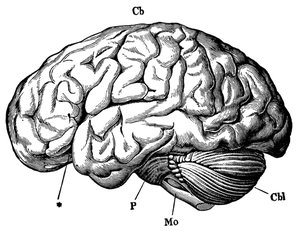
ggplot2

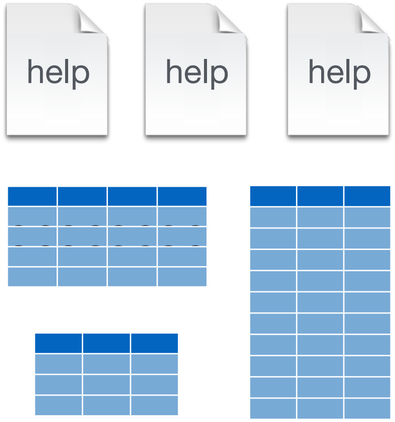
R - A computer language for scientists



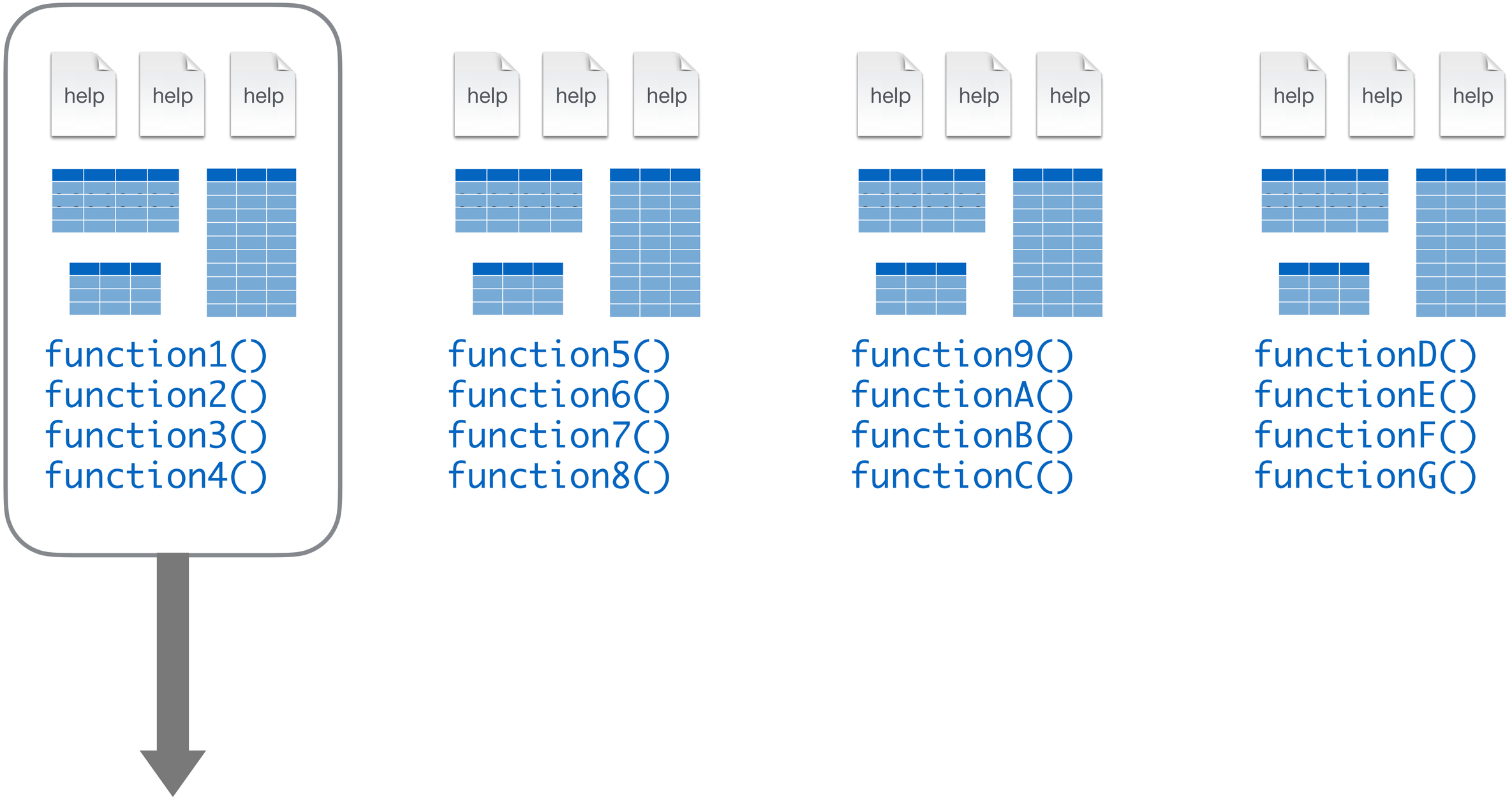
Human thought

C++

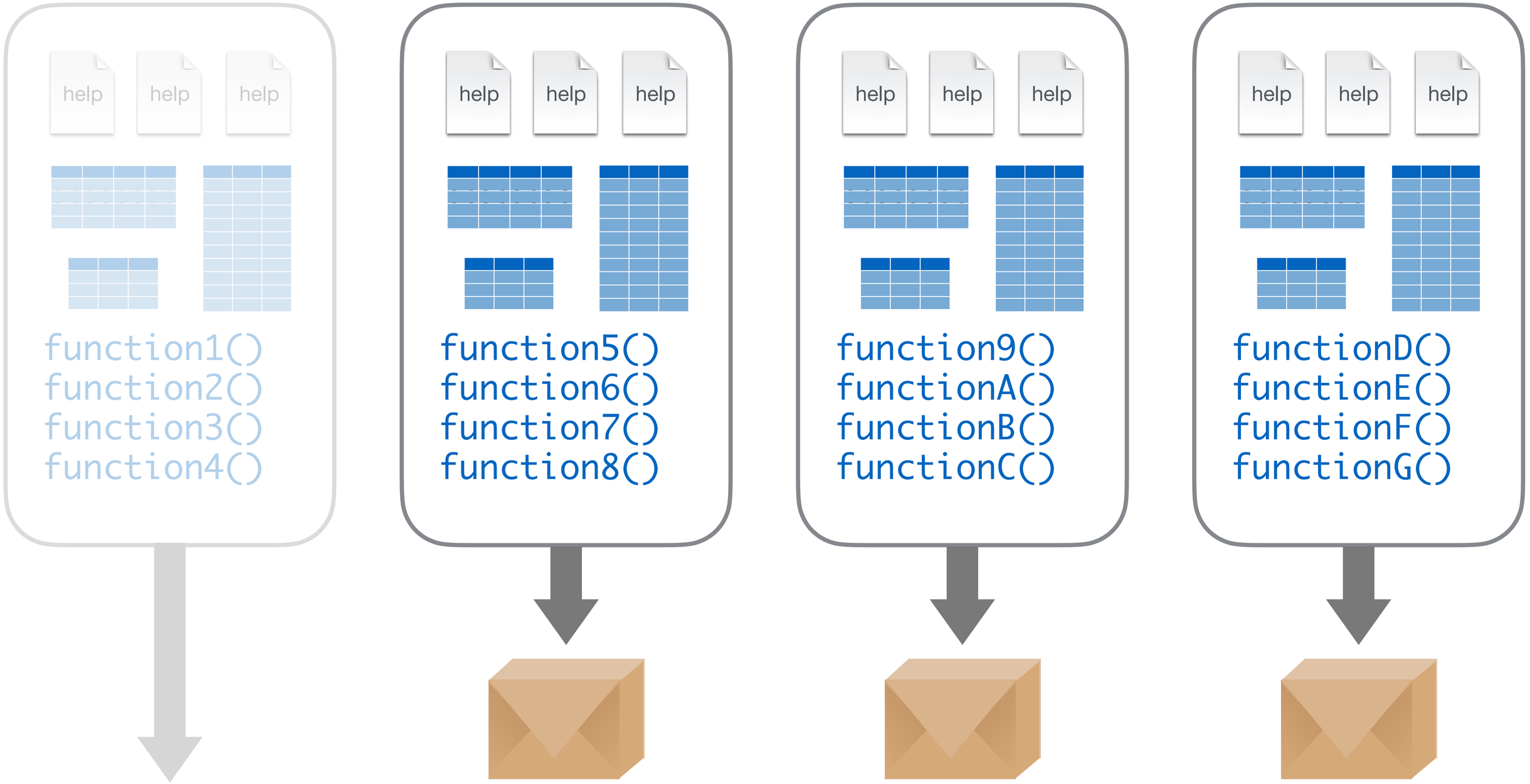
Machine language




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

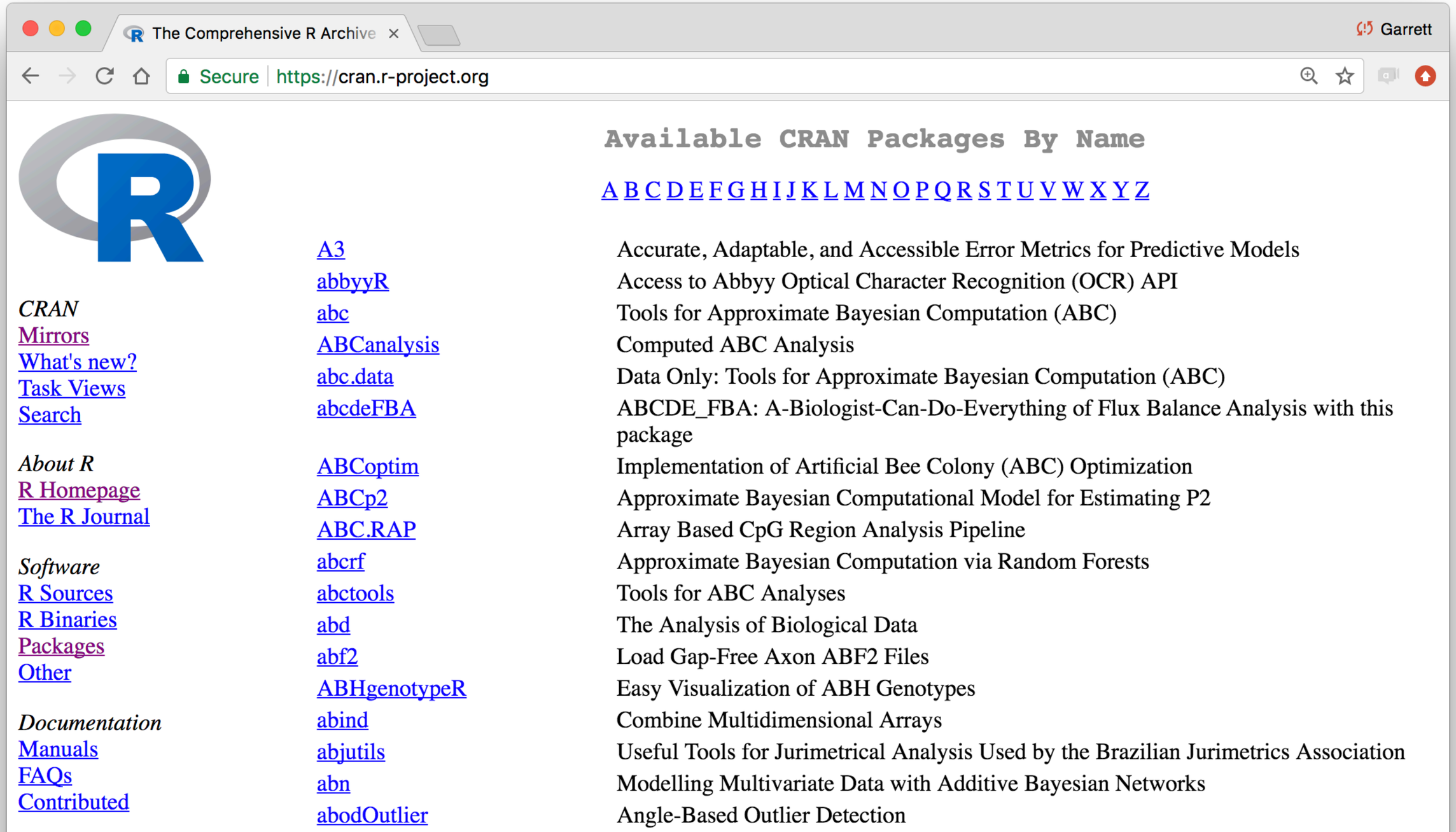

help
help
help
function1()
function2()
function3()
function4()
help
help
help
function5()
function6()
function7()
function8()
help
help
help
function9()
functionA()
functionB()
functionC()
help
help
help
functionD()
functionE()
functionF()
functionG()

Base R



Base R

R Packages



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

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

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
ABCanalysis	Computed ABC Analysis
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCoptim	Implementation of Artificial Bee Colony (ABC) Optimization
ABCp2	Approximate Bayesian Computational Model for Estimating P2
ABC.RAP	Array Based CpG Region Analysis Pipeline
abcrf	Approximate Bayesian Computation via Random Forests
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abf2	Load Gap-Free Axon ABF2 Files
ABHgenotypeR	Easy Visualization of ABH Genotypes
abind	Combine Multidimensional Arrays
abjutils	Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association
abn	Modelling Multivariate Data with Additive Bayesian Networks
abodOutlier	Angle-Based Outlier Detection

Using packages

1

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

Downloads files to "computer"

1 x per "computer"

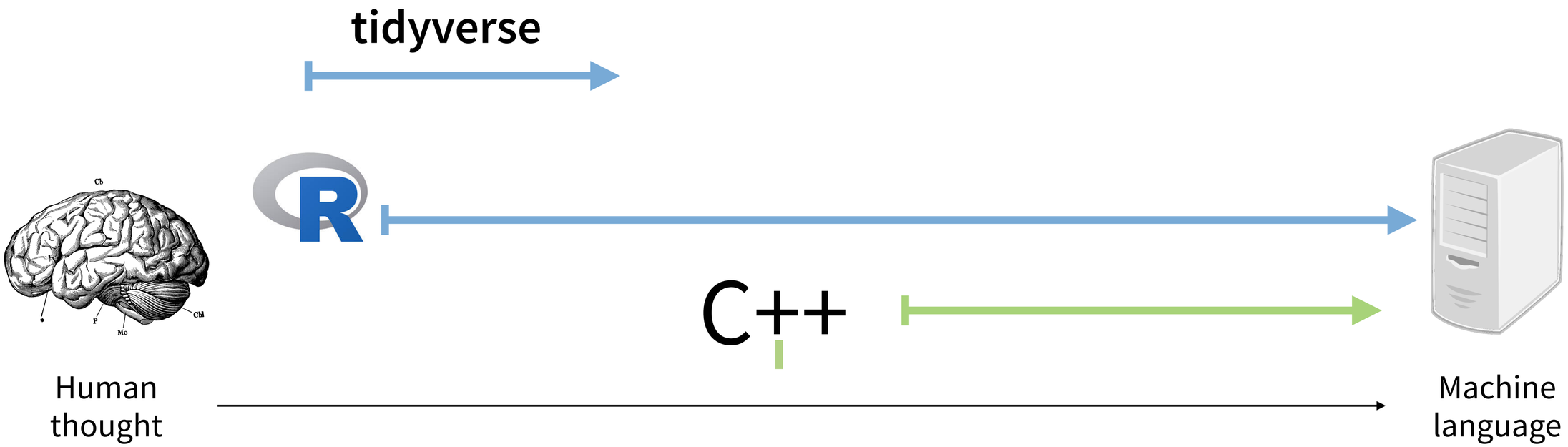
2

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

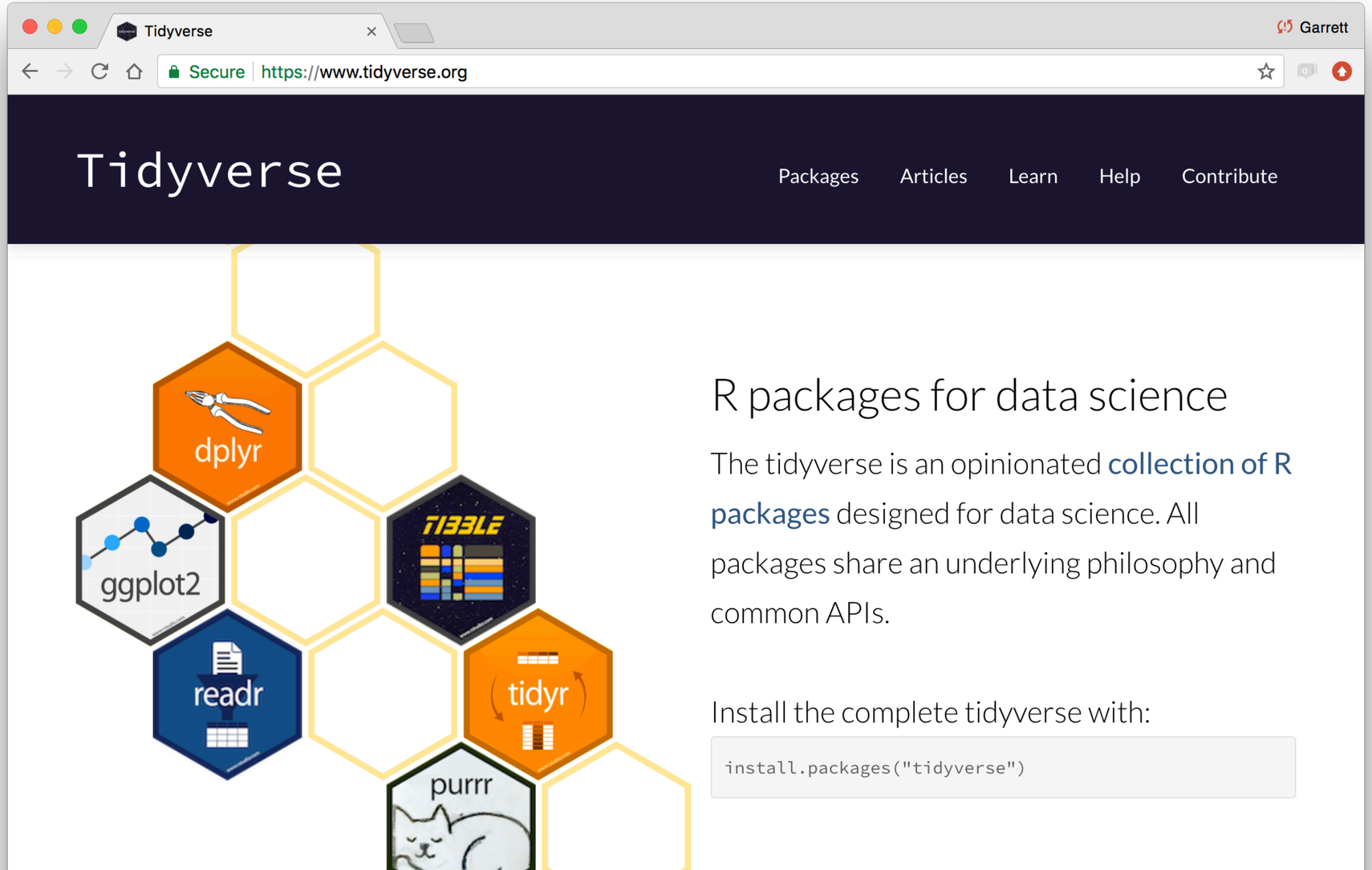
Loads package

1 x per R Session

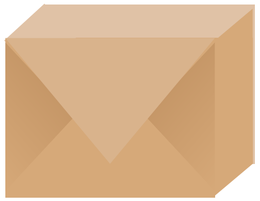
The tidyverse - A set of R packages to unify some data science tasks



tidyverse.org



tidyverse



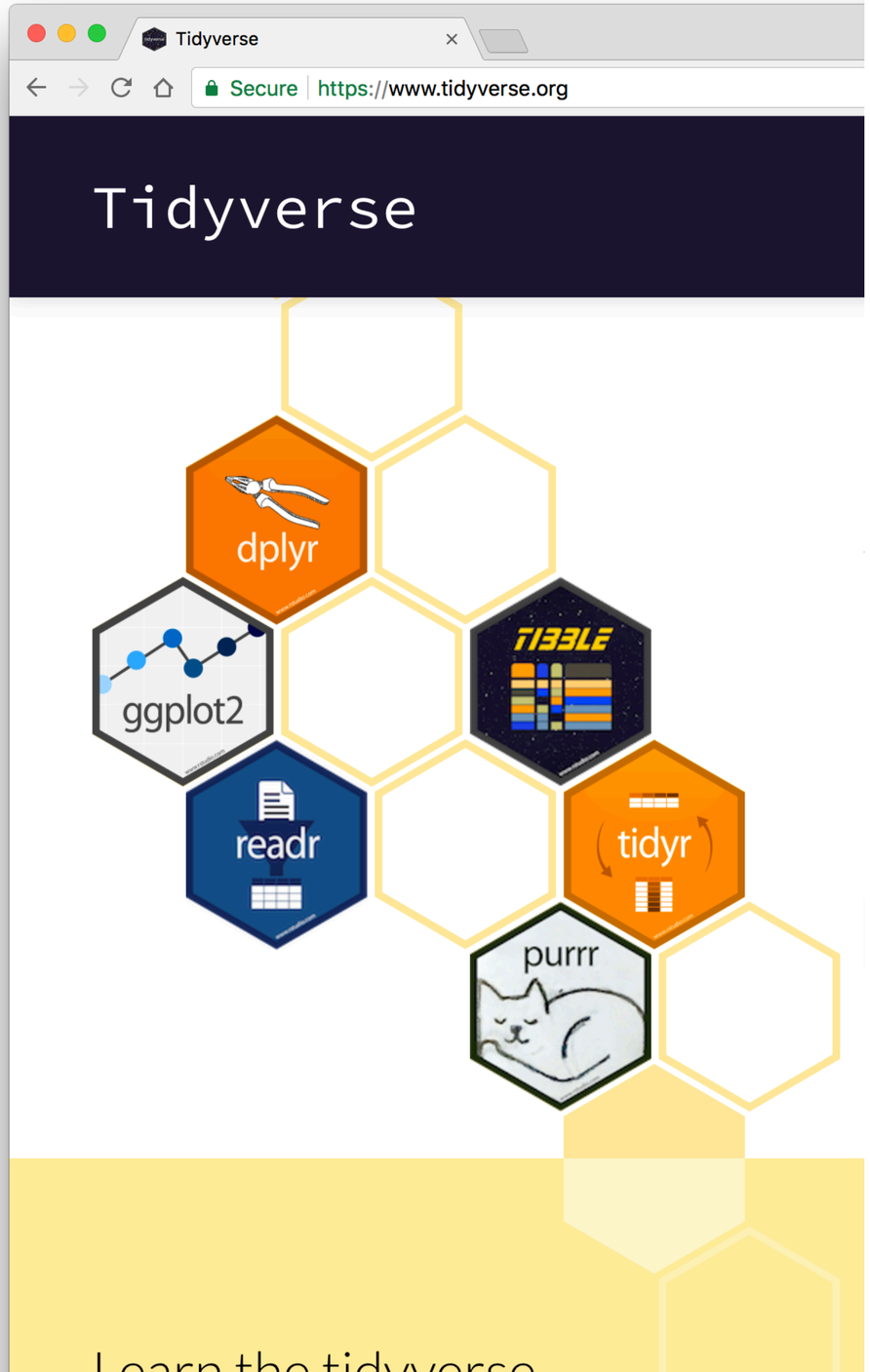
An R package that serves as a short cut for installing and loading the components of the tidyverse.

```
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")
```



