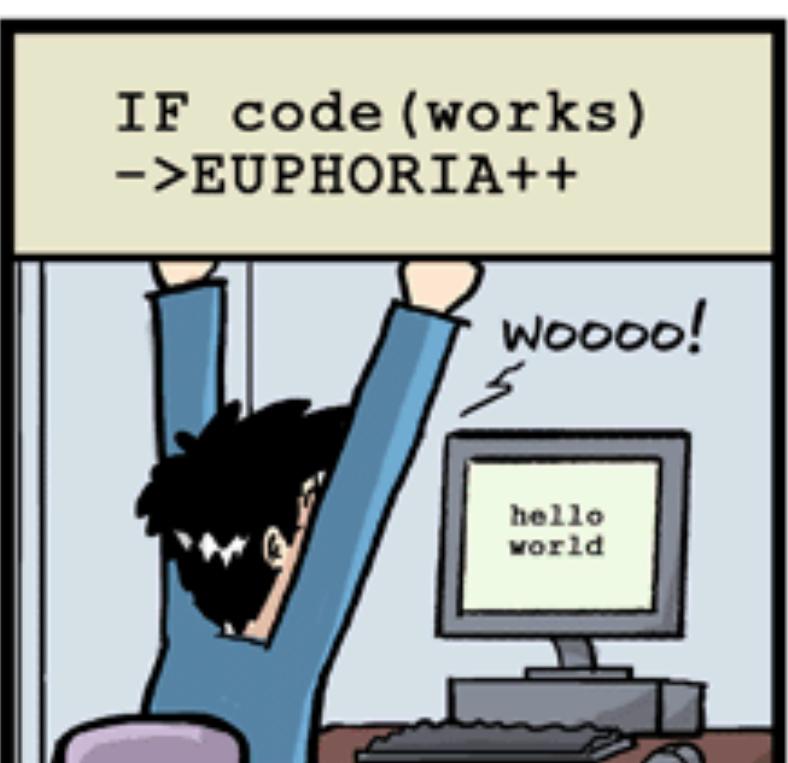
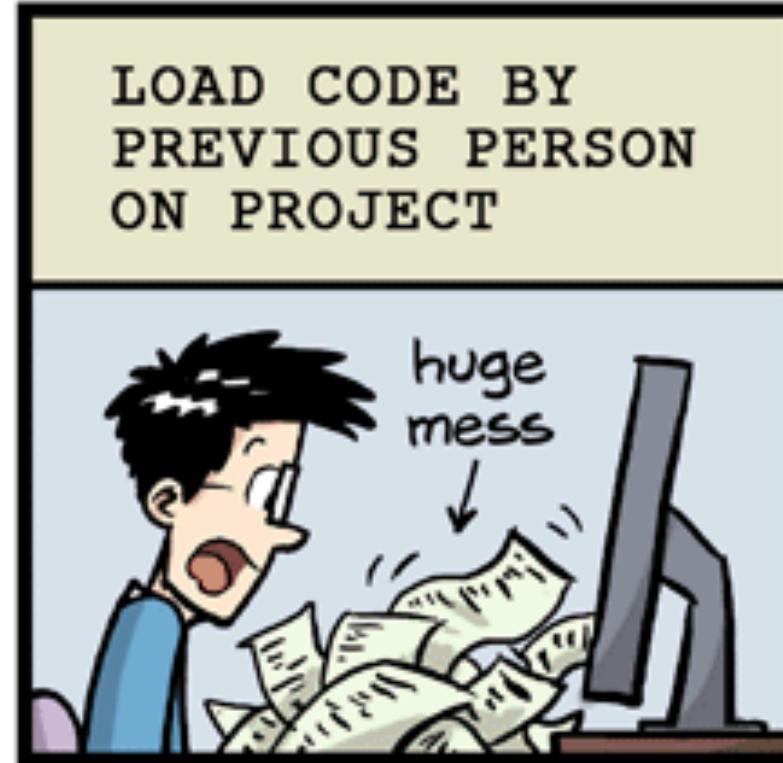


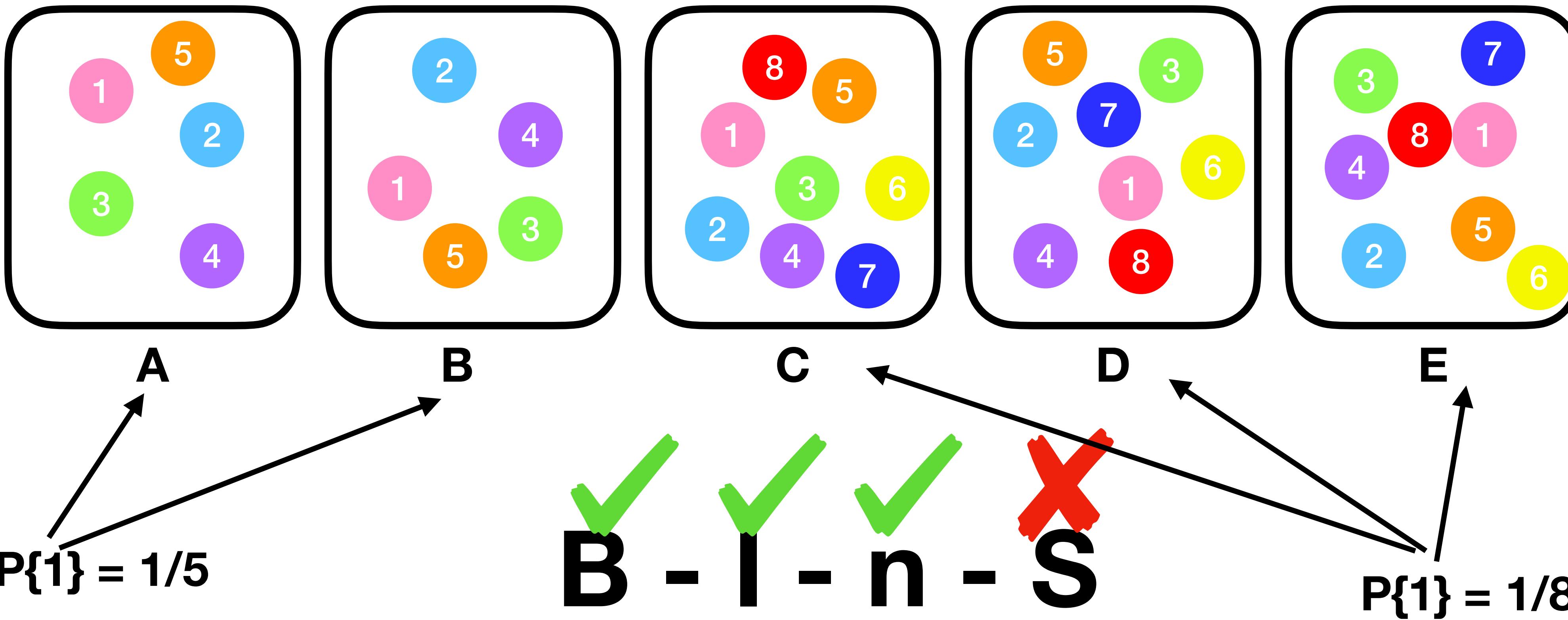
PROGRAMMING FOR NON-PROGRAMMERS



Practicum 01

10.05.2021

You have 5 urns: A, B, C, D, E. Urns A and B have balls numbered 1 through 5; urns C, D, E have ball numbers 1 through 8. There are five trials. In each trial, you draw a ball from an urn. In the first trial you draw from urn A, in the second trial you draw from urn B, etc. Let X be the number of times you draw a ball numbered 1.



Binary - Independent - n - Same p

Intro to the Tidyverse for clean data

- Collection of packages for data manipulation, exploration, and visualization that share a common syntax
- Intended to make data scientists more productive by guiding them through workflows
- Allows for connections between tools

“Happy families are all alike; every unhappy family is unhappy in its own way.” — Leo Tolstoy

“Tidy datasets are all alike, but every messy dataset is messy in its own way.” — Hadley Wickham

Intro to the Tidyverse for clean data

The diagram illustrates a dataset table with several annotations:

- color:** An arrow points to the header row, which is colored orange.
- merged cells:** An arrow points to the first three columns under "Replicate1", which are merged into a single header.
- multiple column titles:** An arrow points to the second column under "Replicate2", which contains multiple column titles separated by slashes.

| | Replicate1 | | | Replicate2 | | | Replicate3 | | | Replicate4 | | |
|----------|------------|------|-----------------|------------|------|------------------|------------|------|------------------|------------|------|-------------------|
| PLATE1 | day0 | day2 | day2/day0 | day0 | day2 | day2/day0 | day0 | day2 | day2/day0 | day0 | day2 | day2/day0 |
| BRC20067 | 229 | 2 | 0.8733624454148 | 138 | 0 | 0 | 234 | 0 | 0 | 136 | 0 | 0 |
| DL238 | 285 | 183 | 64.210526315789 | 334 | 161 | 48.2035928143713 | 179 | 71 | 39.6648044692737 | 224 | 110 | 49.1071428571429 |
| A3 | 166 | 0 | 0 | 104 | 0 | 0 | 231 | 0 | 0 | 251 | 0 | 0 |
| A6_R1 | 57 | 0 | 0 | 62 | 0 | 0 | 60 | 0 | 0 | 75 | 0 | 0 |
| A6_R2 | 150 | 0 | 0 | 256 | 6 | 2.34375 | 265 | 4 | 1.50943396226415 | 236 | 0 | 0 |
| B2 | 187 | 2 | 1.0695187165775 | 165 | 1 | 0.60606060606060 | 183 | 0 | 0 | 171 | 4 | 2.33918128654971 |
| B3 | 179 | 12 | 6.7039106145251 | 202 | 26 | 12.8712871287129 | 251 | 22 | 8.76494023904382 | 243 | 26 | 10.6995884773663 |
| B6 | 128 | 15 | 11.71875 | 113 | 9 | 7.9646017699115 | 220 | 10 | 4.54545454545455 | 214 | 14 | 6.54205607476636 |
| B8 | 268 | 0 | 0 | 155 | 0 | 0 | 164 | 1 | 0.60975609756097 | 175 | 0 | 0 |
| B9 | 93 | 2 | 2.1505376344086 | 118 | 2 | 1.69491525423729 | 132 | 0 | 0 | 111 | 2 | 1.8018018018018 |
| B10 | 188 | 2 | 1.0638297872340 | 379 | 2 | 0.52770448548812 | 225 | 0 | 0 | | | |
| C6 | 169 | 0 | 0 | 129 | 1 | 0.77519379844961 | 130 | 0 | 0 | 115 | 1 | 0.869565217391304 |

Q: Clean or messy?

Intro to the Tidyverse for clean data

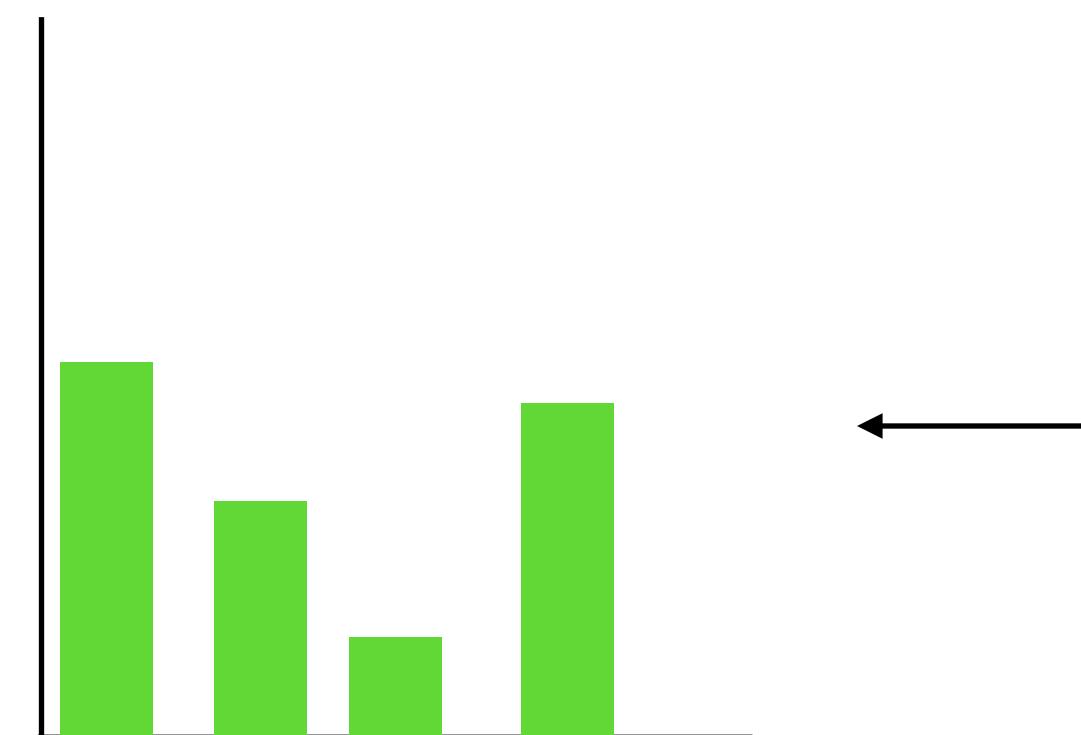
| strain | survival | rep | day |
|----------|------------------|------------|-----|
| BRC20067 | 0.8733624451 | 4A | 1 |
| DL238 | 64.210526315 | 8A | 1 |
| A3 | | 0 A | 1 |
| A6_R1 | | 0 A | 1 |
| A6_R2 | | 0 A | 1 |
| B2 | 1.0695187165 | 7A | 1 |
| B3 | 6.7039106145 | 25A | 1 |
| B6 | | 11.71875 A | 1 |
| B8 | | 0 A | 1 |
| B9 | 2.150537634408 | A | 1 |
| B10 | 1.063829787234 | A | 1 |
| C6 | | 0 A | 1 |
| BRC20067 | | 0 B | 1 |
| DL238 | 48.20359281437 | B | 1 |
| A3 | | 0 B | 1 |
| A6_R1 | | 0 B | 1 |
| A6_R2 | 2.34375 | B | 1 |
| B2 | 0.60606060606060 | B | 1 |
| B3 | 12.871287128711 | B | 1 |
| B6 | 7.964601769971 | B | 1 |
| B8 | | 0 B | 1 |

Rules for tidy data

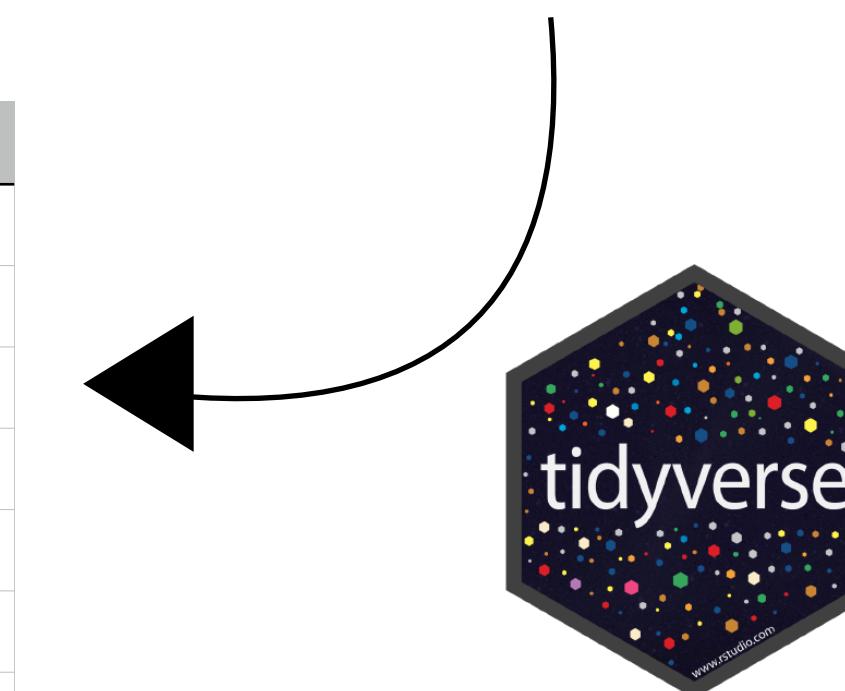
- Each **variable** must have its own **column**
- Each **observation** must have its own **row**
- Each **value** must have its own **cell**

