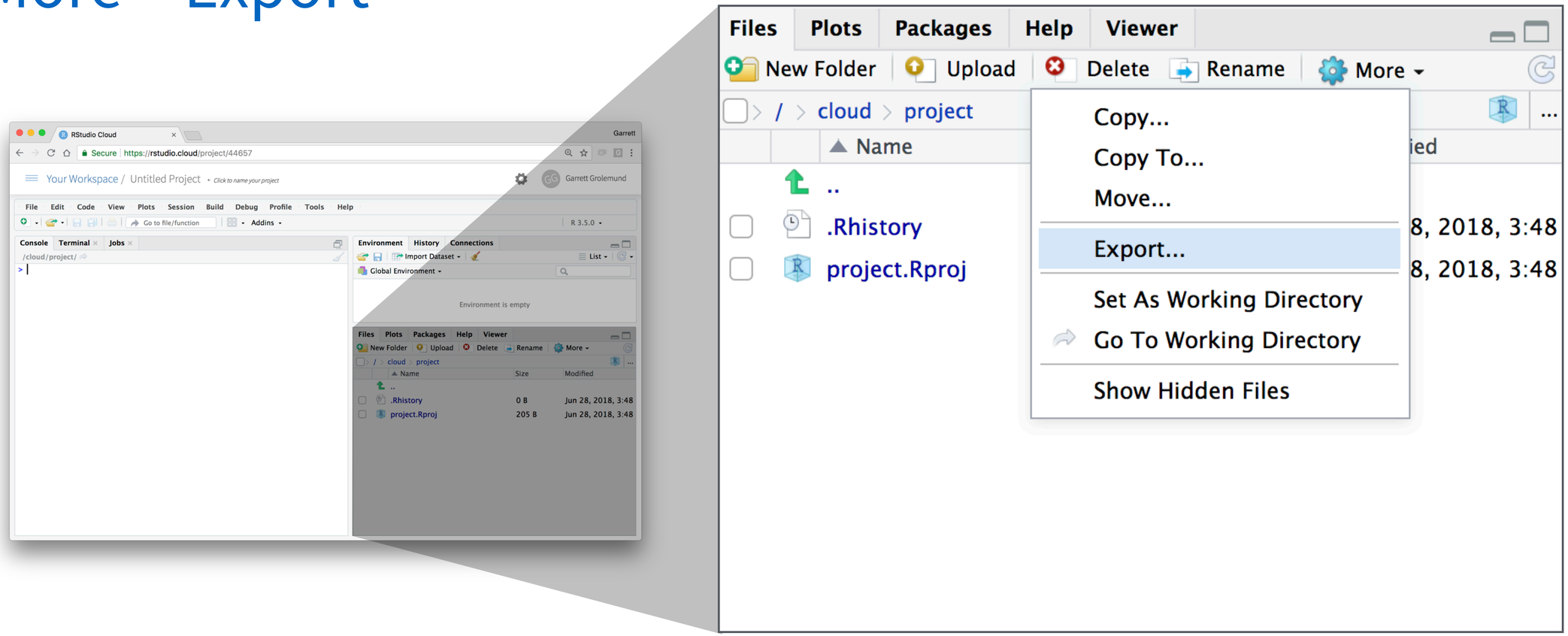
Navigate in the files pane to a new directory. Click More > Set As Working Directory



Download files

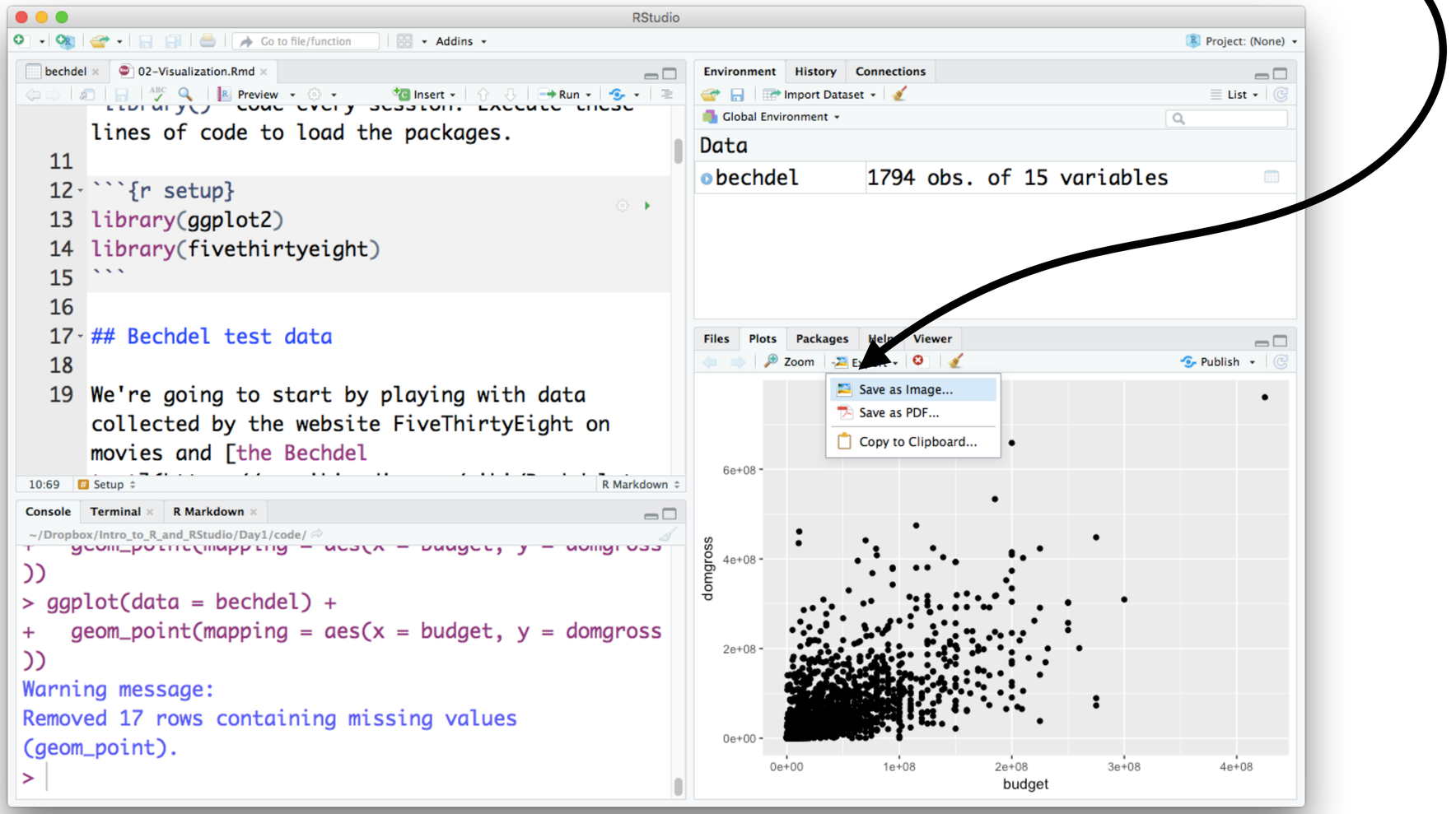
In the files pane, check next to the file(s) to download

More > Export



Manually saving plots

Save plots manually with the export menu



Saving plots

`ggsave()` saves the last plot.