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

does the equivalent of

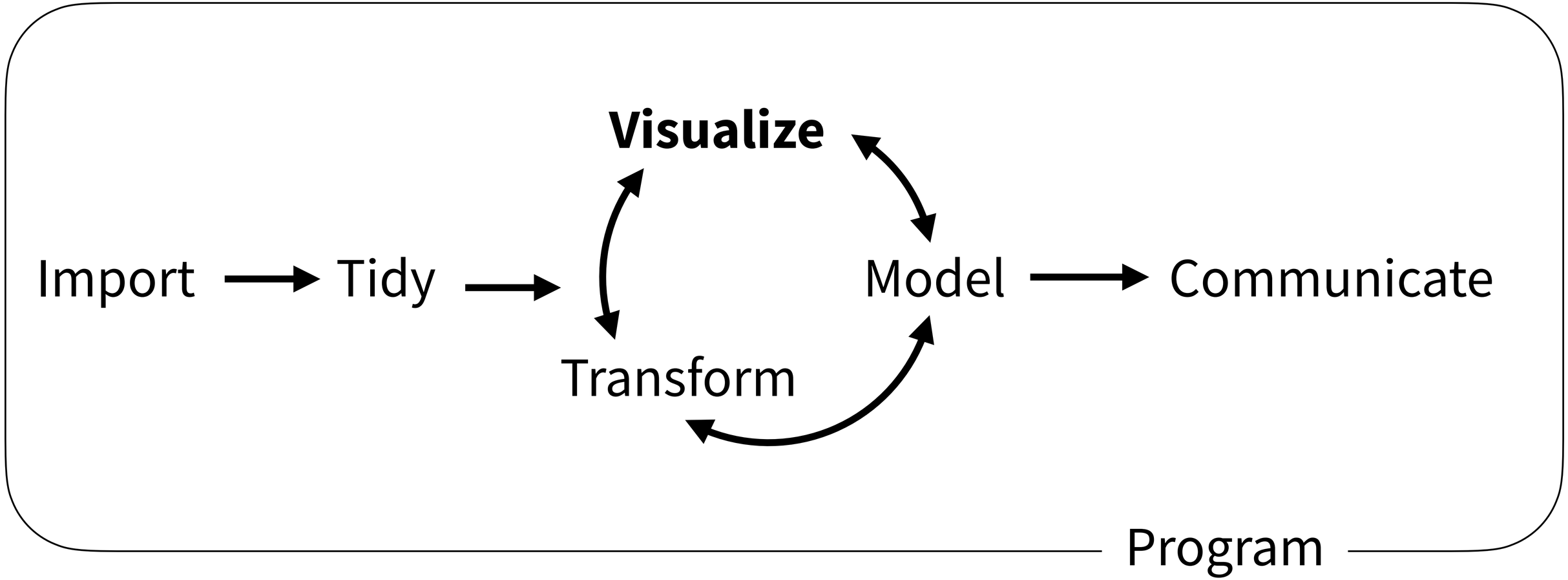
```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
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

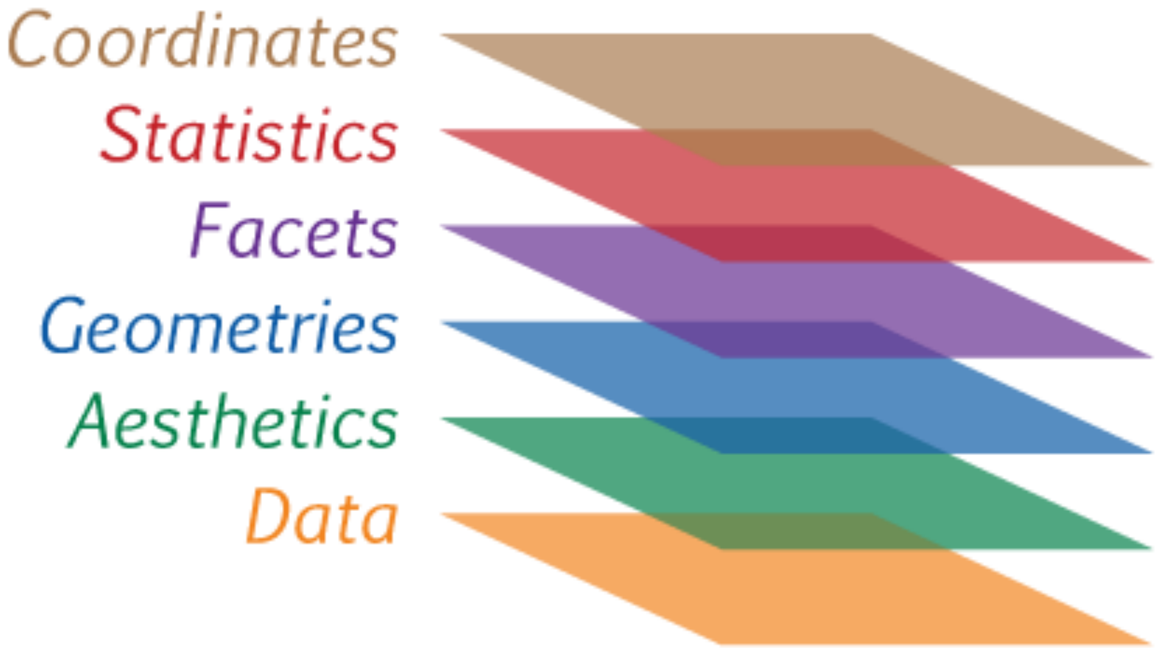
```
library("ggplot2")
library("dplyr")
library("tidyr")
library("readr")
library("purrr")
library("tibble")
```

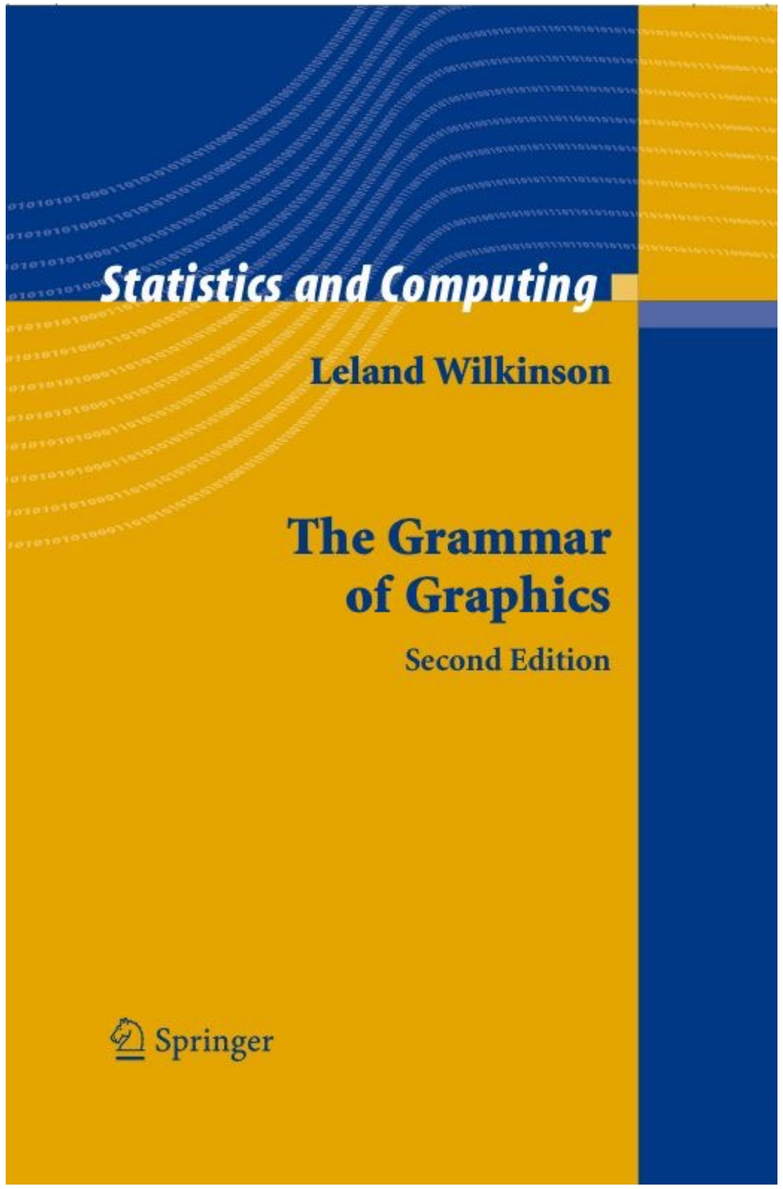
(Applied) **Data Science**



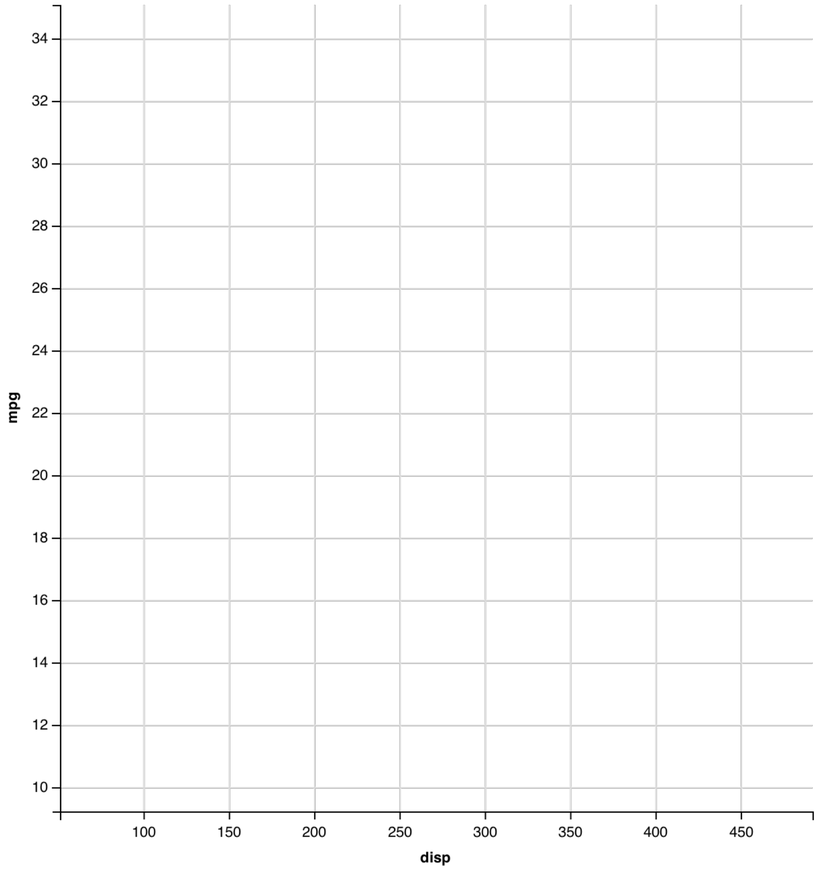
Grammar of Graphics

The Grammar of Graphics





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

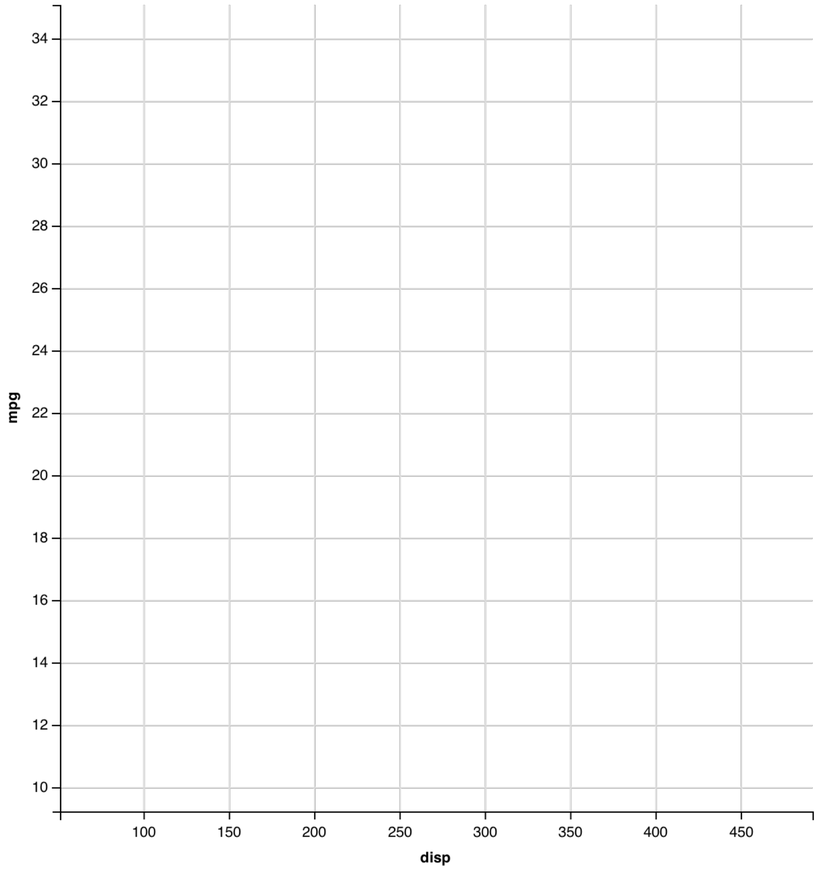


data geom

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



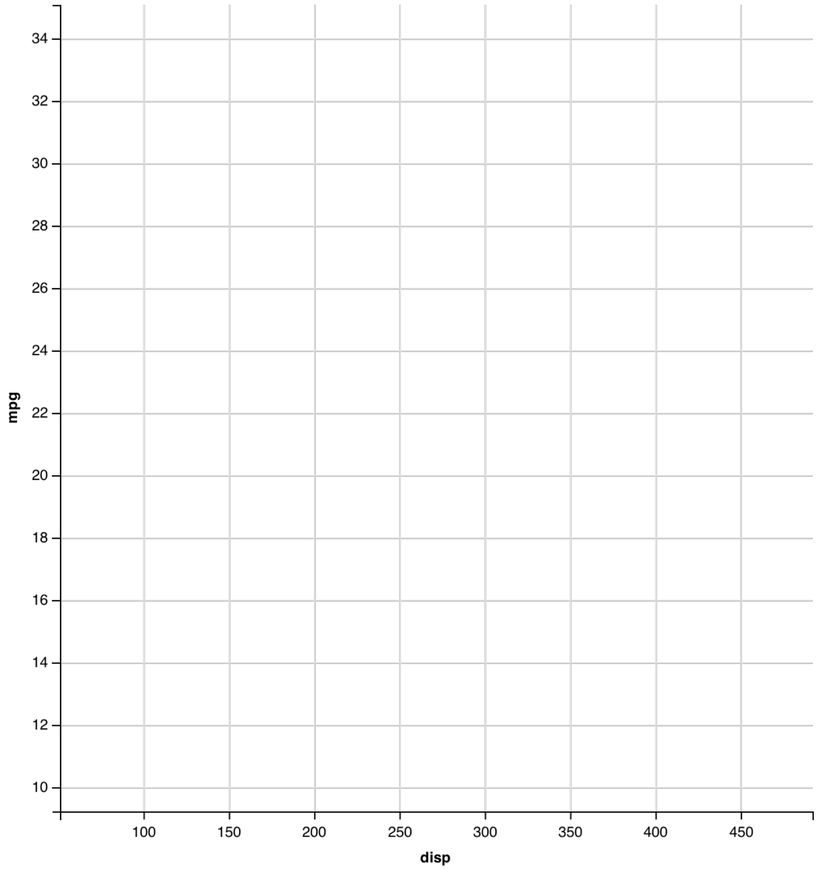
data geom

ggplot2

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

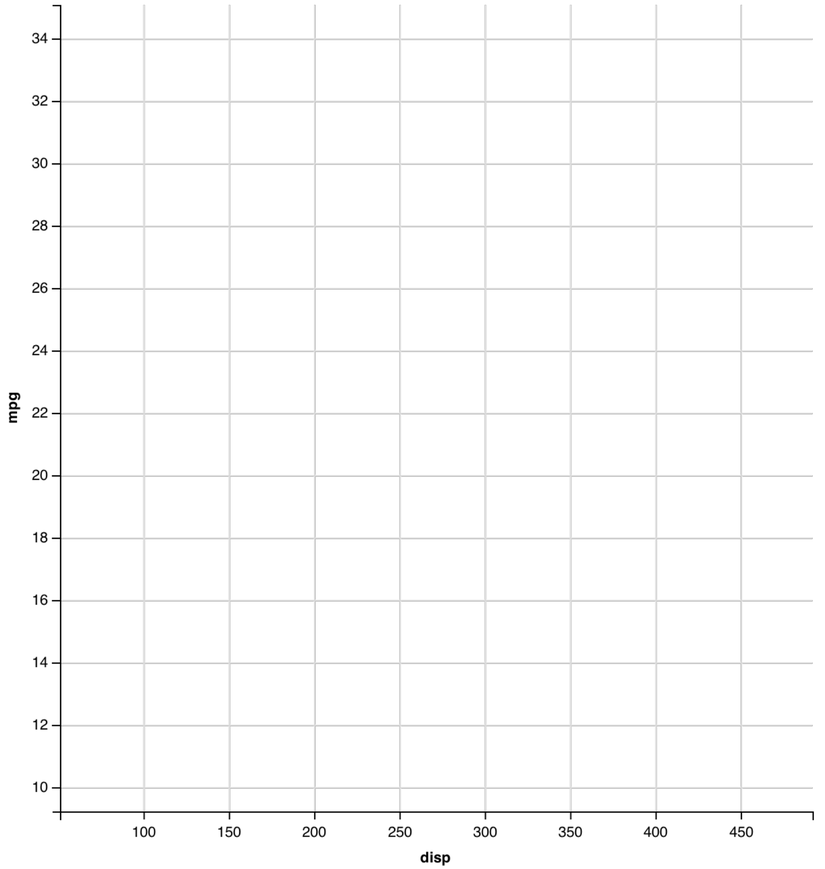


data geom

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

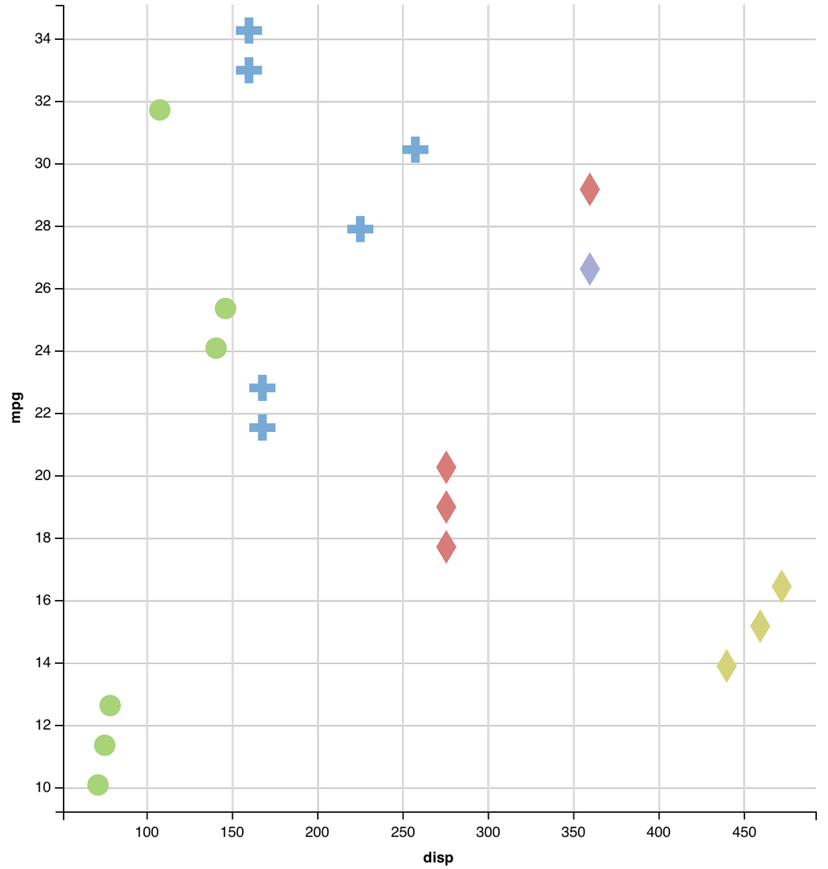


data geom

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



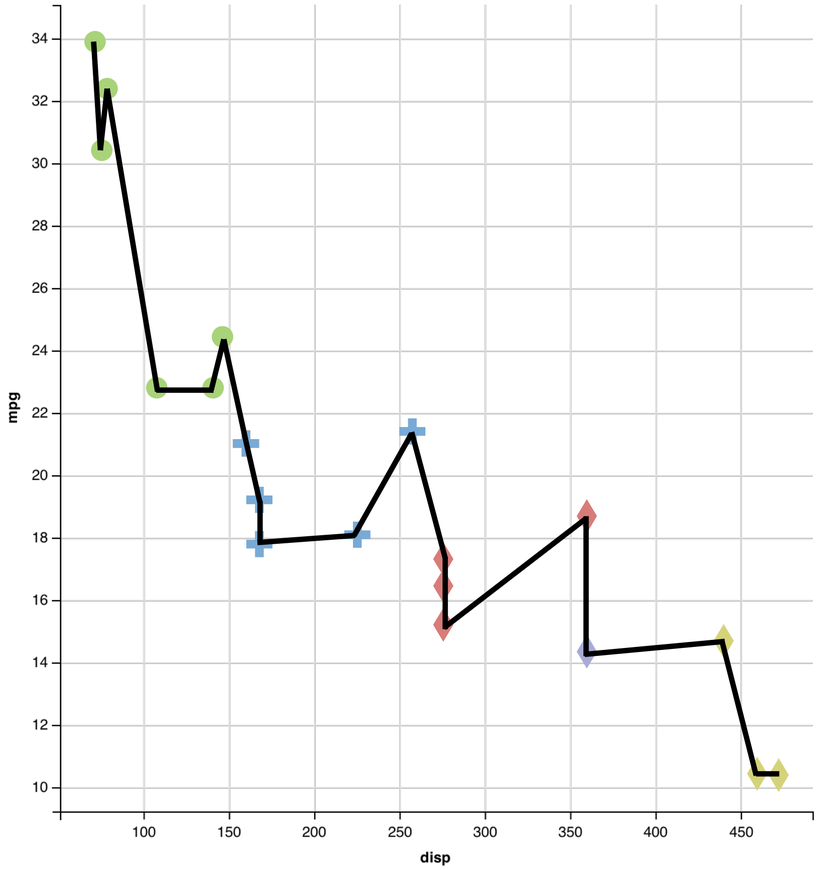
data geom

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

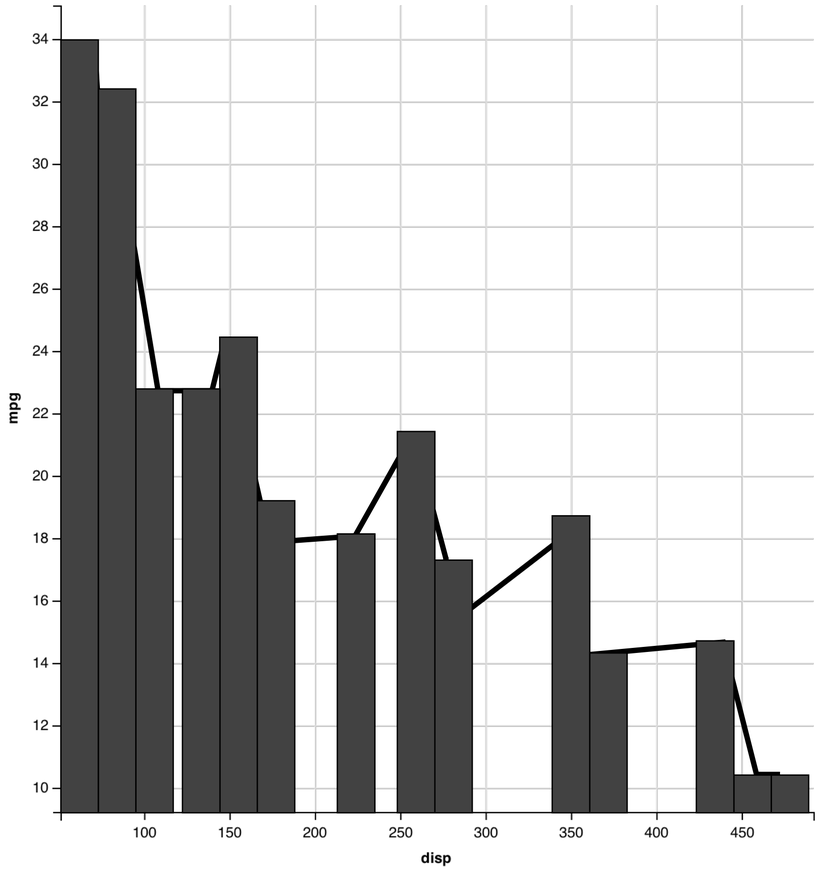
lines

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

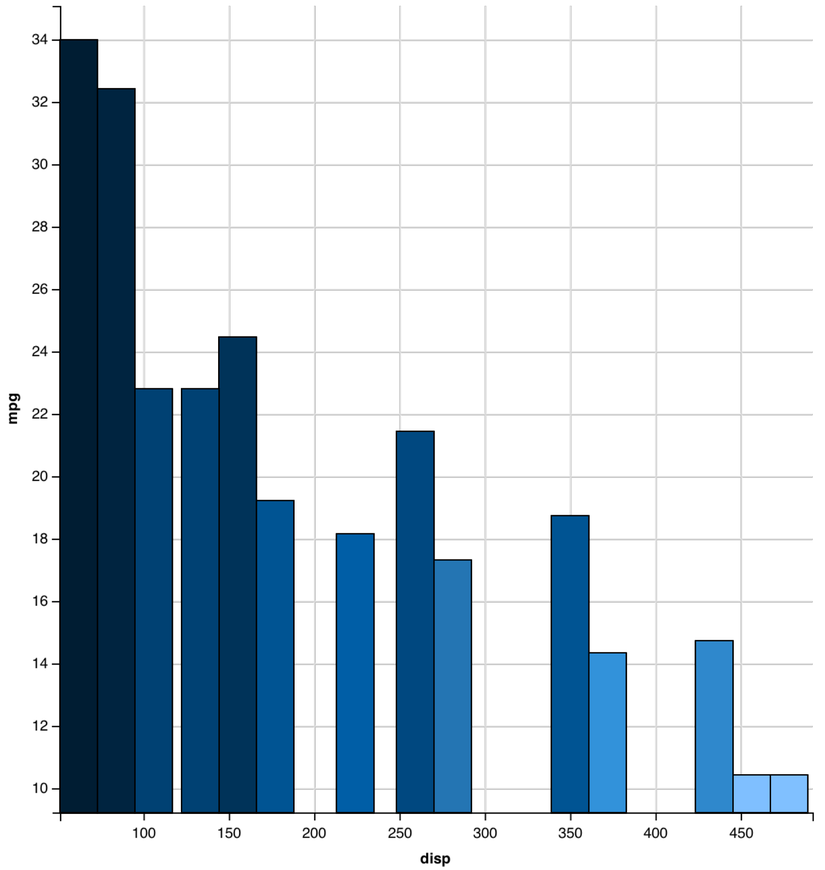
ggplot2

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

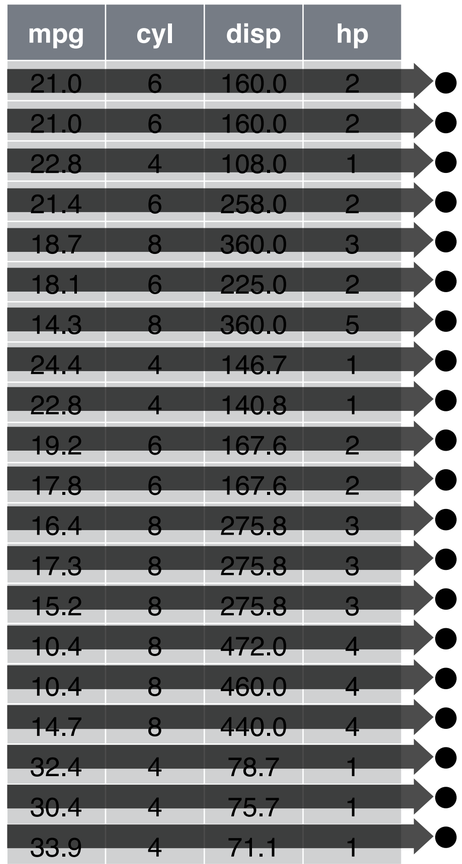
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

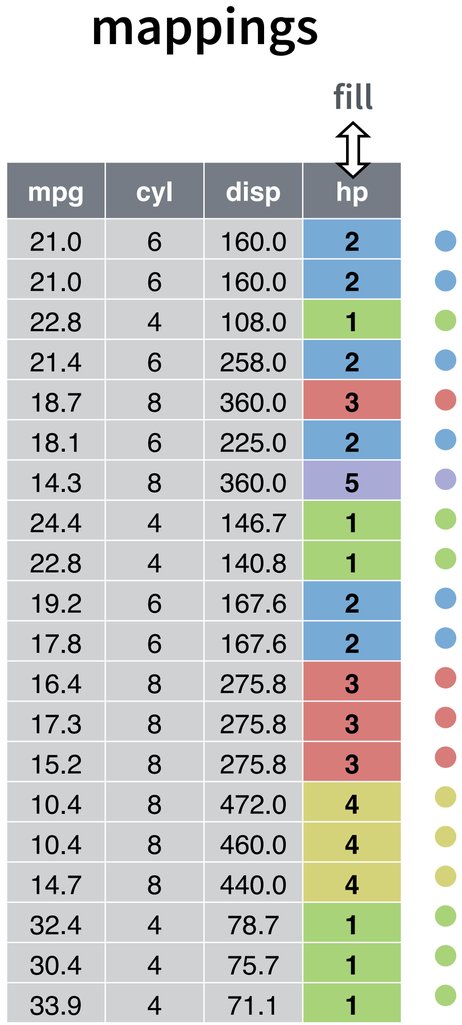
data **geom**

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

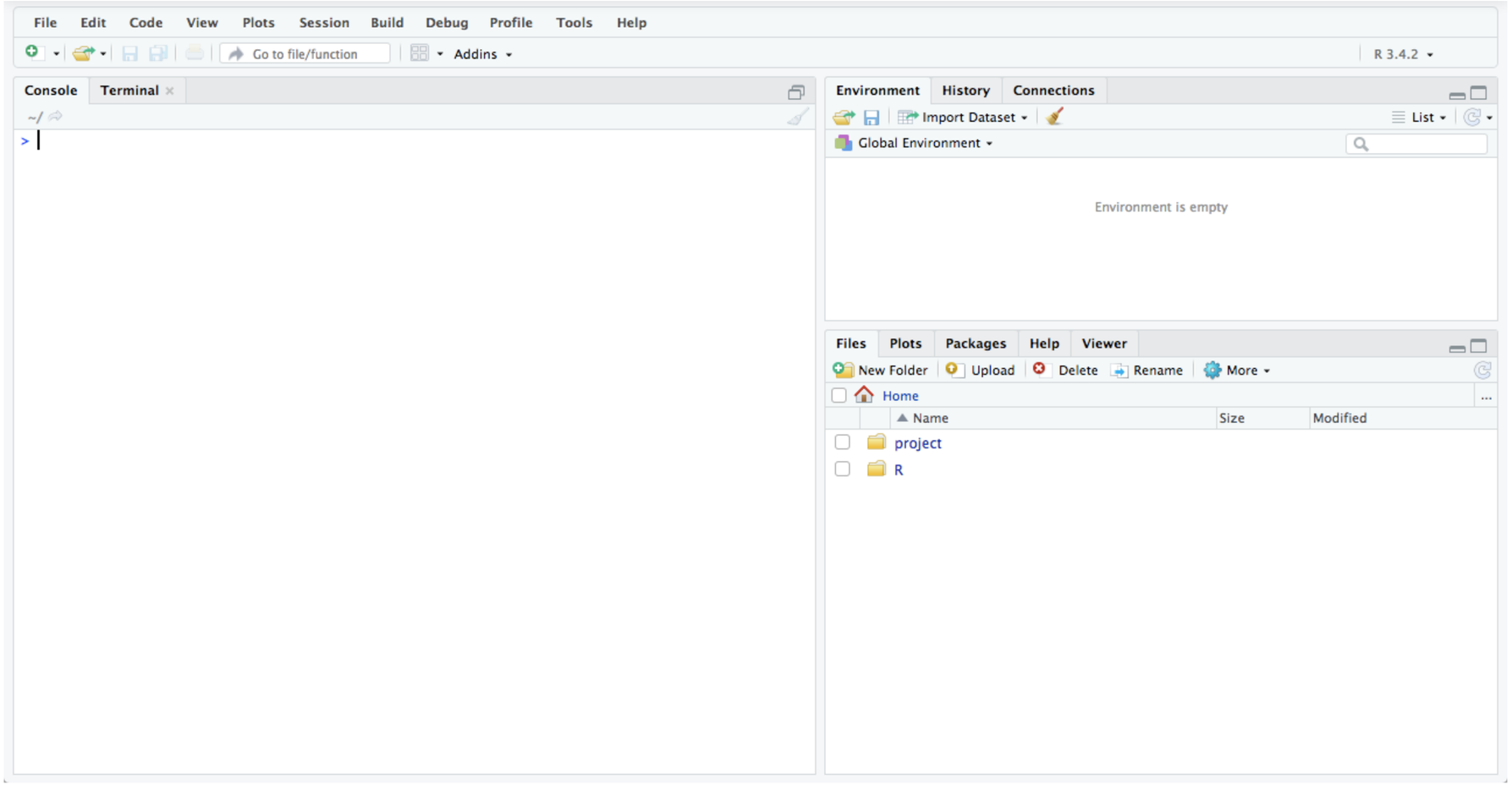
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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

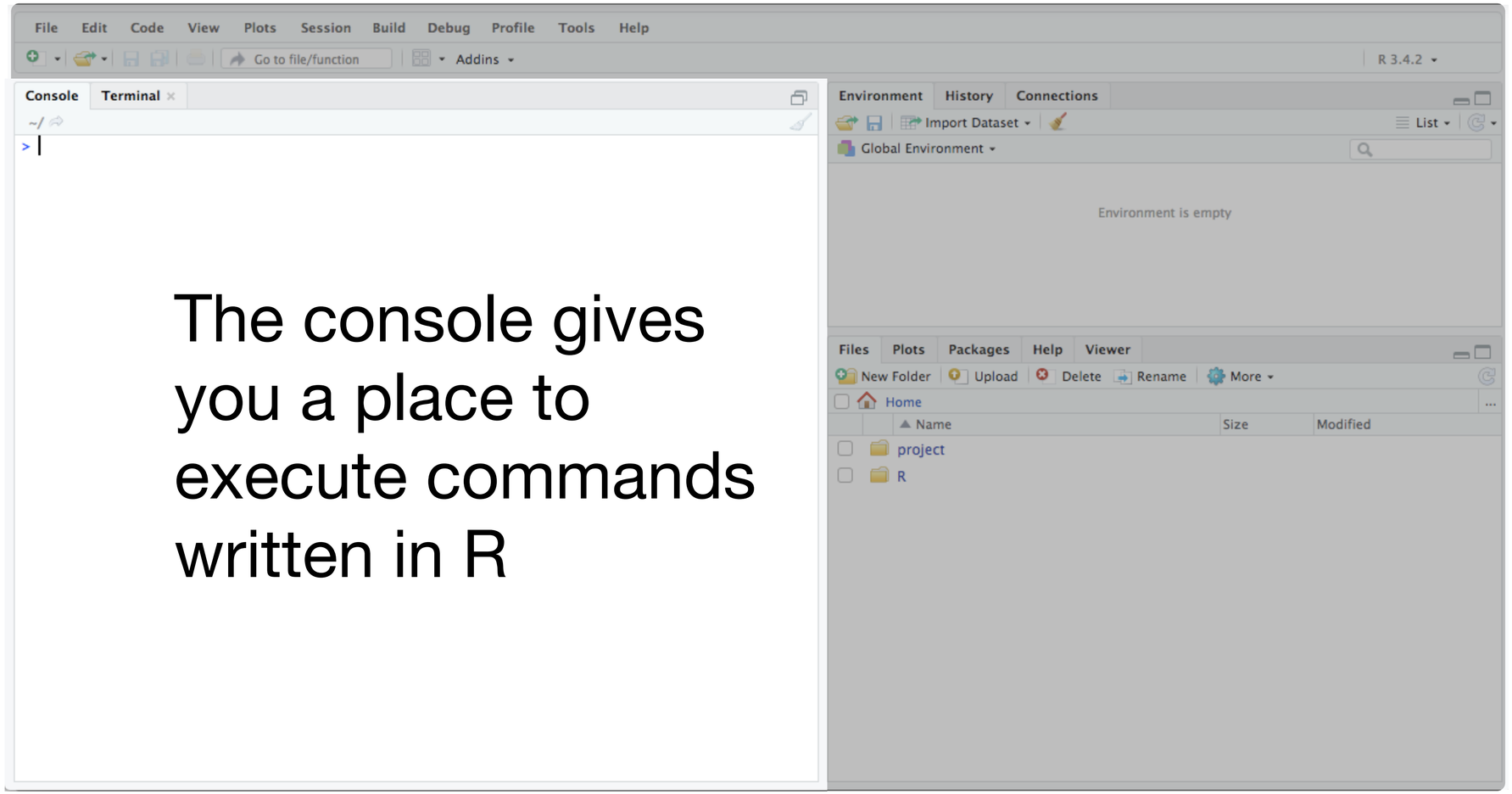
3. **Map** aesthetic properties to variables

RStudio
(let's start!)

RStudio

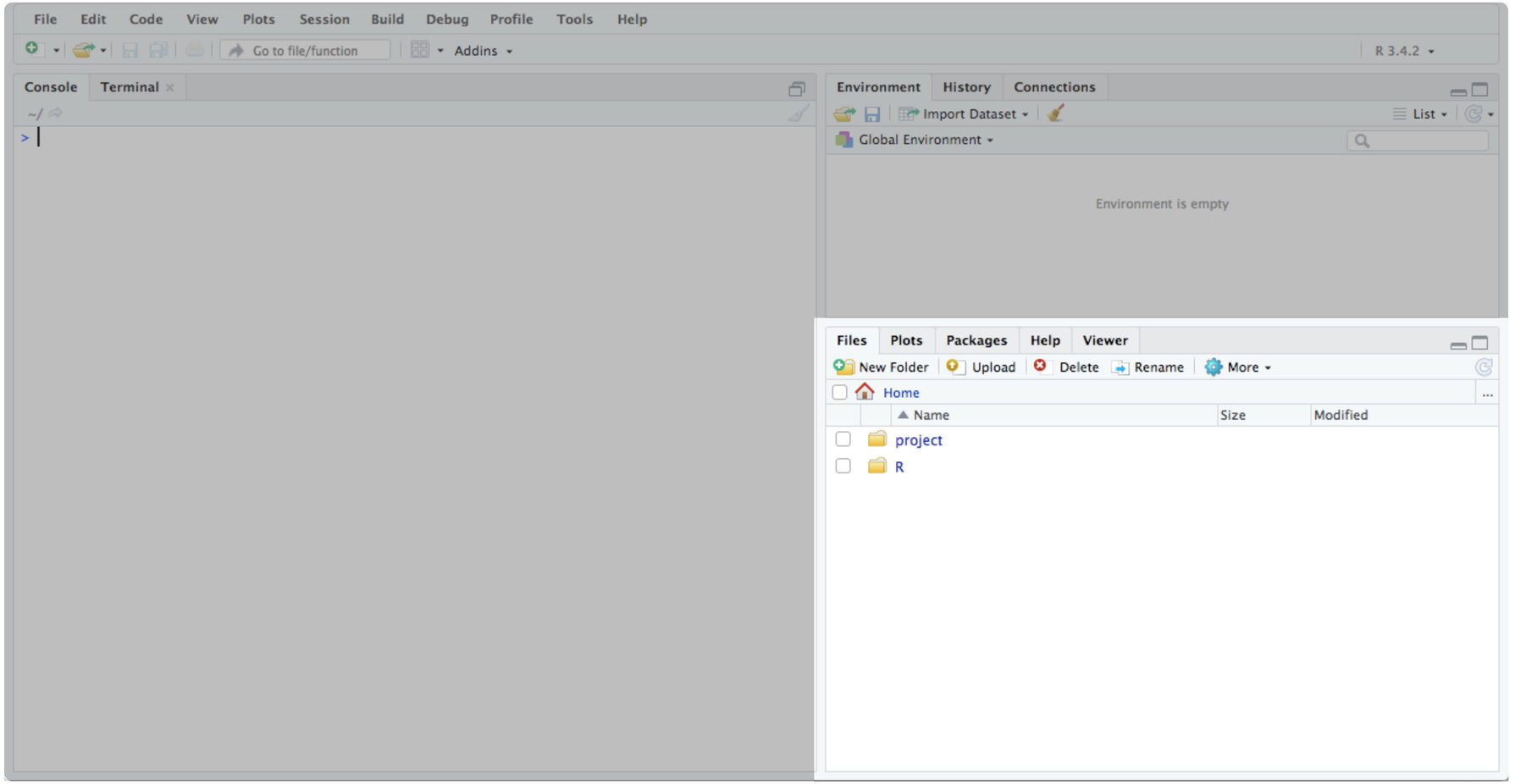

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.4.2
Console
Terminal
~/
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Upload
Delete
Rename
More
Home
Name
Size
Modified
project
R

RStudio

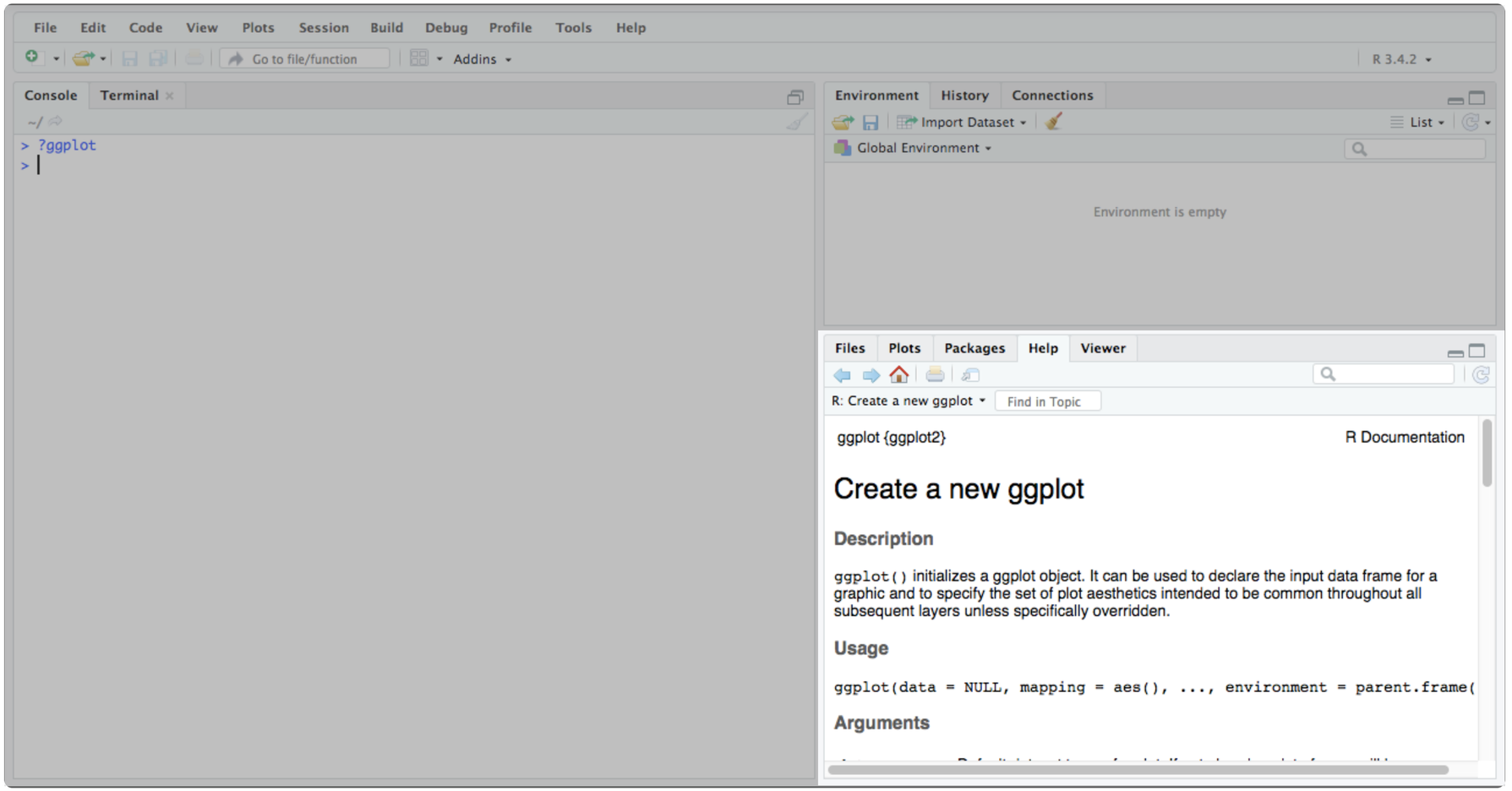


The console gives you a place to execute commands written in R

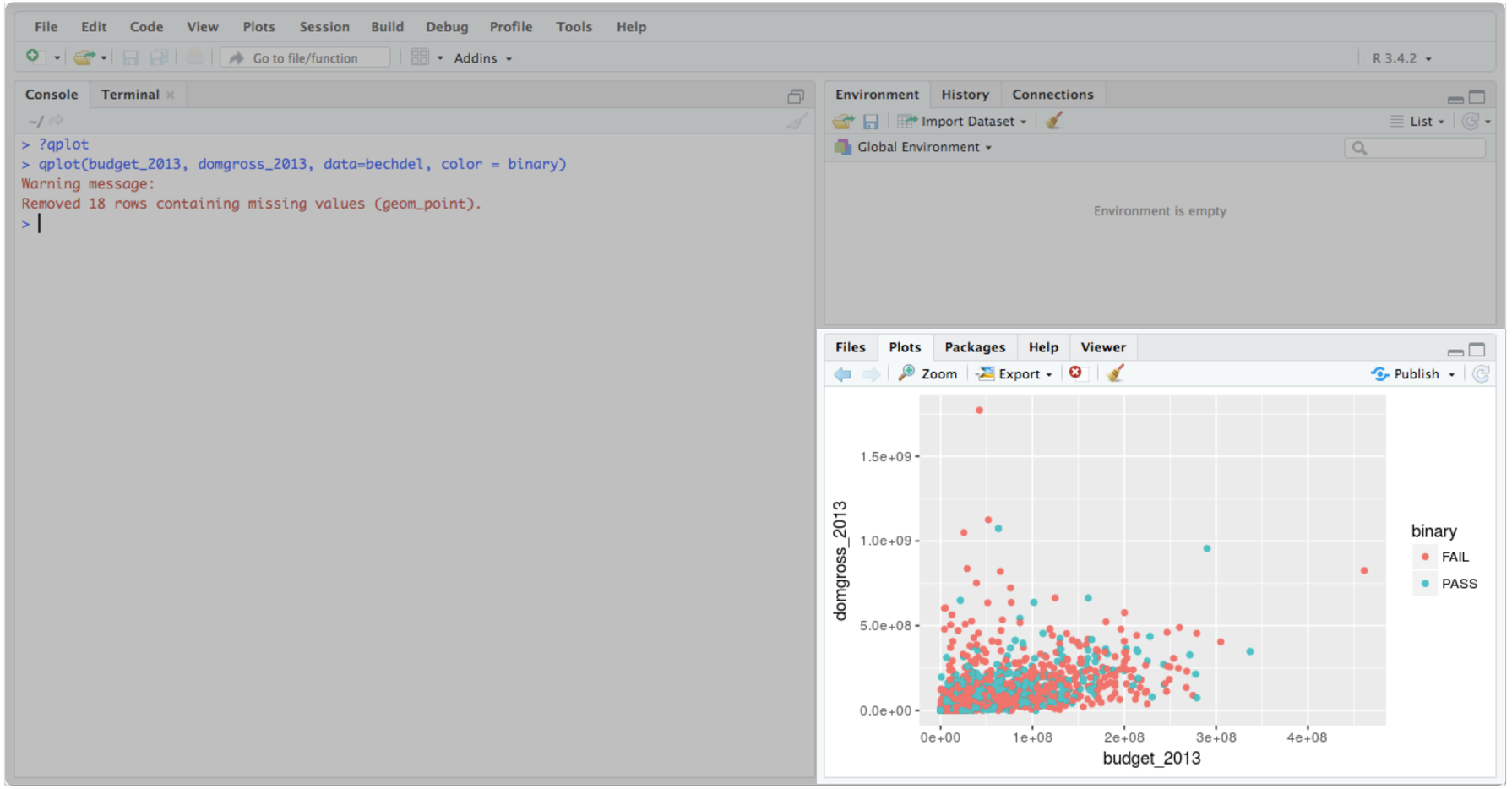
RStudio


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.4.2
Console
Terminal
~/
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Upload
Delete
Rename
More
Home
Name
Size
Modified
project
R

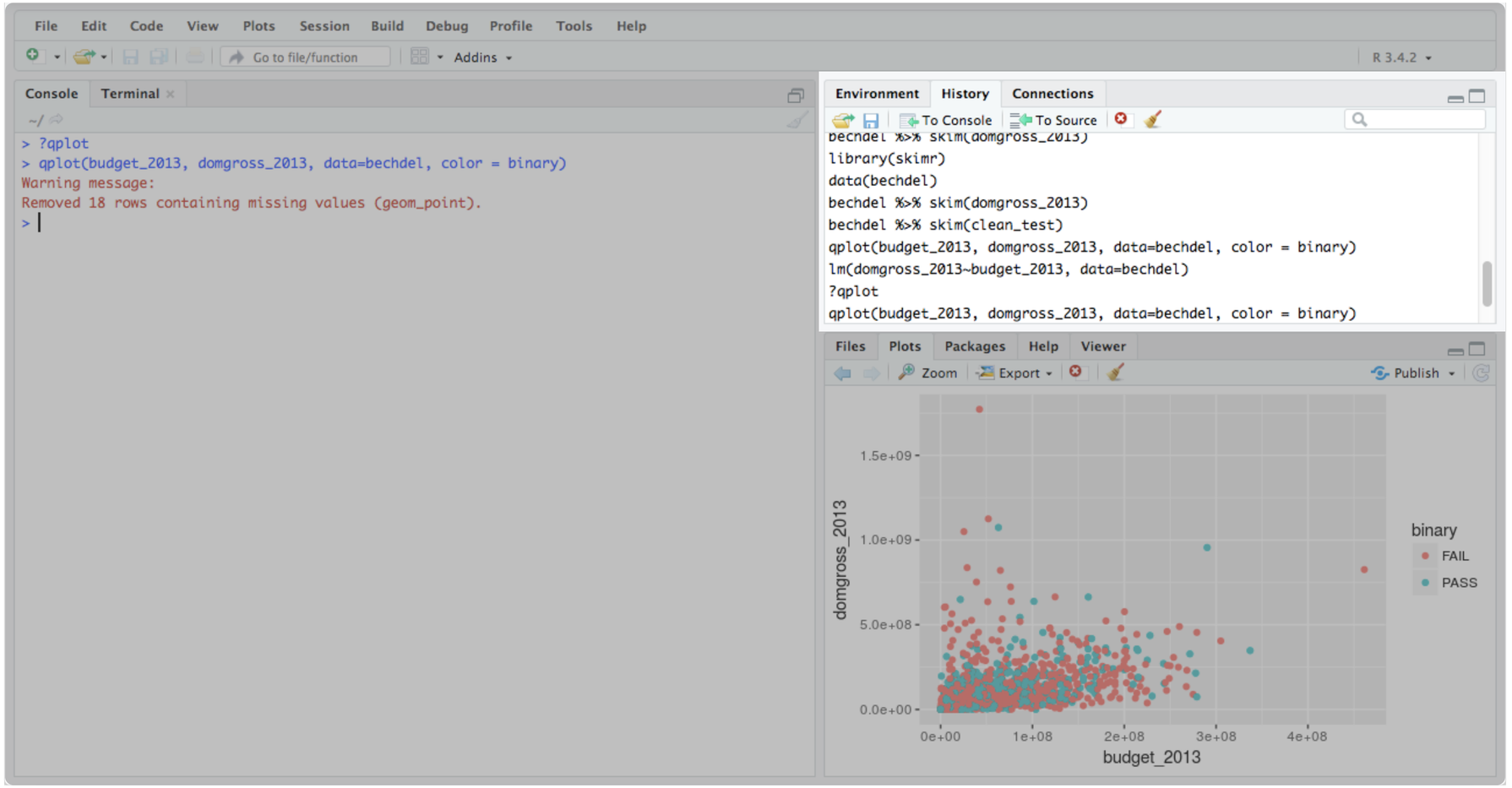
RStudio


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.4.2
Console
Terminal
~/
> ?ggplot
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
R: Create a new ggplot
Find in Topic
ggplot {ggplot2}
R Documentation
Create a new ggplot
Description
ggplot() initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.
Usage
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame(
Arguments

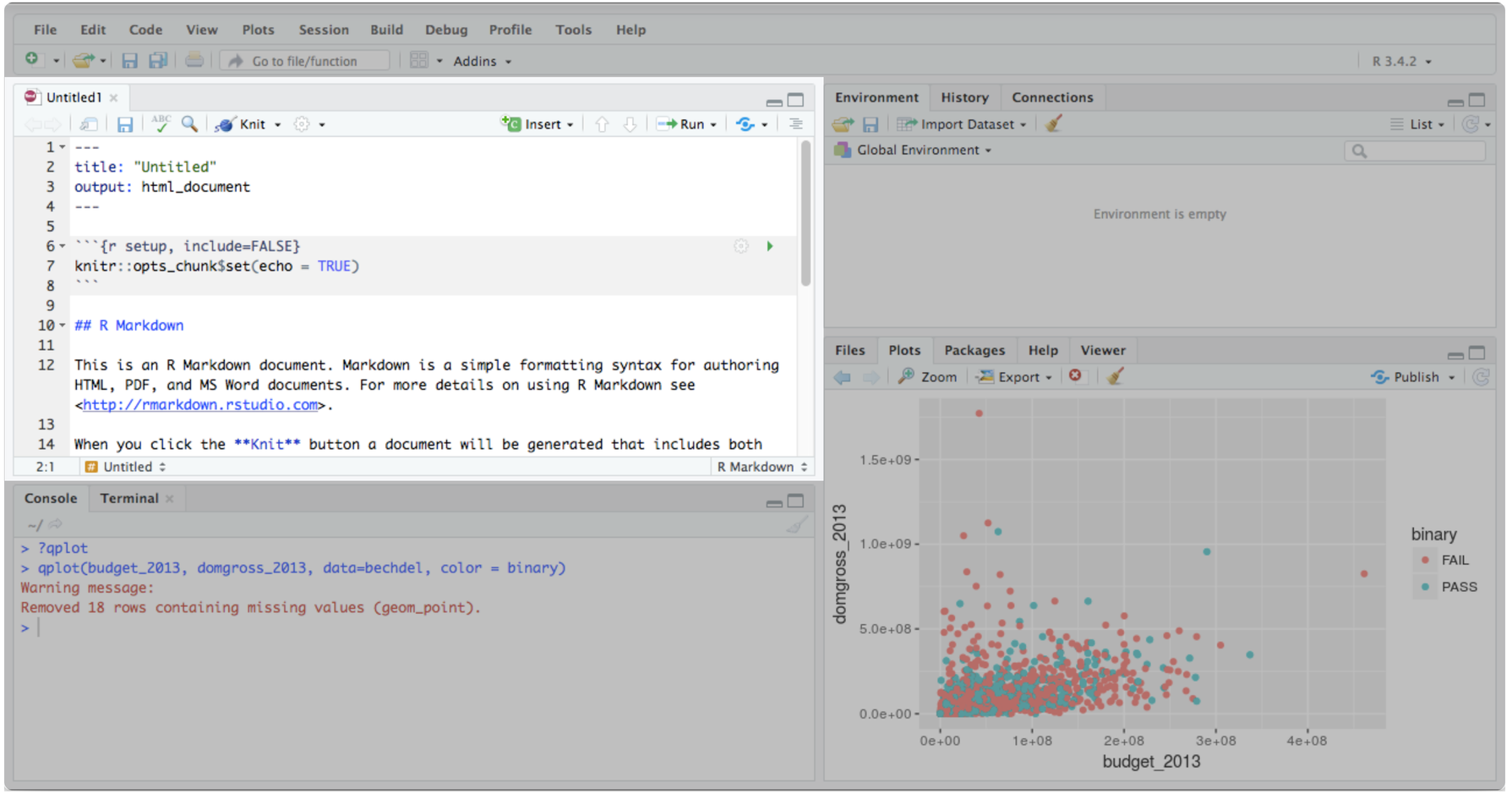
RStudio


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.4.2
Console
Terminal
~/
> ?qplot
> qplot(budget_2013, domgross_2013, data=bechdel, color = binary)
Warning message:
Removed 18 rows containing missing values (geom_point).
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
1.5e+09
1.0e+09
5.0e+08
0.0e+00
domgross_2013
0e+00
1e+08
2e+08
3e+08
4e+08
budget_2013
binary
FAIL
PASS

RStudio

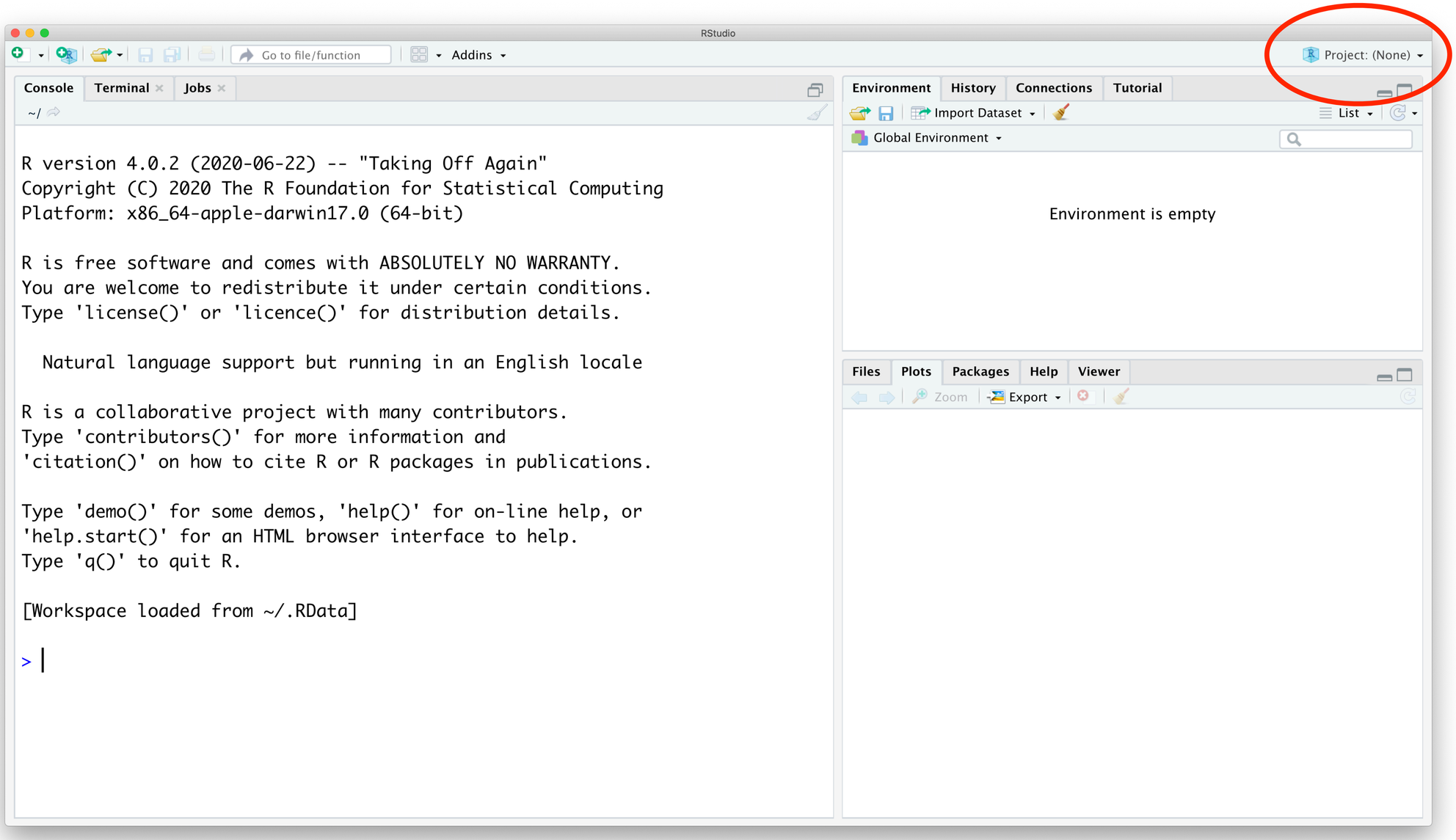

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.4.2
Console
Terminal
> ?qplot
> qplot(budget_2013, domgross_2013, data=bechdel, color = binary)
Warning message:
Removed 18 rows containing missing values (geom_point).
>
Environment
History
Connections
To Console
To Source
library(skimr)
data(bechdel)
bechdel %>% skim(domgross_2013)
bechdel %>% skim(clean_test)
qplot(budget_2013, domgross_2013, data=bechdel, color = binary)
lm(domgross_2013~budget_2013, data=bechdel)
?qplot
qplot(budget_2013, domgross_2013, data=bechdel, color = binary)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
1.5e+09
1.0e+09
5.0e+08
0.0e+00
domgross_2013
0e+00
1e+08
2e+08
3e+08
4e+08
budget_2013
binary
FAIL
PASS

RStudio


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.4.2
Untitled1
Knit
Insert
Run
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 R Markdown
Console Terminal
~/
> ?qplot
> qplot(budget_2013, domgross_2013, data=bechdel, color = binary)
Warning message:
Removed 18 rows containing missing values (geom_point).
>
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Zoom
Export
Publish
1.5e+09
1.0e+09
5.0e+08
0.0e+00
domgross_2013
0e+00
1e+08
2e+08
3e+08
4e+08
budget_2013
binary
FAIL
PASS

RStudio projects

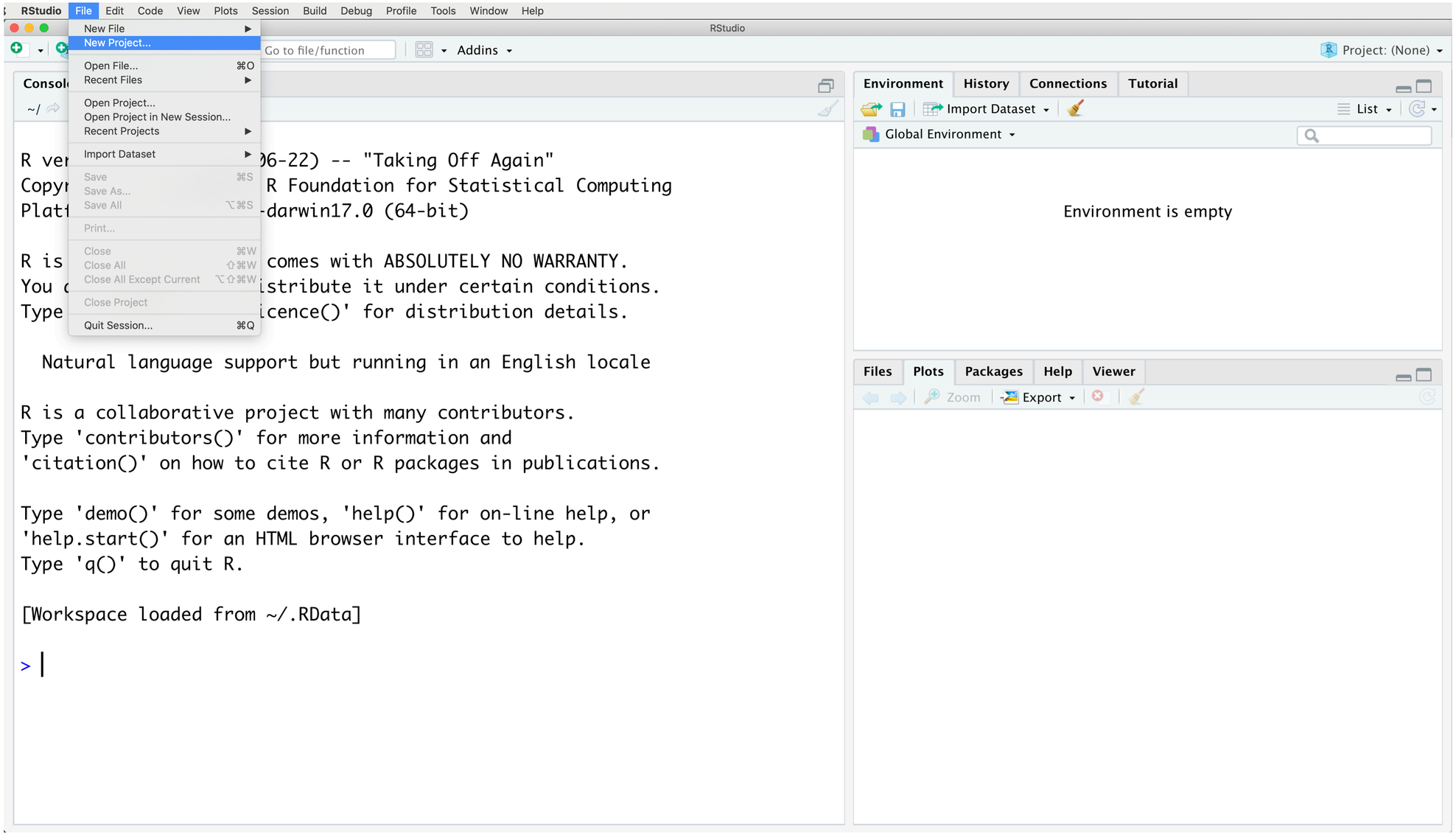
I highly, highly recommend projects

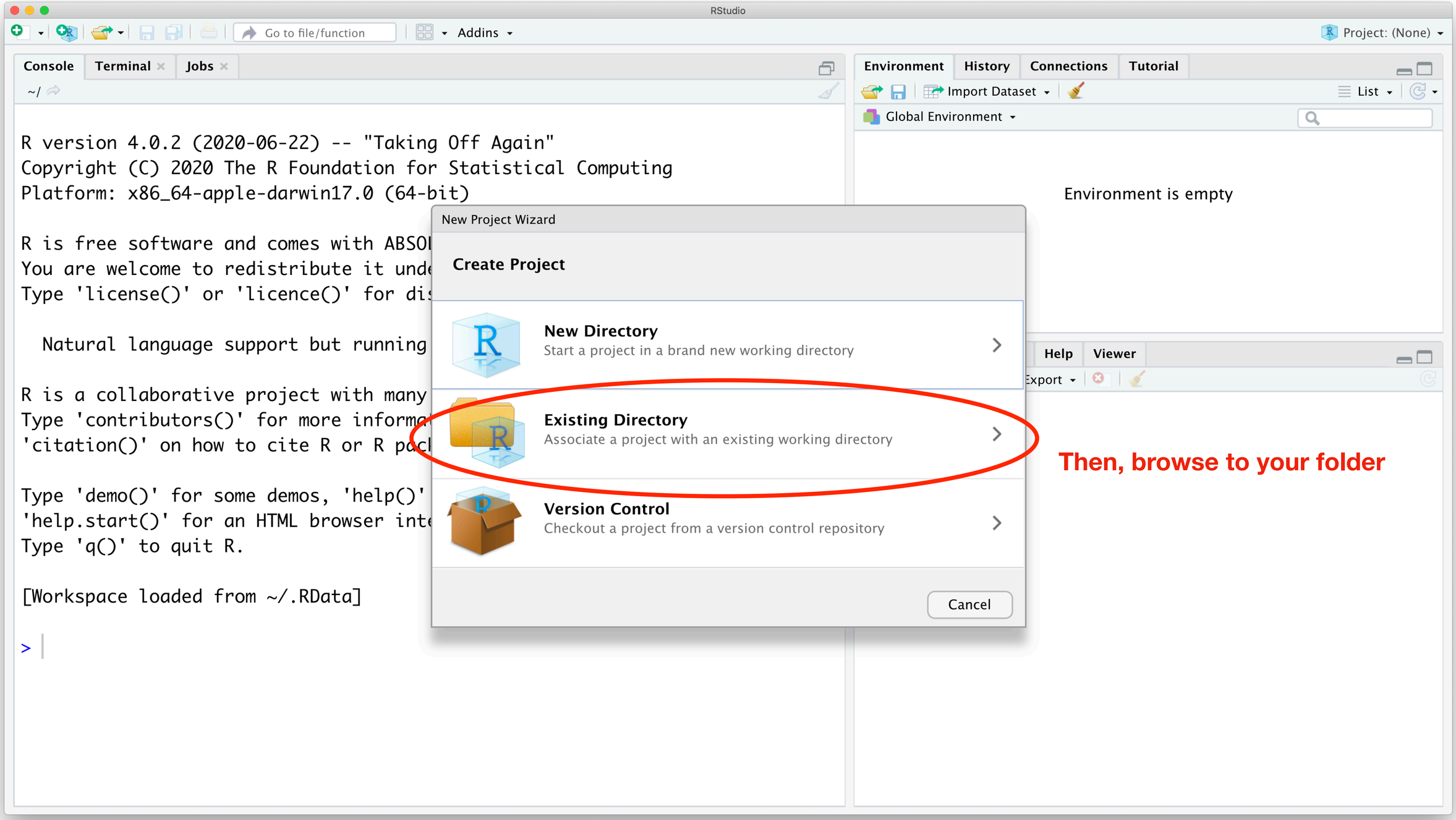


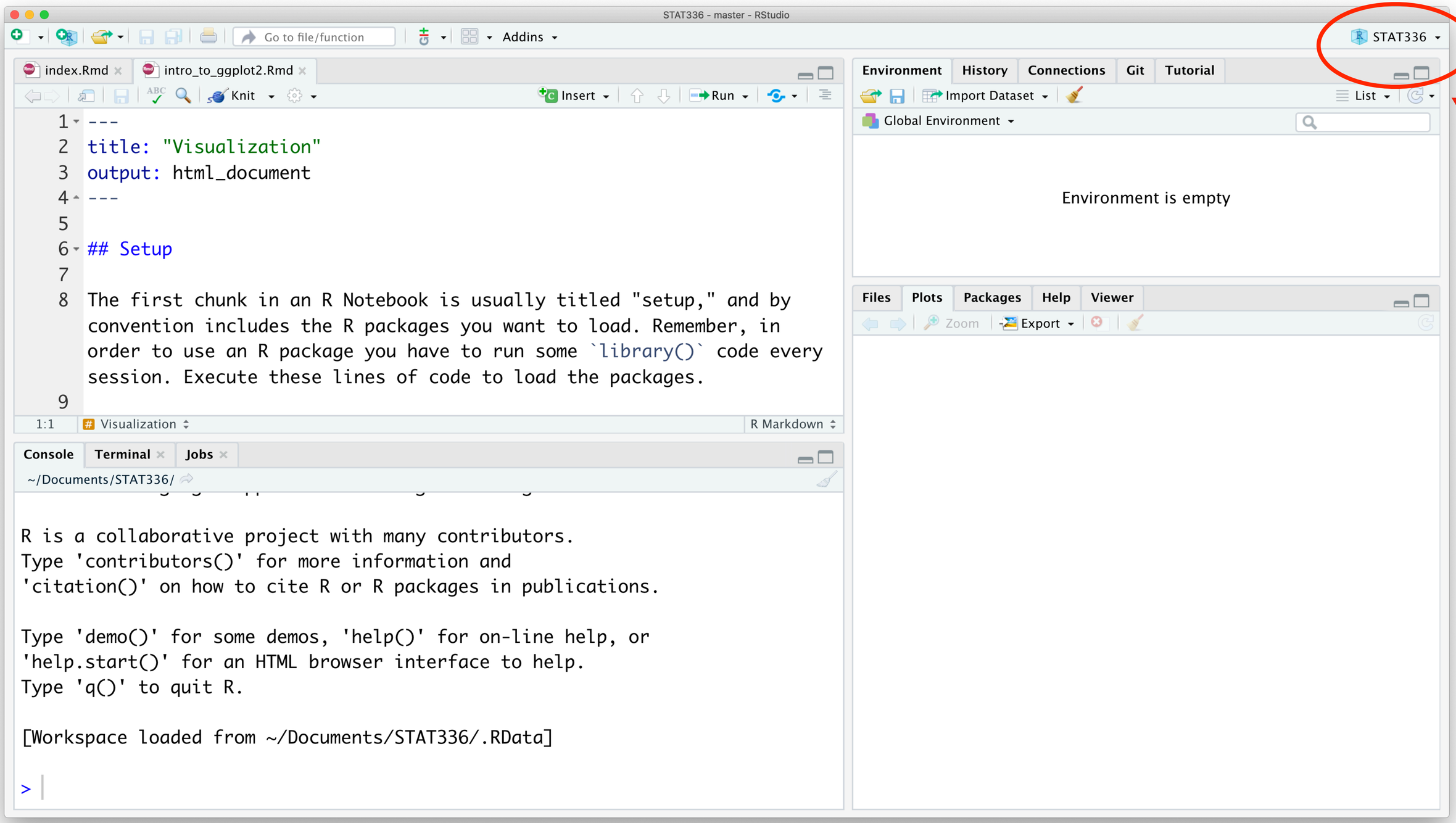
No project :(

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!


RStudio
File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Window
Help
New File
New Project...
Open File... ⌘O
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save ⌘S
Save As...
Save All ⌥⌘S
Print...
Close ⌘W
Close All ⇧⌘W
Close All Except Current ⌥⇧⌘W
Close Project
Quit Session... ⌘Q
Go to file/function
Addins
Project: (None)
Console
~/
R ver
Copyr
Platf
R is
You c
Type
06-22) -- "Taking Off Again"
R Foundation for Statistical Computing
-darwin17.0 (64-bit)
comes with ABSOLUTELY NO WARRANTY.
istribute it under certain conditions.
icence()' 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]
>
Environment
History
Connections
Tutorial
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export


RStudio
Go to file/function
Addins
Project: (None)
Console
Terminal
Jobs
~/
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 ABSOL
You are welcome to redistribute it unde
Type 'license()' or 'licence()' for dis
Natural language support but running
R is a collaborative project with many
Type 'contributors()' for more informa
'citation()' on how to cite R or R pac
Type 'demo()' for some demos, 'help()'
'help.start()' for an HTML browser inte
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
>
Environment
History
Connections
Tutorial
Import Dataset
List
Global Environment
Environment is empty
Help
Viewer
Export
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
Then, browse to your folder

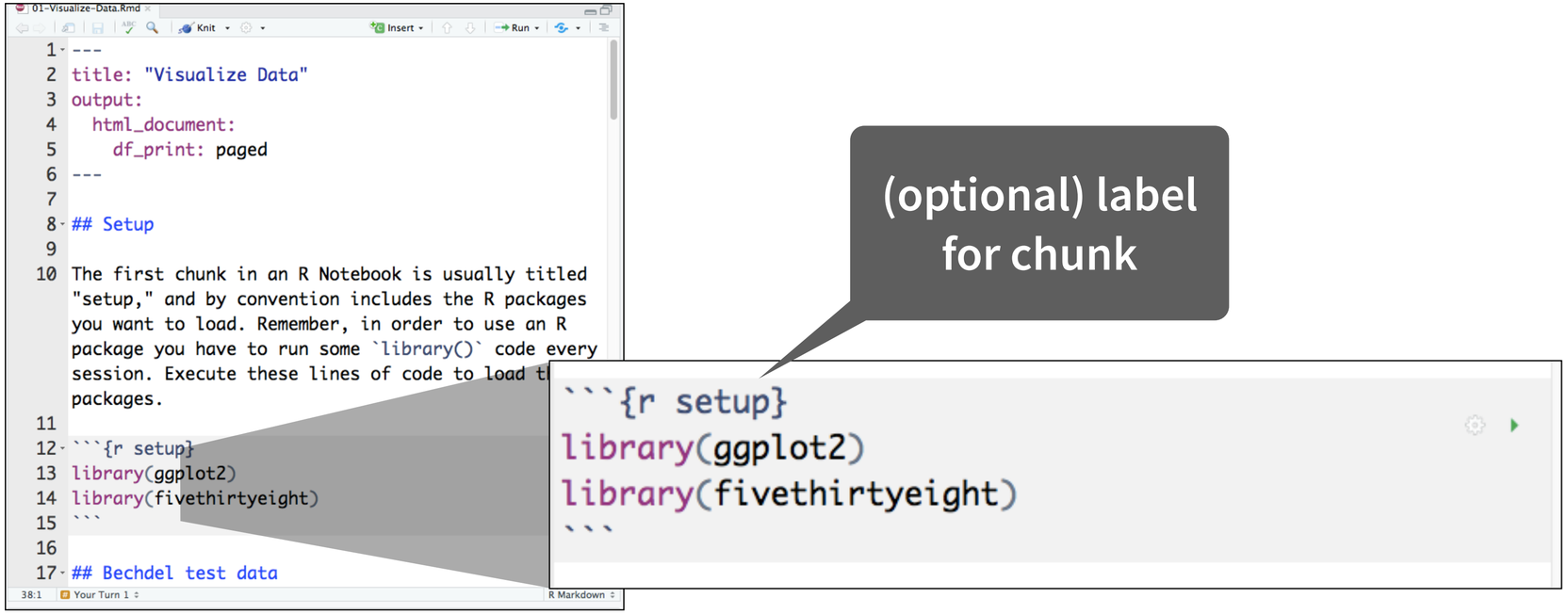

STAT336 - master - RStudio
Go to file/function
Addins
STAT336
Yay!
index.Rmd
intro_to_ggplot2.Rmd
Knit
Insert
Run
1 ---
2 title: "Visualization"
3 output: html_document
4 ---
5
6 ## Setup
7
8 The first chunk in an R Notebook is usually titled "setup," and by convention includes the R packages you want 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
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export

RMarkdown

An authoring format for Data Science.

Setup

The setup chunk is always run once before anything else



ggplot2

bechdel

Data on movies and the Bechtel test

