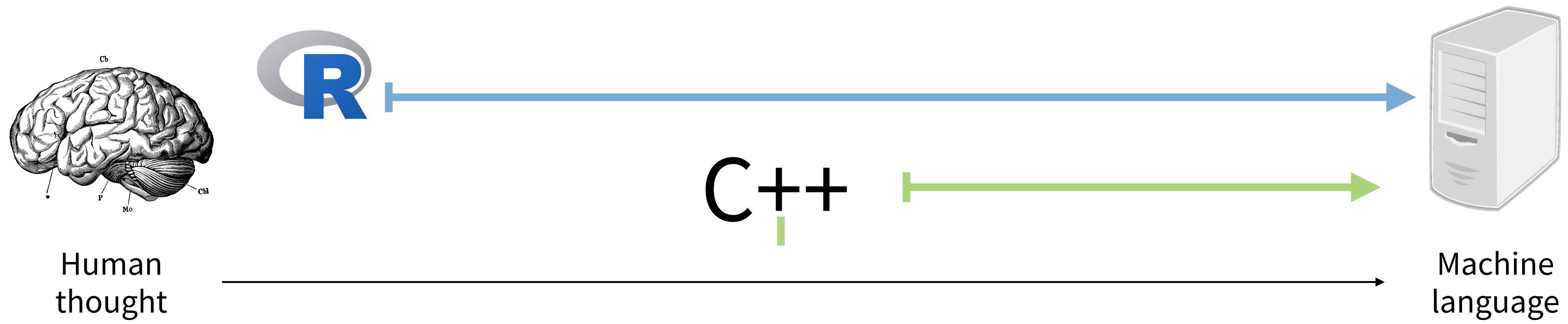


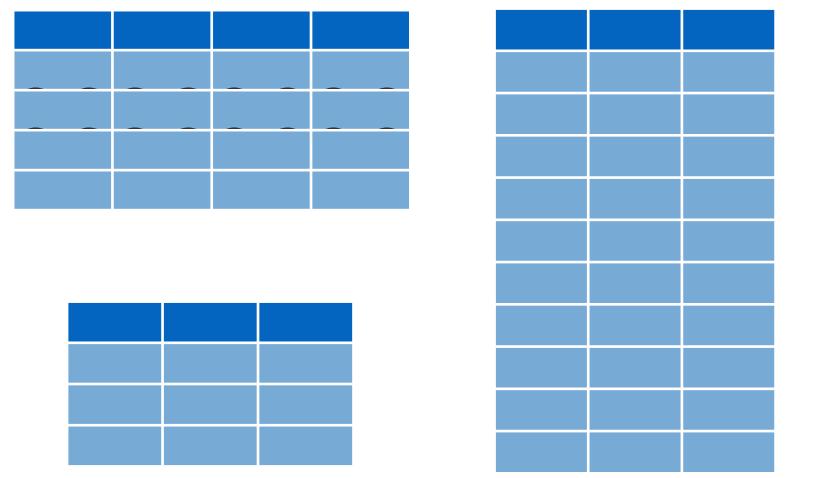
ggplot2

Modified from “Data Science in the tidyverse” materials. Major contributions by Garrett Grolemund, Amelia McNamara, Charlotte Wickham, and Hadley Wickham. Licensed under a Creative Commons Attribution 4.0 International License.

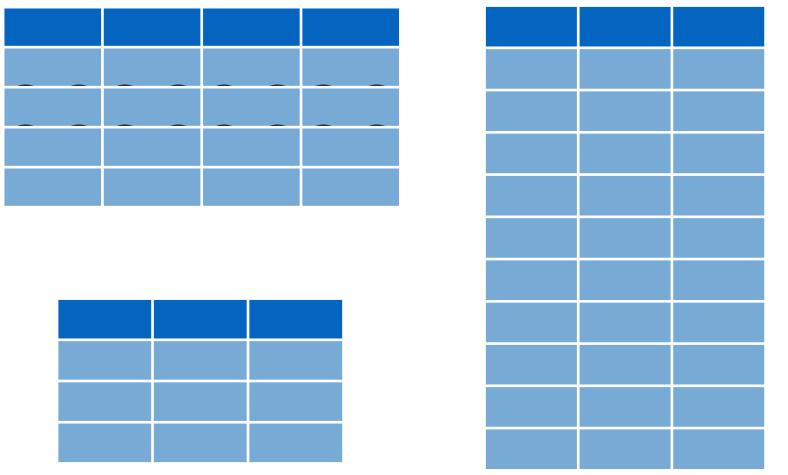


R - A computer language for scientists

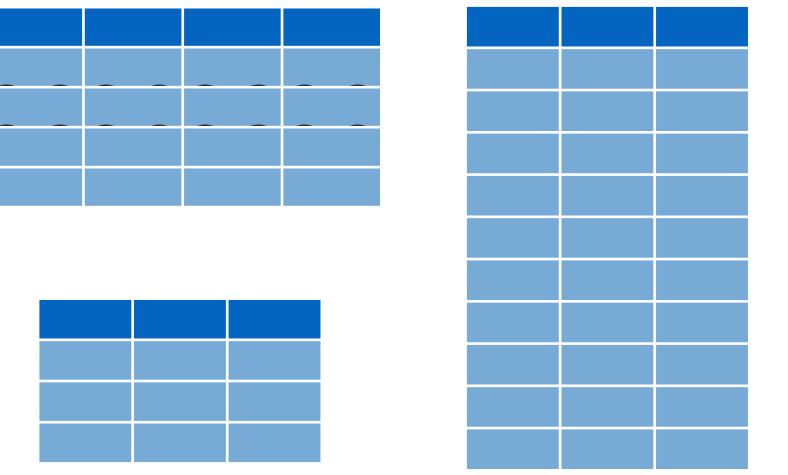
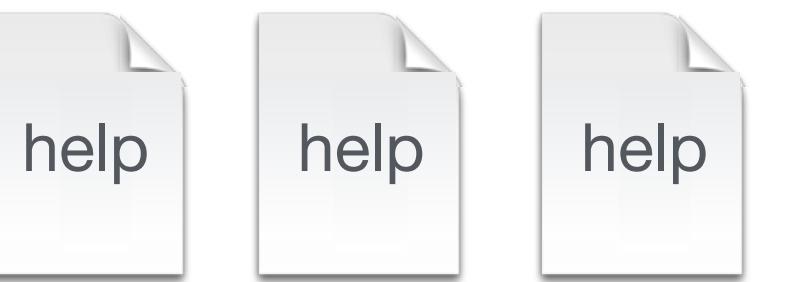




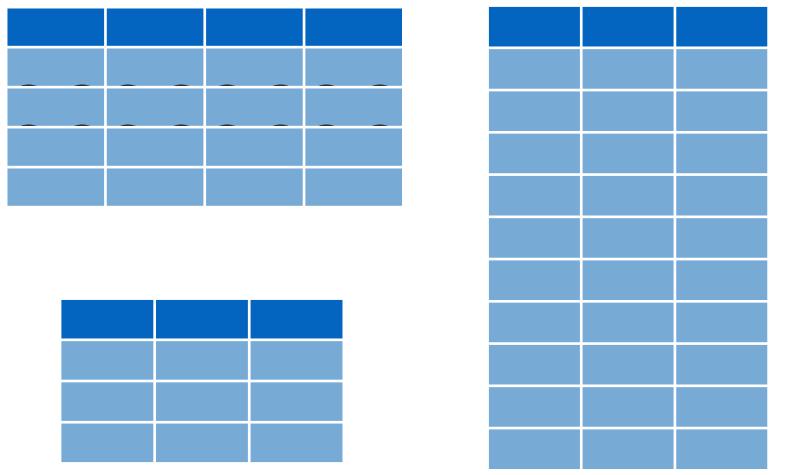
function1()
function2()
function3()
function4()



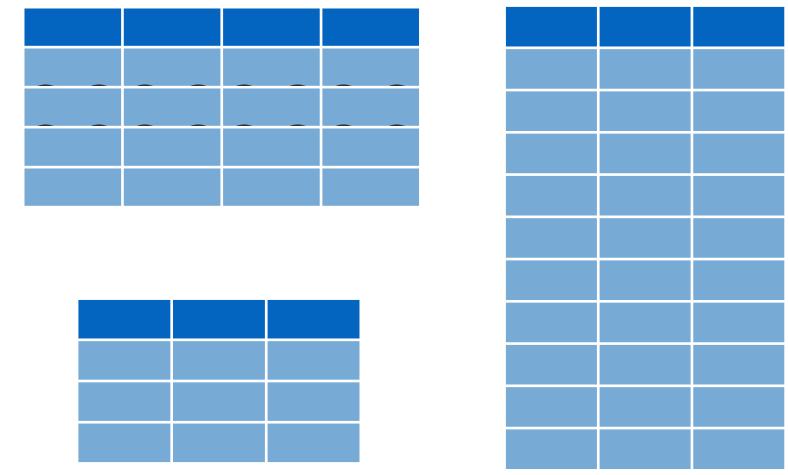
function1()
function2()
function3()
function4()



function5()
function6()
function7()
function8()

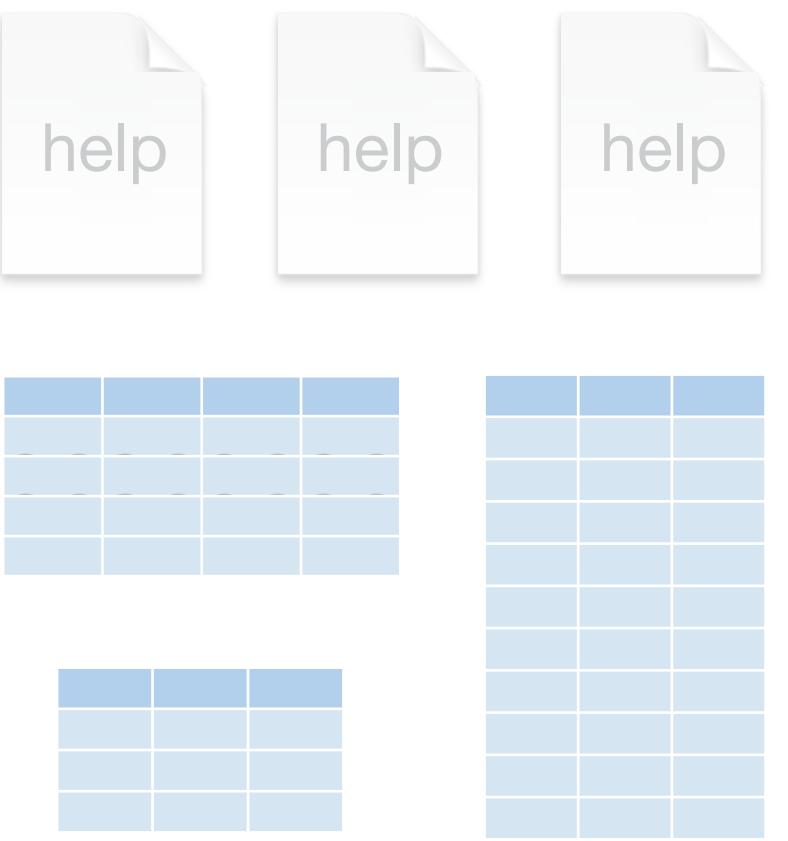


function9()
functionA()
functionB()
functionC()

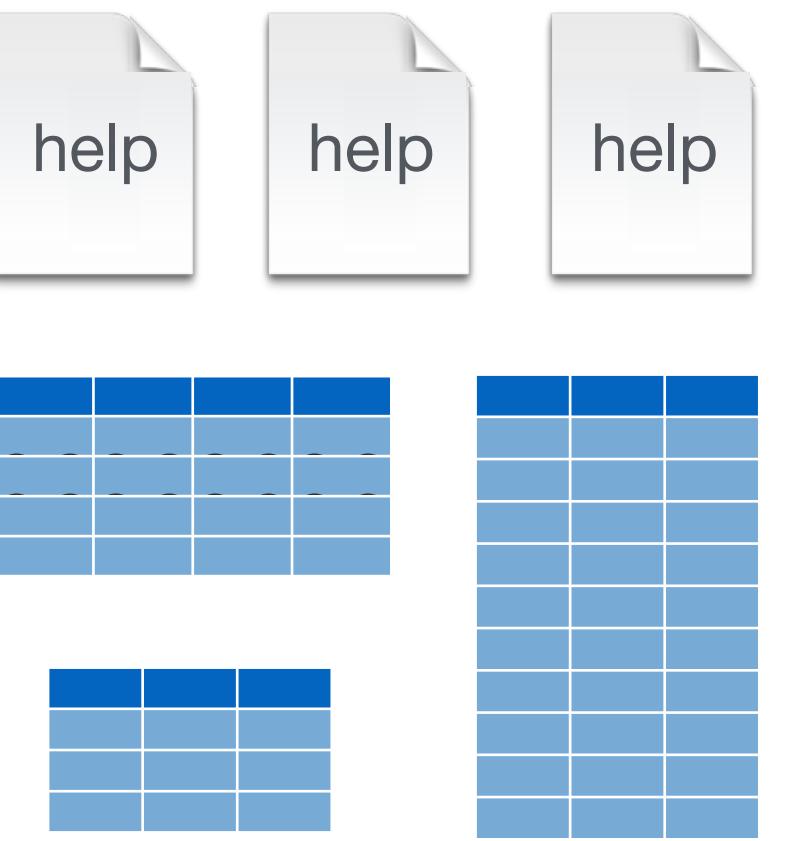


functionD()
functionE()
functionF()
functionG()

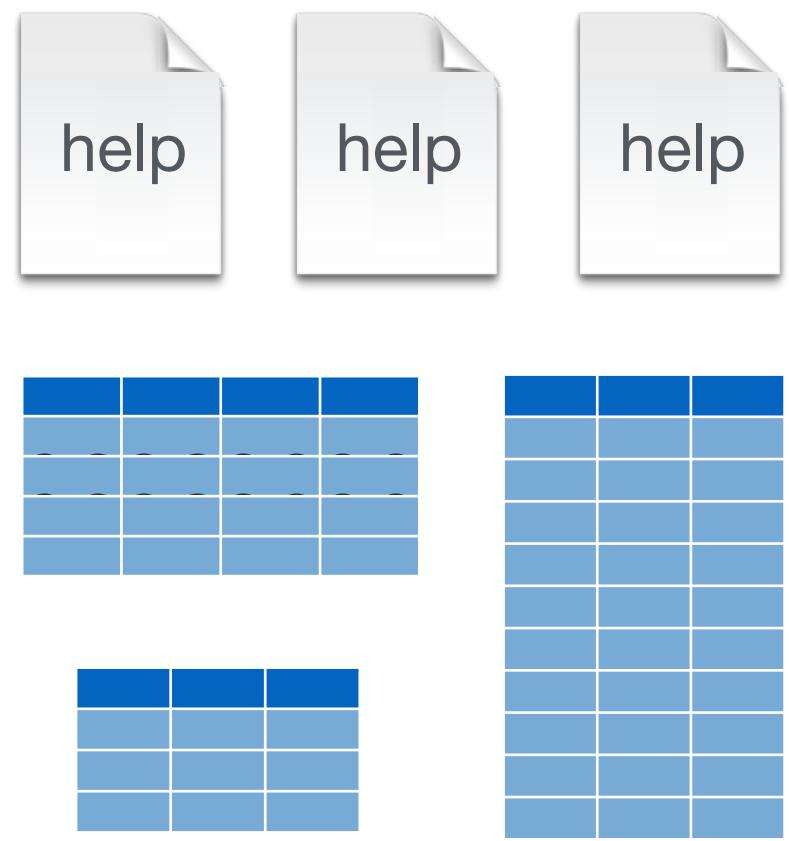
Base R



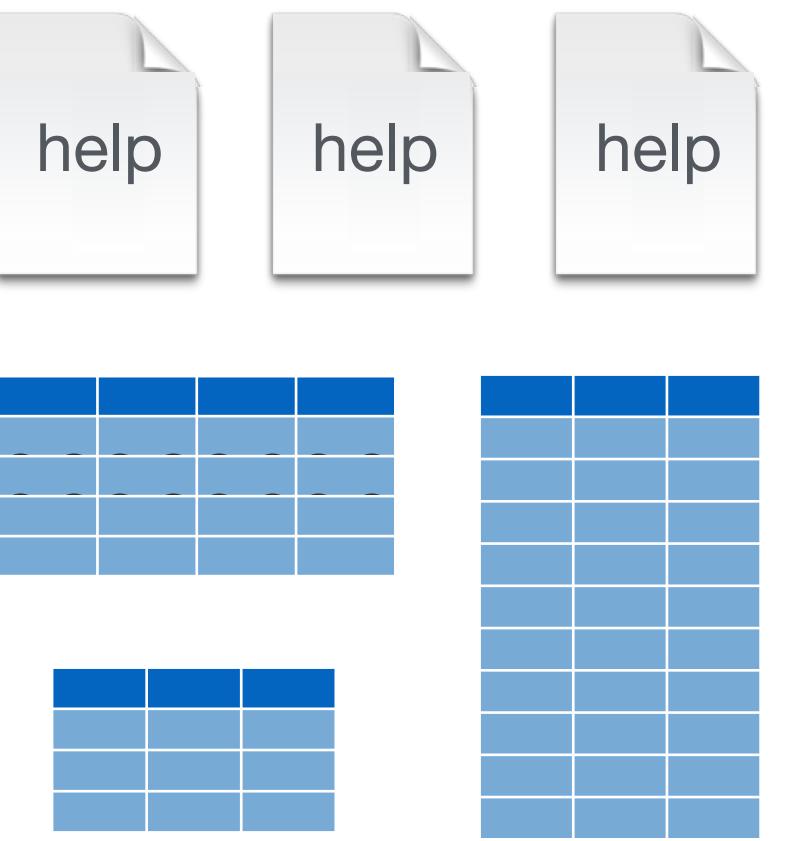
function1()
function2()
function3()
function4()



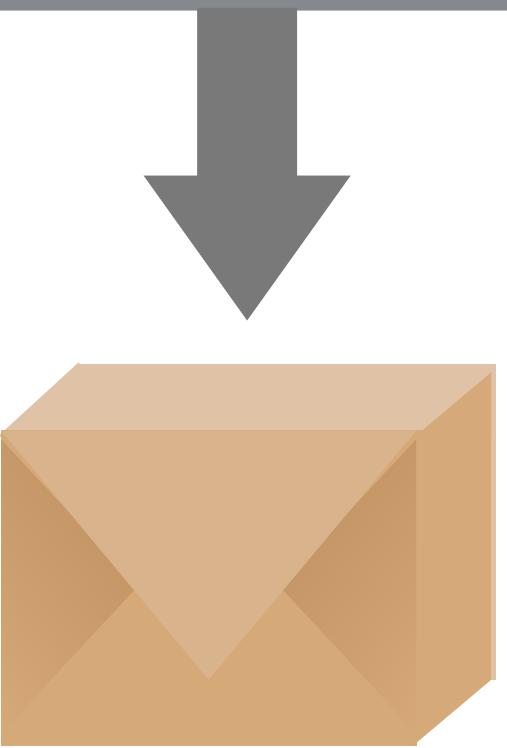
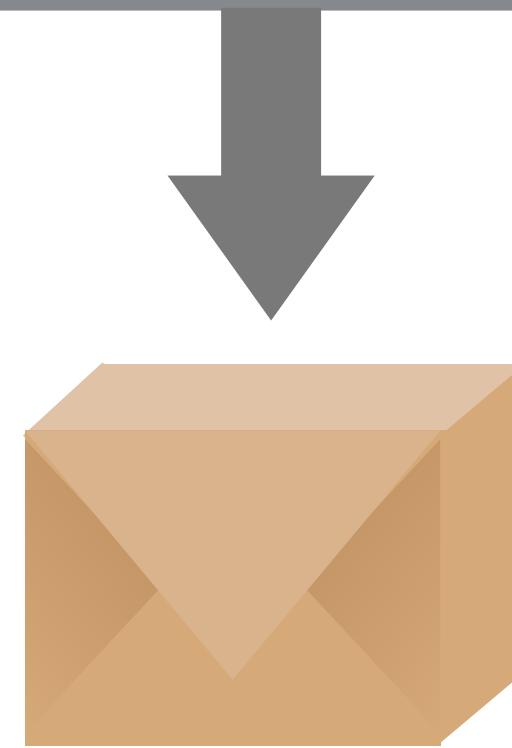
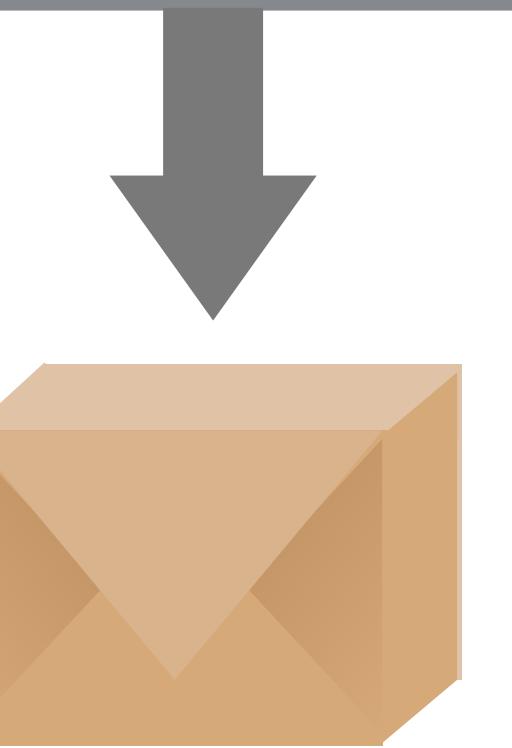
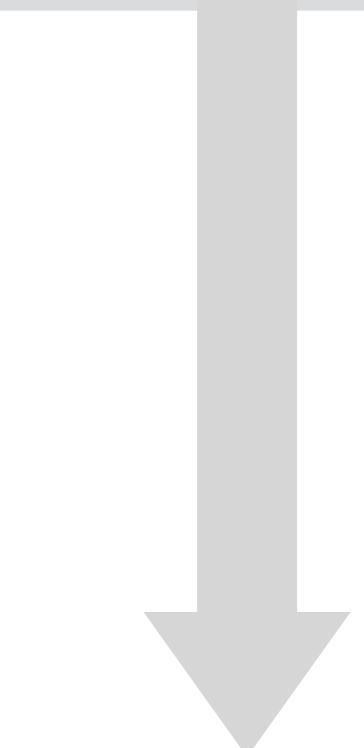
function5()
function6()
function7()
function8()



function9()
functionA()
functionB()
functionC()



functionD()
functionE()
functionF()
functionG()



Base R

R Packages

The screenshot shows a web browser window with the title "The Comprehensive R Archive". The address bar indicates a secure connection to "https://cran.r-project.org". The page content includes the CRAN logo, navigation links for mirrors, what's new, task views, search, about R, software, documentation, and contributed packages. A large section on the right lists available CRAN packages by name, with a link to the full list of packages from A to Z.



[CRAN
Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

[About R](#)
[R Homepage](#)
[The R Journal](#)

[Software](#)
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

[Documentation](#)
[Manuals](#)
[FAQs](#)
[Contributed](#)

[A3](#)
[abyyR](#)
[abc](#)
[ABCAnalysis](#)
[abc.data](#)
[abcdeFBA](#)
[ABCOptim](#)
[ABCp2](#)
[ABC.RAP](#)
[abcrf](#)
[abctools](#)
[abd](#)
[abf2](#)
[ABHgenotypeR](#)
[abind](#)
[abjutils](#)
[abn](#)
[abodOutlier](#)

Available CRAN Packages By Name

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

[Accurate, Adaptable, and Accessible Error Metrics for Predictive Models](#)

[Access to Abbyy Optical Character Recognition \(OCR\) API](#)

[Tools for Approximate Bayesian Computation \(ABC\)](#)

[Computed ABC Analysis](#)

[Data Only: Tools for Approximate Bayesian Computation \(ABC\)](#)

[ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package](#)

[Implementation of Artificial Bee Colony \(ABC\) Optimization](#)

[Approximate Bayesian Computational Model for Estimating P2](#)

[Array Based CpG Region Analysis Pipeline](#)

[Approximate Bayesian Computation via Random Forests](#)

[Tools for ABC Analyses](#)

[The Analysis of Biological Data](#)

[Load Gap-Free Axon ABF2 Files](#)

[Easy Visualization of ABH Genotypes](#)

[Combine Multidimensional Arrays](#)

[Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association](#)

[Modelling Multivariate Data with Additive Bayesian Networks](#)

[Angle-Based Outlier Detection](#)

Using packages

1

```
install.packages("foo")
```

Downloads files to "computer"

1 x per "computer"

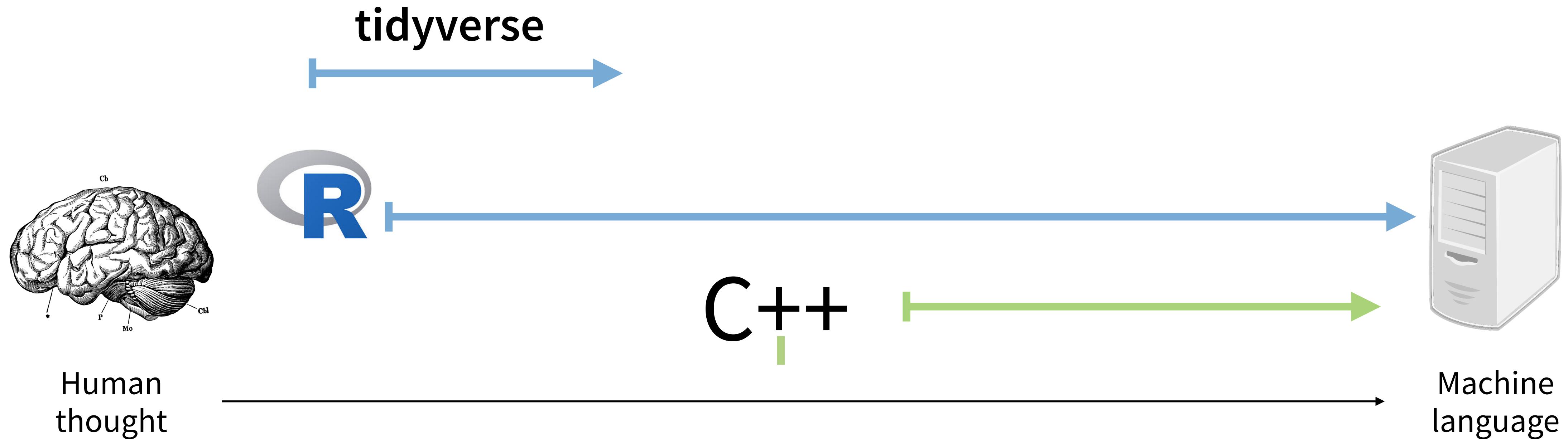
2

```
library("foo")
```

Loads package

1 x per R Session

The tidyverse - A set of R packages to unify some data science tasks



tidyverse.org

The screenshot shows the homepage of tidyverse.org. At the top, there's a navigation bar with links for Packages, Articles, Learn, Help, and Contribute. Below the navigation, there's a large heading "Tidyverse". To the right of the heading, there's a brief description of what the tidyverse is: "R packages for data science. The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying philosophy and common APIs." Below this text, there's a code block showing the command to install the tidyverse: "install.packages("tidyverse")". On the left side of the page, there's a graphic featuring six hexagonal icons representing different tidyverse packages: dplyr (orange, with a pliers icon), ggplot2 (grey, with a line plot icon), readr (blue, with a document icon), purrr (white with a cat icon), tibble (dark blue, with a grid icon), and tidyr (orange, with a circular arrow icon).

Tidyverse

Packages Articles Learn Help Contribute

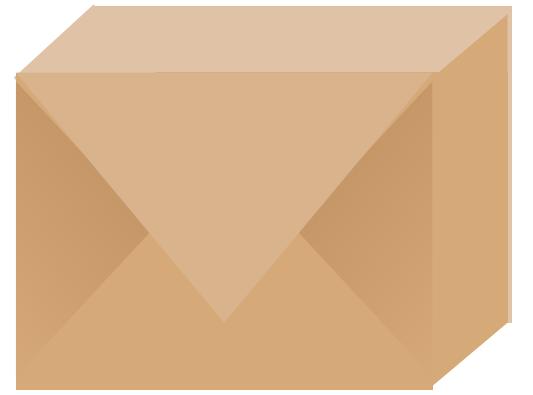
R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying philosophy and common APIs.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

tidyverse



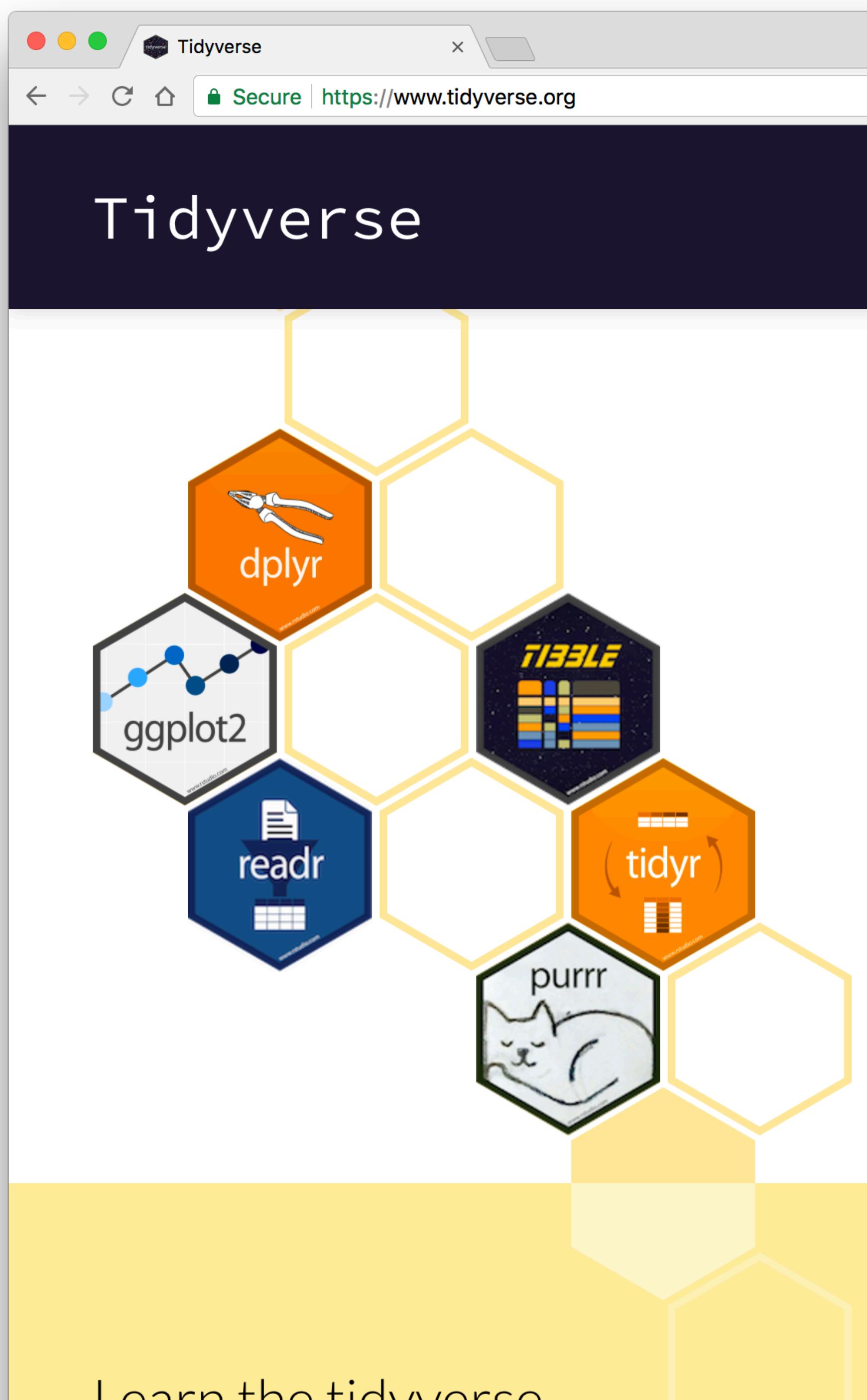
An R package that serves as a short cut for installing and loading the components of the tidyverse.

```
library("tidyverse")
```

```
install.packages("tidyverse")
```

does the equivalent of

```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
```



Learn the tidyverse

```
install.packages("tidyverse")
```

does the equivalent of

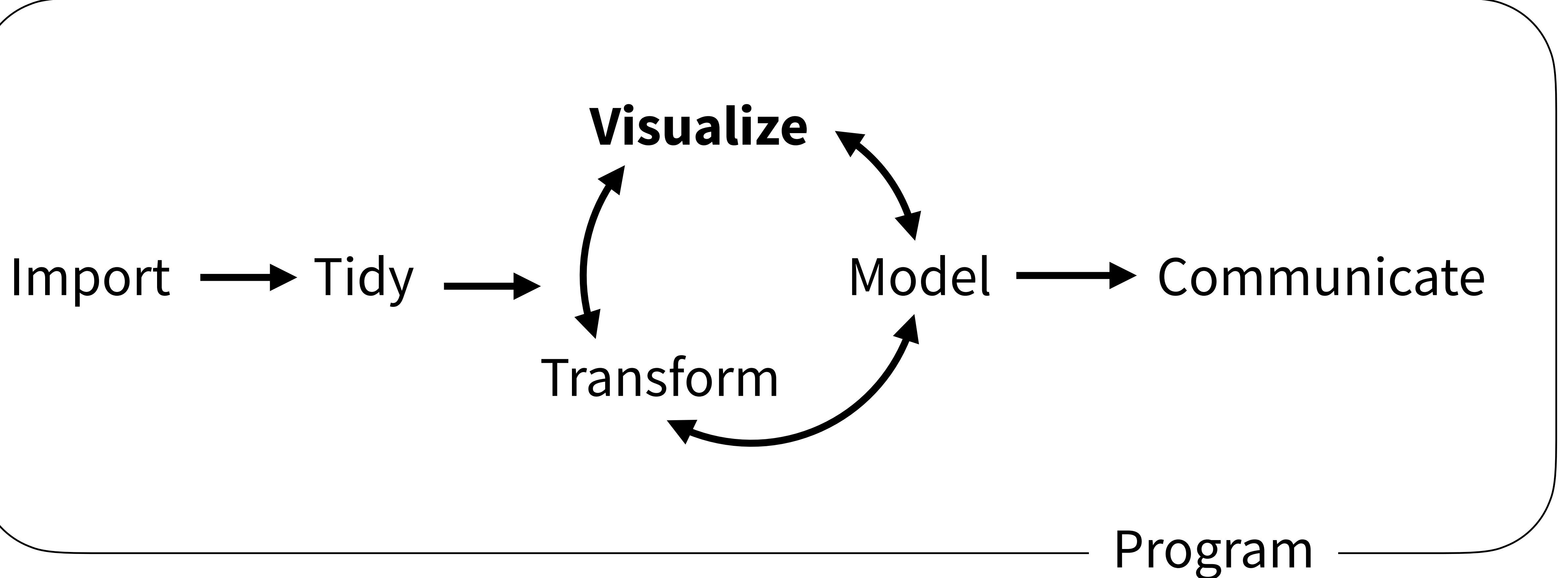
```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
```

```
library("tidyverse")
```

does the equivalent of

```
library("ggplot2")
library("dplyr")
library("tidyr")
library("readr")
library("purrr")
library("tibble")
```

(Applied) Data Science

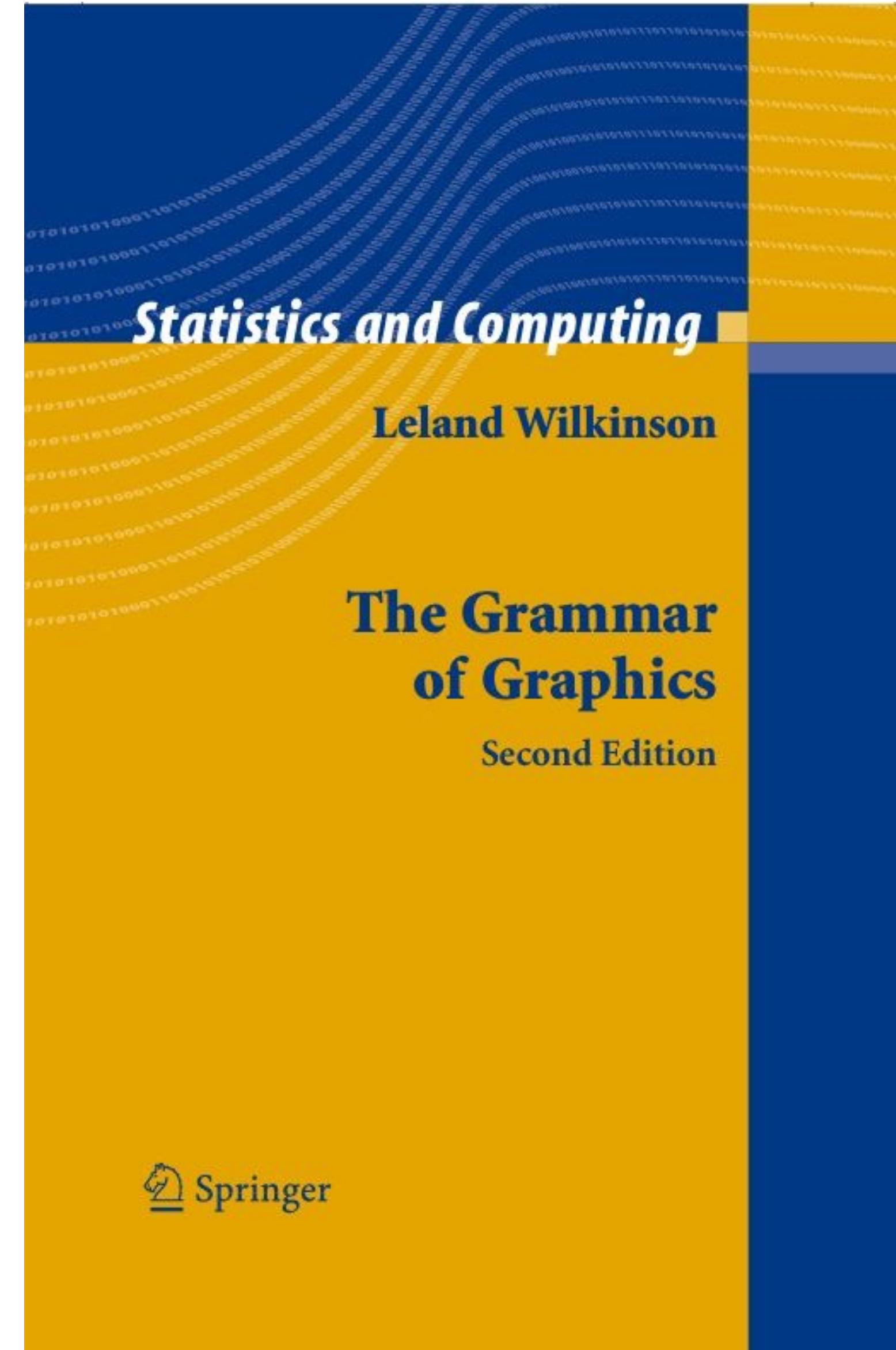


Grammar of Graphics



The Grammar of Graphics

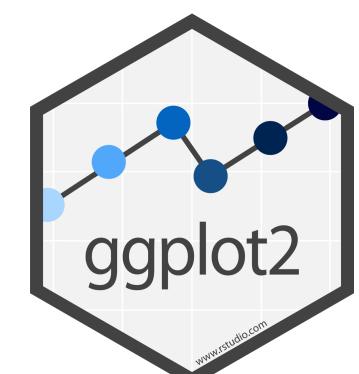
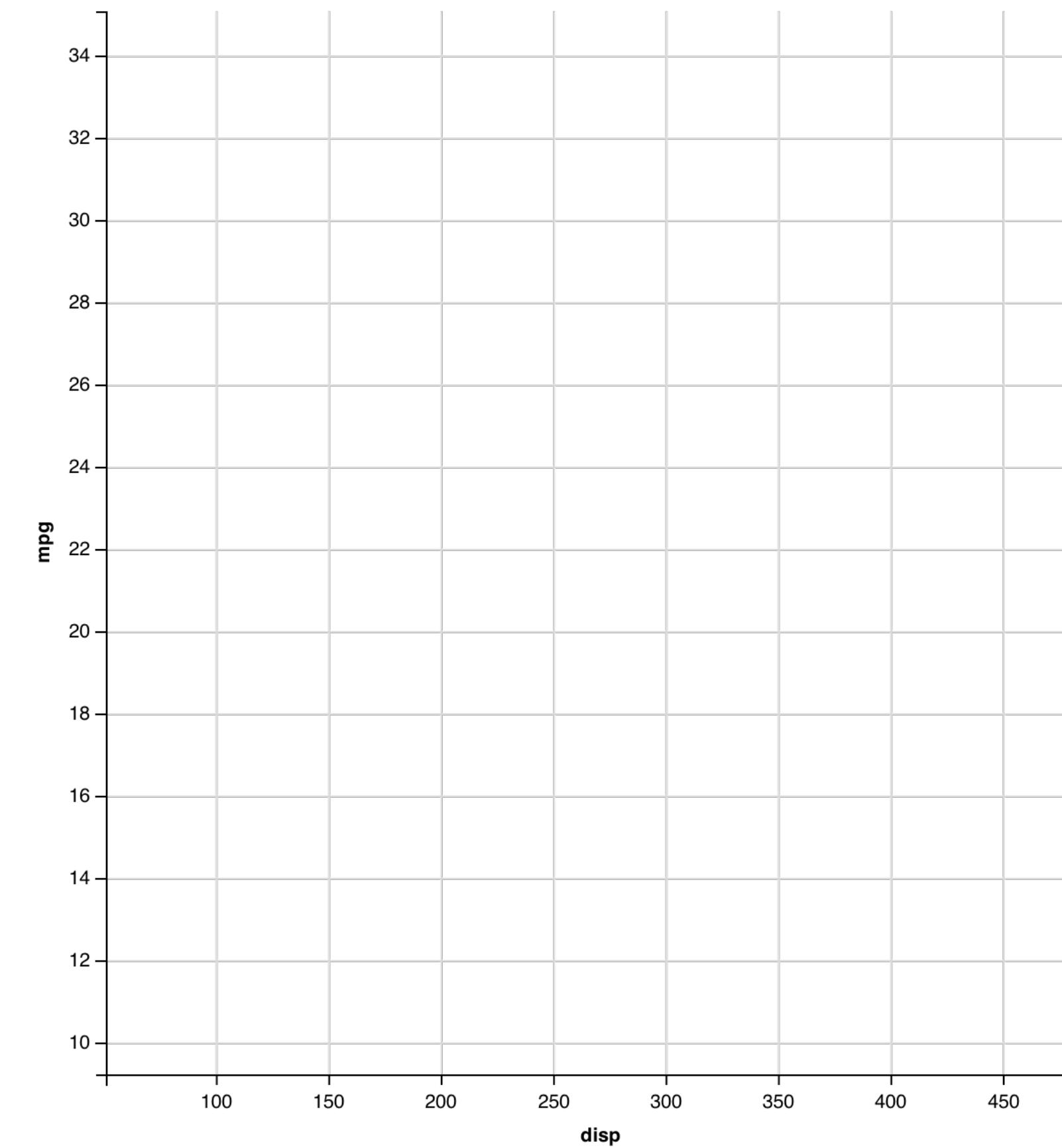
*Coordinates
Statistics
Facets
Geometries
Aesthetics
Data*



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

data

geom



mappings

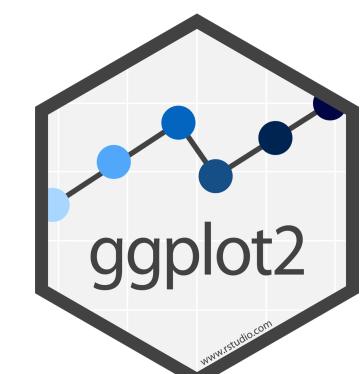
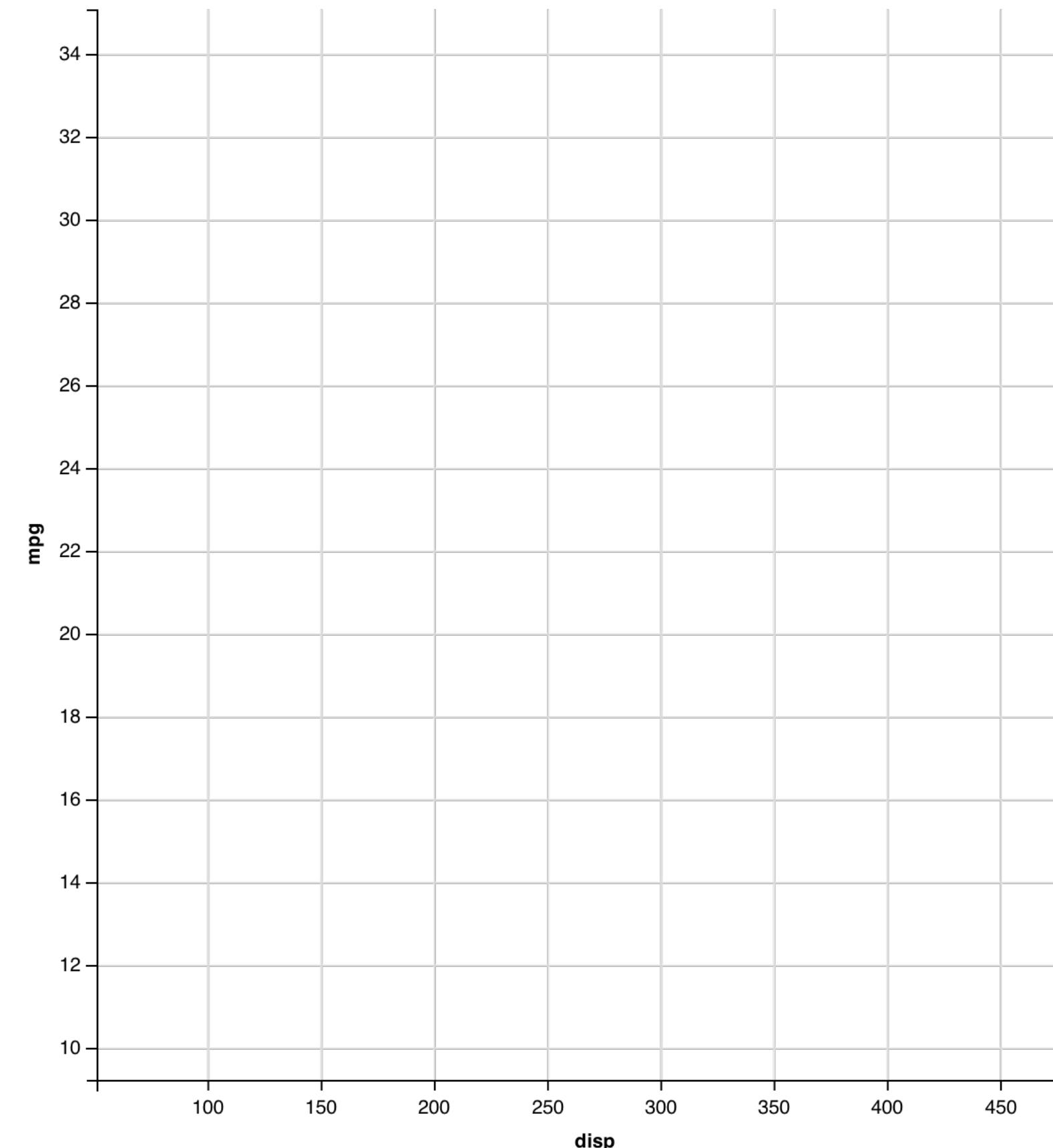
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