Uses size on screen:

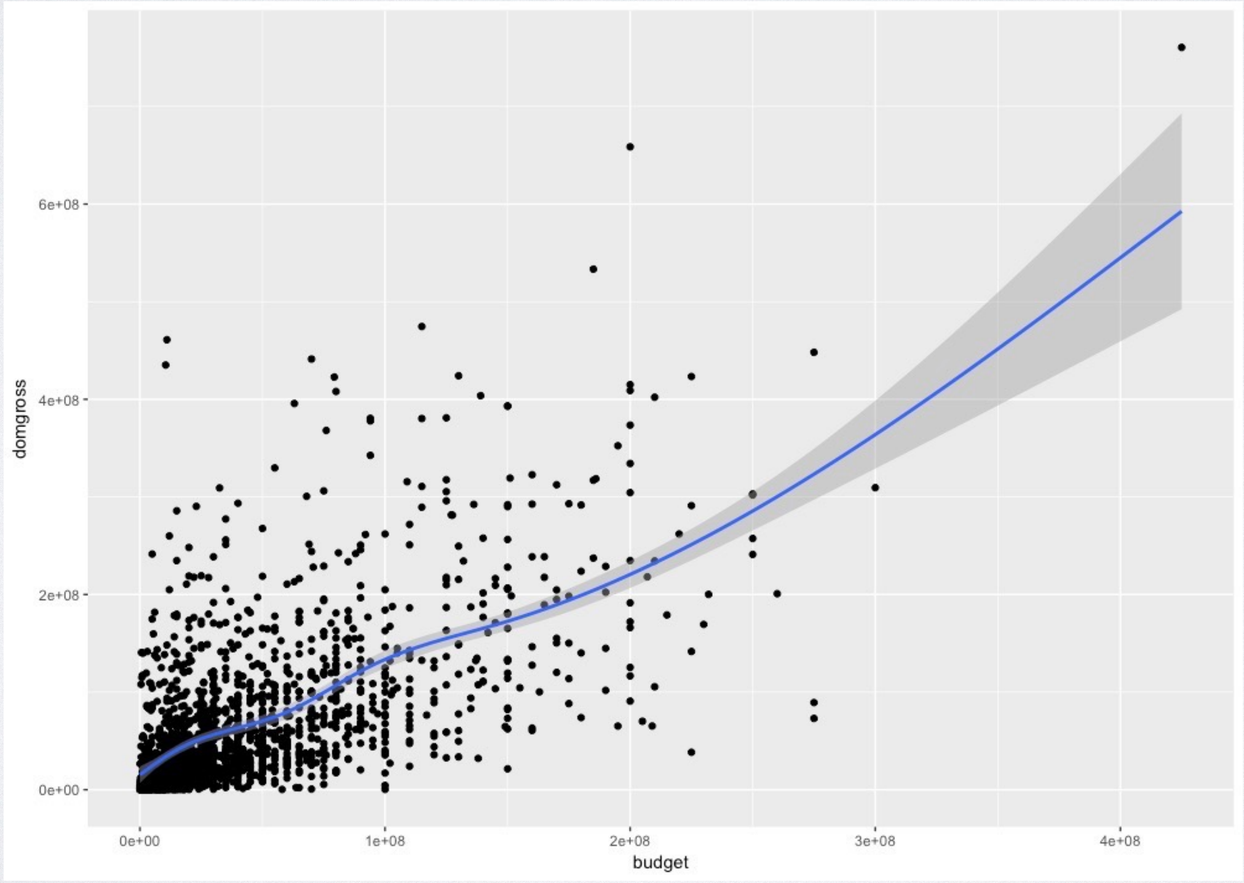
```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

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

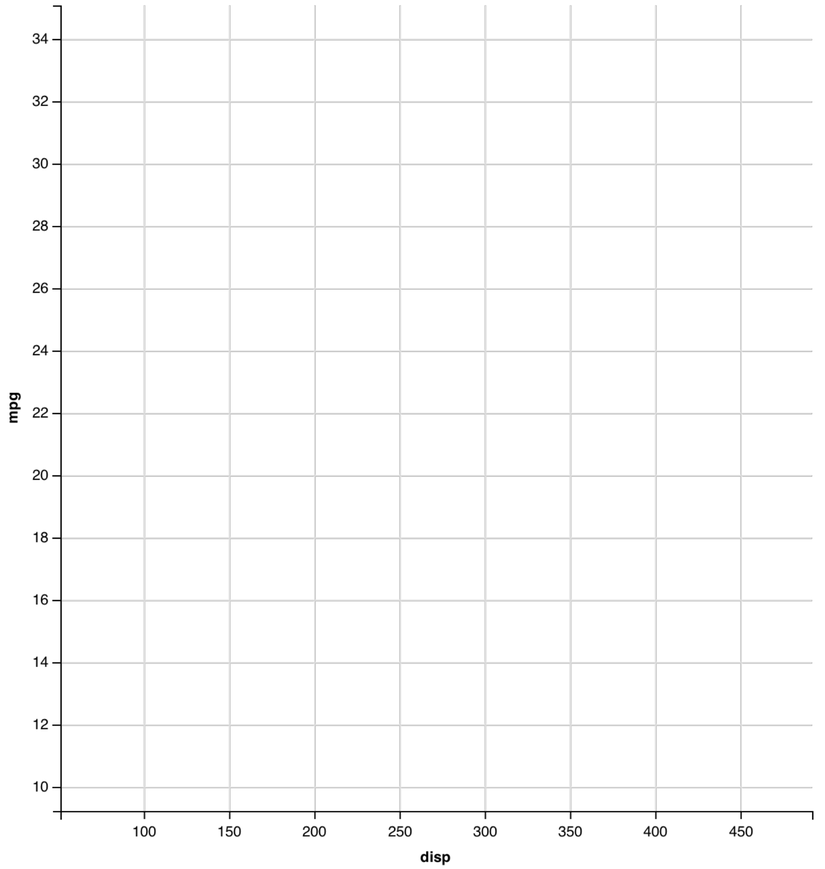
Your Turn 9

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



Grammar of Graphics

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

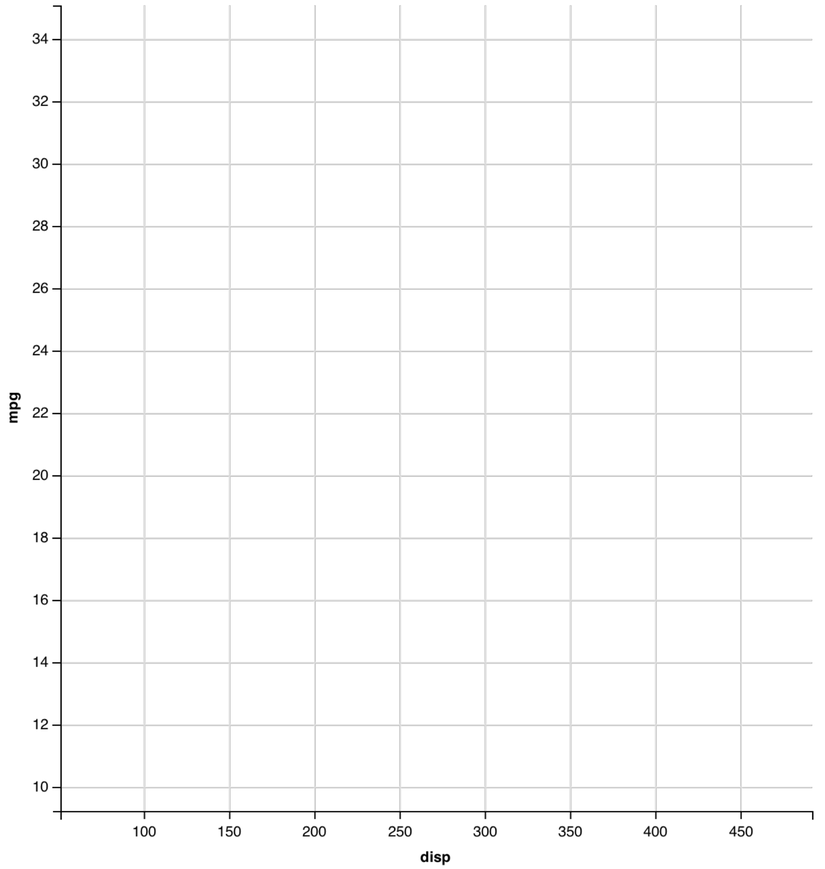


data **geom**

mappings

fill

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



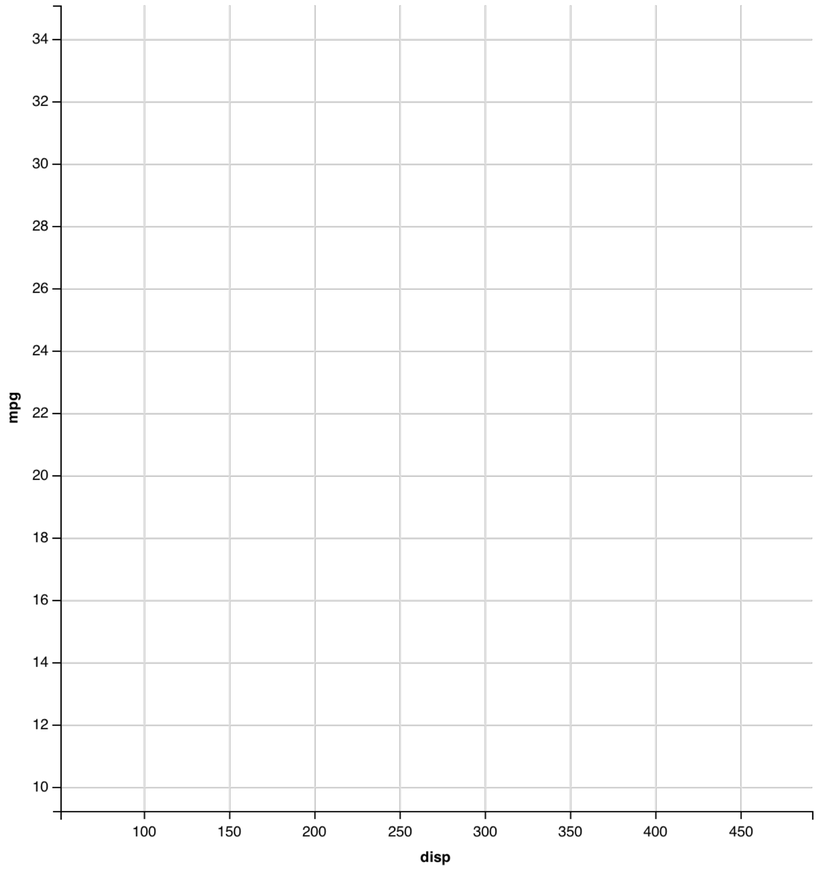
data geom

mappings