```
bechdel
```

```
?bechdel
```

Consider

Confer with the people around you.

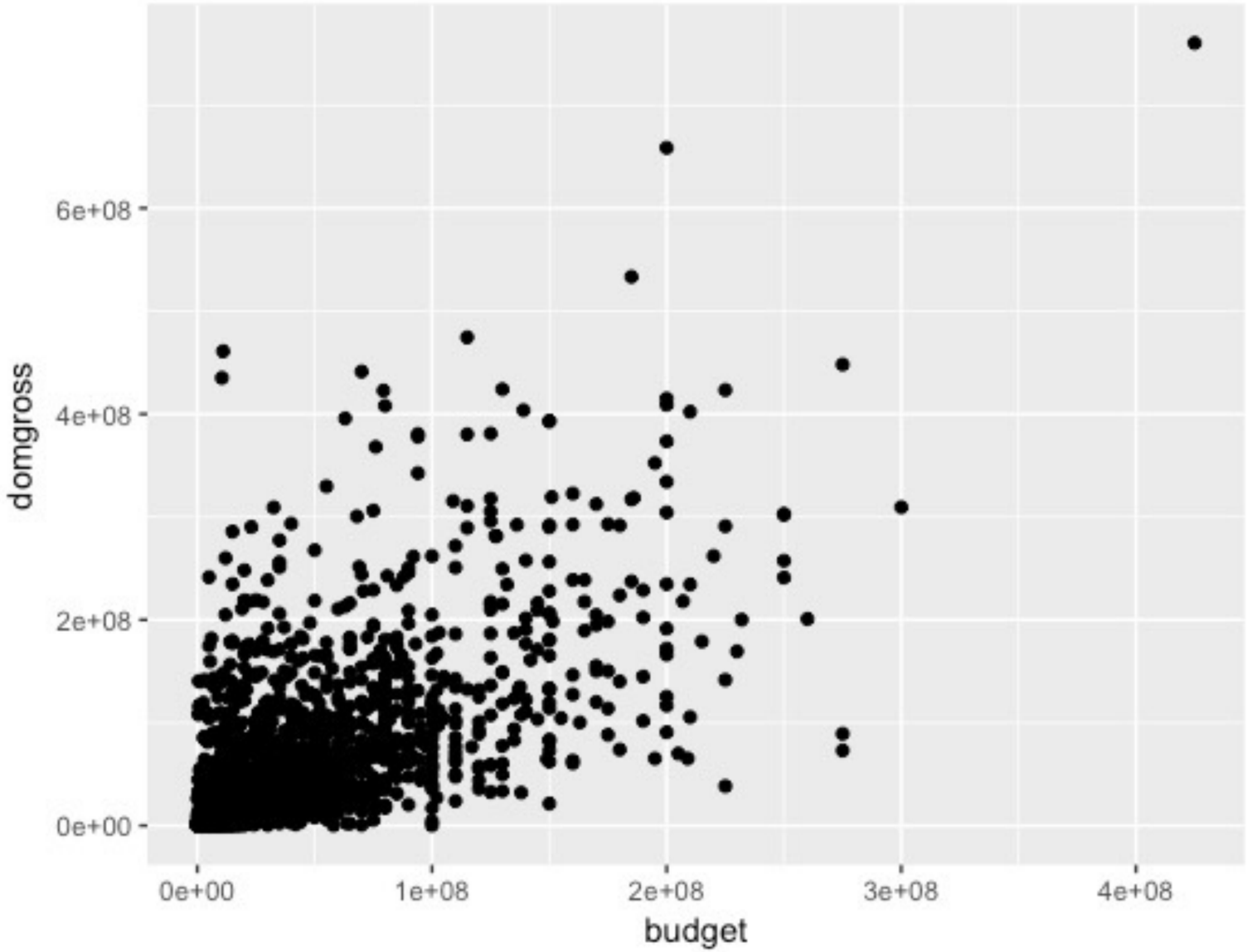
What relationship do you expect to see between movie budget (`budget`) and domestic gross(`domgross`)?

Your Turn 1

Run this code in your document to make a graph.

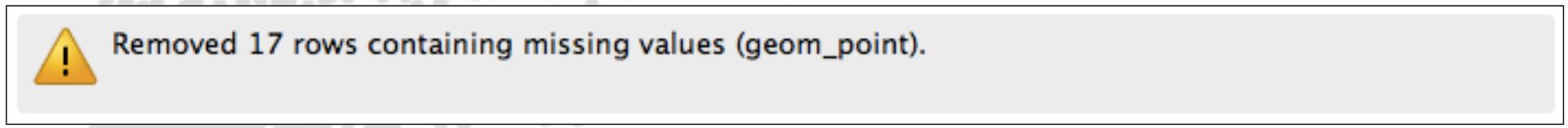
Pay strict attention to spelling, capitalization, and parentheses!