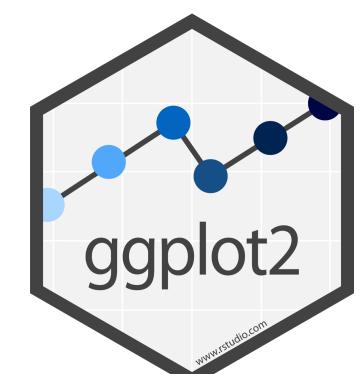
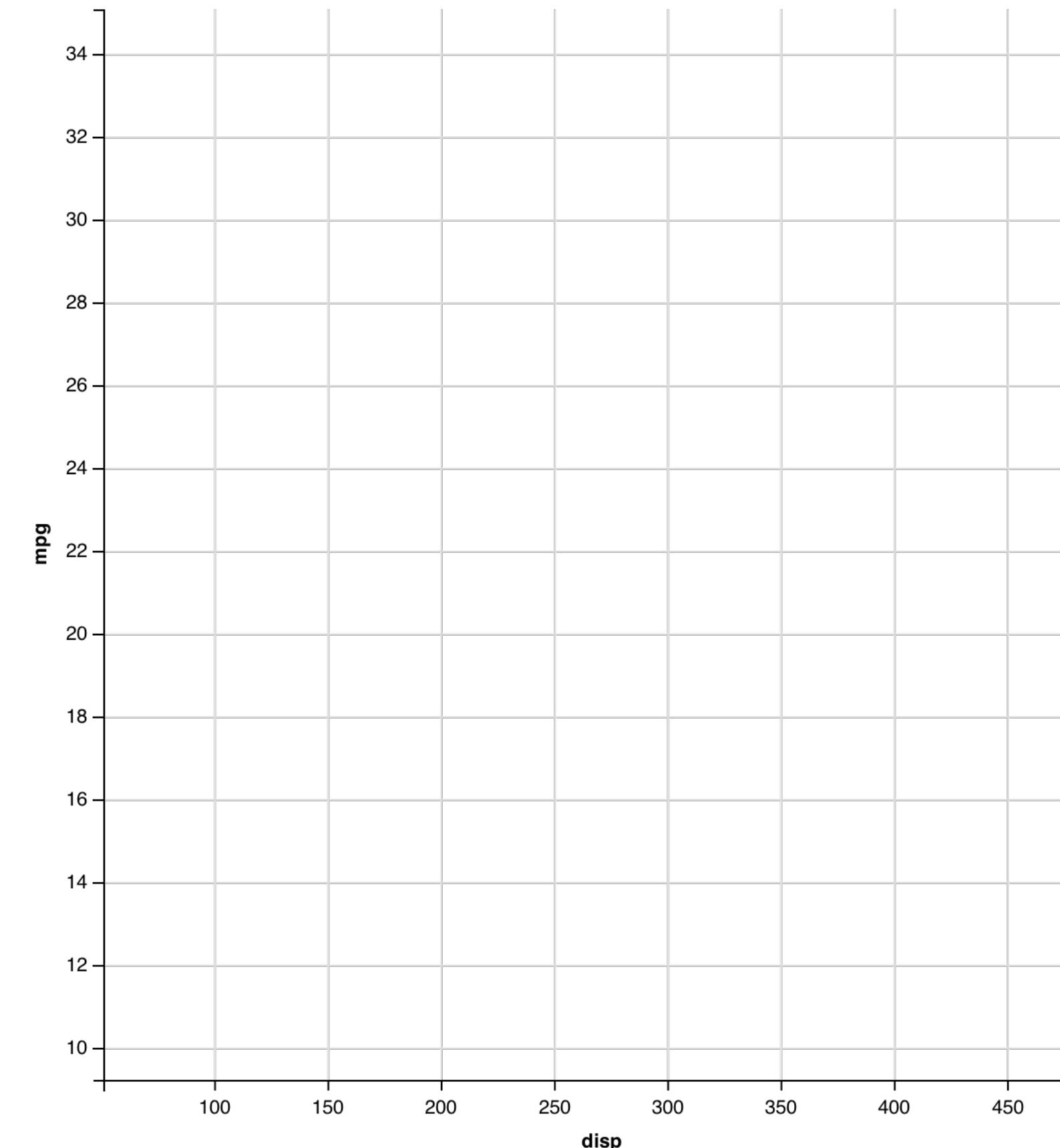


mappings

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

data

geom

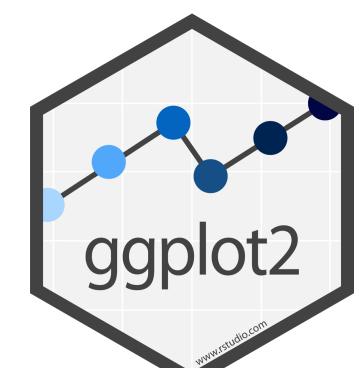
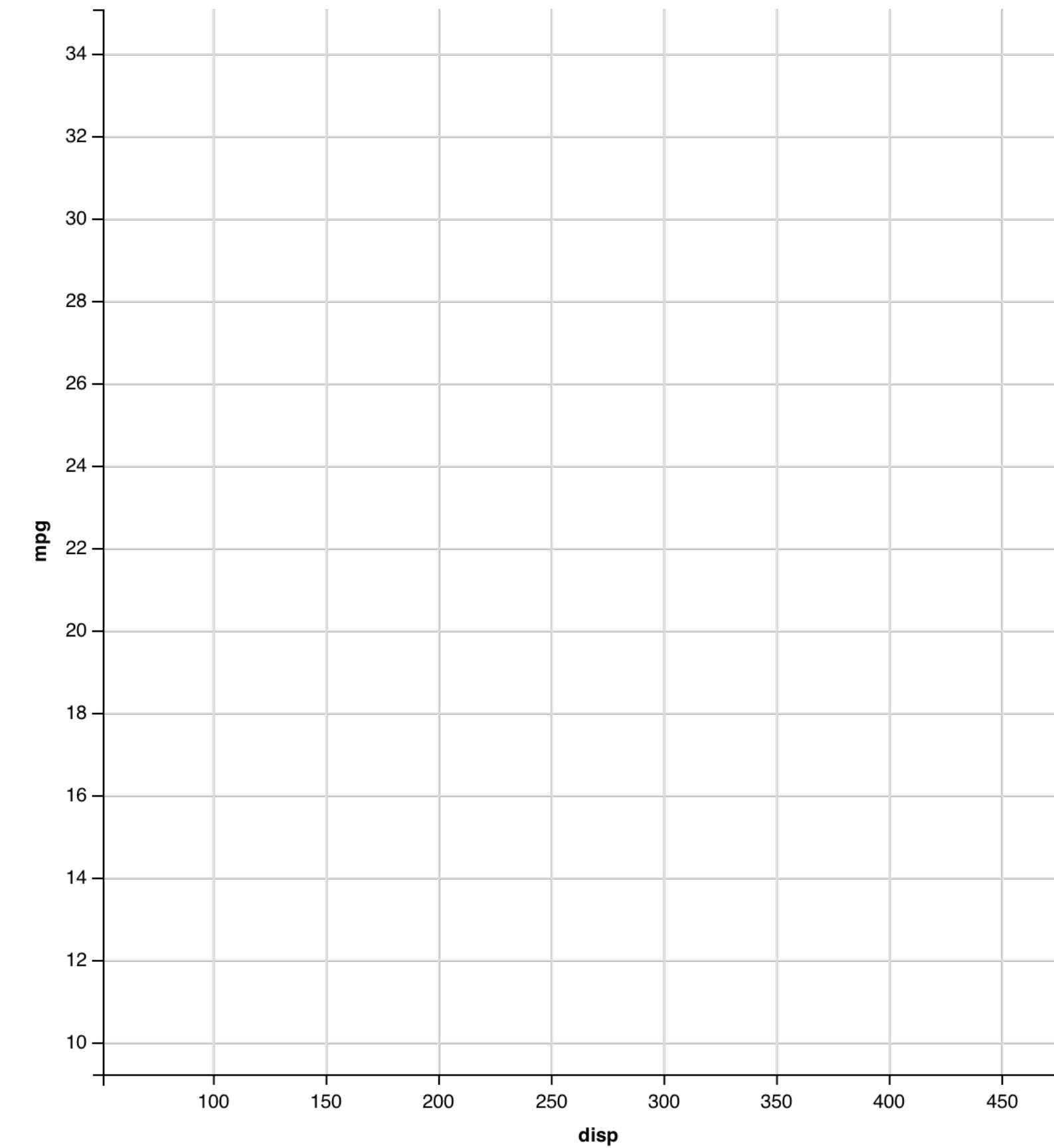


mappings

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

data

geom

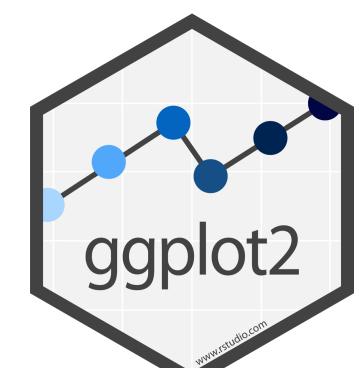
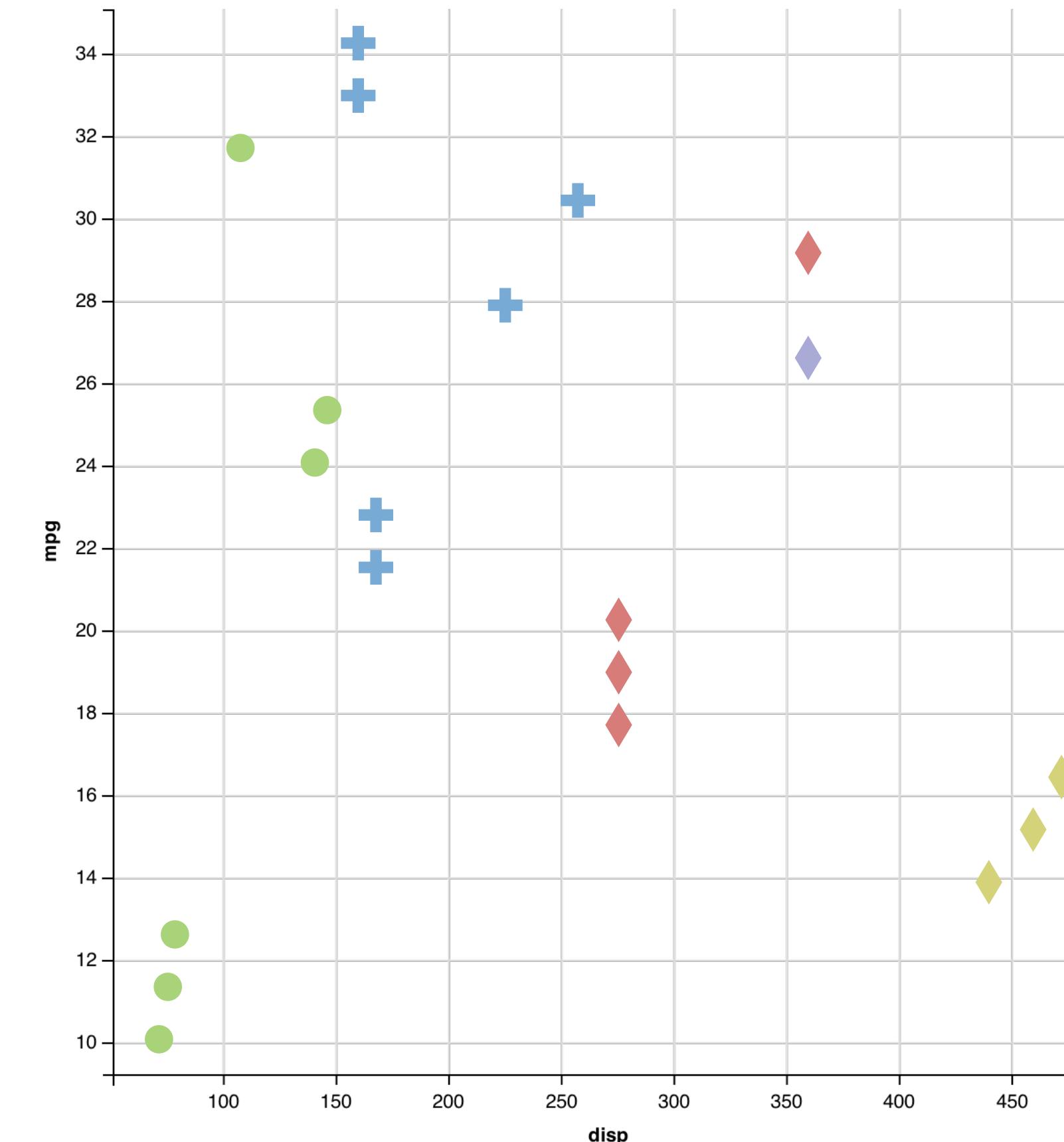


mappings

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

data

geom

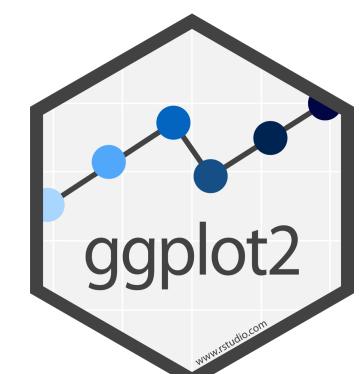
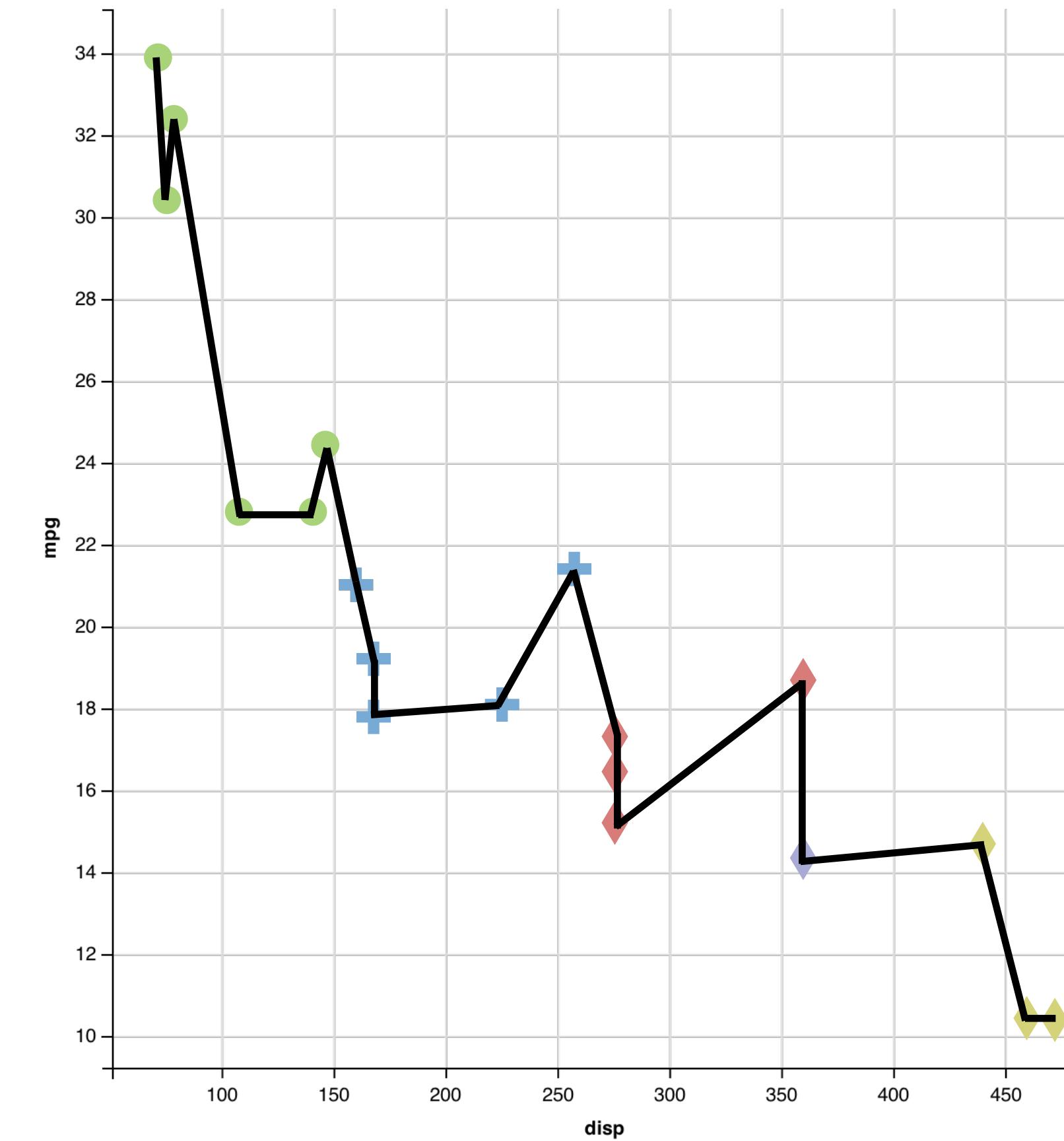


mappings

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

data

geom
points
lines

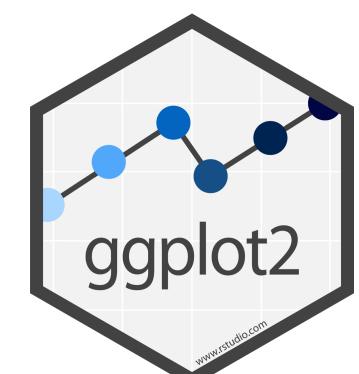
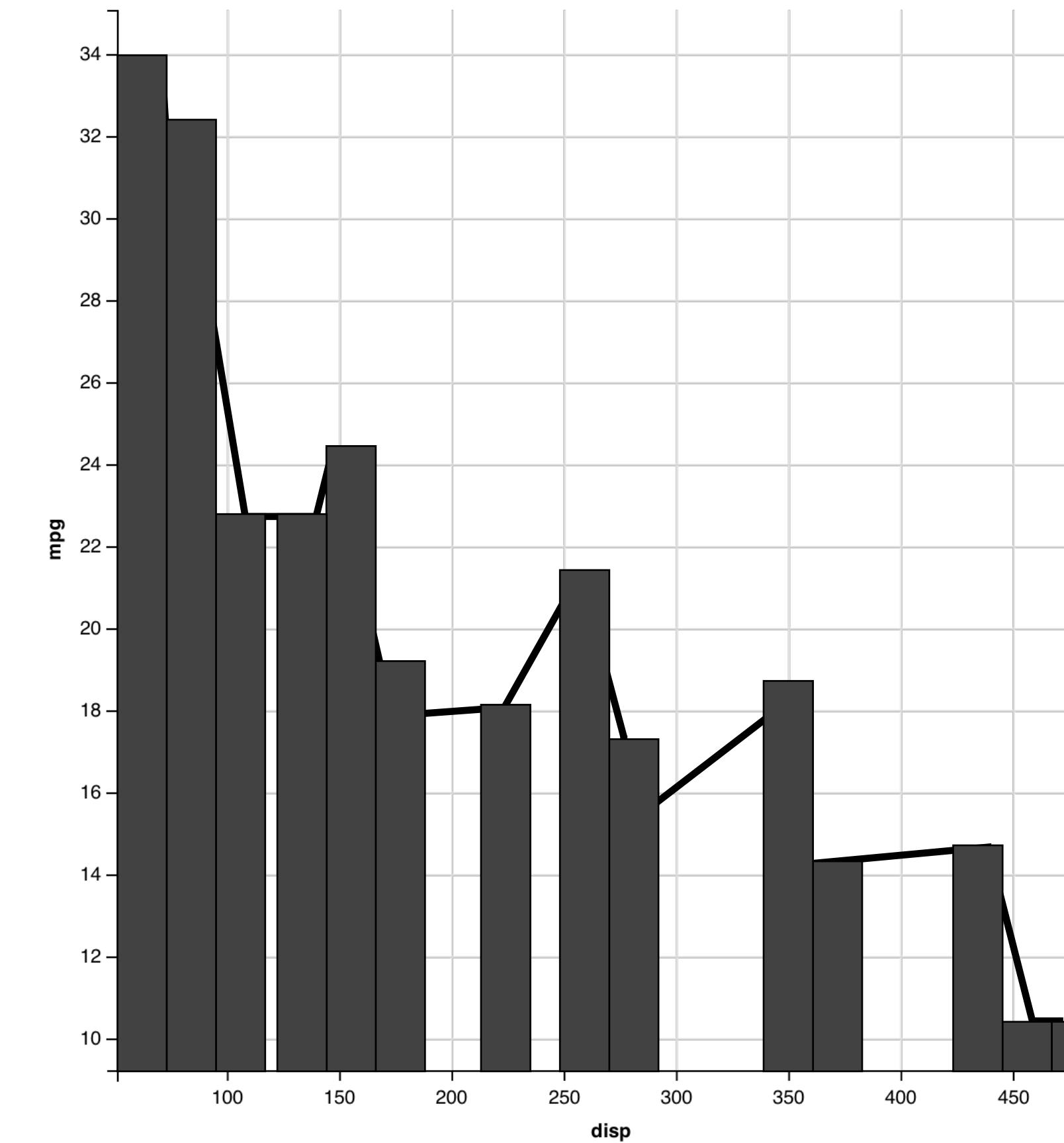


mappings

	y	x
mpg	↓	↑
cyl		
21.0	6	160.0
21.0	6	160.0
22.8	4	108.0
21.4	6	258.0
18.7	8	360.0
18.1	6	225.0
14.3	8	360.0
24.4	4	146.7
22.8	4	140.8
19.2	6	167.6
17.8	6	167.6
16.4	8	275.8
17.3	8	275.8
15.2	8	275.8
10.4	8	472.0
10.4	8	460.0
14.7	8	440.0
32.4	4	78.7
30.4	4	75.7
33.9	4	71.1

data

geom
points
lines
bars

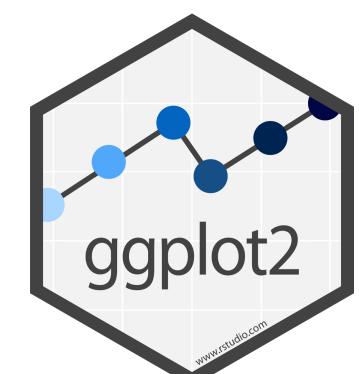
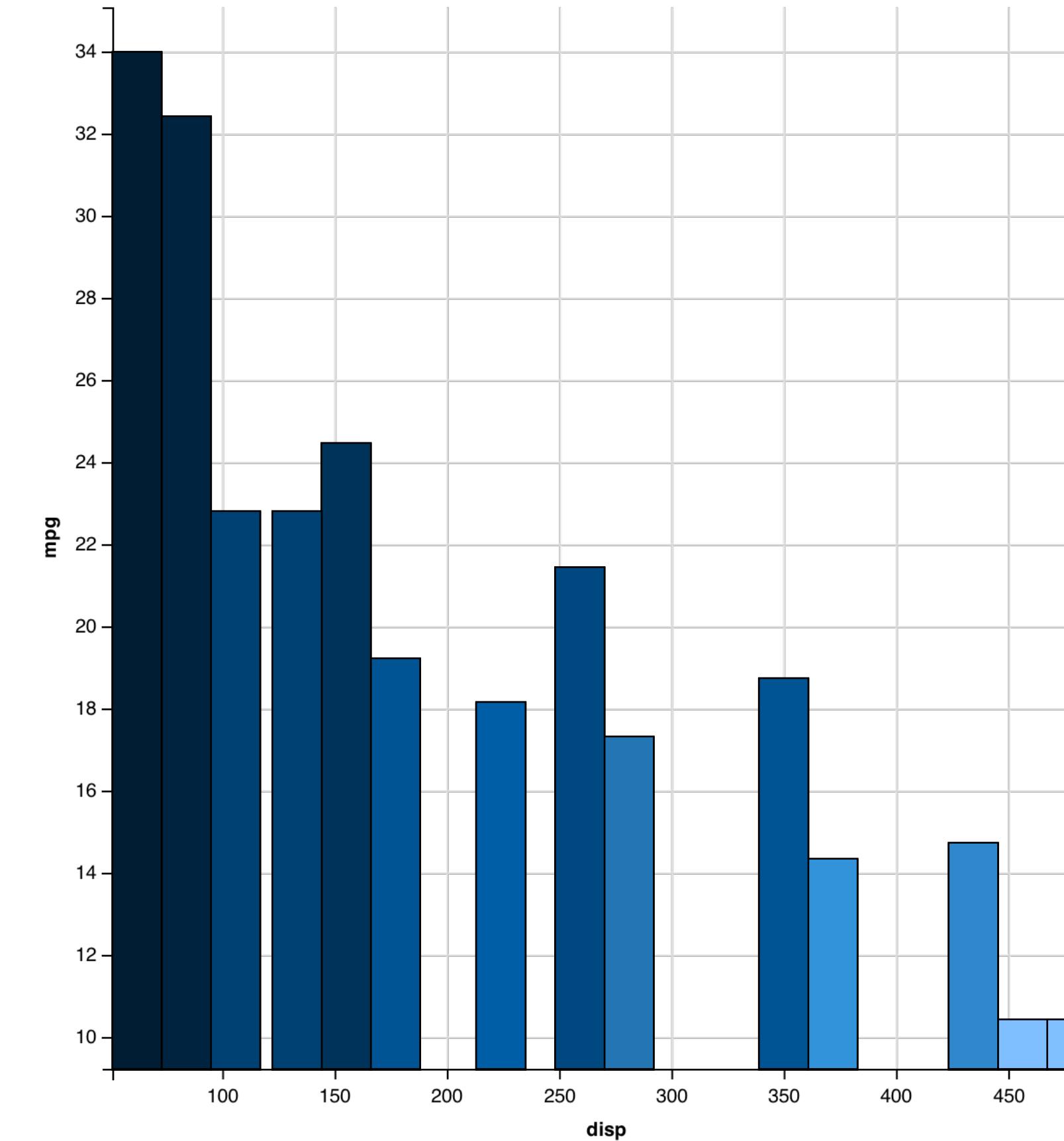


mappings

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

data

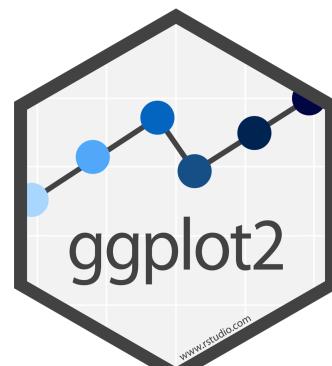
geom
points
lines
bars



To make a graph

[template]

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



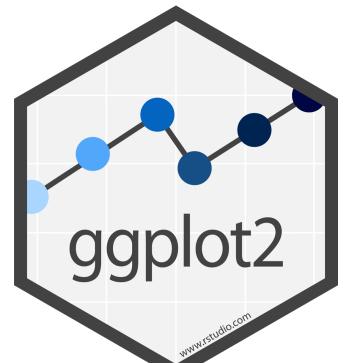
To make a graph

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

data

1. Pick a **data** set

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



To make a graph

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

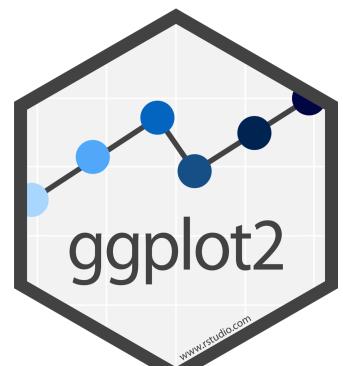
data

geom

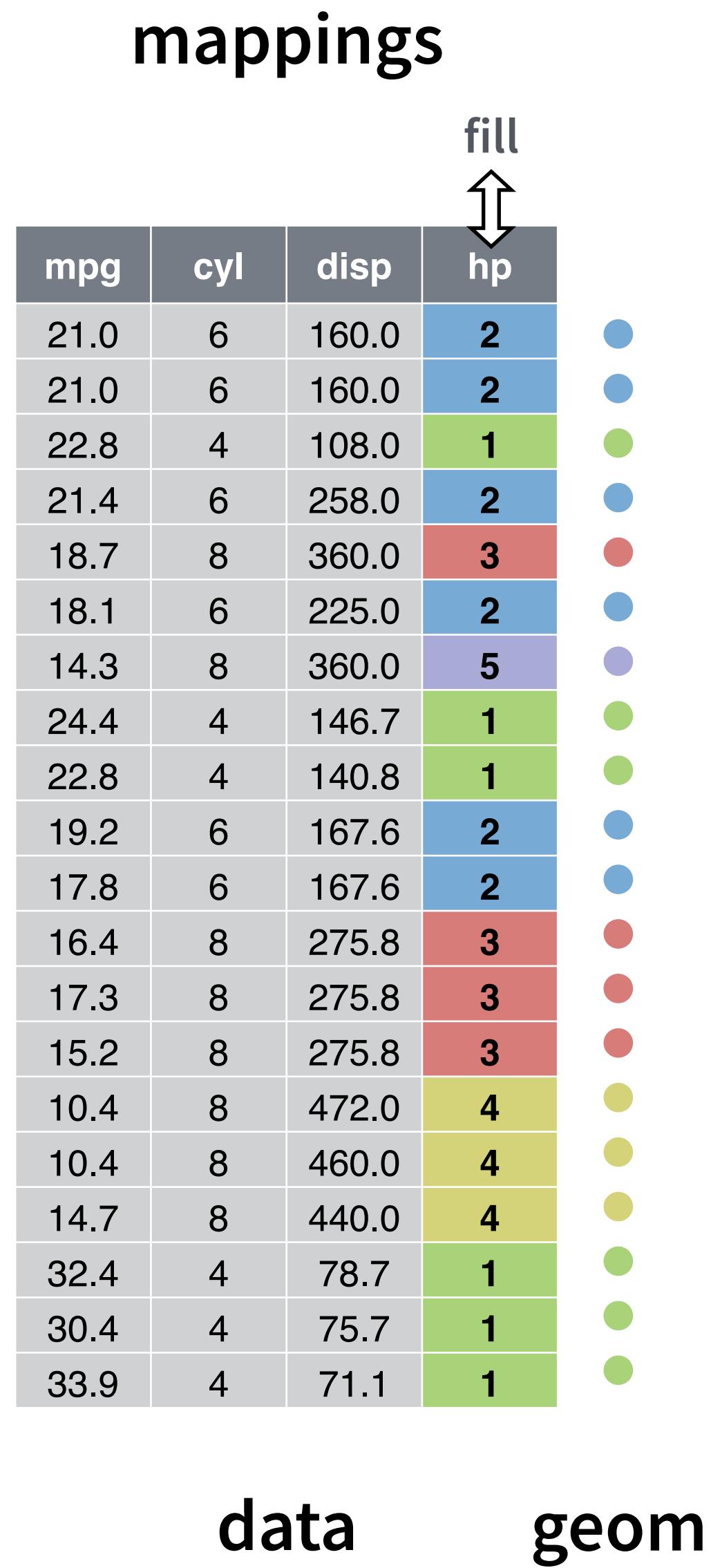
1. Pick a **data** set

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

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



To make a graph

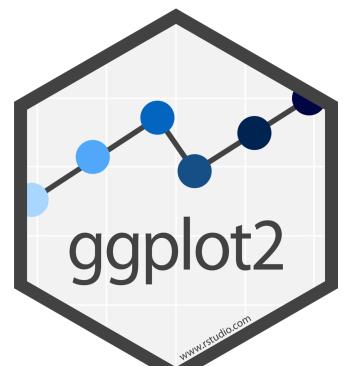


1. Pick a **data** set

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

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

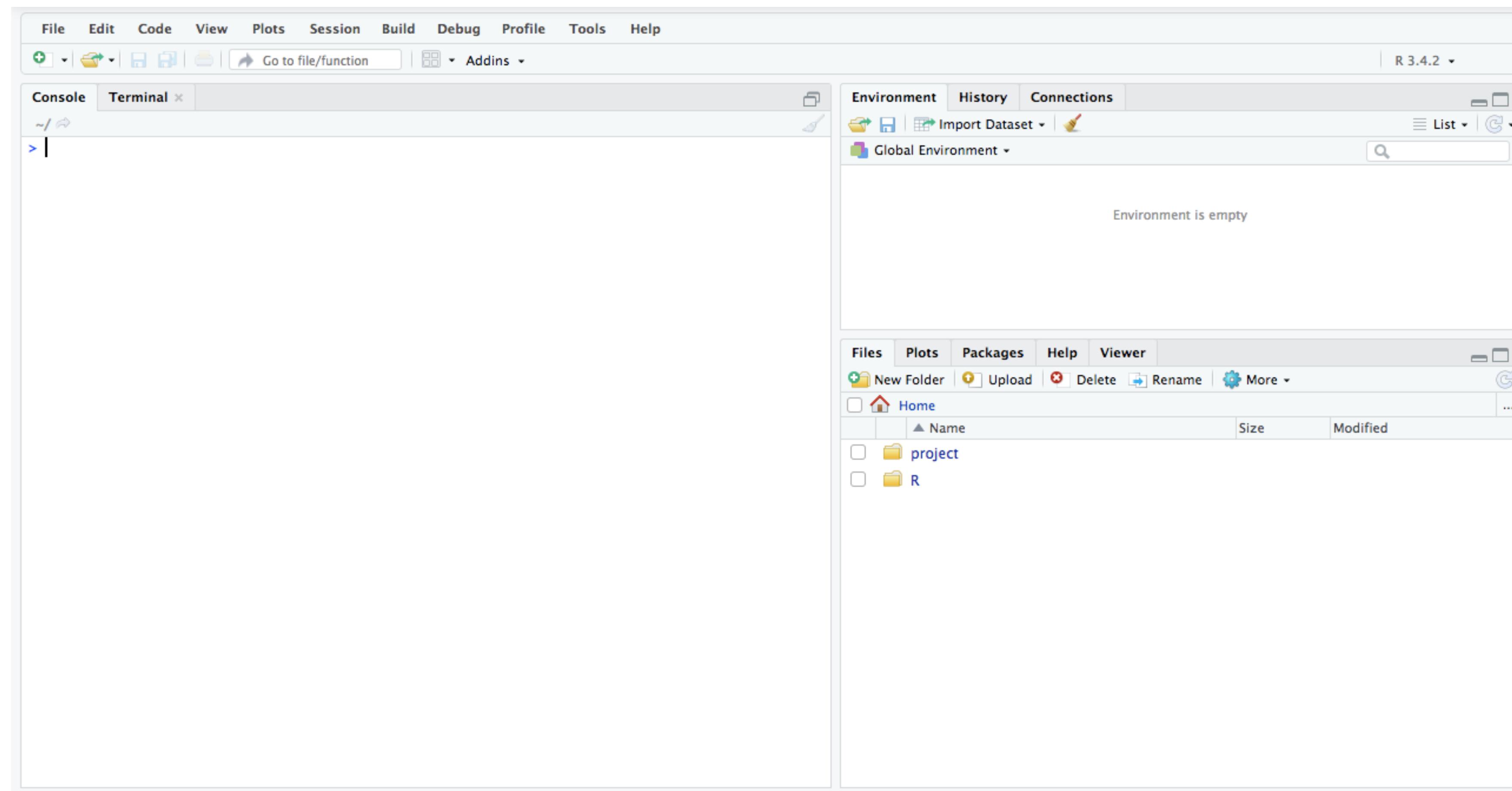
3. **Map** aesthetic
properties to
variables



RStudio

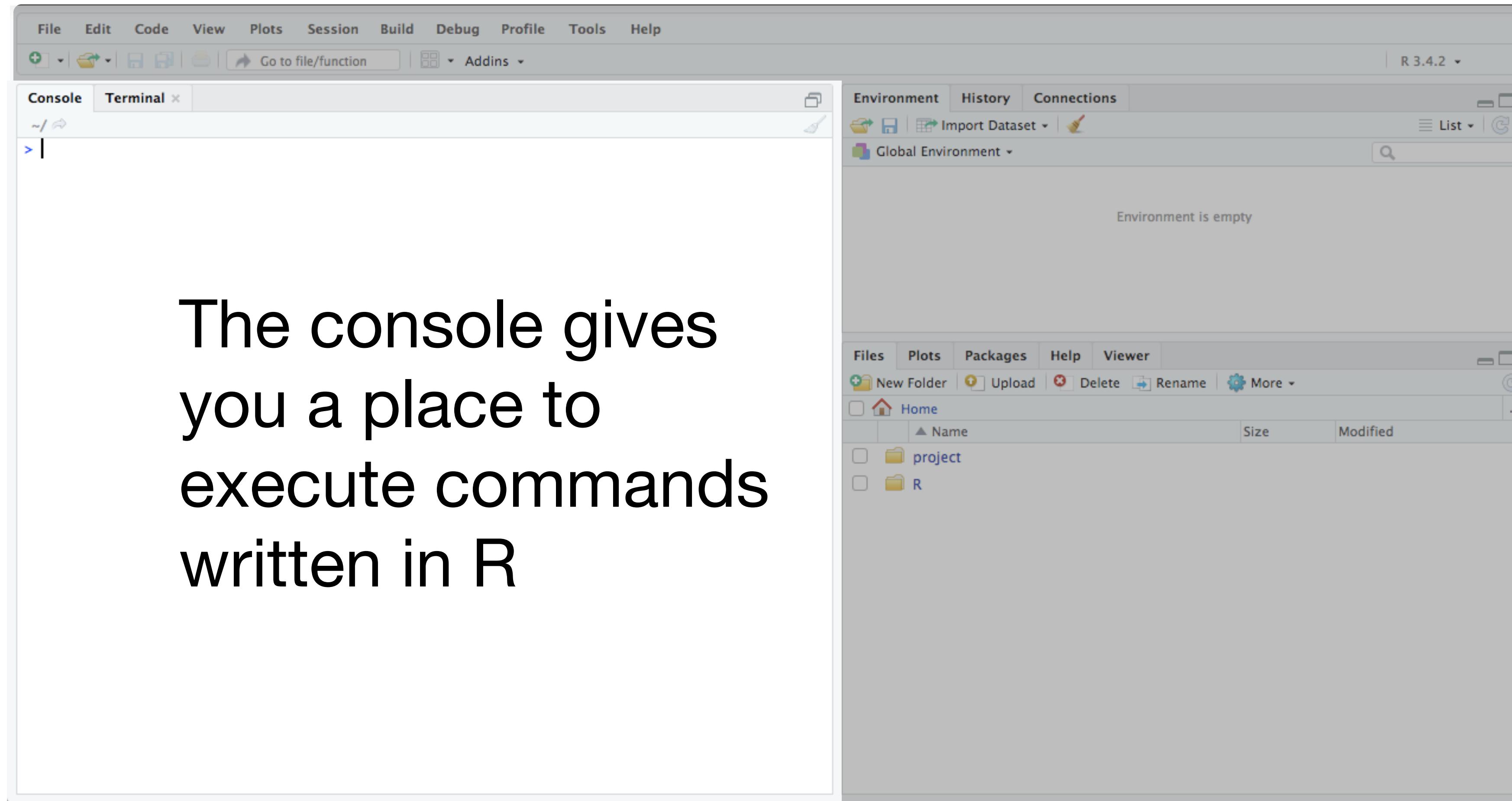
(let's start!)