shape fill

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

data

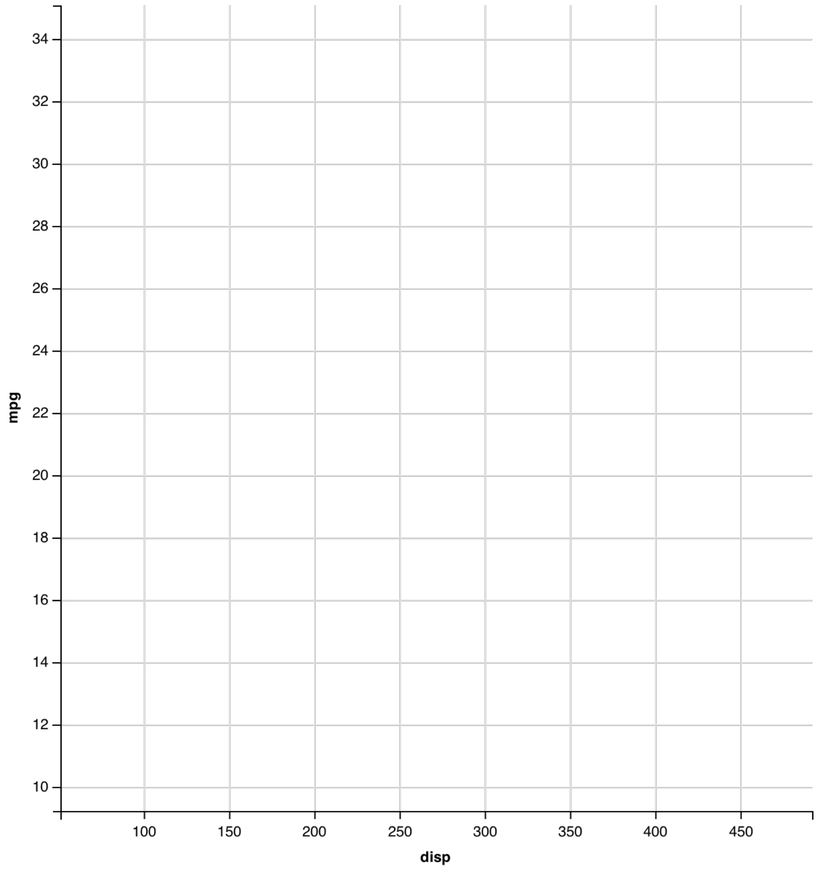


geom

mappings

shape x fill

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



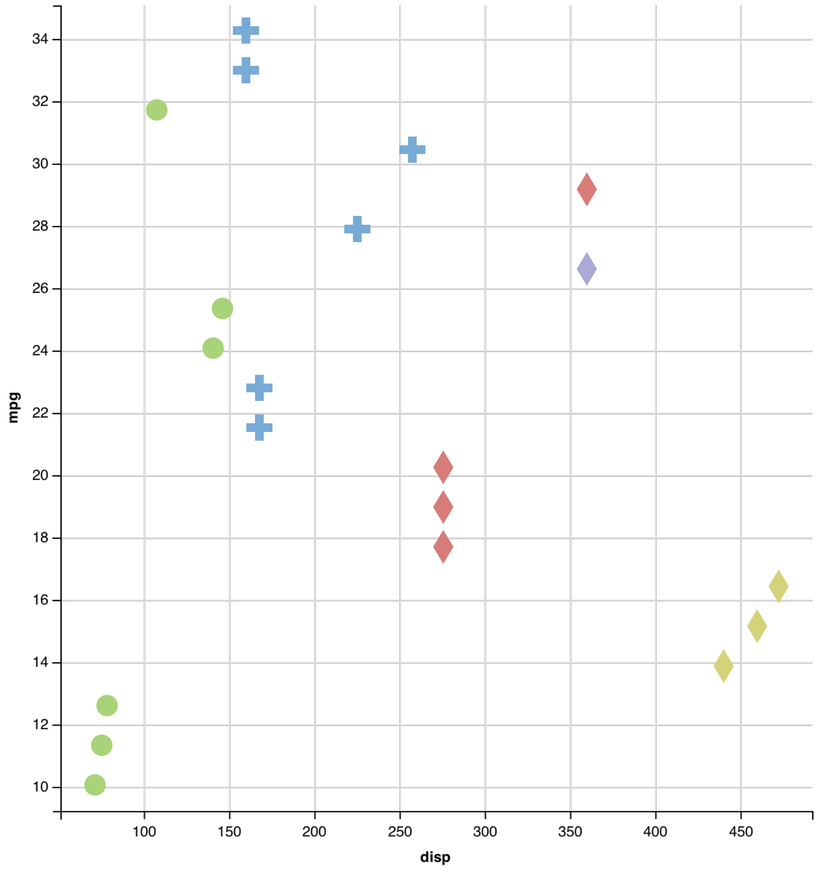
data geom

ggplot2

mappings

y shape x fill

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



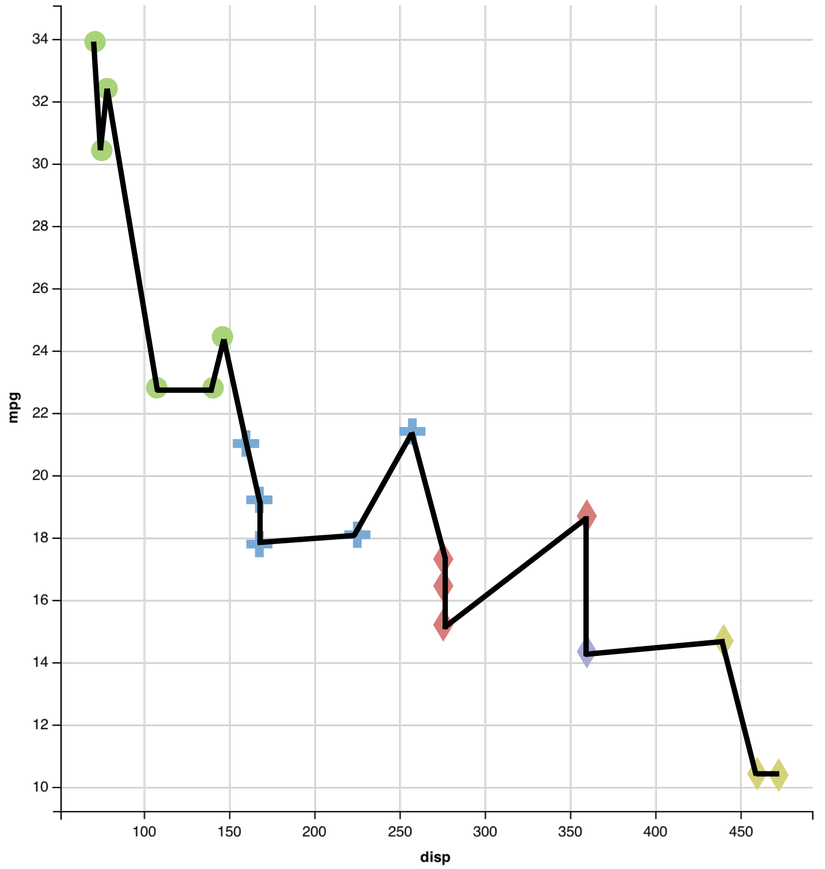
data geom

ggplot2

mappings

y shape x fill

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



data

geom

points

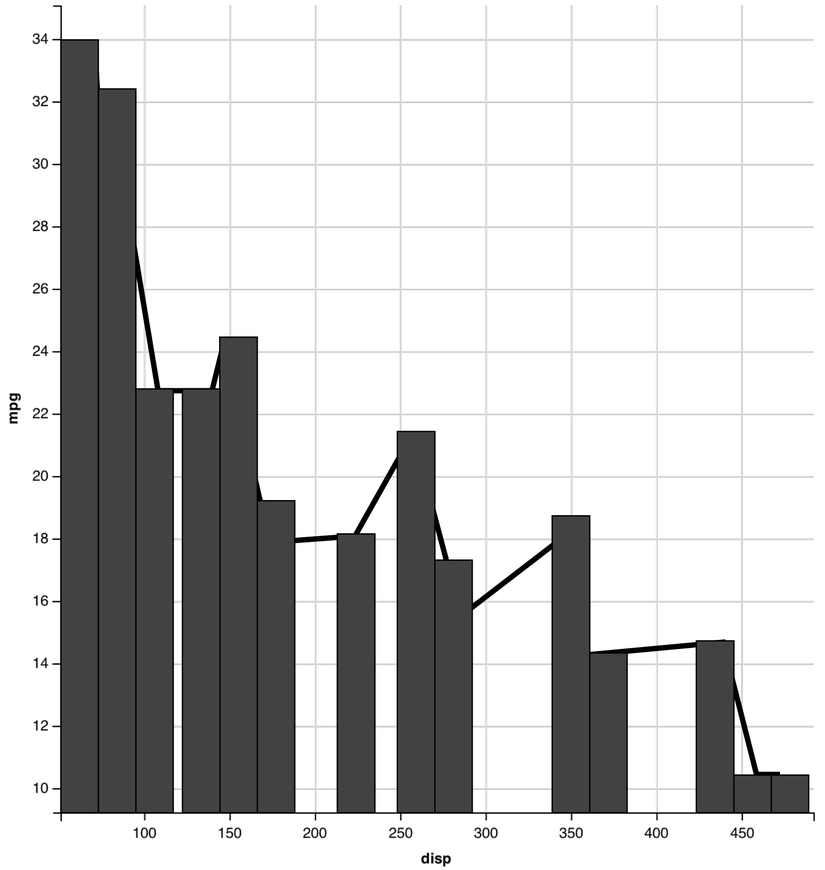
lines