Intro to the Tidyverse for clean data

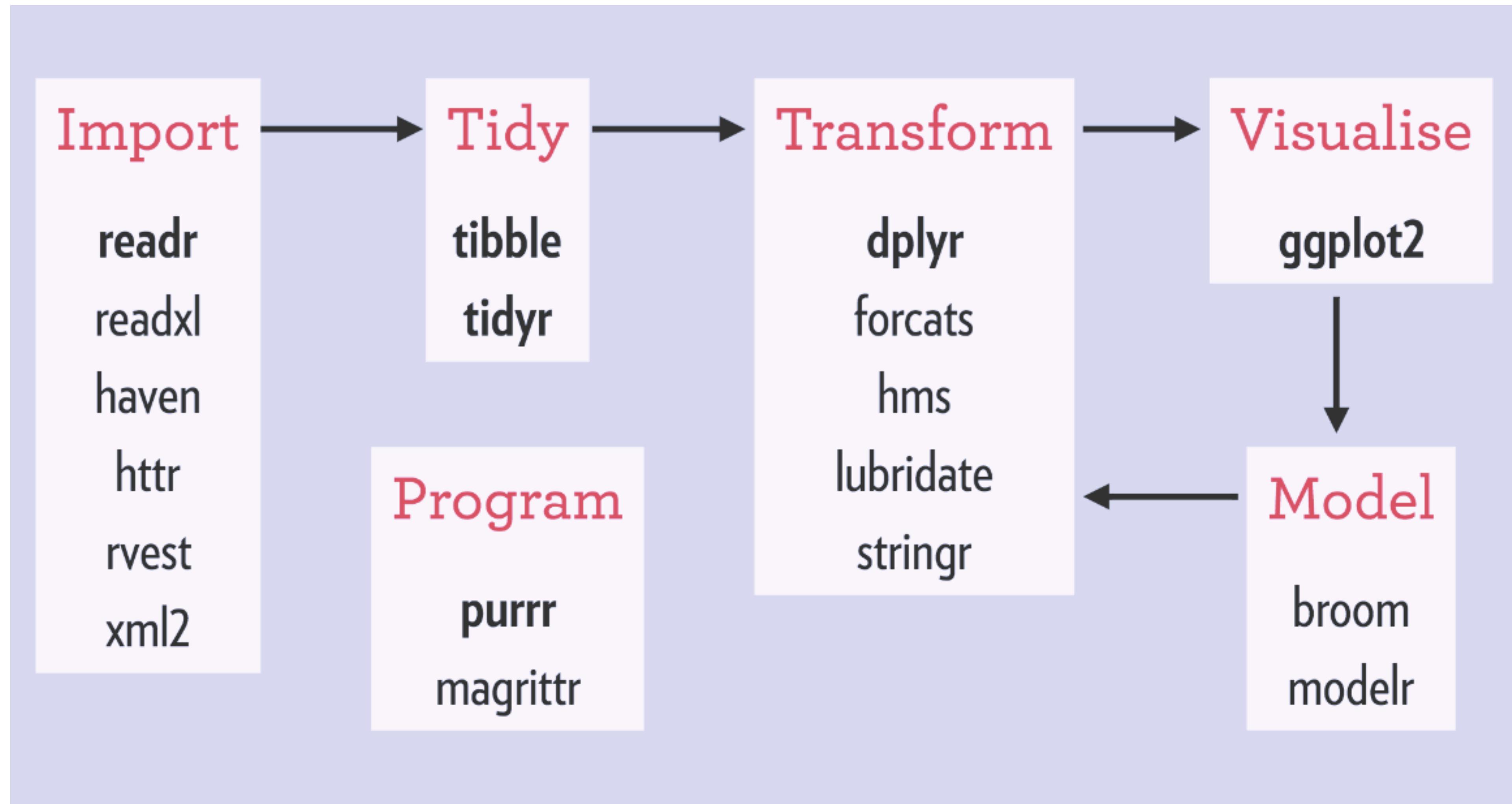
| | Replicate1 | | | Replicate2 | | | Replicate3 | | | Replicate4 | | |
|----------|------------|------|-----------------|------------|------|------------------|------------|------|------------------|------------|------|-------------------|
| | day0 | day2 | day2/day0 | day0 | day2 | day2/day0 | day0 | day2 | day2/day0 | day0 | day2 | day2/day0 |
| PLATE1 | 229 | 2 | 0.8733624454148 | 138 | 0 | 0 | 234 | 0 | 0 | 136 | 0 | 0 |
| BRC20067 | 285 | 183 | 64.210526315789 | 334 | 161 | 48.2035928143713 | 179 | 71 | 39.6648044692737 | 224 | 110 | 49.1071428571429 |
| A3 | 166 | 0 | 0 | 104 | 0 | 0 | 231 | 0 | 0 | 251 | 0 | 0 |
| A6_R1 | 57 | 0 | 0 | 62 | 0 | 0 | 60 | 0 | 0 | 75 | 0 | 0 |
| A6_R2 | 150 | 0 | 0 | 256 | 6 | 2.34375 | 265 | 4 | 1.50943396226415 | 236 | 0 | 0 |
| B2 | 187 | 2 | 1.0695187165771 | 165 | 1 | 0.60606060606060 | 183 | 0 | 0 | 171 | 4 | 2.33918128654971 |
| B3 | 179 | 12 | 6.7039106145251 | 202 | 26 | 12.8712871287129 | 251 | 22 | 8.76494023904382 | 243 | 26 | 10.6995884773663 |
| B6 | 128 | 15 | 11.71875 | 113 | 9 | 7.9646017699115 | 220 | 10 | 4.54545454545455 | 214 | 14 | 6.54205607476636 |
| B8 | 268 | 0 | 0 | 155 | 0 | 0 | 164 | 1 | 0.60975609756097 | 175 | 0 | 0 |
| B9 | 93 | 2 | 2.1505376344086 | 118 | 2 | 1.69491525423729 | 132 | 0 | 0 | 111 | 2 | 1.8018018018018 |
| B10 | 188 | 2 | 1.0638297872340 | 379 | 2 | 0.52770448548812 | 225 | 0 | 0 | | | |
| C6 | 169 | 0 | 0 | 129 | 1 | 0.77519379844961 | 130 | 0 | 0 | 115 | 1 | 0.869565217391304 |



| strain | survival | rep | day |
|----------|-----------------|-----|-----|
| BRC20067 | 0.873362445414A | | 1 |
| DL238 | 64.21052631578A | | 1 |
| A3 | 0 A | | 1 |
| A6_R1 | 0 A | | 1 |
| A6_R2 | 0 A | | 1 |
| B2 | 1.069518716577A | | 1 |
| B3 | 6.703910614525A | | 1 |
| B6 | 11.71875 A | | 1 |
| B8 | 0 A | | 1 |
| B9 | 2.150537634408A | | 1 |
| B10 | 1.063829787234A | | 1 |
| C6 | 0 A | | 1 |

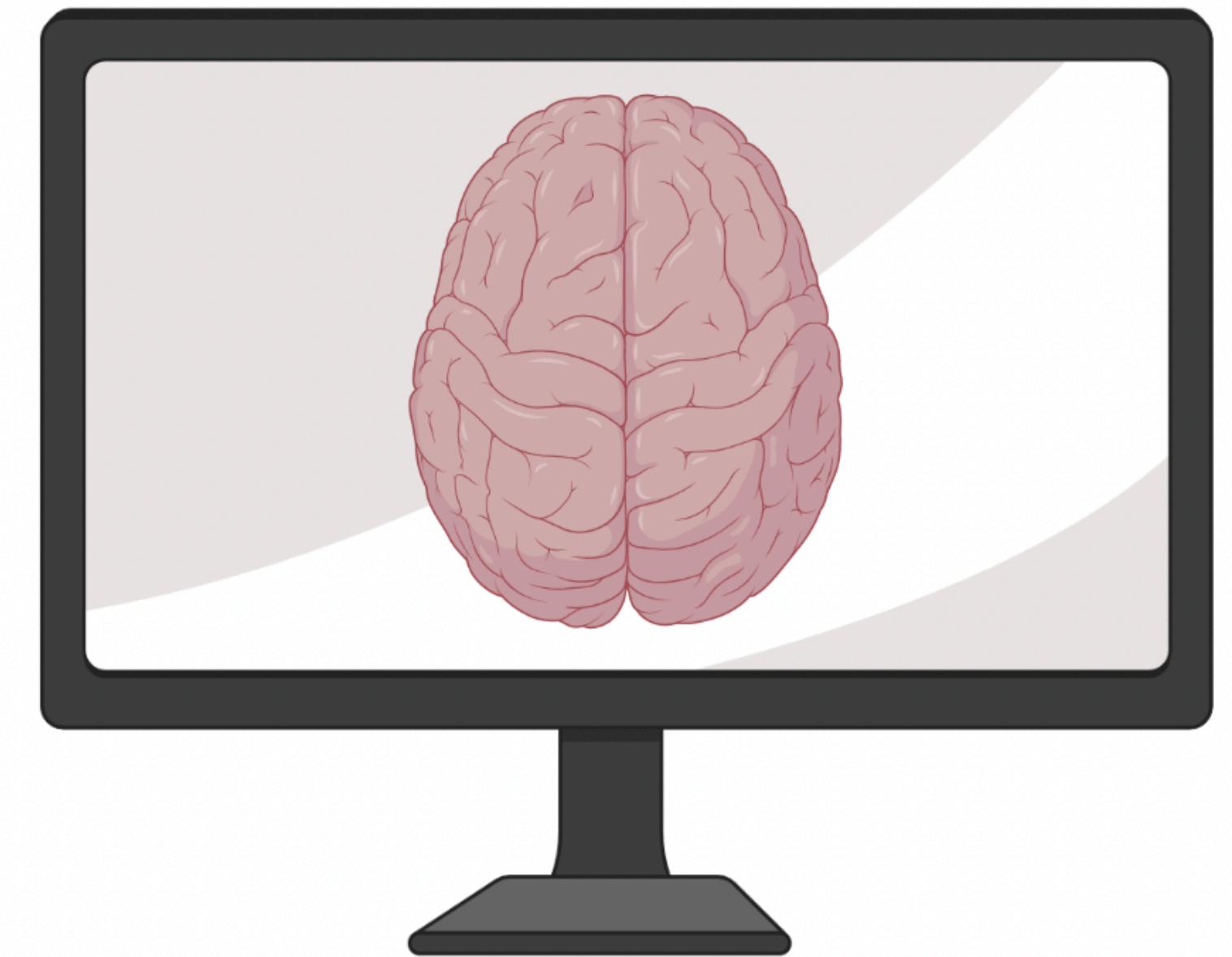


Intro to the Tidyverse for clean data



Coding: thinking like a computer

1. Figure out what you want to do
2. Describe those tasks words
3. Describe those tasks in code



R Studio

Script files

- Saves your script
- Allows code & comments
- Can have multiple files open at a time

Console/Command line

- Can use as calculator
- Does not save code
- This is where your output is displayed

The screenshot shows the R Studio interface with four main panes:

- Script Editor:** Displays two R script files: `initial_functions.R` and `plot_functions.R`. The `initial_functions.R` file contains code for a `block_summary` function.
- Environment:** Shows the global environment with variables `m`, `n`, `r`, and `t` defined.
- Console:** Displays command-line interactions, including the execution of the `block_summary` function and other arithmetic calculations.
- Help:** Provides documentation for the `mean` function, including its description, usage, and arguments.

Workspace environment

- Holds your objects
- Can review history

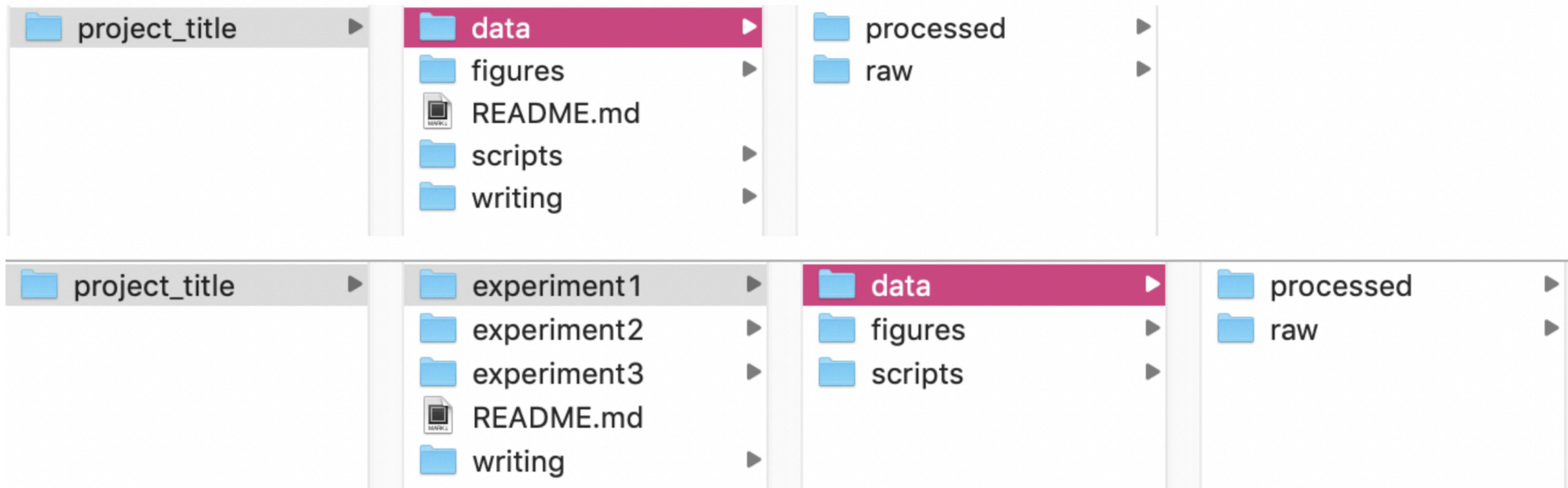
Misc - Displays:

- files in working directory
- plots when produced
- help files/search

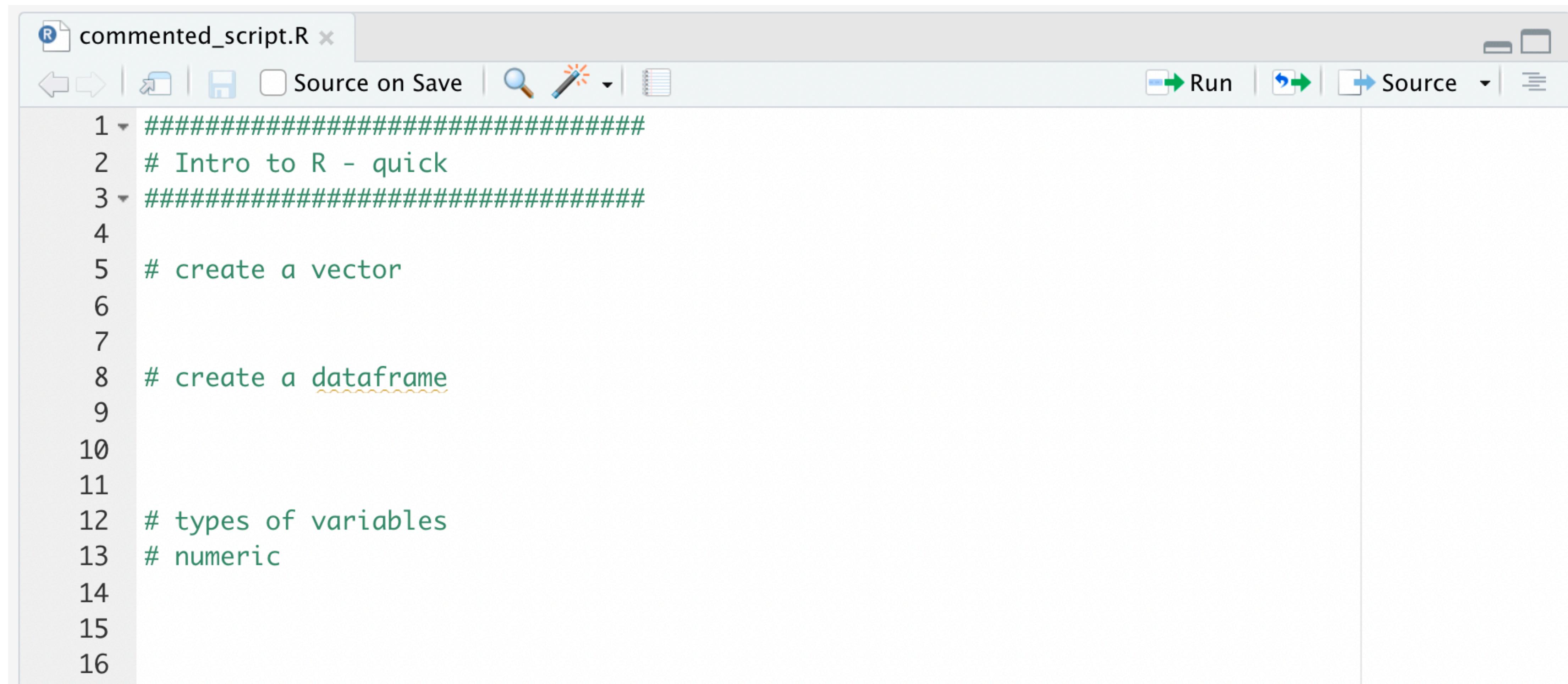
Data analysis 101

- Do not use spaces in names of files/folders. Try also to not use spaces in column names (although sometimes necessary)
 - Computers often have a hard time reading spaces and code used to ignore spaces can differ between programs. Instead, try using “-“ or “_” or “.” Or capitalization (i.e. fileName.txt)
- NEVER replace raw data!!!!!!!
 - Save data in rawest form, write a script to analyze it, and save the processed data (if you want)
- Comment your code! I promise, you will forget what you were doing one day (maybe sooner than you hope...)
- For R specifically: please please please ALWAYS use namespaces (i.e. **dplyr::filter()**)

Data analysis 101



Follow along coding



The screenshot shows the RStudio interface with a script editor window titled "commented_script.R x". The window contains the following R code:

```
1 #####  
2 # Intro to R - quick  
3 #####  
4  
5 # create a vector  
6  
7  
8 # create a dataframe  
9  
10  
11  
12 # types of variables  
13 # numeric  
14  
15  
16
```

The code is syntax-highlighted, with comments in green and variable names in blue. The RStudio toolbar at the top includes icons for file operations, search, and run.