RStudio

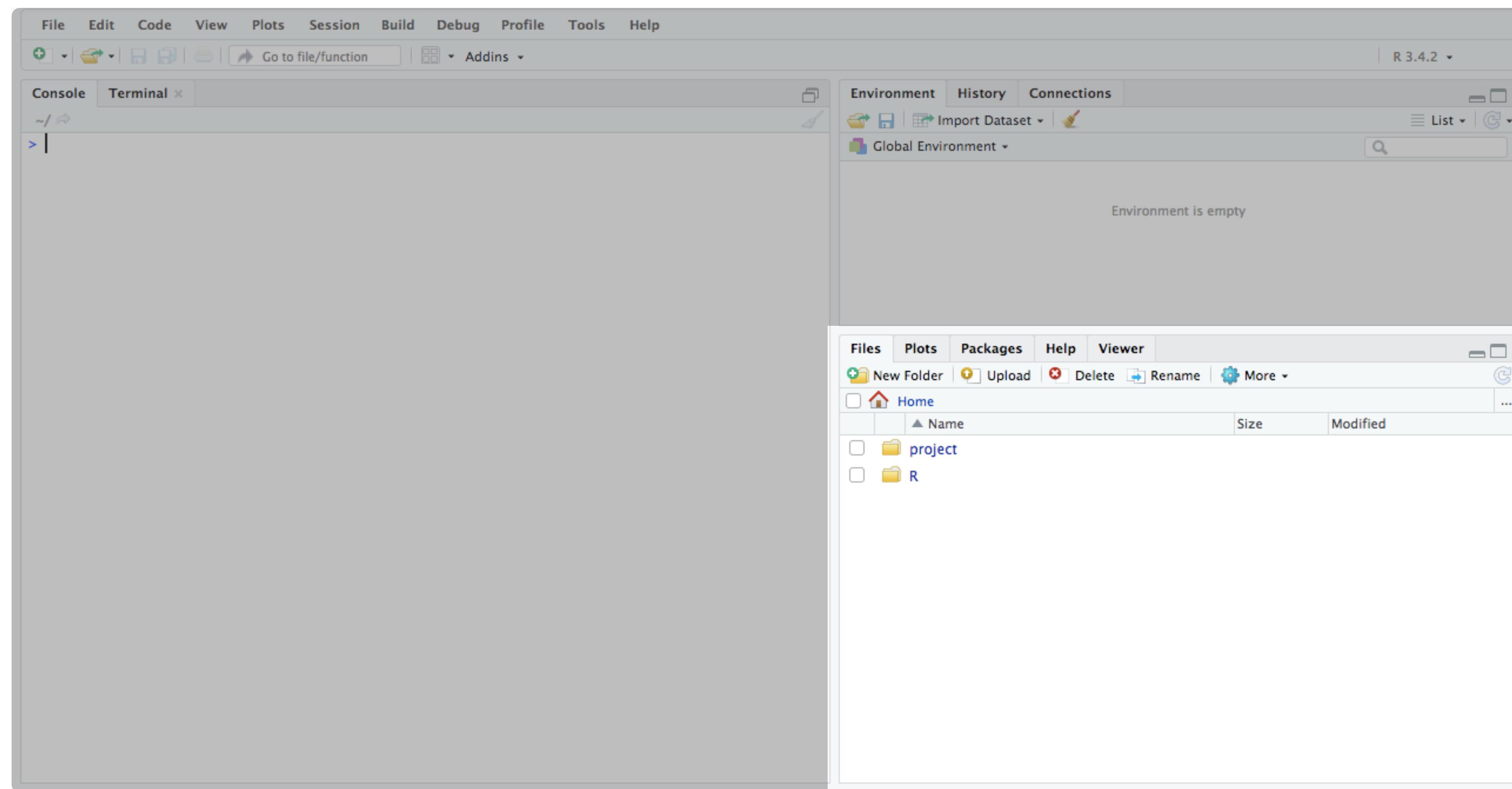


RStudio

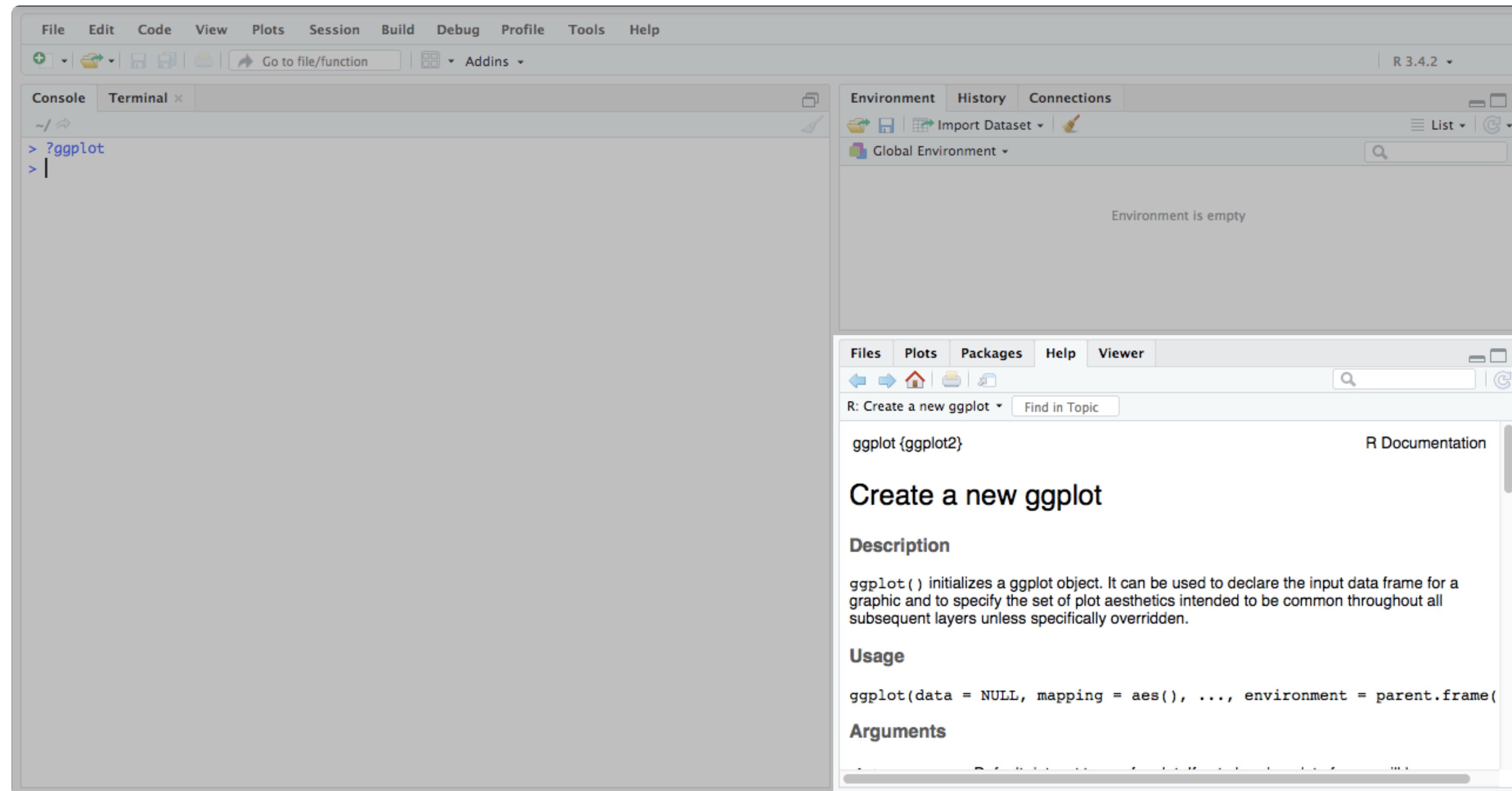
The console gives
you a place to
execute commands
written in R



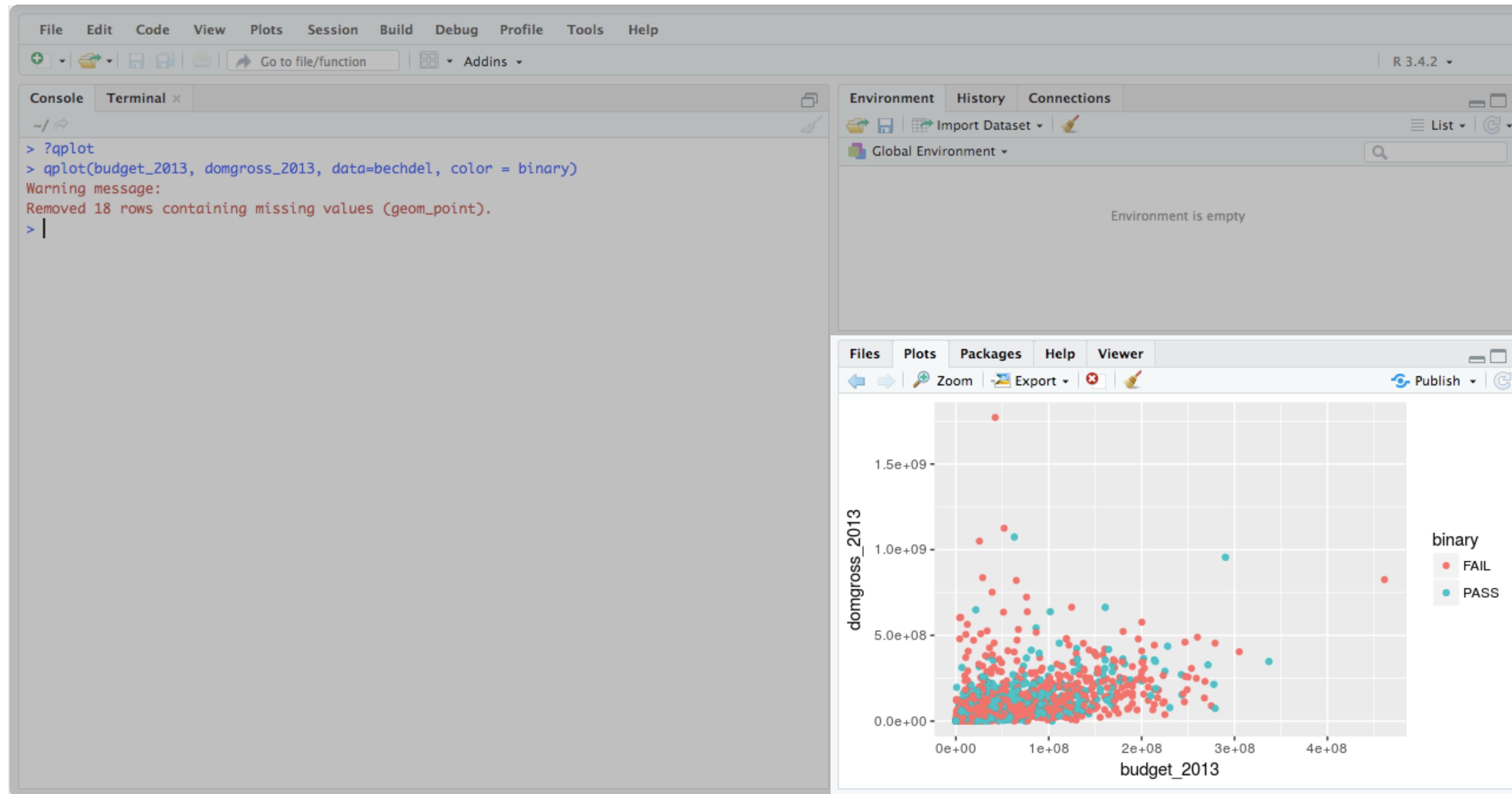
RStudio



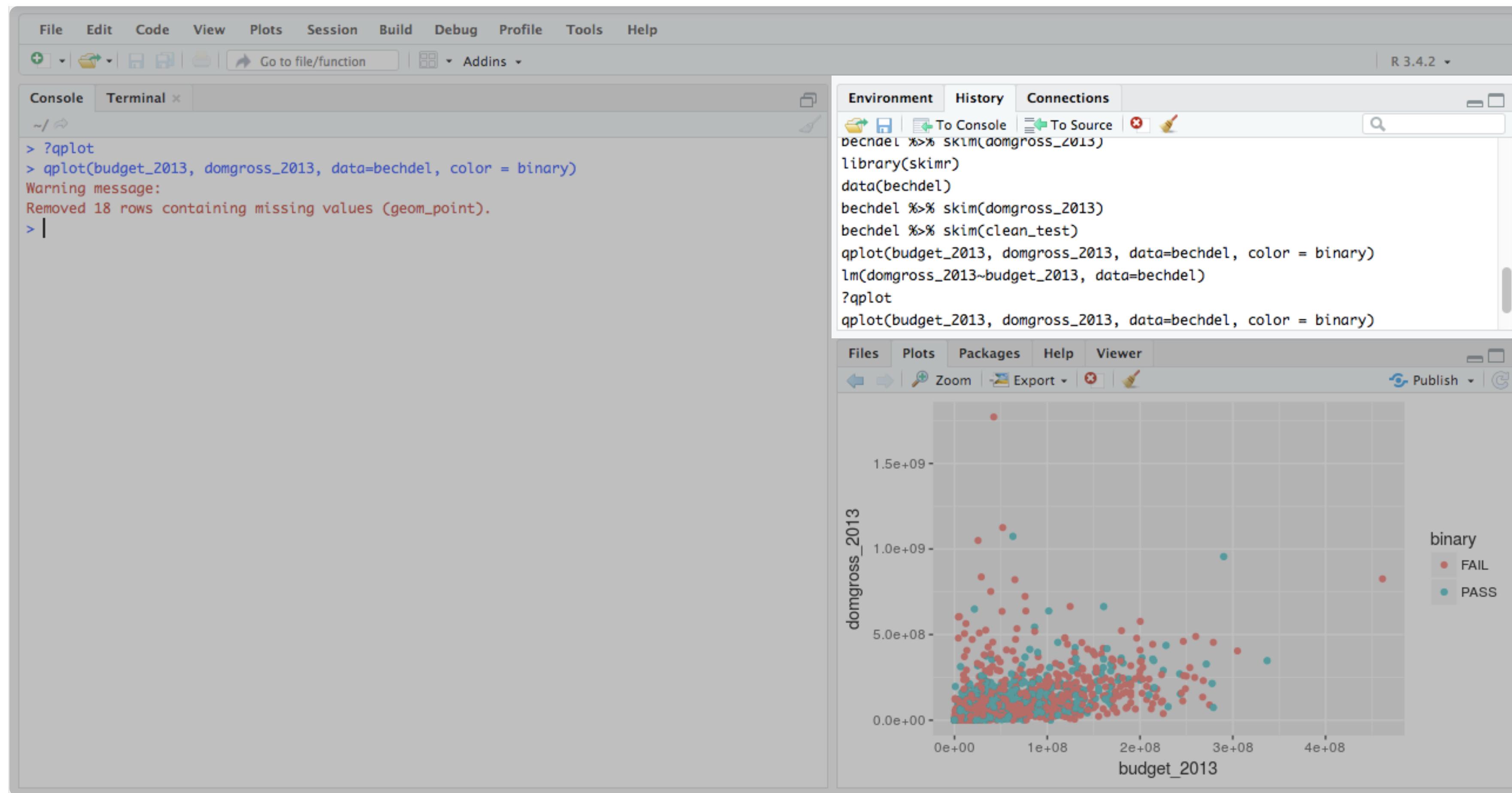
RStudio



RStudio



RStudio



RStudio

Screenshot of the RStudio IDE interface.

Left Panel (Code Editor):

```
1 ---  
2 title: "Untitled"  
3 output: html_document  
4 ---  
5  
6 ```{r setup, include=FALSE}  
7 knitr::opts_chunk$set(echo = TRUE)  
8  
9  
10 ## R Markdown  
11  
12 This is an R Markdown document. Markdown is a simple formatting syntax for authoring  
HTML, PDF, and MS Word documents. For more details on using R Markdown see  
http://rmarkdown.rstudio.com.  
13  
14 When you click the **Knit** button a document will be generated that includes both  
2:1 # Untitled
```

Bottom Left Panel (Console):

```
> ?qplot  
> qplot(budget_2013, domgross_2013, data=bechdel, color = binary)  
Warning message:  
Removed 18 rows containing missing values (geom_point).  
>
```

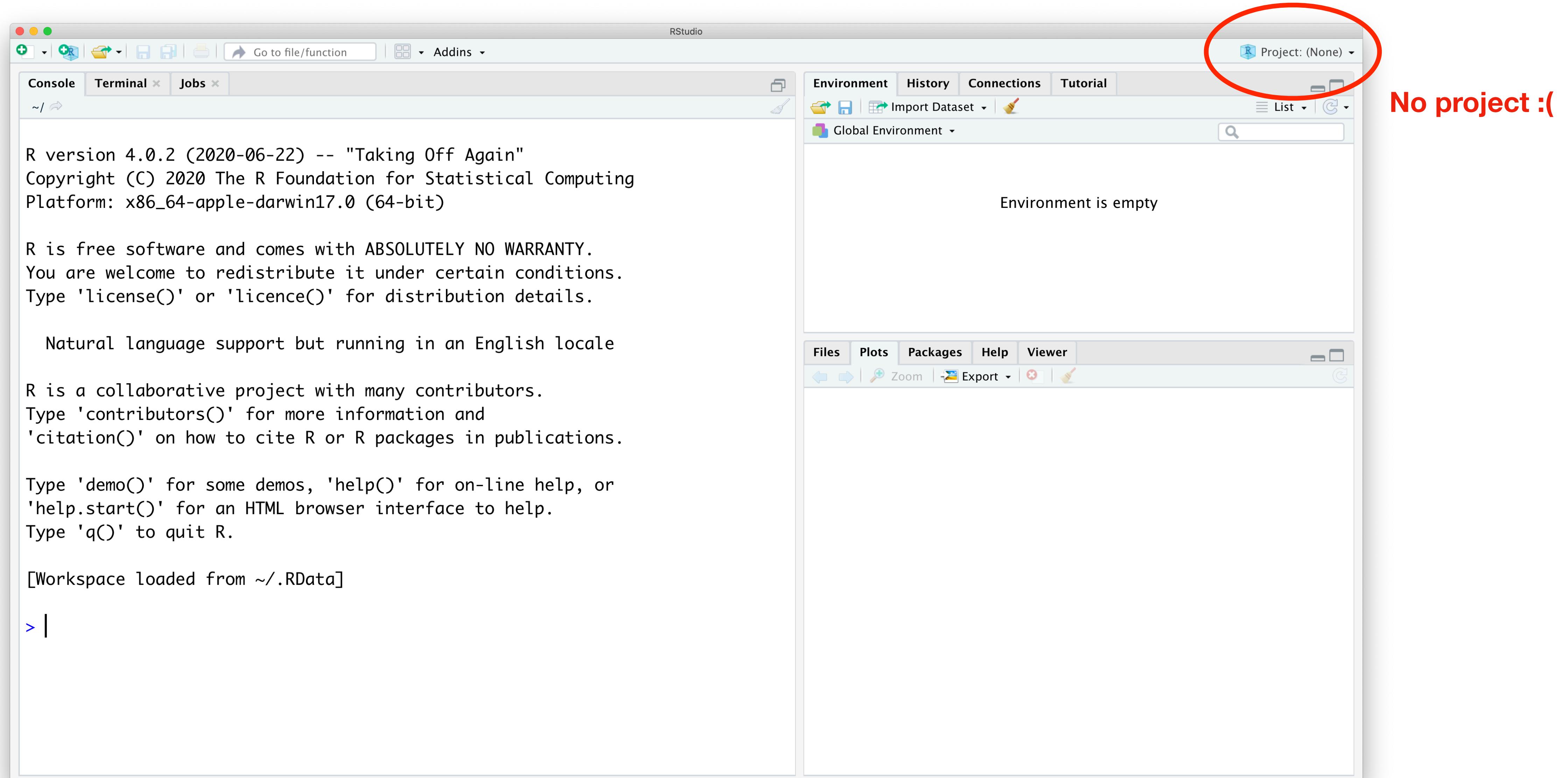
Right Panel (Plots):

The plot is titled "domgross_2013". The X-axis is labeled "budget_2013" and the Y-axis is labeled "domgross_2013". A legend on the right side of the plot area is titled "binary" and shows two categories: "FAIL" represented by a red circle and "PASS" represented by a teal circle.

RStudio projects



I highly, highly recommend projects



Make a project

- Create a folder for this class
 - If you already have one, use that
 - If you don't have one, make one
 - Probably you want that folder on OneDrive
- Put all your files related to this class into that folder
- Make a new Project in RStudio
- ...
- Profit!

The screenshot shows the RStudio interface with the following components:

- File Menu:** Opened, showing options like New File, New Project..., Open File..., and Recent Files.
- Console:** Displays the R startup message and workspace loading information.
- Environment:** Shows the Global Environment pane which is currently empty.
- Plots:** Shows the standard R plotting interface with tabs for Files, Plots, Packages, Help, and Viewer.

Key text visible in the console:

```
R version 3.6.2 (2019-12-12) -- "Taking Off Again"  
Copyright (C) 2019 The R Foundation for Statistical Computing  
Platform: darwin17.0 (64-bit)  
  
comes with ABSOLUTELY NO WARRANTY.  
Distribute it under certain conditions.  
Type 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
[Workspace loaded from ~/.RData]  
  
> |
```

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for details.

Natural language support but running

R is a collaborative project with many
Type 'contributors()' for more information
'citation()' on how to cite R or R packages.

Type 'demo()' for some demos, 'help()' or
'help.start()' for an HTML browser interface.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> |

New Project Wizard

Create Project

New Directory
Start a project in a brand new working directory >

Existing Directory
Associate a project with an existing working directory >

Version Control
Checkout a project from a version control repository >

Cancel

Environment History Connections Tutorial

Import Dataset

Global Environment

Environment is empty

Help Viewer

Export

Then, browse to your folder

STAT336 - master - RStudio

index.Rmd intro_to_ggplot2.Rmd

1 ---
2 **title: "Visualization"**
3 **output: html_document**
4 ---
5
6 **## Setup**
7
8 The first chunk in an R Notebook is usually titled "setup," and by convention includes the R packages you want to load. Remember, in order to use an R package you have to run some `library()` code every session. Execute these lines of code to load the packages.
9

1:1 # Visualization ▾ R Markdown ▾

Console Terminal Jobs

~/Documents/STAT336/ ↗

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/Documents/STAT336/.RData]

> |

Environment History Connections Git Tutorial

Import Dataset

Global Environment

Environment is empty

Files Plots Packages Help Viewer

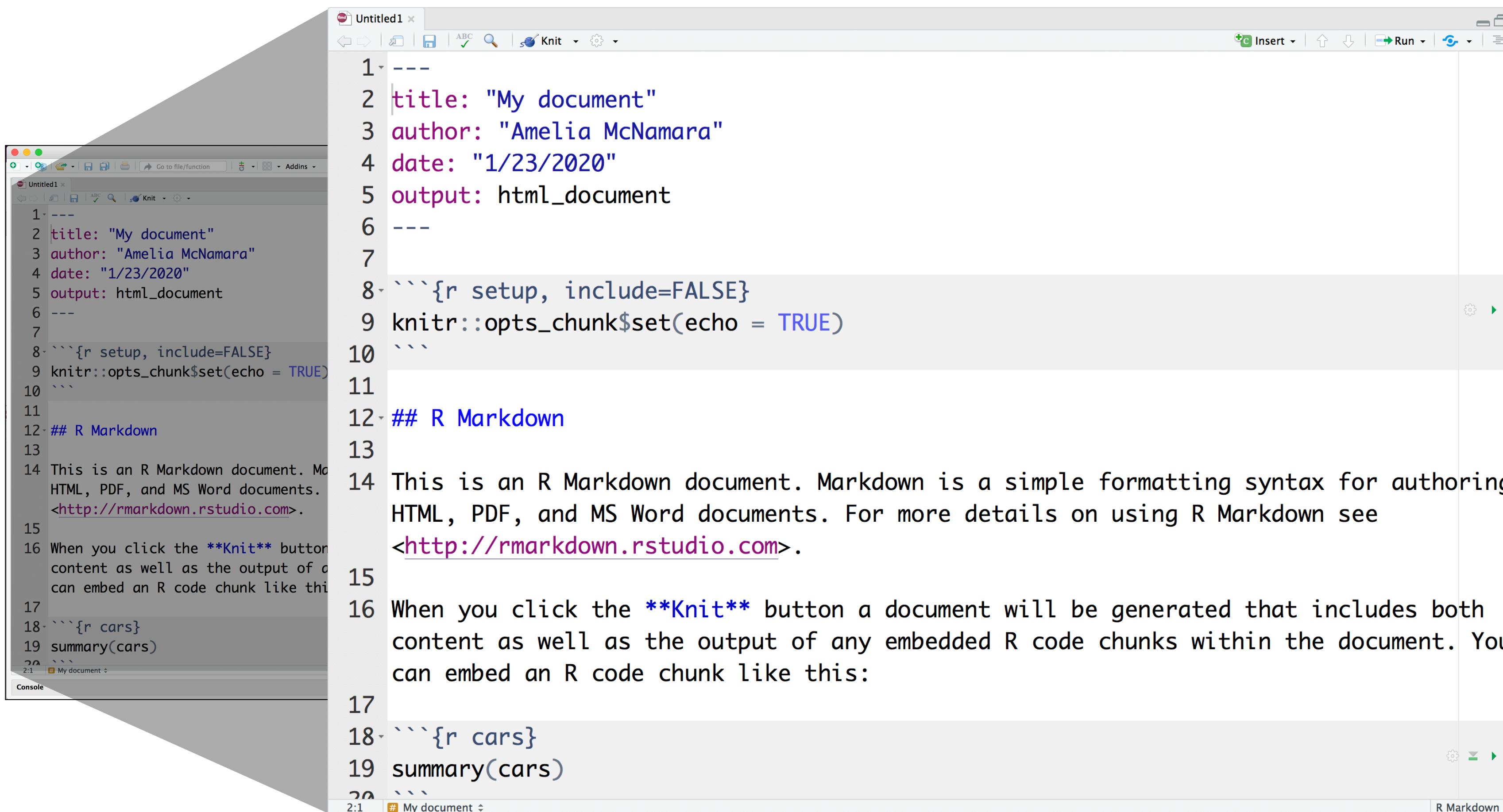
Zoom Export

Yay!

The screenshot shows the RStudio interface with a new project titled 'STAT336'. The 'File' menu is circled in red at the top right. The 'Global Environment' pane in the bottom right shows 'Environment is empty'. A large red 'Yay!' is overlaid on the bottom right of the environment pane. The 'Console' tab in the bottom left shows the standard R startup message about being a collaborative project and how to use various functions like 'contributors()' and 'citation()'. The 'Plots' tab is selected in the bottom right pane.

RMarkdown

An authoring format for Data Science.



```
1 ---  
2 title: "My document"  
3 author: "Amelia McNamara"  
4 date: "1/23/2020"  
5 output: html_document  
---  
---  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12 ## R Markdown  
13  
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring  
HTML, PDF, and MS Word documents.  
http://rmarkdown.rstudio.com.  
15  
16 When you click the **Knit** button a document will be generated that includes both  
content as well as the output of any embedded R code chunks within the document. You  
can embed an R code chunk like this:  
17  
18 ```{r cars}  
19 summary(cars)  
20 ````
```

Setup

The setup chunk is always run once before anything else

(optional) label for chunk

```
1 ---  
2 title: "Visualize Data"  
3 output:  
4   html_document:  
5     df_print: paged  
6 ---  
7  
8 ## Setup  
9  
10 The first chunk in an R Notebook is usually titled  
11 "setup," and by convention includes the R packages  
12 you want to load. Remember, in order to use an R  
13 package you have to run some `library()` code every  
14 session. Execute these lines of code to load the  
15 packages.  
16  
17 # Bechdel test data
```

```
```{r setup}  
library(ggplot2)
library(fivethirtyeight)
...
Bechdel test data
```

# ggplot2

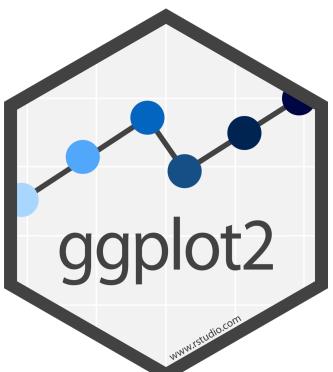


# bechdel

## 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)?

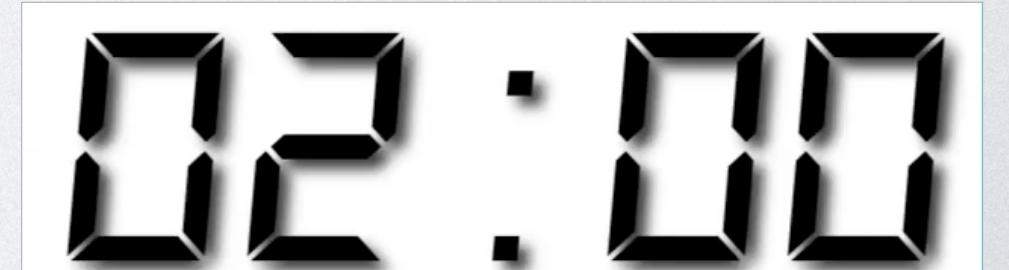


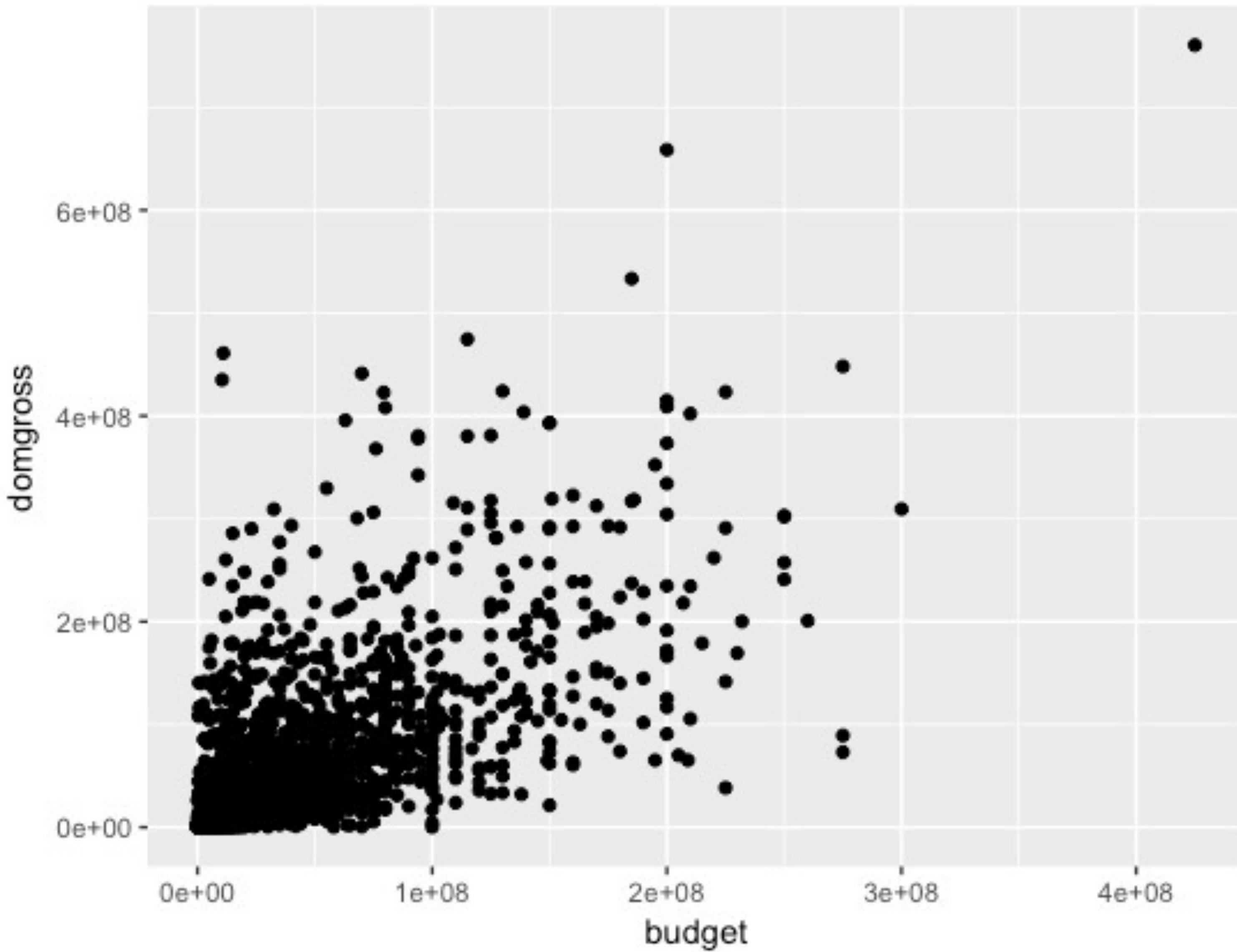
# Your Turn 1

Run this code in your document to make a graph.

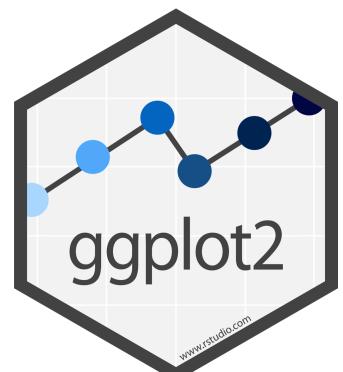
Pay strict attention to spelling, capitalization, and parentheses!

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

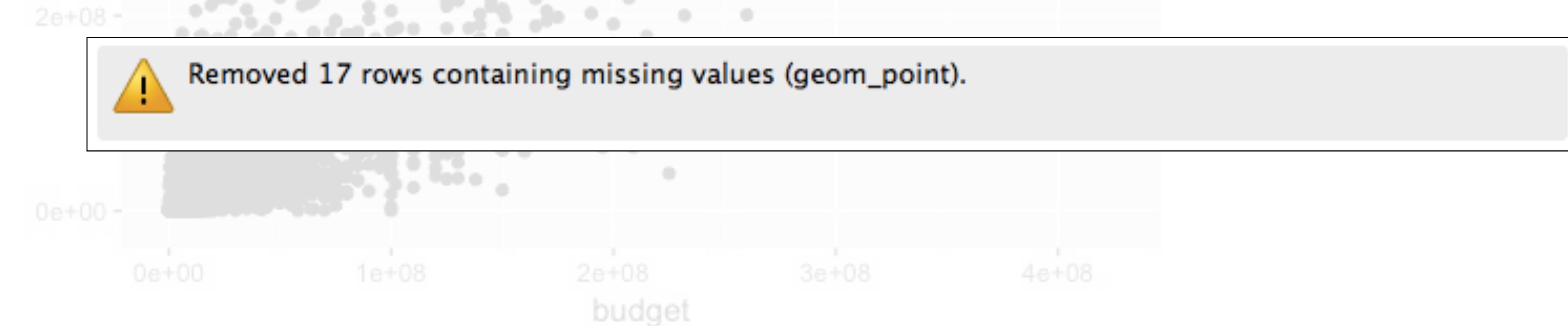




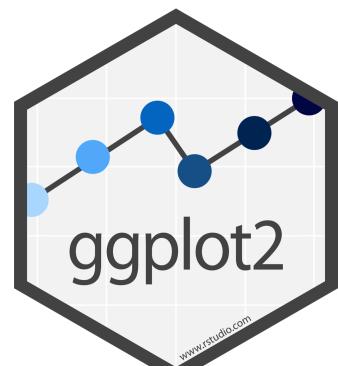
```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```



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

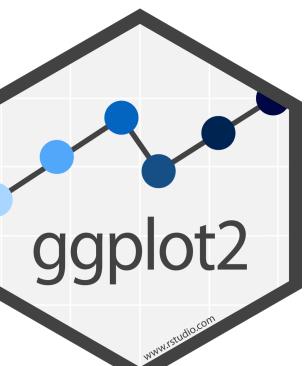


```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross))
```



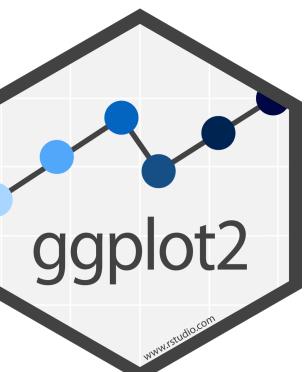
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

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



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

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



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

data

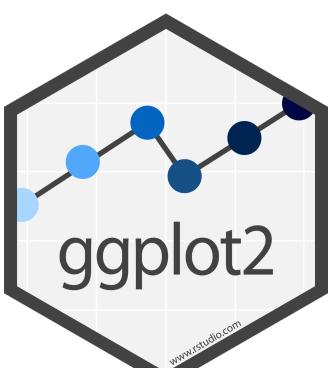
+ before new line

type of layer

aes()

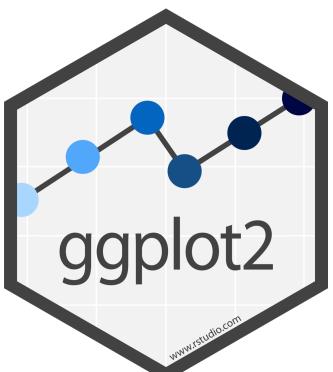
x variable

y variable



# A template

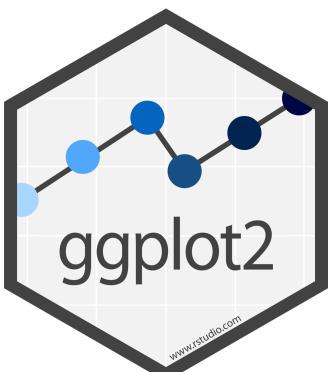
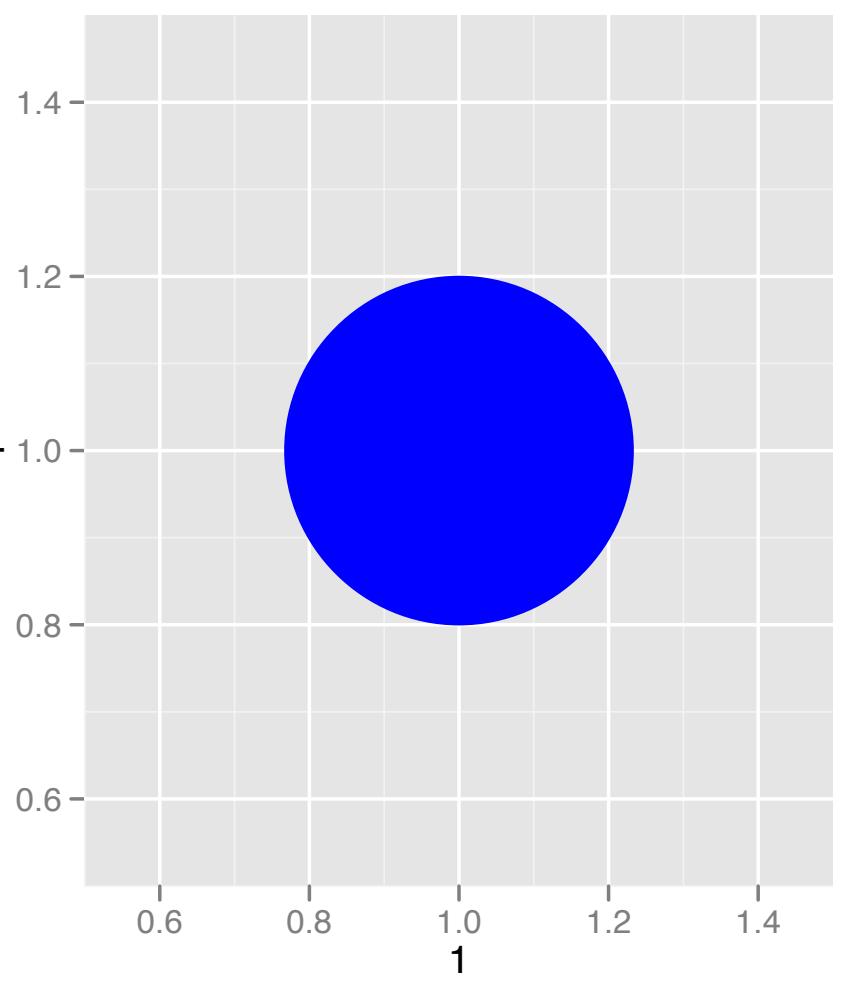
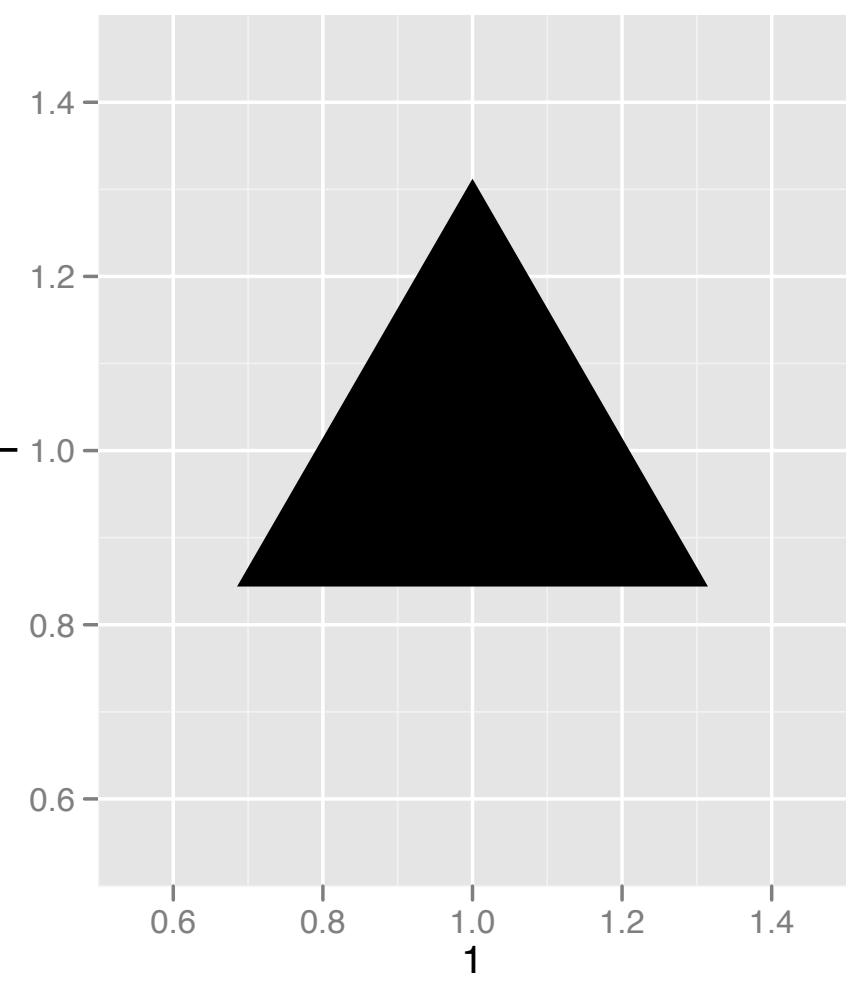
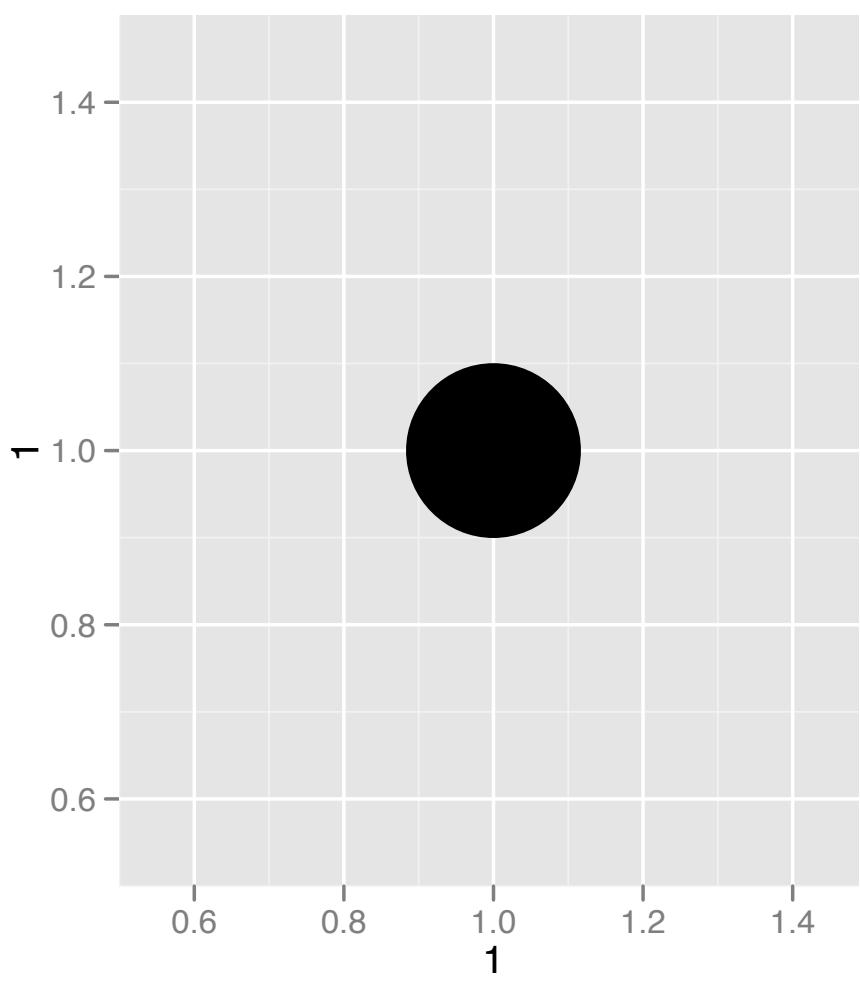
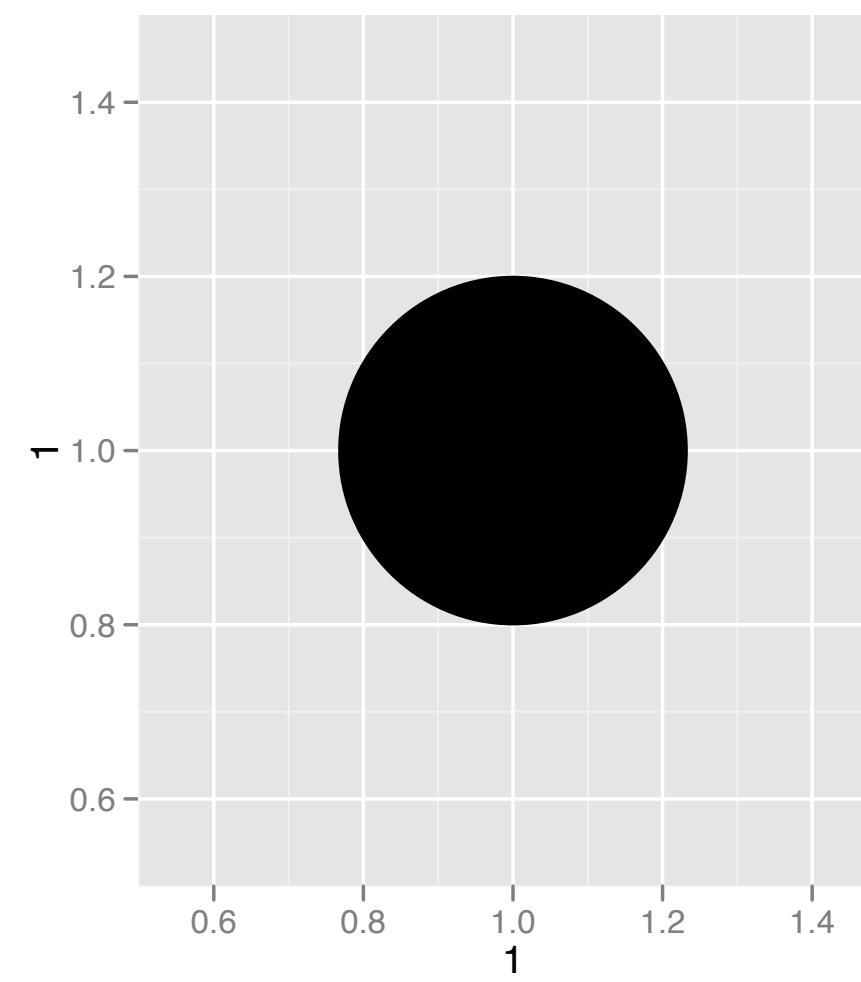
```
ggplot(data = <DATA>) +
 <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
 geom_point(mapping = aes(x = budget, y = domgross))
```



# Aesthetics

R

# Aesthetics



## Visual Space      Data Space

color ←→ clean\_test

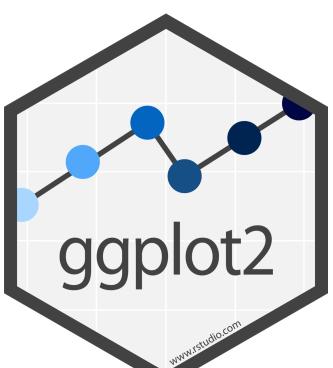
Purple ←→ nowomen

Blue ←→ notalk

Teal ←→ men

Lime ←→ dubious

Yellow ←→ ok



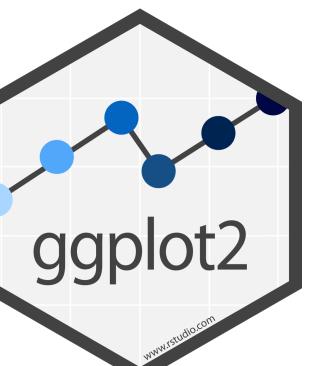
# Aesthetics

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

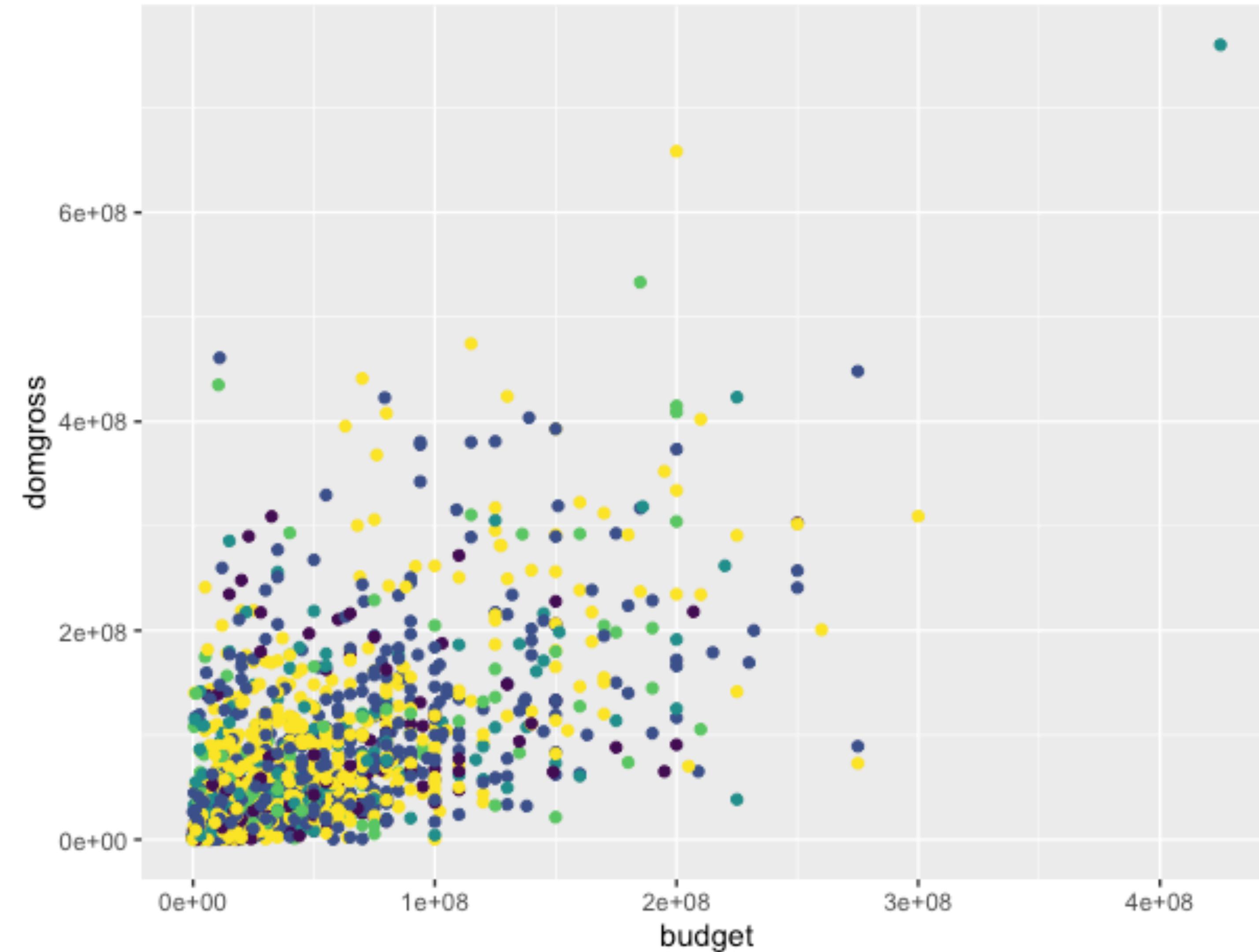
aesthetic  
property

Variable to  
map it to

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

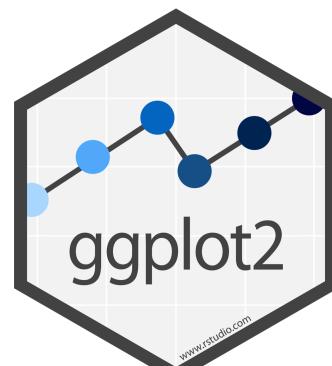


```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color = clean_test))
```



Legend added  
automatically

clean_test
nowomen
notalk
men
dubious
ok



# Your Turn 2

recall

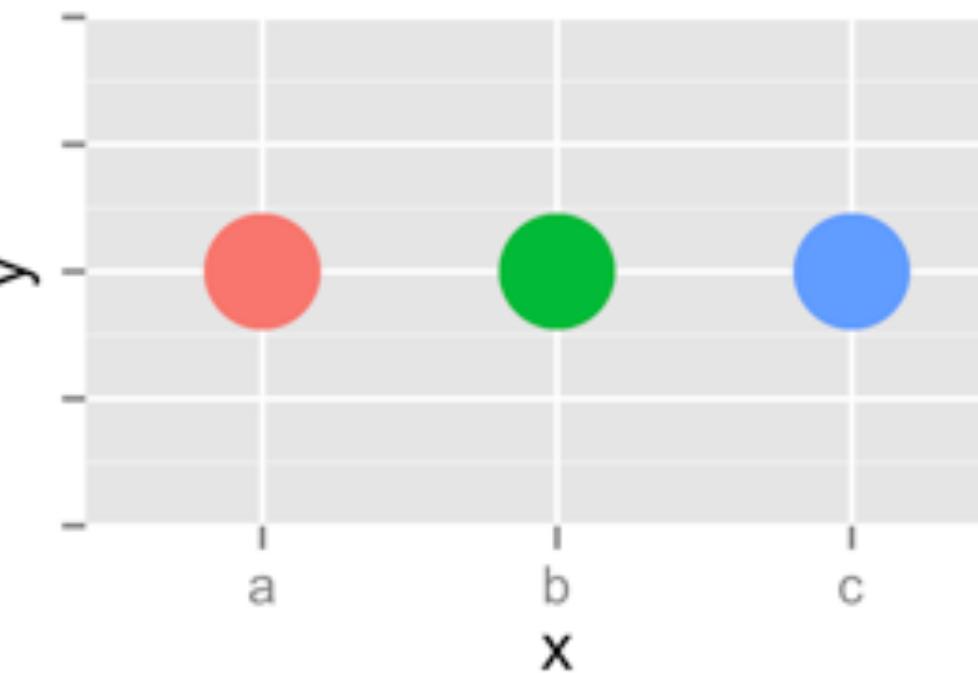
```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color = clean_test))
```

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

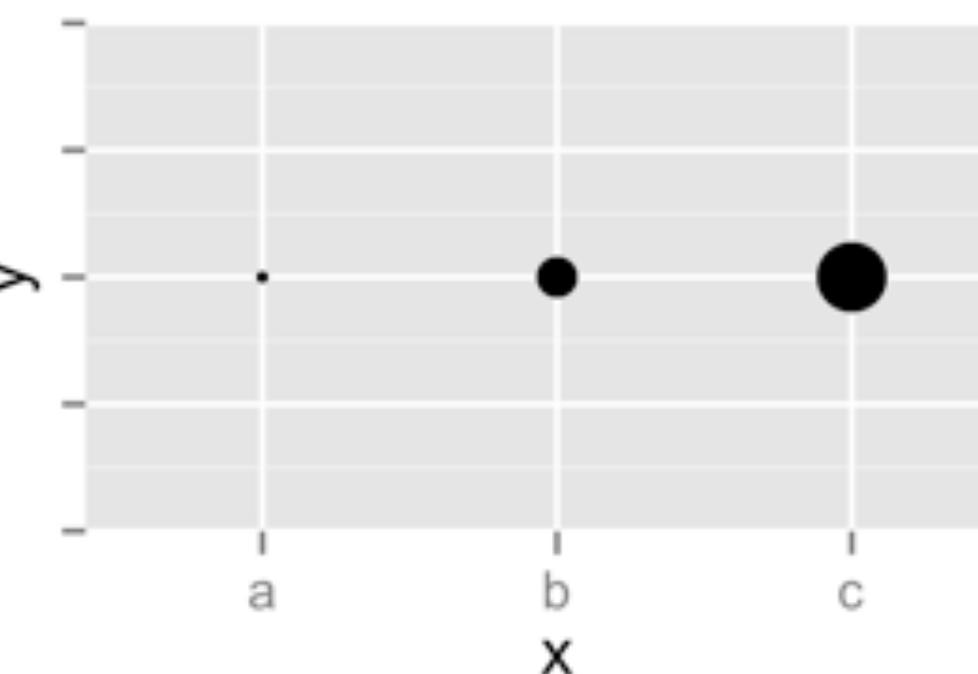
- Do different things happen when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?