ggplot2

mappings

y x

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



data **geom**

points

lines

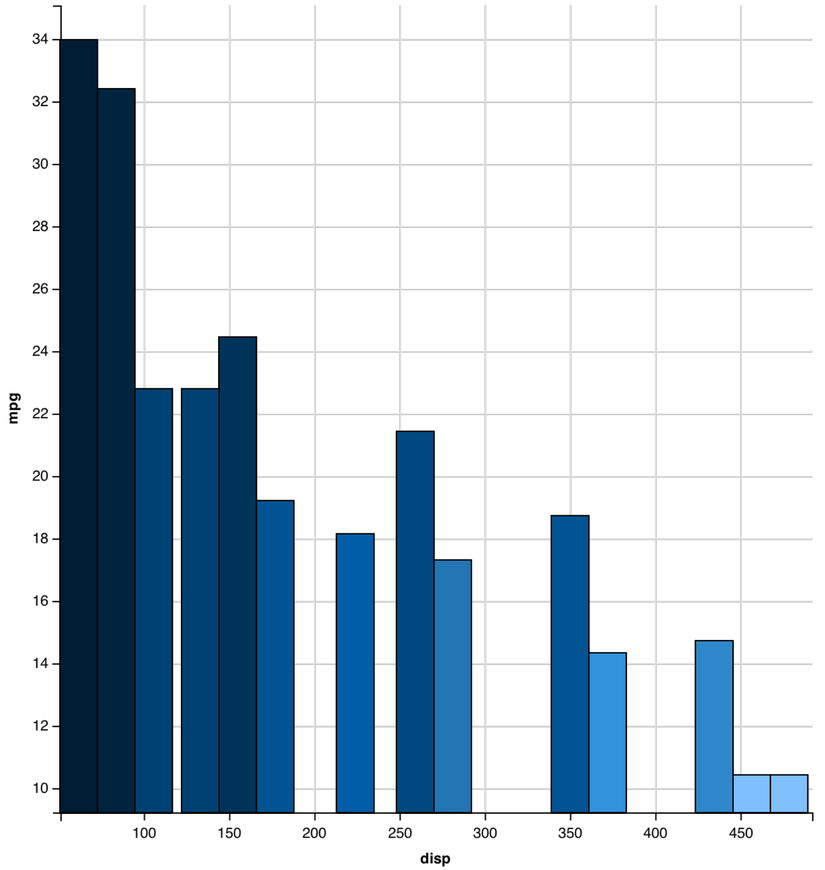
bars

ggplot2

mappings

y fill

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



data

geom

points

lines

bars

ggplot2

To make a graph

[template]

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

To make a graph

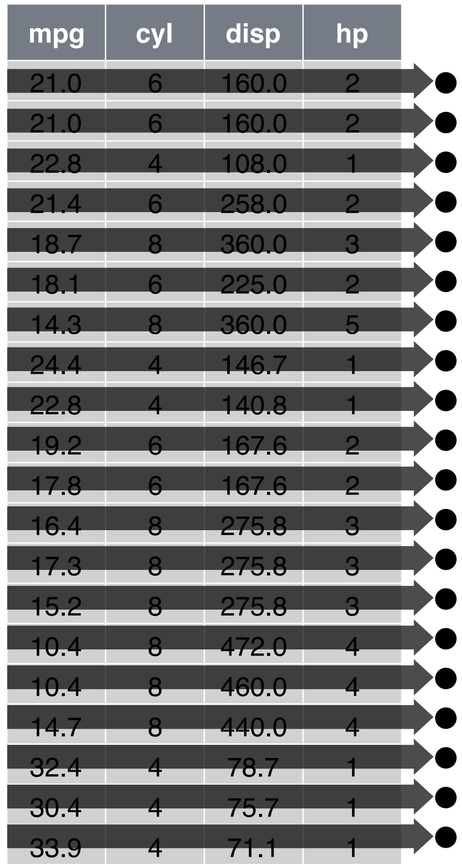
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

data

1. Pick a **data** set

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