```
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))
```

data

+ before new line

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

type of layer

aes()

x variable

y variable

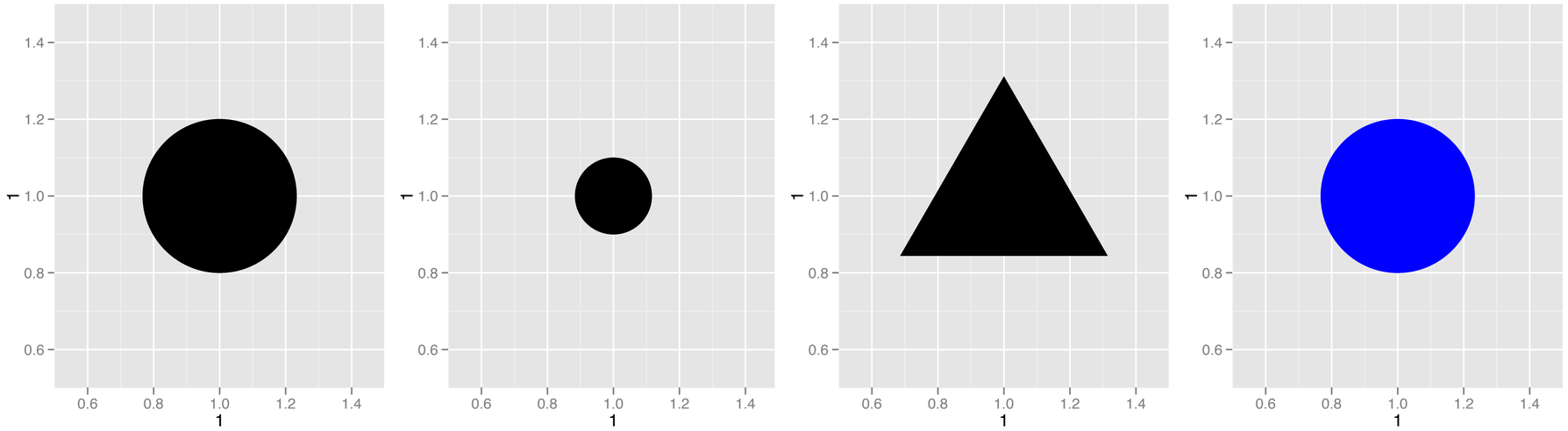
A template

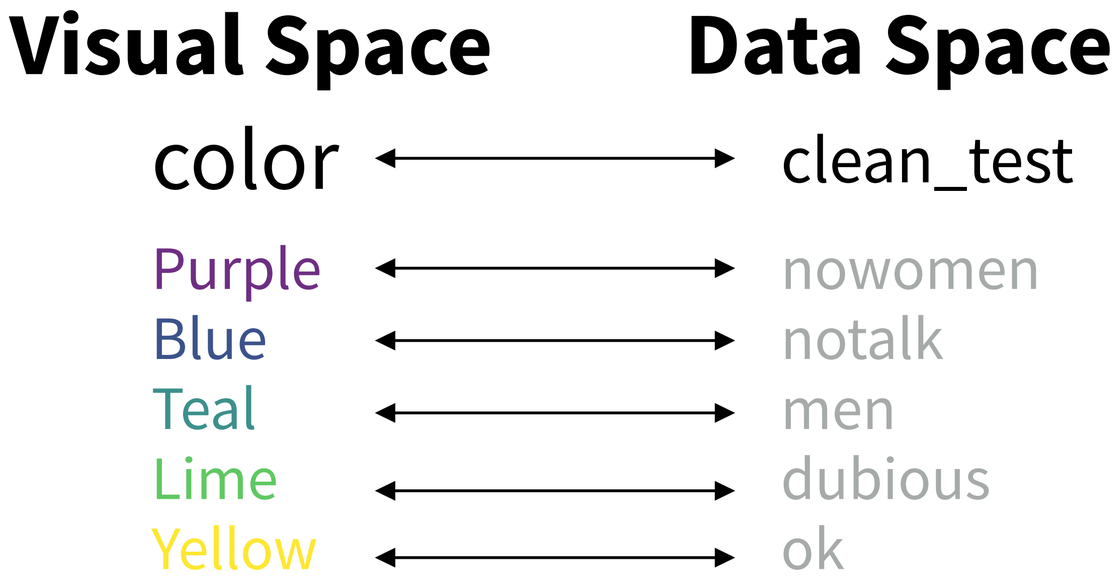
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

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

Aesthetics

Aesthetics




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

Aesthetics

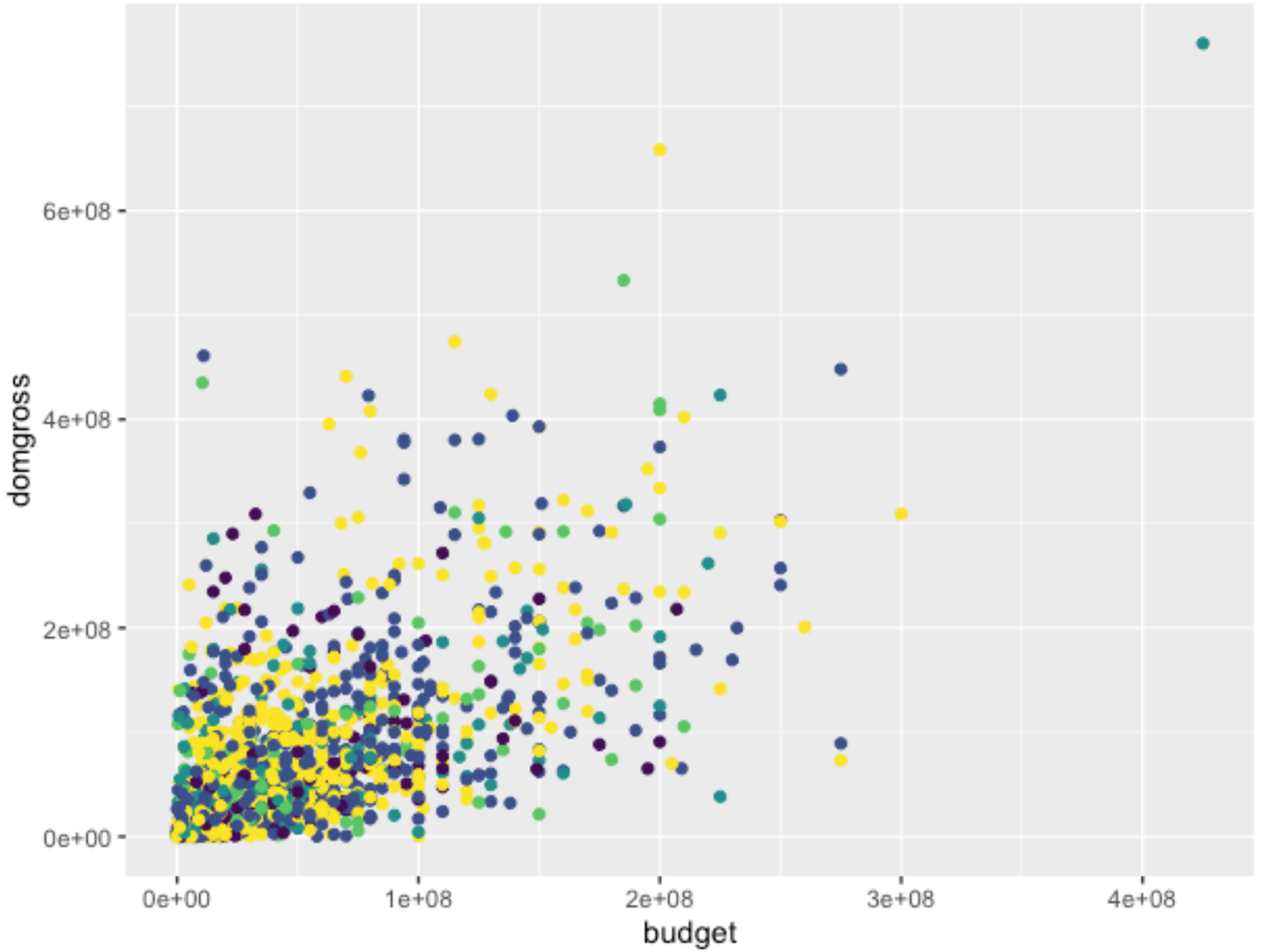
aesthetic property

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

clean_test
- nowomen
- notalk
- men
- dubious
- ok

ggplot2

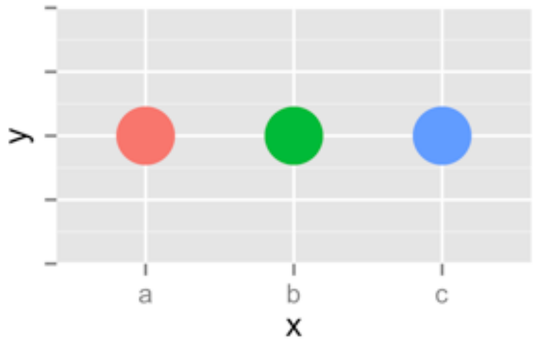
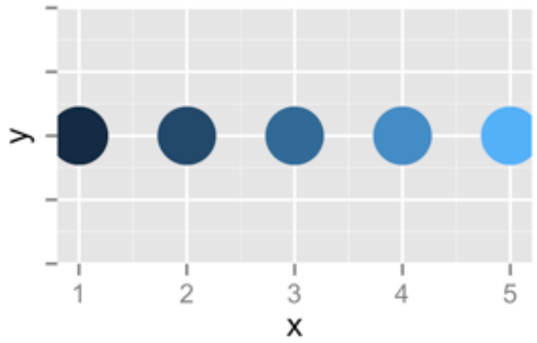
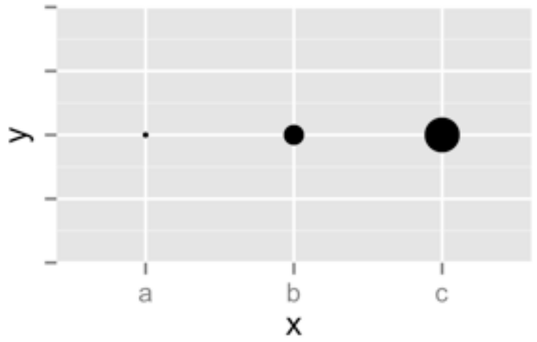
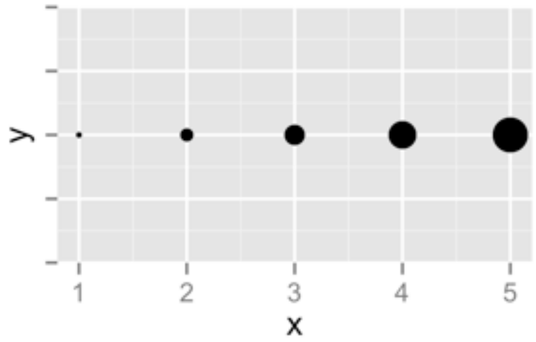
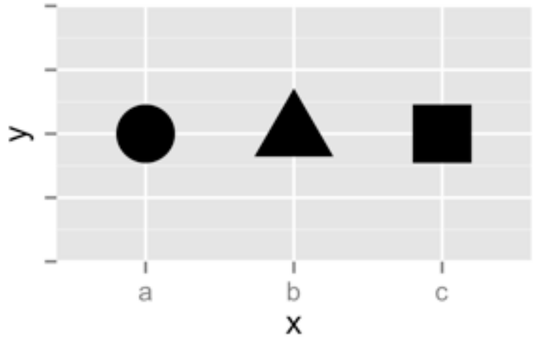
Your Turn 2

recall

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

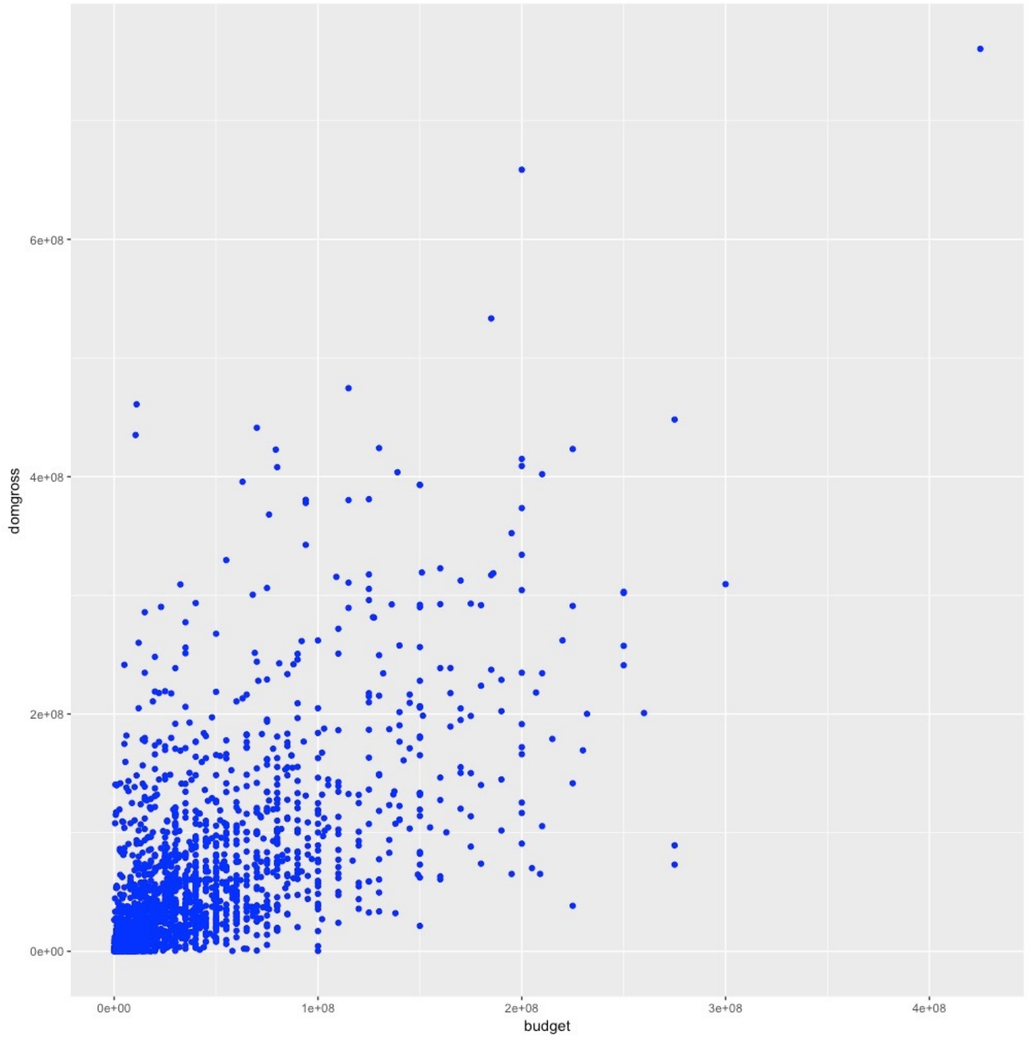
In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

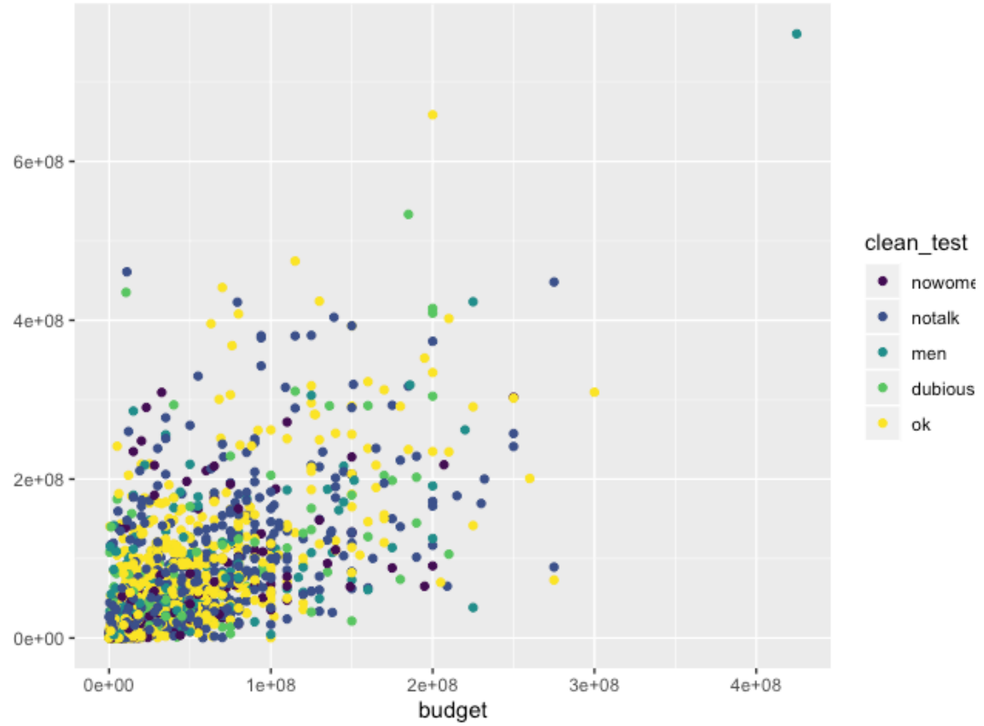
- Do different things happen when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

set vs. map

How would you make this plot?

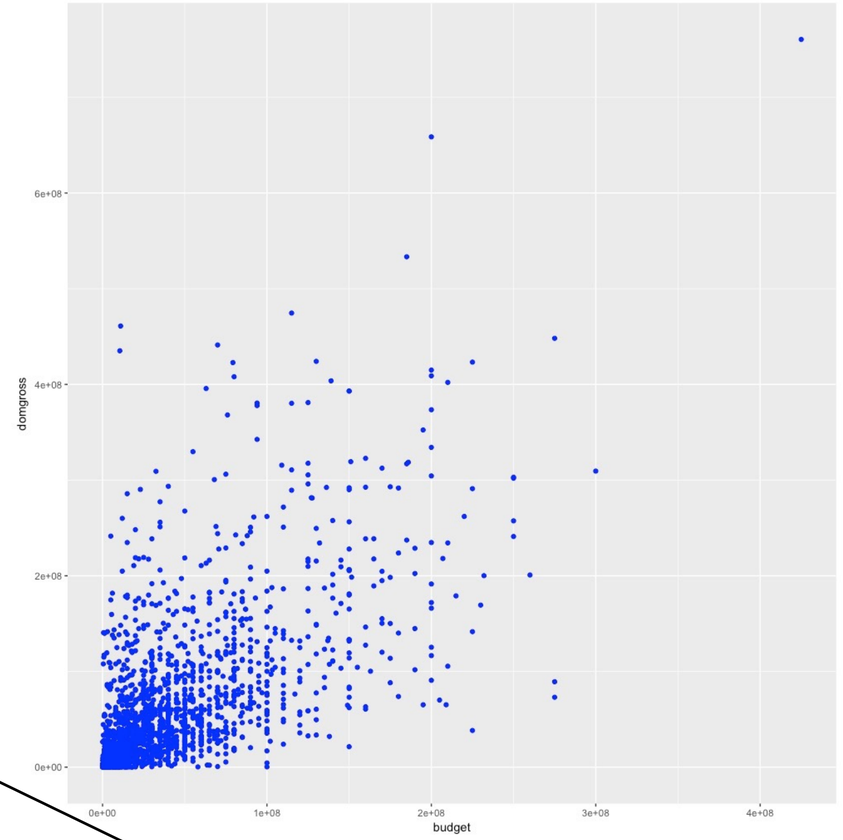




Inside of aes(): maps an aesthetic to a variable

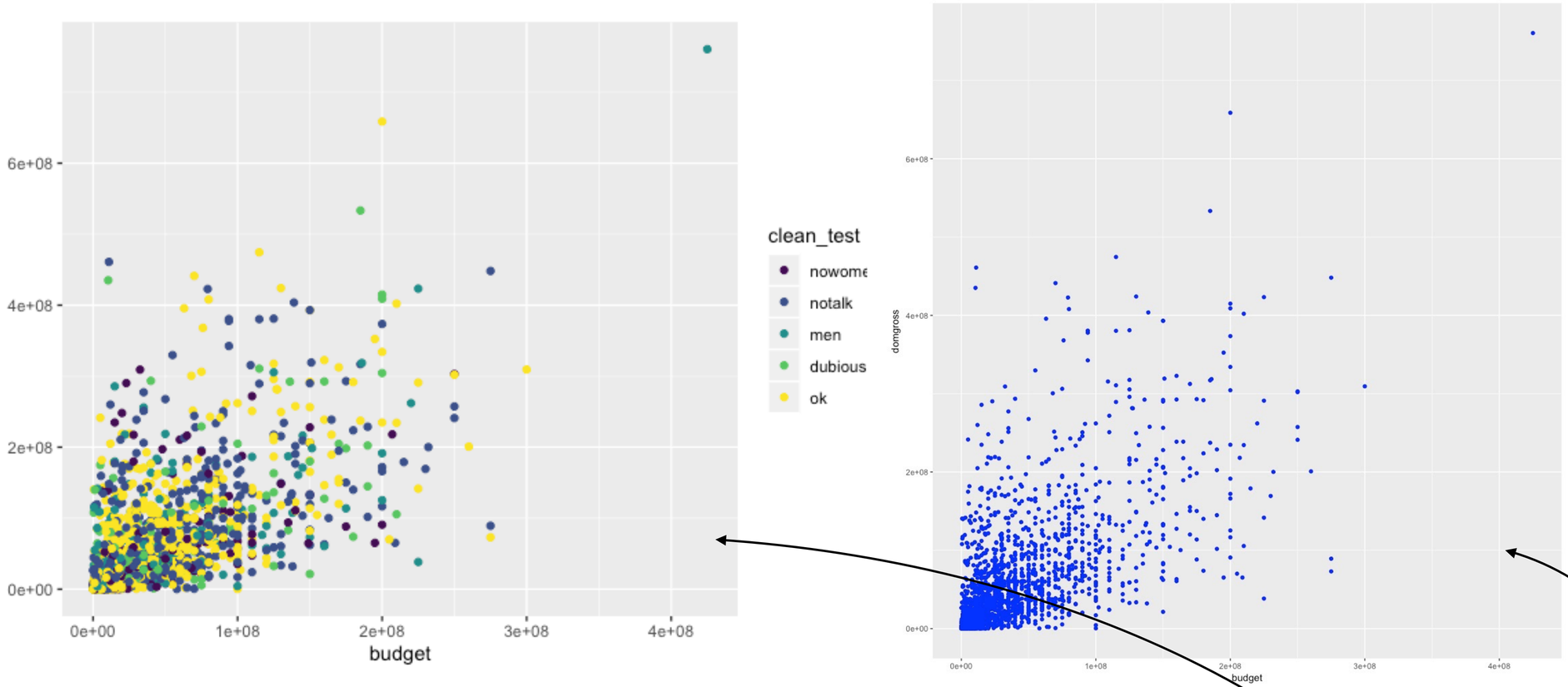
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test))
```

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



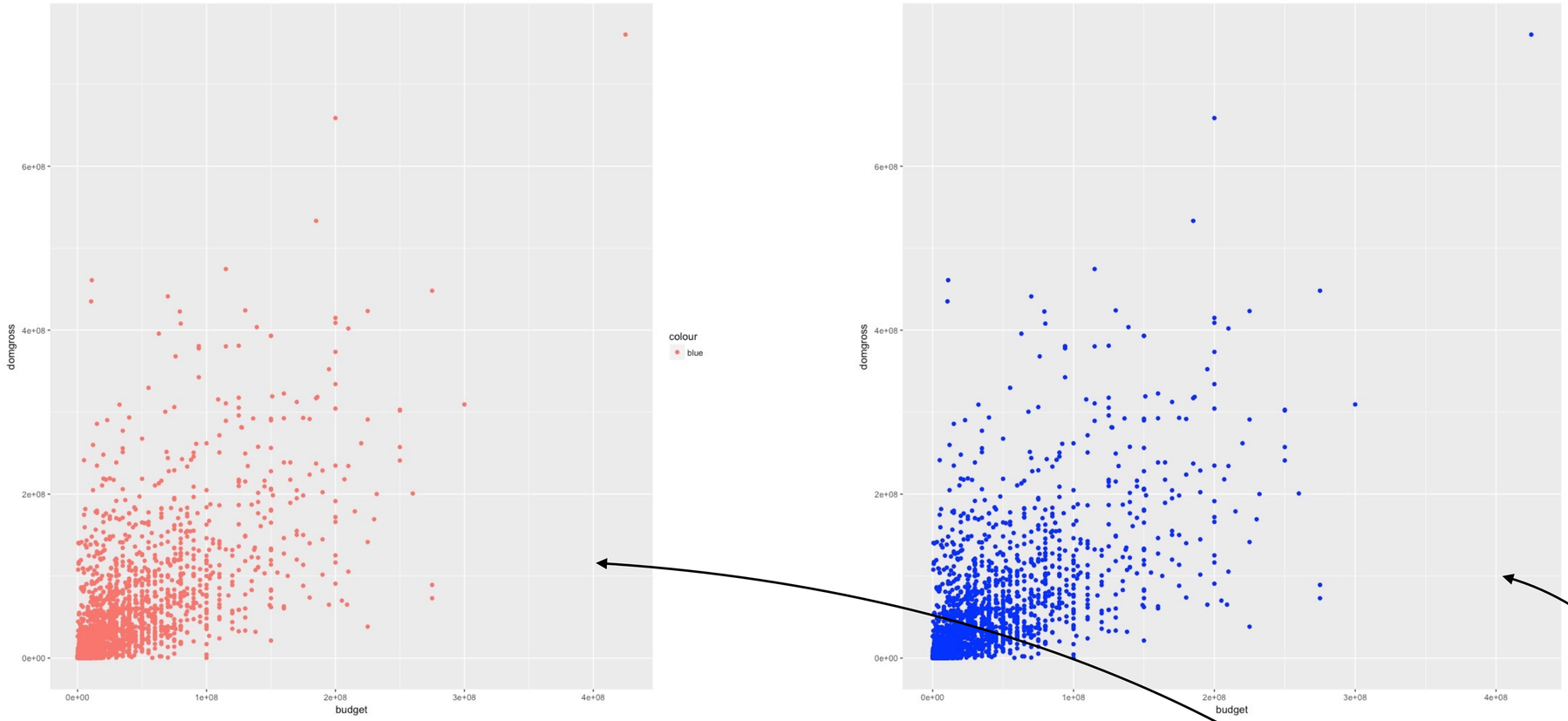

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color=clean_test)
```

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




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

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



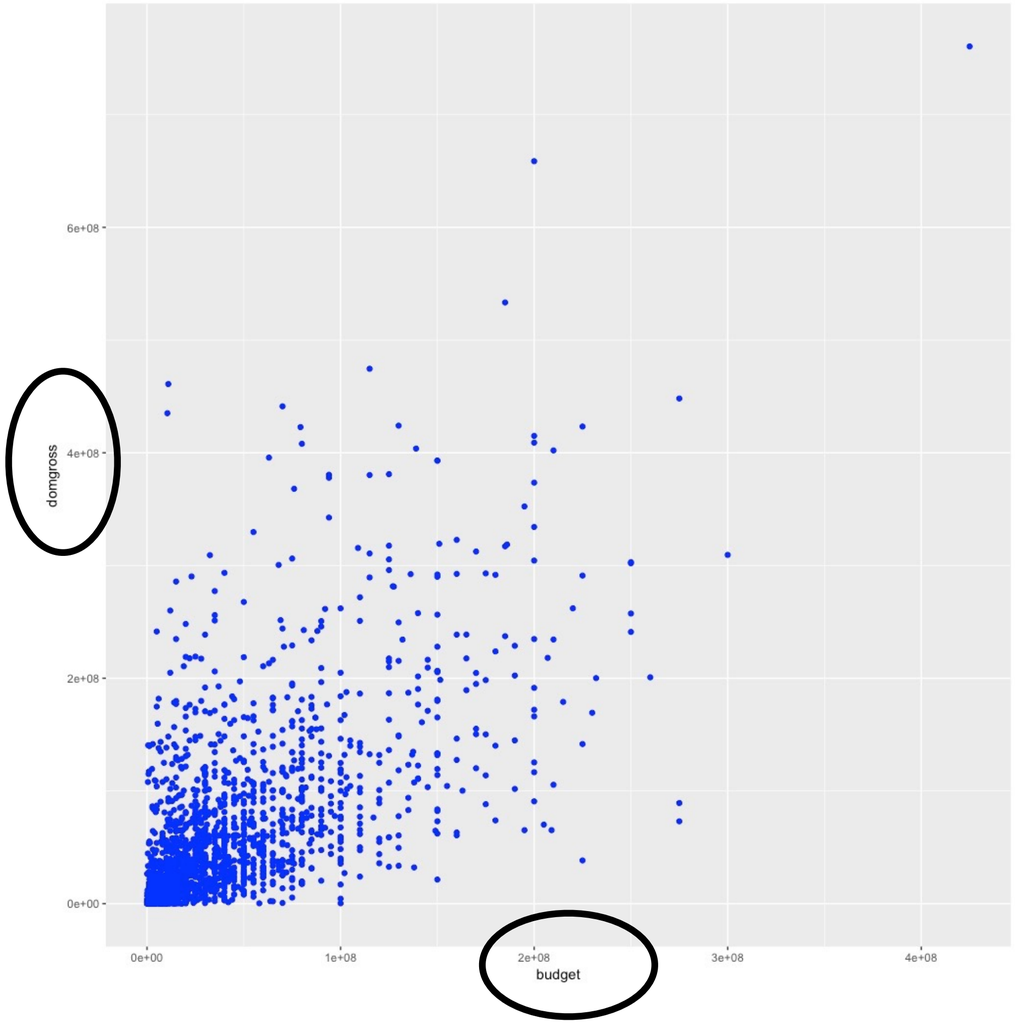

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color="blue"))
```

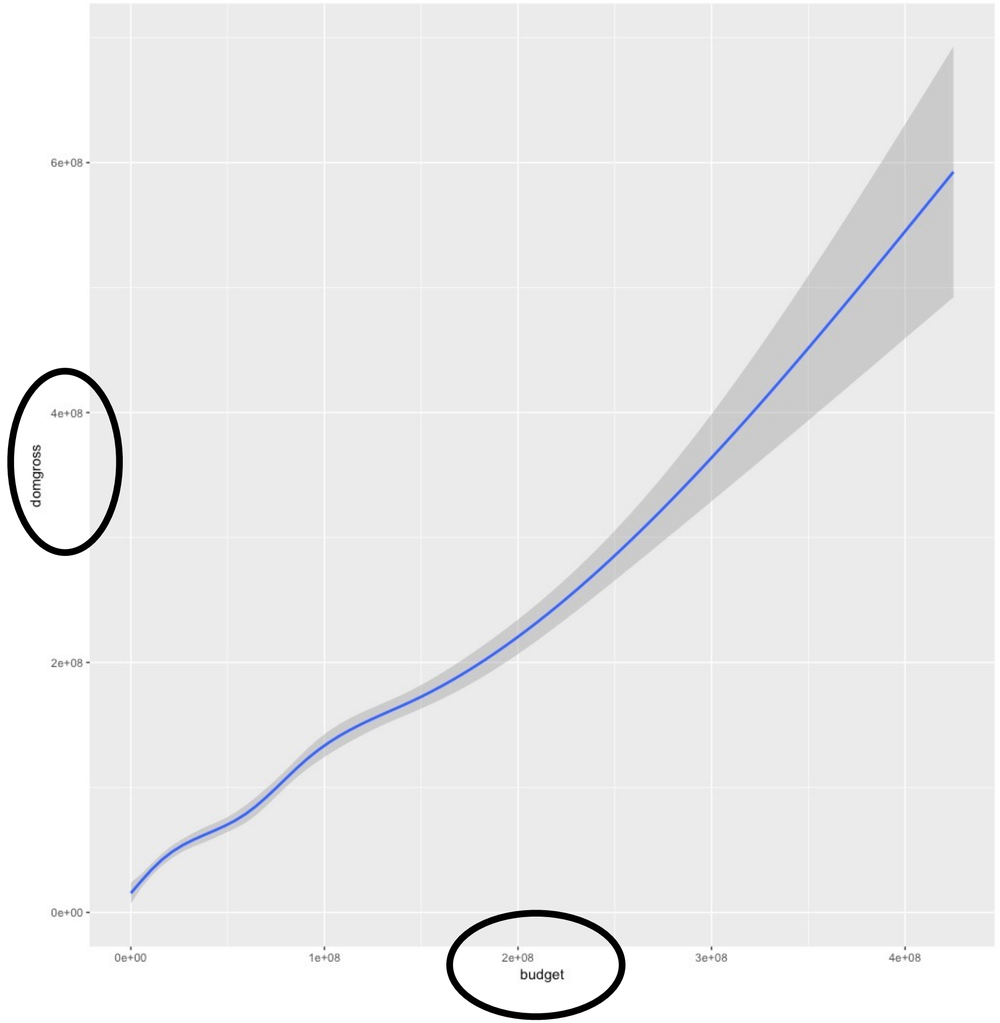
```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross), color="blue")
```

geoms

How are these plots similar?