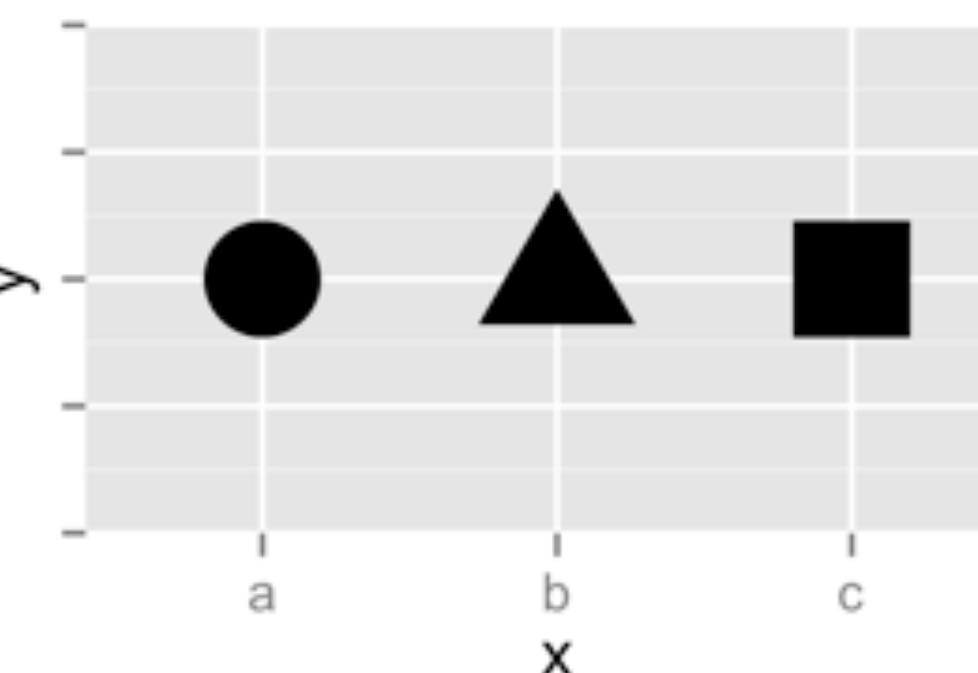
Color



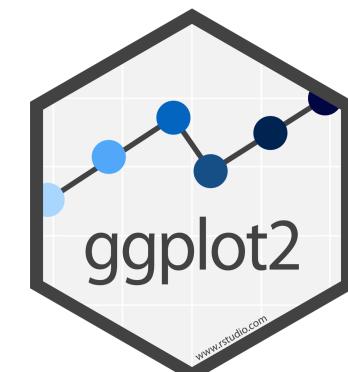
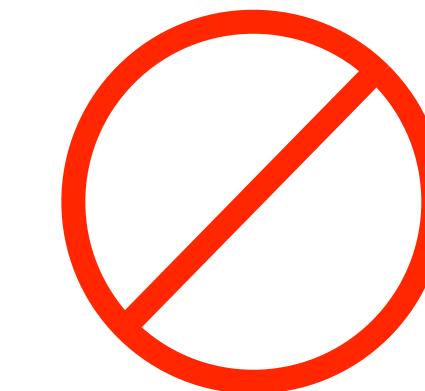
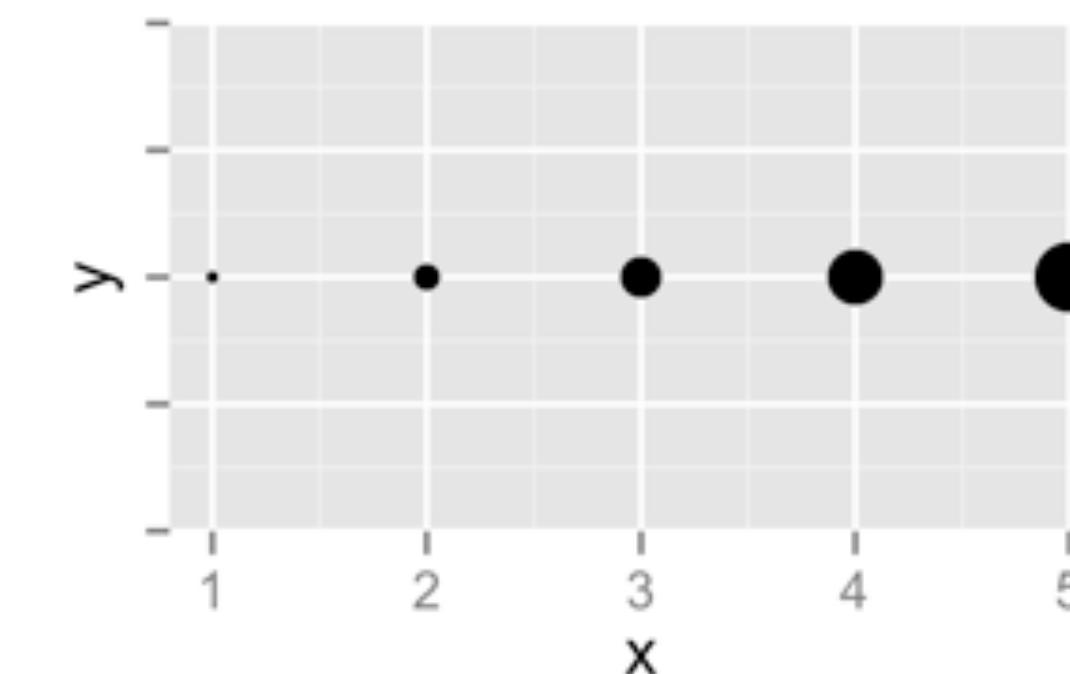
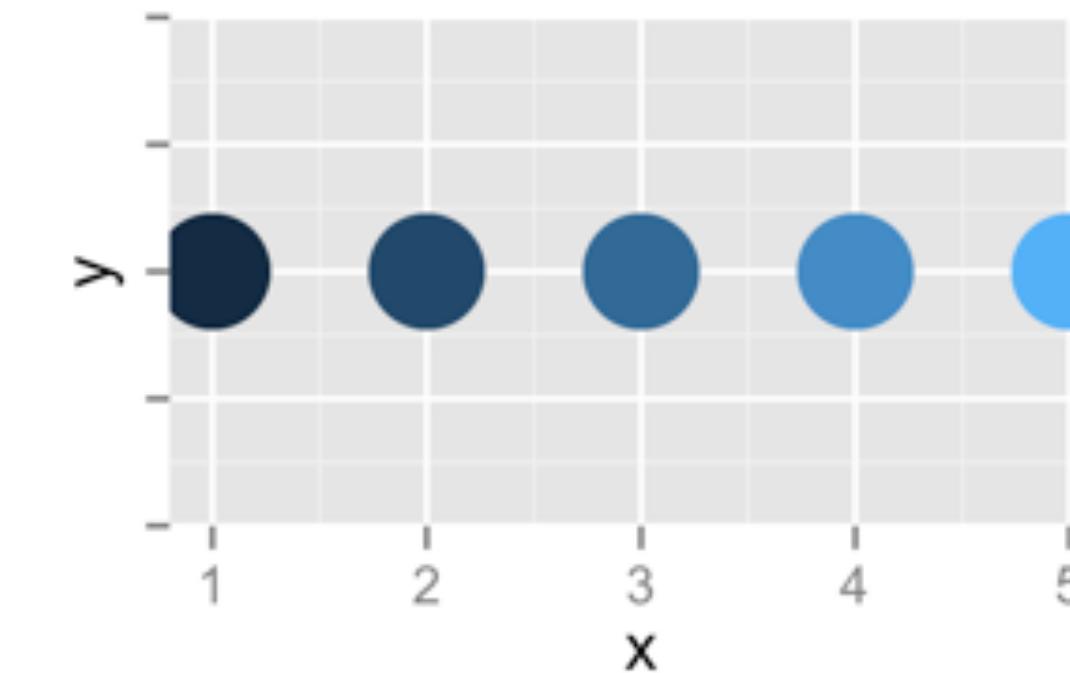
Size



Shape



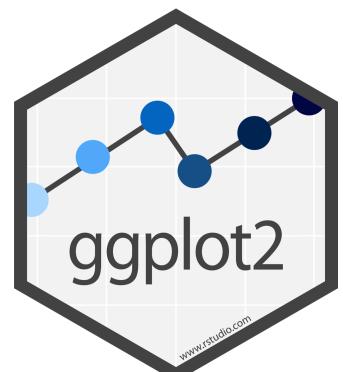
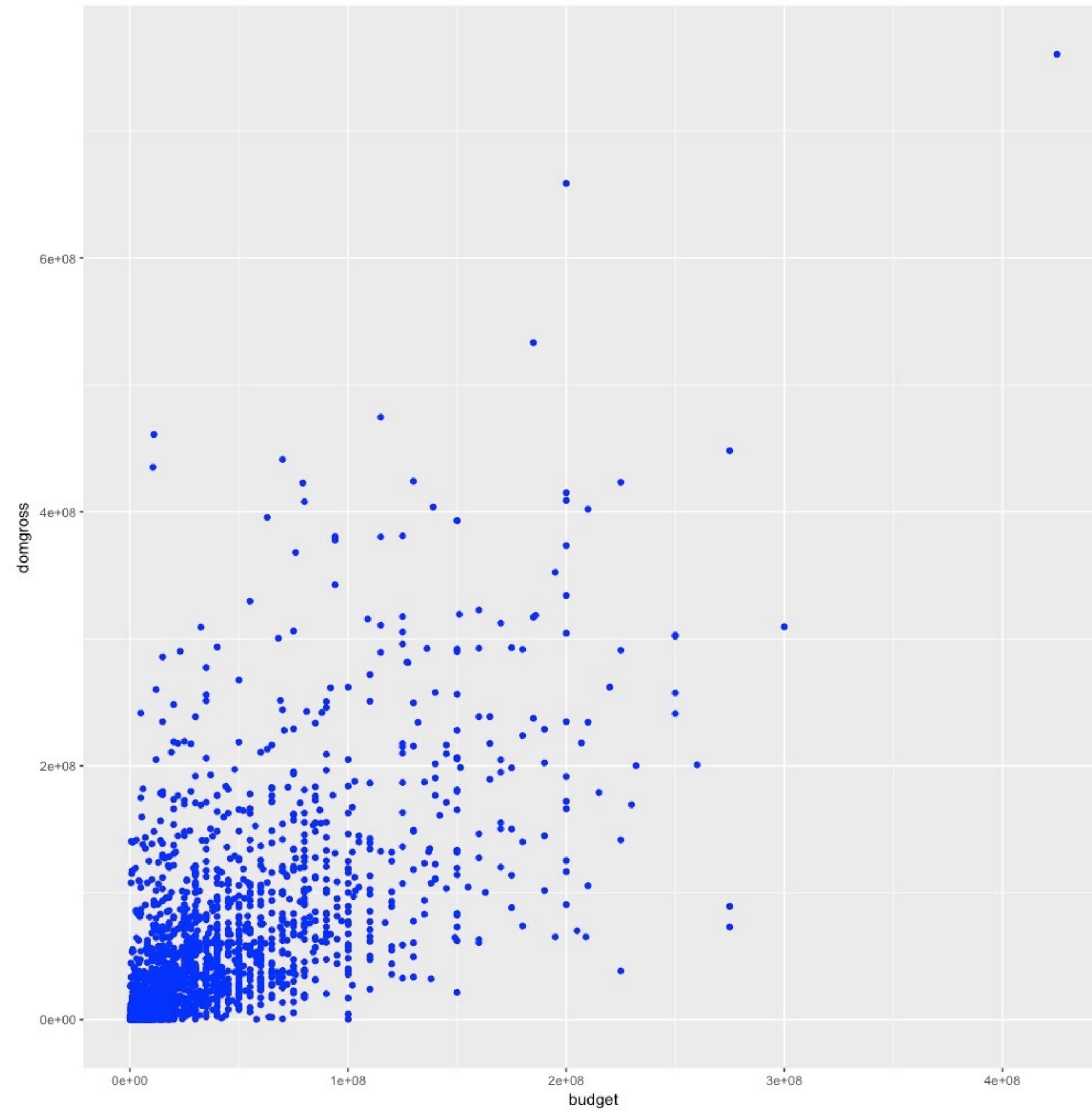
Continuous

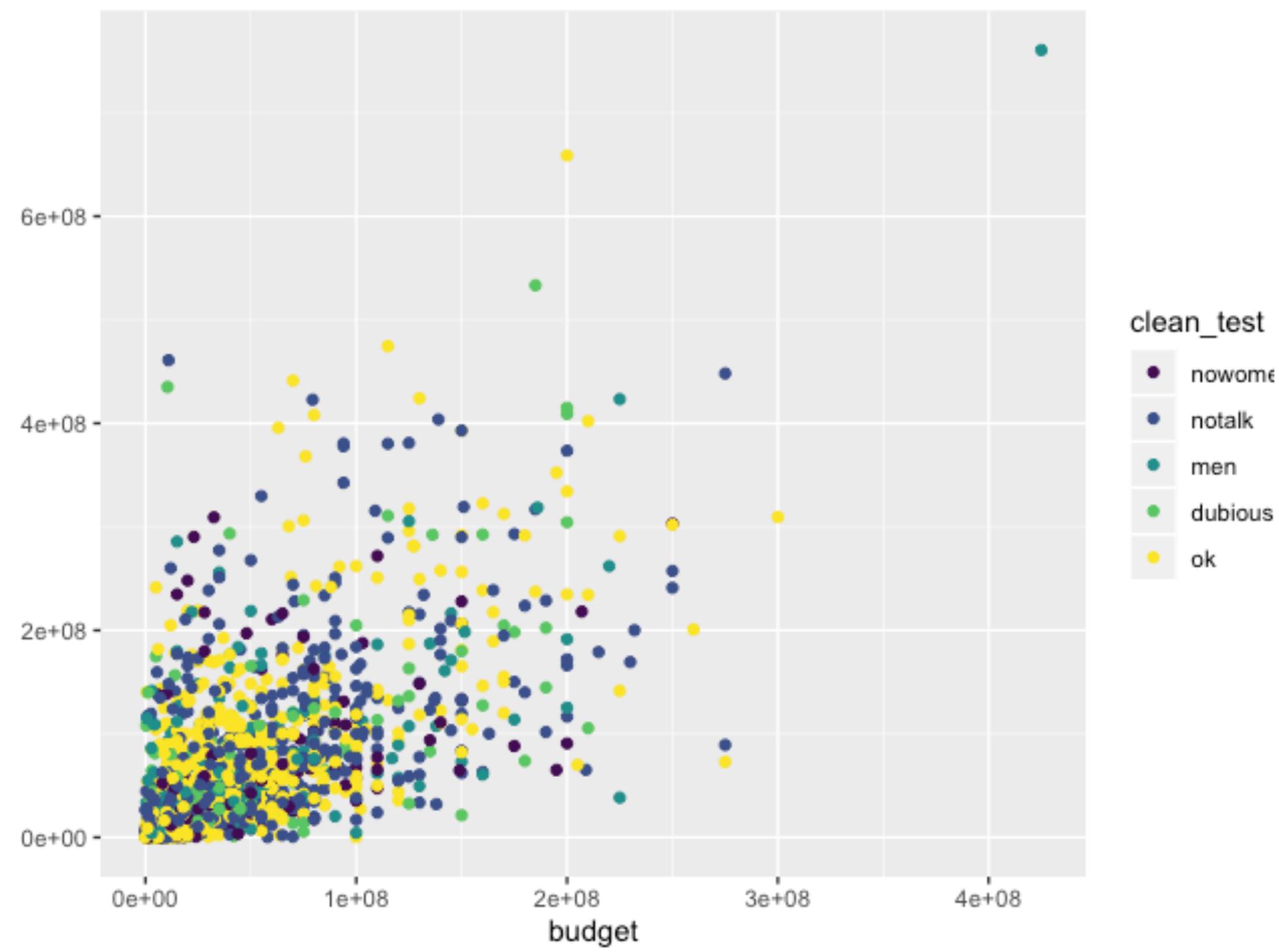


# set vs. map

A large, semi-transparent watermark of the R logo is positioned in the bottom right corner. The logo consists of a circular emblem with the letters "R" inside.

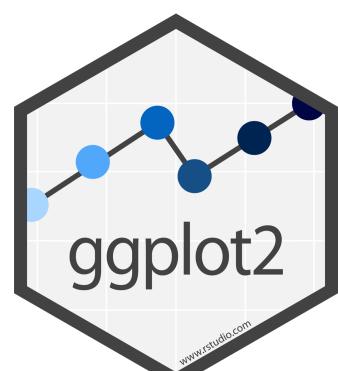
# How would you make this plot?



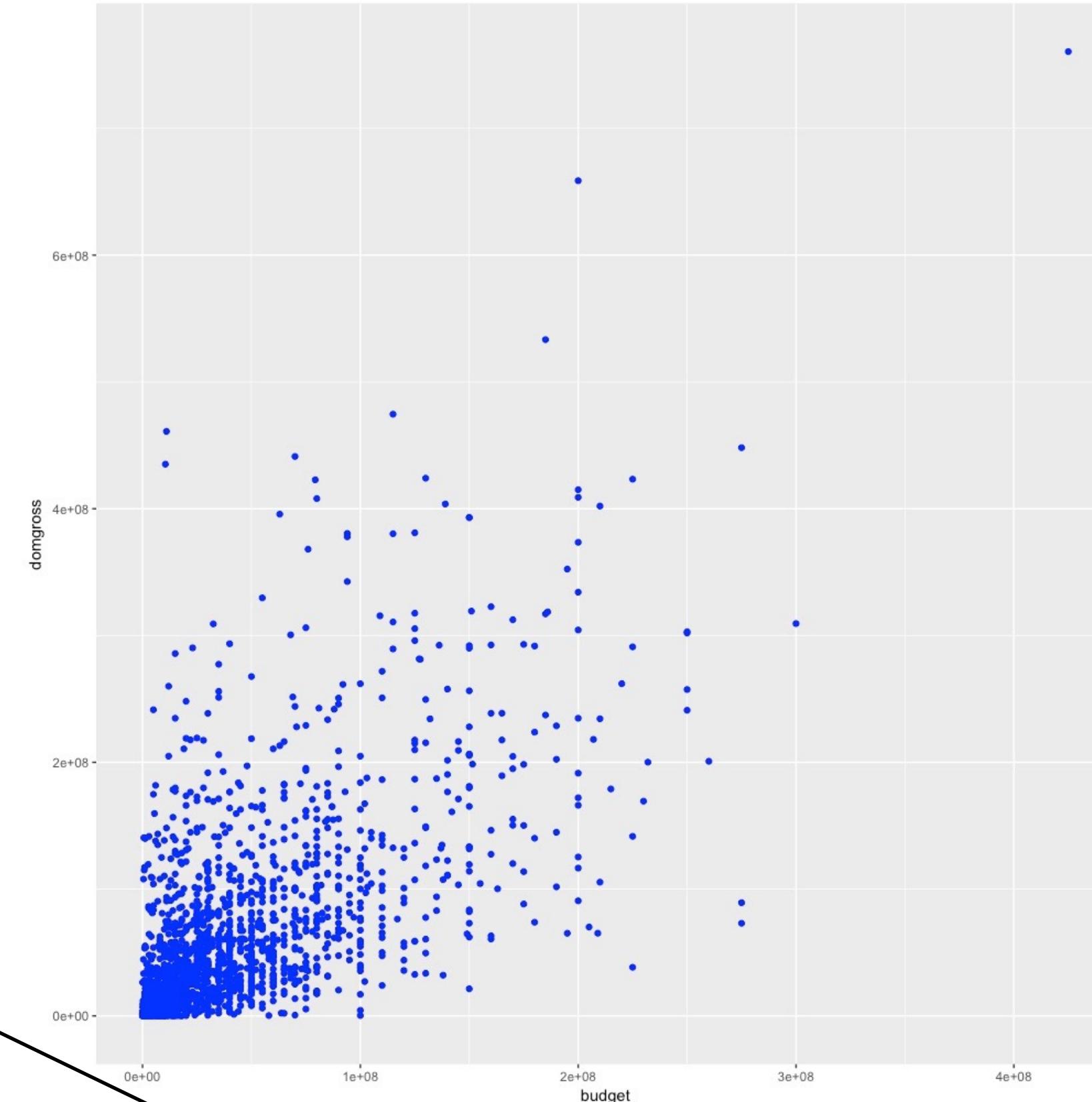


**Inside of aes(): maps an aesthetic to a variable**

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

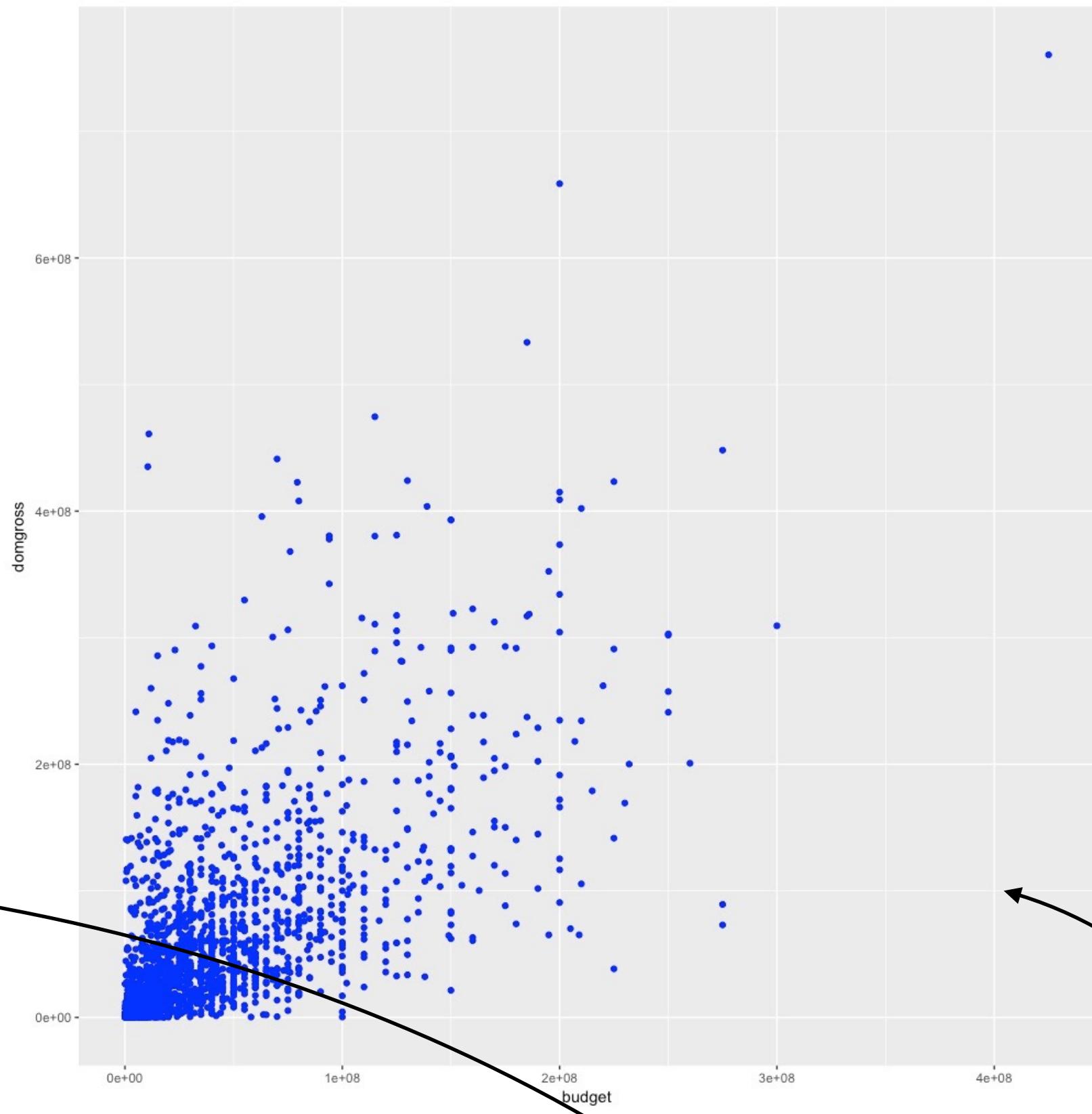
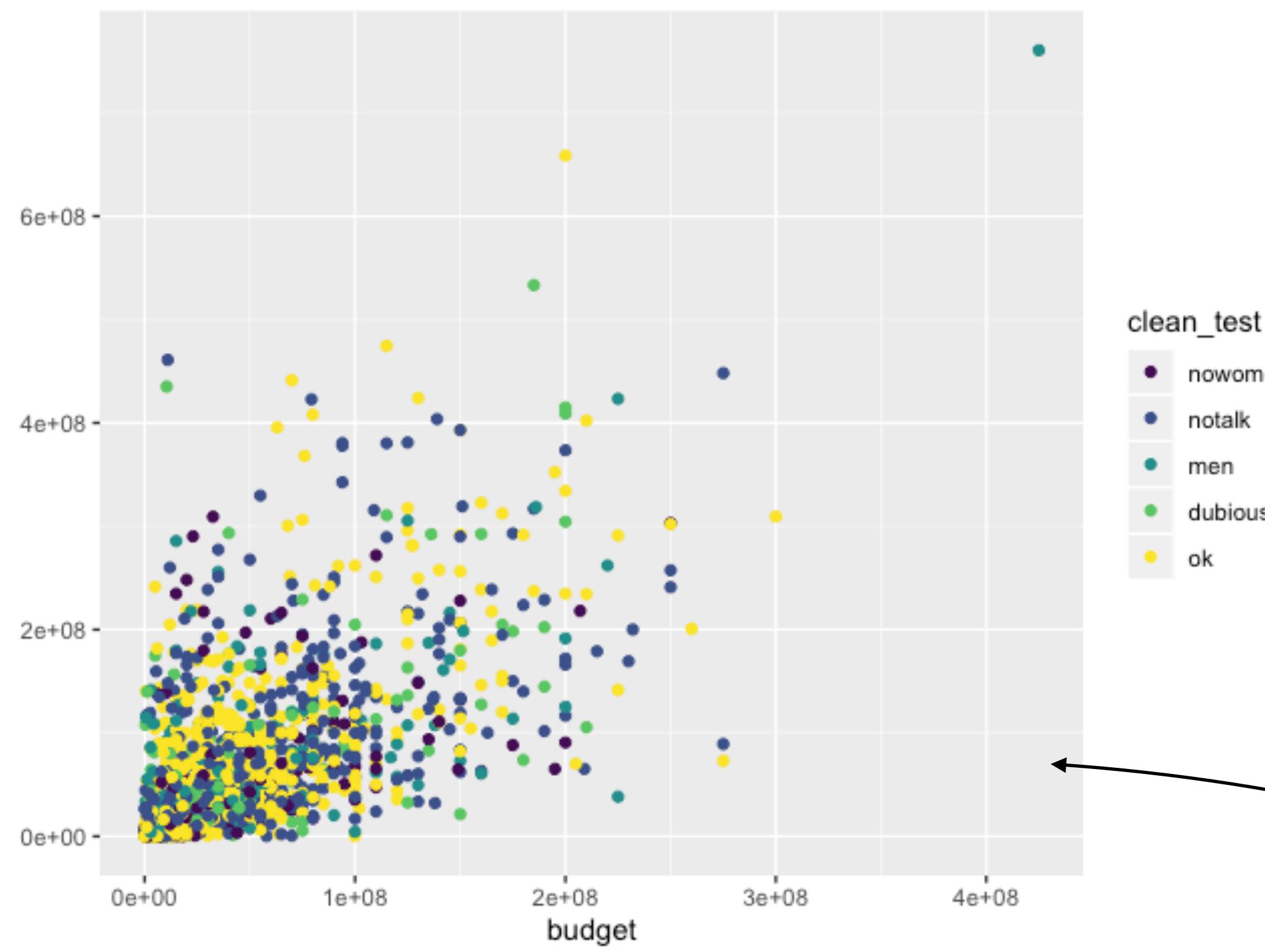


**Outside of aes():** sets  
an aesthetic to a value



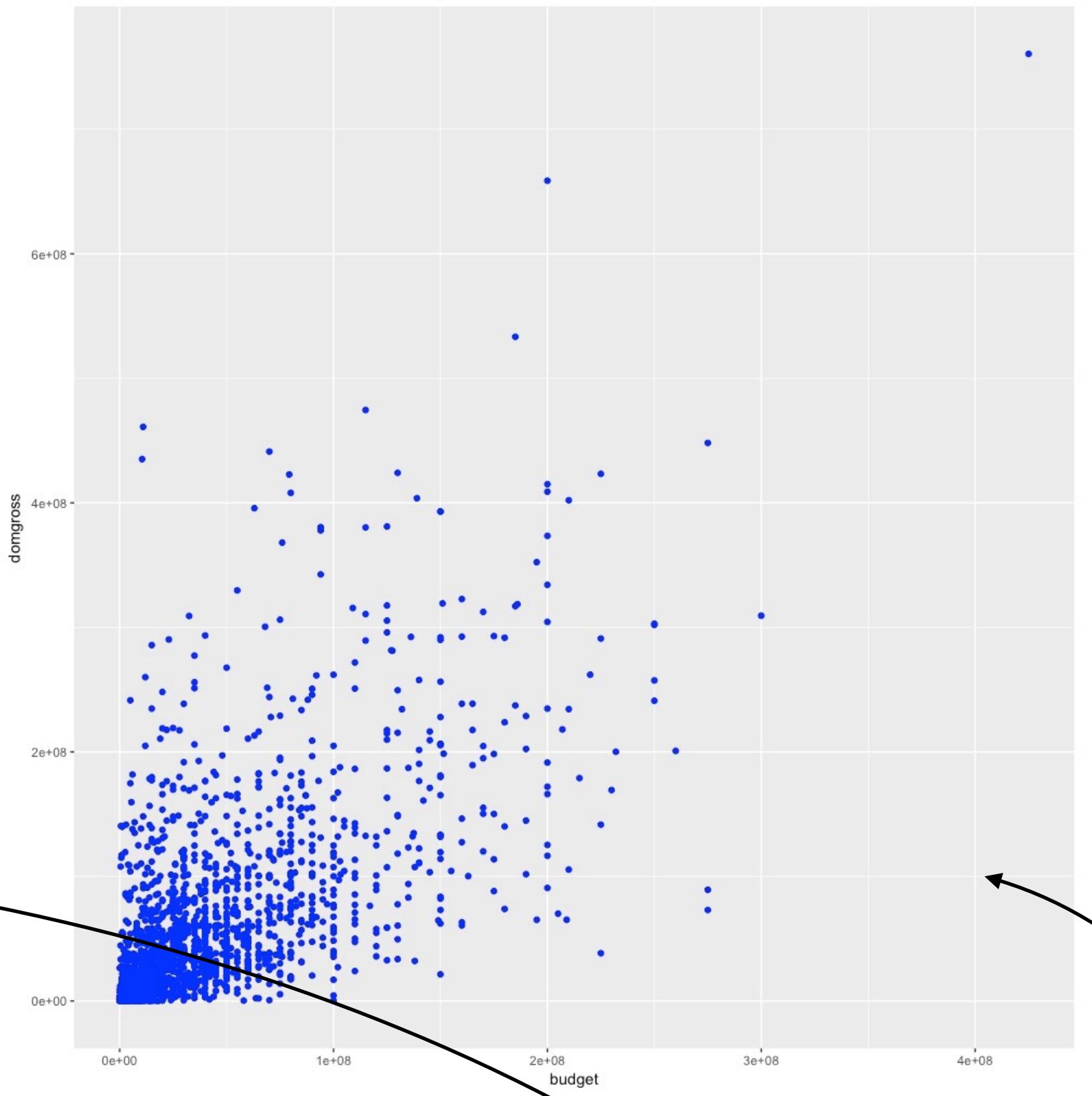
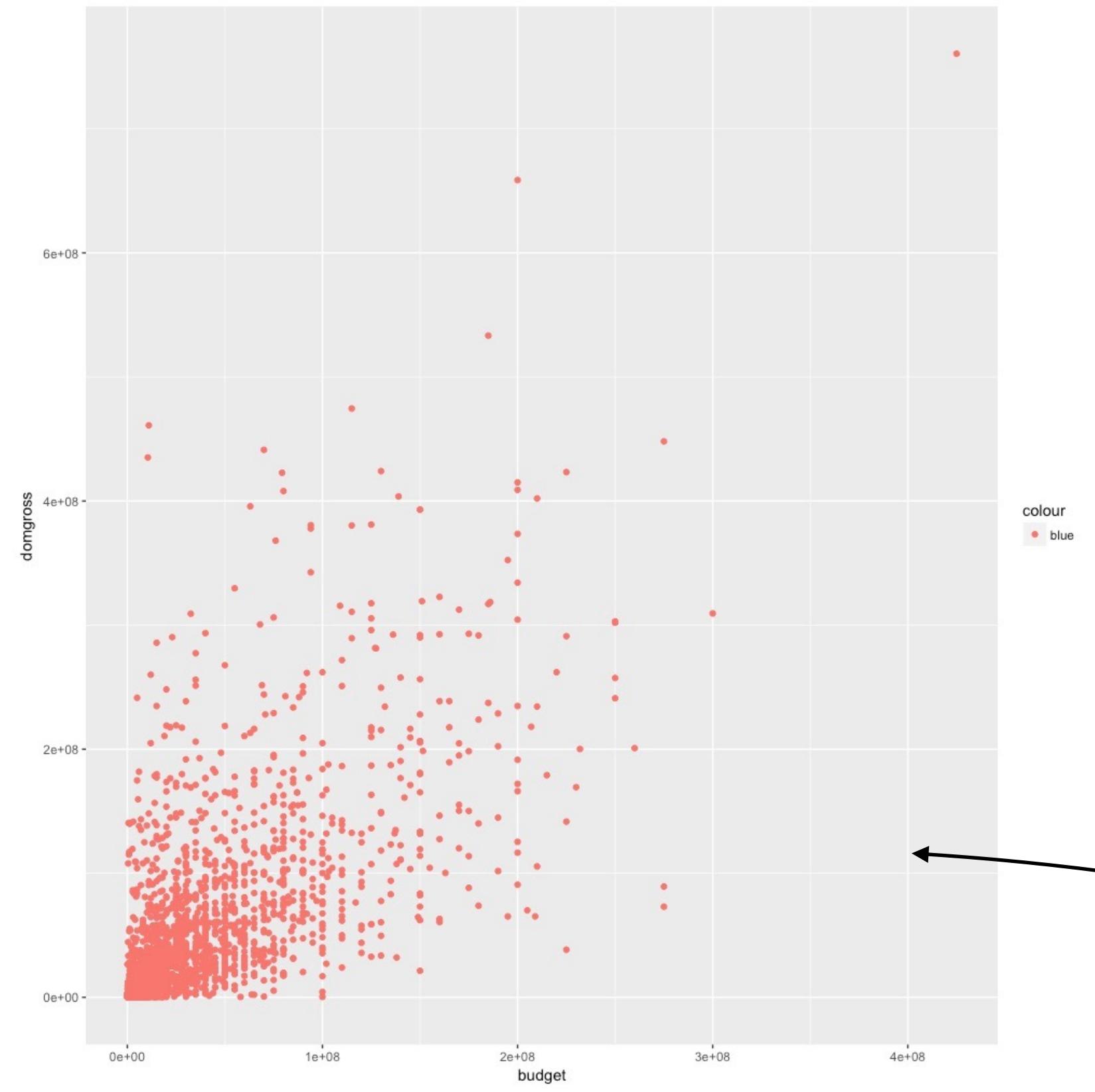
```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

```
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"))
```

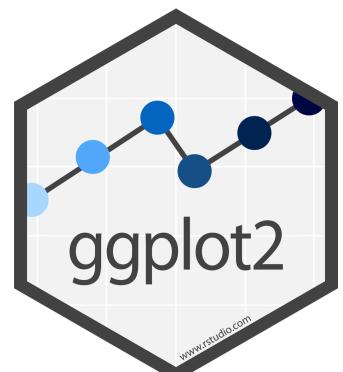
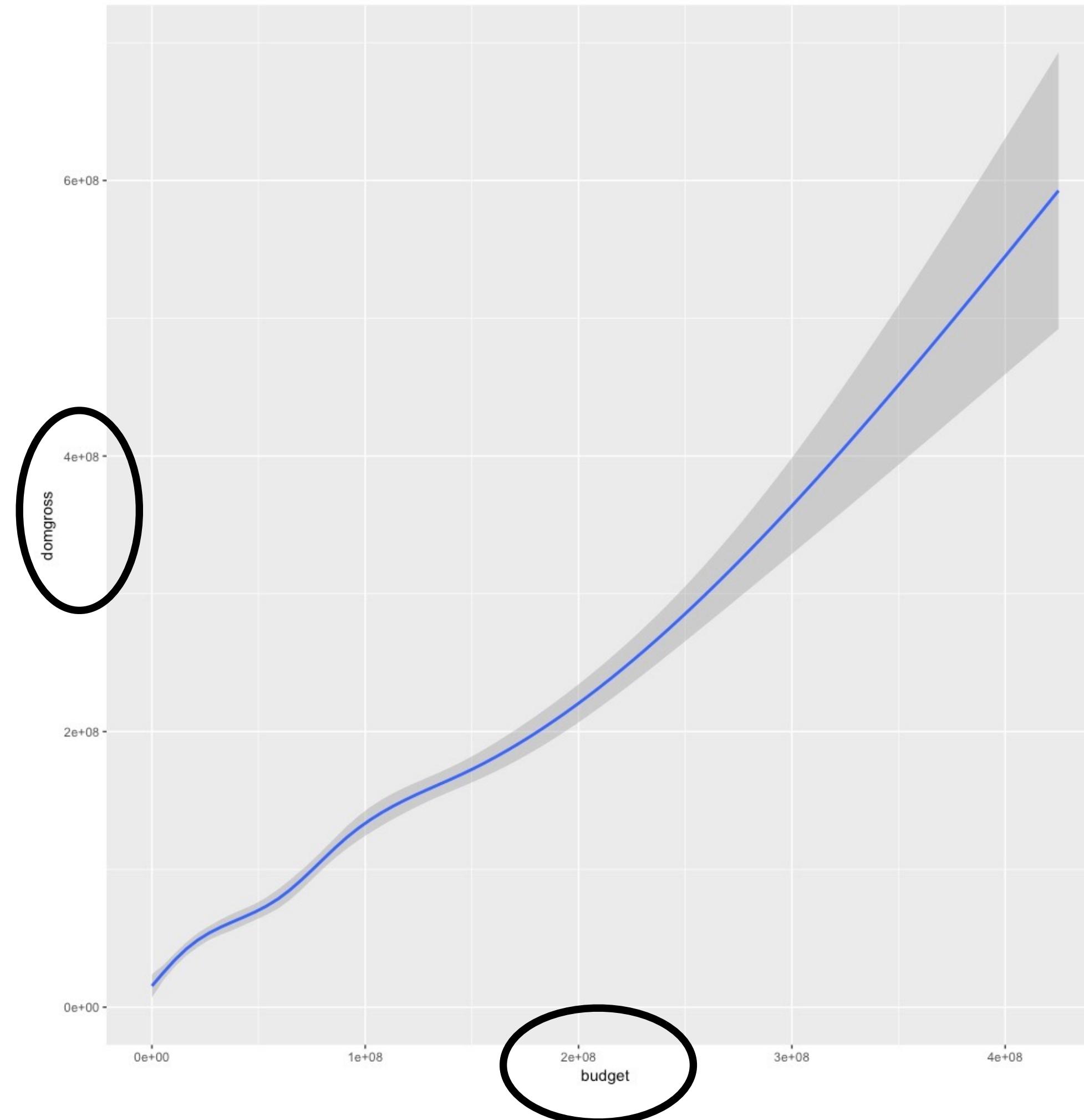
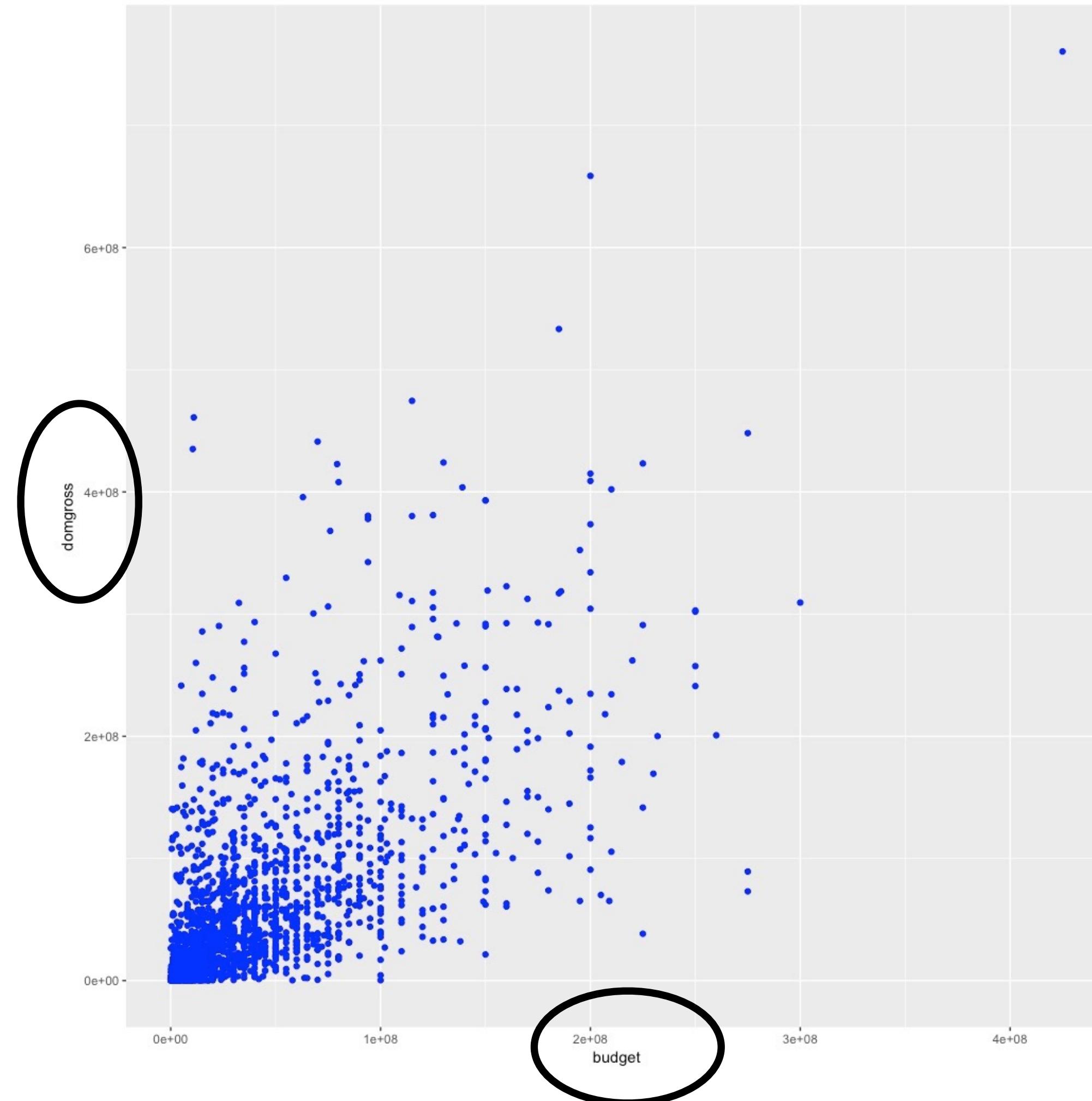
```
ggplot(bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