To make a graph



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

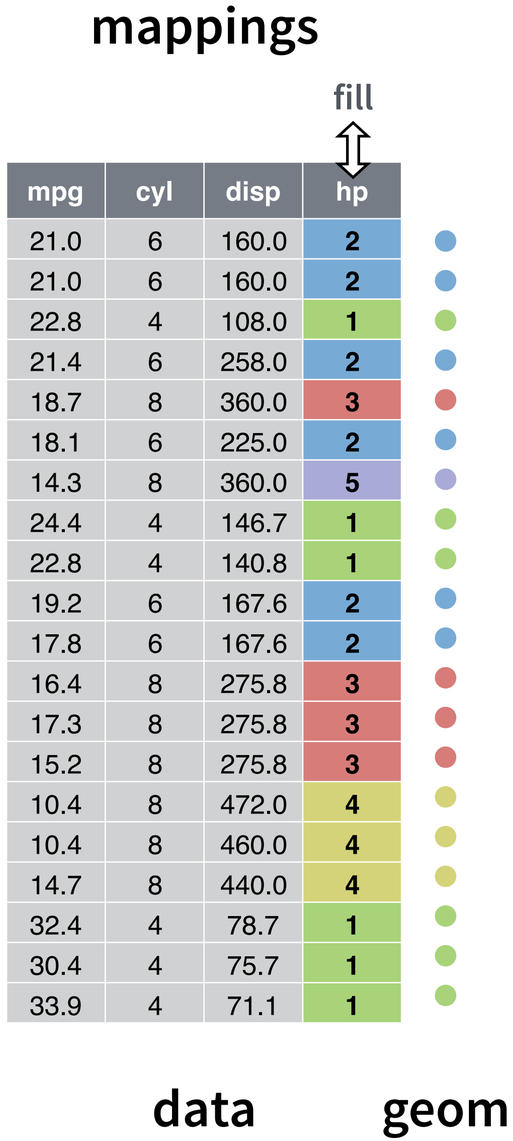
data **geom**

1. Pick a **data** set

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

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

To make a graph



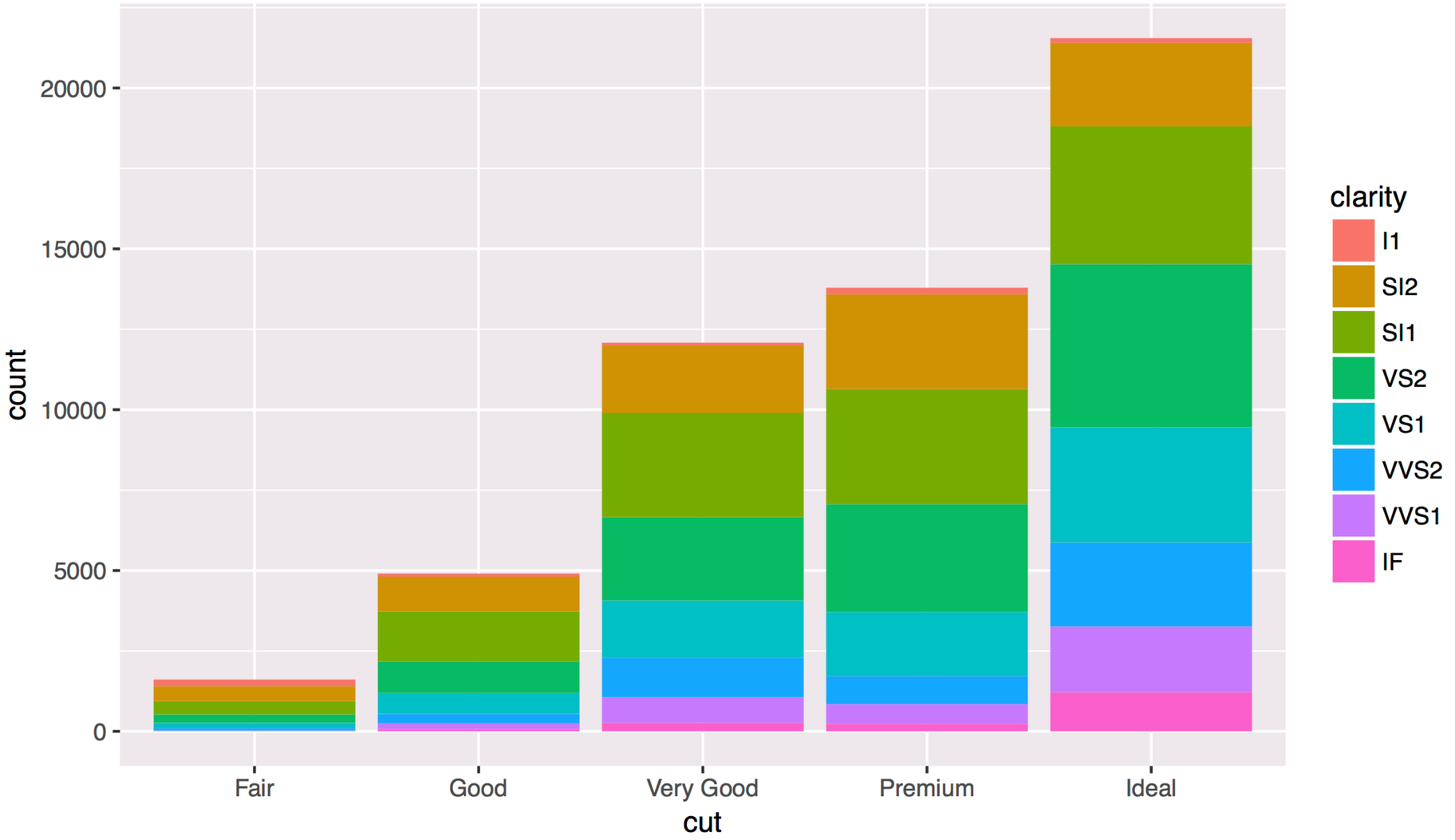
1. Pick a **data** set

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

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

3. **Map** aesthetic properties to variables

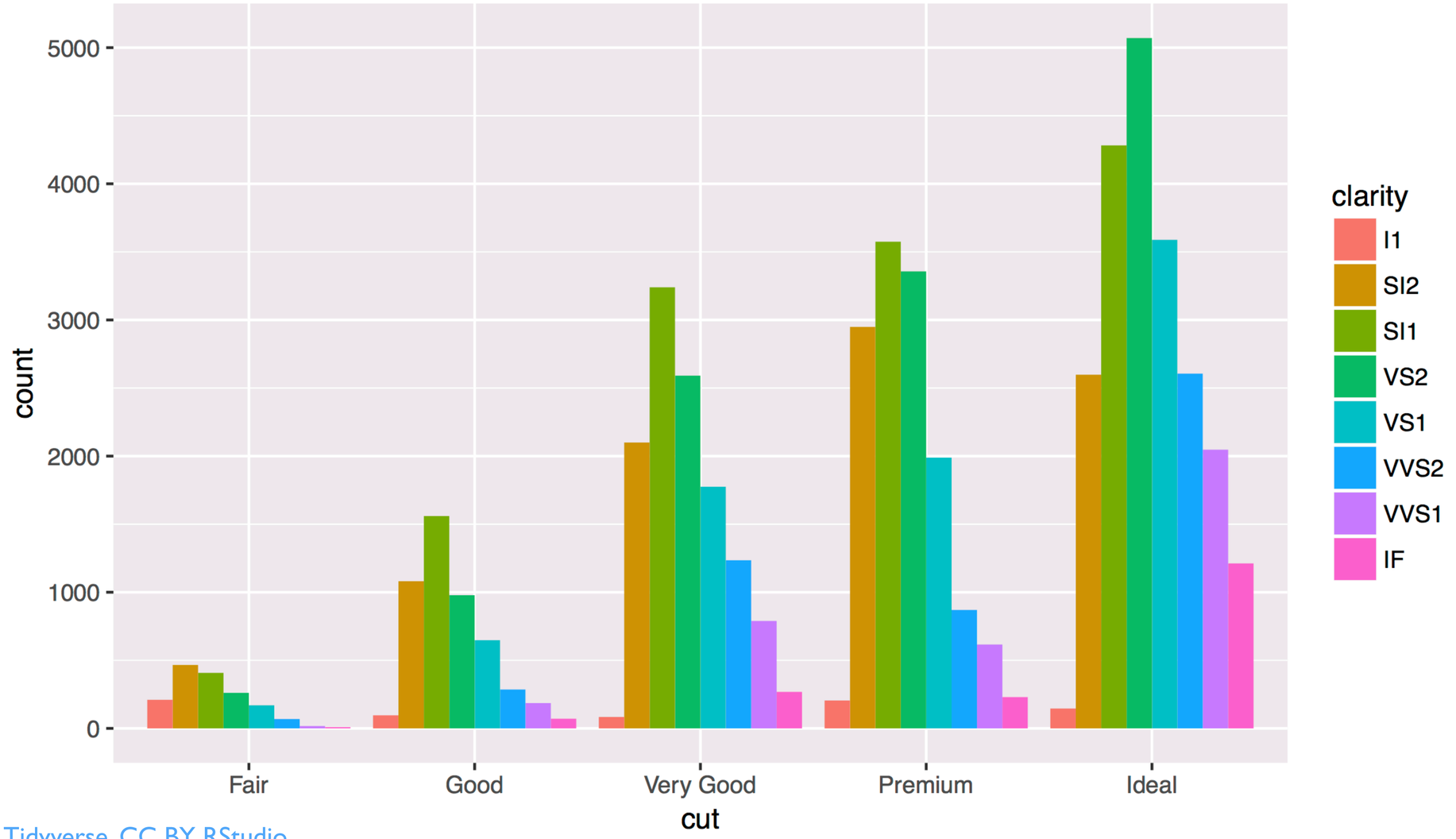
What else?