Basic R things to know before we start

Variable

Stores a value that might change

Vector

Collection of elements

```
c(3, 5, 10)
```

Data frame

Collection of vectors (tabular format)

```
data.frame(color = c("red", "white"),  
           count = c(45, 80))
```

| color | count |
|-------|-------|
| red | 45 |
| white | 80 |

Lists

A special type of vector

{swirl}

Basic R things to know before we start

Character

```
x <- "gene-A"
```

Integer/numeric

```
x <- 5
```

Logical

```
x <- TRUE
```

**R logic expressions
(returns TRUE or FALSE)**

```
x <- 5
```

```
x == 5
```

```
x != 4
```

```
x > 2
```

```
x <= 10
```

```
x > 4 & x < 6
```

```
x > 6 | x < 7
```

```
is.na(x)
```

```
x <- c(4, 6, 10)
```

```
x == 6
```

```
x > 5
```

```
10 %in% x
```

```
x[1]
```

```
x[2:3]
```

```
x[-1]
```

```
x[4] <- 15
```

```
which(x > 5)
```



Basic R things to know before we start

```
x <- c(2, 5, 10, 12)
```

Get the first element

```
x[1]
```

Change the first element

```
x[1] <- 9
```

Remove the second element

```
x[-2]
```

Keep the 2nd and 3rd elements

```
x[2:3]
```

Get the elements > 2

```
x[x > 2]
```



Basic R things to know before we start

```
ifelse(condition, what to do if TRUE, what to do if FALSE)
```

```
x <- c(2, 5, 10, 12)
```

```
# Write an 'ifelse' statement to print “less than 9” if the element is less  
than 9 or “greater than 9” if the element is greater than 9
```

```
ifelse(x < 9, "less than 9", "greater than 9")
```

Basic R things to know before we start

```
if(condition) {  
    what to do if TRUE  
} else {  
    what to do if FALSE  
}
```

Basic R things to know before we start

```
if(condition) {  
    what to do if TRUE  
} else {  
    what to do if FALSE  
}
```

What is the difference between
ifelse and if/else?
ifelse is element-wise!

x <- 10

Write an 'if' statement to print "less than 9" if the variable is less than 9 or "greater than 9" if the variable is greater than 9

```
if(x < 9) {  
    print("less than 9")  
} else {  
    print("greater than 9")  
}
```

Basic R things to know before we start

```
for(element in start:end) {  
  function(s)  
}  
  
x <- c(2, 5, 10, 12)
```

Write an ‘for loop’ to add 5 to each element of vector x

```
for(i in 1:length(x)) {  
  print(x[i] + 5)  
}
```

What is a package?

- Contains functions useful for certain tasks
- Can download and load them easily
 - `install.packages("dplyr")`
 - `library(dplyr)`
- Can specifically direct R to use a function within a package using a namespace (e.g. `dplyr::filter()`)
- **Try downloading and loading dplyr()**

Starwars dataset



```
library(dplyr)  
data(starwars)  
View(starwars)
```

variables

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships |
|----|-----------------------|--------|--------|---------------|------------------|-----------|------------|---------------|------------|----------------|--|---|---|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") |
| 2 | C-3PO | 167 | 75.0 | NA | gold | yellow | 112.0 | NA | Tatooine | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 3 | R2-D2 | 96 | 32.0 | NA | white, blue | red | 33.0 | NA | Naboo | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | character(0) | TIE Advanced x1 |
| 5 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | Imperial Speeder Bike | character(0) |
| 6 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 7 | Beru Whitesun lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 8 | R5-D4 | 97 | 32.0 | NA | white, red | red | NA | NA | Tatooine | Droid | A New Hope | character(0) | character(0) |
| 9 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine | Human | A New Hope | character(0) | X-wing |
| 10 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male | Stewjon | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... |
| 11 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... |
| 12 | Wilhuff Tarkin | 180 | NA | auburn, grey | fair | blue | 64.0 | male | Eriadu | Human | c("Revenge of the Sith", "A New Hope") | character(0) | character(0) |
| 13 | Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male | Kashyyyk | Wookiee | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | AT-ST | c("Millennium Falcon", "Imperial shuttle") |
| 14 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | character(0) | c("Millennium Falcon", "Imperial shuttle") |
| 15 | Greedo | 173 | 74.0 | NA | green | black | 44.0 | male | Rodia | Rodian | A New Hope | character(0) | character(0) |
| 16 | Jabba Desilijic Tiure | 175 | 1358.0 | NA | green-tan, brown | orange | 600.0 | hermaphrodite | Nal Hutta | Hutt | c("The Phantom Menace", "Return of the Jedi", "A New ... | character(0) | character(0) |
| 17 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | Snowspeeder | X-wing |
| 18 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | NA | male | Bestine IV | Human | A New Hope | character(0) | X-wing |
| 19 | Yoda | 66 | 17.0 | white | green | brown | 896.0 | male | NA | Yoda's species | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 20 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male | Naboo | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 21 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male | Kamino | Human | c("Attack of the Clones", "Return of the Jedi", "The Em... | character(0) | Slave 1 |
| 22 | IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none | NA | Droid | The Empire Strikes Back | character(0) | character(0) |
| 23 | Bossk | 190 | 113.0 | none | green | red | 53.0 | male | Trandosha | Trandoshan | The Empire Strikes Back | character(0) | character(0) |
| 24 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male | Socorro | Human | c("Return of the Jedi", "The Empire Strikes Back") | character(0) | Millennium Falcon |

observations

values

Starwars dataset



```
library(dplyr)  
data(starwars)  
View(starwars)
```

variables

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships |
|----|-----------------------|--------|--------|---------------|------------------|-----------|------------|---------------|------------|----------------|---|---|---|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Empire...") | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") |
| 2 | C-3PO | 167 | 75.0 | NA | gold | yellow | 112.0 | NA | Tatooine | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev...") | character(0) | character(0) |
| 3 | R2-D2 | 96 | 32.0 | NA | white, blue | red | 33.0 | NA | Naboo | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev...") | character(0) | character(0) |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Empire...") | character(0) | TIE Advanced x1 |
| 5 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Empire...") | Imperial Speeder Bike | character(0) |
| 6 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New...") | character(0) | character(0) |
| 7 | Beru Whitesun lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New...") | character(0) | character(0) |
| 8 | R5-D4 | 97 | 32.0 | NA | white, red | red | NA | NA | Tatooine | Droid | A New Hope | character(0) | character(0) |
| 9 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine | Human | A New Hope | character(0) | X-wing |
| 10 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male | Stewjon | Human | c("Attack of the Clones", "The Phantom Menace", "Rev...") | Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo...") |
| 11 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace", "Rev...") | c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo...") |
| 12 | Wilhuff Tarkin | 180 | NA | auburn, grey | fair | blue | 64.0 | male | Eriadu | Human | c("Revenge of the Sith", "A New Hope") | character(0) | character(0) |
| 13 | Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male | Kashyyyk | Wookiee | c("Revenge of the Sith", "Return of the Jedi", "The Empire...") | AT-ST | c("Millennium Falcon", "Imperial shuttle") |
| 14 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N...") | character(0) | c("Millennium Falcon", "Imperial shuttle") |
| 15 | Greedo | 173 | 74.0 | NA | green | black | 44.0 | male | Rodia | Rodian | A New Hope | character(0) | character(0) |
| 16 | Jabba Desilijic Tiure | 175 | 1358.0 | NA | green-tan, brown | orange | 600.0 | hermaphrodite | Nal Hutta | Hutt | c("The Phantom Menace", "Return of the Jedi", "A New ...") | character(0) | character(0) |
| 17 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N...") | Snowspeeder | X-wing |
| 18 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | NA | male | Bestine IV | Human | A New Hope | character(0) | X-wing |
| 19 | Yoda | 66 | 17.0 | white | green | brown | 896.0 | male | NA | Yoda's species | c("Attack of the Clones", "The Phantom Menace", "Rev...") | character(0) | character(0) |
| 20 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male | Naboo | Human | c("Attack of the Clones", "The Phantom Menace", "Rev...") | character(0) | character(0) |
| 21 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male | Kamino | Human | c("Attack of the Clones", "Return of the Jedi", "The Em...") | character(0) | Slave 1 |
| 22 | IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none | NA | Droid | The Empire Strikes Back | character(0) | character(0) |
| 23 | Bossk | 190 | 113.0 | none | green | red | 53.0 | male | Trandosha | Trandoshan | The Empire Strikes Back | character(0) | character(0) |
| 24 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male | Socorro | Human | c("Return of the Jedi", "The Empire Strikes Back") | character(0) | Millennium Falcon |

observations

values

MESSY DATA

Introduction: dplyr

- Collection of functions as **verbs** to easily describe what you want to do with your data

Functions:

- `filter()` to keep rows based on values
- `select()` to keep columns based on names
- `mutate()` to add new (or change existing) columns
- `group_by()` to group rows by columns
- `summarize()` to condense multiple columns
- `arrange()` to reorder the rows
- `rename()` to give columns new names
- `xxx_join()` to combine data frames
- `bind_rows()` or `bind_cols()` to combine dataframes



dplyr::filter()

```
dplyr::filter(dataframe, condition(s))
```

starwars

only humans

1. Figure out what you want to do

2. Describe your goal in words

Filter the starwars dataframe to only include humans

3. Describe your goal in code

```
dplyr::filter(starwars, ?)
```

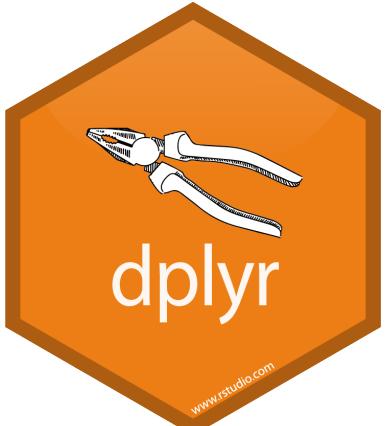


dplyr::filter() to keep rows based on values

dplyr::filter()

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships |
|----|-----------------------|--------|--------|---------------|------------------|-----------|------------|---------------|------------|----------------|--|---|---|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") |
| 2 | C-3PO | 167 | 75.0 | NA | gold | yellow | 112.0 | NA | Tatooine | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 3 | R2-D2 | 96 | 32.0 | NA | white, blue | red | 33.0 | NA | Naboo | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | character(0) | TIE Advanced x1 |
| 5 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | Imperial Speeder Bike | character(0) |
| 6 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 7 | Beru Whitesun Lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 8 | R5-D4 | 97 | 32.0 | NA | white, red | red | NA | NA | Tatooine | Droid | A New Hope | character(0) | character(0) |
| 9 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine | Human | A New Hope | character(0) | X-wing |
| 10 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male | Stewjon | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... |
| 11 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... |
| 12 | Wilhuff Tarkin | 180 | NA | auburn, grey | fair | blue | 64.0 | male | Eriadu | Human | c("Revenge of the Sith", "A New Hope") | character(0) | character(0) |
| 13 | Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male | Kashyyyk | Wookiee | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | AT-ST | c("Millennium Falcon", "Imperial shuttle") |
| 14 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | character(0) | c("Millennium Falcon", "Imperial shuttle") |
| 15 | Greedo | 173 | 74.0 | NA | green | black | 44.0 | male | Rodia | Rodian | A New Hope | character(0) | character(0) |
| 16 | Jabba Desilijic Tiure | 175 | 1358.0 | NA | green-tan, brown | orange | 600.0 | hermaphrodite | Nal Hutta | Hutt | c("The Phantom Menace", "Return of the Jedi", "A New ... | character(0) | character(0) |
| 17 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | Snowspeeder | X-wing |
| 18 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | NA | male | Bestine IV | Human | A New Hope | character(0) | X-wing |
| 19 | Yoda | 66 | 17.0 | white | green | brown | 896.0 | male | NA | Yoda's species | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 20 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male | Naboo | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 21 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male | Kamino | Human | c("Attack of the Clones", "Return of the Jedi", "The Em... | character(0) | Slave 1 |
| 22 | IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none | NA | Droid | The Empire Strikes Back | character(0) | character(0) |
| 23 | Bossk | 190 | 113.0 | none | green | red | 53.0 | male | Trandosha | Trandoshan | The Empire Strikes Back | character(0) | character(0) |
| 24 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male | Socorro | Human | c("Return of the Jedi", "The Empire Strikes Back") | character(0) | Millennium Falcon |

species == "Human"



dplyr::filter() to keep rows based on values

dplyr::filter()

```
dplyr::filter(dataframe, condition(s))
```

starwars

only humans

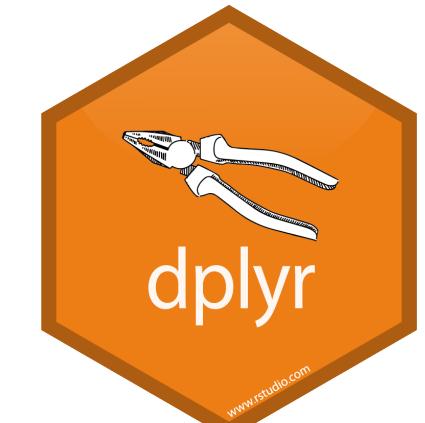
1. Figure out what you want to do

2. Describe your goal in words

Filter the starwars dataframe to only include humans

3. Describe your goal in code

```
dplyr::filter(starwars, species == "Human")
```



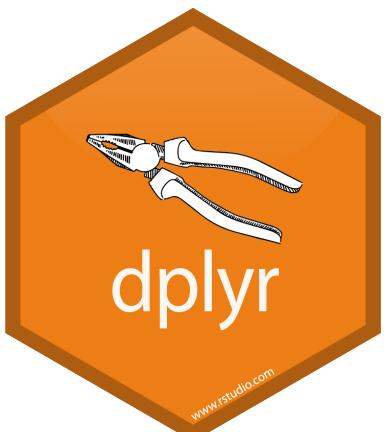
dplyr::filter() to keep rows based on values

dplyr::filter()

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships |
|----|--------------------|--------|-------|---------------|------------|-----------|------------|--------|------------|---------|--|---|---|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") |
| 2 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | character(0) | TIE Advanced x1 |
| 3 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | Imperial Speeder Bike | character(0) |
| 4 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 5 | Beru Whitesun lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 6 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine | Human | A New Hope | character(0) | X-wing |
| 7 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male | Stewjon | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... |
| 8 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... |
| 9 | Wilhuff Tarkin | 180 | NA | auburn, grey | fair | blue | 64.0 | male | Eriadu | Human | c("Revenge of the Sith", "A New Hope") | character(0) | character(0) |
| 10 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | character(0) | c("Millennium Falcon", "Imperial shuttle") |
| 11 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | Snowspeeder | X-wing |
| 12 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | NA | male | Bestine IV | Human | A New Hope | character(0) | X-wing |
| 13 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male | Naboo | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 14 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male | Kamino | Human | c("Attack of the Clones", "Return of the Jedi", "The Em... | character(0) | Slave 1 |
| 15 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male | Socorro | Human | c("Return of the Jedi", "The Empire Strikes Back") | character(0) | Millennium Falcon |
| 16 | Lobot | 175 | 79.0 | none | light | blue | 37.0 | male | Bespin | Human | The Empire Strikes Back | character(0) | character(0) |
| 17 | Mon Mothma | 150 | NA | auburn | fair | blue | 48.0 | female | Chandrila | Human | Return of the Jedi | character(0) | character(0) |
| 18 | Arvel Crynyd | NA | NA | brown | fair | brown | NA | male | NA | Human | Return of the Jedi | character(0) | A-wing |
| 19 | Qui-Gon Jinn | 193 | 89.0 | brown | fair | blue | 92.0 | male | NA | Human | The Phantom Menace | Tribubble bongo | character(0) |
| 20 | Finis Valorum | 170 | NA | blond | fair | blue | 91.0 | male | Coruscant | Human | The Phantom Menace | character(0) | character(0) |
| 21 | Shmi Skywalker | 163 | NA | black | fair | brown | 72.0 | female | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace") | character(0) | character(0) |

```
dplyr::filter(starwars, species == "Human")
```

dplyr::filter() to keep rows based on values



dplyr::filter()

```
dplyr::filter(dataframe, condition(s))
```



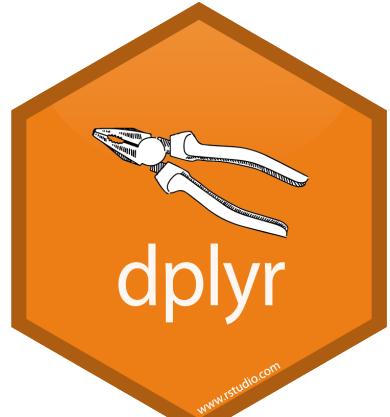
You can also filter based on multiple conditions

Filter the starwars dataframe to include only male humans.

```
dplyr::filter(starwars, species == "Human", gender == "male")
```

Filter the starwars dataframe to include humans and droids.

```
dplyr::filter(starwars, species %in% c("Human", "Droid"))
```



dplyr::filter() to keep rows based on values

dplyr::filter()

```
dplyr::filter(dataframe, condition(s))
```

Filter starwars to include only people with height less than 100.

```
dplyr::filter(starwars, height < 100)
```

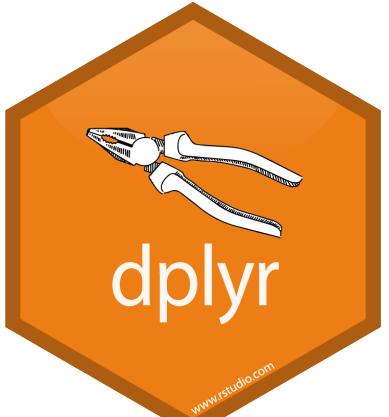
Keep all non-humans

```
dplyr::filter(starwars, species != "Human")
```

Remove characters with no known species

```
dplyr::filter(starwars, !is.na(species))
```

dplyr::filter() to keep rows based on values



Piping in Tidyverse

Filter the starwars dataframe to include only male humans.