Same: x var , y var , data

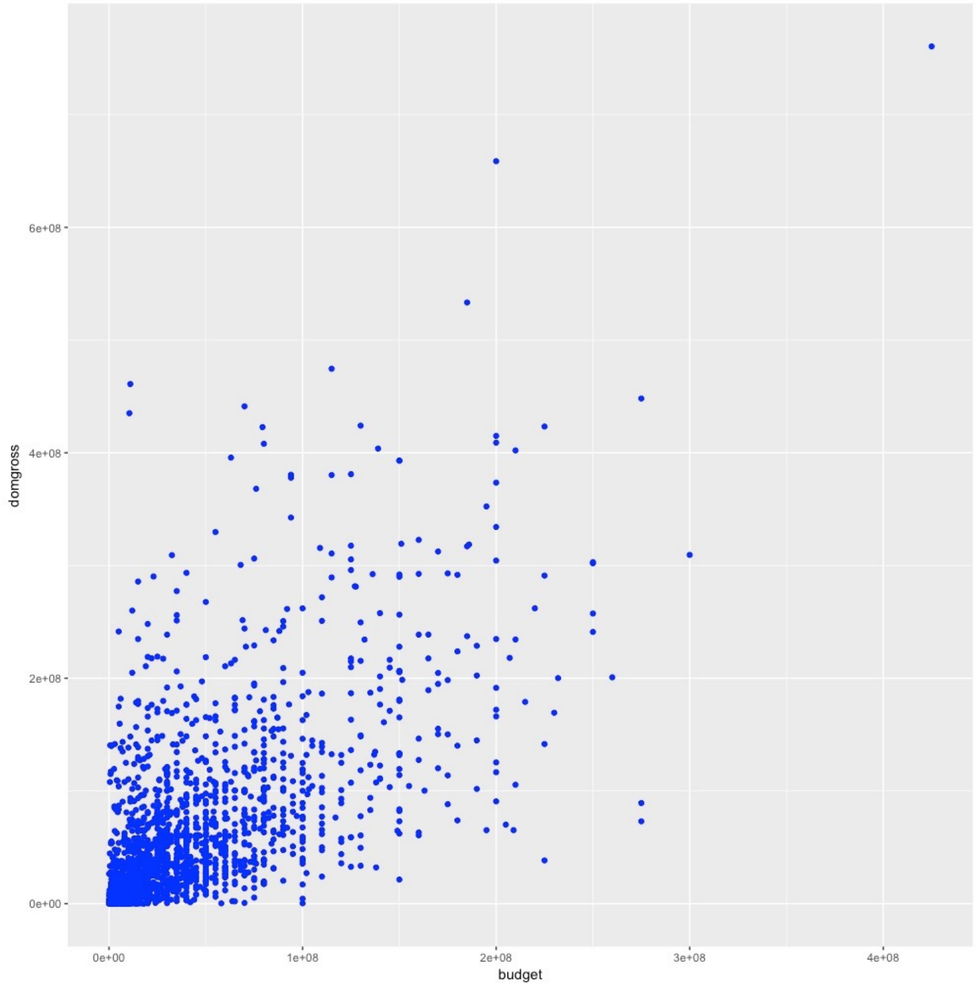


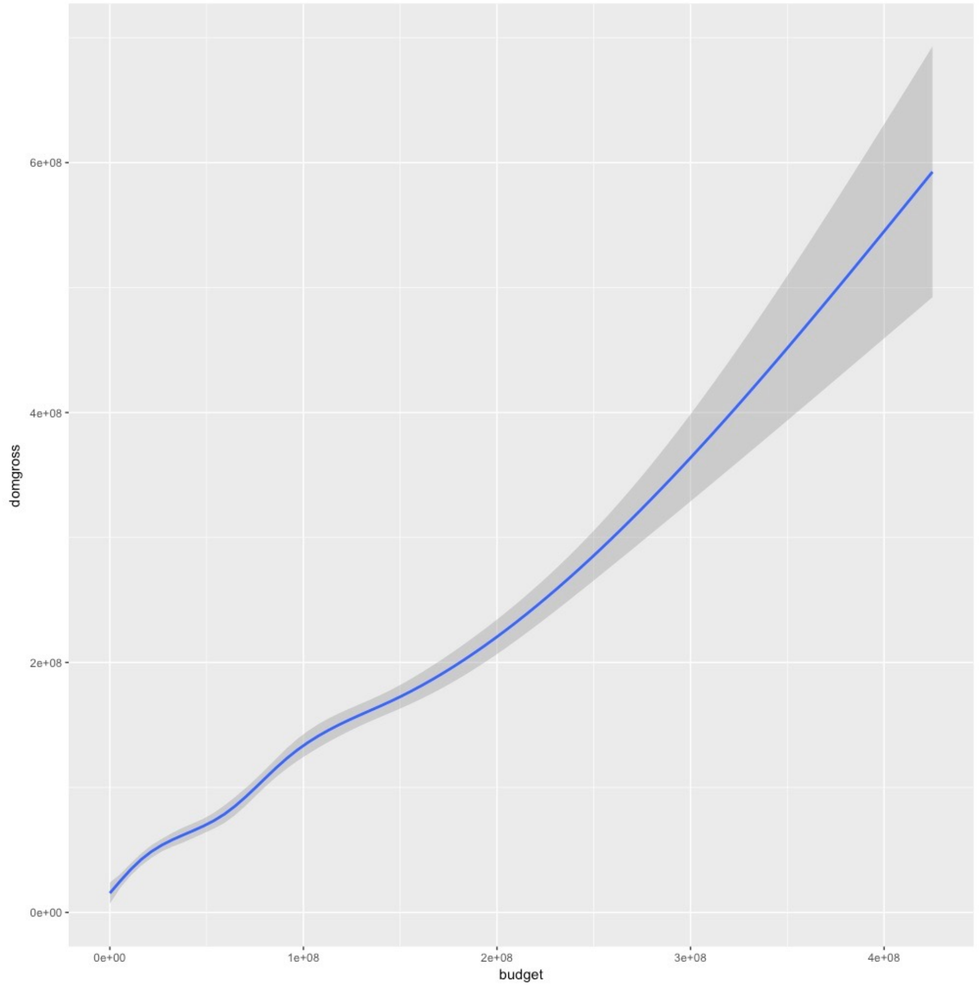


How are these plots different?

Different: geometric object (geom),

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



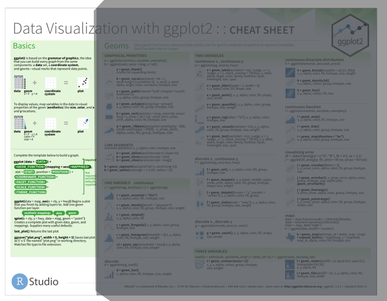


geoms

```
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, 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

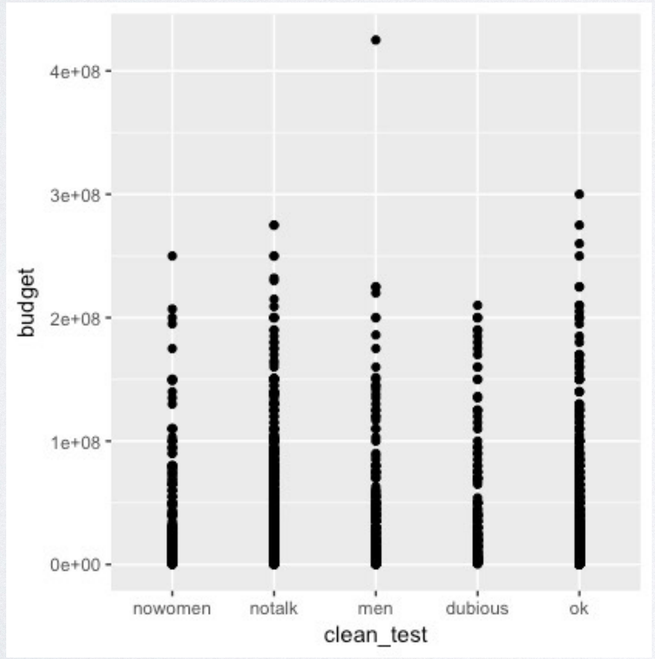
ggplot2

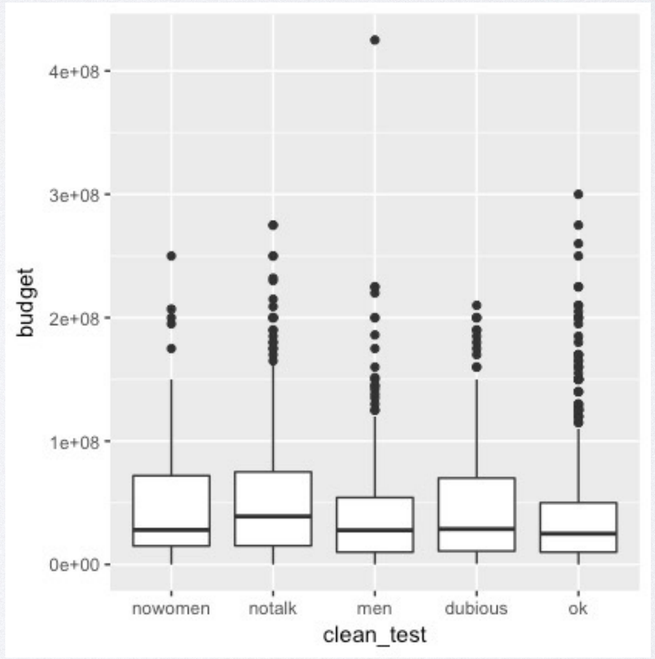
Your Turn 3

recall

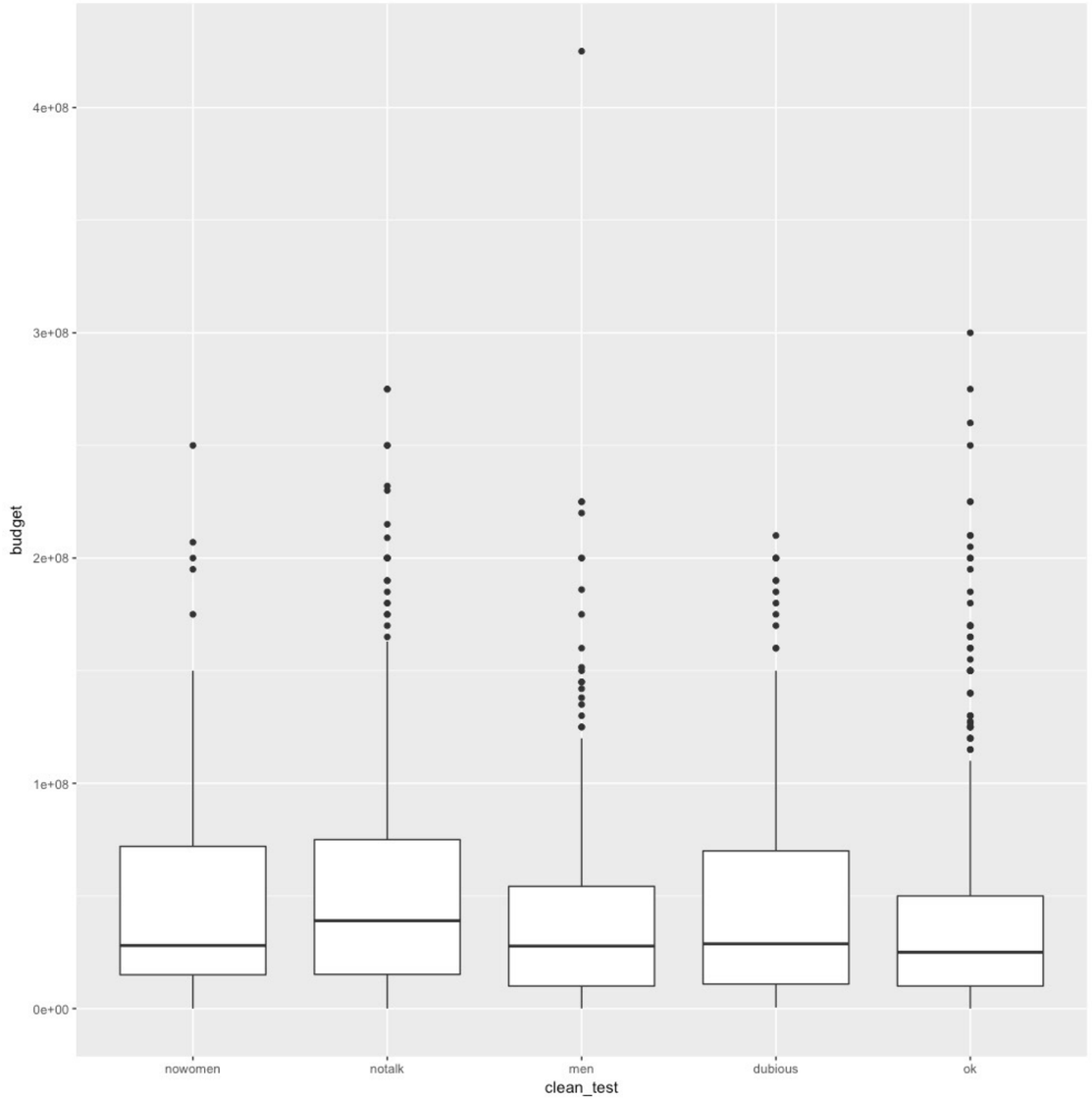
```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = clean_test, y = budget))
```

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





02:00

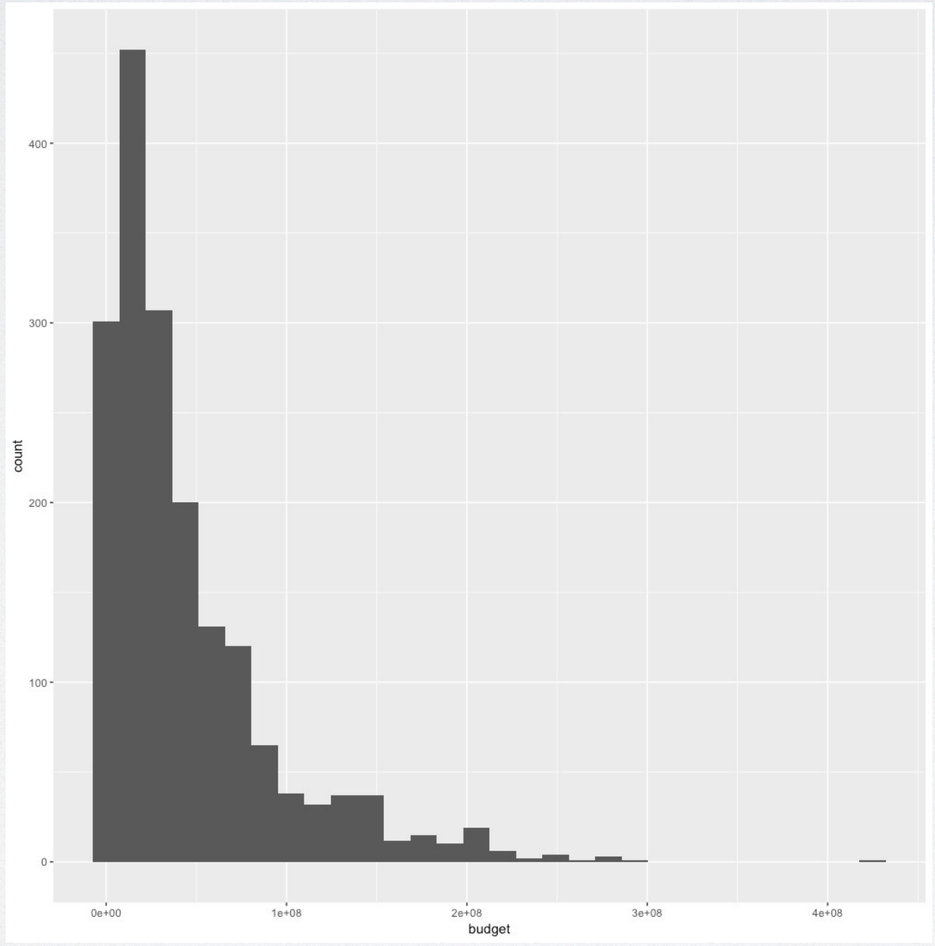



```
ggplot(data = bechdel) +
  geom_boxplot(mapping = aes(x = clean_test, y = budget))
```

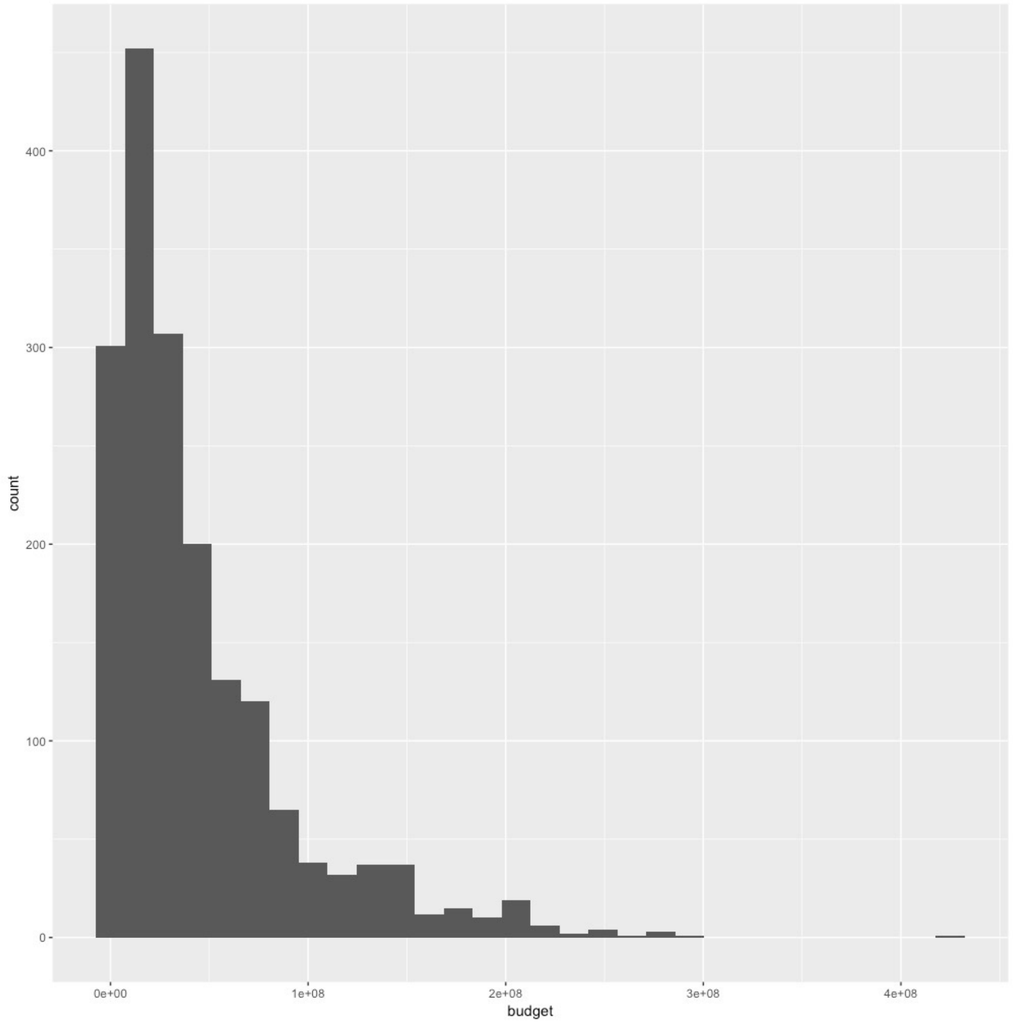
Your Turn 4

With your partner, make the histogram of **budget** below.

Use the cheatsheet. Hint: do not supply a **y** variable.

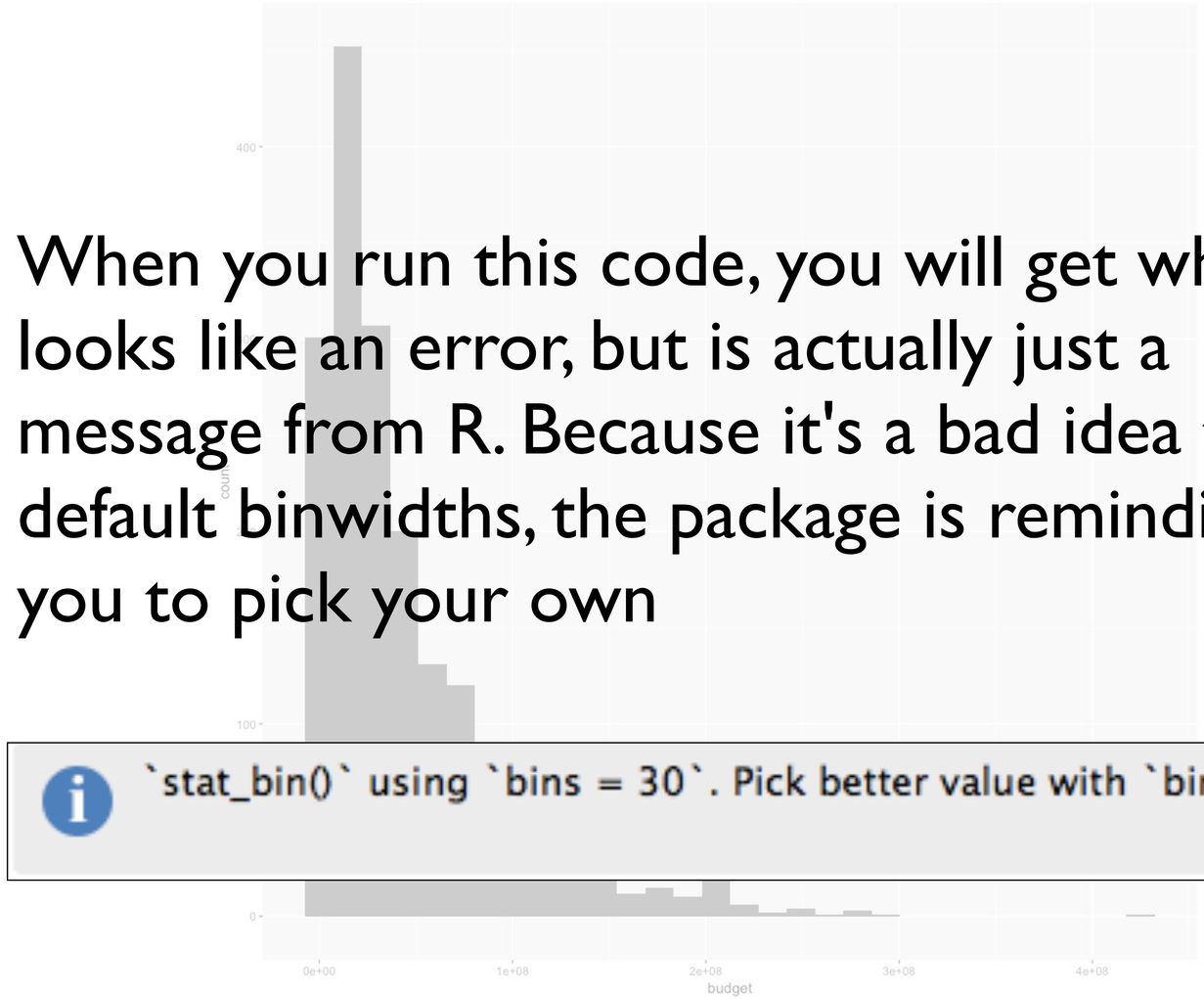


02:00




```
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



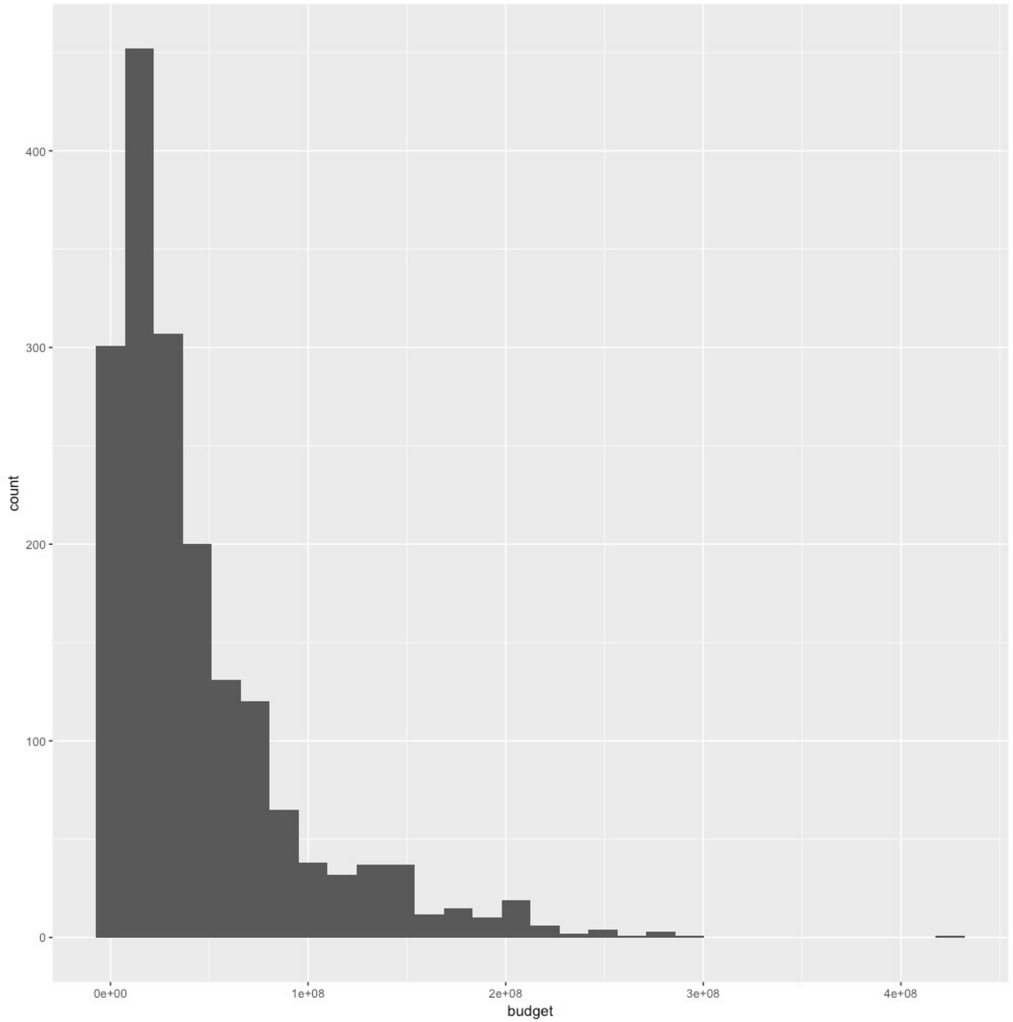

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

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

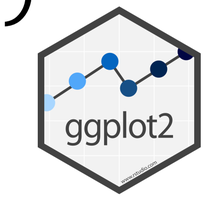
Your Turn 5

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

Try it out

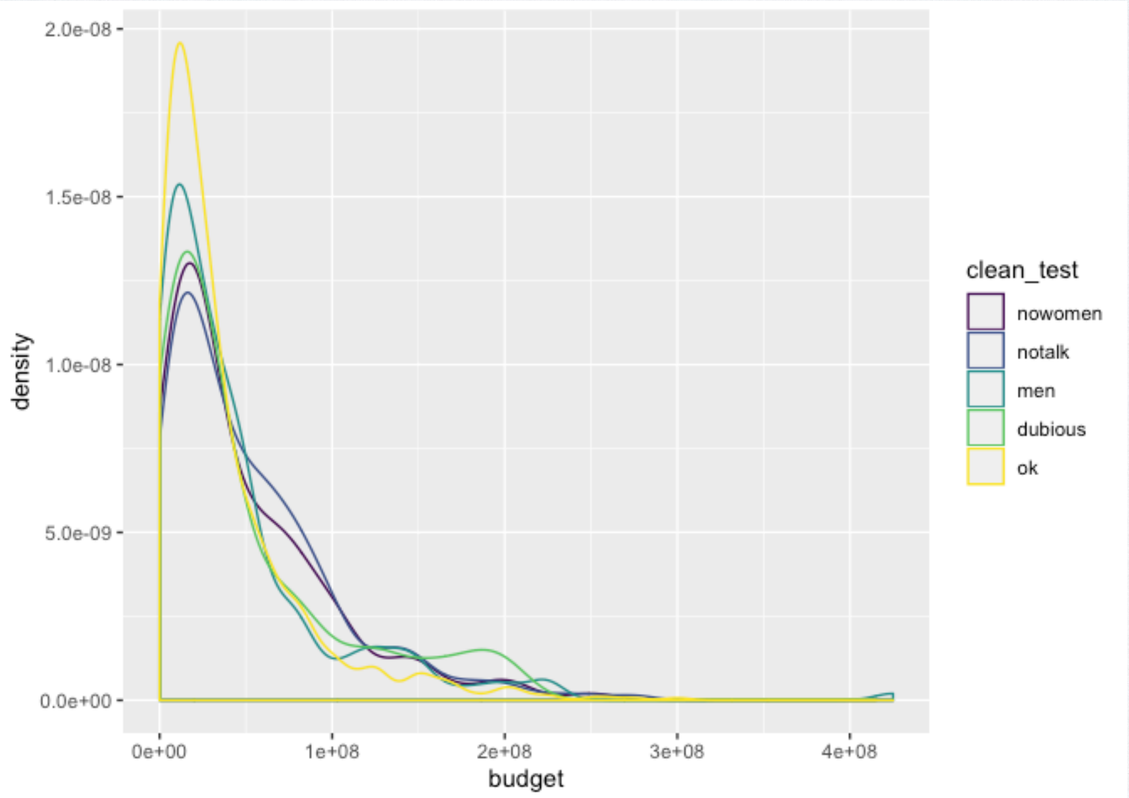



```
ggplot(data = bechdel) +
  geom_histogram(mapping = aes(x = budget), binwidth=10000000)
```

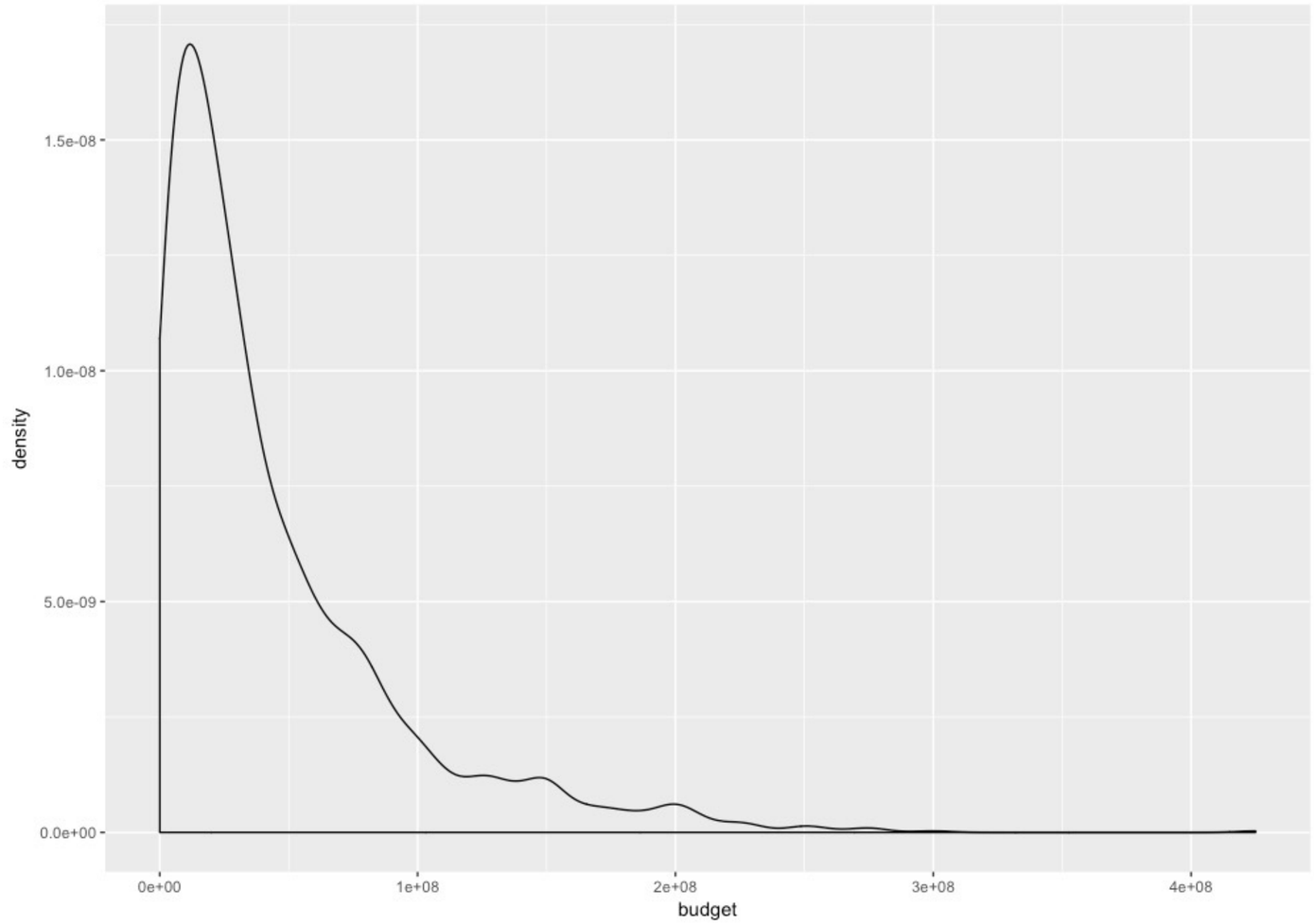


Your Turn 6

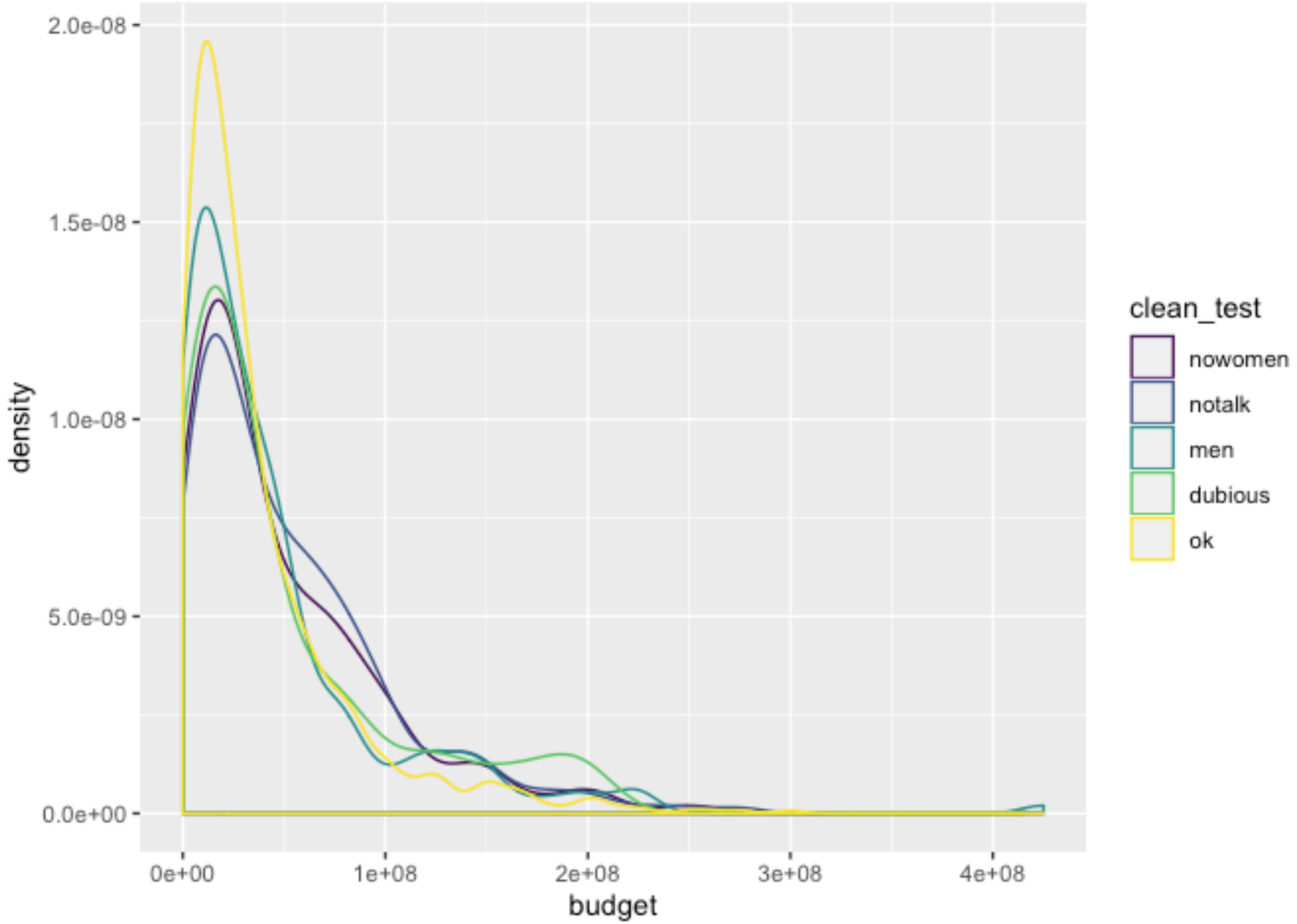
With your partner, make the density plot of **budget** colored by **clean_test** below. Use the cheatsheet. Try your best guess.