Adapted from Master the

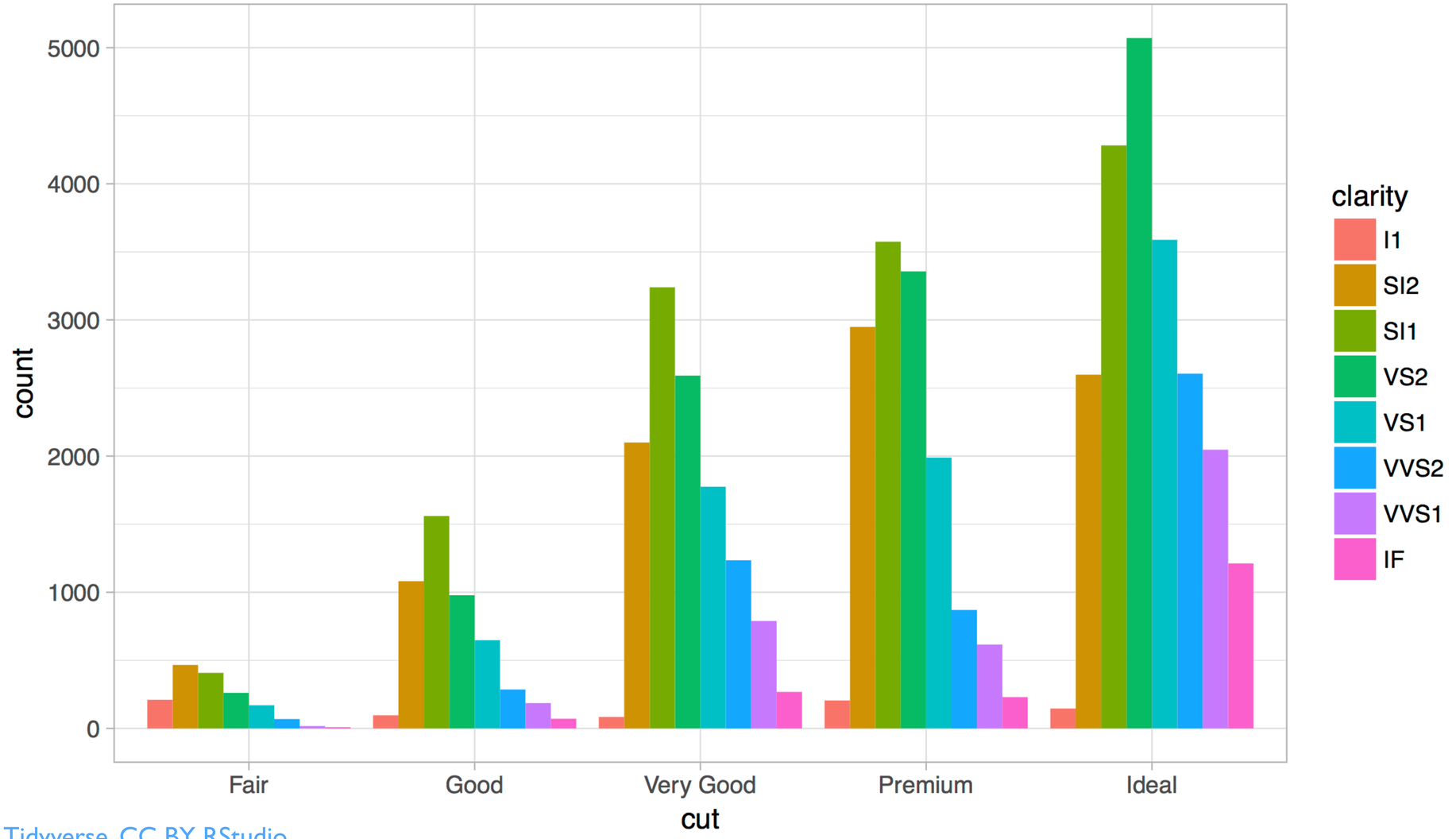
Position Adjustments

How overlapping objects are arranged



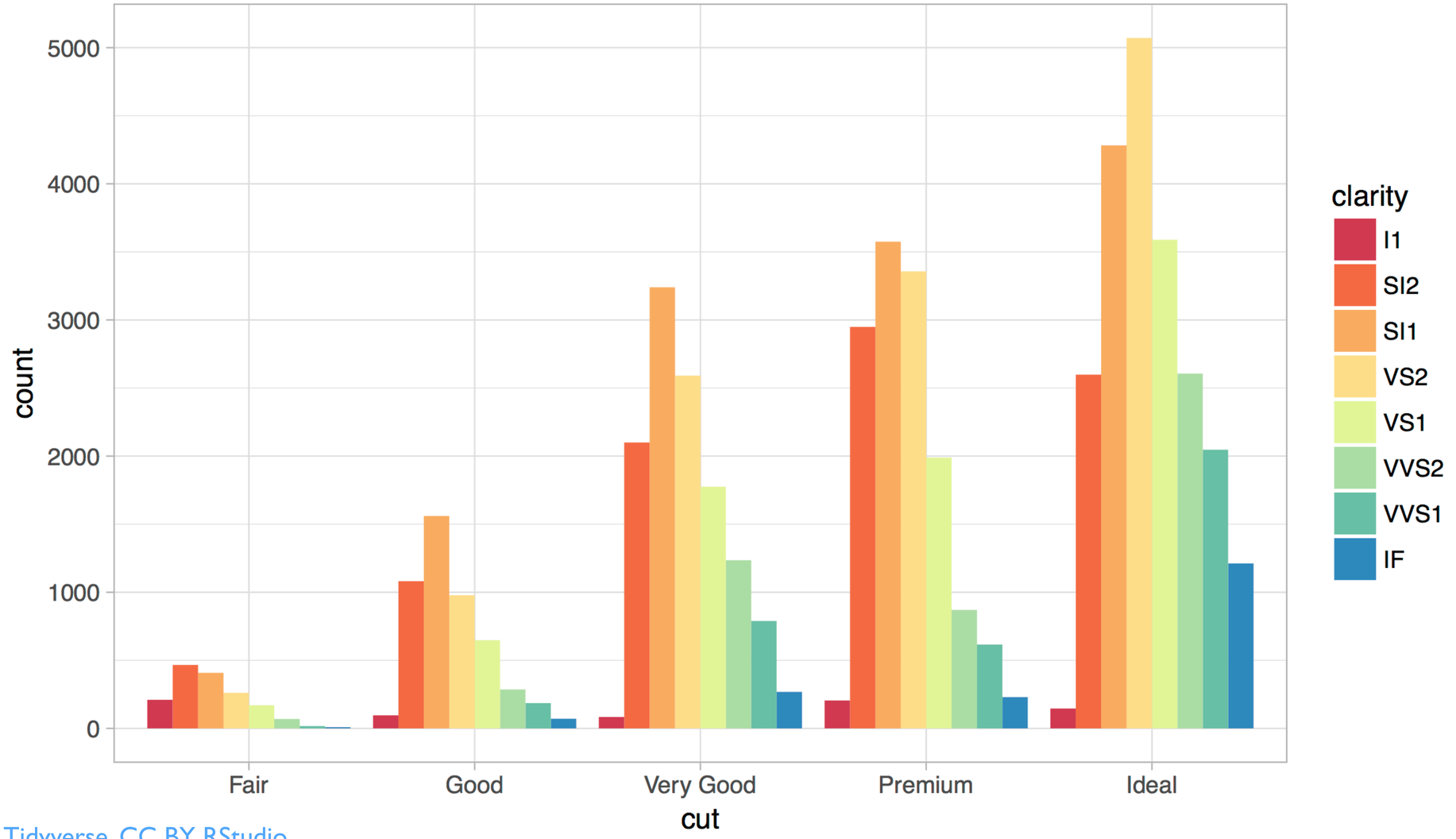
Themes

Visual appearance of non-data elements



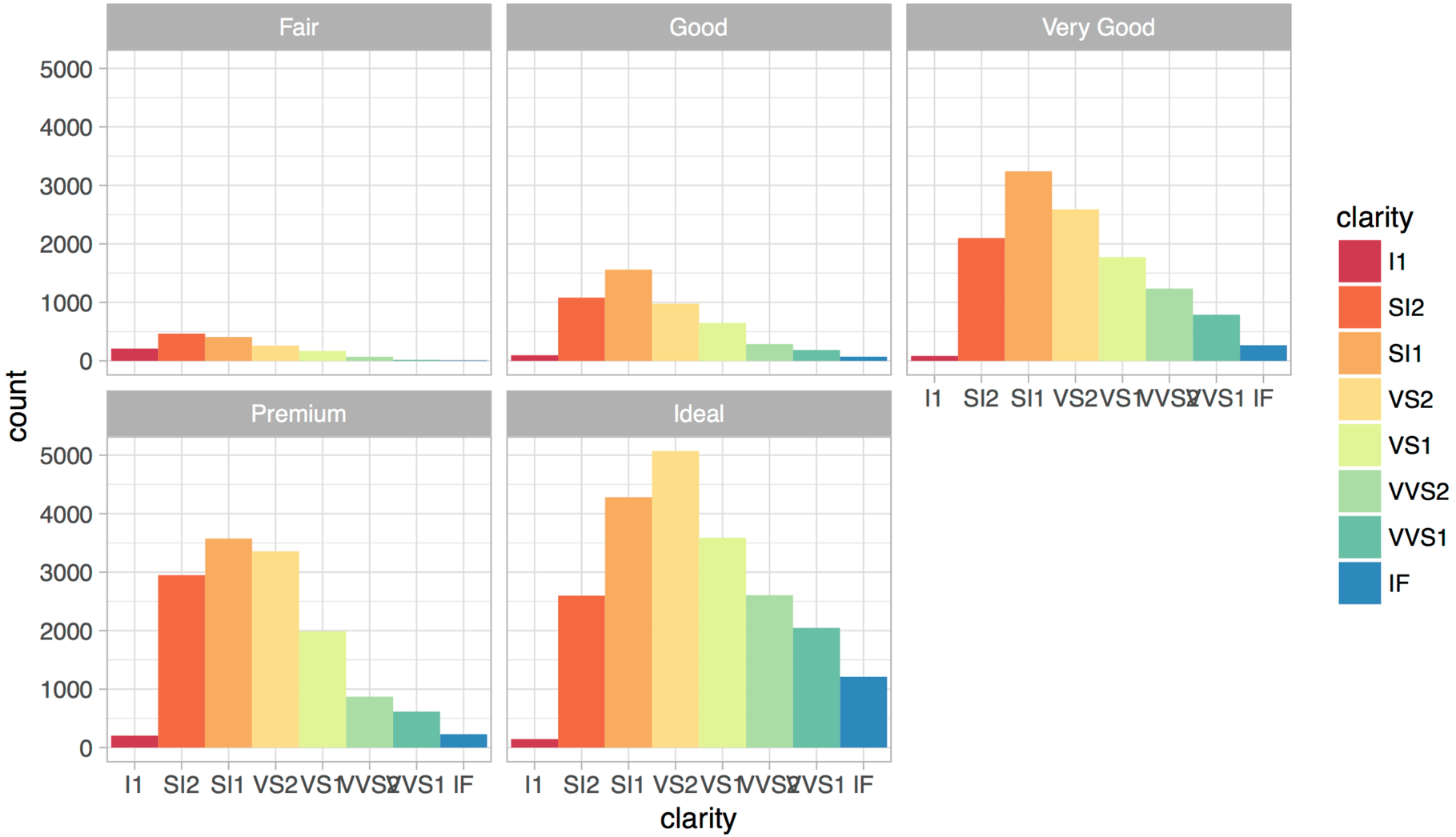
Scales

Customize color scales, other mappings