Without pipes

```
dplyr::filter(starwars, species == "Human", gender == "male")
```

With pipes

```
starwars %>%  
  dplyr::filter(species == "Human") %>%  
  dplyr::filter(gender == "male")
```

1. start with the starwars dataframe

2. filter to keep only humans

3. filter to keep only males

**Can be useful to combine several tidyverse
“verbs” to one block of code**

takes output of left side and makes it input of right side



dplyr::select()

```
dplyr::select(dataframe, columns_to_keep)
```

starwars name, species, films

Select only name, species, and films variables from the starwars dataframe

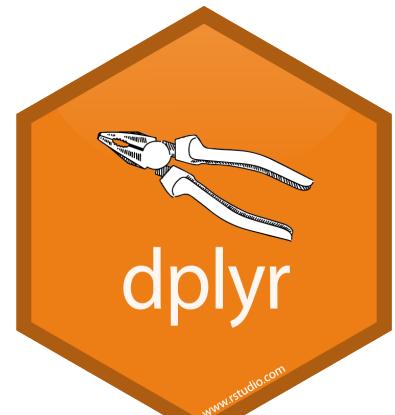
```
dplyr::select(starwars, name, species, films)
```



Deselect a column with -column_name

Select all columns except the gender

```
dplyr::select(starwars, -gender)
```



dplyr::select() to keep columns based on names

dplyr::select()

```
dplyr::select(dataframe, columns_to_keep)
```



Select a range of columns with column1:column2

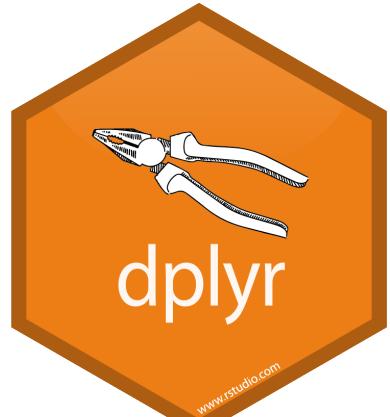
Select columns name, height, mass, hair color, and skin color

```
dplyr::select(starwars, name:skin_color)
```

Select all columns except hair color, skin color, and eye color

```
dplyr::select(starwars, -hair_color, -skin_color, -eye_color)
```

```
dplyr::select(starwars, -(hair_color:eye_color))
```



dplyr::select() to keep columns based on names

dplyr::select()

```
dplyr::select(dataframe, columns_to_keep)
```



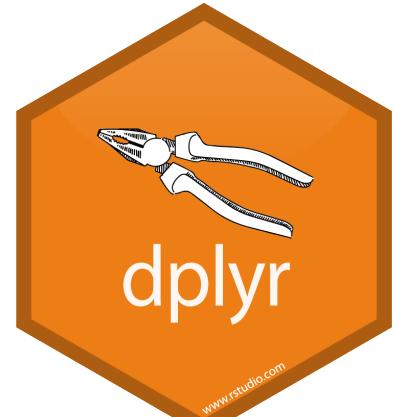
Reorder columns with select too!

Select columns name, height, mass, hair color, and skin color

```
dplyr::select(starwars, name:skin_color)
```

Select columns name, mass, skin color, hair color, and height (in order)

```
dplyr::select(starwars, name, mass, skin_color, hair_color, height)
```



dplyr::select() to keep columns based on names

dplyr::select()

```
dplyr::select(dataframe, new_name = old_name)
```



You can also use `select()` to rename columns

Rename the “name” column to “character” and the “mass” column to “weight” using `select()`

```
renamed <- starwars %>%  
  dplyr::select(character = name, weight = mass)
```



dplyr::select() to keep columns based on names

| character | weight |
|----------------|--------|
| Luke Skywalker | 77.0 |
| C-3PO | 75.0 |
| R2-D2 | 32.0 |
| Darth Vader | 136.0 |
| Leia Organa | 49.0 |
| Owen Lars | 120.0 |

Combine filter and select

```
dplyr::filter(dataframe, condition(s))
```

```
dplyr::select(dataframe, columns_to_keep)
```

Challenge OYO:

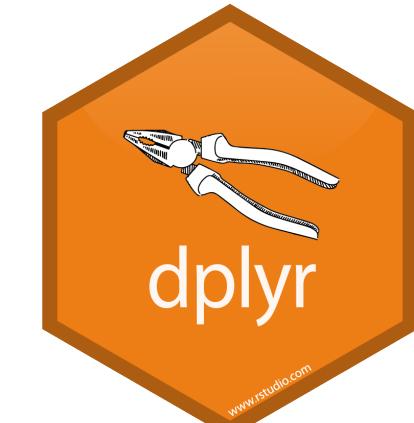
[Keep: height greater than 100 &

Keep: humans &

Remove: brown hair color &

Remove: vehicles &

Keep: name, homeworld, height, species, hair color]



dplyr::filter() to keep rows based on values

dplyr::select() to keep columns based on names



Combine filter and select

| | name | homeworld | height | species | hair_color |
|----|-------------------|-----------|--------|---------|---------------|
| 1 | Luke Skywalker | Tatooine | 172 | Human | blond |
| 2 | Darth Vader | Tatooine | 202 | Human | none |
| 3 | Owen Lars | Tatooine | 178 | Human | brown, grey |
| 4 | Biggs Darklighter | Tatooine | 183 | Human | black |
| 5 | Obi-Wan Kenobi | Stewjon | 182 | Human | auburn, white |
| 6 | Anakin Skywalker | Tatooine | 188 | Human | blond |
| 7 | Wilhuff Tarkin | Eriadu | 180 | Human | auburn, grey |
| 8 | Palpatine | Naboo | 170 | Human | grey |
| 9 | Boba Fett | Kamino | 183 | Human | black |
| 10 | Lando Calrissian | Socorro | 177 | Human | black |
| 11 | Lobot | Bespin | 175 | Human | none |
| 12 | Mon Mothma | Chandrila | 150 | Human | auburn |
| 13 | Finis Valorum | Coruscant | 170 | Human | blond |
| 14 | Shmi Skywalker | Tatooine | 163 | Human | black |
| 15 | Mace Windu | Coruscant | 198 | Human | none |

```
starwars %>%
  dplyr::filter(height > 100) %>%
  dplyr::filter(species == "Human") %>%
  dplyr::filter(hair_color != "brown") %>%
  dplyr::select(-vehicles) %>%
  dplyr::select(name, homeworld, height, species, hair_color)
```

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

1. Figure out what you want to do

calculate BMI of all starwars characters

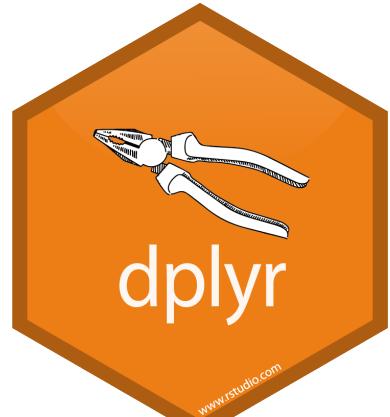
2. Describe your goal in words

cm

$$\text{BMI} = \text{weight} \text{ (kg)} / [\text{height} \text{ (m)}]^2$$

| | name | height | mass | hair_color | skin_color | eye_color |
|---|----------------|--------|-------|------------|-------------|-----------|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue |
| 2 | C-3PO | 167 | 75.0 | NA | gold | yellow |
| 3 | R2-D2 | 96 | 32.0 | NA | white, blue | red |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow |

?starwars



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

1. Figure out what you want to do

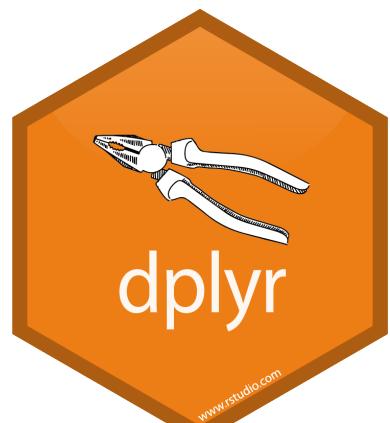
calculate BMI of all starwars characters

2. Describe your goal in words

BMI = **weight** (kg) / [**height** (m)]²

First: convert height in cm to height in meters

Second: calculate BMI as new column



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

starwars

height_m

height / 100

2. Describe your goal in words

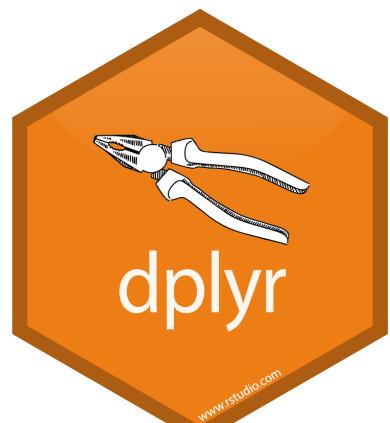
BMI = **weight** (kg) / [**height** (m)]²

First: convert height in cm to height in meters

Second: calculate BMI as new column

3. Describe your goal in code

```
new_starwars <- dplyr::mutate(starwars, height_m = height / 100)
```



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

new_starwars bmi mass / height_m^2

2. Describe your goal in words

BMI = **weight** (kg) / [**height** (m)]²

First: convert height in cm to height in meters

Second: calculate BMI as new column

3. Describe your goal in code

```
new_starwars <- starwars %>%  
  dplyr::mutate(height_m = height / 100) %>%  
dplyr::mutate(bmi = mass / height_m^2)
```



dplyr::mutate()

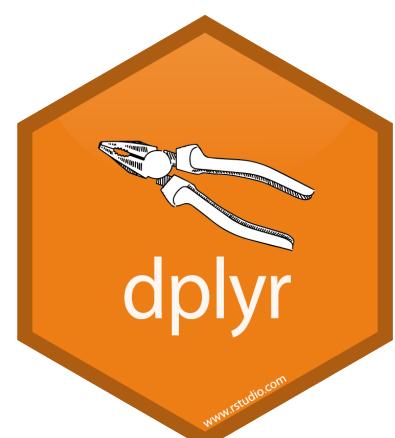
```
dplyr::mutate(dataframe, new_column = expression)
```

Select only name, height, mass, height_m, and bmi and view dataframe

```
new_starwars <- new_starwars %>%  
  dplyr::select(name, height, mass, height_m, bmi)
```

View(new_starwars)

| name | height | mass | height_m | bmi |
|--------------------|--------|-------|----------|----------|
| Luke Skywalker | 172 | 77.0 | 1.72 | 26.02758 |
| C-3PO | 167 | 75.0 | 1.67 | 26.89232 |
| R2-D2 | 96 | 32.0 | 0.96 | 34.72222 |
| Darth Vader | 202 | 136.0 | 2.02 | 33.33007 |
| Leia Organa | 150 | 49.0 | 1.50 | 21.77778 |
| Owen Lars | 178 | 120.0 | 1.78 | 37.87401 |
| Beru Whitesun Lars | 165 | 75.0 | 1.65 | 27.54821 |
| R5-D4 | 97 | 32.0 | 0.97 | 34.00999 |
| Biggs Darklighter | 183 | 84.0 | 1.83 | 25.08286 |



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

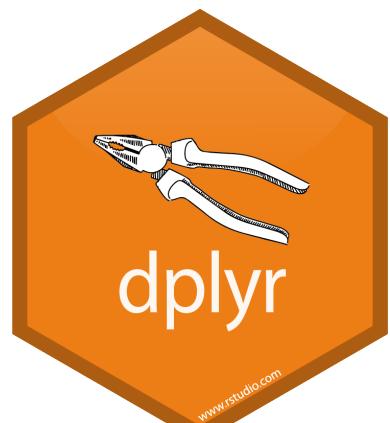


We can change existing columns if we use the same name

```
new_starwars <- starwars %>%  
  dplyr::mutate(height = height / 100) %>%  
  dplyr::mutate(bmi = mass / height^2)
```

```
View(new_starwars)
```

| name | height | mass | bmi |
|----------------|--------|-------|----------|
| Luke Skywalker | 1.72 | 77.0 | 26.02758 |
| C-3PO | 1.67 | 75.0 | 26.89232 |
| R2-D2 | 0.96 | 32.0 | 34.72222 |
| Darth Vader | 2.02 | 136.0 | 33.33007 |
| Leia Organa | 1.50 | 49.0 | 21.77778 |
| Owen Lars | 1.78 | 120.0 | 37.87401 |



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Make a new column to find the average height (in meters)

```
starwars %>%
```

```
  dplyr::mutate(height
```

```
    dplyr::mutate(avg_heig
```

```
?mean
```

| name | height | mass | avg_height |
|-----------------------|--------|-------|------------|
| Palpatine | 1.70 | 75.0 | NA |
| Boba Fett | 1.83 | 78.2 | NA |
| IG-88 | 2.00 | 140.0 | NA |
| Bossk | 1.90 | 113.0 | NA |
| Lando Calrissian | 1.77 | 79.0 | NA |
| Lobot | 1.75 | 79.0 | NA |
| Ackbar | 1.80 | 83.0 | NA |
| Mon Mothma | 1.50 | NA | NA |
| Arvel Crynyd | NA | NA | NA |
| Wicket Systri Warrick | 0.88 | 20.0 | NA |
| Nien Nunb | 1.60 | 68.0 | NA |
| Qui-Gon Jinn | 1.93 | 89.0 | NA |
| Nute Gunray | 1.91 | 90.0 | NA |
| Finis Valorum | 1.70 | NA | NA |



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Make a new column to find the average height (in meters)

```
starwars %>%
```

```
  dplyr::muta
```

```
  dplyr::muta
```

```
?mean
```

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

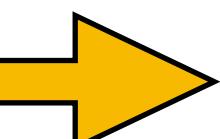
Arguments

x An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for **trim** = 0, only.

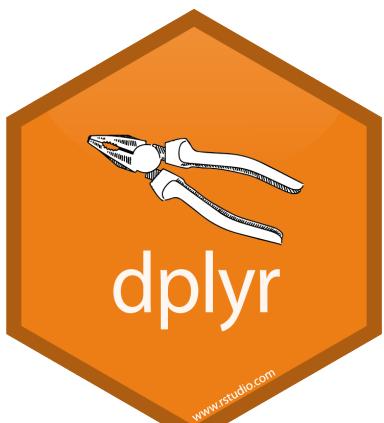
trim the fraction (0 to 0.5) of observations to be trimmed from each end of **x** before the mean is computed. Values of **trim** outside that range are taken as the nearest endpoint.

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

... further arguments passed to or from other methods.



dplyr::mutate() to add new (or change existing) columns



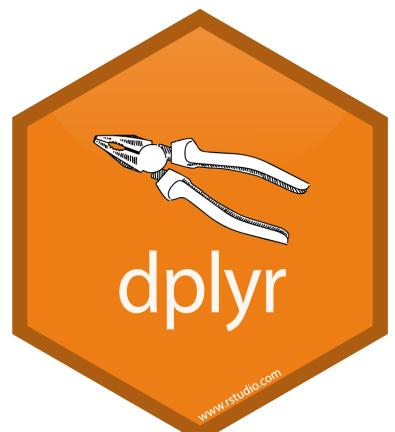
dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Make a new column to find the average height (in meters)

```
starwars %>%
  dplyr::mutate(height = height / 100) %>%
dplyr::mutate(avg_height = mean(height, na.rm = TRUE))
```

| name | height | mass | avg_height |
|--------------------|--------|-------|------------|
| Luke Skywalker | 172 | 77.0 | 1.74358 |
| C-3PO | 167 | 75.0 | 1.74358 |
| R2-D2 | 96 | 32.0 | 1.74358 |
| Darth Vader | 202 | 136.0 | 1.74358 |
| Leia Organa | 150 | 49.0 | 1.74358 |
| Owen Lars | 178 | 120.0 | 1.74358 |
| Beru Whitesun Lars | 165 | 75.0 | 1.74358 |
| R5-D4 | 97 | 32.0 | 1.74358 |



dplyr::mutate() to add new (or change existing) columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Standardize the heights of the starwars characters (height / avg_height)

```
starwars %>%  
  dplyr::mutate(height = height / 100) %>%  
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE)) %>%  
  dplyr::mutate(std_height = height / avg_height)
```

| name | height | mass | avg_height | std_height |
|--------------------|--------|-------|------------|------------|
| Luke Skywalker | 1.72 | 77.0 | 1.74358 | 0.9864760 |
| C-3PO | 1.67 | 75.0 | 1.74358 | 0.9577993 |
| R2-D2 | 0.96 | 32.0 | 1.74358 | 0.5505912 |
| Darth Vader | 2.02 | 136.0 | 1.74358 | 1.1585357 |
| Leia Organa | 1.50 | 49.0 | 1.74358 | 0.8602988 |
| Owen Lars | 1.78 | 120.0 | 1.74358 | 1.0208879 |
| Beru Whitesun Lars | 1.65 | 75.0 | 1.74358 | 0.9463287 |
| R5-D4 | 0.97 | 32.0 | 1.74358 | 0.5563266 |
| Biggs Darklighter | 1.83 | 84.0 | 1.74358 | 1.0495645 |



dplyr::mutate() transforms columns

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Make a new column to see if each character is above or below the average height

hint: try using ifelse(condition, if_true_do_this, if_false_do_this)

```
test <- starwars %>%
  dplyr::mutate(height = height / 100) %>%
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE)) %>%
  dplyr::mutate(std_height = height / avg_height) %>%
  dplyr::select(name, height, mass, avg_height, std_height) %>%
dplyr::mutate(relative_height =
  ifelse(std_height > 1, "above", "below")) %>%
dplyr::filter(relative_height == "below")
```