02:00



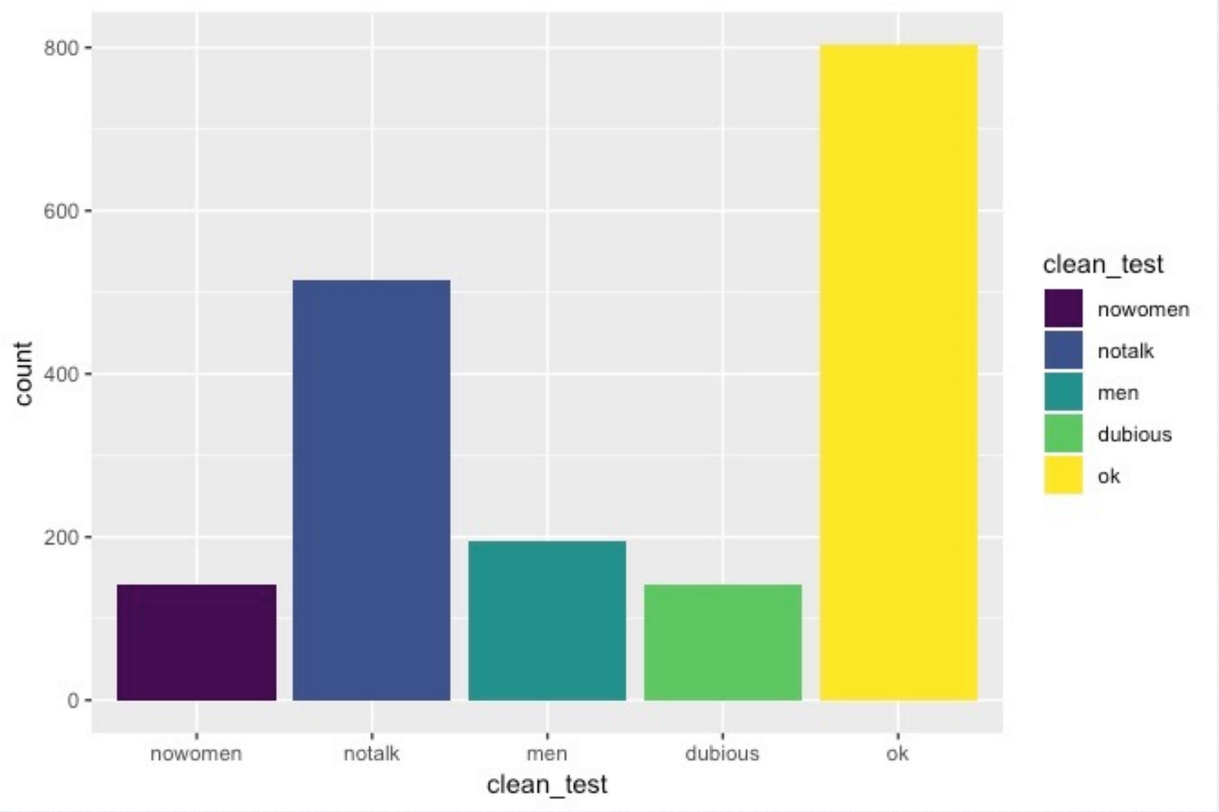

```
ggplot(data = bechdel) +
  geom_density(mapping = aes(x = budget))
```



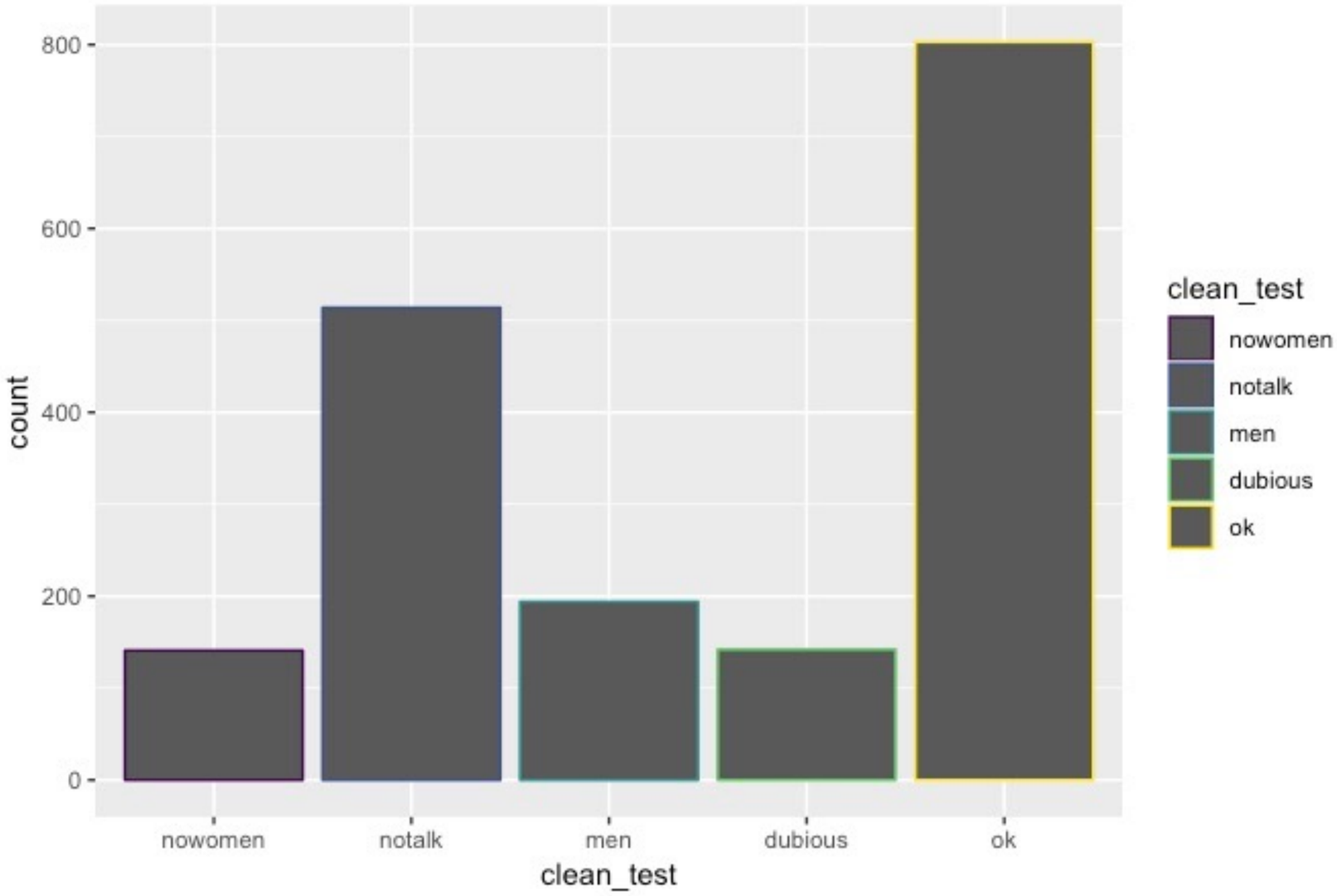

```
ggplot(data = bechdel) +
  geom_density(mapping = aes(x = budget, color=clean_test))
```

Your Turn 7

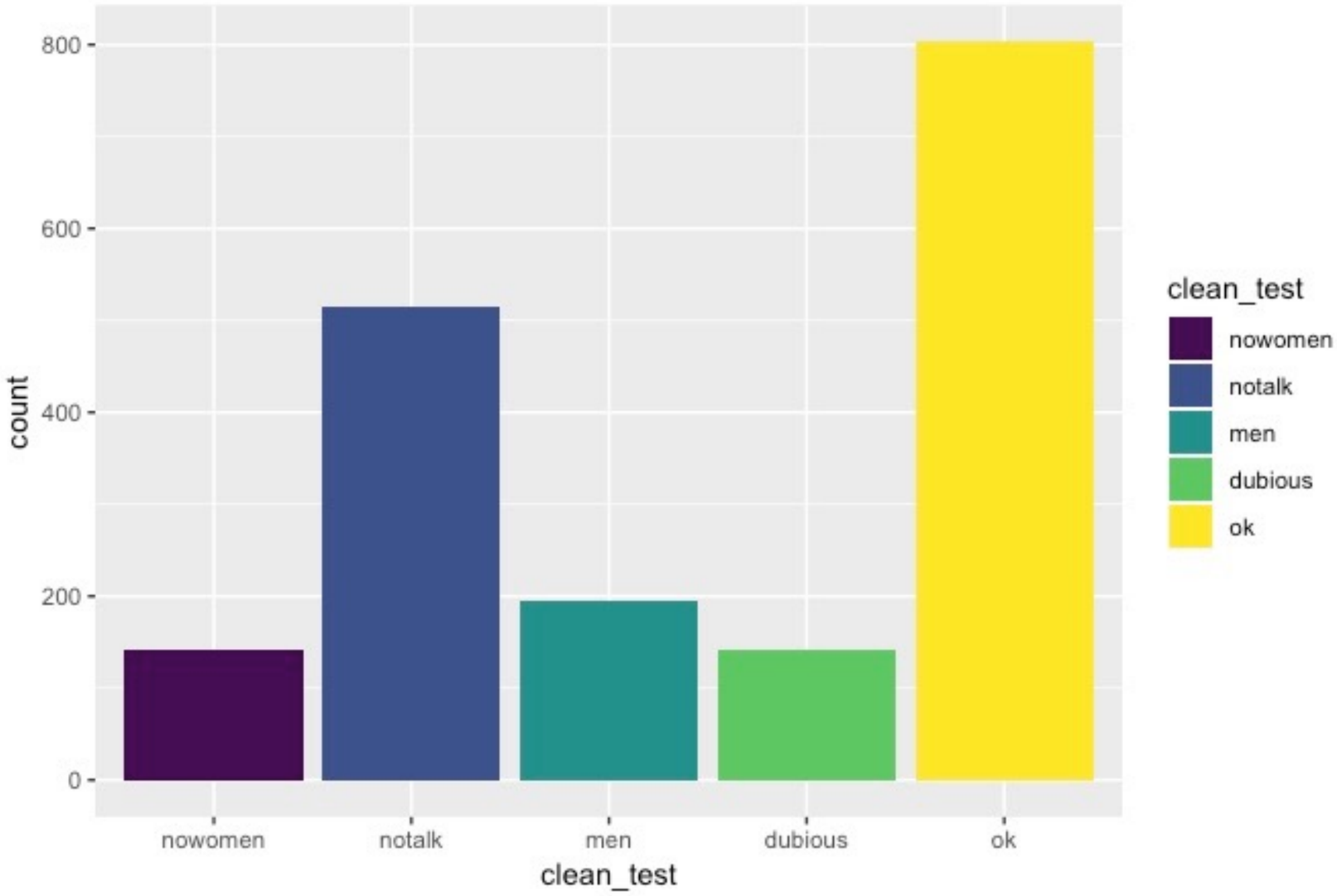
With your partner, make the bar chart of **clean_test** colored by **clean_test** below. Use the cheatsheet. Try your best guess.



02:00




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



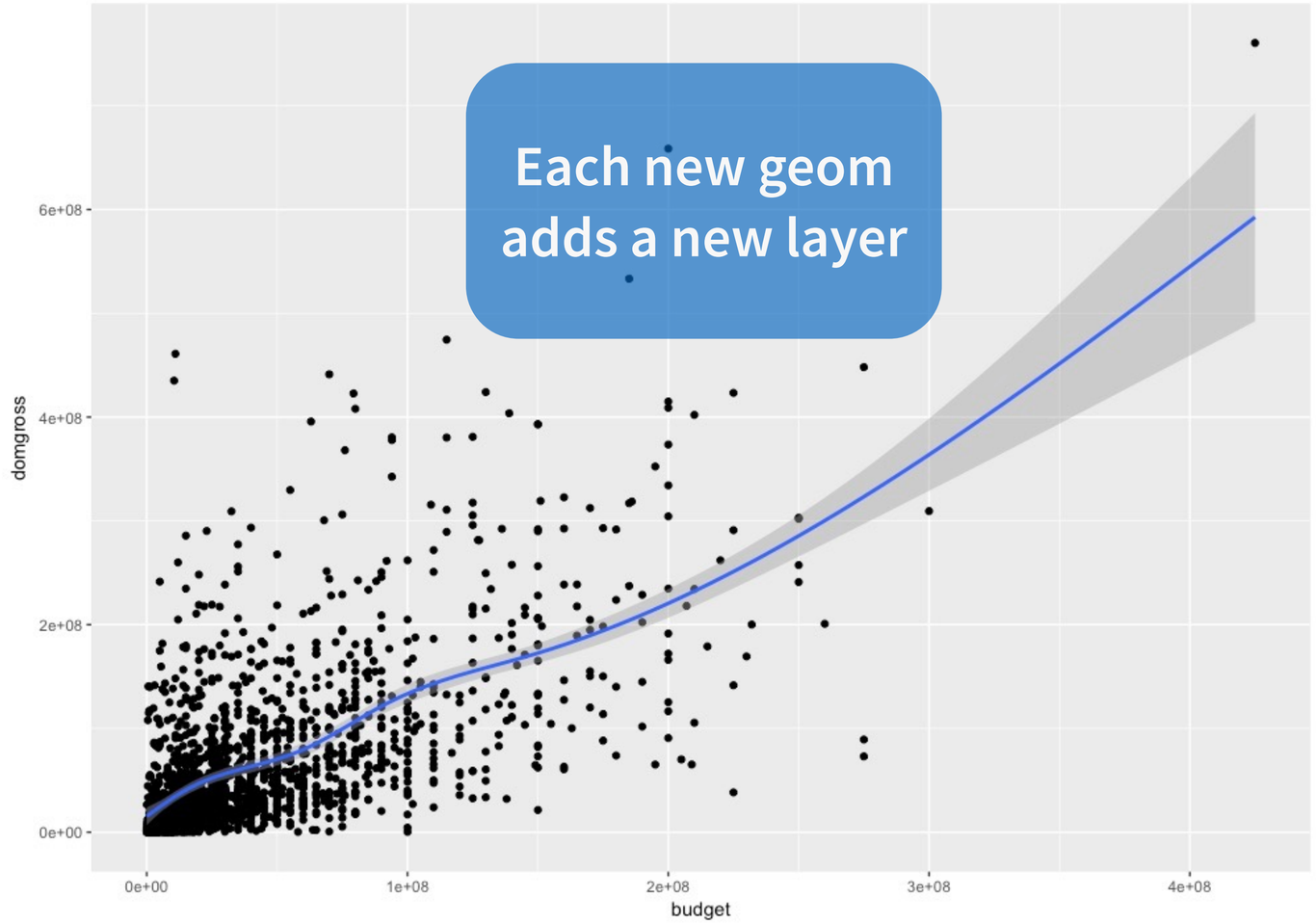

```
ggplot(data=bechdel) +
    geom_bar(aes(x=clean_test, fill=clean_test))
```

Your Turn 8

With a partner, predict what this code will do.

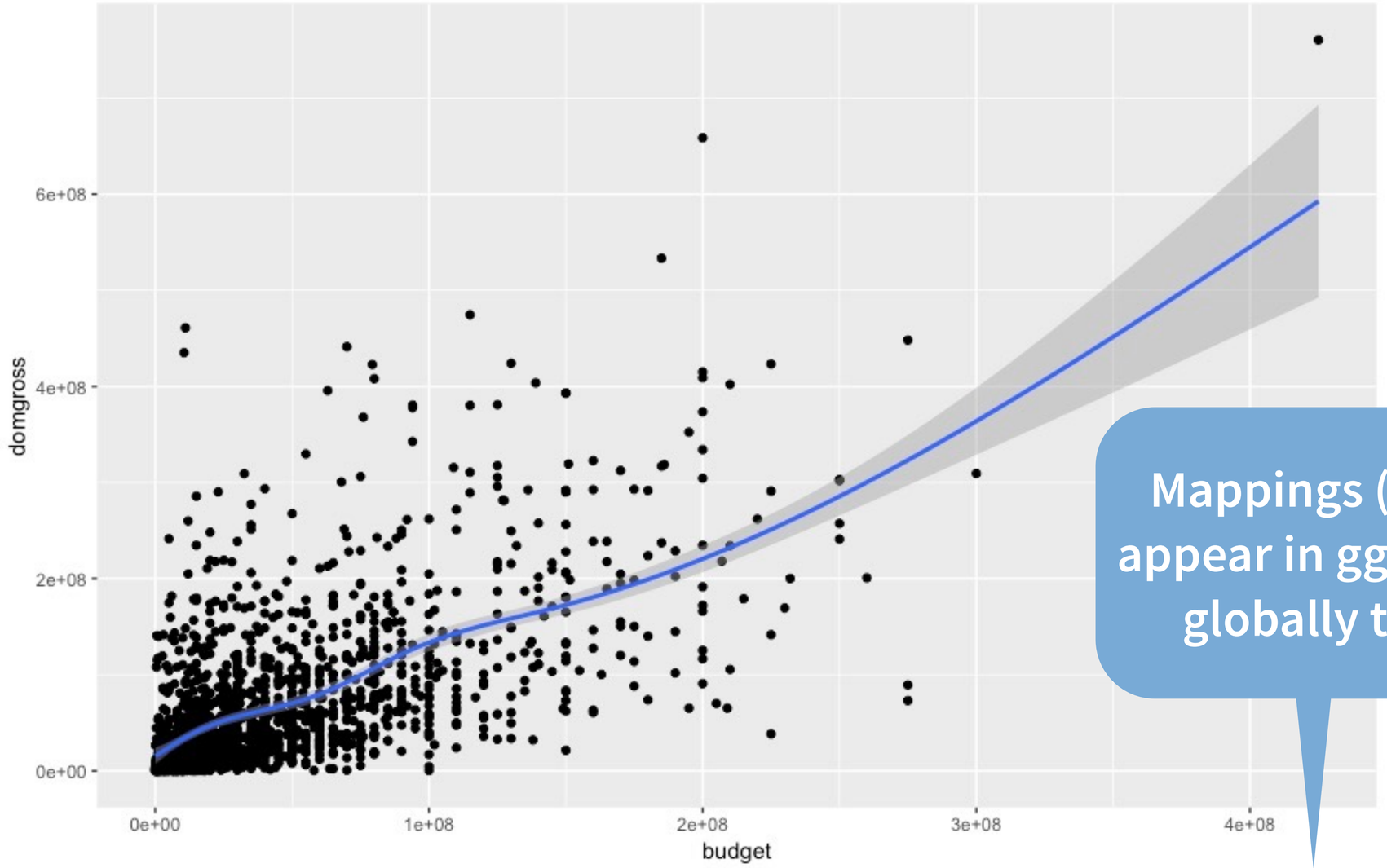
Then run it.

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



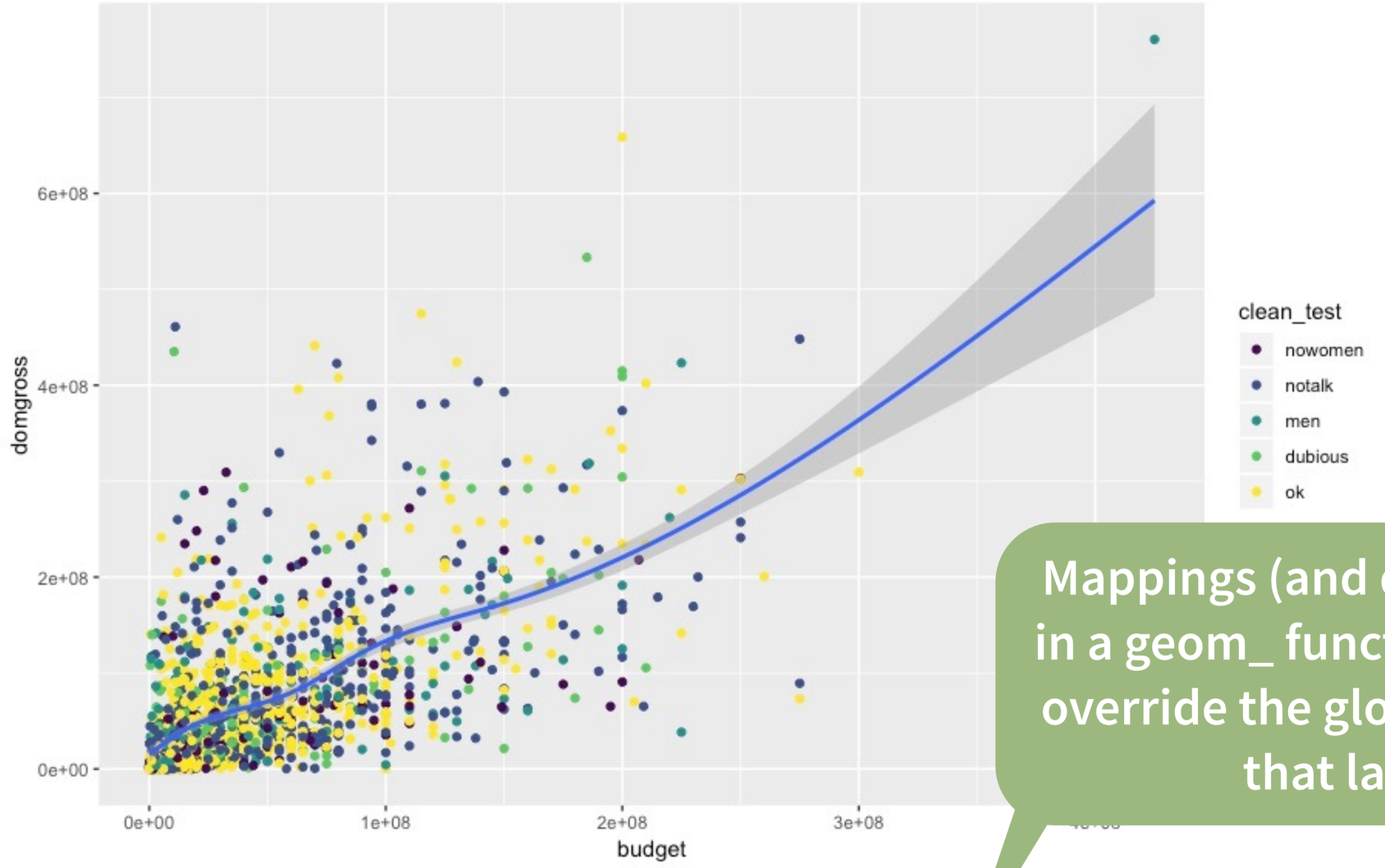

```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = budget, y = domgross)) +
  geom_smooth(mapping = aes(x = budget, y = domgross))
```

global vs. local



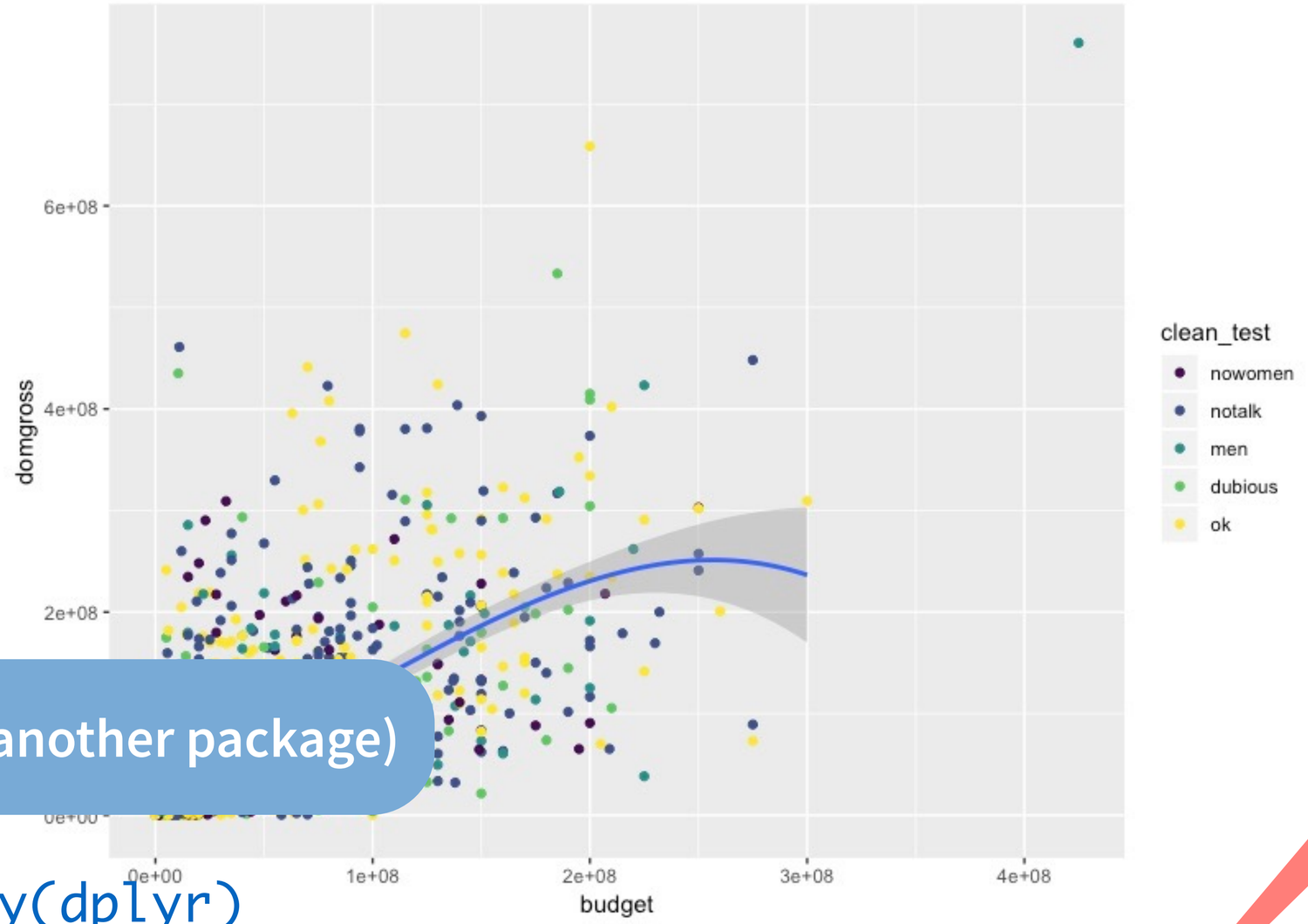
Mappings (and data) that appear in ggplot() will apply globally to every layer

```
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()
```



(requires another package)