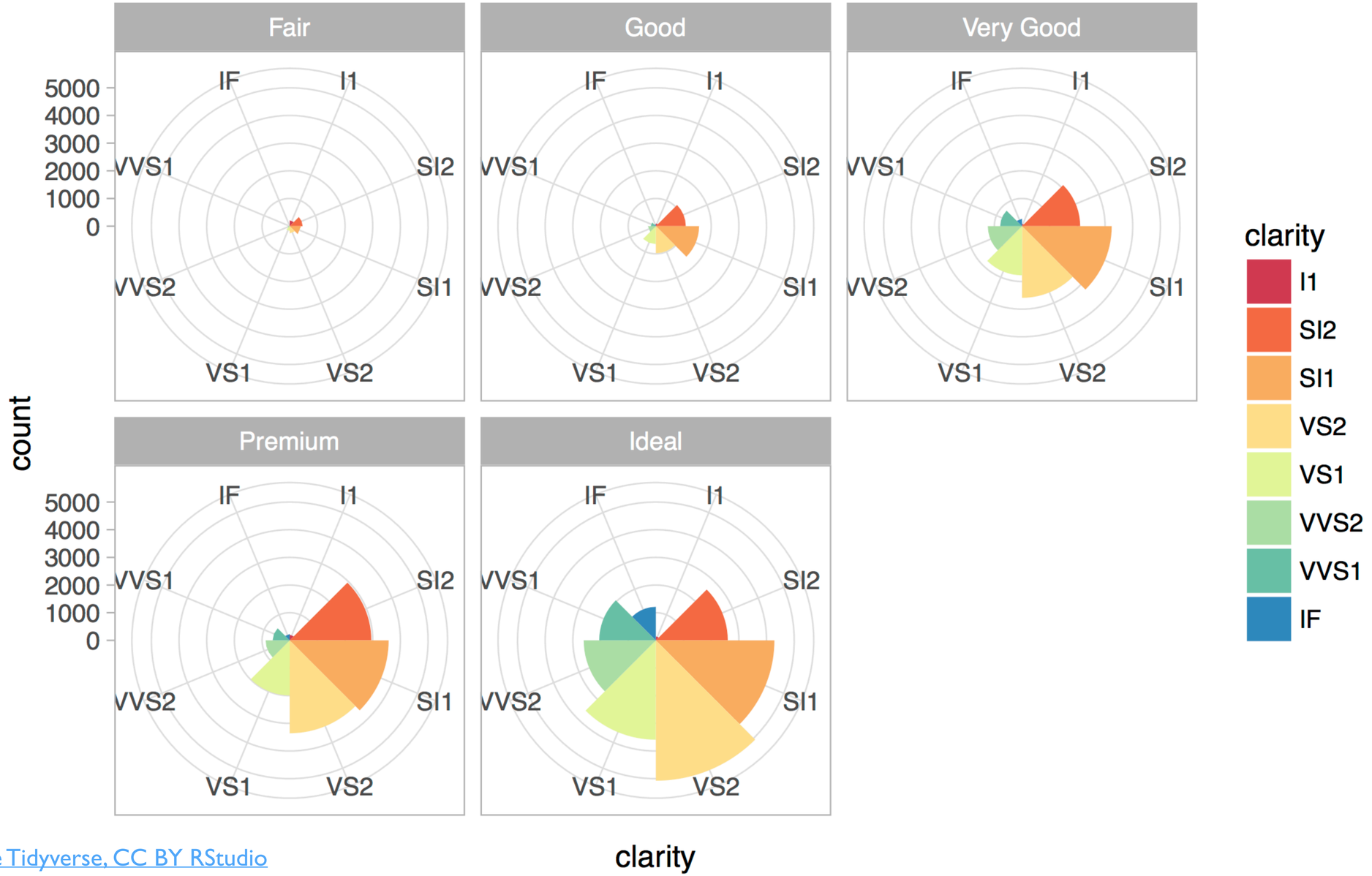


Facets

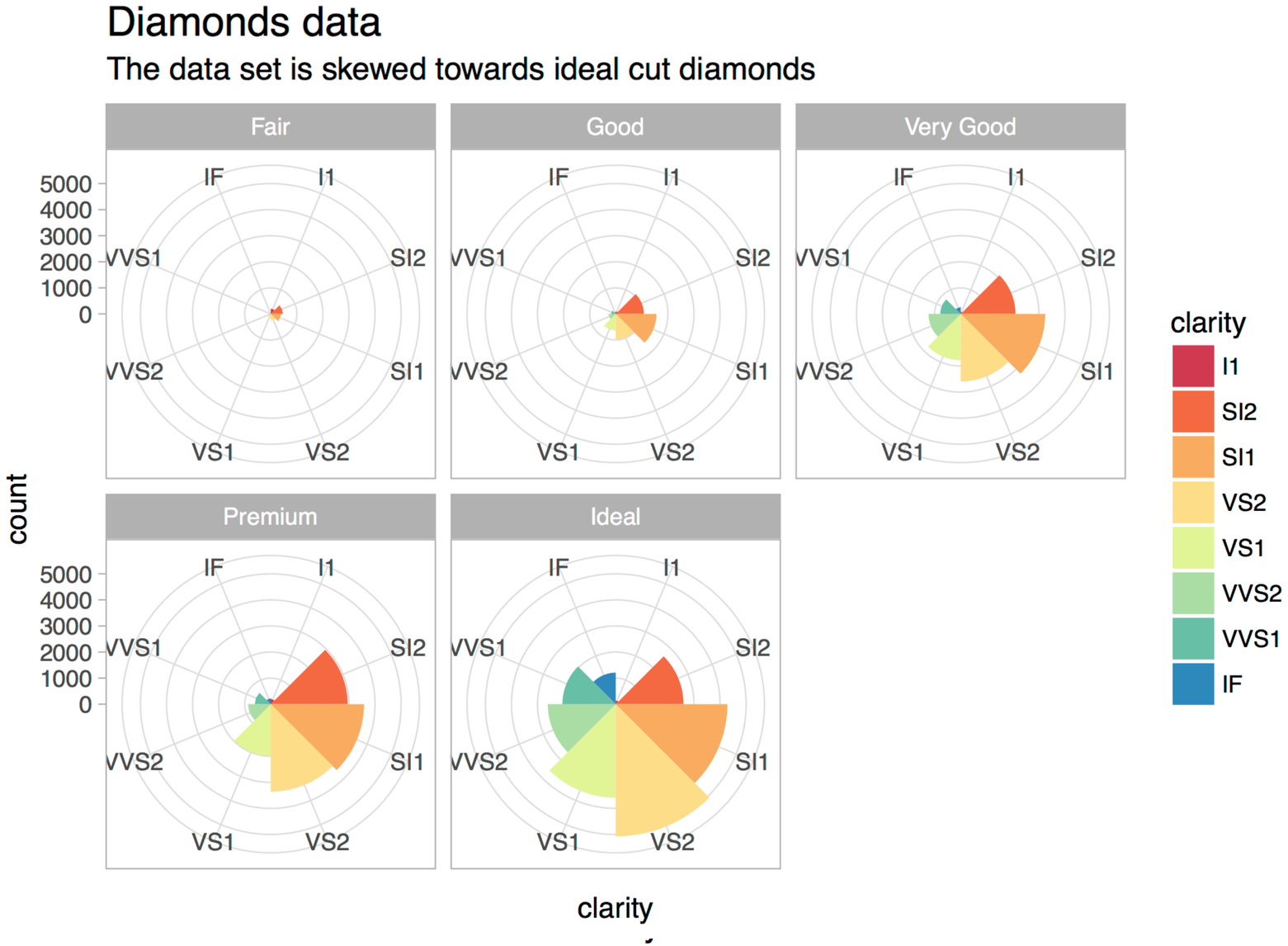
Subplots that display subsets of the data.



Coordinate systems



Titles and captions



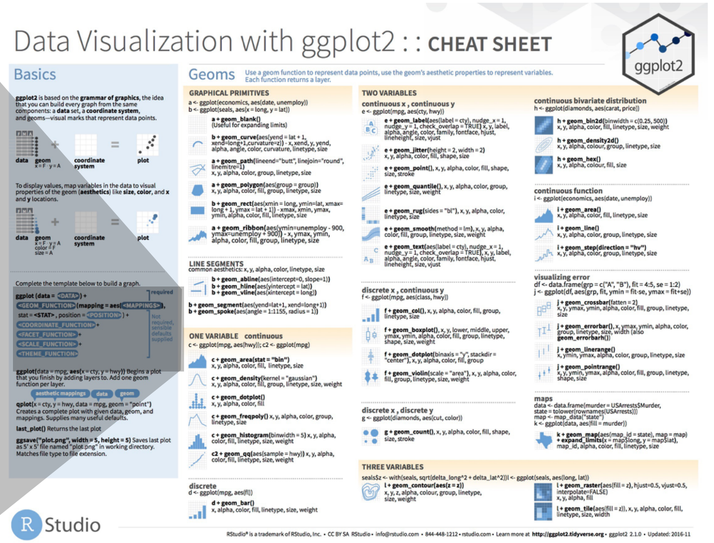
Data by Hadley Wickham

A ggplot2 template

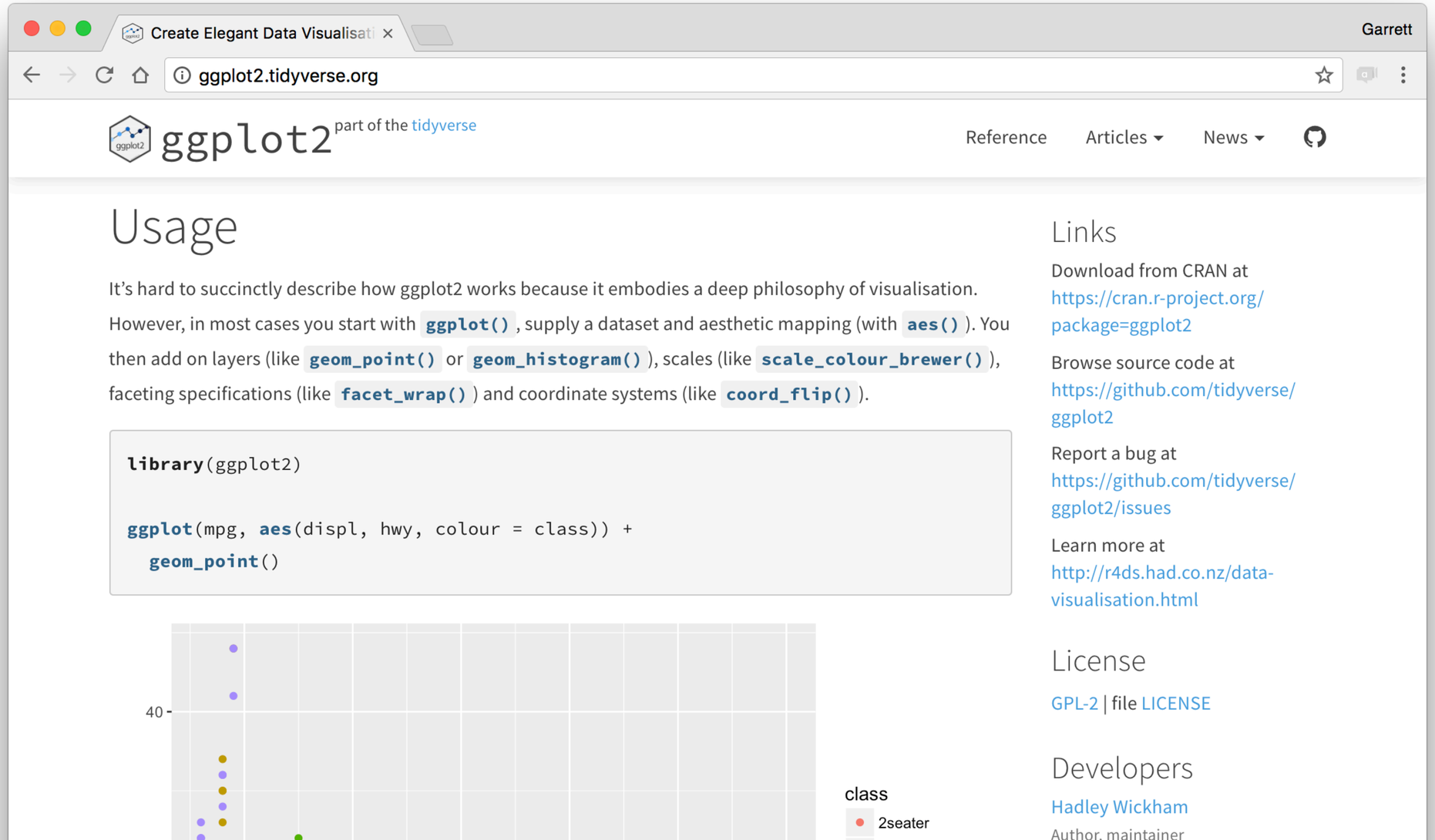
Make any plot by filling in the parameters of this template

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   ] required
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION>) +                  ] Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```



ggplot2.tidyverse.org



Visualize Data with