Bonus: make a new dataframe that only keeps characters with heights BELOW average

dplyr::mutate() to add new (or change existing) columns



dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

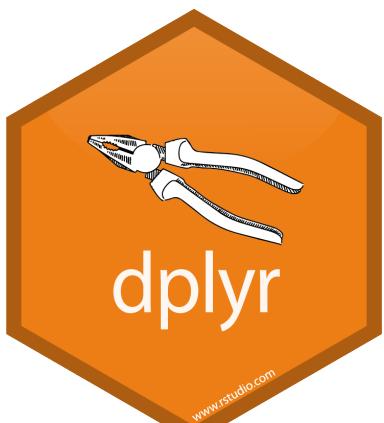
Make a new column to see if each character is above or below the average height

hint: try using ifelse(condition, if_true_do_this, if_false_do_this)

```
test <- starwars %>%
  dplyr::mutate(height = height / 100) %>%
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE)) %>%
  dplyr::mutate(std_height = height / avg_height) %>%
  dplyr::select(name, height, mass, avg_height, std_height) %>%
dplyr::mutate(relative_height =
    ifelse(std_height > 1, "above", "below")) %>%
dplyr::filter(std_height < 1)
```

Bonus: make a new dataframe that only keeps characters with heights BELOW average

dplyr::mutate() to add new (or change existing) columns



dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

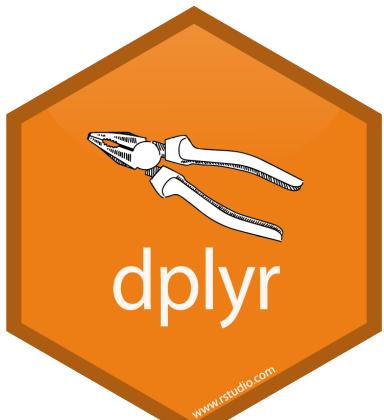
Make a new column to see if each character is above or below the average height

hint: try using ifelse(condition, if_true_do_this, if_false_do_this)

```
test <- starwars %>%
  dplyr::mutate(height = height / 100) %>%
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE)) %>%
  dplyr::mutate(std_height = height / avg_height) %>%
  dplyr::select(name, height, mass, avg_height, std_height) %>%
dplyr::mutate(relative_height =
    ifelse(std_height > 1, "above", "below")) %>%
dplyr::filter(height < avg_height)
```

Bonus: make a new dataframe that only keeps characters with heights BELOW average

dplyr::mutate() to add new (or change existing) columns



dplyr::group_by()

```
dplyr::group_by(dataframe, column_to_group_by)
```



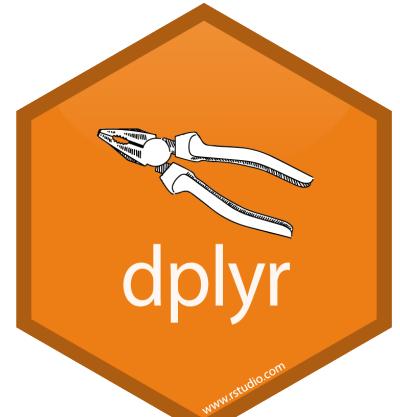
Doesn't change how the data looks, changes how the data interacts with other dplyr verbs

Group starwars dataframe by gender

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender)
```

```
View(starwars)
```

```
View(grouped_starwars)
```



dplyr::group_by() to group rows by columns

dplyr::group_by()

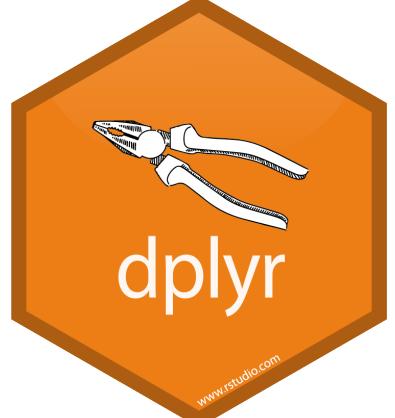
dplyr::group_by(dataframe, column_to_group_by)

ungrouped

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender |
|----|-----------------------|--------|--------|---------------|------------------|-----------|------------|---------------|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male |
| 2 | C-3PO | 167 | 75.0 | NA | gold | yellow | 112.0 | NA |
| 3 | R2-D2 | 96 | 32.0 | NA | white, blue | red | 33.0 | NA |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male |
| 5 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female |
| 6 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male |
| 7 | Beru Whitesun Lars | 165 | 75.0 | brown | light | blue | 47.0 | female |
| 8 | R5-D4 | 97 | 32.0 | NA | white, red | red | NA | NA |
| 9 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male |
| 10 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male |
| 11 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male |
| 12 | Wilhuff Tarkin | 180 | NA | auburn, grey | fair | blue | 64.0 | male |
| 13 | Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male |
| 14 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male |
| 15 | Greedo | 173 | 74.0 | NA | green | black | 44.0 | male |
| 16 | Jabba Desilijic Tiure | 175 | 1358.0 | NA | green-tan, brown | orange | 600.0 | hermaphrodite |
| 17 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male |
| 18 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | NA | male |
| 19 | Yoda | 66 | 17.0 | white | green | brown | 896.0 | male |
| 20 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male |
| 21 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male |
| 22 | IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none |
| 23 | Bossk | 190 | 113.0 | none | green | red | 53.0 | male |
| 24 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male |

grouped by gender

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender |
|----|-----------------------|--------|--------|---------------|------------------|-----------|------------|---------------|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male |
| 2 | C-3PO | 167 | 75.0 | NA | gold | yellow | 112.0 | NA |
| 3 | R2-D2 | 96 | 32.0 | NA | white, blue | red | 33.0 | NA |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male |
| 5 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female |
| 6 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male |
| 7 | Beru Whitesun Lars | 165 | 75.0 | brown | light | blue | 47.0 | female |
| 8 | R5-D4 | 97 | 32.0 | NA | white, red | red | NA | NA |
| 9 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male |
| 10 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male |
| 11 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male |
| 12 | Wilhuff Tarkin | 180 | NA | auburn, grey | fair | blue | 64.0 | male |
| 13 | Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male |
| 14 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male |
| 15 | Greedo | 173 | 74.0 | NA | green | black | 44.0 | male |
| 16 | Jabba Desilijic Tiure | 175 | 1358.0 | NA | green-tan, brown | orange | 600.0 | hermaphrodite |
| 17 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male |
| 18 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | NA | male |
| 19 | Yoda | 66 | 17.0 | white | green | brown | 896.0 | male |
| 20 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male |
| 21 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male |
| 22 | IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none |
| 23 | Bossk | 190 | 113.0 | none | green | red | 53.0 | male |
| 24 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male |



dplyr::group_by() to group rows by columns

dplyr::group_by()

```
dplyr::group_by(dataframe, column_to_group_by)
```



Doesn't change how the data looks, changes how the data interacts with other dplyr verbs

Group starwars dataframe by gender

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender)
```

Calculate average height *PER GENDER*

hint: group by gender FIRST

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender) %>%  
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE))
```

dplyr::group_by() to group rows by columns



dplyr::group_by()

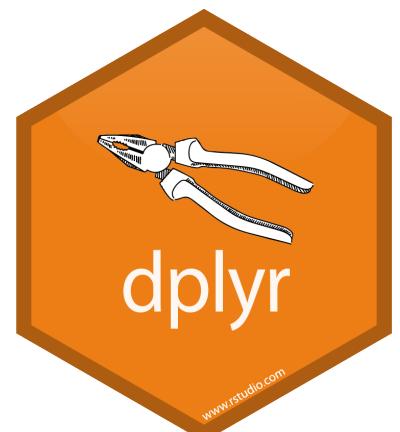
dplyr::group_by(dataframe, column_to_group_by)

Calculate average height *PER GENDER*

| name | gender | height | avg_height |
|--------------------|--------|--------|------------|
| Luke Skywalker | male | 1.72 | 1.792373 |
| C-3PO | NA | 1.67 | 1.200000 |
| R2-D2 | NA | 0.96 | 1.200000 |
| Darth Vader | male | 2.02 | 1.792373 |
| Leia Organa | female | 1.50 | 1.654706 |
| Owen Lars | male | 1.78 | 1.792373 |
| Beru Whitesun lars | female | 1.65 | 1.654706 |
| R5-D4 | NA | 0.97 | 1.200000 |
| Biggs Darklighter | male | 1.83 | 1.792373 |
| Obi-Wan Kenobi | male | 1.82 | 1.792373 |
| Anakin Skywalker | male | 1.88 | 1.792373 |
| Wilhuff Tarkin | male | 1.80 | 1.792373 |
| Chewbacca | male | 2.28 | 1.792373 |
| Han Solo | male | 1.80 | 1.792373 |

Calculate average height

| name | gender | height | avg_height |
|--------------------|--------|--------|------------|
| Luke Skywalker | male | 1.72 | 1.74358 |
| C-3PO | NA | 1.67 | 1.74358 |
| R2-D2 | NA | 0.96 | 1.74358 |
| Darth Vader | male | 2.02 | 1.74358 |
| Leia Organa | female | 1.50 | 1.74358 |
| Owen Lars | male | 1.78 | 1.74358 |
| Beru Whitesun lars | female | 1.65 | 1.74358 |
| R5-D4 | NA | 0.97 | 1.74358 |
| Biggs Darklighter | male | 1.83 | 1.74358 |
| Obi-Wan Kenobi | male | 1.82 | 1.74358 |
| Anakin Skywalker | male | 1.88 | 1.74358 |
| Wilhuff Tarkin | male | 1.80 | 1.74358 |
| Chewbacca | male | 2.28 | 1.74358 |
| Han Solo | male | 1.80 | 1.74358 |



dplyr::group_by() to group rows by columns

dplyr::group_by()

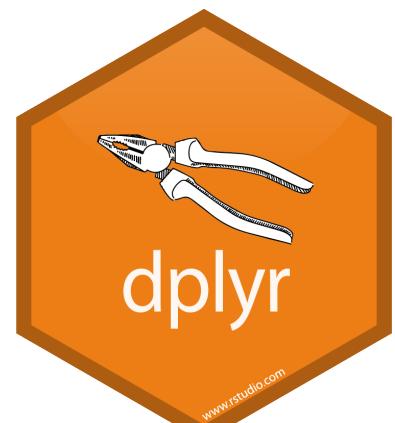
```
dplyr::group_by(dataframe, column_to_group_by)
```



dplyr::ungroup() removes all groups

Ungroup your grouped dataframe and re-calculate average height

```
ungrouped_starwars <- grouped_starwars %>%  
  dplyr::ungroup() %>%  
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE))
```



dplyr::group_by() to group rows by columns

dplyr::group_by()

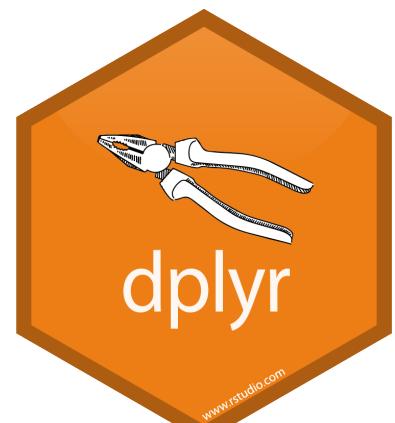
```
dplyr::group_by(dataframe, column_to_group_by)
```



You can also group by multiple columns

Calculate the average height per gender AND eye color

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender, eye_color) %>%  
  dplyr::mutate(avg_height = mean(height, na.rm = TRUE))
```



dplyr::group_by() to group rows by columns

dplyr::summarize()

dplyr::summarize(dataframe, new_column = expression)



summarize() is similar to mutate() but only keeps grouped columns

Calculate average height *PER GENDER*

hint: group by gender FIRST

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender) %>%  
  dplyr::summarize(avg_height = mean(height, na.rm = TRUE))
```



dplyr::summarize() to condense multiple columns

| gender | avg_height |
|---------------|------------|
| NA | 120.0000 |
| female | 165.4706 |
| hermaphrodite | 175.0000 |
| male | 179.2373 |
| none | 200.0000 |

dplyr::summarize()



summarize() is similar to mutate() but only keeps grouped columns

Compare mutate() and summarize() to calculate average height by gender

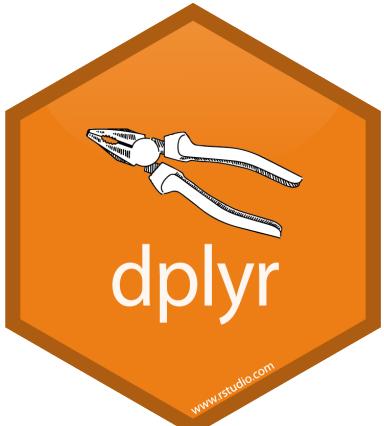
```
dplyr::mutate(avg_height = mean(height, na.rm = TRUE))
```

| name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships | avg_height |
|--------------------|--------|-------|-------------|-------------|-----------|------------|--------|-----------|---------|--|---|---------------------------------|------------|
| Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") | 179.2373 |
| C-3PO | 167 | 75.0 | NA | gold | yellow | 112.0 | NA | Tatooine | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) | 120.0000 |
| R2-D2 | 96 | 32.0 | NA | white, blue | red | 33.0 | NA | Naboo | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) | 120.0000 |
| Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | character(0) | TIE Advanced x1 | 179.2373 |
| Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | Imperial Speeder Bike | character(0) | 165.4706 |
| Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) | 179.2373 |
| Beru Whitesun Lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) | 165.4706 |
| R5-D4 | 97 | 32.0 | NA | white, red | red | NA | NA | Tatooine | Droid | A New Hope | character(0) | character(0) | 120.0000 |