# geoms



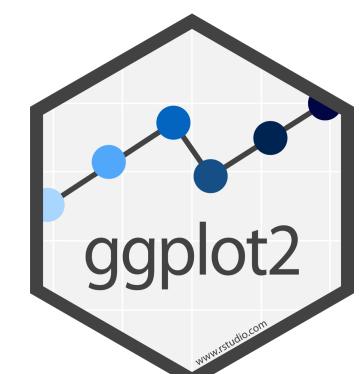
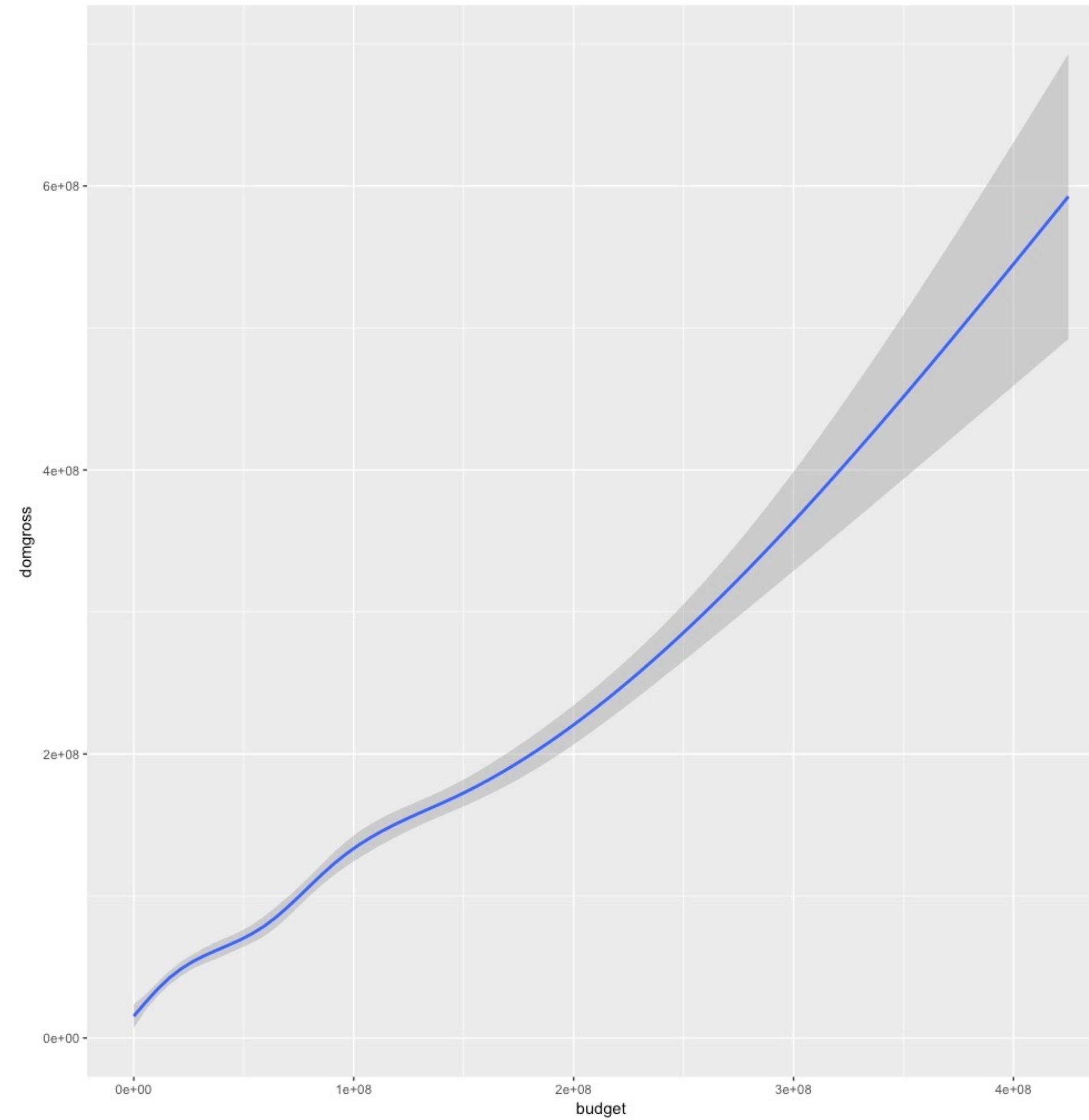
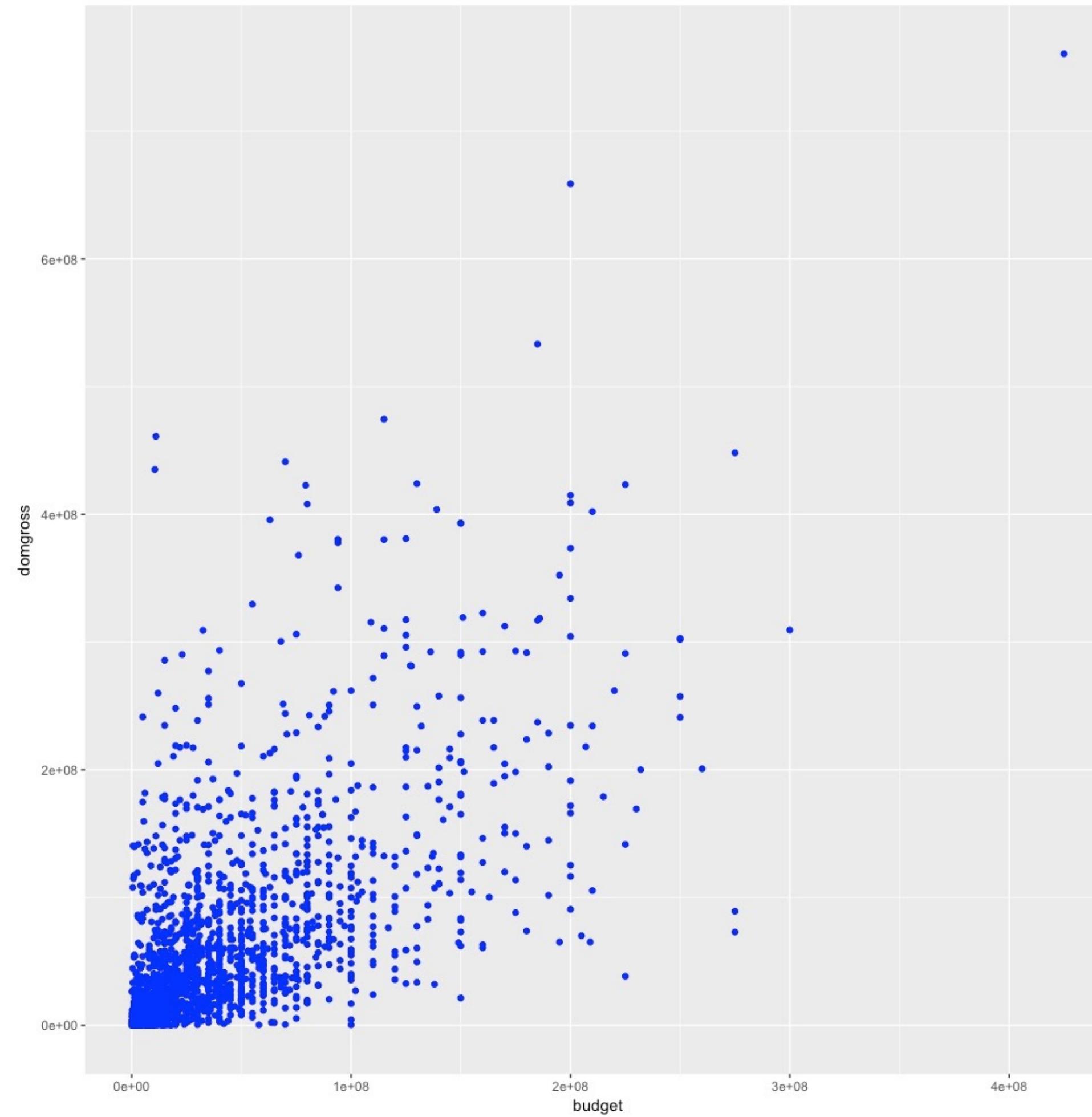
How are these plots similar?

Same: x var , y var , data



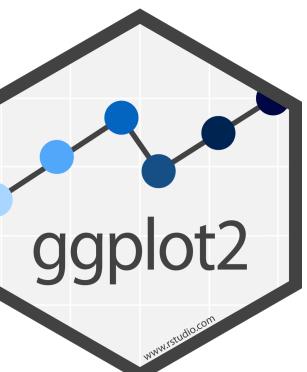
How are these plots different?

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



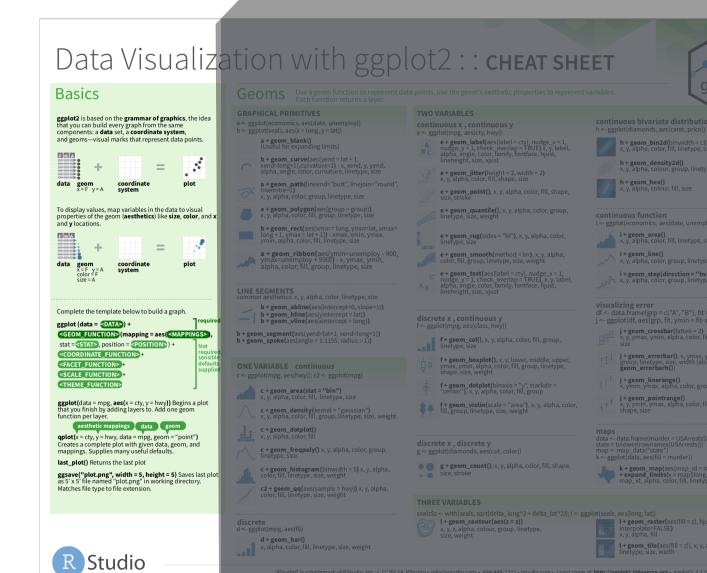
# geoms

```
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.

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

**TWO VARIABLES**

**continuous x , continuous y**

```
e <- ggplot(mpg, aes(cty, hwy))
```

- e + geom\_label(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- e + geom\_jitter(height = 2, width = 2)** x, y, alpha, color, fill, shape, size
- e + geom\_point()** x, y, alpha, color, fill, shape, size, stroke
- e + geom\_quantile()** x, y, alpha, color, group, linetype, size, weight
- e + geom\_rug(sides = "bl")** x, y, alpha, color, linetype, size
- e + geom\_smooth(method = lm)** x, y, alpha, color, fill, group, linetype, size, weight
- e + geom\_text(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

**LINE SEGMENTS** common aesthetics: x, y, alpha, color, linetype, size

```
b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(intercept = 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 x , continuous y**

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

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

**discrete x , discrete y**

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

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

**THREE VARIABLES**

```
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

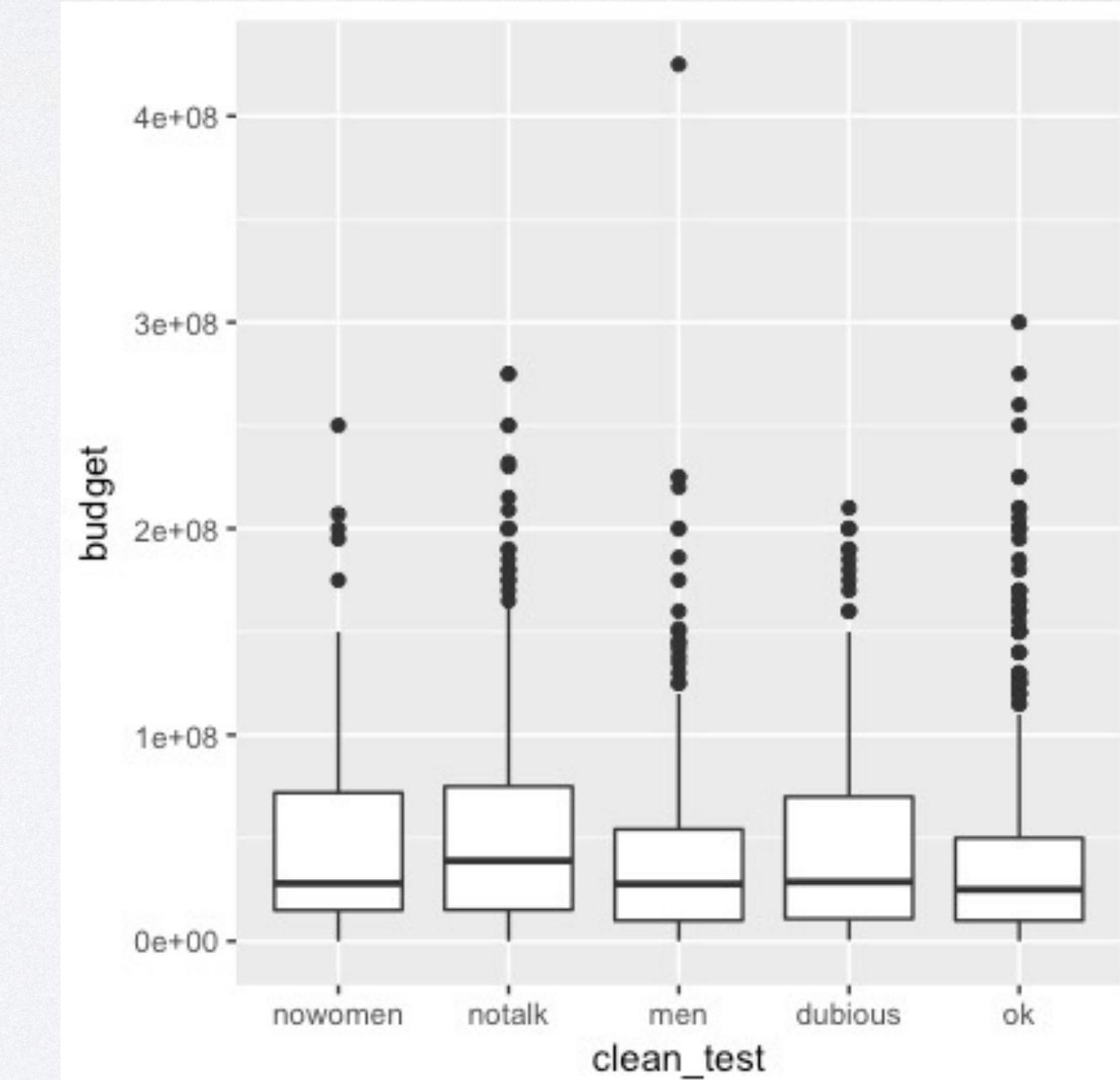
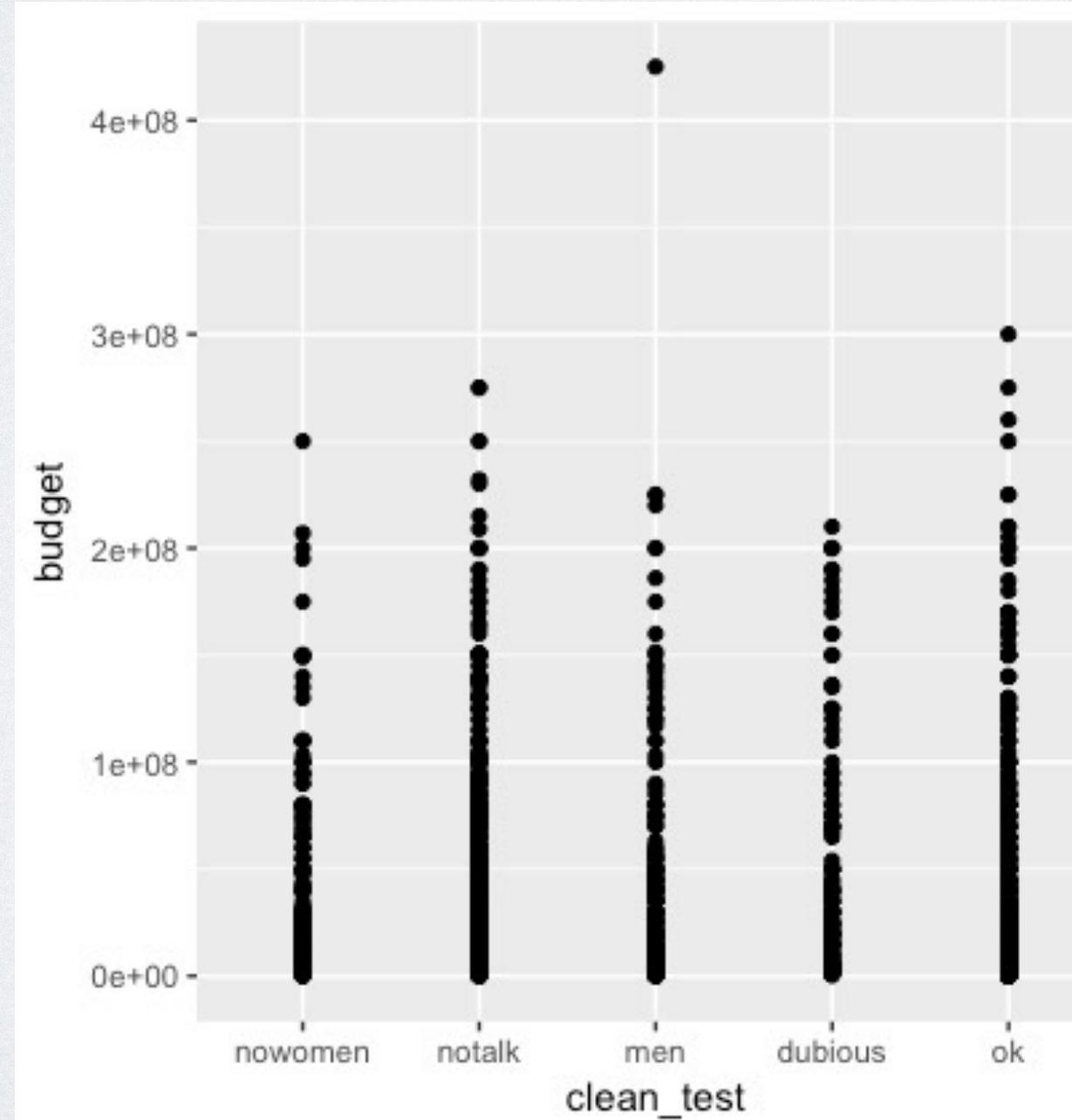
**ggplot2**

# Your Turn 3

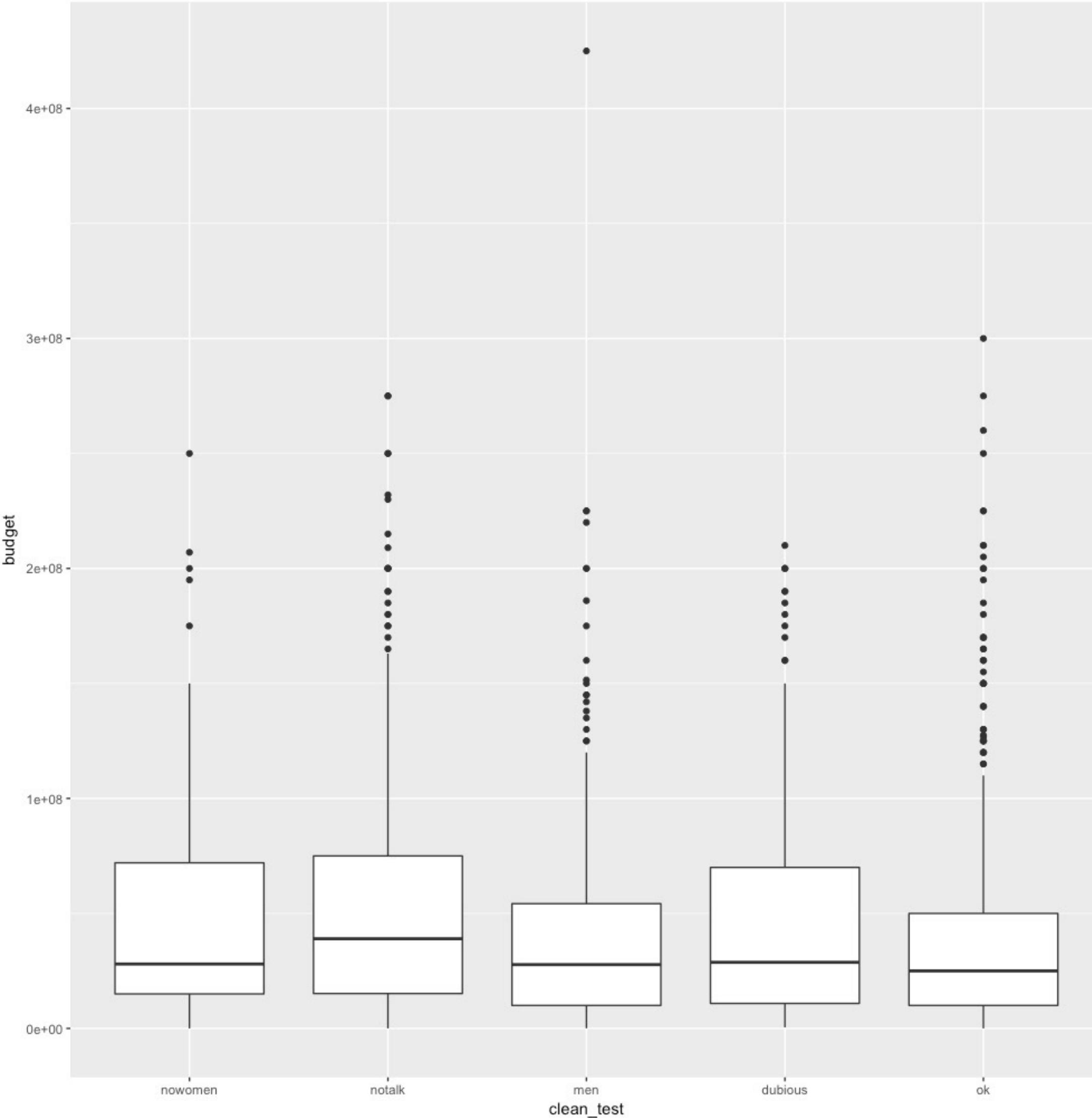
**recall**

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

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



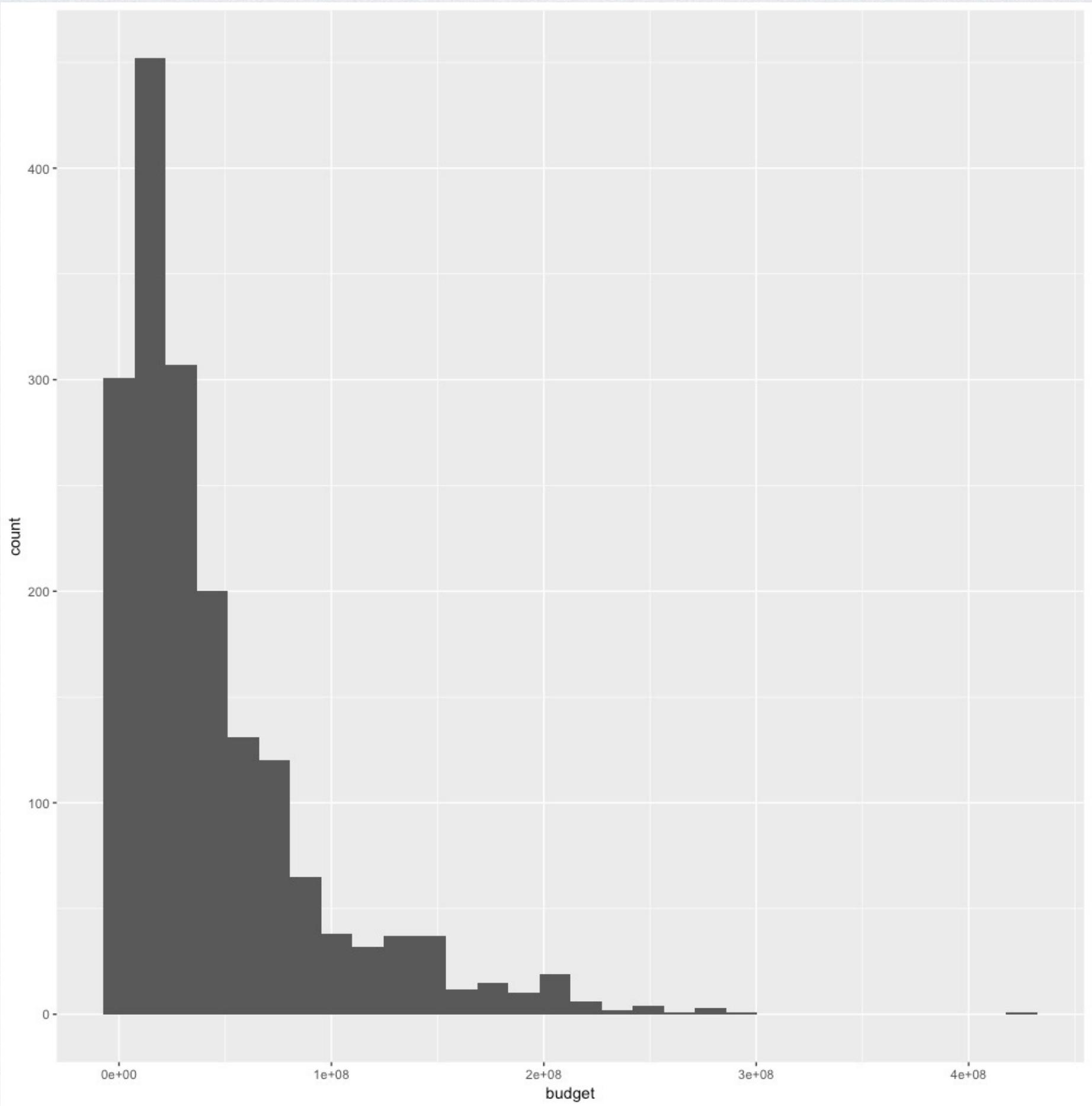
02 : 00

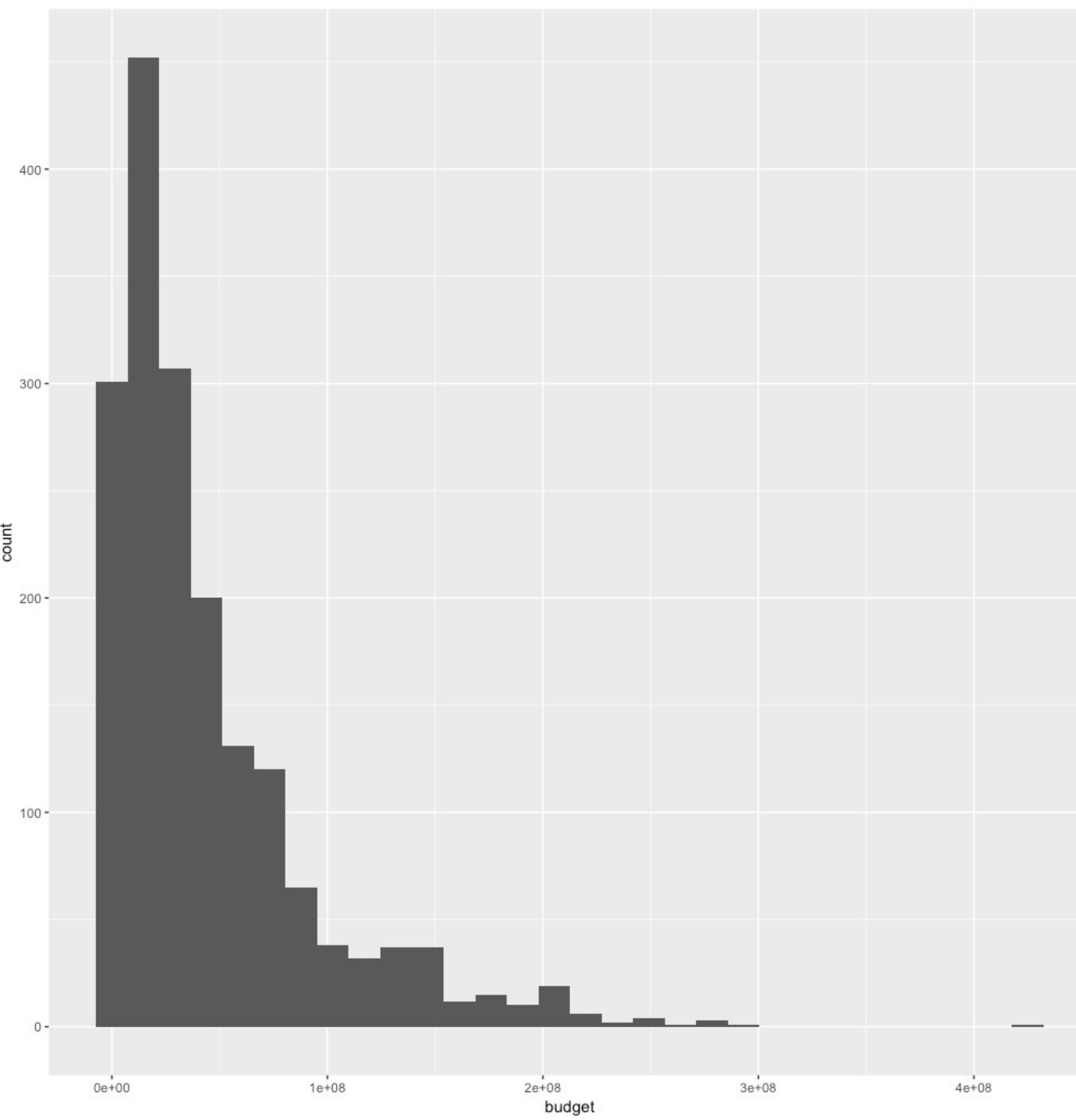


```
ggplot(data = bechdel) +
 geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

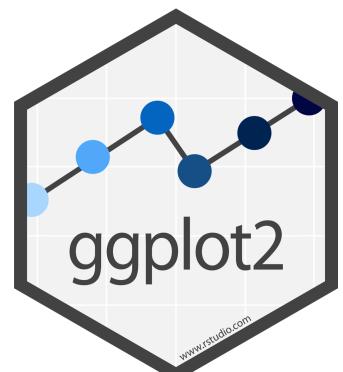
# Your Turn 4

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

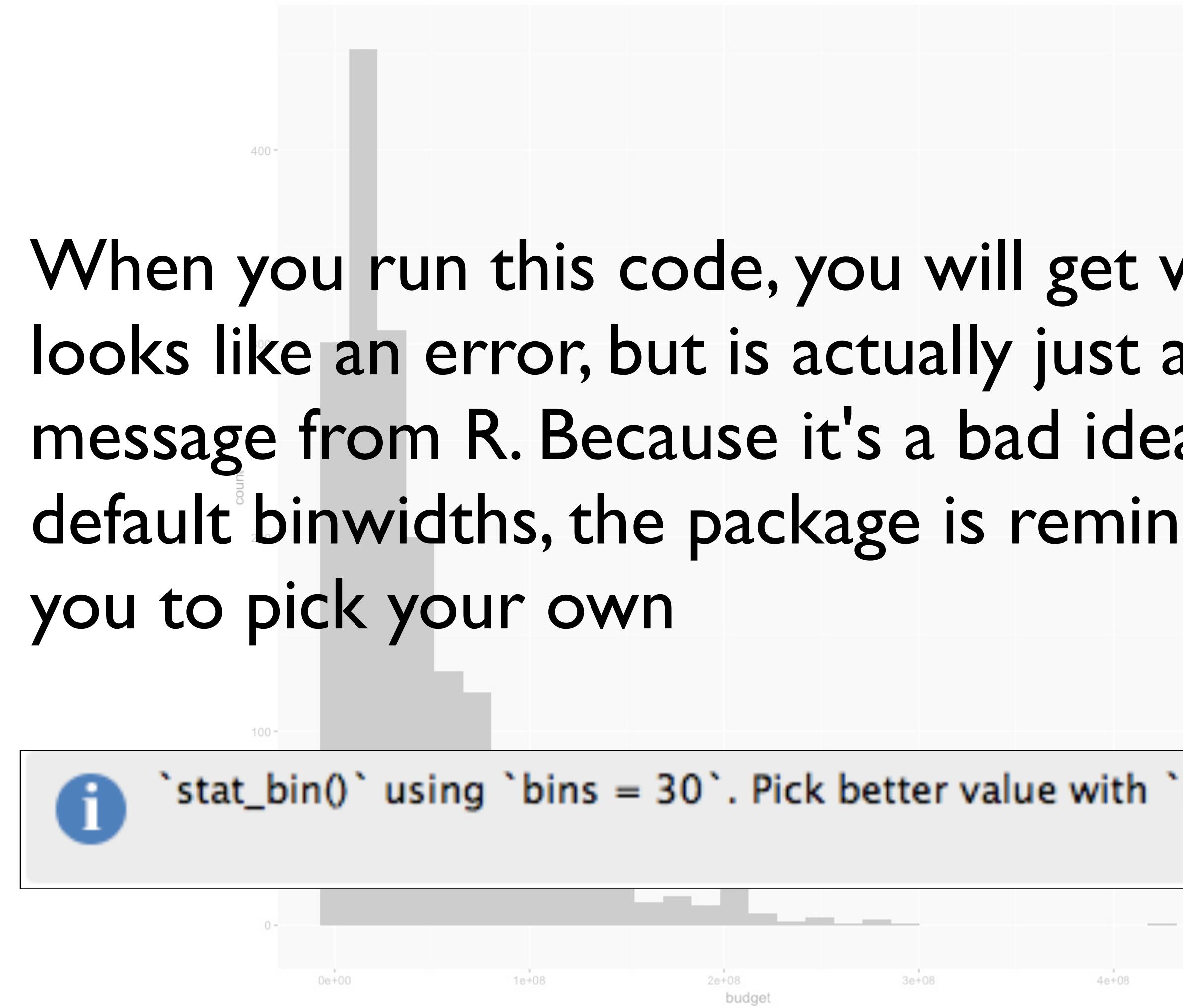




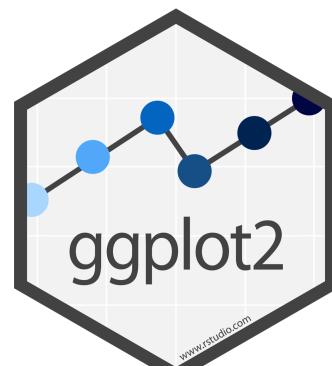
```
ggplot(data = bechdel) +
 geom_histogram(mapping = aes(x = budget))
```



When you run this code, you will get what looks like an error, but is actually just a message from R. Because it's a bad idea to use default binwidths, the package is reminding you to pick your own



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

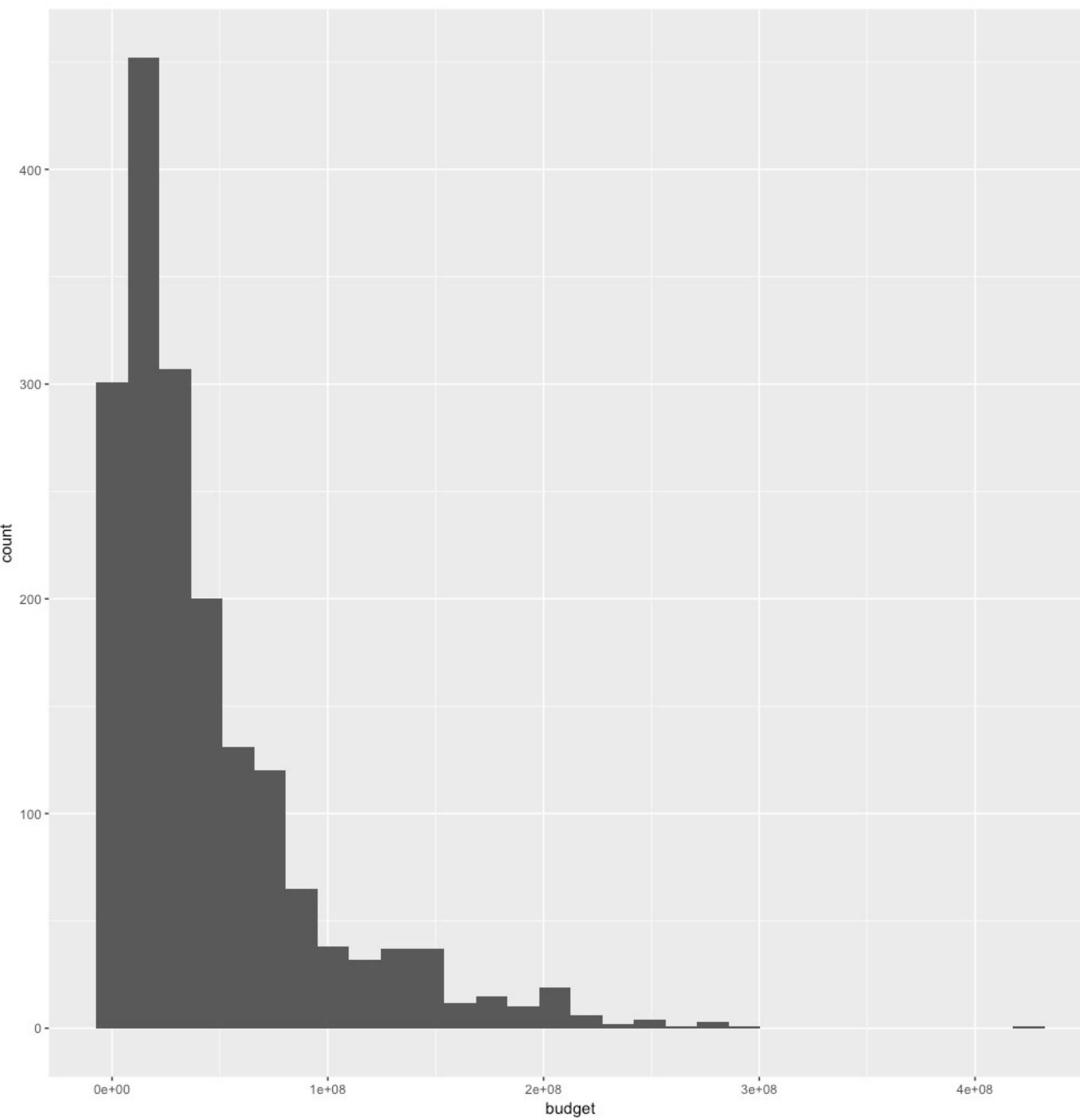


# Your Turn 5

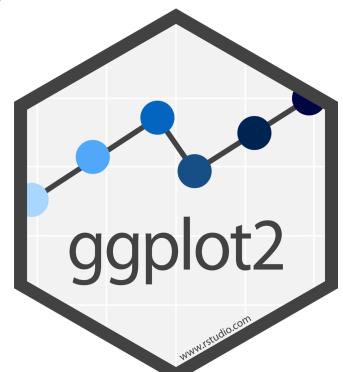
What would be a reasonable  
binwidth for budget?