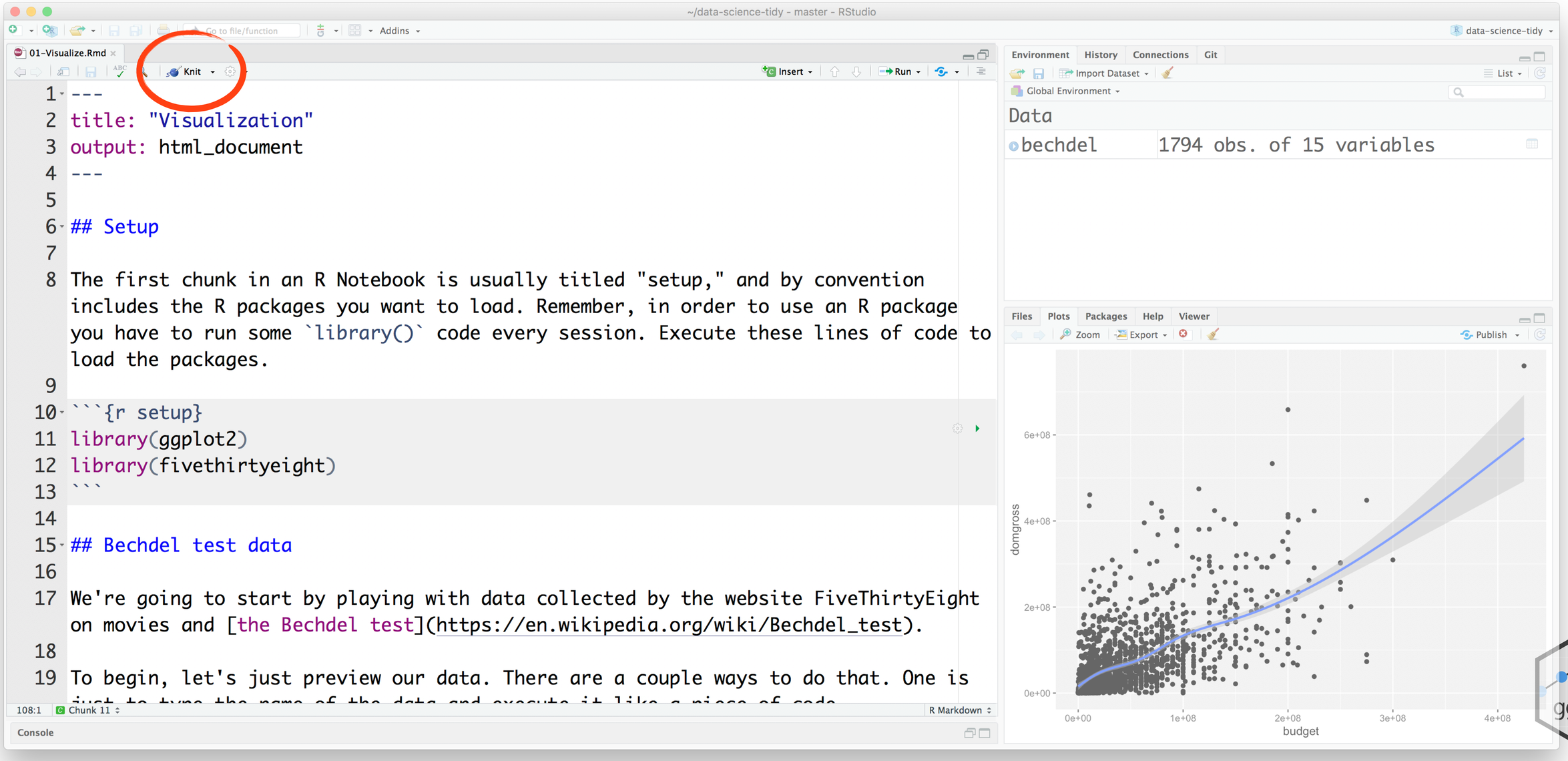
data can also be set locally or globally

```
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

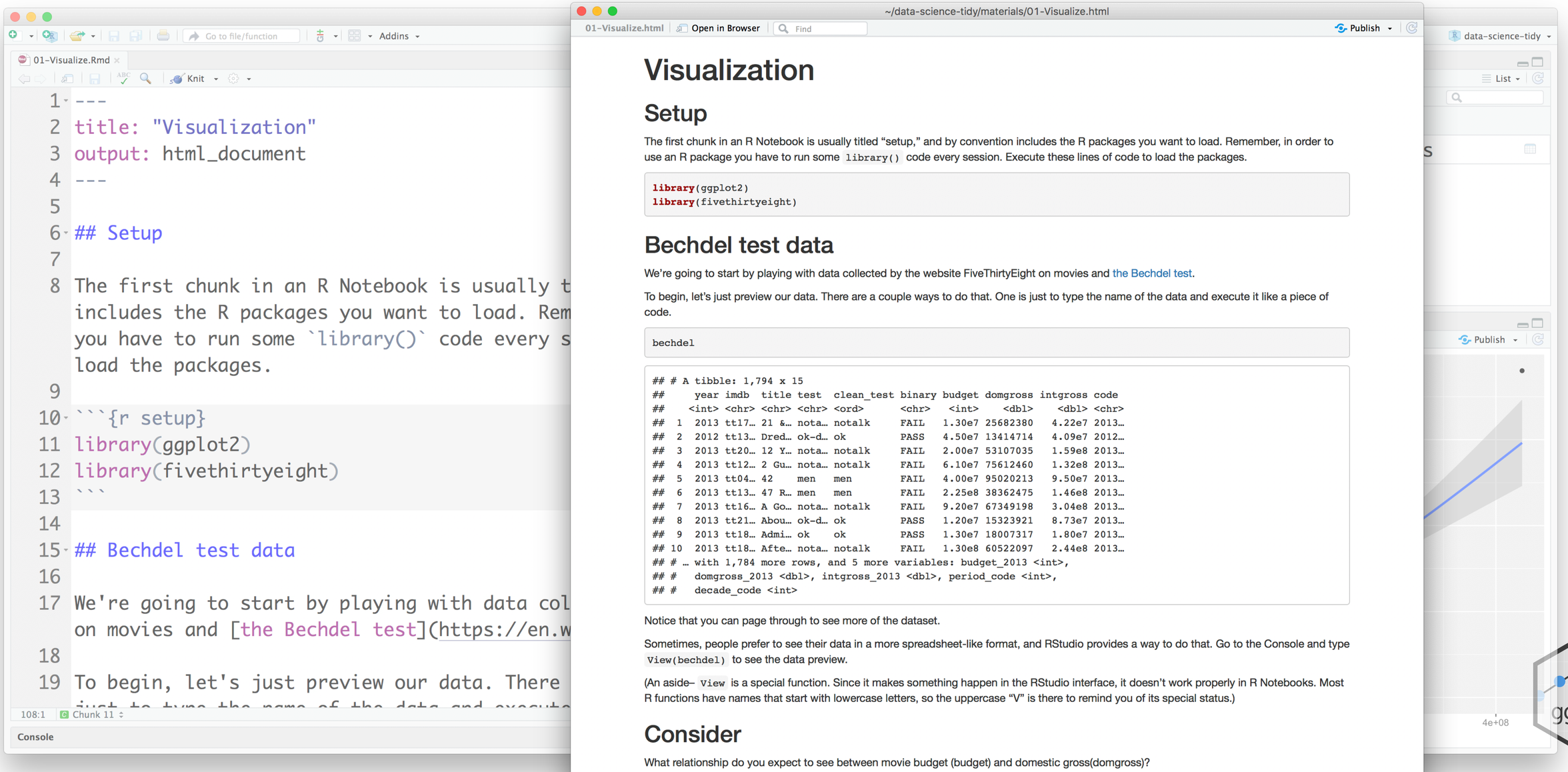
“knitting” an RMarkdown document

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



HTML preview

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



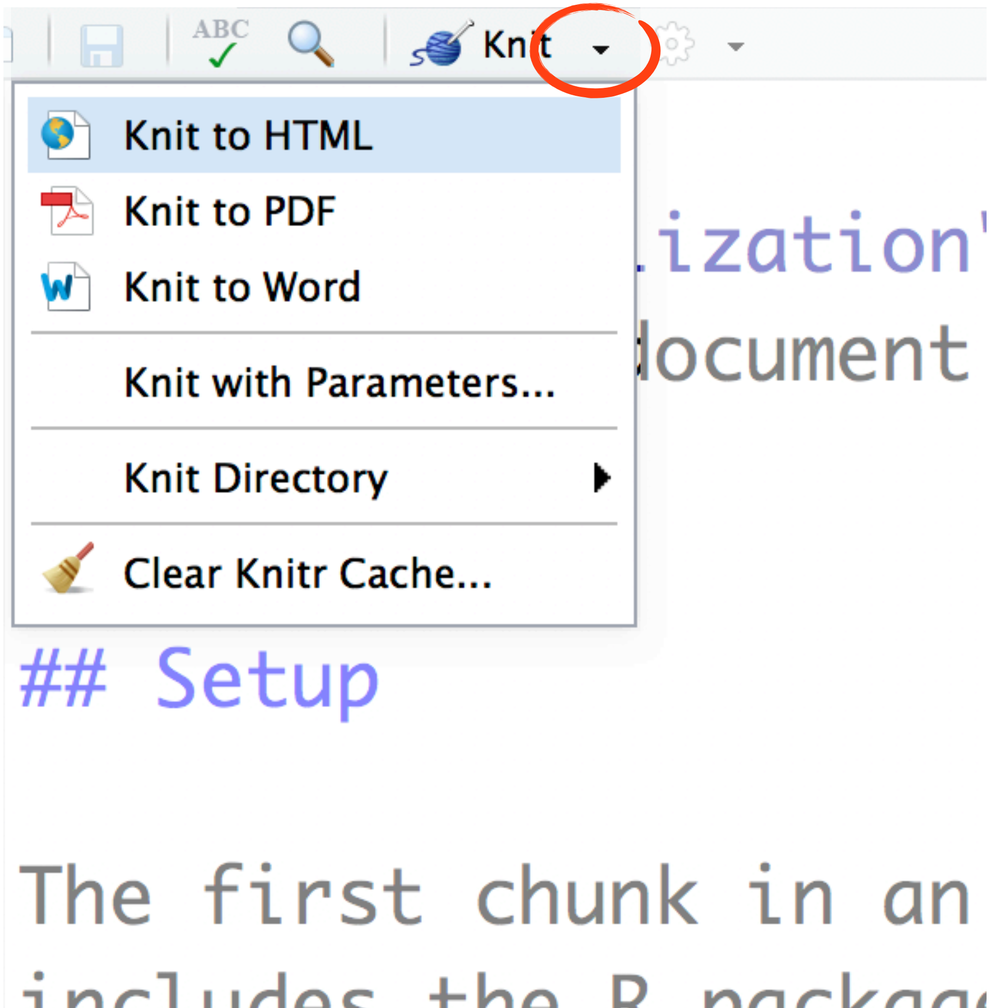
Sharing your work

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.



Your Turn

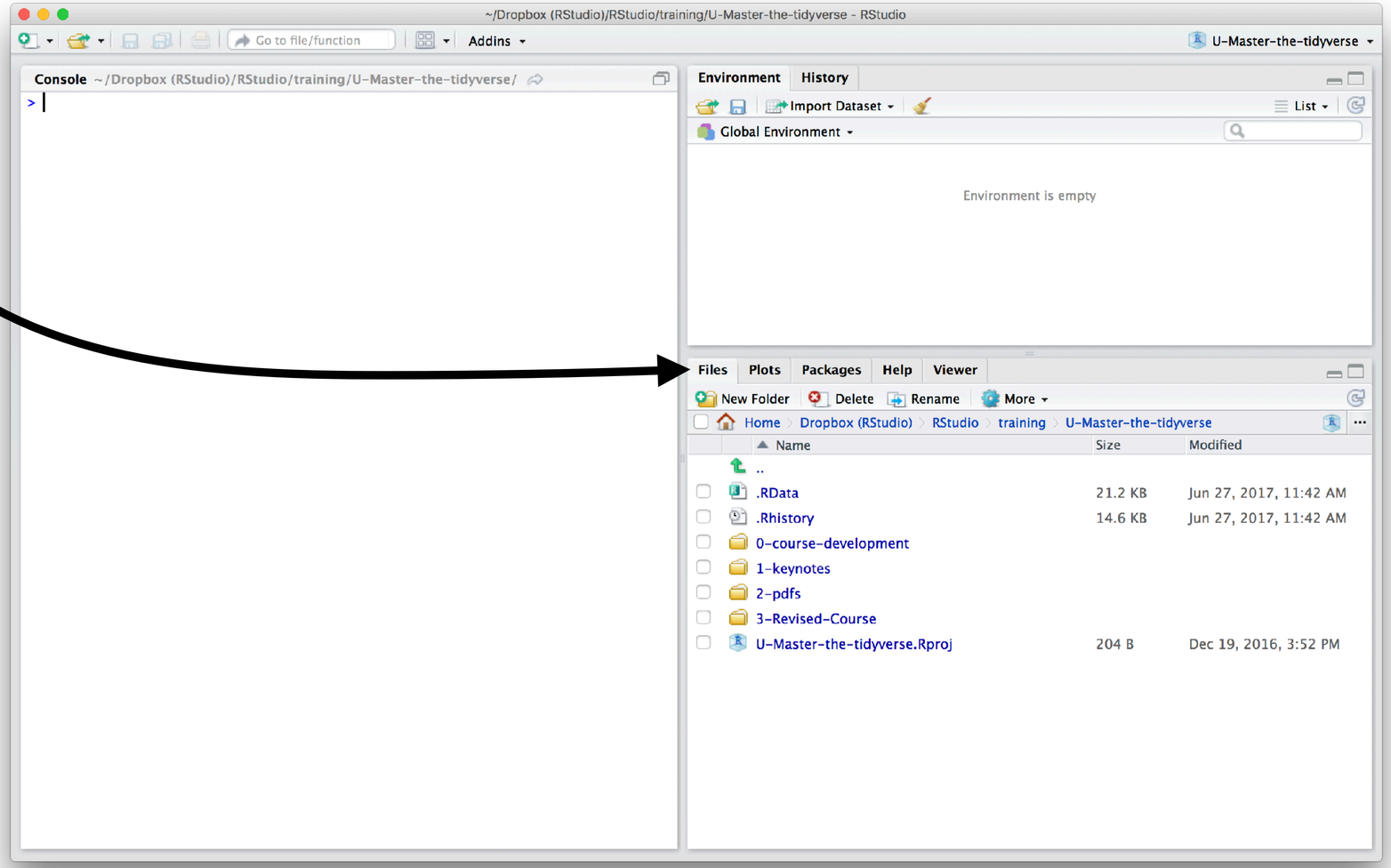
Locate the intro_to_ggplot2.html file in your Files pane. It will be located in your **working directory**

Working directory

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

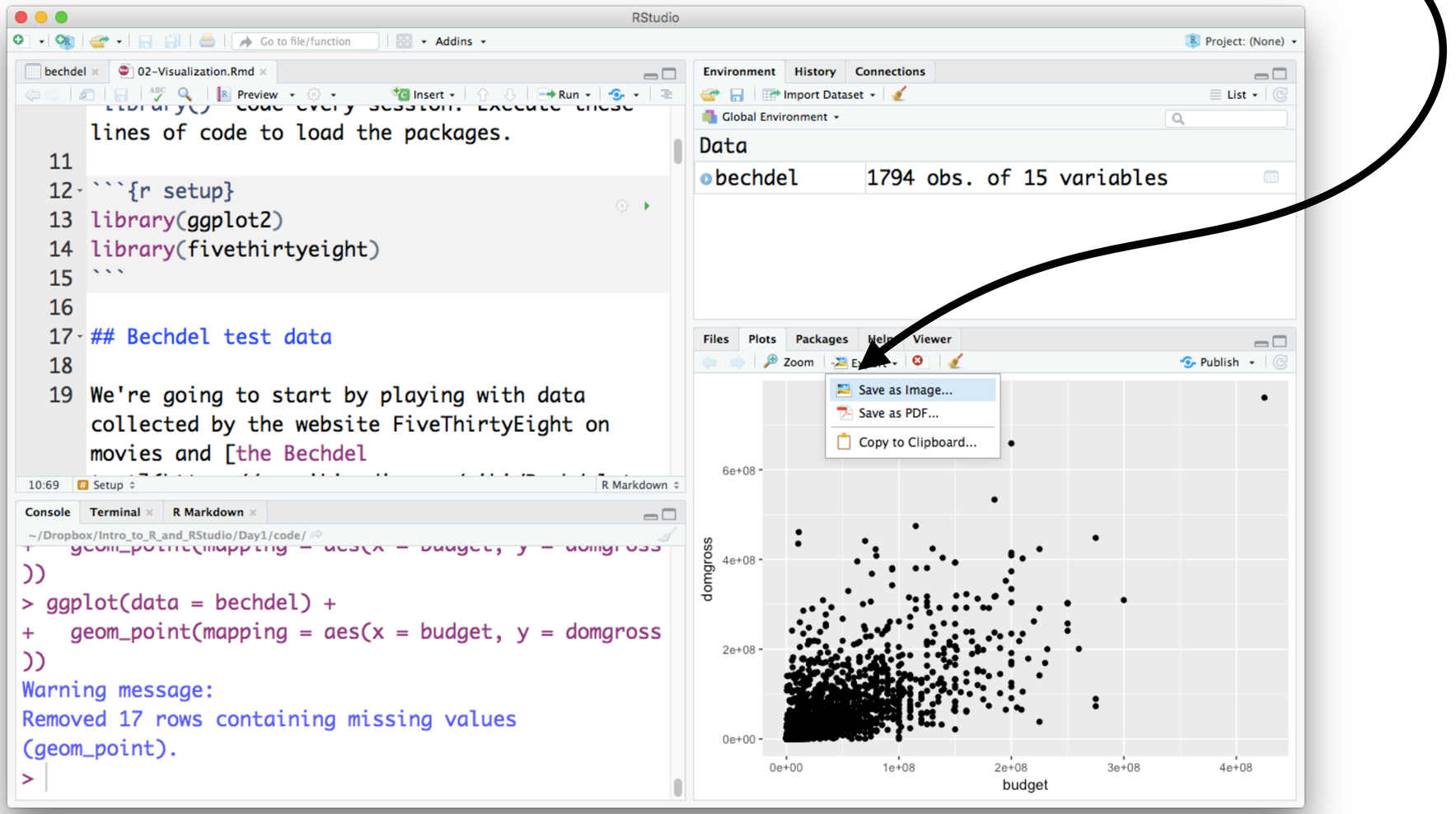
- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here
- If you are using **projects** you don't have to worry about working directories!

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:

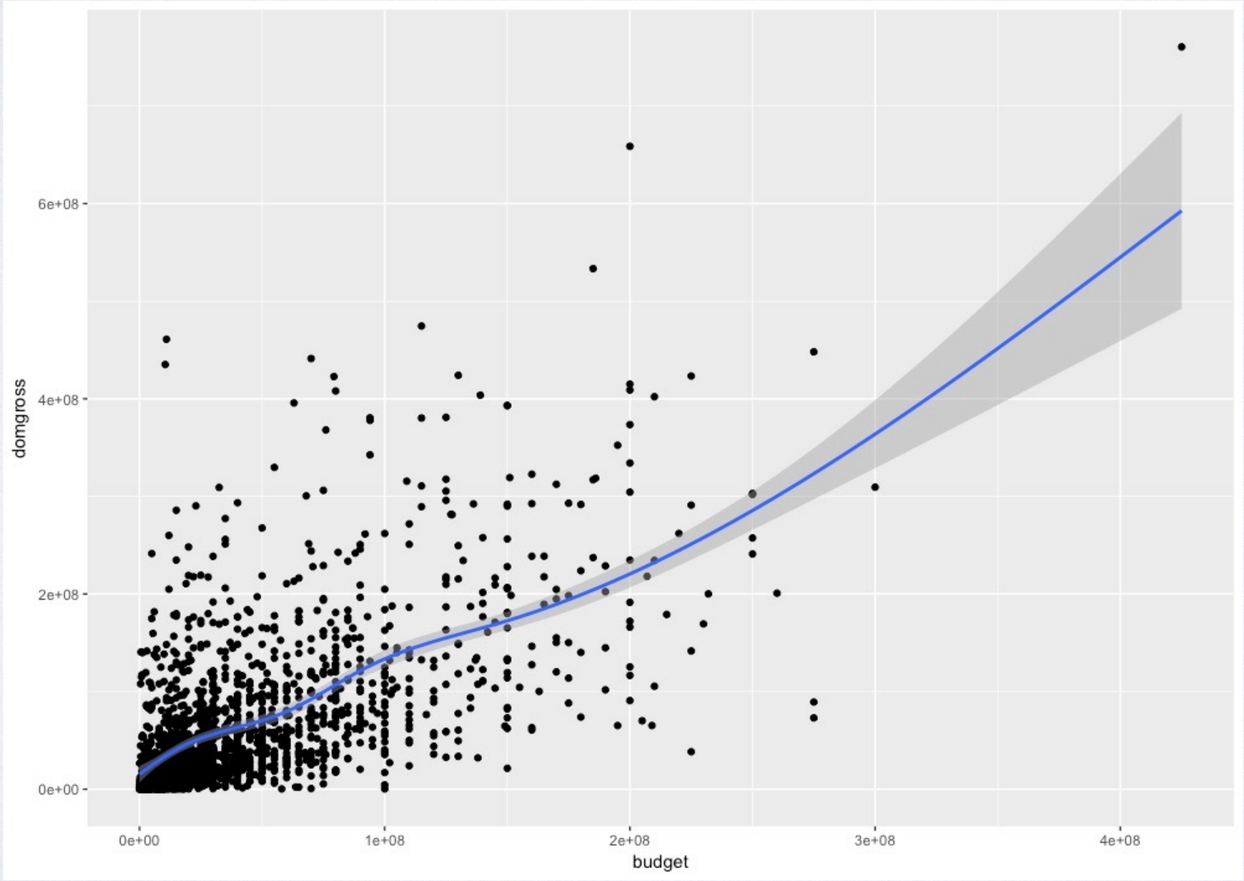
```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

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.

data **geom** x = F · y = A **coordinate system** **plot**

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

data **geom** x = F · y = A color = F size = A **coordinate system** **plot**

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>
```

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, 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

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

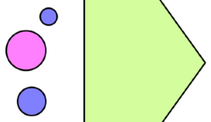
```
ggplot(data = gapminder,
```

2. Mapping

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

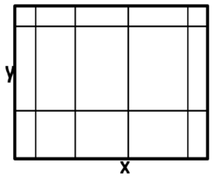
```
mapping =
aes(x = gdp,
y = lifespan,
color = continent,
size = pop))
```

3. Geom



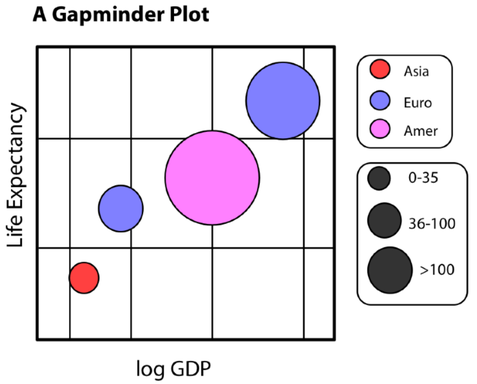
```
geom_point()
```

4. Co-Ordinates, Scales



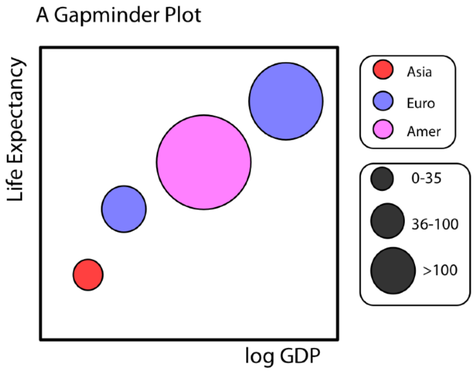

```
coord_cartesian()
scale_x_log10()
```

5. Labels & Guides




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

6. Themes




```
theme_minimal()
```

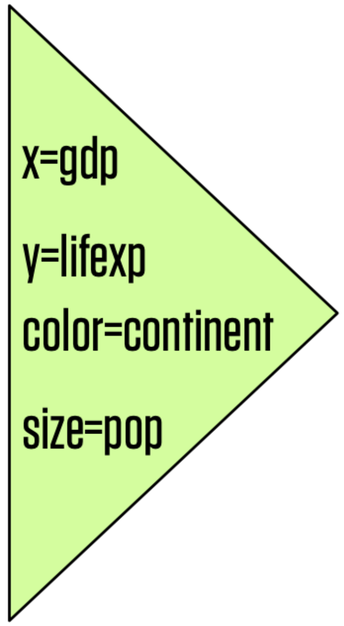
via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

1. Tidy Data

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

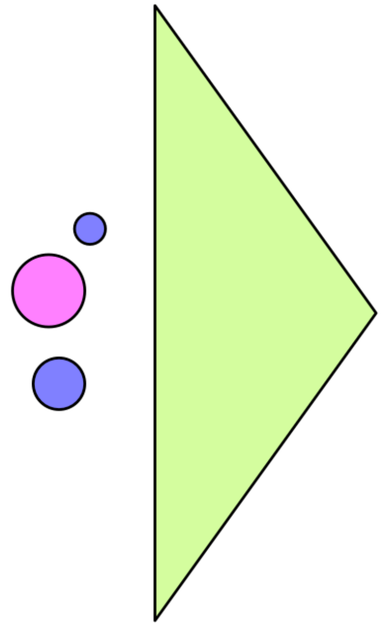
```
ggplot(data = gapminder,
```

2. Mapping



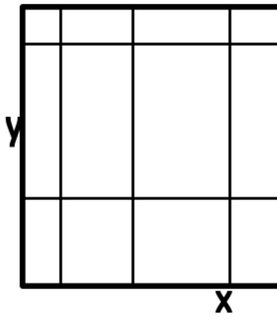

```
mapping =
aes(x = gdp,
y = lifespan,
color = continent,
size = pop))
```

3. Geom