```
dplyr::summarize(avg_height = mean(height, na.rm = TRUE))
```

| gender | avg_height |
|---------------|------------|
| NA | 120.0000 |
| female | 165.4706 |
| hermaphrodite | 175.0000 |
| male | 179.2373 |
| none | 200.0000 |

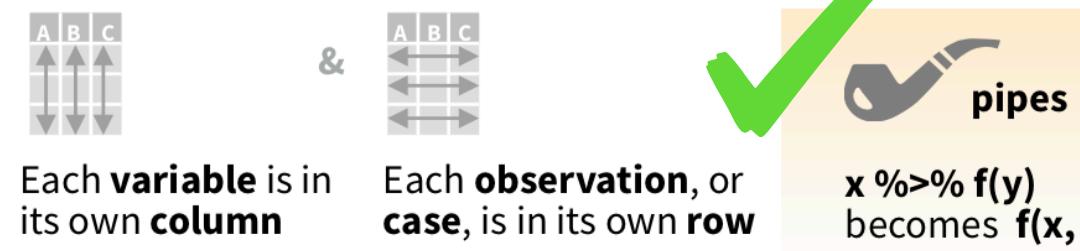
dplyr::summarize() to condense multiple columns



Data Transformation with dplyr :: CHEAT SHEET

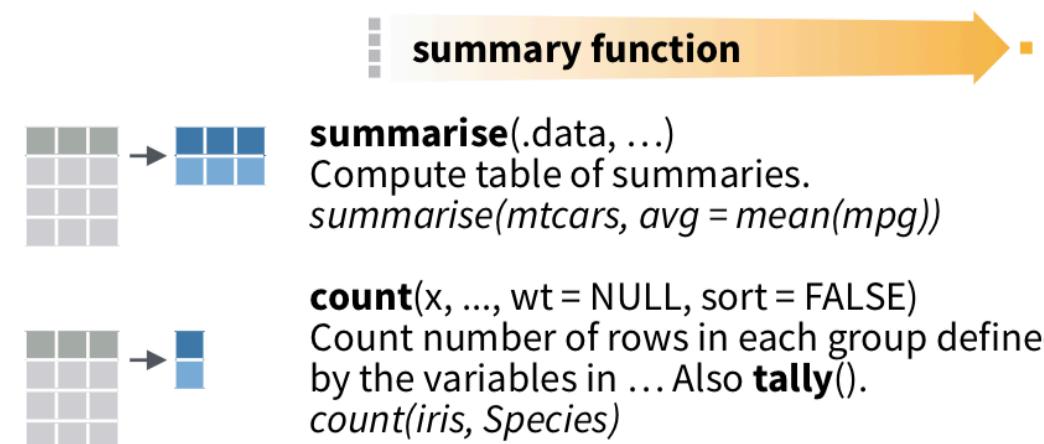


dplyr functions work with pipes and expect **tidy data**. In tidy data:



Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).



VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



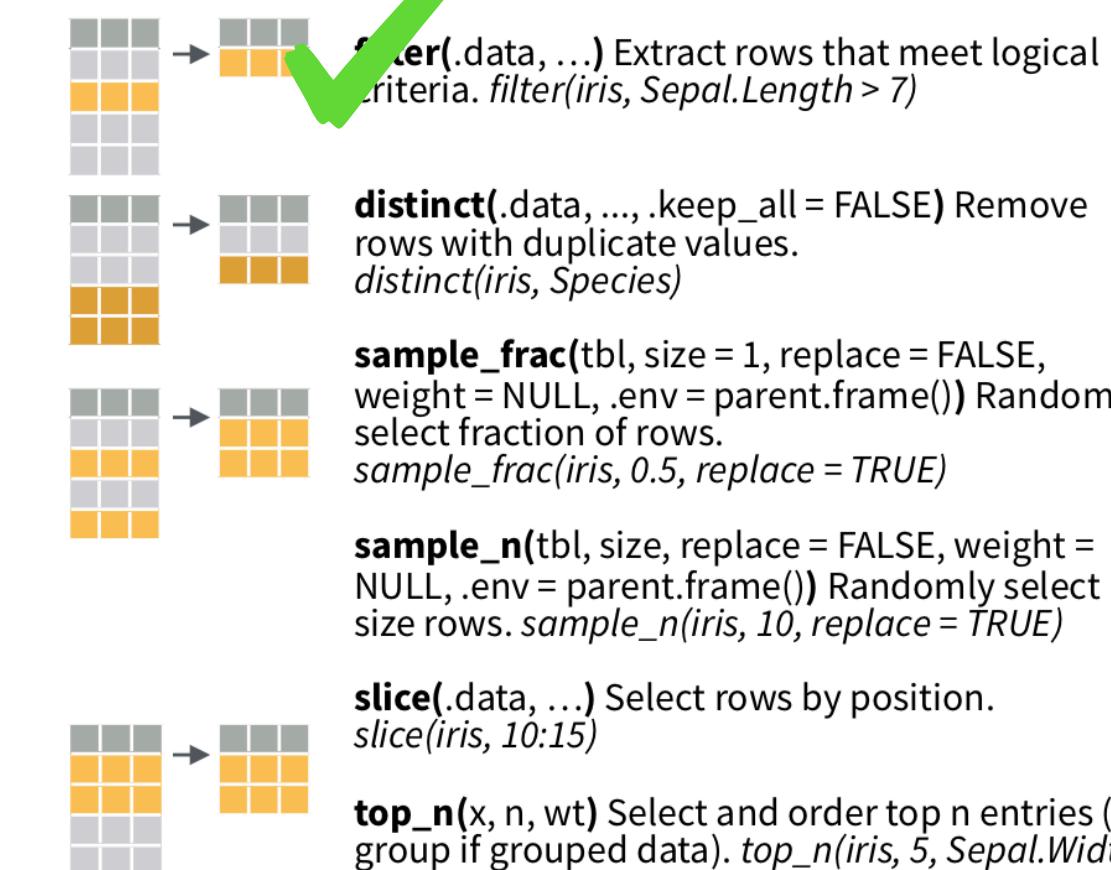
group_by(.data, ..., add = FALSE)
Returns copy of table grouped by ...
`g_iris <- group_by(iris, Species)`

ungroup(x, ...)
Returns ungrouped copy of table.
`ungroup(g_iris)`

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.



Logical and boolean operators to use with filter()

< <= is.na() %in% | xor()
> >= !is.na() ! &

See [?base::Logic](#) and [?Comparison](#) for help.

ARRANGE CASES

arrange(.data, ...) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
`arrange(mtcars, mpg)`
`arrange(mtcars, desc(mpg))`

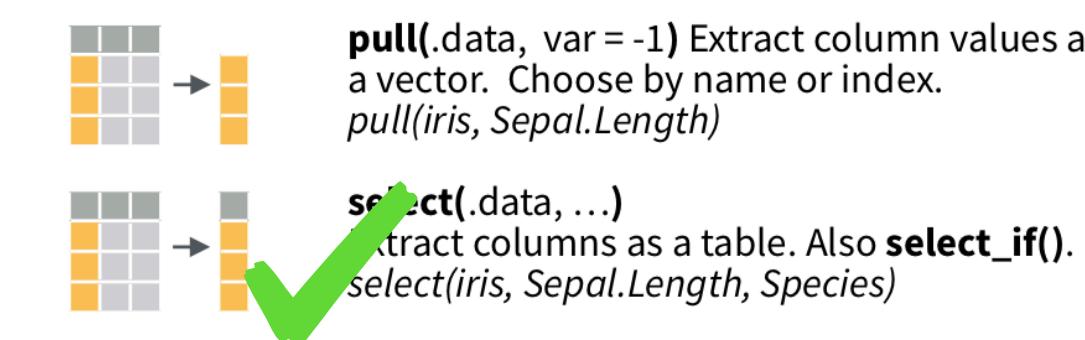
ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL)
Add one or more rows to a table.
`add_row(faithful, eruptions = 1, waiting = 1)`

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

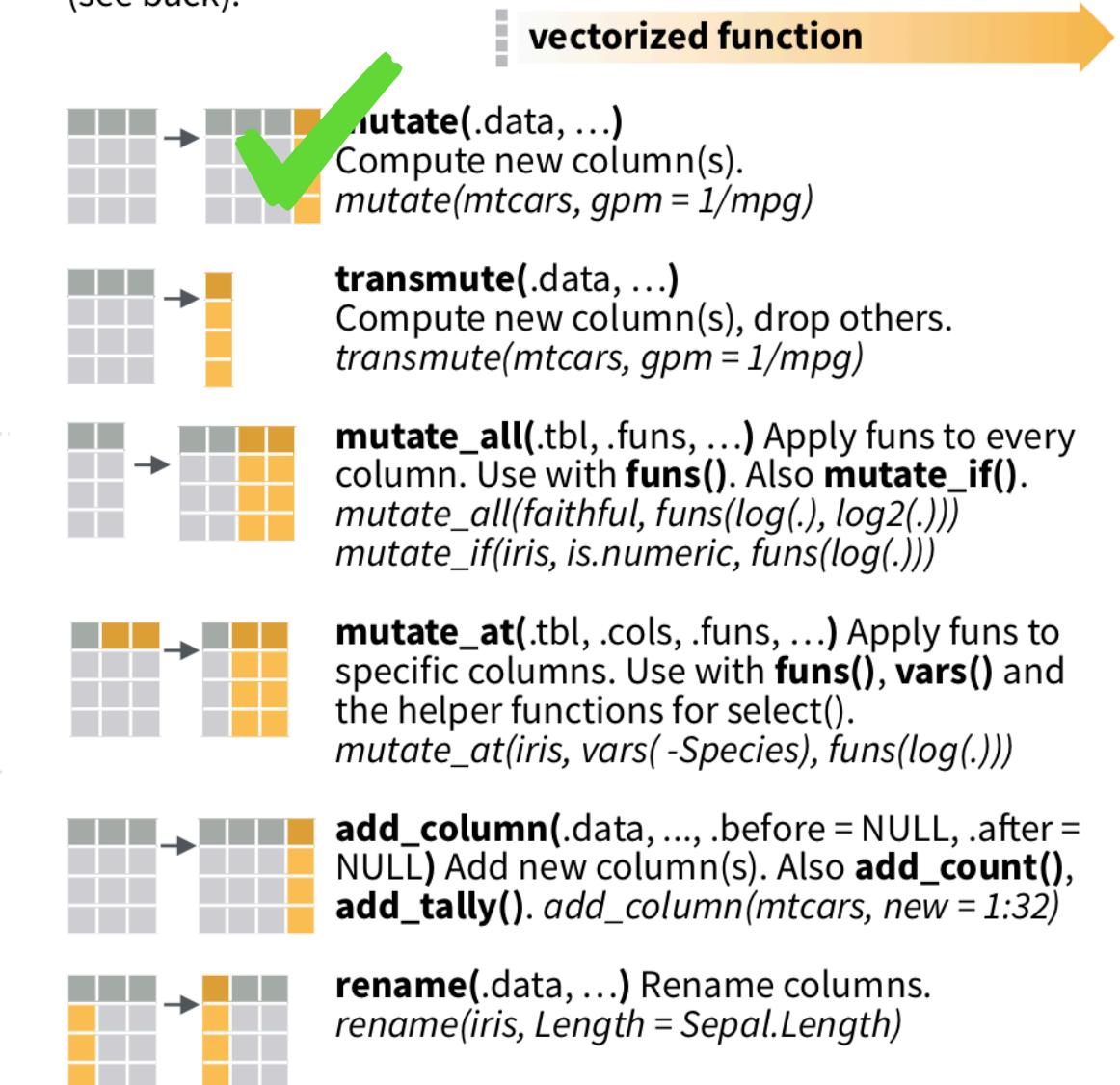


Use these helpers with **select()**,
e.g. `select(iris, starts_with("Sepal"))`

contains(match) **num_range(prefix, range)** : e.g. `mpg:cyl`
ends_with(match) **one_of(...)** -, e.g. `-Species`
matches(match) **starts_with(match)**

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).





Vector Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSETS

dplyr::lag() - Offset elements by 1
dplyr::lead() - Offset elements by -1

CUMULATIVE AGGREGATES

dplyr::cumall() - Cumulative all()
dplyr::cumany() - Cumulative any()
cummax() - Cumulative max()
dplyr::cummean() - Cumulative mean()
cummin() - Cumulative min()
cumprod() - Cumulative prod()
cumsum() - Cumulative sum()

RANKINGS

dplyr::cume_dist() - Proportion of all values <= dplyr::dense_rank() - rank w ties = min, no gaps dplyr::min_rank() - rank with ties = min dplyr::ntile() - bins into n bins dplyr::percent_rank() - min_rank scaled to [0,1] dplyr::row_number() - rank with ties = "first"

MATH

+, -, *, /, ^, %/%, %% - arithmetic ops
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - logical comparisons
dplyr::between() - x >= left & x <= right
dplyr::near() - safe == for floating point numbers

MISC

dplyr::case_when() - multi-case if_else()
iris %>% mutate(Species = case_when(
Species == "versicolor" ~ "versi",
Species == "virginica" ~ "virgi",
TRUE ~ Species))

dplyr::coalesce() - first non-NA values by element across a set of vectors
dplyr::if_else() - element-wise if() + else()
dplyr::na_if() - replace specific values with NA
pmax() - element-wise max()
pmin() - element-wise min()
dplyr::recode() - Vectorized switch()
dplyr::recode_factor() - Vectorized switch() for factors

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNTS

dplyr::n() - number of values/rows
dplyr::n_distinct() - # of uniques
sum(is.na()) - # of non-NA's

LOCATION

mean() - mean, also **mean(!is.na())**
median() - median

LOGICALS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

dplyr::first() - first value
dplyr::last() - last value
dplyr::nth() - value in nth location of vector

RANK

quantile() - nth quantile
min() - minimum value
max() - maximum value

SPREAD

IQR() - Inter-Quartile Range
mad() - median absolute deviation
sd() - standard deviation
var() - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

rownames_to_column()
Move row names into col.
a <- rownames_to_column(iris, var = "C")

column_to_rownames()
Move col in row names.
column_to_rownames(a, var = "C")

Also **has_rownames()**, **remove_rownames()**

Combine Tables

COMBINE VARIABLES

| X | Y | = |
|-----------|-----------|-----------------------|
| A B C | A B D | A B C A B D |
| a t 1 | a t 3 | a t 1 a t 3 |
| b u 2 | b u 2 | b u 2 b u 2 |
| c v 3 | d w 1 | c v 3 d w 1 |

Use **bind_cols()** to paste tables beside each other as they are.

bind_cols(...) Returns tables placed side by side as a single table.
BE SURE THAT ROWS ALIGN.

Use a "**Mutating Join**" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
Join data. Retain all values, all rows.

Use by = c("col1", "col2", ...) to specify one or more common columns to match on.
left_join(x, y, by = "A")

Use a named vector, by = c("col1" = "col2"), to match on columns that have different names in each table.
left_join(x, y, by = c("C" = "D"))

Use suffix to specify the suffix to give to unmatched columns that have the same name in both tables.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

COMBINE CASES

| X | Y | = |
|-----------|-----------|---------------------|
| A B C | A B C | A B C |
| a t 1 | b u 2 | a t 1 b u 2 |
| b u 2 | c v 3 | b u 2 c v 3 |
| c v 3 | d w 4 | c v 3 d w 4 |

Use **bind_rows()** to paste tables below each other as they are.

bind_rows(..., .id = NULL)
Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured)

intersect(x, y, ...)
Rows that appear in both x and y.

setdiff(x, y, ...)
Rows that appear in x but not y.

union(x, y, ...)
Rows that appear in x or y.
(Duplicates removed). union_all() retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

EXTRACT ROWS

| X | Y | = |
|-----------|-----------|-----------|
| A B C | A B D | A B C |
| a t 1 | a t 3 | a t 1 |
| b u 2 | b u 2 | b u 2 |
| c v 3 | d w 1 | c v 3 |

Use a "**Filtering Join**" to filter one table against the rows of another.

semi_join(x, y, by = NULL, ...)
Return rows of x that have a match in y.
USEFUL TO SEE WHAT WILL BE JOINED.

anti_join(x, y, by = NULL, ...)
Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.



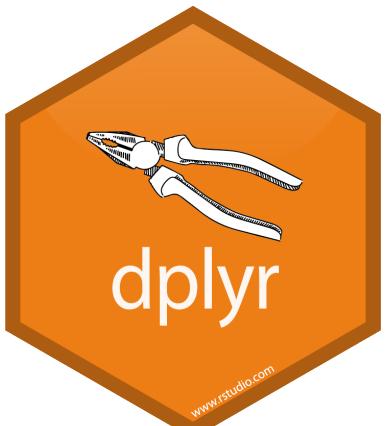
dplyr practice!

Calculate the birth year of each character given the column ‘birth_year’ is actually age in 2021...

GOAL:

| name | age | birth_year |
|-----------------------|-------|------------|
| Luke Skywalker | 19.0 | 2000.0 |
| C-3PO | 112.0 | 1907.0 |
| R2-D2 | 33.0 | 1986.0 |
| Darth Vader | 41.9 | 1977.1 |
| Leia Organa | 19.0 | 2000.0 |
| Owen Lars | 52.0 | 1967.0 |
| Beru Whitesun lars | 47.0 | 1972.0 |
| Biggs Darklighter | 24.0 | 1995.0 |
| Obi-Wan Kenobi | 57.0 | 1962.0 |
| Anakin Skywalker | 41.9 | 1977.1 |
| Wilhuff Tarkin | 64.0 | 1955.0 |
| Chewbacca | 200.0 | 1819.0 |
| Han Solo | 29.0 | 1990.0 |
| Greedo | 44.0 | 1975.0 |
| Jabba Desilijic Tiure | 600.0 | 1419.0 |
| Wedge Antilles | 21.0 | 1998.0 |
| Yoda | 896.0 | 1123.0 |
| Palpatine | 82.0 | 1937.0 |
| Boba Fett | 31.5 | 1987.5 |
| IG-88 | 15.0 | 2004.0 |

*Hint: rename “birth_year” as “age” first
then calculate “birth_year”*



dplyr practice!

Calculate the birth year of each character given the column ‘birth_year’ is actually age in 2021...

GOAL:

| name | age | birth_year |
|----------------|-------|------------|
| Luke Skywalker | 19.0 | 2000.0 |
| C-3PO | 112.0 | 1907.0 |
| R2-D2 | 33.0 | 1986.0 |
| Darth Vader | 41.9 | 1977.1 |
| Leia Organa | 19.0 | 2000.0 |
| Owen Lars | 52.0 | 1967.0 |

CODE:

```
practice1 <- starwars %>%
  dplyr::select(name, age = birth_year) %>%
  dplyr::mutate(birth_year = 2019 - age) %>%
  dplyr::filter(!is.na(age))
```

| | | |
|-----------|-------|--------|
| Yoda | 896.0 | 1123.0 |
| Palpatine | 82.0 | 1937.0 |
| Boba Fett | 31.5 | 1987.5 |
| IG-88 | 15.0 | 2004.0 |

dplyr practice!

Calculate the number of characters with each hair color combination

GOAL:

| hair_color | number |
|---------------|--------|
| auburn | 1 |
| auburn, grey | 1 |
| auburn, white | 1 |
| black | 13 |
| blond | 3 |
| blonde | 1 |
| brown | 18 |
| brown, grey | 1 |
| grey | 1 |
| none | 37 |
| unknown | 1 |
| white | 4 |
| NA | 5 |

*Note: “brown, grey” is different than
“brown” or “grey”*

*Hint: the function `n()` counts the
number of observations in a group*

BONUS: Combine
“blond” and “blonde” as
one category and re-
calculate



dplyr practice!

Calculate the number of characters with each hair color combination

GOAL:

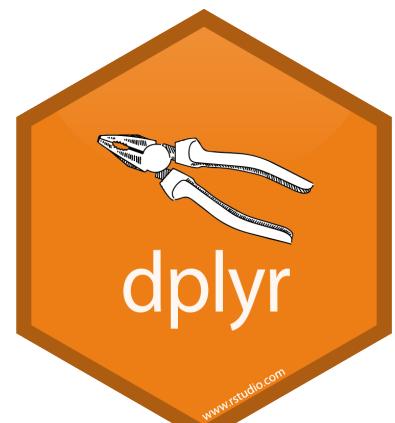
| hair_color | number |
|---------------|--------|
| auburn | 1 |
| auburn, grey | 1 |
| auburn, white | 1 |
| black | 12 |
| blond | |
| blonde | |
| brown | |
| brown, grey | |
| grey | |
| none | |
| unknown | |
| white | 4 |
| NA | 5 |

*Note: “brown, grey” is different than
“brown” or “grey”*

CODE:

```
practice2 <- starwars %>%  
  dplyr::select(name, hair_color) %>%  
  dplyr::group_by(hair_color) %>%  
  dplyr::summarize(number = n())
```

calculate



dplyr practice!

Calculate the number of characters with each hair color combination

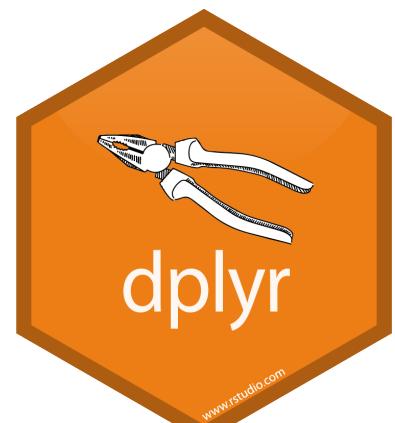
| hair_color | number |
|---------------|--------|
| auburn | 1 |
| auburn, grey | 1 |
| auburn, white | 1 |

GOAL:

*Note: “brown, grey” is different than
“brown” or “grey”*

CODE:

```
practice2 <- starwars %>%
  dplyr::select(name, hair_color) %>%
  dplyr::mutate(hair_color = ifelse(hair_color == "blond",
                "blonde", hair_color)) %>%
  dplyr::group_by(hair_color) %>%
  dplyr::summarize(number = n())
```



dplyr practice!

Calculate the number of characters with each hair color combination
Note: Keep ALL characters

GOAL:

| name | hair_color | number |
|-----------------------|---------------|--------|
| Luke Skywalker | blond | 3 |
| C-3PO | NA | 5 |
| R2-D2 | NA | 5 |
| Darth Vader | none | 37 |
| Leia Organa | brown | 18 |
| Owen Lars | brown, grey | 1 |
| Beru Whitesun lars | brown | 18 |
| R5-D4 | NA | 5 |
| Biggs Darklighter | black | 13 |
| Obi-Wan Kenobi | auburn, white | 1 |
| Anakin Skywalker | blond | 3 |
| Wilhuff Tarkin | auburn, grey | 1 |
| Chewbacca | brown | 18 |
| Han Solo | brown | 18 |
| Greedo | NA | 5 |
| Jabba Desilijic Tiure | NA | 5 |
| Wedge Antilles | brown | 18 |
| Jek Tono Porkins | brown | 18 |

Note: “brown, grey” is different than “brown” or “grey”

Hint: the function n() counts the number of observations in a group



dplyr practice!

Calculate the number of characters with each hair color combination
Note: Keep ALL characters

GOAL:

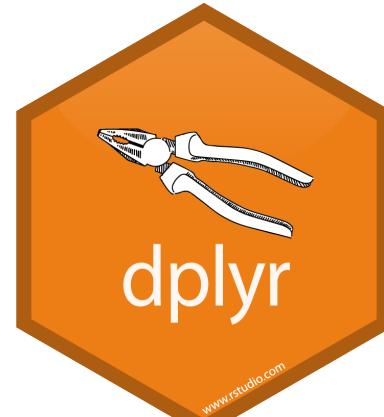
| name | hair_color | number |
|-----------------------|---------------|--------|
| Luke Skywalker | blond | 3 |
| C-3PO | NA | 5 |
| R2-D2 | NA | 5 |
| Darth Vader | none | 37 |
| Leia Organa | brown | 18 |
| Owen Lars | brown, grey | 1 |
| Beru Whitesun Lars | brown | |
| R5-D4 | NA | |
| Biggs Darklighter | black | |
| Obi-Wan Kenobi | auburn, white | |
| Anakin Skywalker | blond | |
| Wilhuff Tarkin | auburn, grey | |
| Chewbacca | brown | |
| Han Solo | brown | |
| Greedo | NA | |
| Jabba Desilijic Tiure | NA | |
| Wedge Antilles | brown | 18 |
| Jek Tono Porkins | brown | 18 |

Note: “brown, grey” is different than “brown” or “grey”

Hint: the function n() counts the

CODE:

```
practice3 <- starwars %>%  
  dplyr::select(name, hair_color) %>%  
  dplyr::group_by(hair_color) %>%  
  dplyr::mutate(number = n())
```



dplyr practice!

Select all characters and their homeworld in the film “Return of the Jedi”

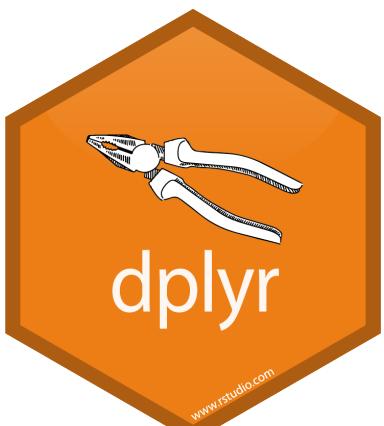
GOAL:

| name | homeworld |
|----------------|-----------|
| Luke Skywalker | Tatooine |
| C-3PO | Tatooine |
| R2-D2 | Naboo |
| Darth Vader | Tatooine |
| Leia Organa | Alderaan |
| Obi-Wan Kenobi | Stewjon |

Hint: try using grep1 ()

CODE:

```
practice4 <- starwars %>%
  dplyr::filter(grep1("Return of the Jedi", films)) %>%
  dplyr::select(name, homeworld)
```



| | |
|-----------------------|---------|
| Arvel Crynyd | NA |
| Wicket Systri Warrick | Endor |
| Nien Nunb | Sullust |
| Bib Fortuna | Ryloth |

Summary statistics and general math

```
vec <- c(1, 1, 3, 4, 5, 5, 7, 8, 9, 9)

# calculate sum of vector          # calculate product of vector
> sum(vec)                      > prod(vec)
52                                1360800

# calculate mean of vector        # calculate summary statistics
> mean(vec)                     > summary(vec)
5.2                               Min. 1st Qu. Median     Mean 3rd Qu. Max.
                                 1.00   3.25   5.00   5.20   7.75   9.00

# calculate sd of vector          # calculate table of values
> sd(vec)                        > table(vec)
3.01                             1 3 4 5 7 8 9
                                2 1 1 2 1 1 2
```

Sampling from a distribution

```
vec <- c(1, 1, 3, 4, 5, 5, 7, 8, 9, 9)

# sample without replacement          # sample with set probabilities
> sample(vec, 3, replace = FALSE) > sample(c("Yes", "No"),
[1] 7 9 3                               10,
                                         replace = TRUE,
                                         prob = c(0.7, 0.3))

# sample with replacement
> sample(vec, 3, replace = TRUE)
[1] 8 5 5                               [1] "Yes" "Yes" "No"   "Yes" "Yes"
                                         [6] "Yes" "Yes" "Yes"  "Yes" "Yes"
```

```
# set random generator seed
> set.seed(192)
```

Generating a random distribution

```
# generate NORMAL distribution  
> rnorm(5, mean = 0, sd = 1)  
0.8838757 1.8028252 -1.4362813 0.6163822 -0.6357746
```

```
# generate UNIFORM distribution  
> runif(5, min = 1, max = 10)  
9.294435 1.518637 5.570818 5.031591 6.430223
```

```
# generate BINOMIAL distribution  
> rbinom(10, size = 1, prob = 0.5)  
1 0 1 1 1 0 0 0 1
```

Plotting – Histograms

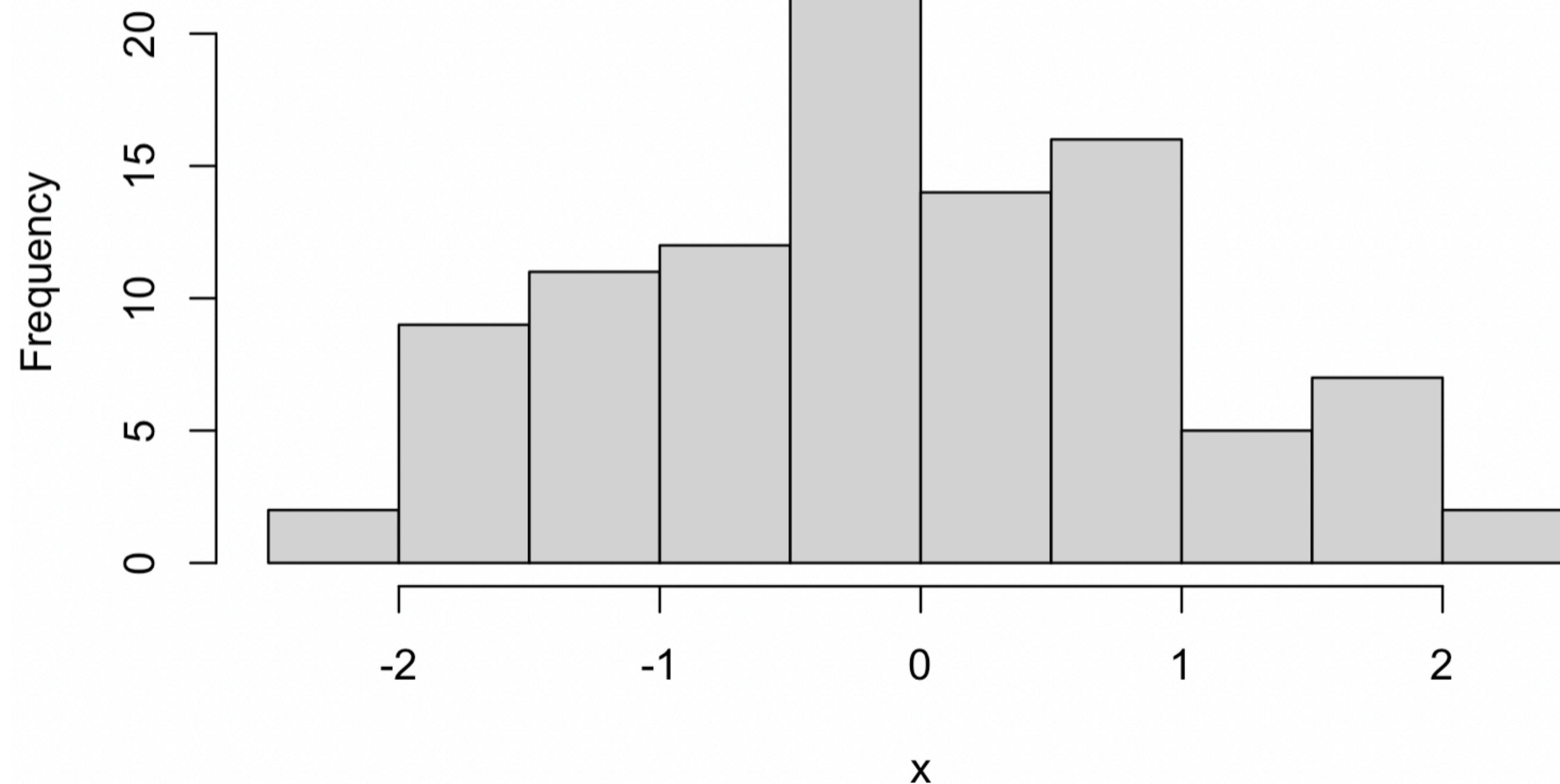
```
# generate NORMAL distribution
```

```
> x <- rnorm(100, mean = 0, sd = 1)
```

```
# plot with base R
```

```
> hist(x)
```

Histogram of x



Plotting – Histograms

```
# generate NORMAL distribution
```

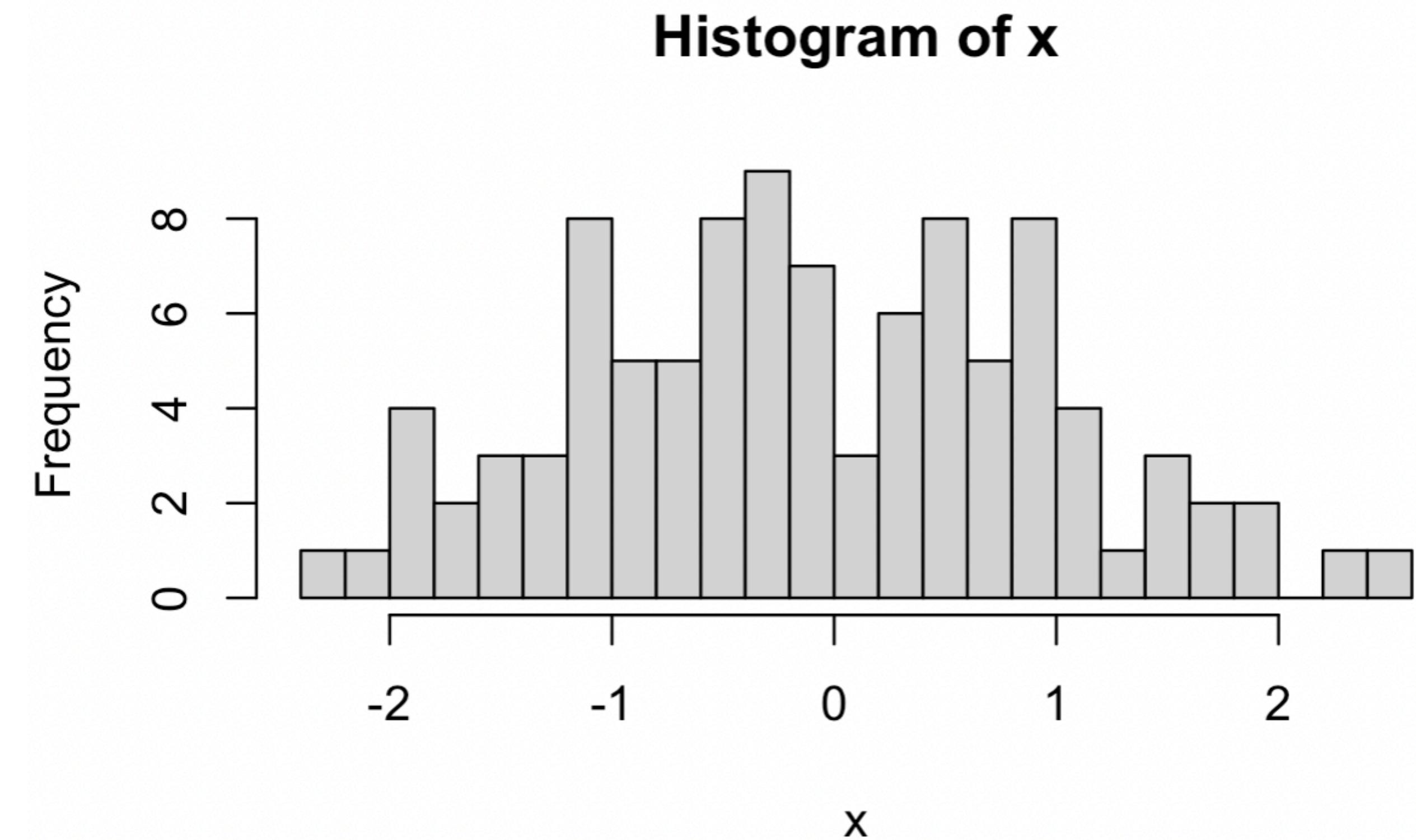
```
> x <- rnorm(100, mean = 0, sd = 1)
```

```
# plot with base R
```

```
> hist(x)
```

```
# change number of bins
```

```
> hist(x, breaks = 20)
```



Plotting with ggplot2

ggplot

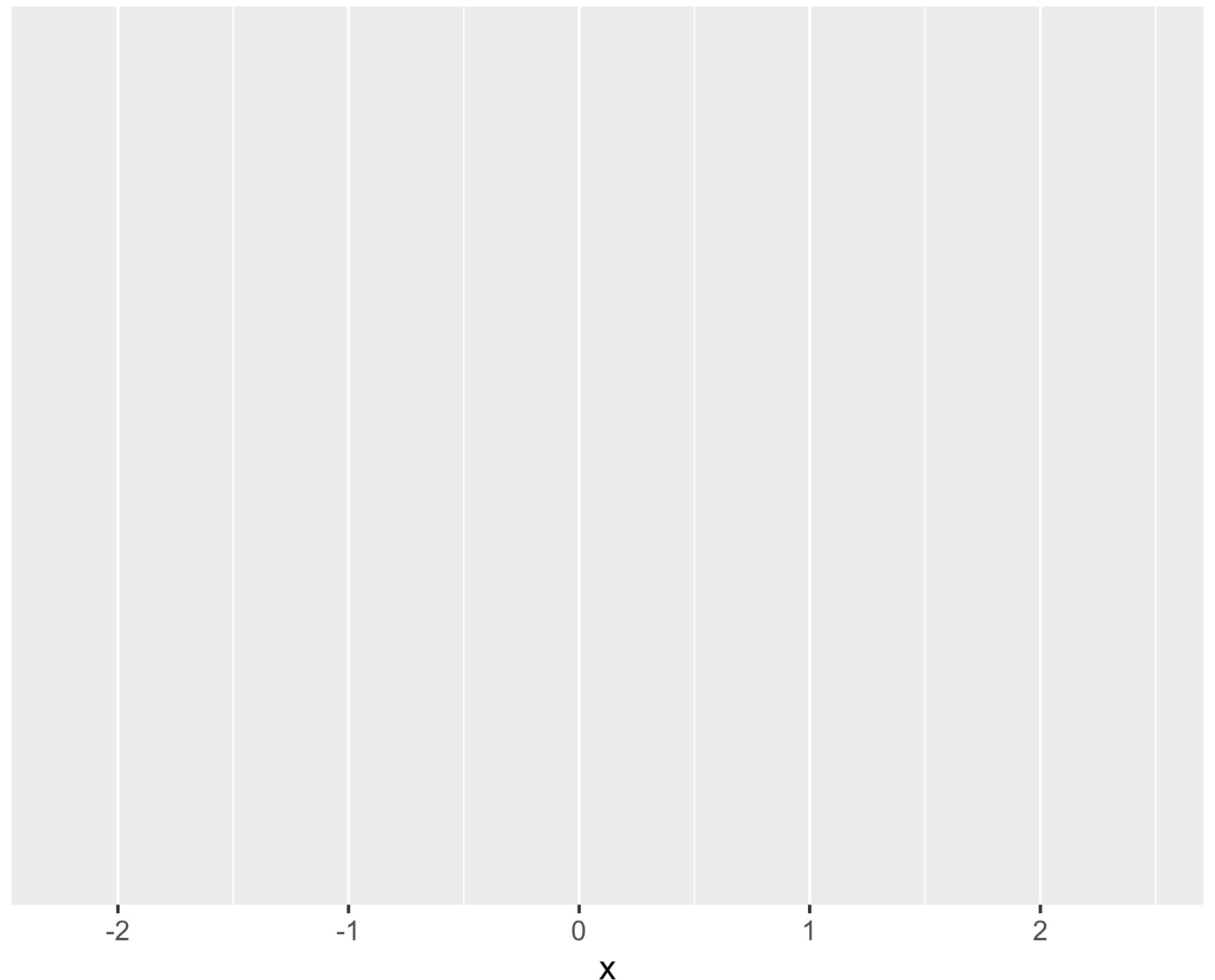


Plotting with ggplot2