Try it out



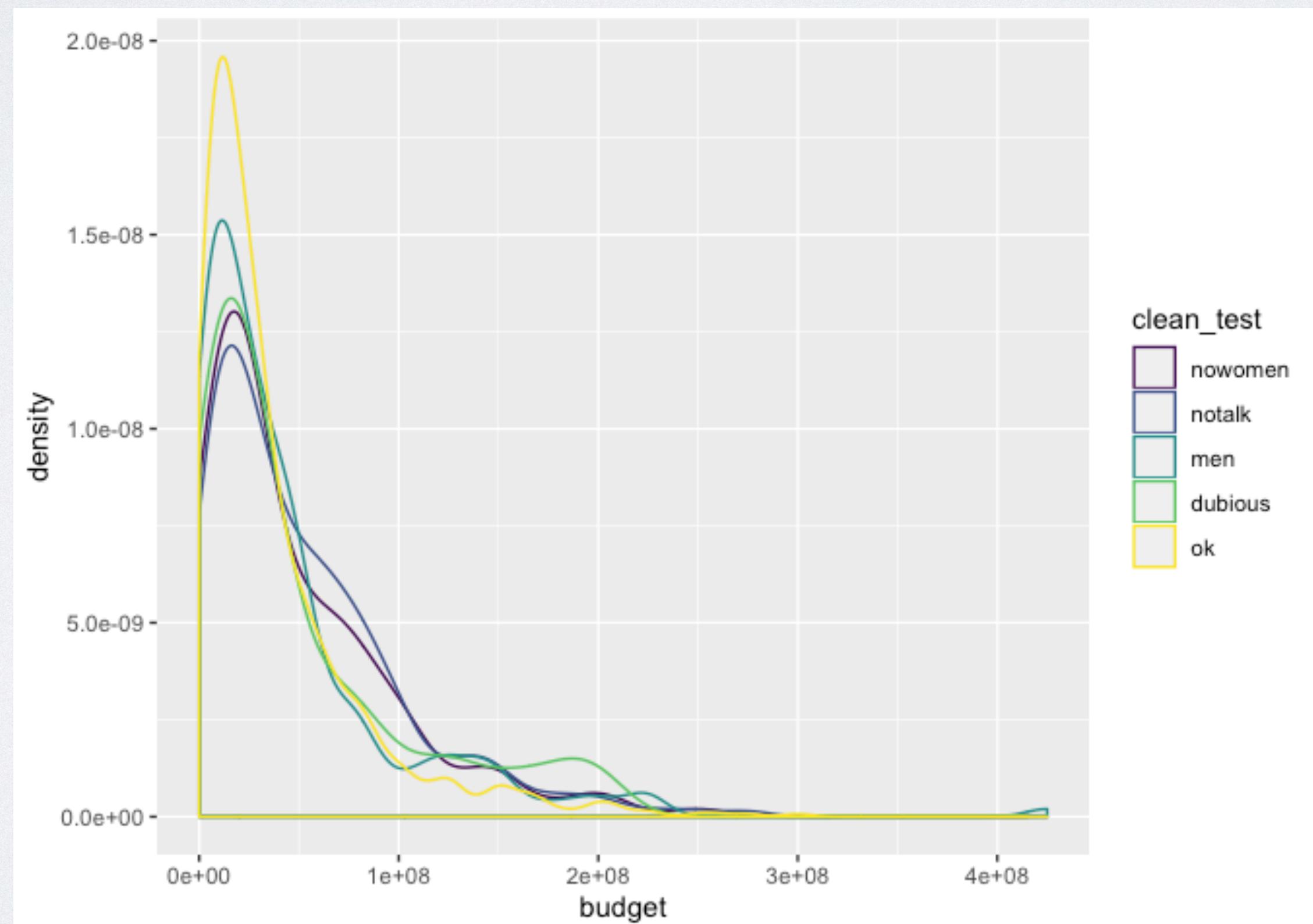


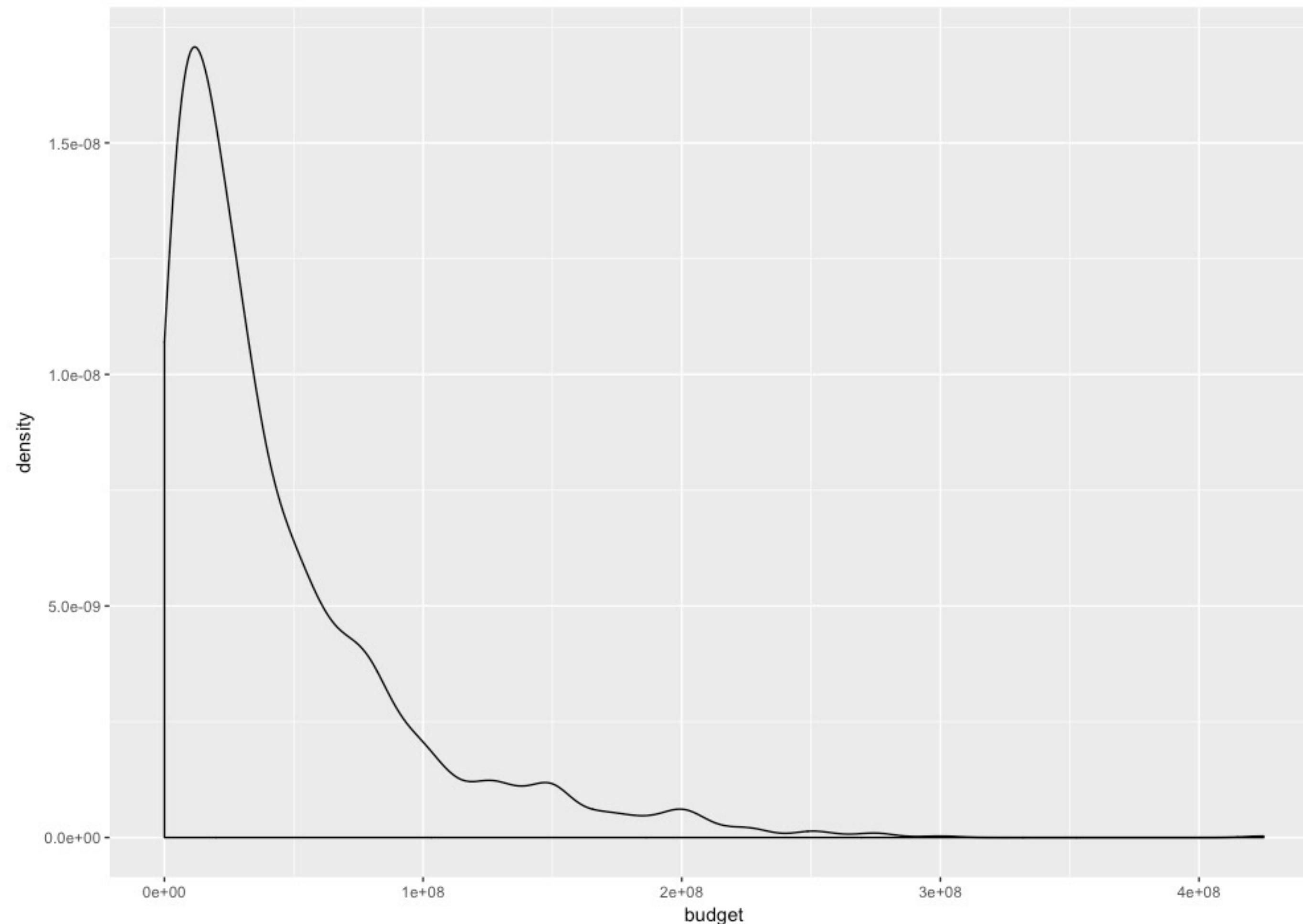
```
ggplot(data = bechdel) +
 geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```



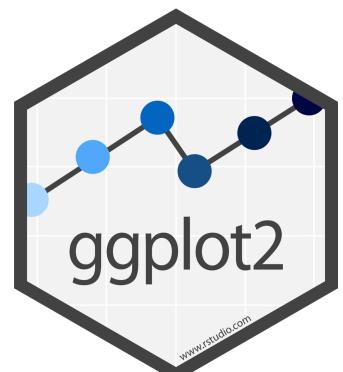
# Your Turn 6

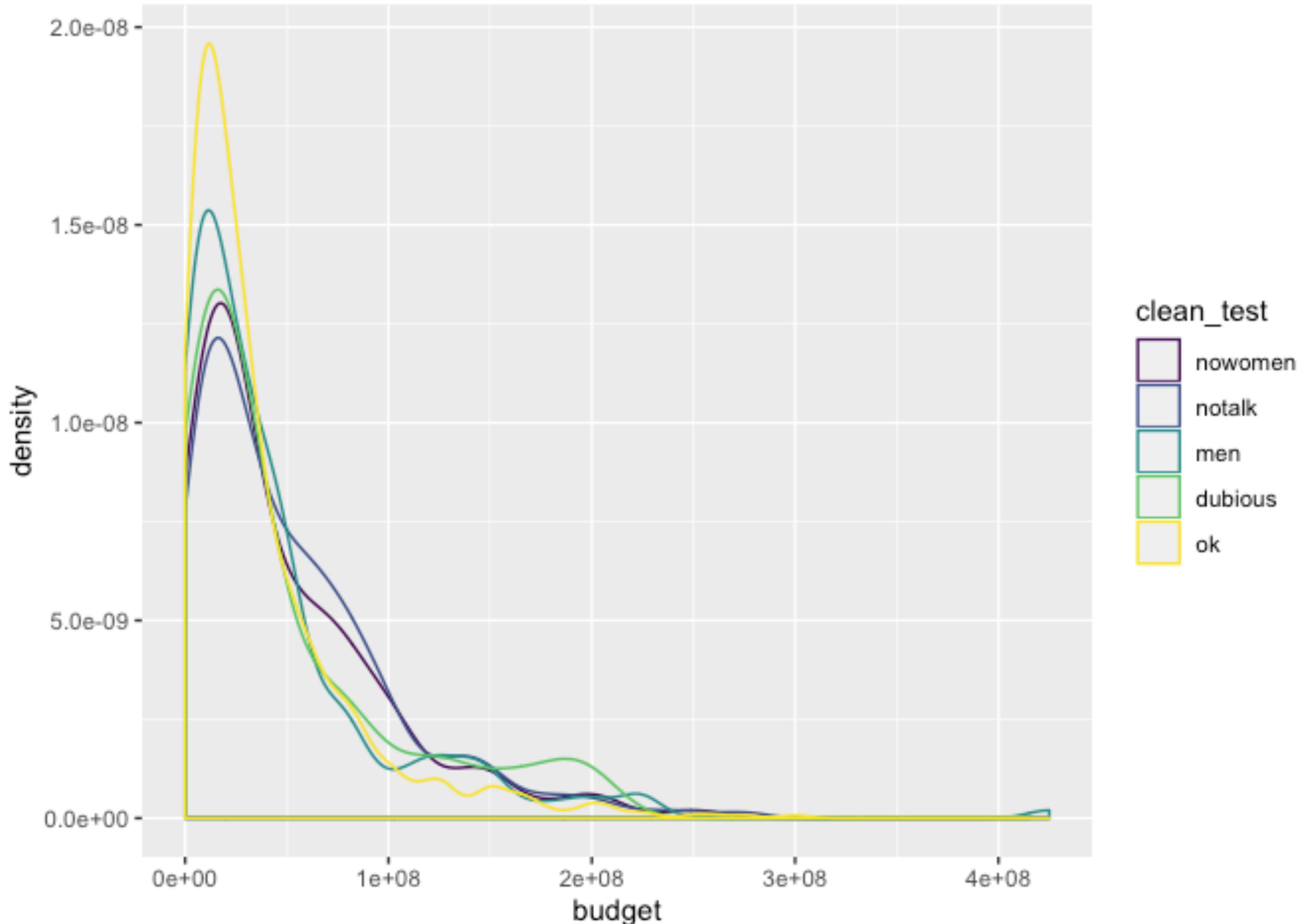
With your partner, make the density plot of **budget** colored by **clean\_test** below. Use the cheatsheet. Try your best guess.



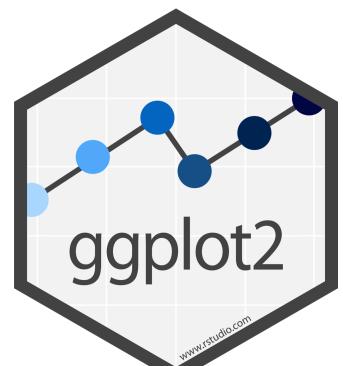


```
ggplot(data = bechdel) +
 geom_density(mapping = aes(x = budget))
```



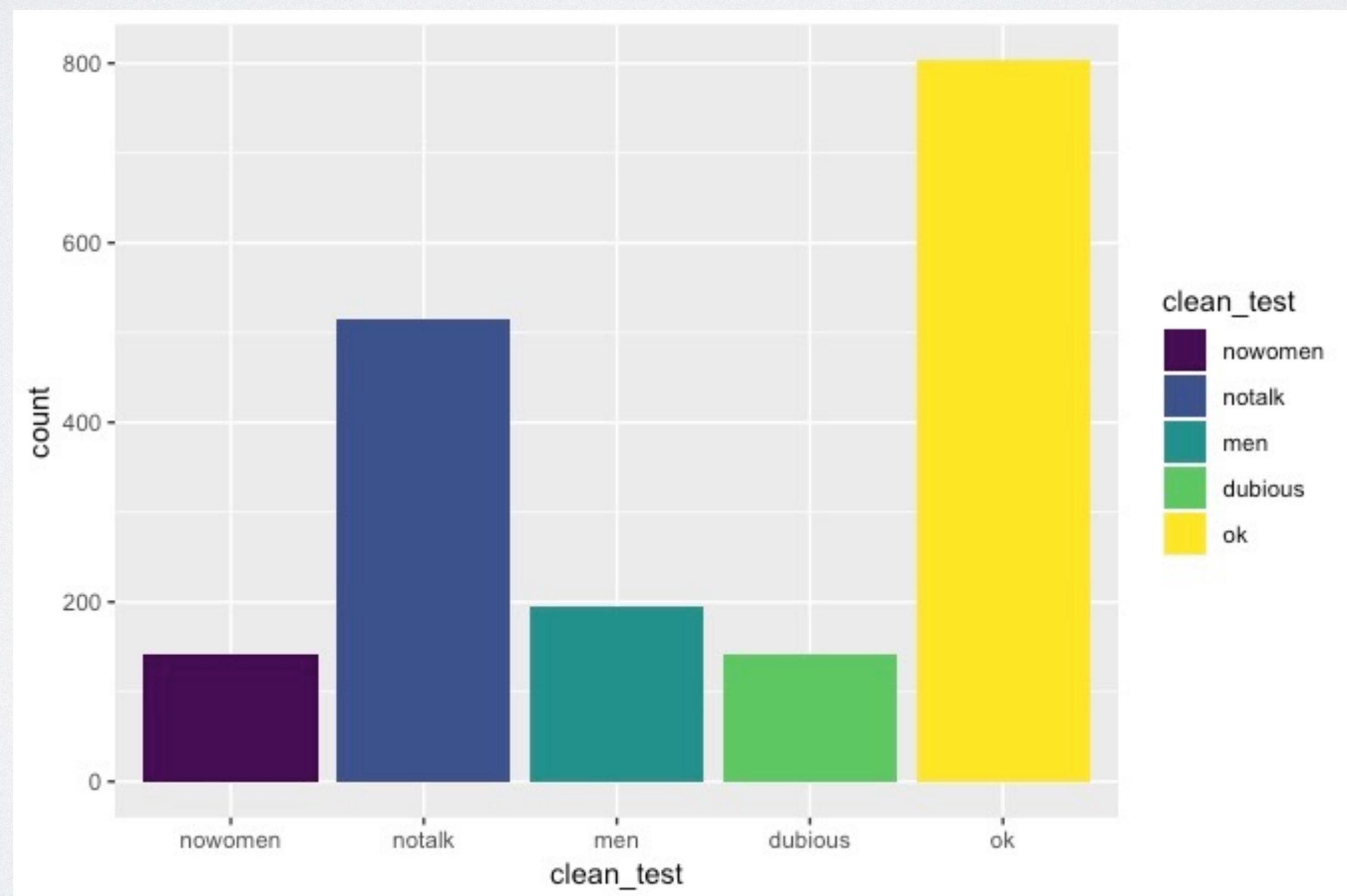


```
ggplot(data = bechdel) +
 geom_density(mapping = aes(x = budget, color=clean_test))
```

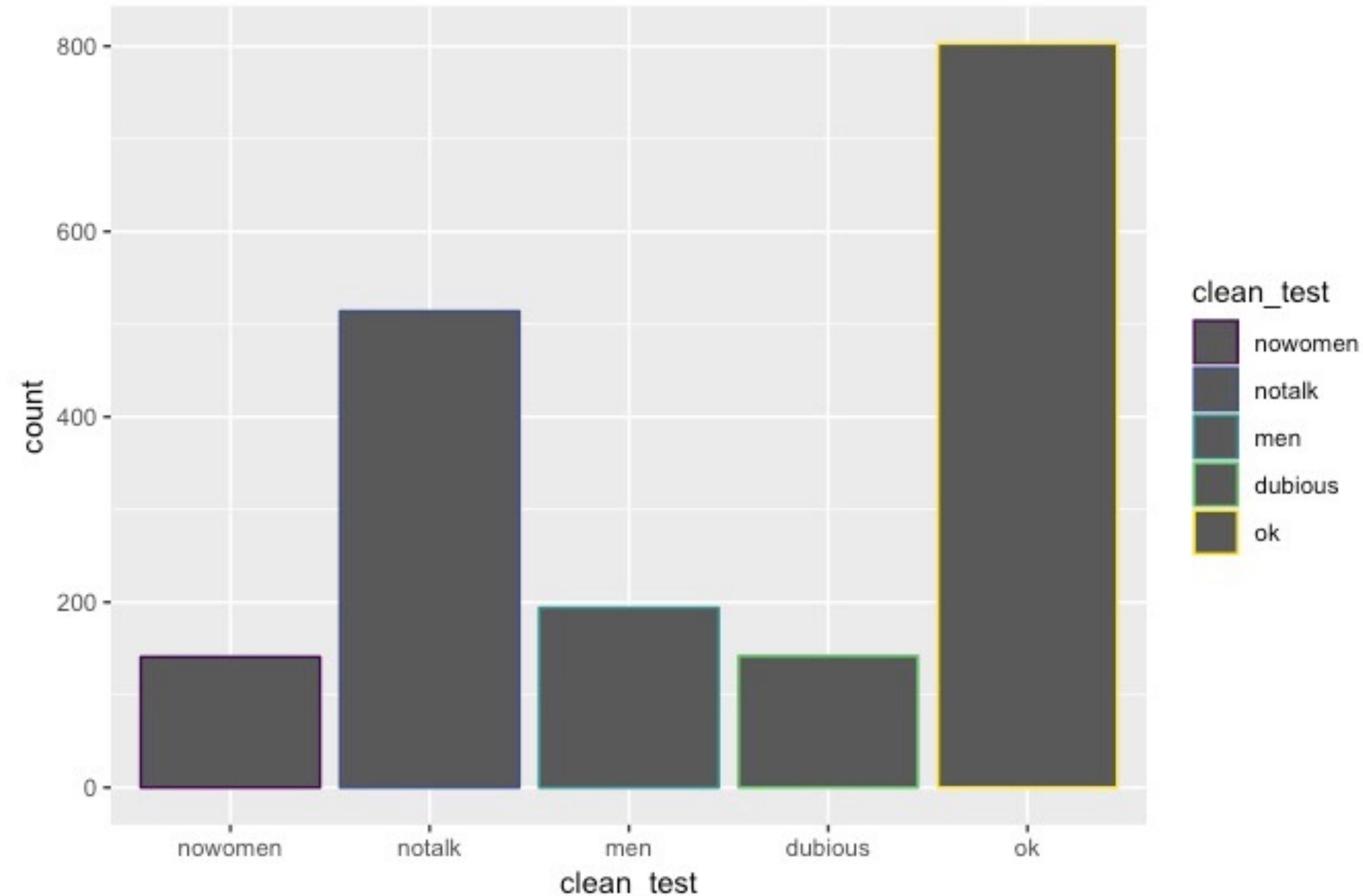


# Your Turn 7

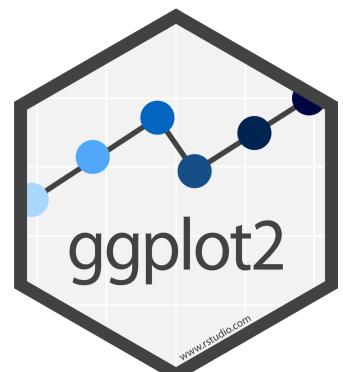
With your partner, make the bar chart of **clean\_test** colored by **clean\_test** below. Use the cheatsheet. Try your best guess.

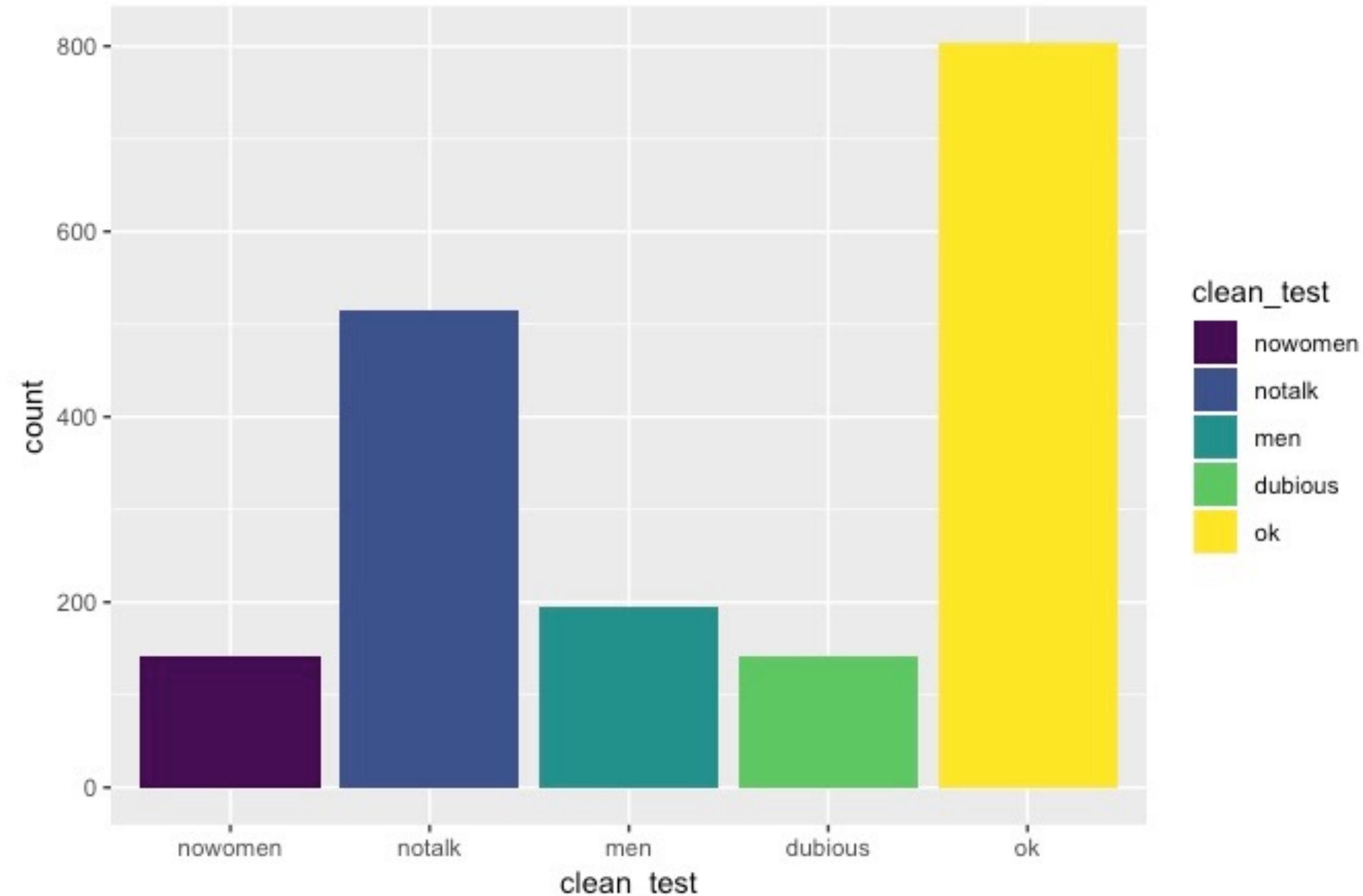


02 : 00

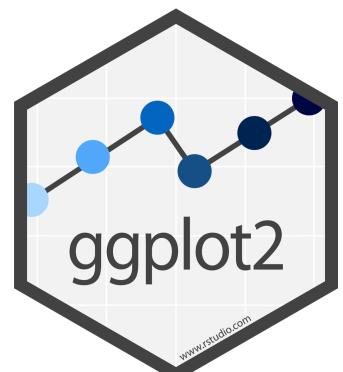


```
ggplot(data=bechdel) +
 geom_bar(aes(x=clean_test, color=clean_test))
```





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

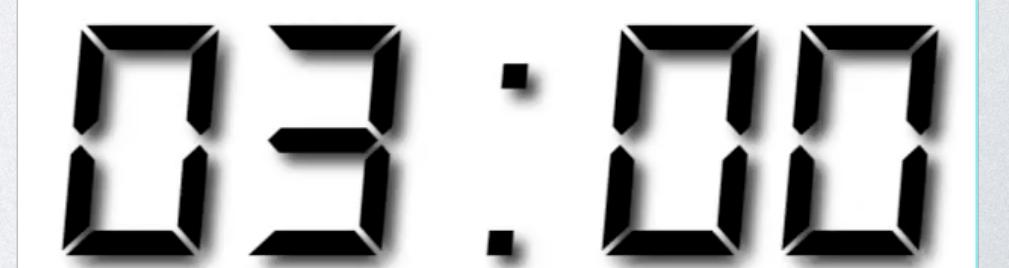


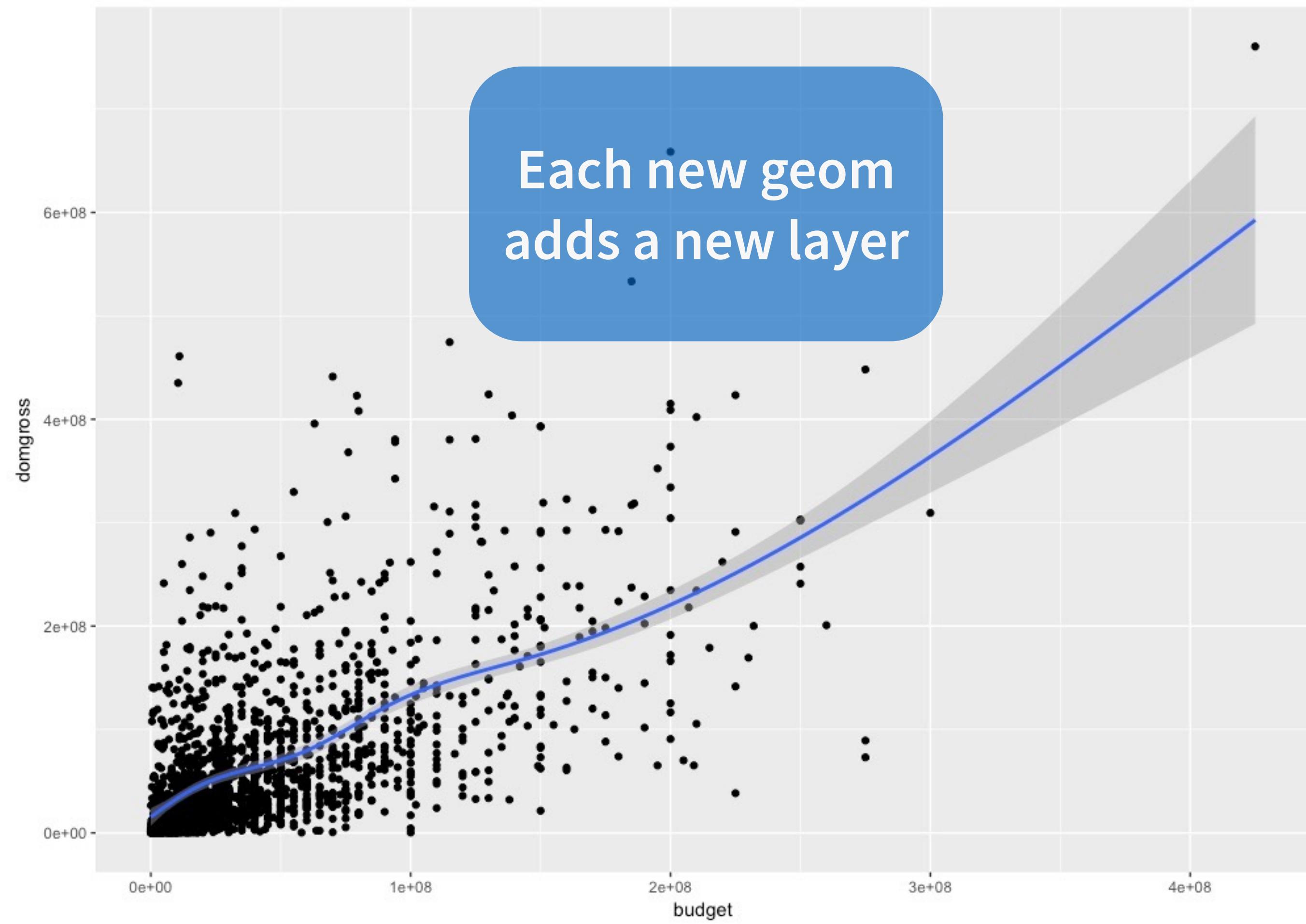
# Your Turn 8

With a partner, predict what this code will do.

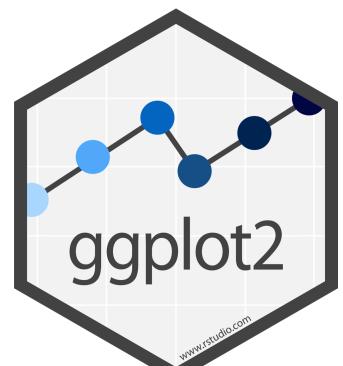
Then run it.

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



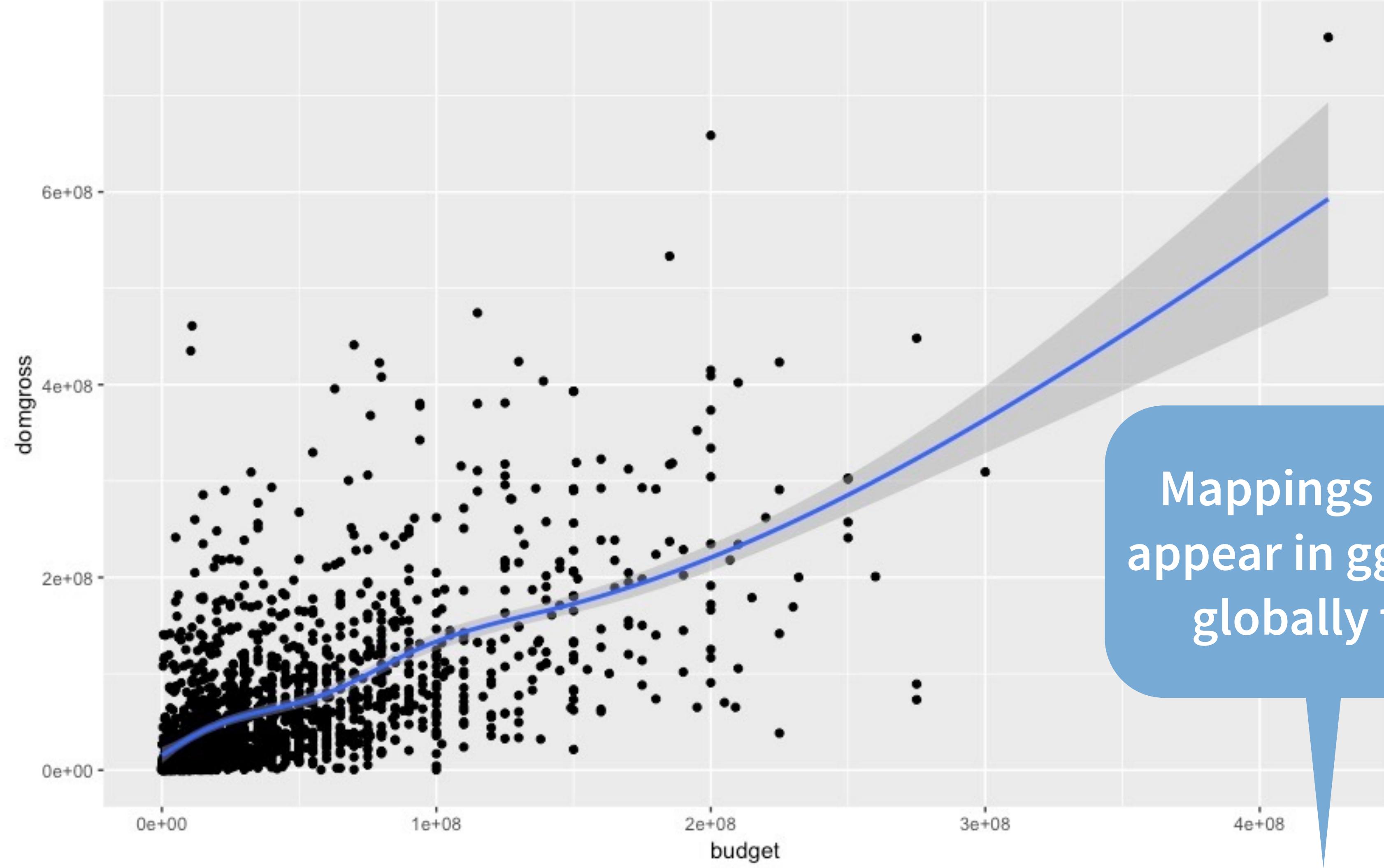


```
ggplot(data = bechdel) +
 geom_point(mapping = aes(x = budget, y = domgross)) +
 geom_smooth(mapping = aes(x = budget, y = domgross))
```

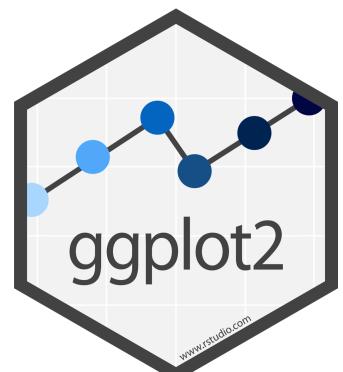


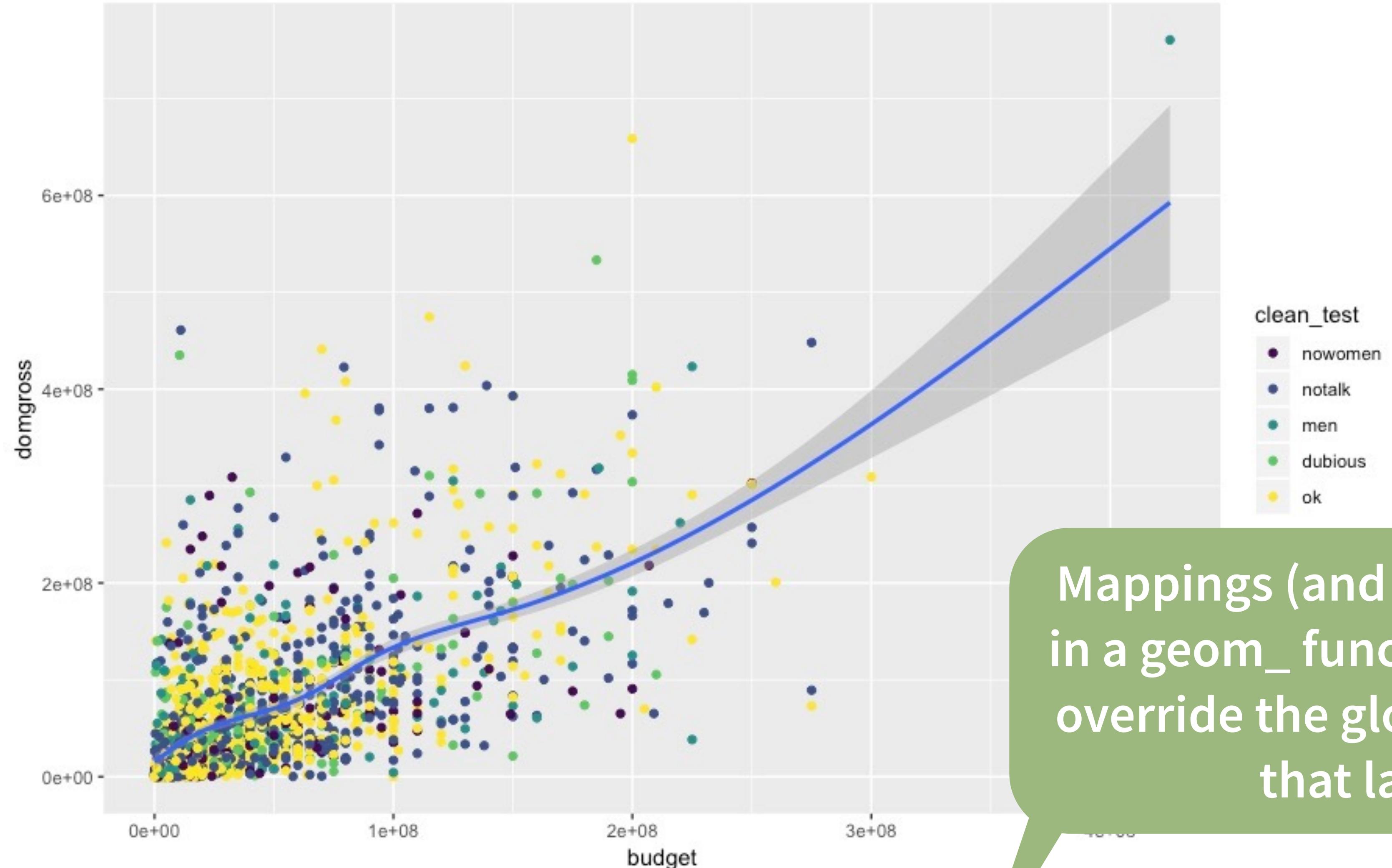
# global vs. local

R



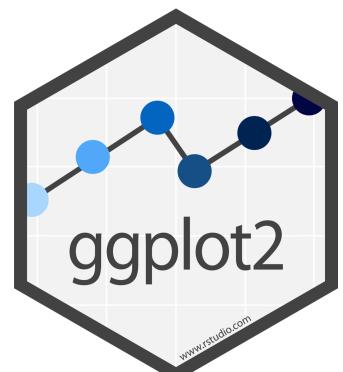
```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
 geom_point() +
 geom_smooth()
```

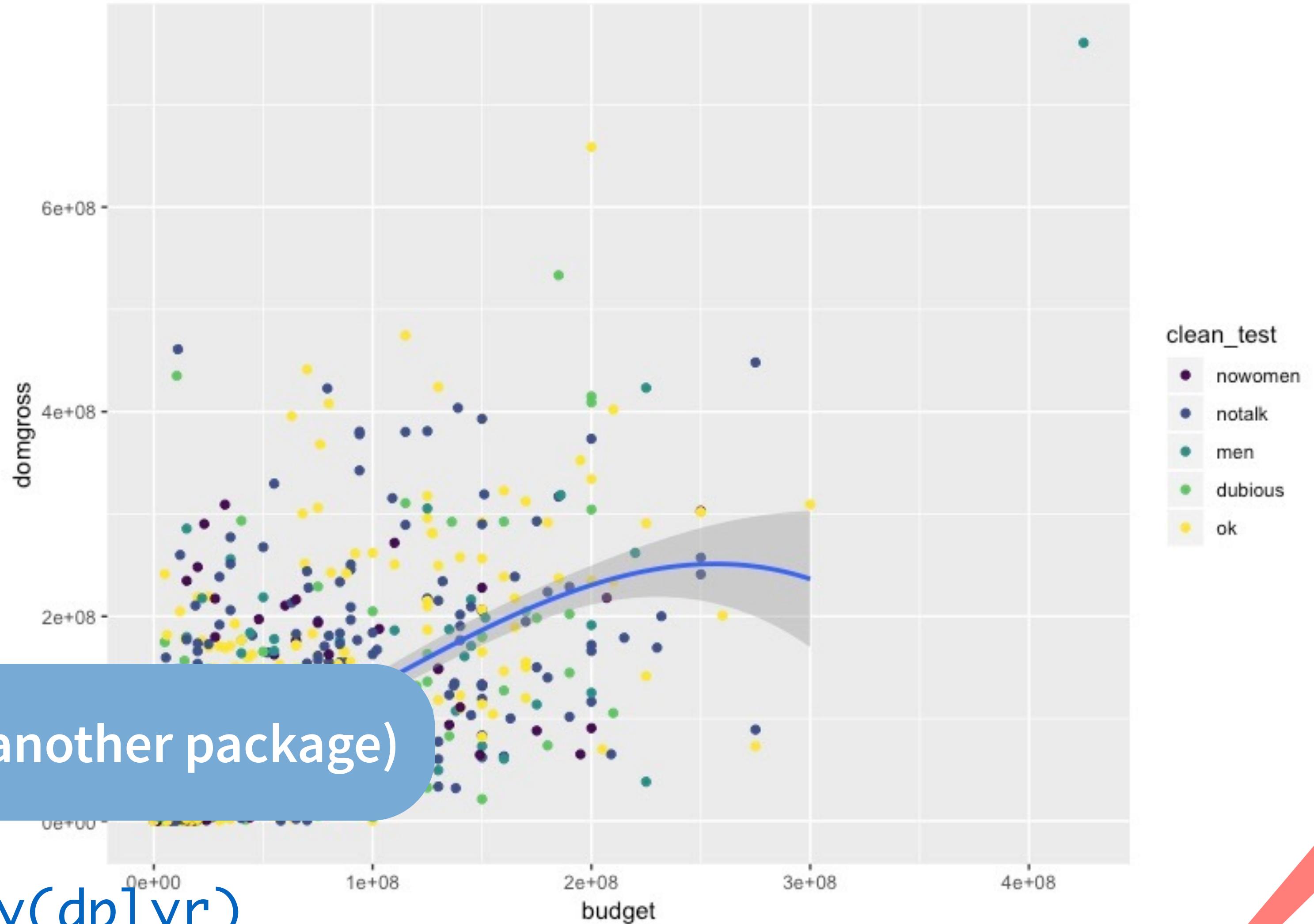




Mappings (and data) that appear in a `geom_` function will add to or override the global mappings for that layer only

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



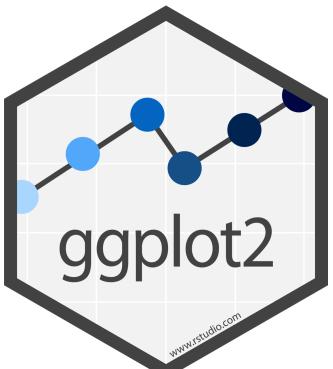


(requires another package)

data can also be set  
locally or globally

```
library(dplyr)
```

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



# Saving graphs



# “knitting” an R Markdown document

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

A screenshot of the RStudio interface demonstrating the knitting process. On the left, the R Markdown editor shows a code chunk titled "Visualization" with the following content:

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

The "Knit" button in the toolbar is circled in red. On the right, the Data pane shows the "bechdel" dataset with 1794 observations and 15 variables. Below it, the Plots pane displays a scatter plot of movie budget versus domestic gross. The x-axis is labeled "budget" and ranges from 0e+00 to 4e+08. The y-axis is labeled "domgross" and ranges from 0e+00 to 6e+08. A blue regression line with a gray shaded confidence interval is overlaid on a dense cloud of black data points.

# HTML preview

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

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

**Left Panel (R Notebook):**

- File menu: File, New, Open, Save, Save As, Import, Export, Print, Go to file/function.
- Addins menu: Addins.
- Editor tab: 01-Visualize.Rmd
- Toolbar icons: ABC, Knit, Publish.
- Code:

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

108:1 C Chunk 11

Console

**Right Panel (HTML Preview):**

  - Title: 01-Visualize.html
  - Header: ~data-science-tidy/materials/01-Visualize.html
  - Section: 

## Visualization
  - Section: 

### 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)
```
  - Section: 