```
geom_point()
```

4. Co-Ordin Scales




```
coord_cartes
 scale_x_log
```

via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

2. Mapping

3. Geom

4. Co-Ordinates, Scales

5. Labels & Guides

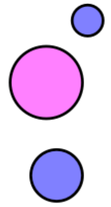
	continent
	Euro
	Amer
	Euro
	Asia

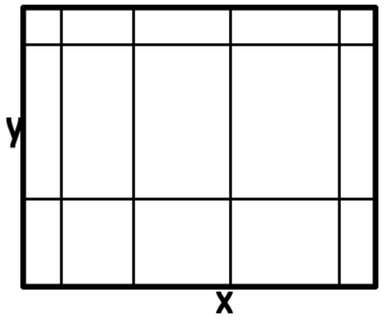
x=gdp

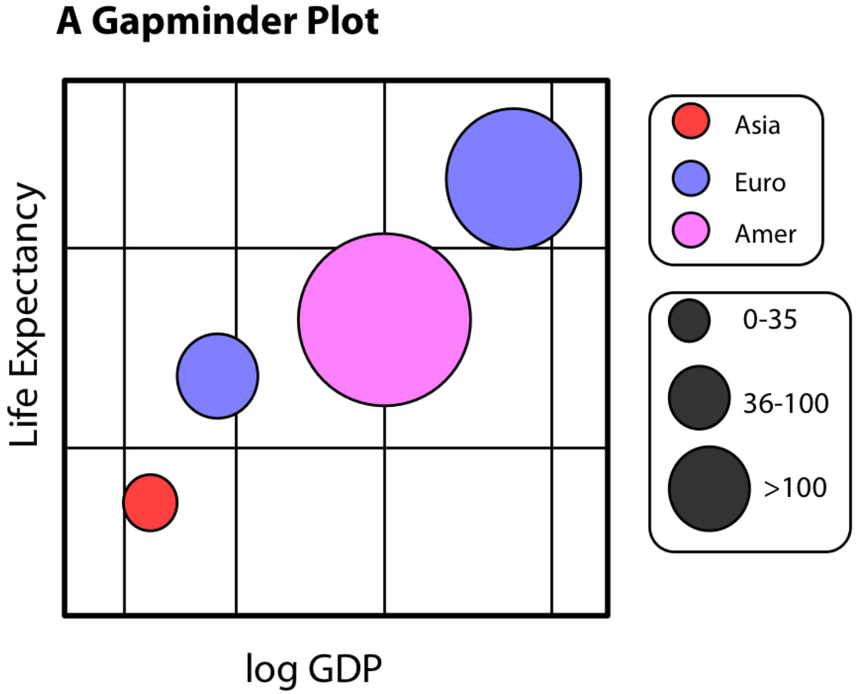
y=lifexp

color=continent

size=pop







a = gapminder,

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

geom_point()

coord_cartesian()
scale_x_log10()

labs()

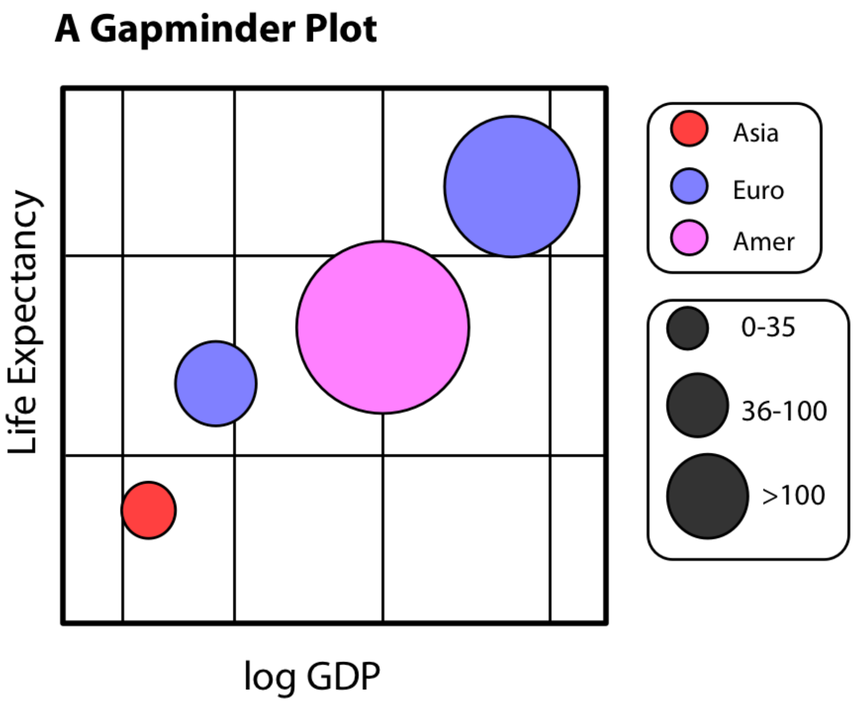
guides()

ates,

ian()

10()

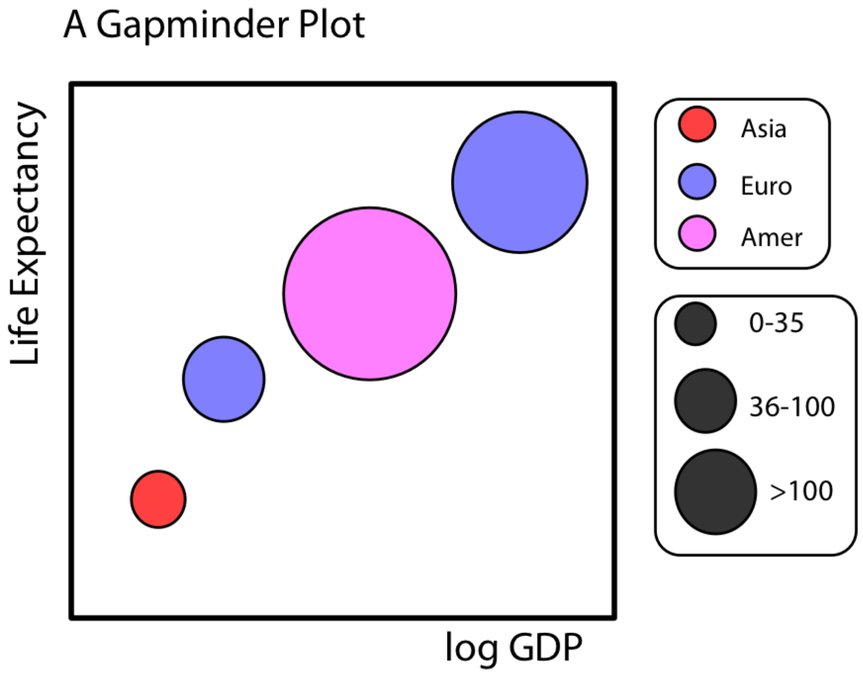
5. Labels & Guides



labs()

guides()

6. Themes



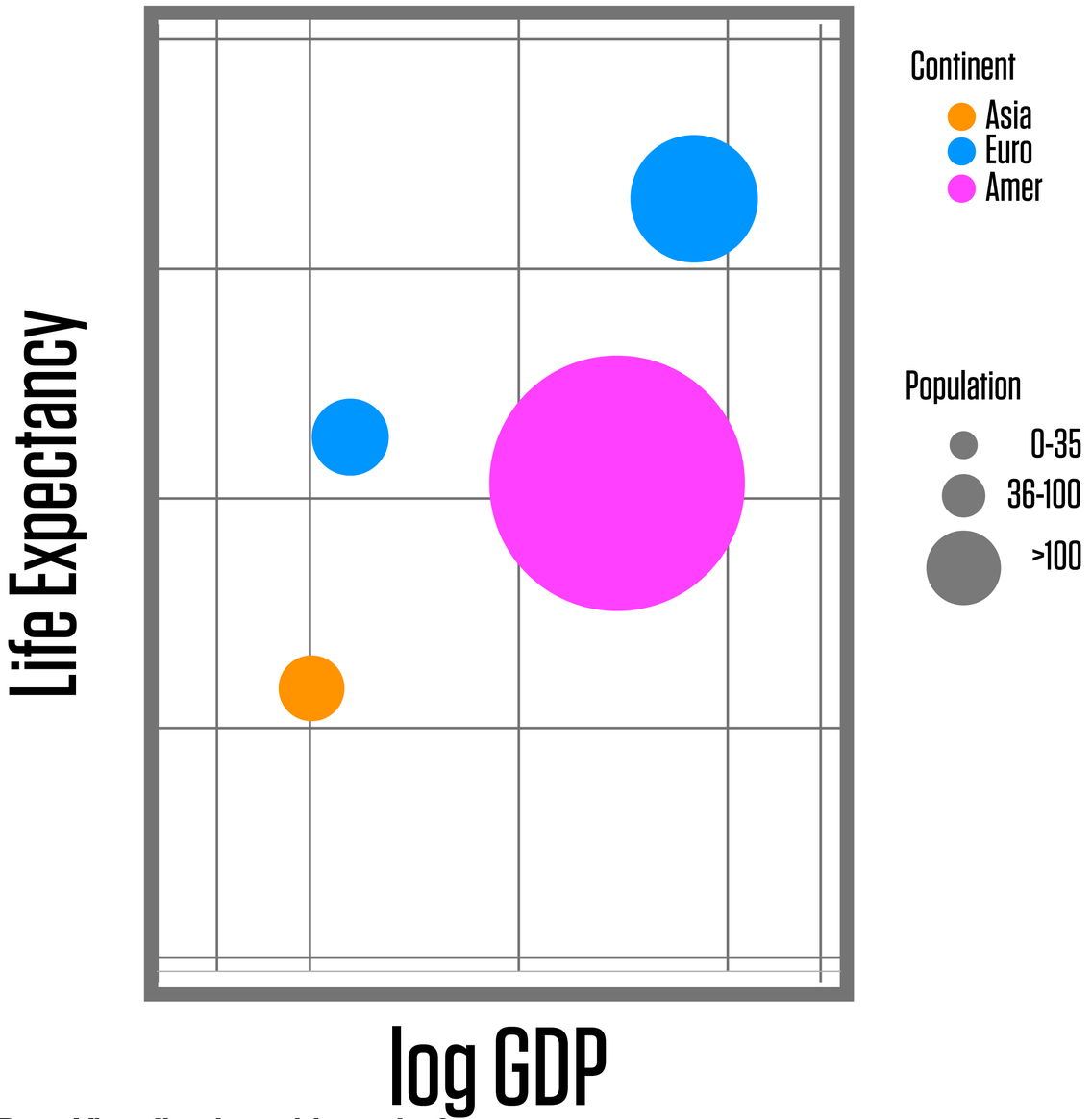
theme_minimal()

via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

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

via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

A Gapminder Plot



via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

1. Tidy Data

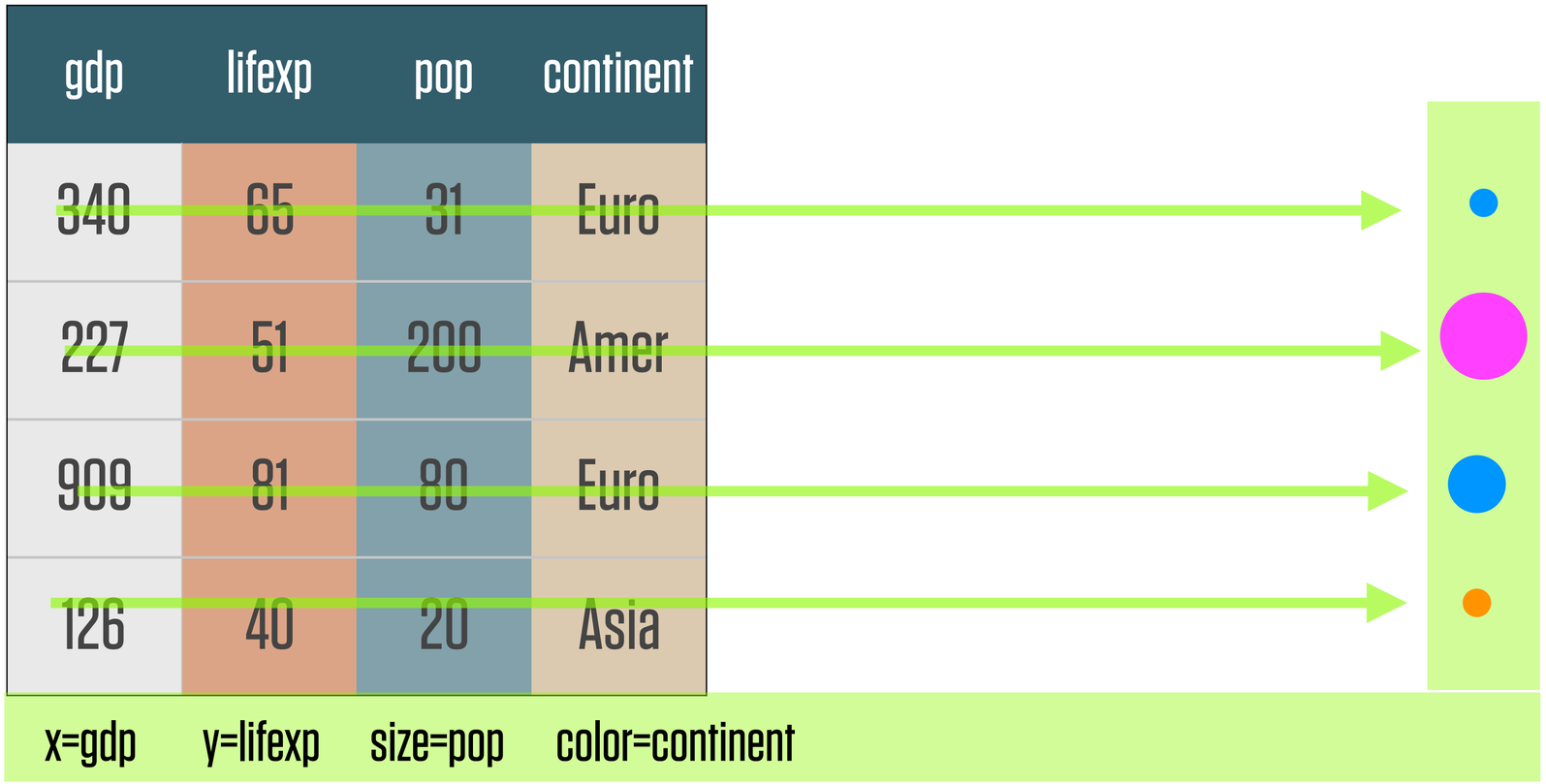
ggplot(data = gapminder)

2. Mapping

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

3. Geom

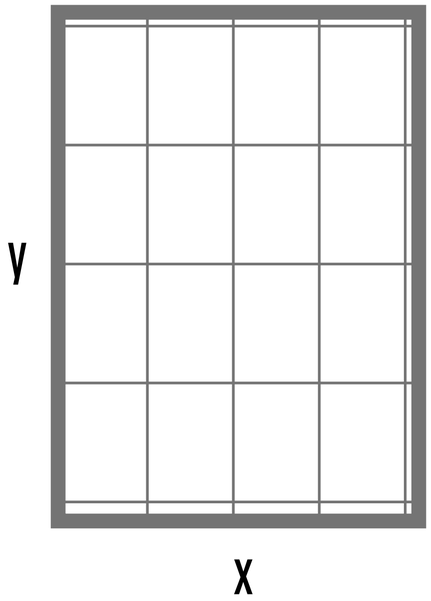
geom_point()



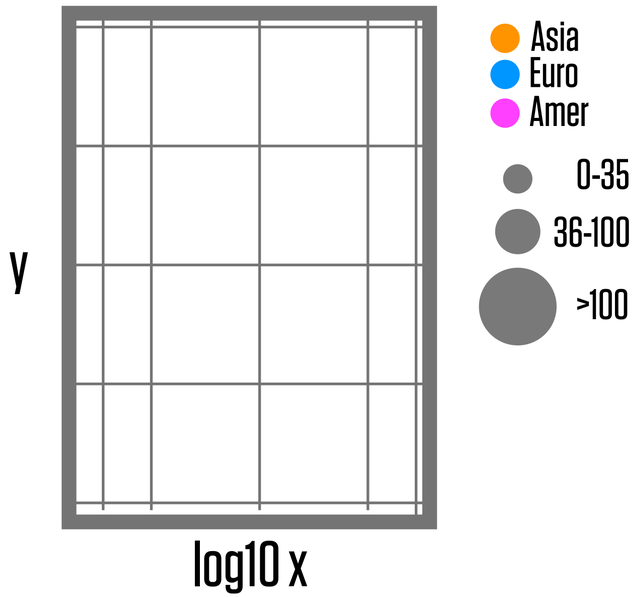
Required

via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

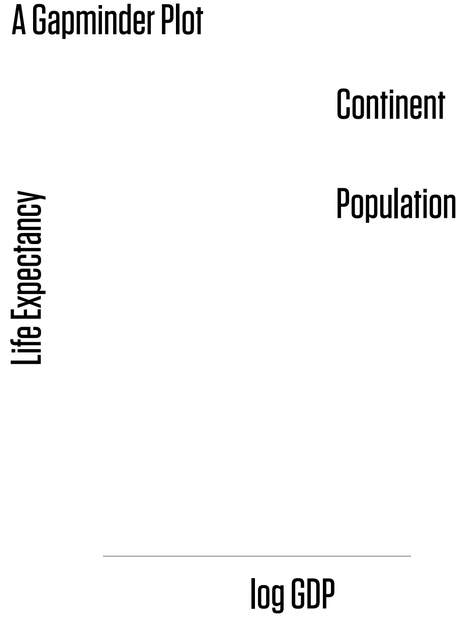
4. Coordinate System



5. Scales



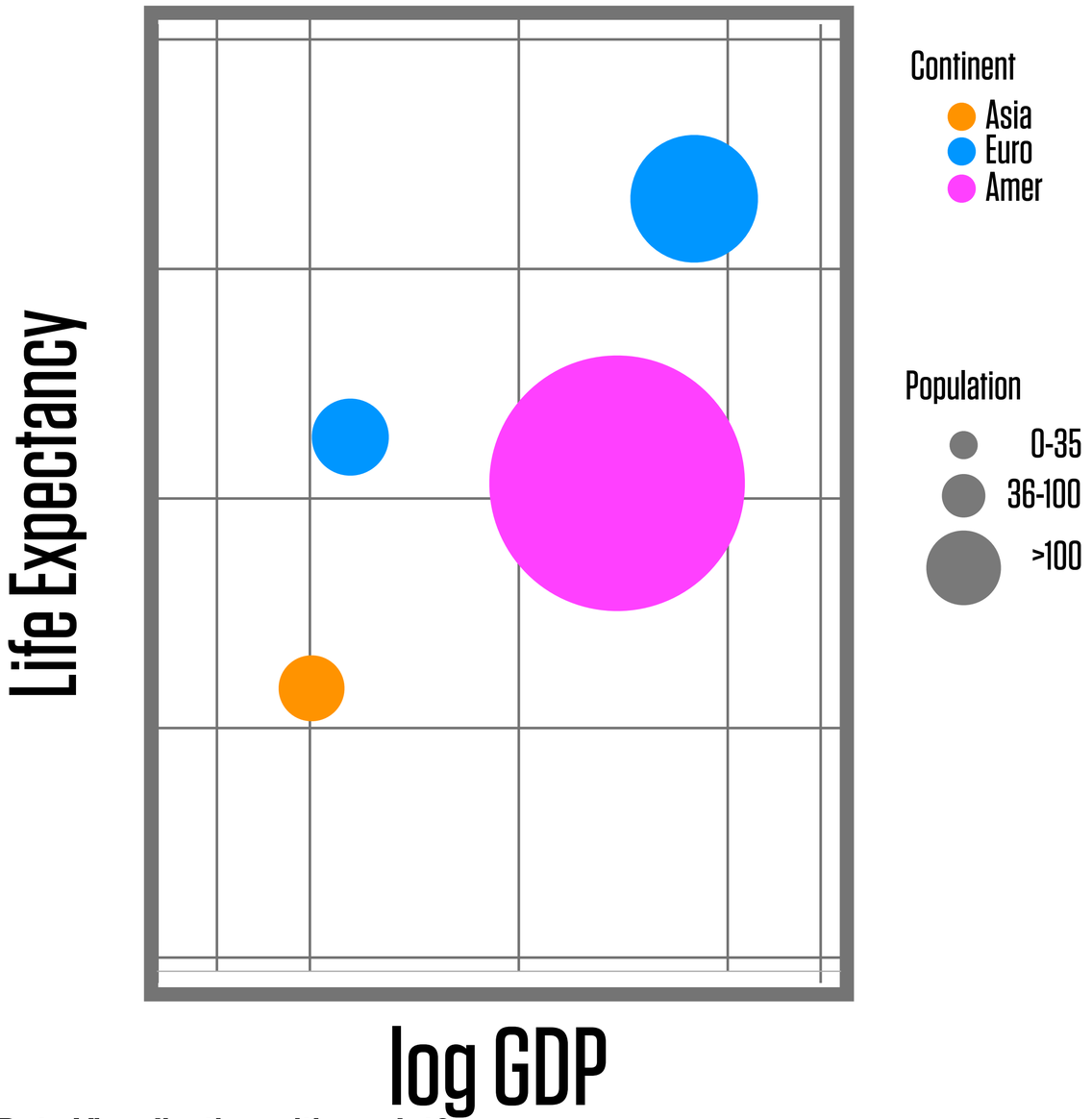
6. Labels & Guides



Initially Implicit

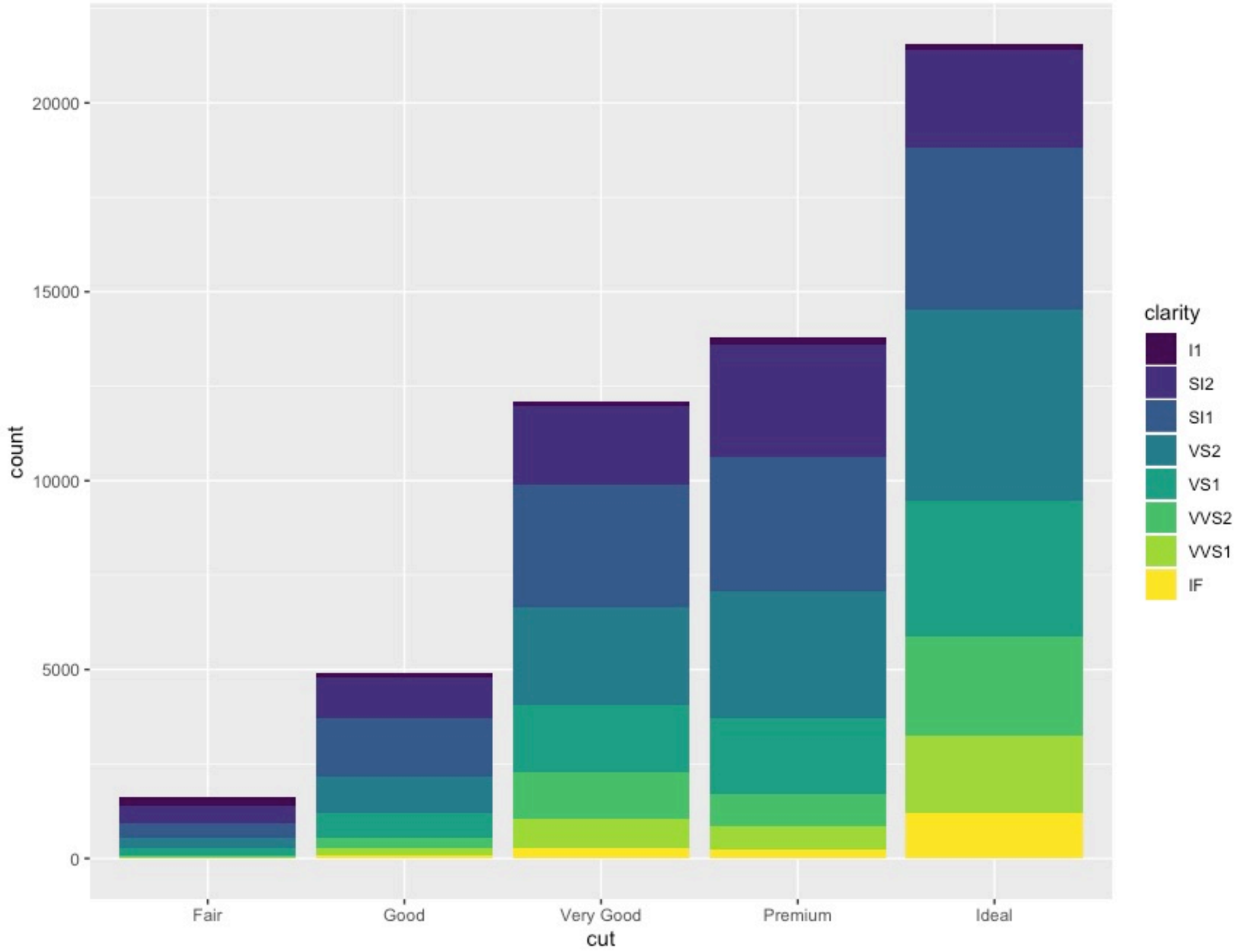
via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

A Gapminder Plot



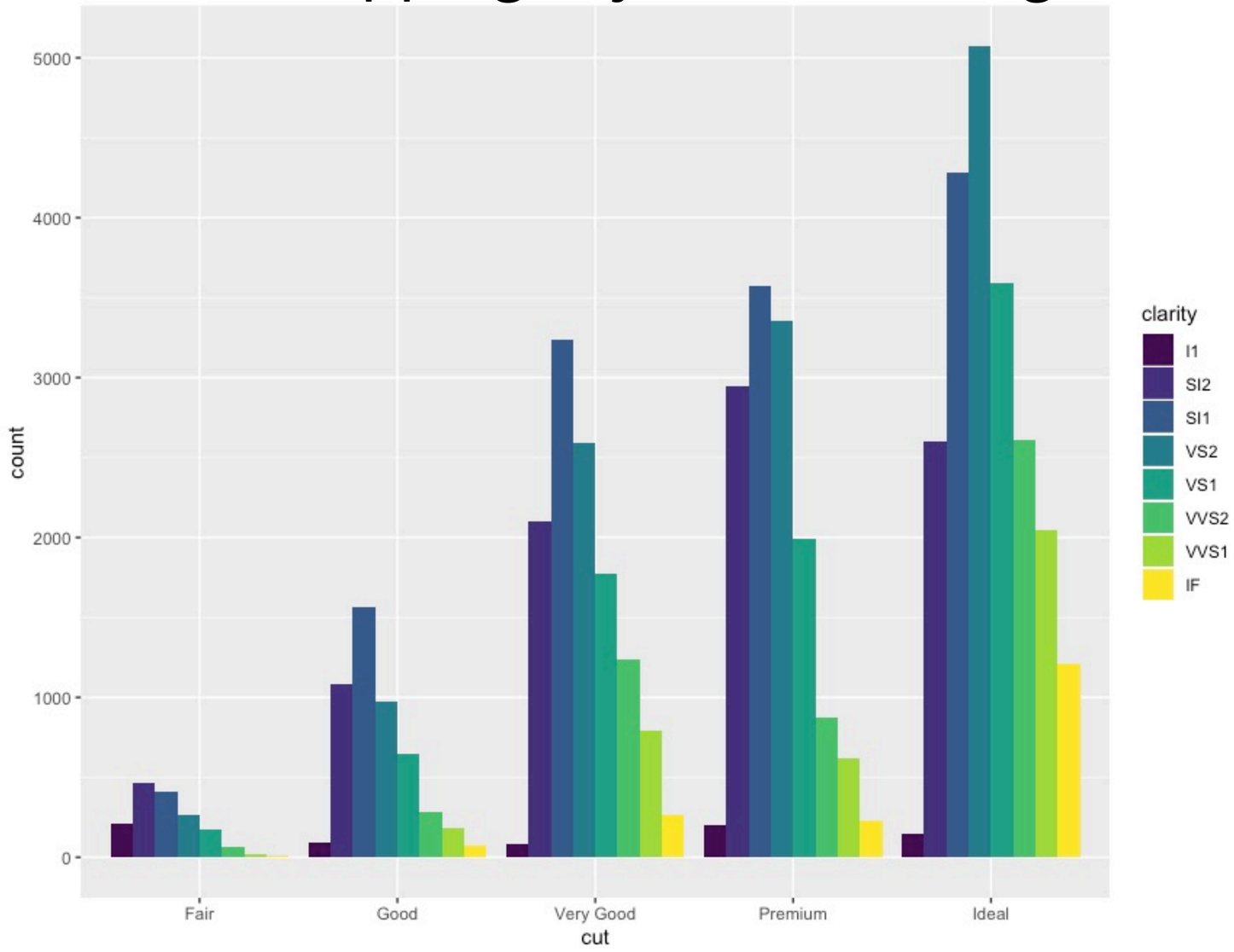
via Kieran Healey's A Practical Introduction to Data Visualization with ggplot2

What else?


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

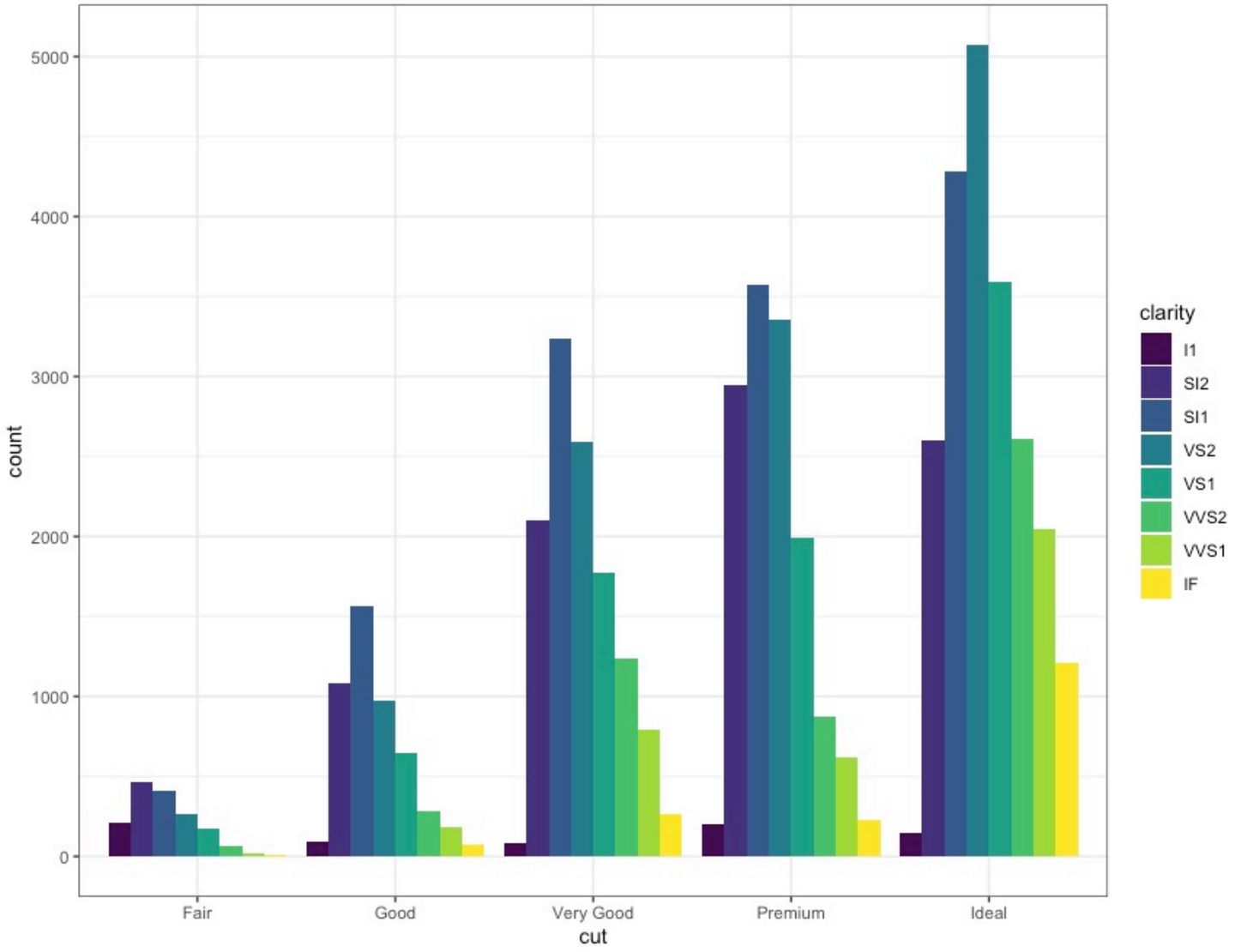
Position Adjustments

How overlapping objects are arranged



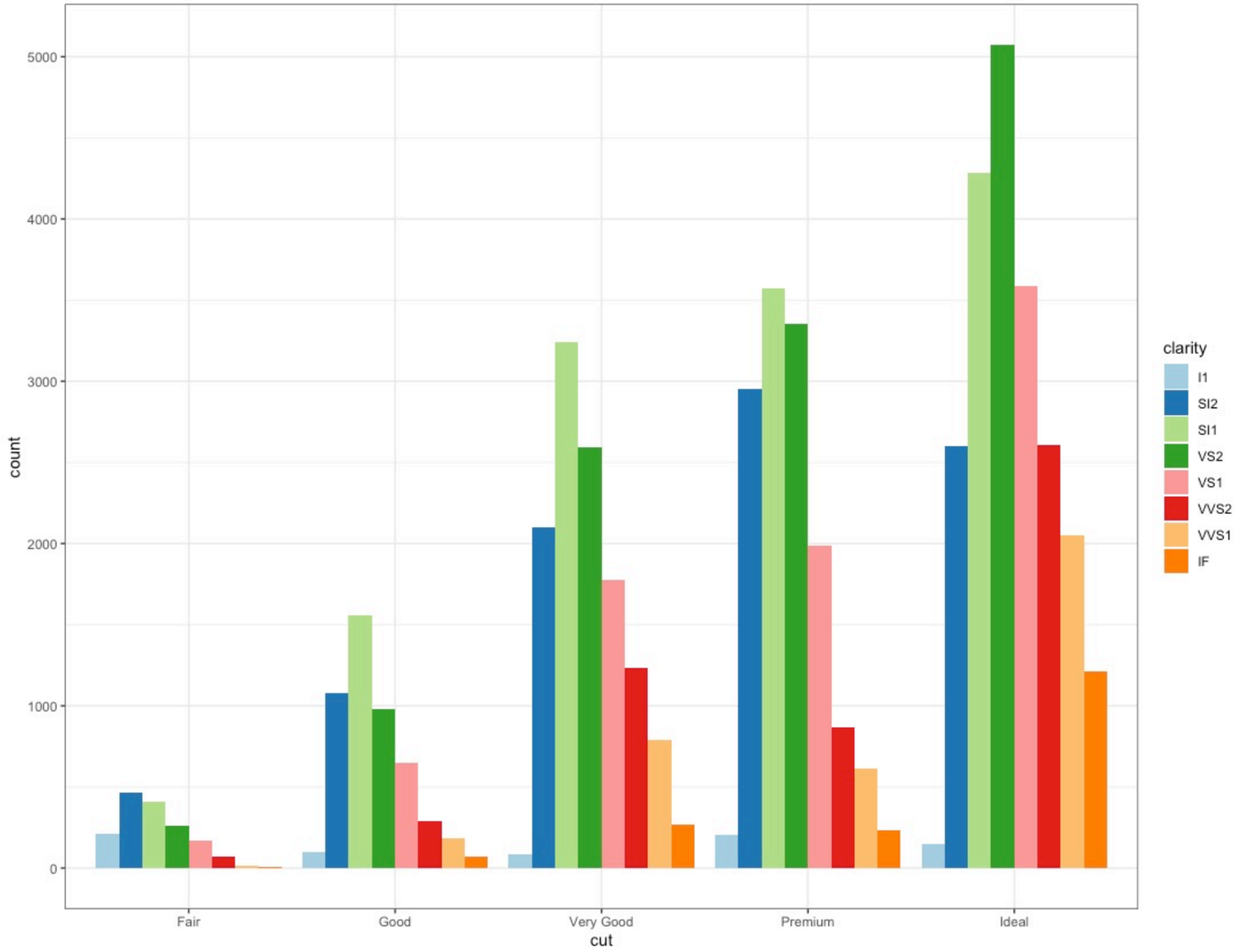
Themes

Visual appearance of non-data elements



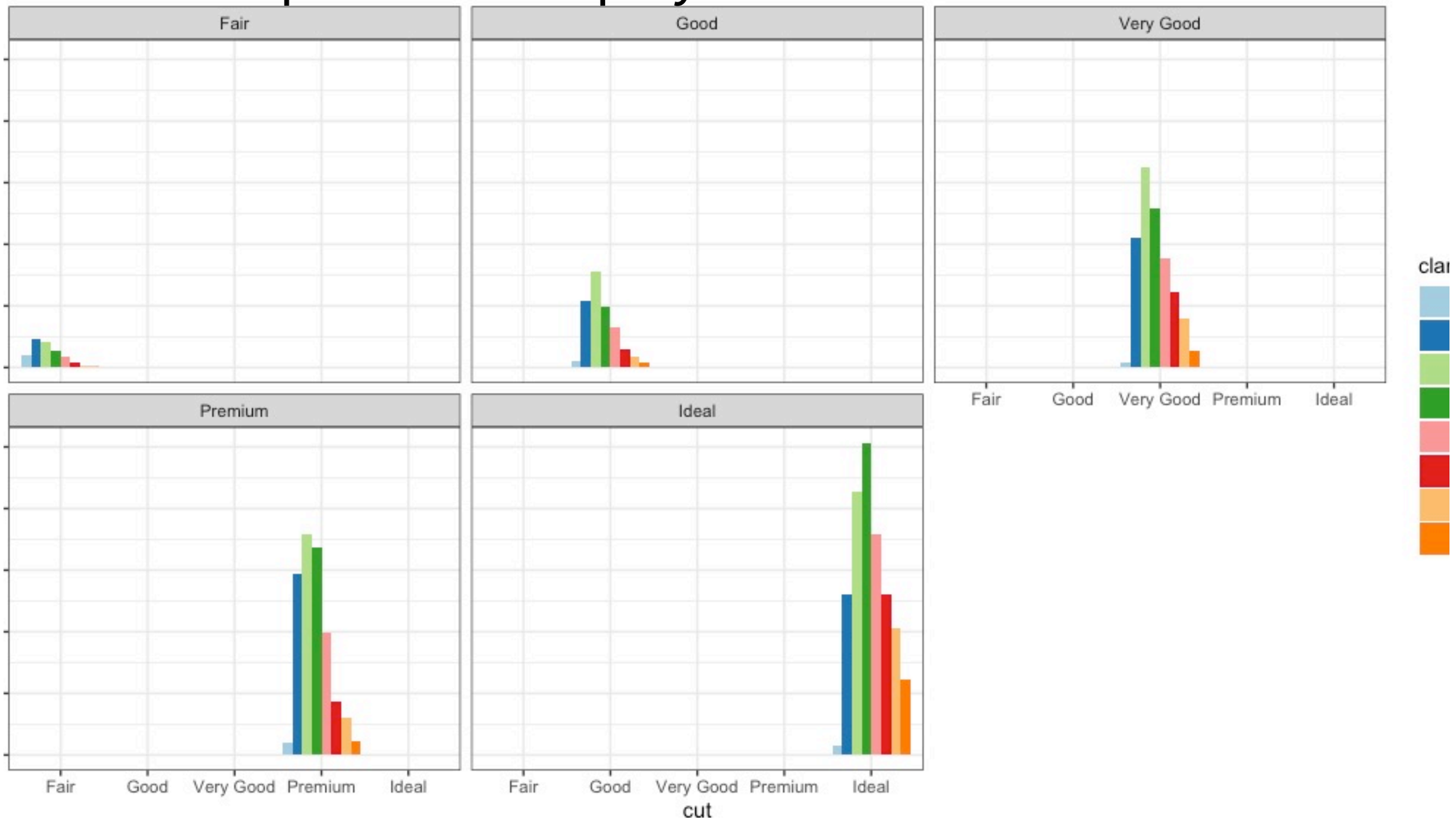
Scales

Customize color scales, other mappings

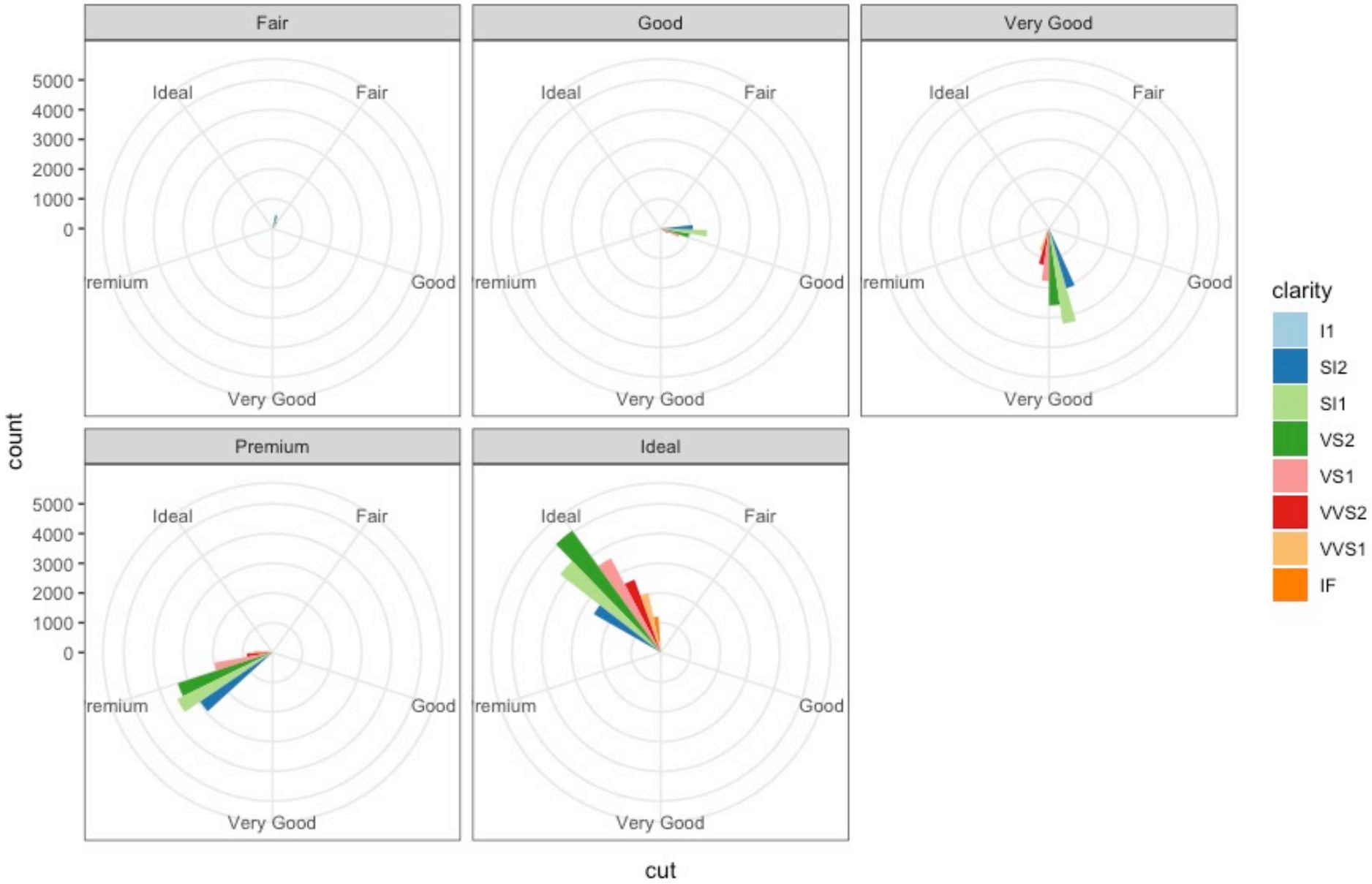


Facets

Subplots that display subsets of the data.

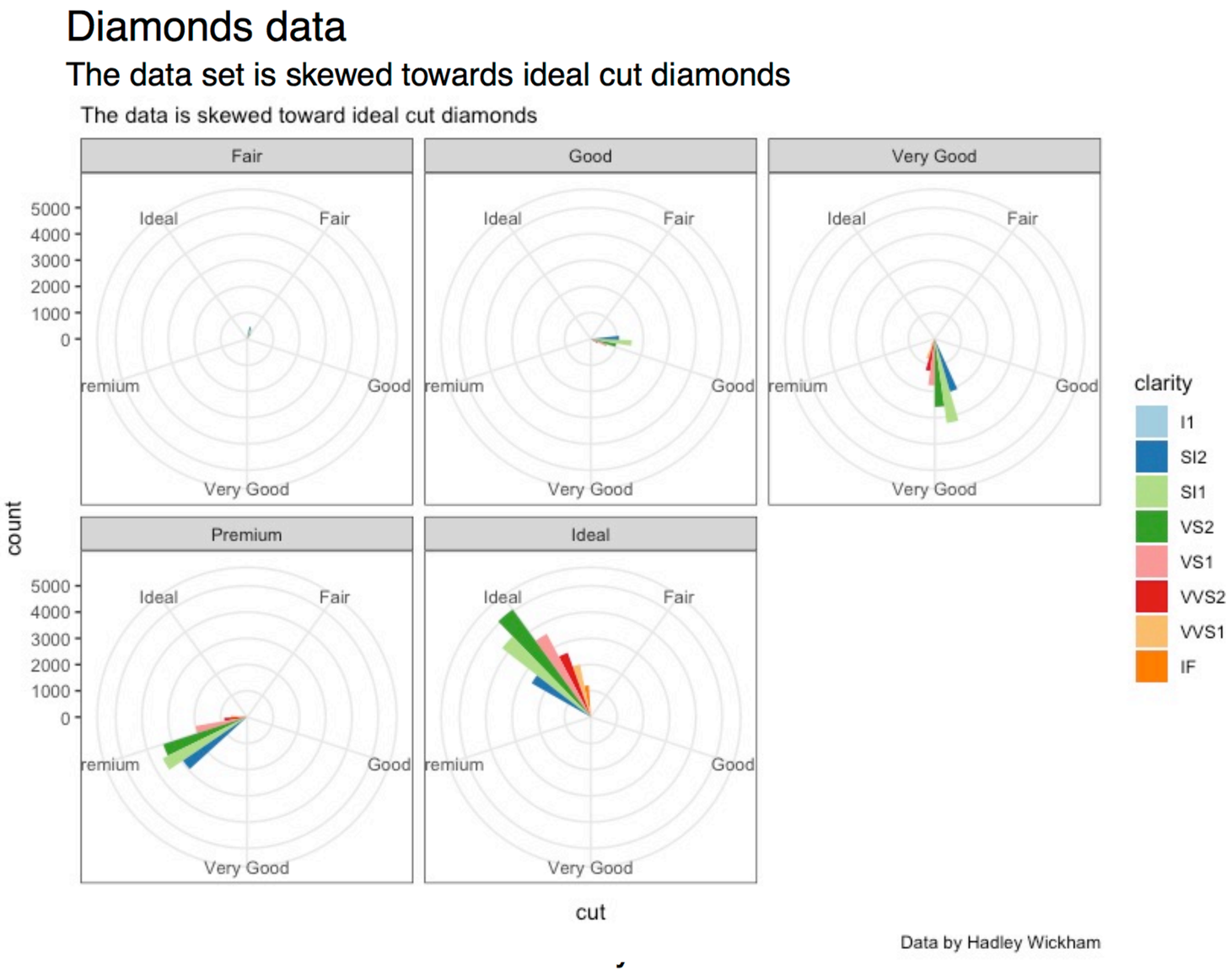


Coordinate systems



ggplot2

Titles and captions

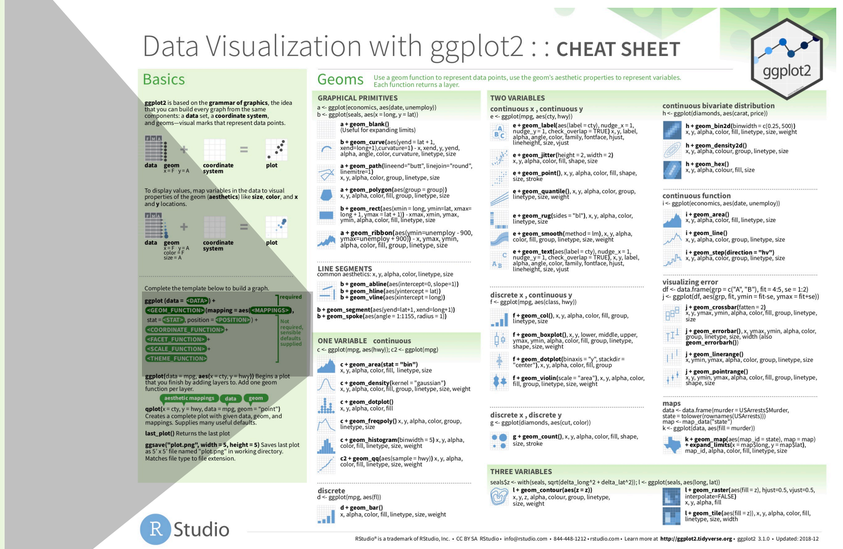


Data by Hadley Wickham

A ggplot2 template

Make any plot by filling in the parameters of this template

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



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

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, 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

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

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

RStudio®

ggplot2.tidyverse.org

Create Elegant Data Visualisatic

https://ggplot2.tidyverse.org

ggplot2 part of the tidyverse 3.2.1

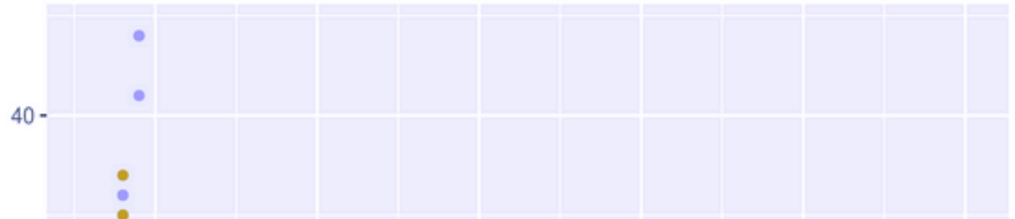
Reference Articles News Extensions

Usage

It's hard to succinctly describe how ggplot2 works because it embodies a deep philosophy of visualisation. However, in most cases you start with `ggplot()`, supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_flip()`).

```
library(ggplot2)

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



Developers

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...