### 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
```

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

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 interface, it doesn't work properly in R Notebooks. Most R functions have names that start with lowercase letters, so the uppercase "V" is there to remind you of its special status.)

  - Section: 

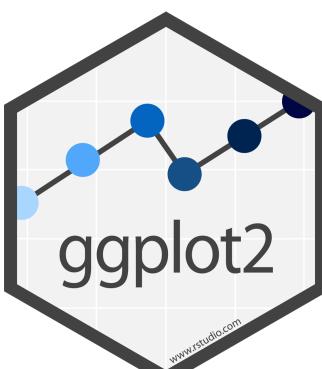
### Consider

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

# Sharing your work

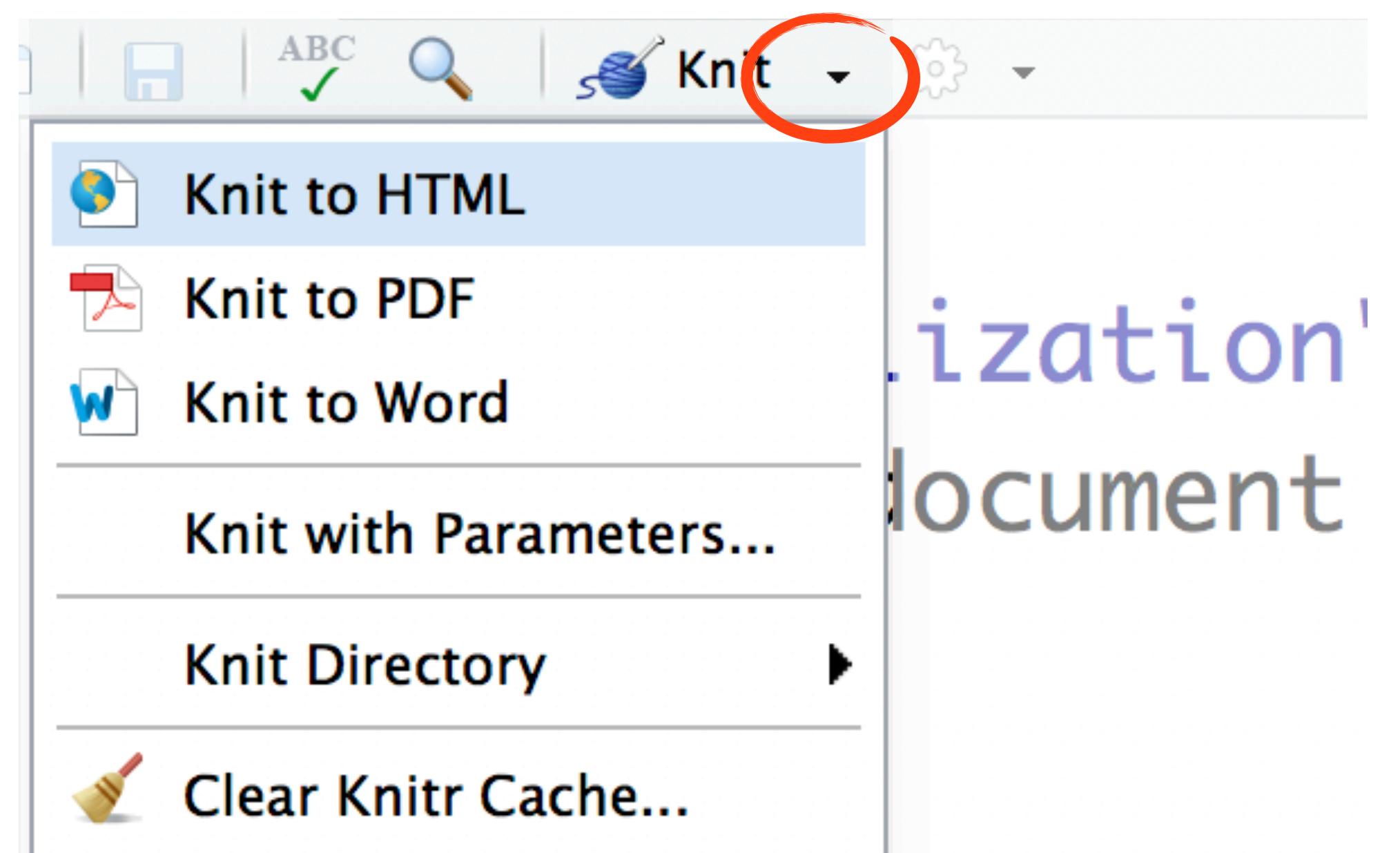
The preview is something that pops up for you to look at as you work, but RStudio automatically creates a file you can share with others when you knit an RMarkdown document. By default, it creates an HTML file, and the file's name corresponds to the name of your RMarkdown document.

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



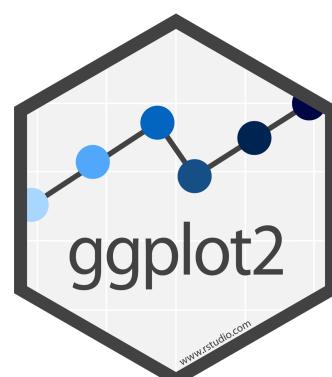
# Other formats

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



## Setup

The first chunk in an  
includes the R package



# Your Turn

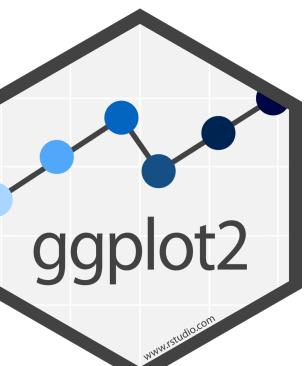
Locate the `intro_to_ggplot2.html` file in your Files pane. It will be located in your **working directory**



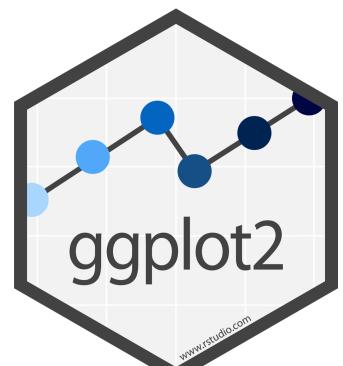
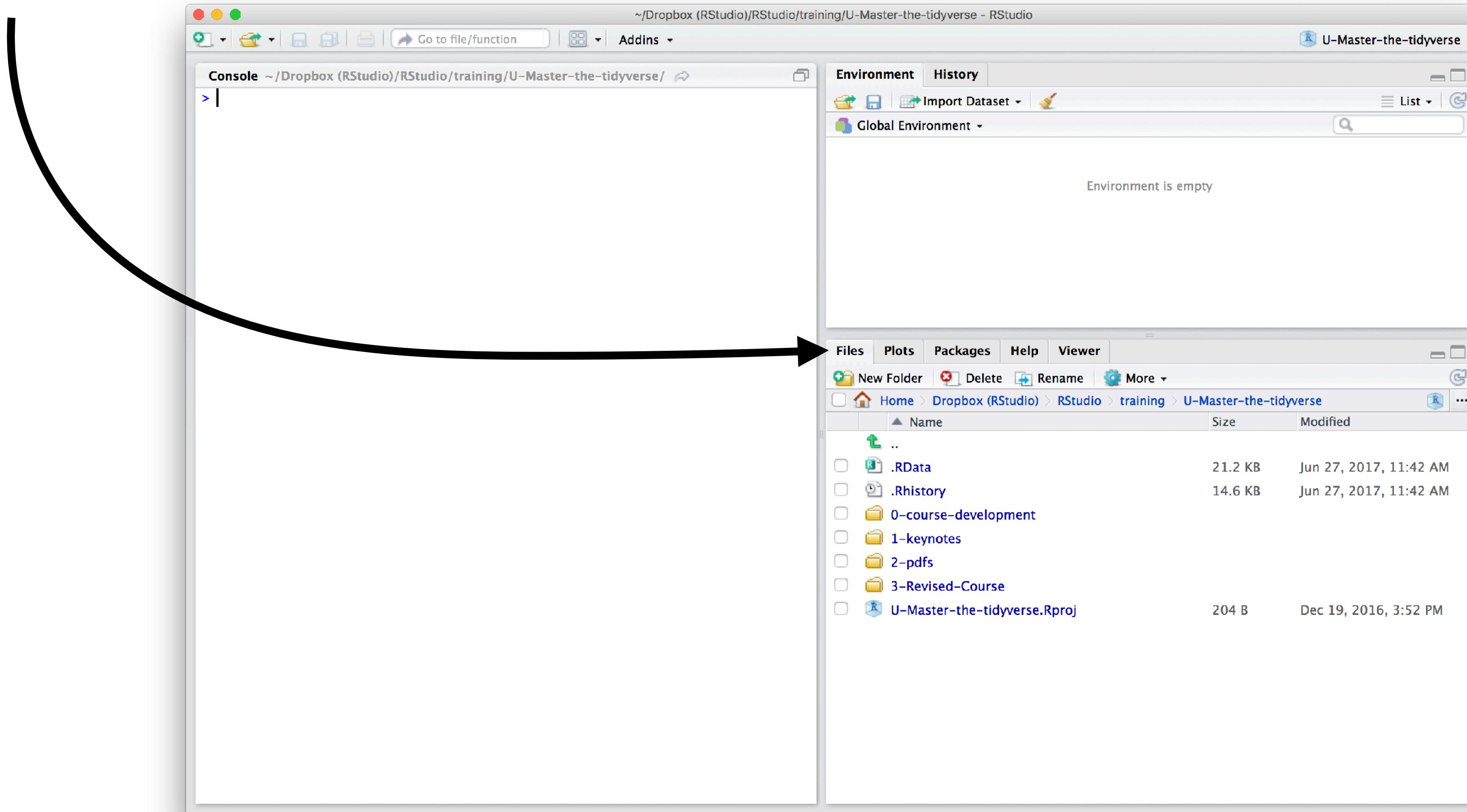
# Working directory

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
- If you are using **projects** you don't have to worry about working directories!

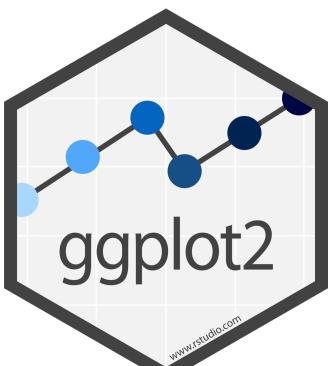
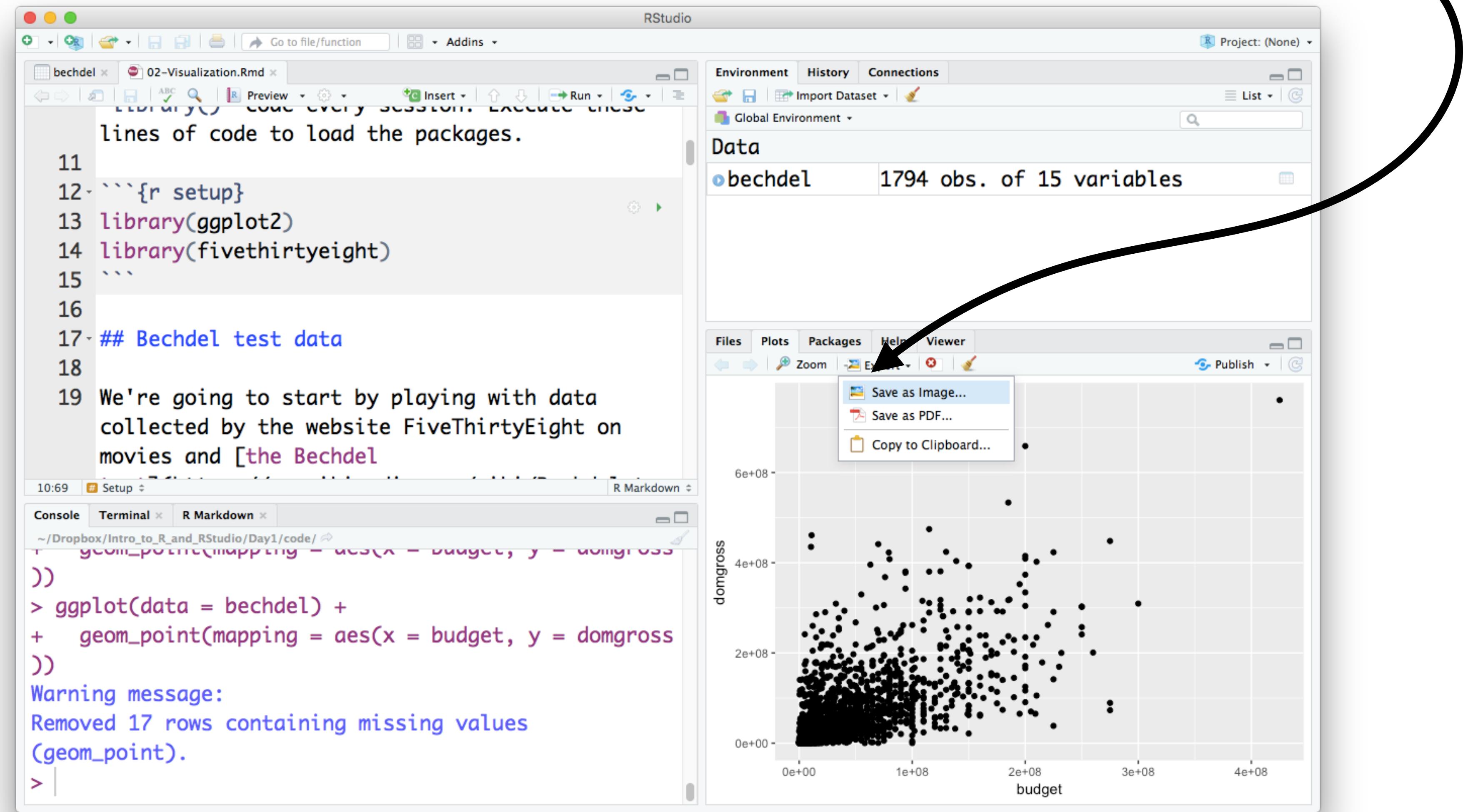


# The files pane of the IDE displays the contents of your working directory



# Manually saving plots

Save plots manually with the export menu



# Saving plots

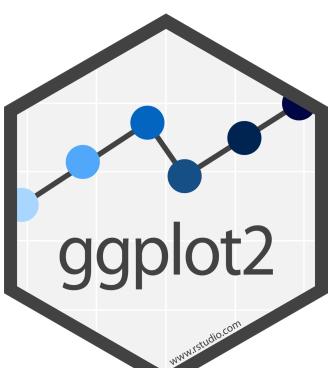
**ggsave()** saves the last plot.

Uses size on screen:

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

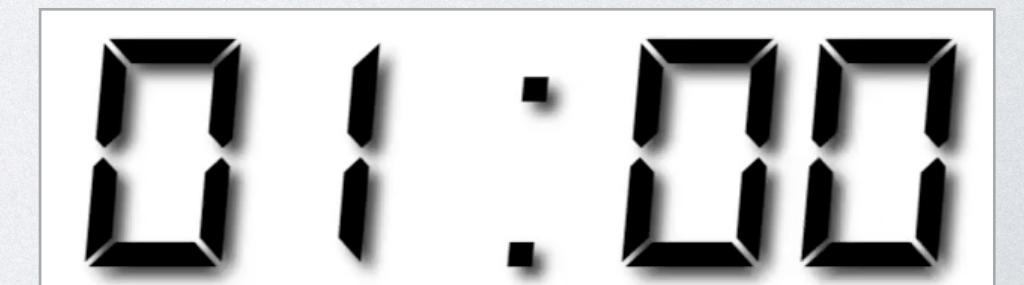
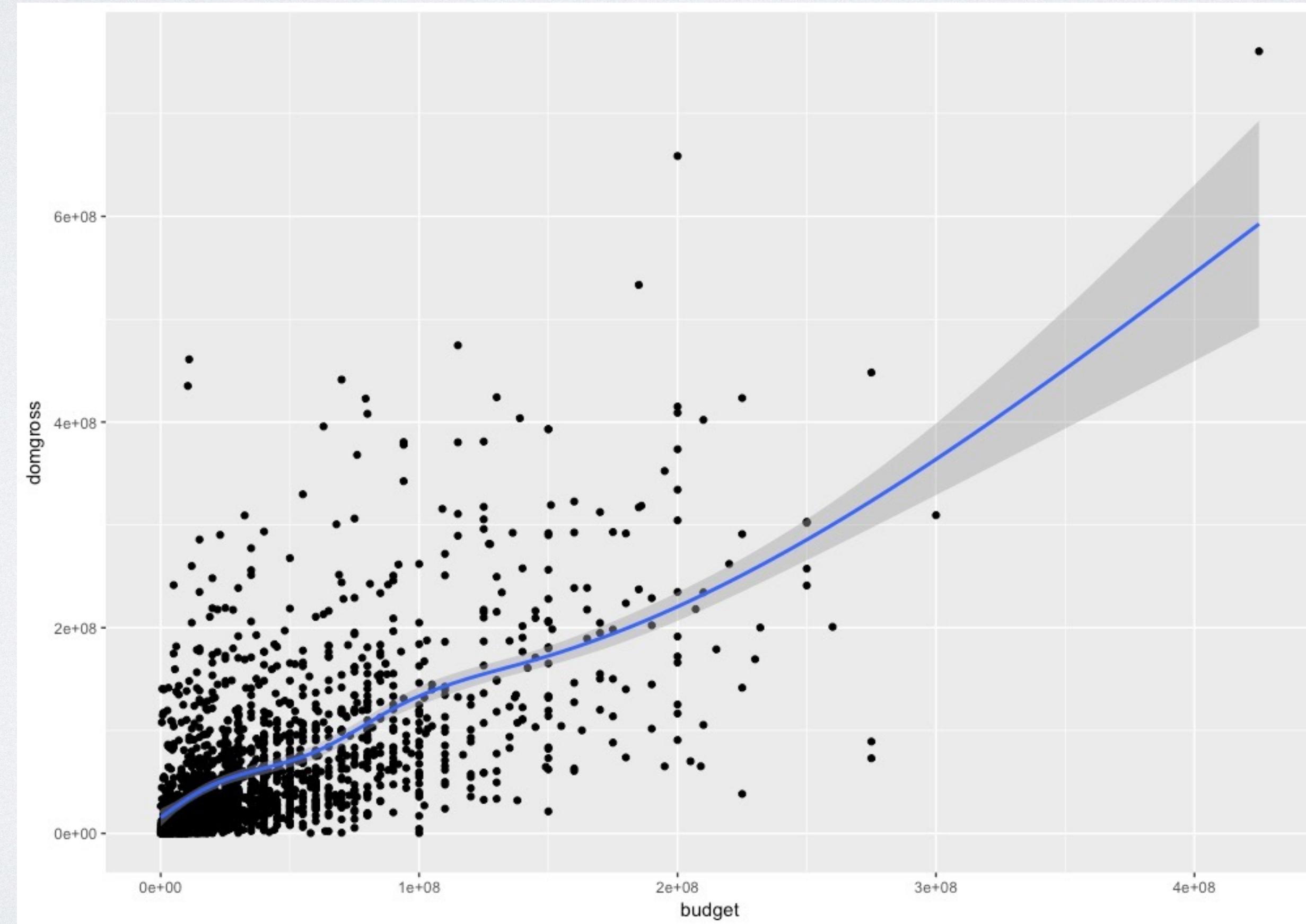
Specify size in inches

```
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).



# ggplot2



# Data Visualization with ggplot2 :: CHEAT SHEET

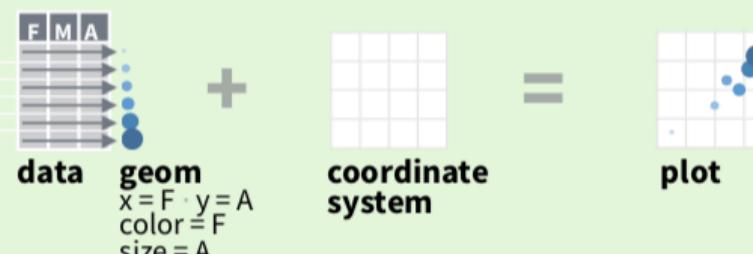


## Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and geoms—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



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

`ggplot(data = mpg, aes(x = cty, y = hwy))` Begins a plot that you finish by adding layers to. Add one geom function per layer.

**aesthetic mappings**    **data**    **geom**  
`qplot(x = cty, y = hwy, data = mpg, geom = "point")`  
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

`last_plot()` Returns the last plot

`ggsave("plot.png", width = 5, height = 5)` Saves last plot as 5' x 5' file named "plot.png" in working directory.  
Matches file type to file extension.

## Geoms

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

### 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 = 1)** - x, 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(f1))
```

**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, linelheight, 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, linelheight, 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

### 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

#### 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

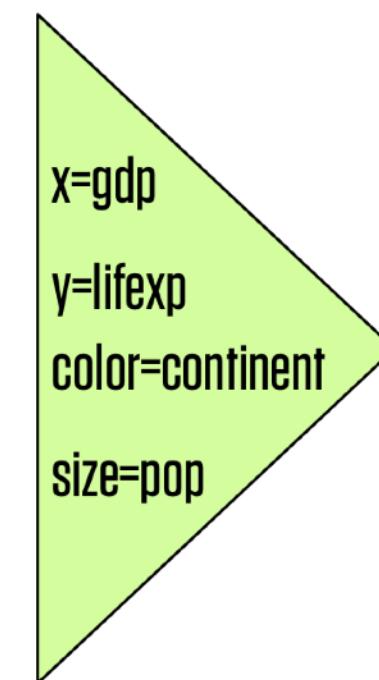
# ggplot's FLOW OF ACTION

## 1. Tidy Data

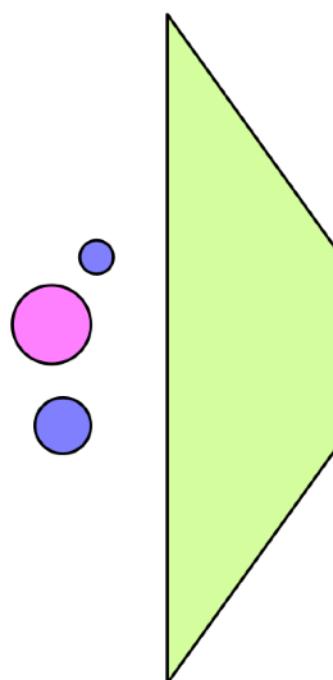
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

```
ggplot(data = gapminder, mapping =
aes(x = gdp,
y = lifespan,
color = continent,
size = pop))
```

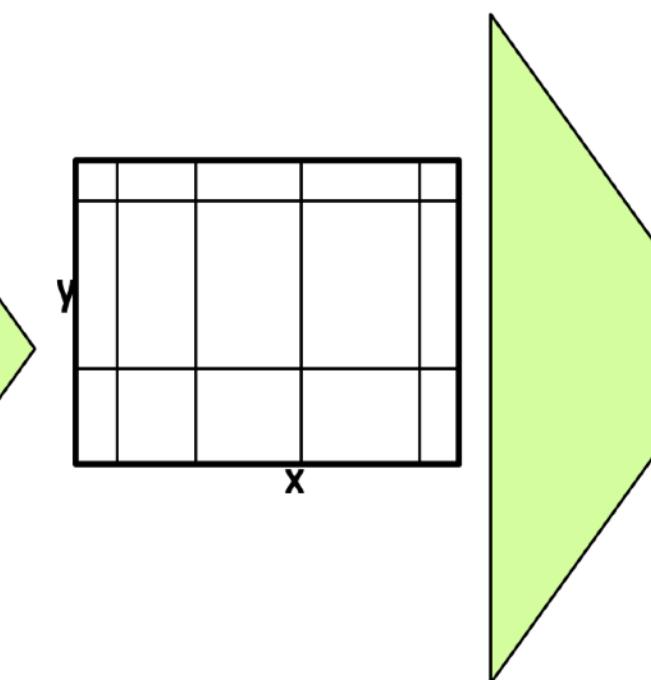
## 2. Mapping



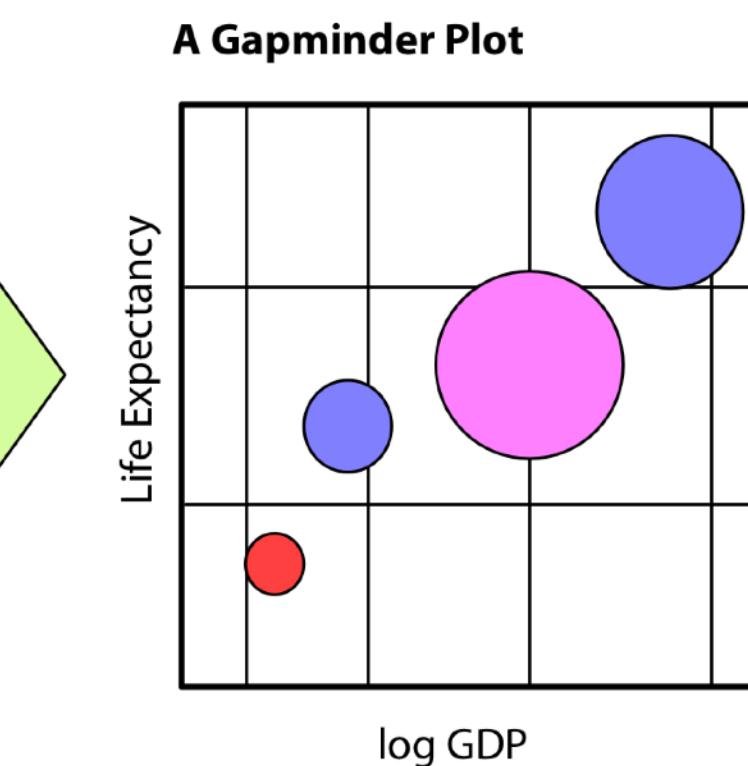
## 3. Geom



## 4. Co-ordinates, Scales

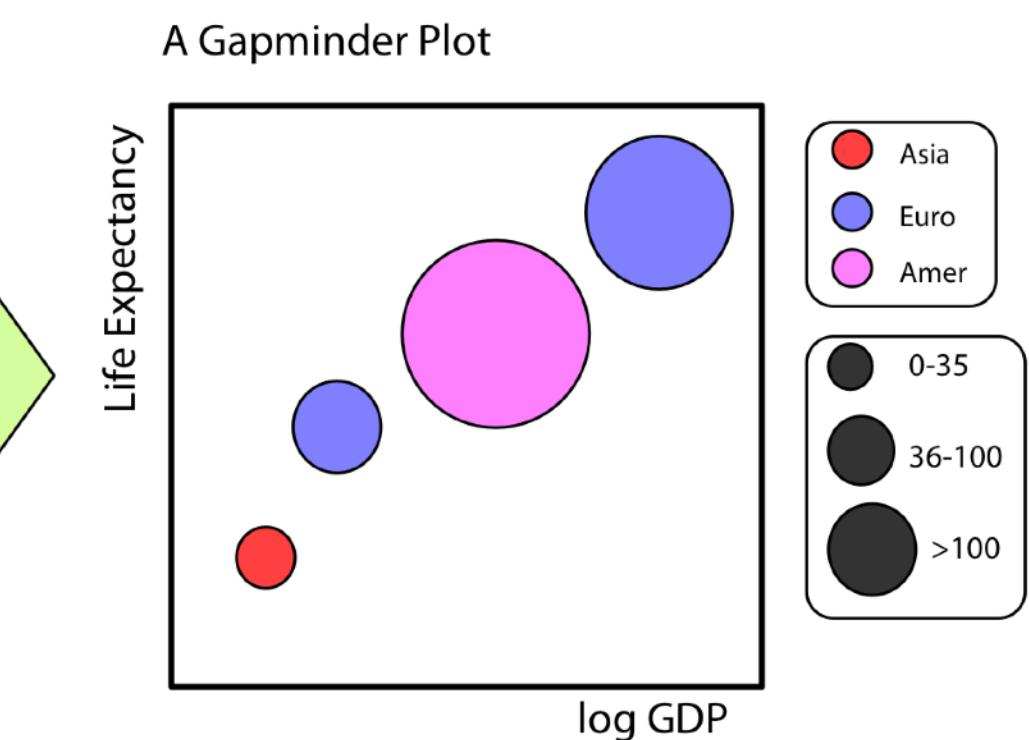


## 5. Labels & Guides



```
labs()
guides()
```

## 6. Themes

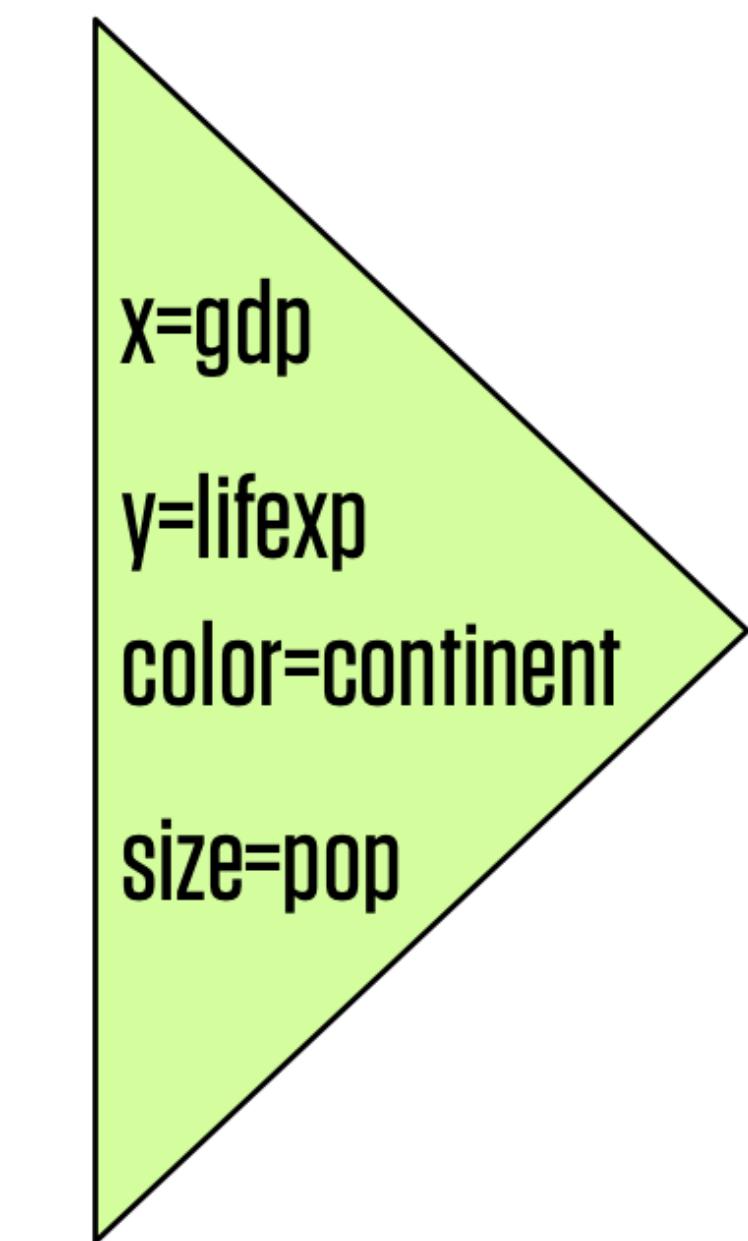


```
theme_minimal()
```

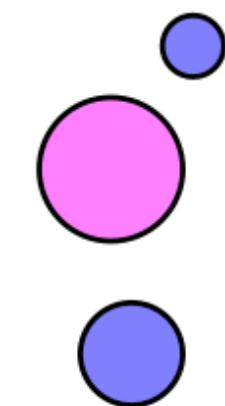
# 1. Tidy Data

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

# 2. Mapping



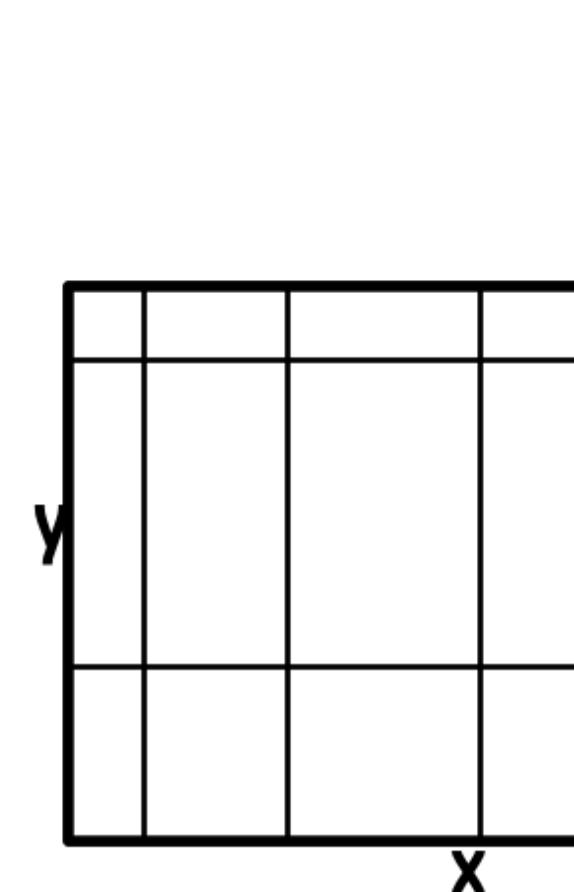
# 3. Geom



```
ggplot(data = gapminder, mapping =
aes(x = gdp,
y = lifespan,
color = continent,
size = pop))
```

geom\_point()

coord\_cartes  
scale\_x\_log



## 2. Mapping

	continent
0	Euro
1	Amer
2	Euro
3	Asia

```
x=gdp
y=lifexp
color=continent
size=pop
```

## 3. Geom

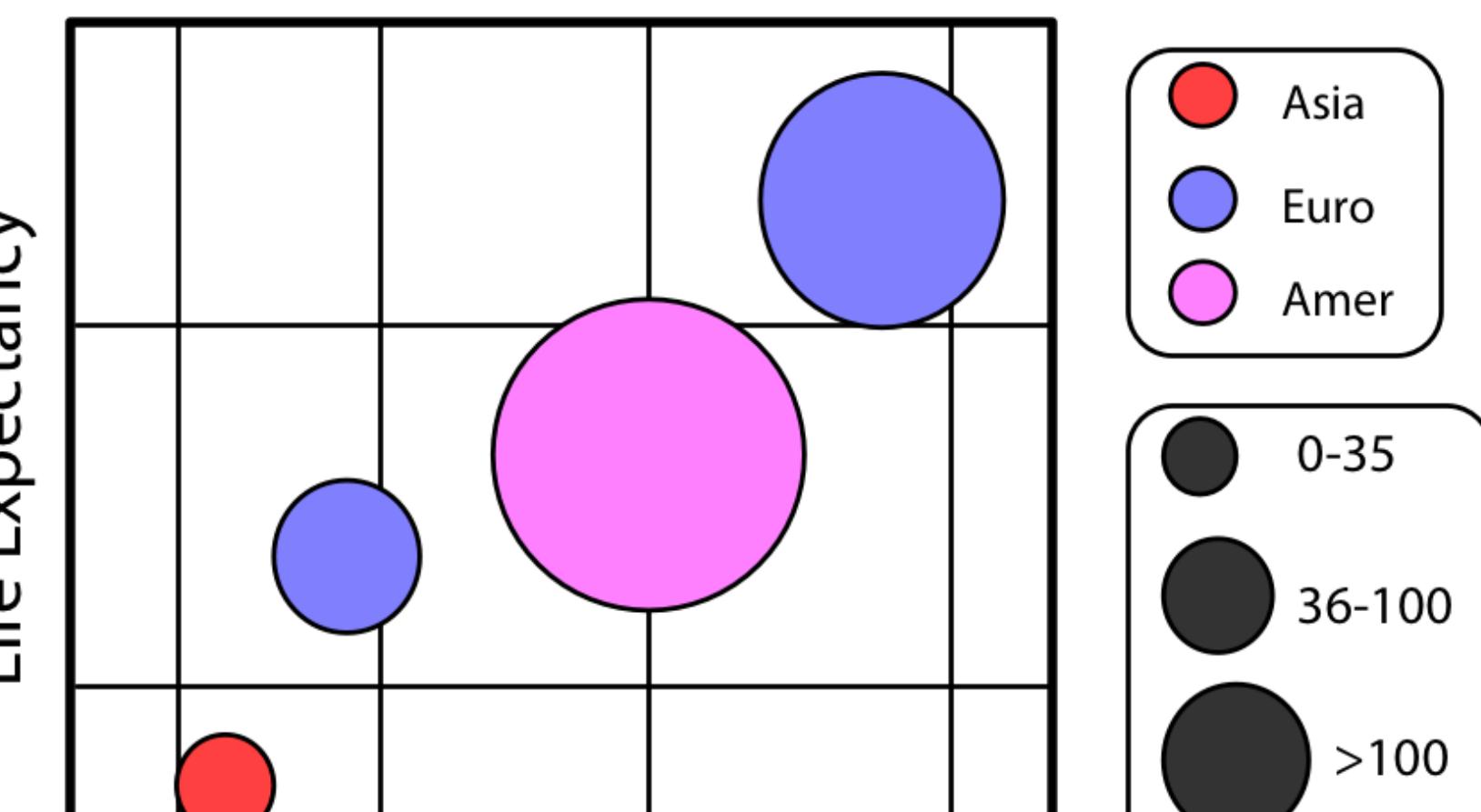
```
geom_point()
```

## 4. Co-ordinates, Scales

```
coord_cartesian()
scale_x_log10()
```

## 5. Labels & Guides

A Gapminder Plot



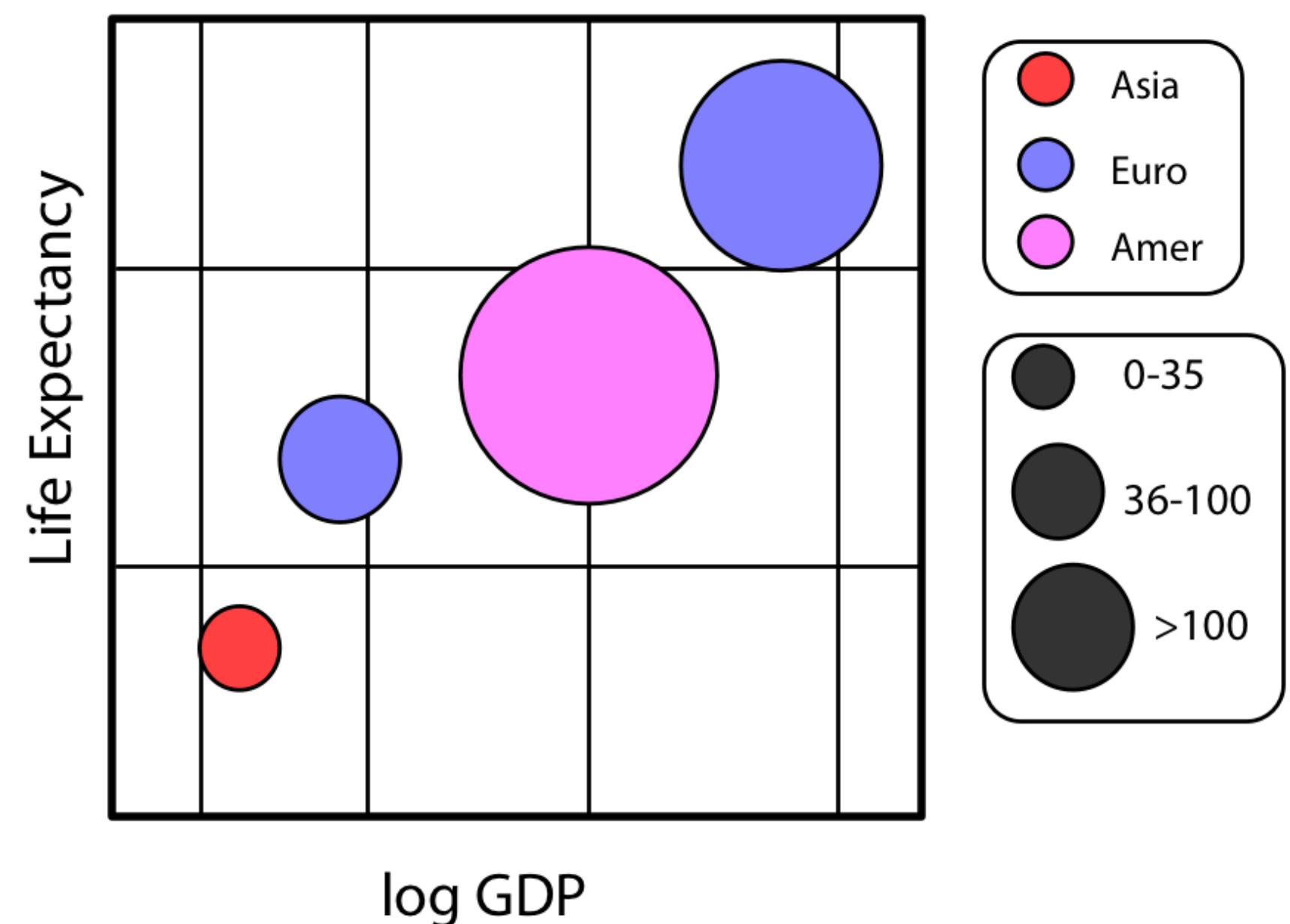
```
a = gapminder, mapping =
aes(x = gdp,
y = lifespan,
color = continent,
size = pop))
```

```
labs()
guides()
```

ates,

## 5. Labels & Guides

A Gapminder Plot

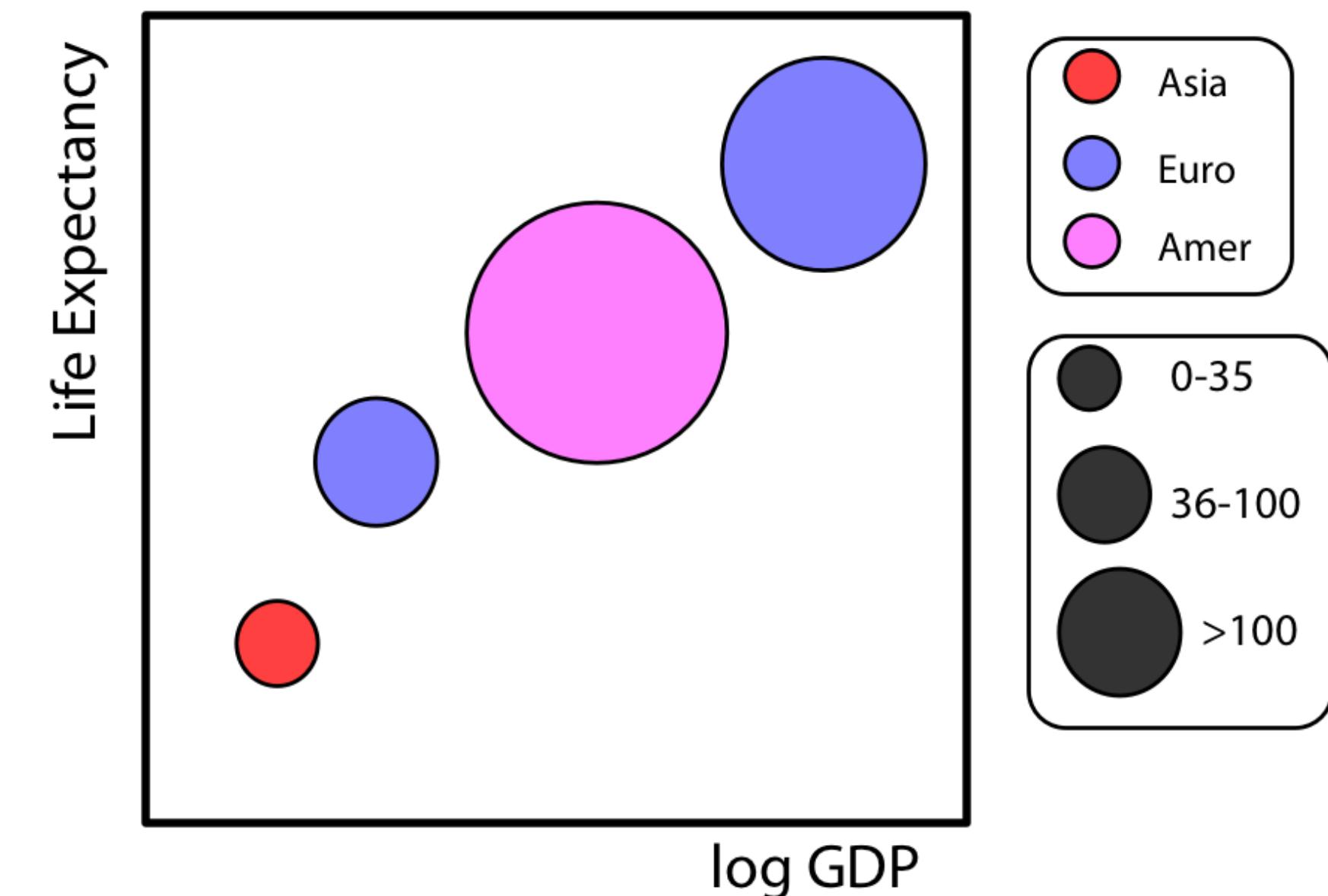


labs()

guides()

## 6. Themes

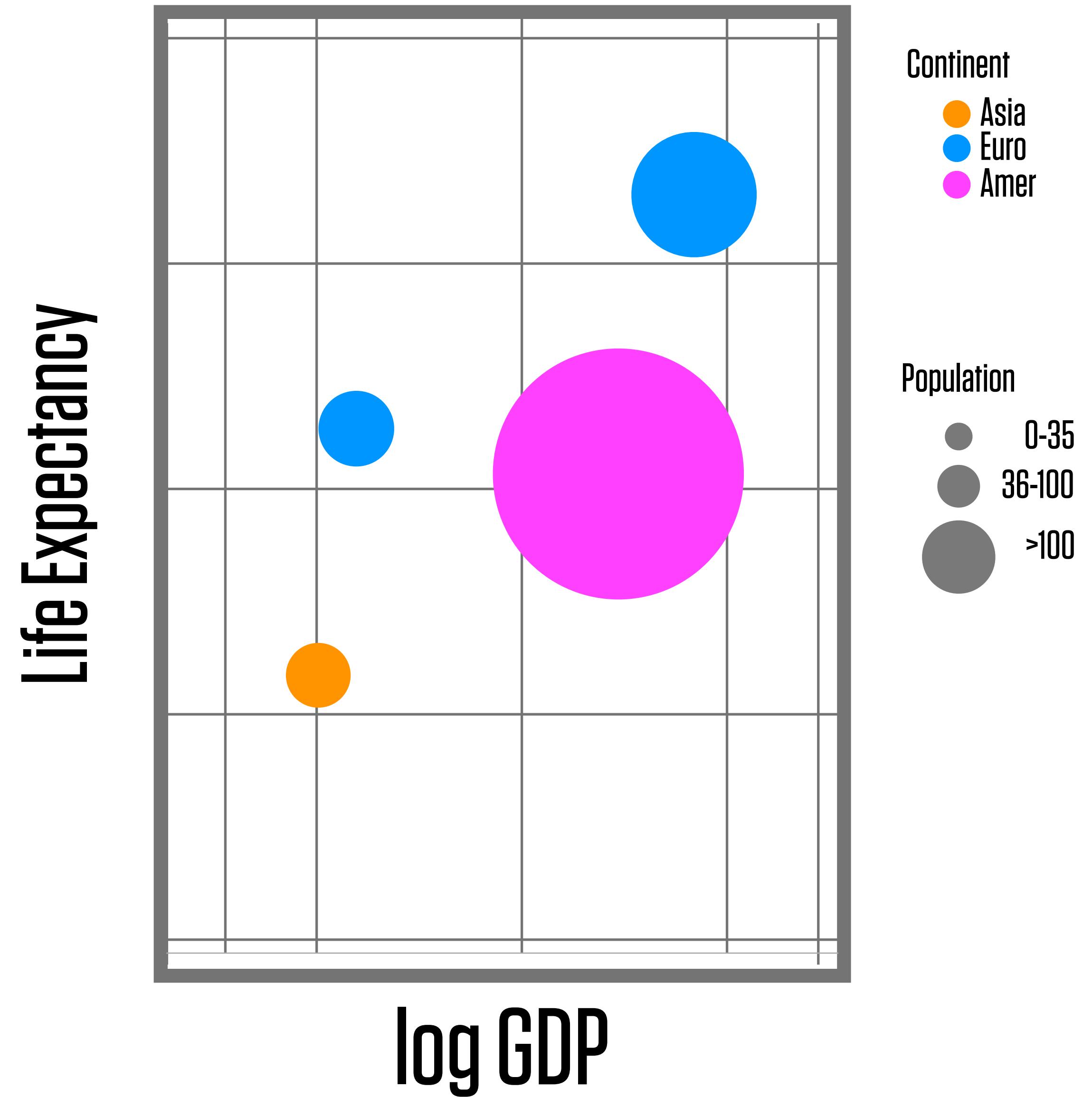
A Gapminder Plot



theme\_minimal()

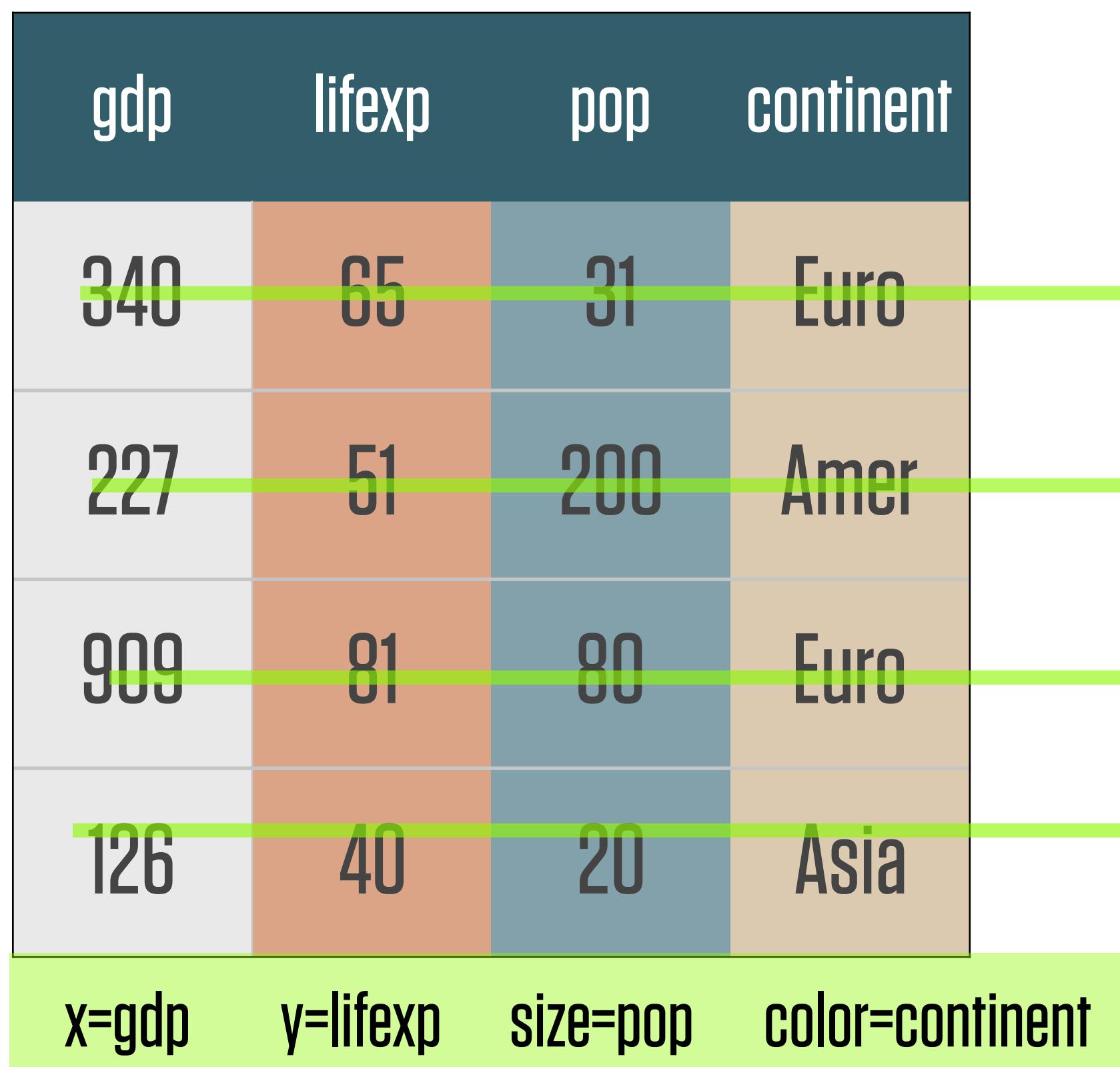
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

# A Gapminder Plot



# 1. Tidy Data

`ggplot(data = gapminder)`



# 2. Mapping

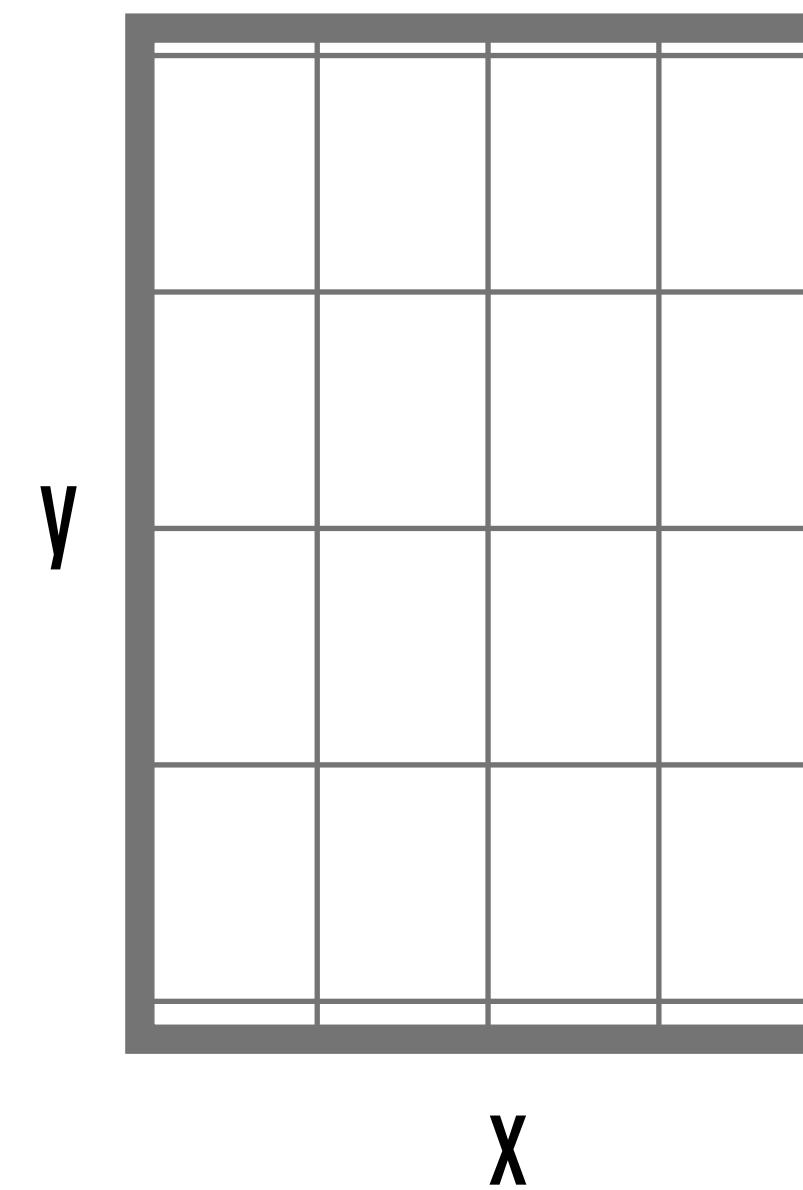
`ggplot(mapping = aes(x = ...))`

# 3. Geom

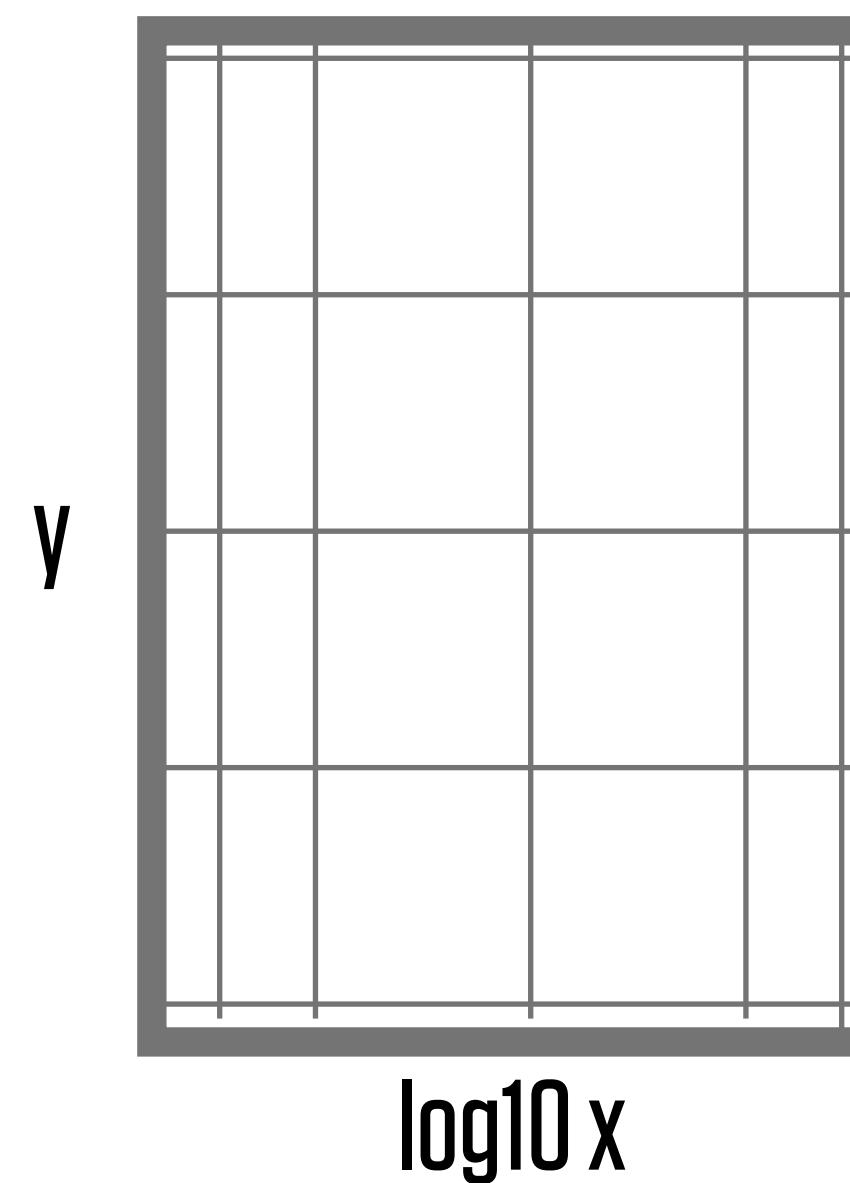
`geom_point()`

Required

## 4. Coordinate System

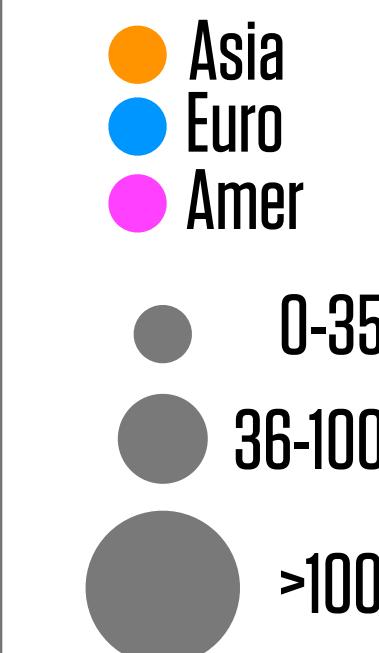


## 5. Scales



## 6. Labels & Guides

A Gapminder Plot



Life Expectancy

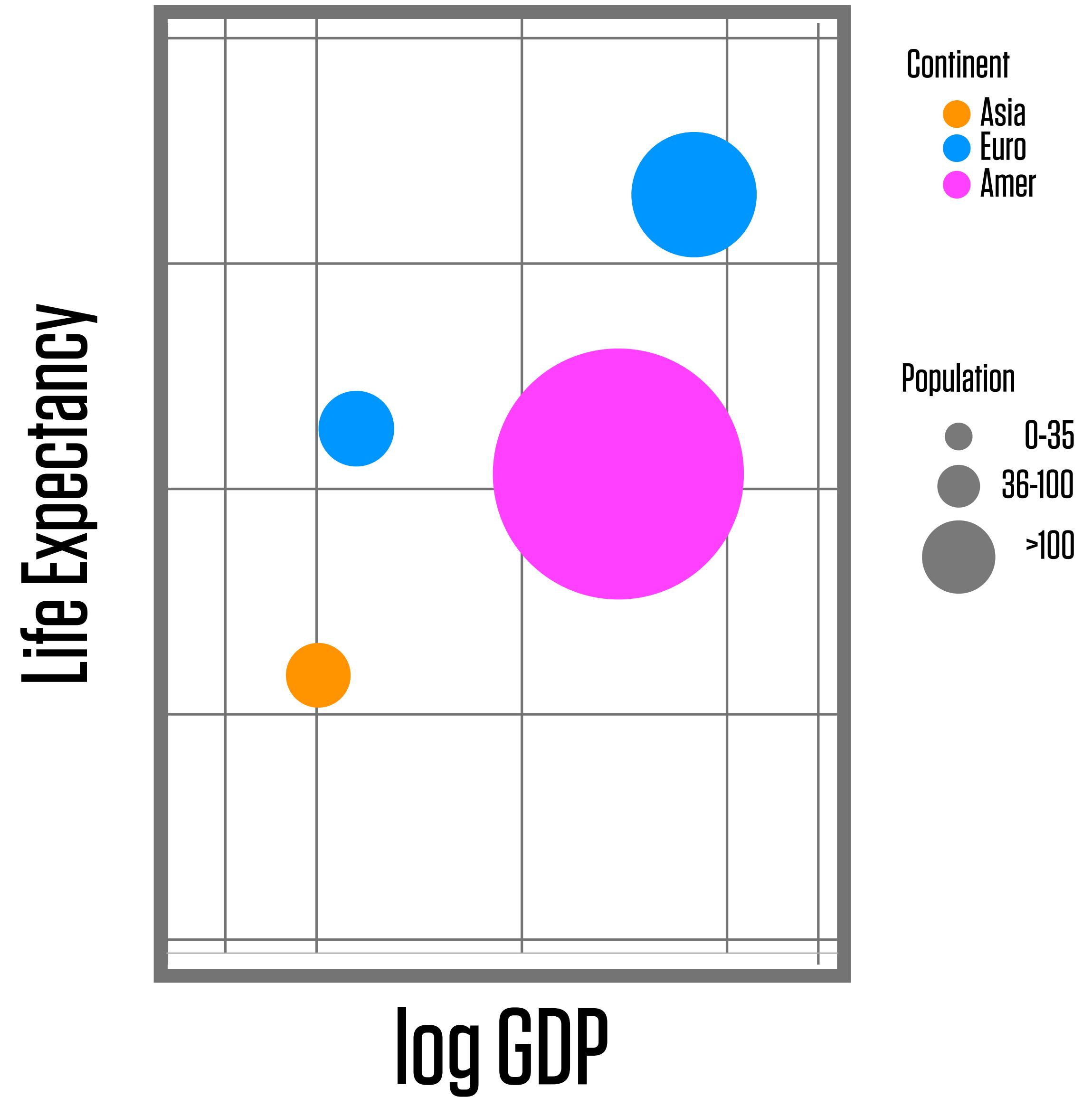
log GDP

Continent

Population

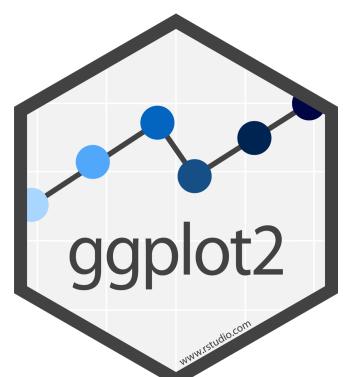
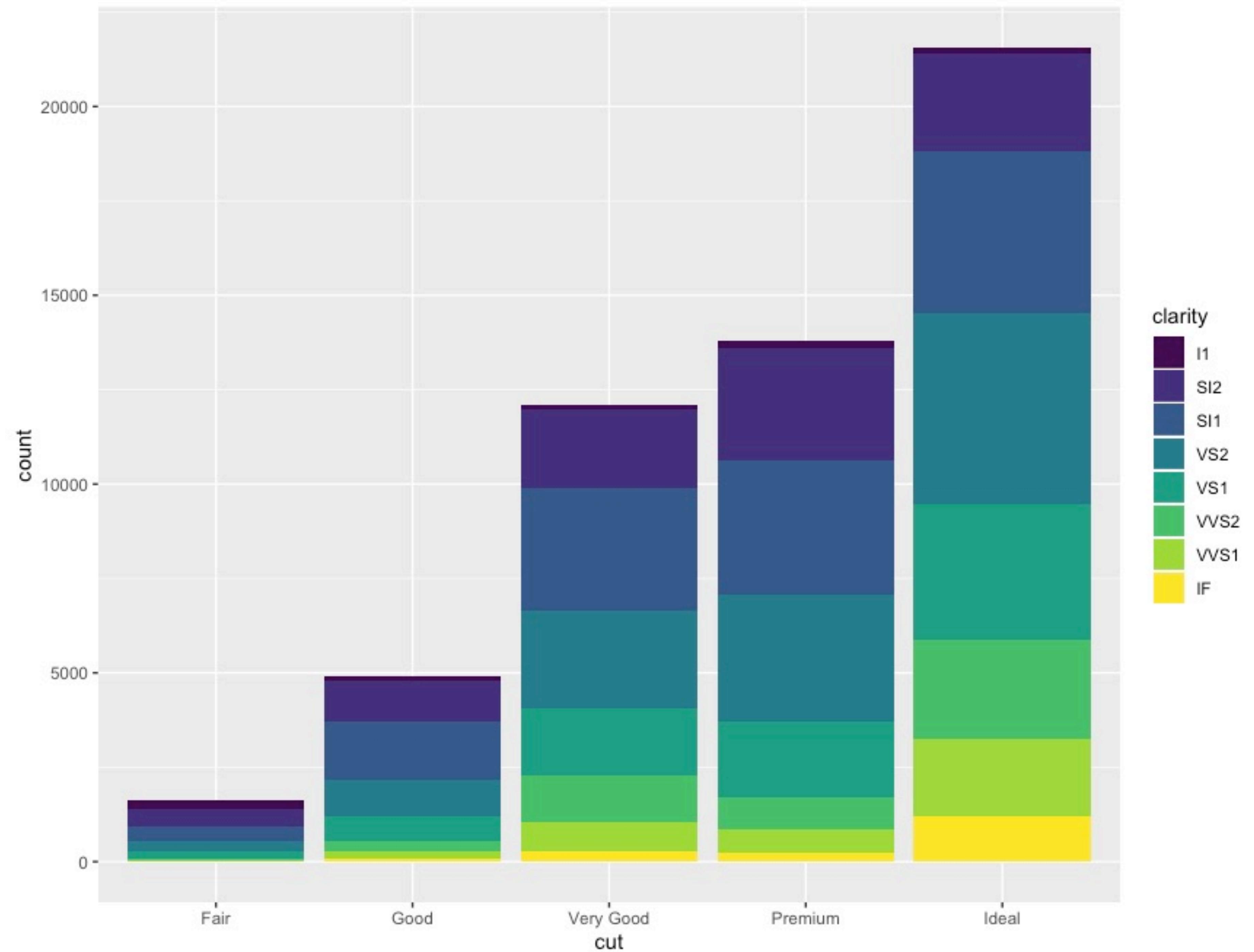
Initially Implicit

# A Gapminder Plot



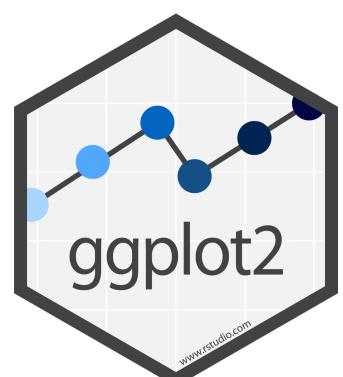
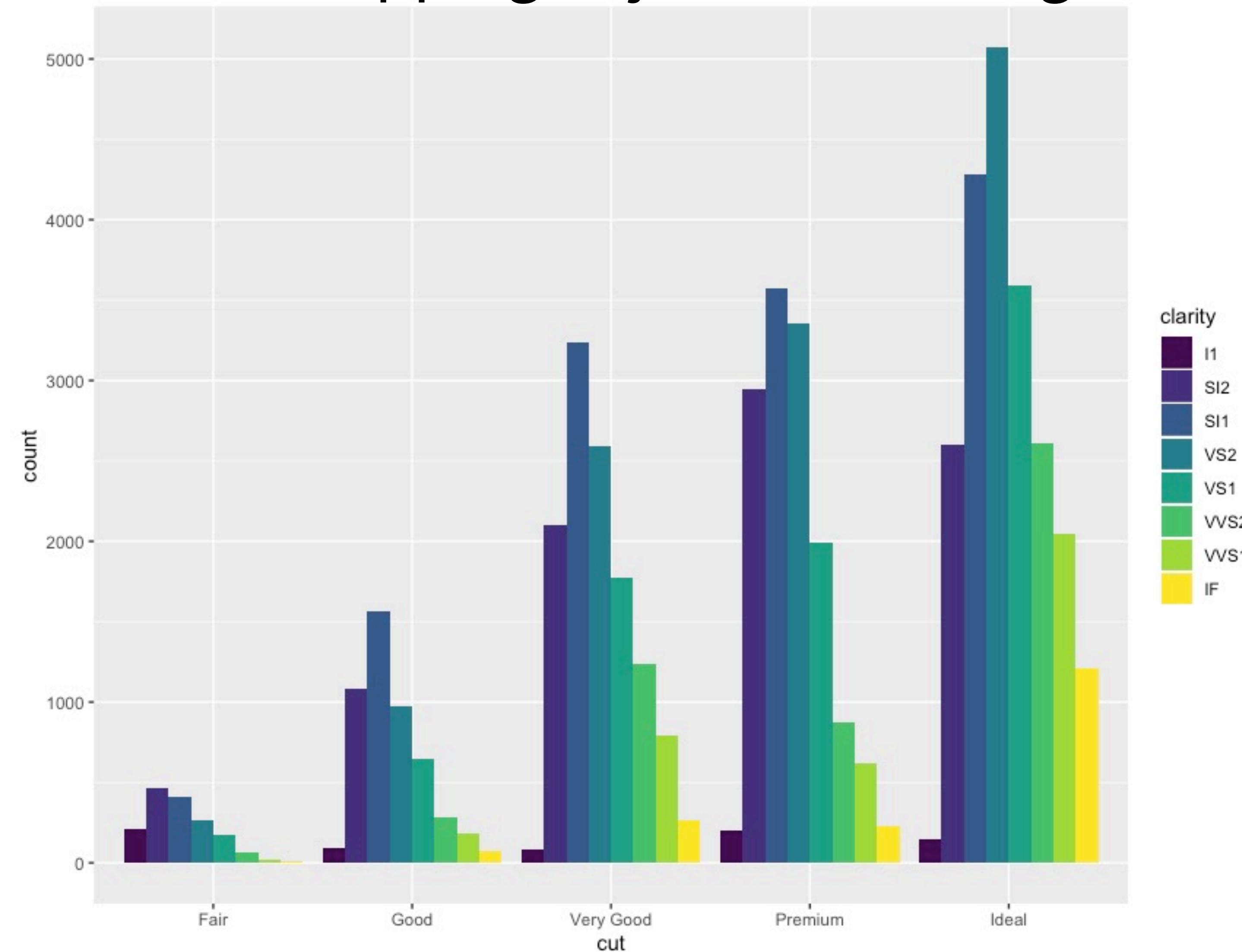
# what else?

R



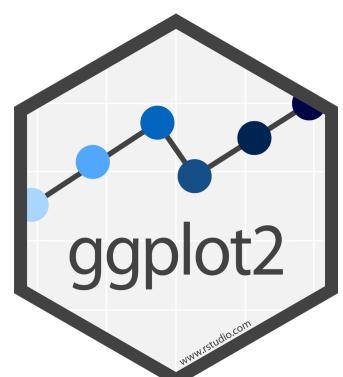
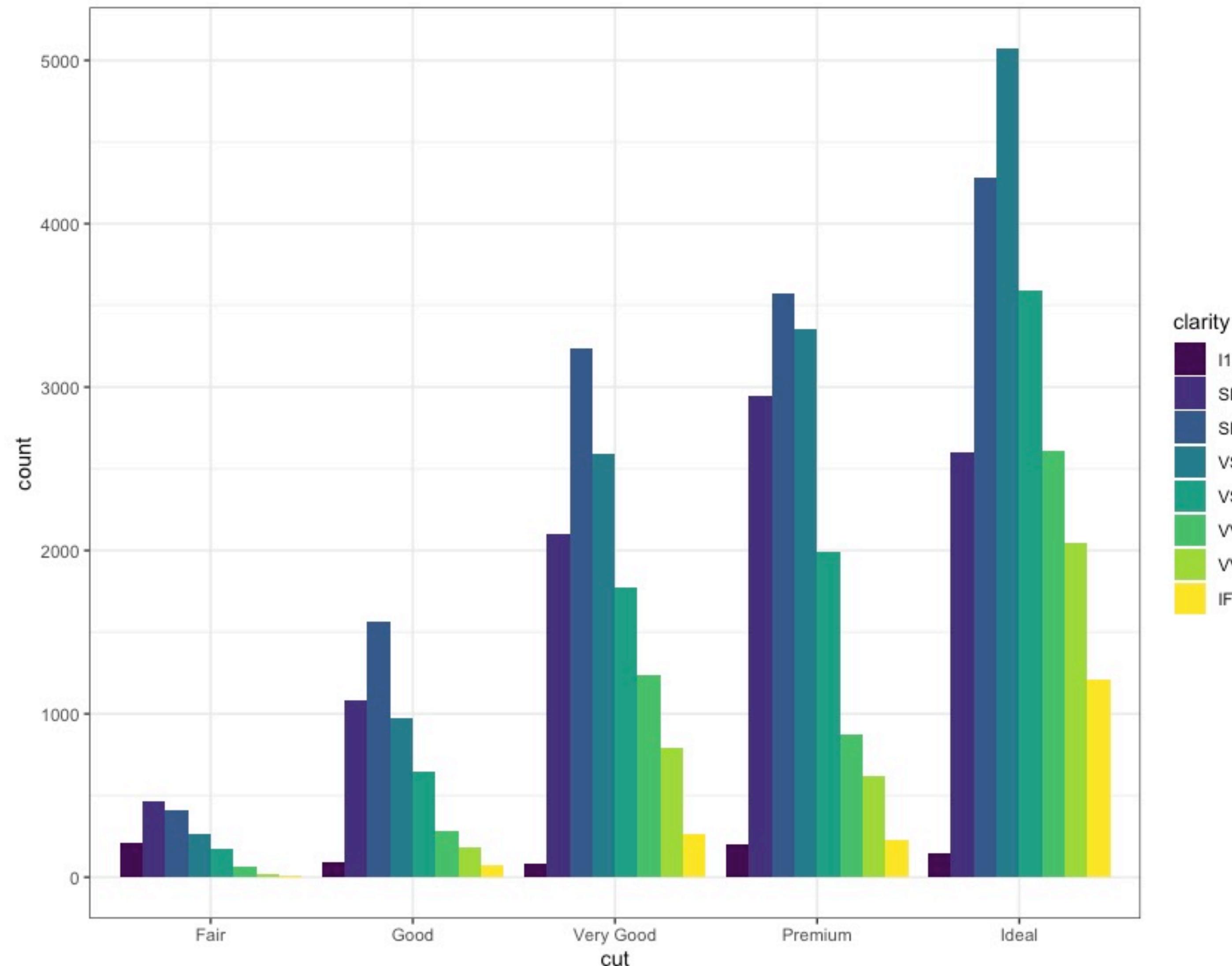
# Position Adjustments

How overlapping objects are arranged



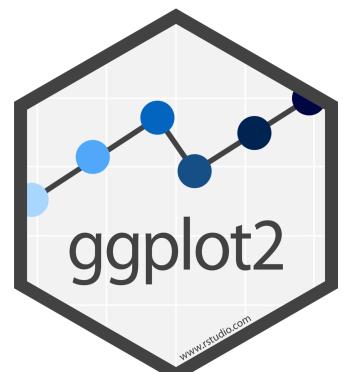
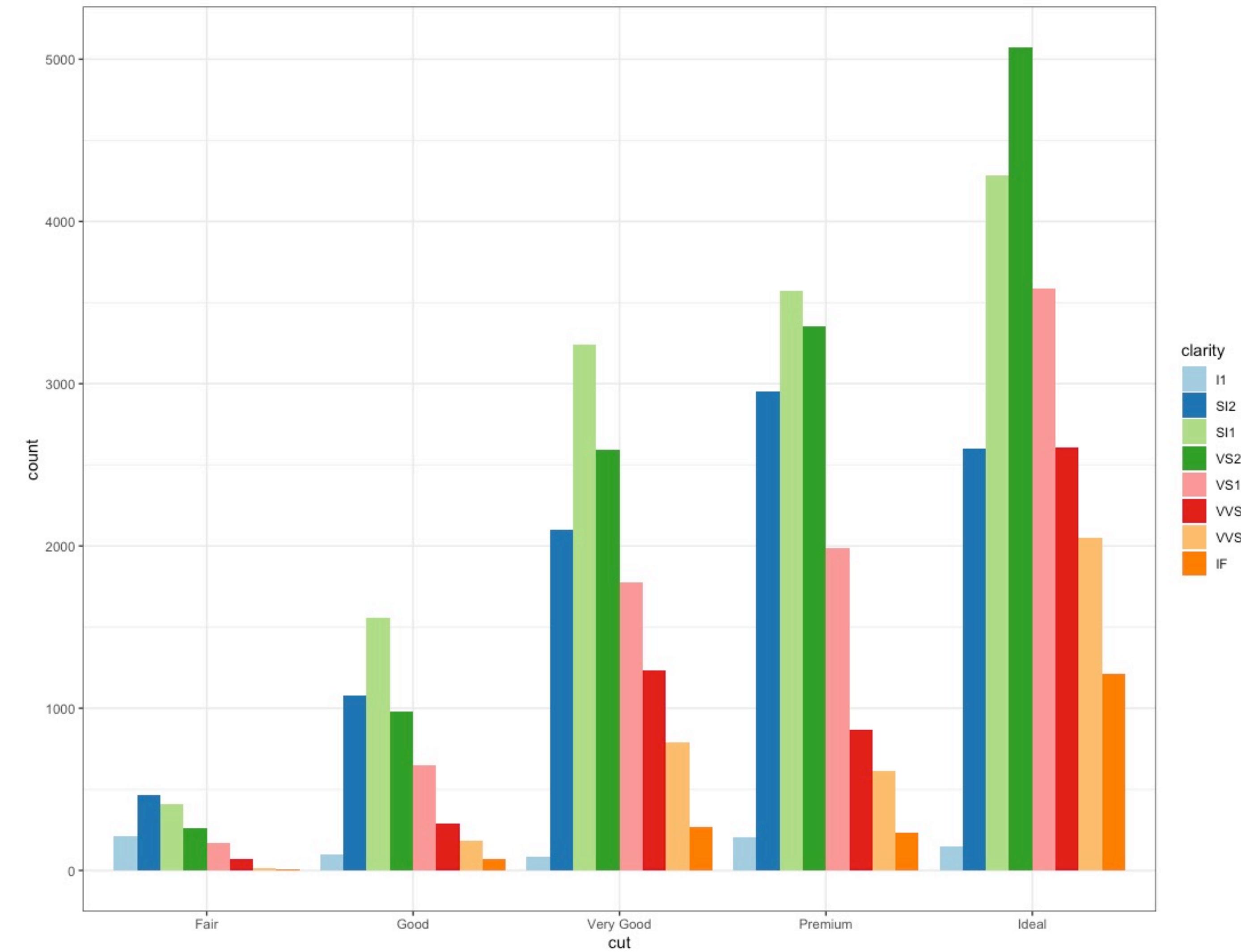
# Themes

Visual appearance of non-data elements



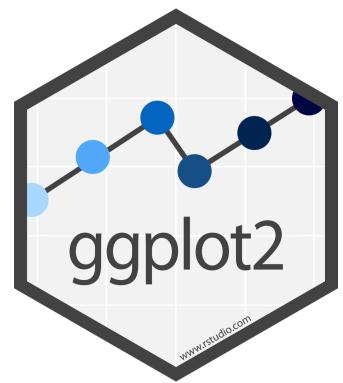
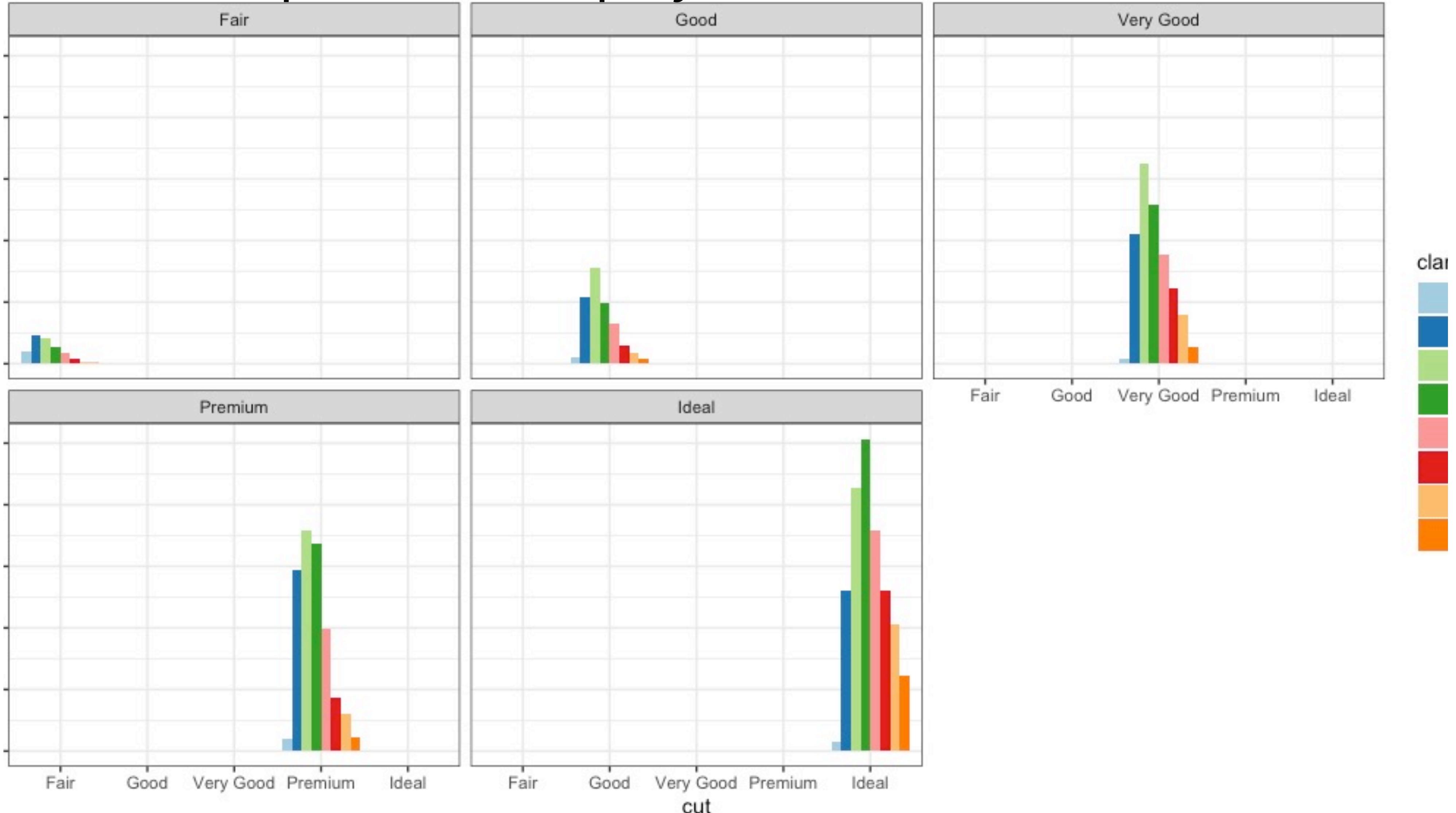
# Scales

Customize color scales, other mappings

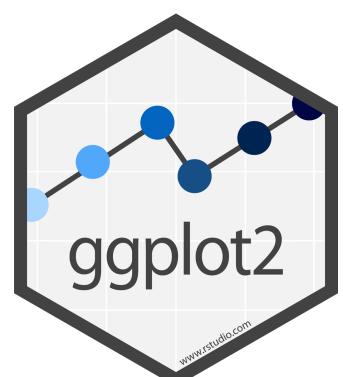
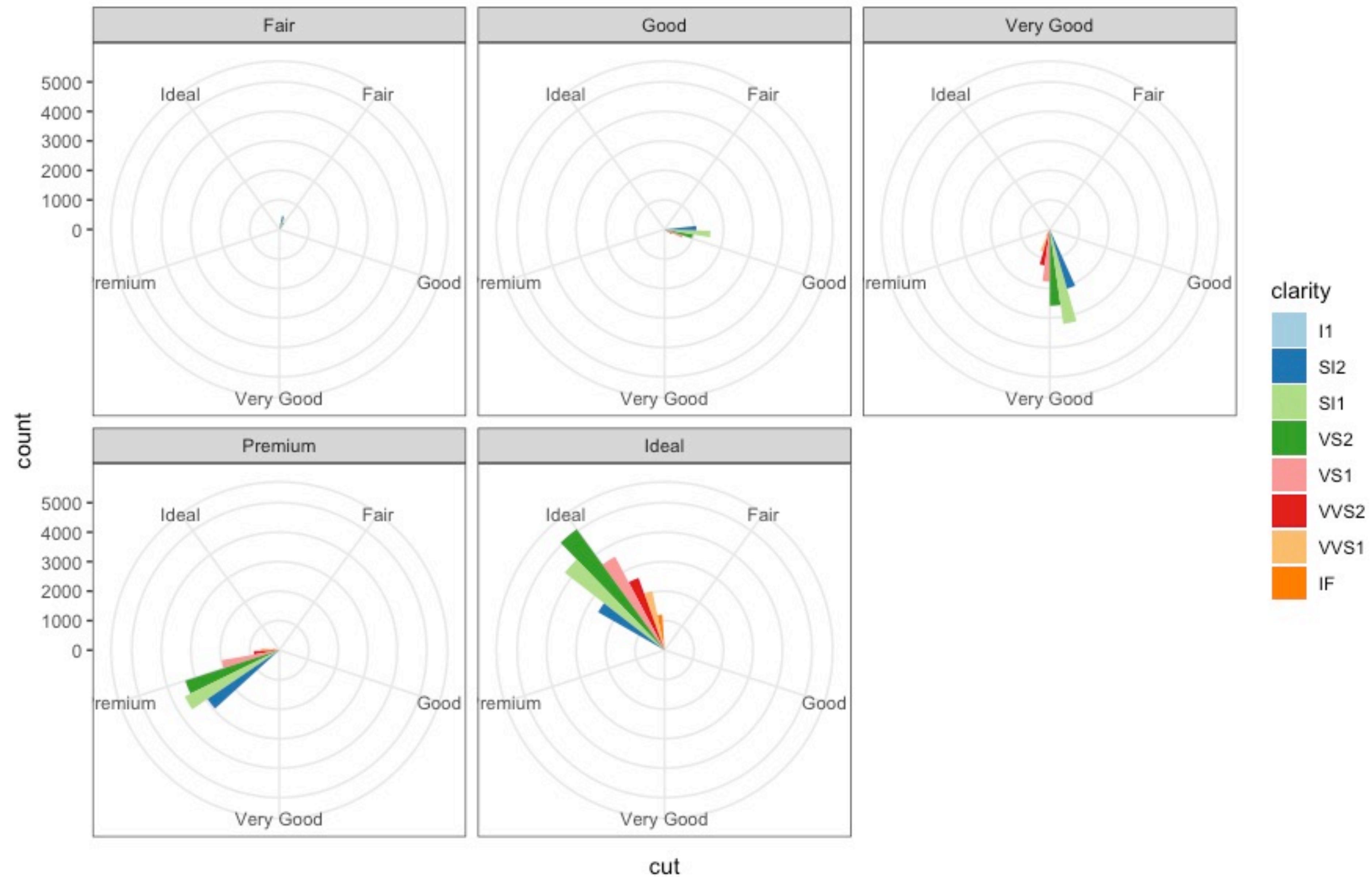


# Facets

Subplots that display subsets of the data.



# Coordinate systems

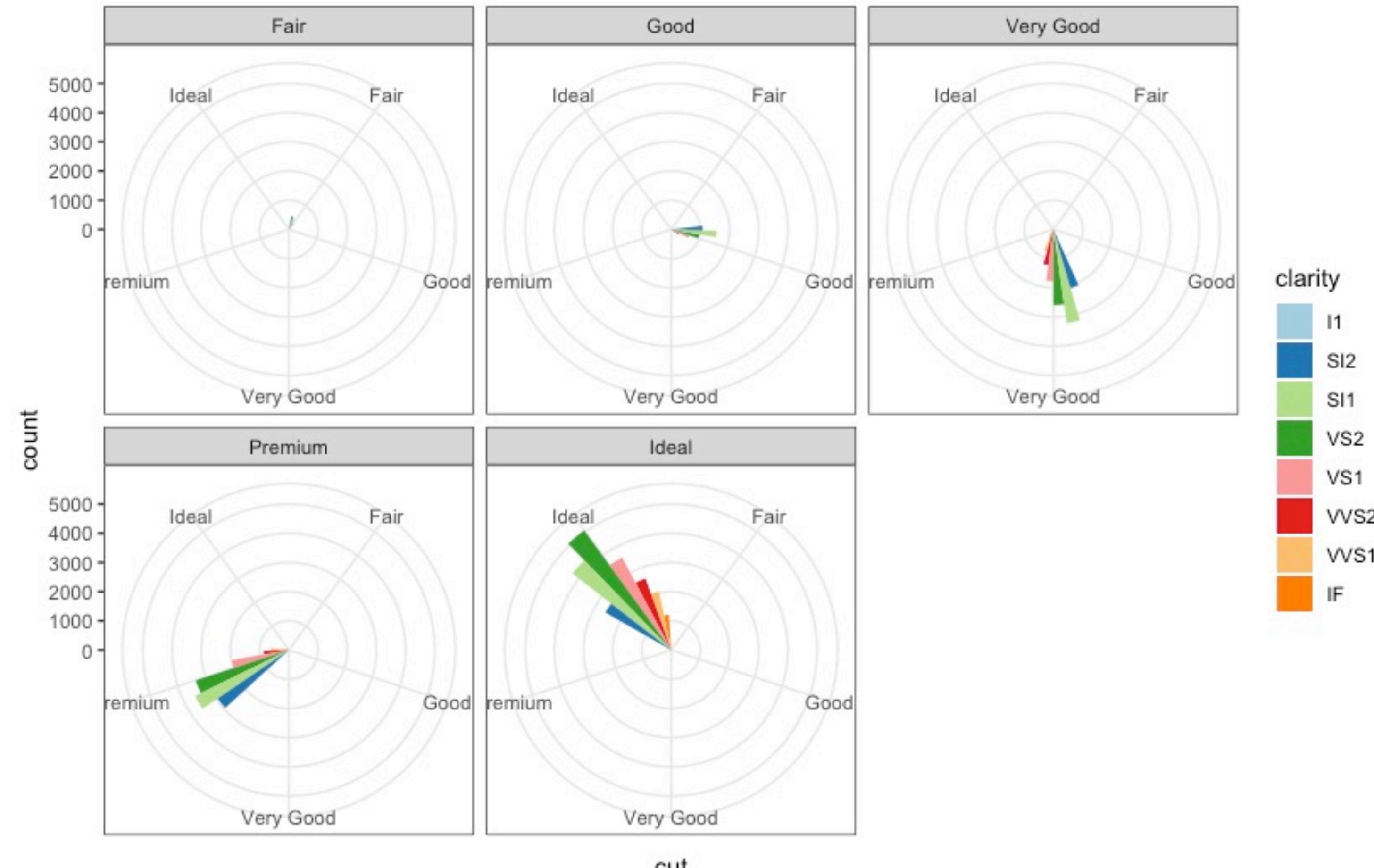


# Titles and captions

## Diamonds data

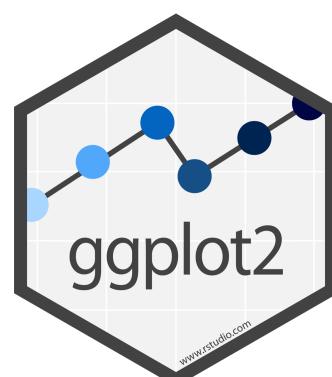
The data set is skewed towards ideal cut diamonds

The data is skewed toward ideal cut diamonds



Data by Hadley Wickham

Data by Hadley Wickham



# Aggplot2 template

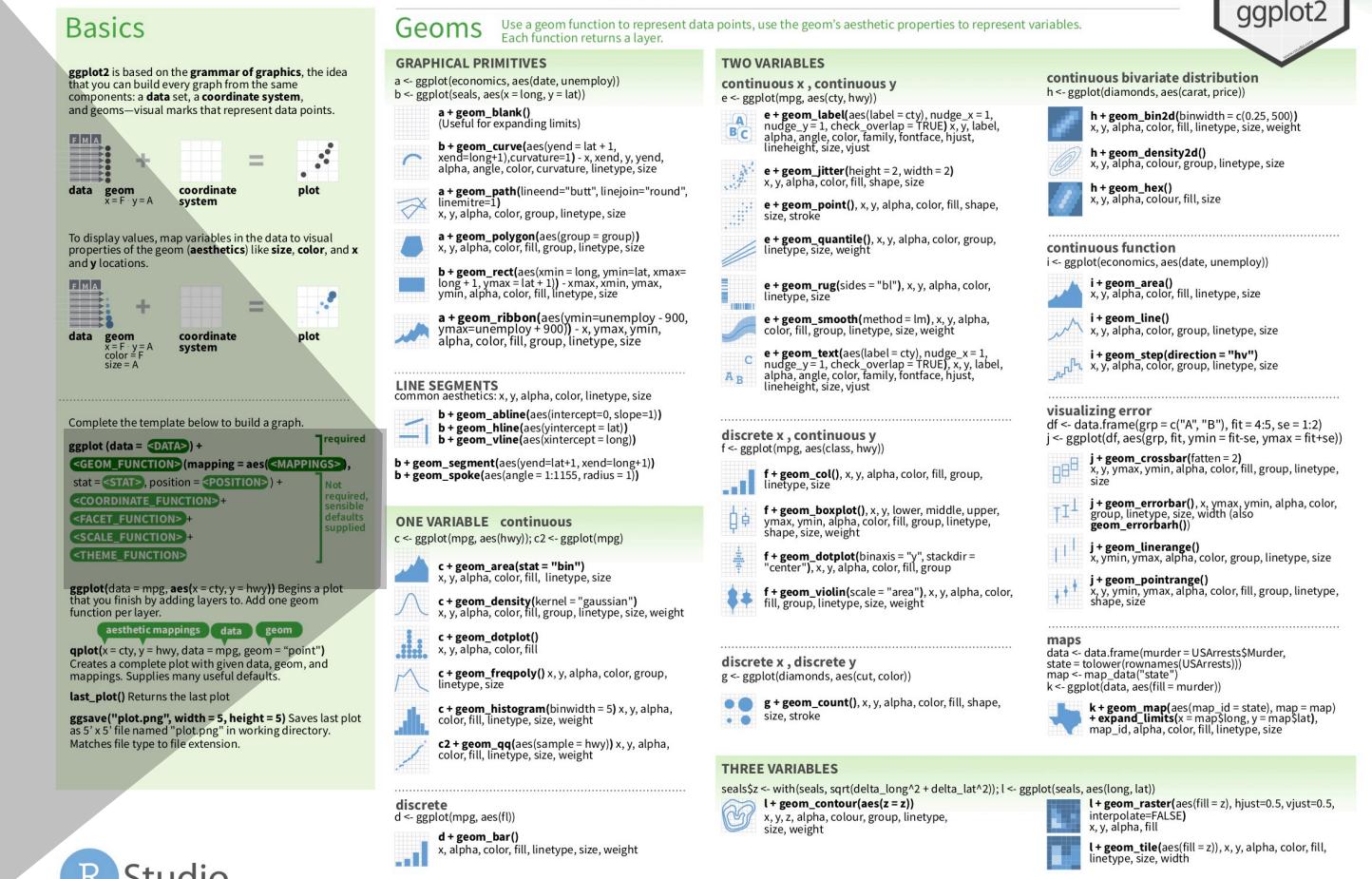
Make any plot by filling in the parameters of this template

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

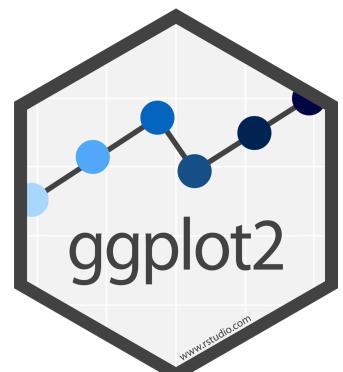
Not required,  
sensible  
defaults  
supplied

required

## Data Visualization with ggplot2 :: CHEAT SHEET



R Studio



# ggplot2.tidyverse.org

The screenshot shows a web browser window with the URL <https://ggplot2.tidyverse.org>. The page title is "Create Elegant Data Visualisation". The main content area features the ggplot2 logo and version 3.2.1. A "Usage" section describes the philosophy of ggplot2 and provides a sample R code snippet:

```
library(ggplot2)

ggplot(mpg, aes(displ, hwy, colour = class)) +
 geom_point()
```

Below the code is a scatter plot showing the relationship between engine displacement (displ) and fuel economy (hwy) by vehicle class (class). The plot includes a light gray grid.

The right sidebar lists the developers of ggplot2:

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

A small ggplot2 logo is located in the bottom right corner of the page.