```
# generate NORMAL distribution  
> x <- rnorm(100, mean = 0, sd = 1)  
  
# make a ggplot object  
> ggplot2::ggplot(data.frame(x))
```

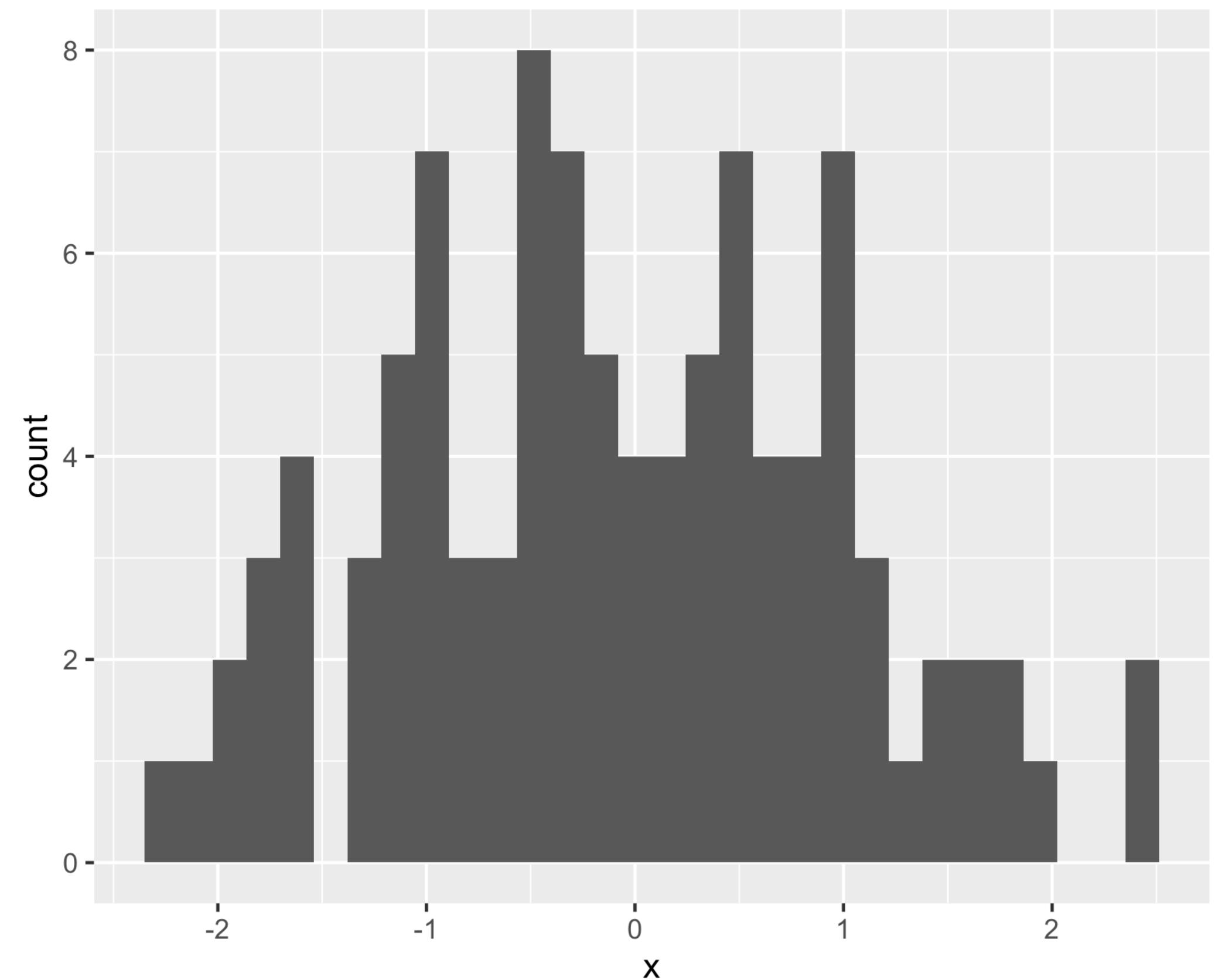
Plotting with ggplot2

```
# generate NORMAL distribution  
> x <- rnorm(100, mean = 0, sd = 1)  
  
# make a ggplot object  
> ggplot2::ggplot(data.frame(x)) +  
  
# add x and y axes  
  ggplot2::aes(x = x)
```



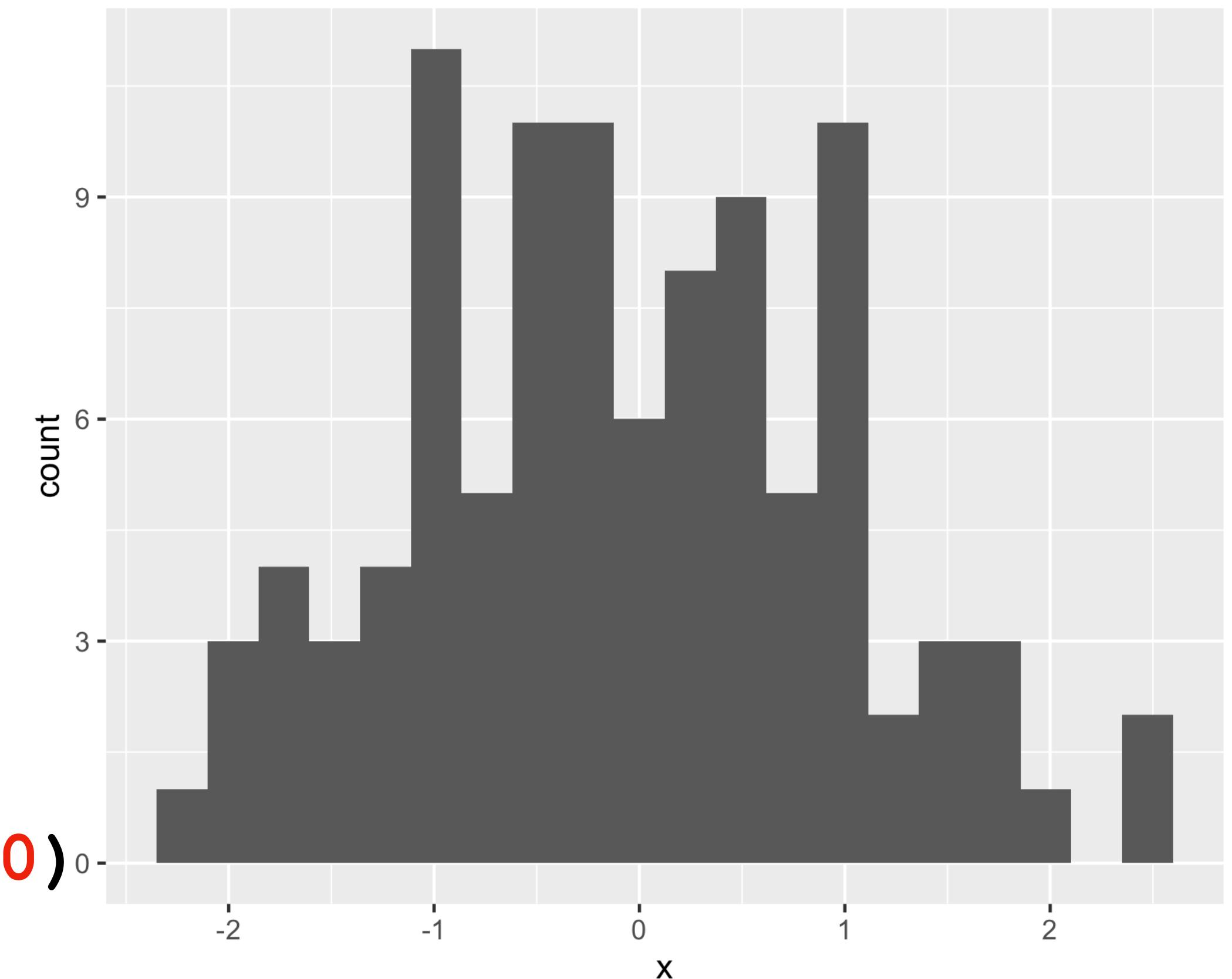
Plotting with ggplot2

```
# generate NORMAL distribution  
> x <- rnorm(100, mean = 0, sd = 1)  
  
# make a ggplot object  
> ggplot2::ggplot(data.frame(x)) +  
  
# add x and y axes  
    ggplot2::aes(x = x) +  
  
# add how to plot data  
    ggplot2::geom_histogram()
```



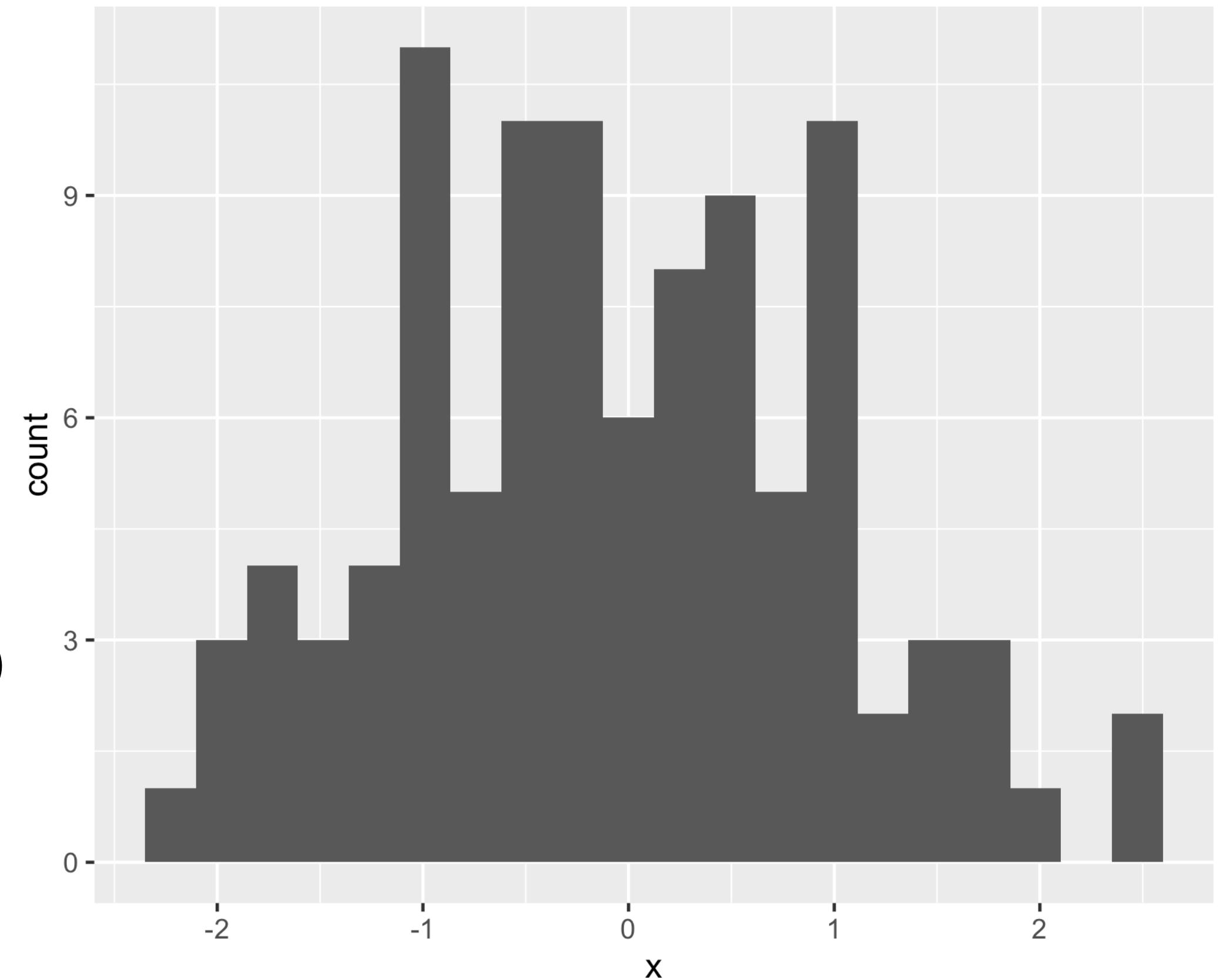
Plotting with ggplot2

```
# generate NORMAL distribution  
  
> x <- rnorm(100, mean = 0, sd = 1)  
  
# make a ggplot object  
  
> ggplot2::ggplot(data.frame(x)) +  
  
# add x and y axes  
  
    ggplot2::aes(x = x) +  
  
# add how to plot data  
  
    ggplot2::geom_histogram(bins = 20)
```



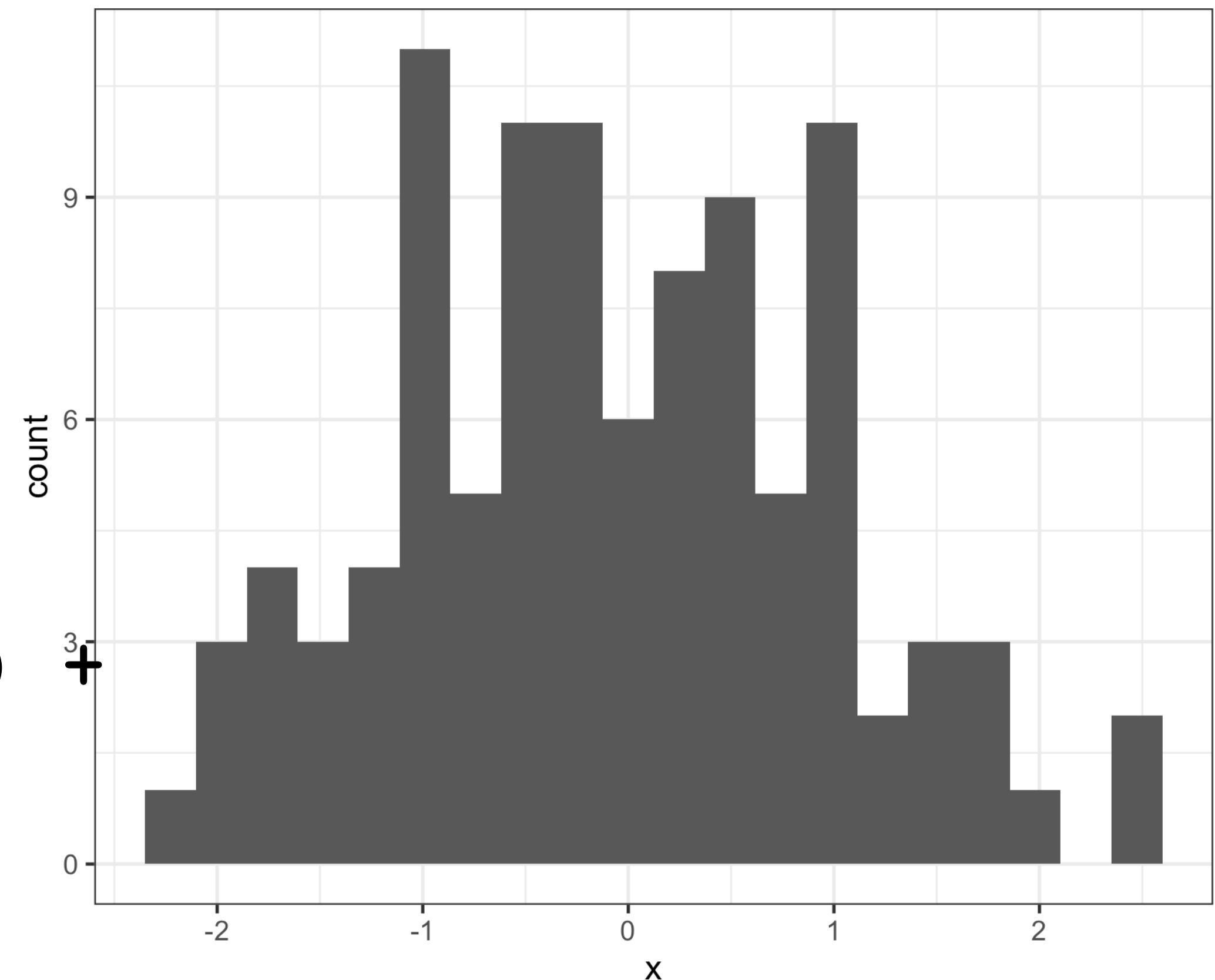
Plotting with ggplot2

```
# generate NORMAL distribution  
> x <- rnorm(100, mean = 0, sd = 1)  
  
# make a ggplot object  
> ggplot2::ggplot(data.frame(x)) +  
  # add x and y axes  
  ggplot2::aes(x = x) +  
  # add how to plot data  
  ggplot2::geom_histogram(bins = 20)  
  # add pre-set theme  
  ggplot2::theme_bw()
```



Plotting with ggplot2

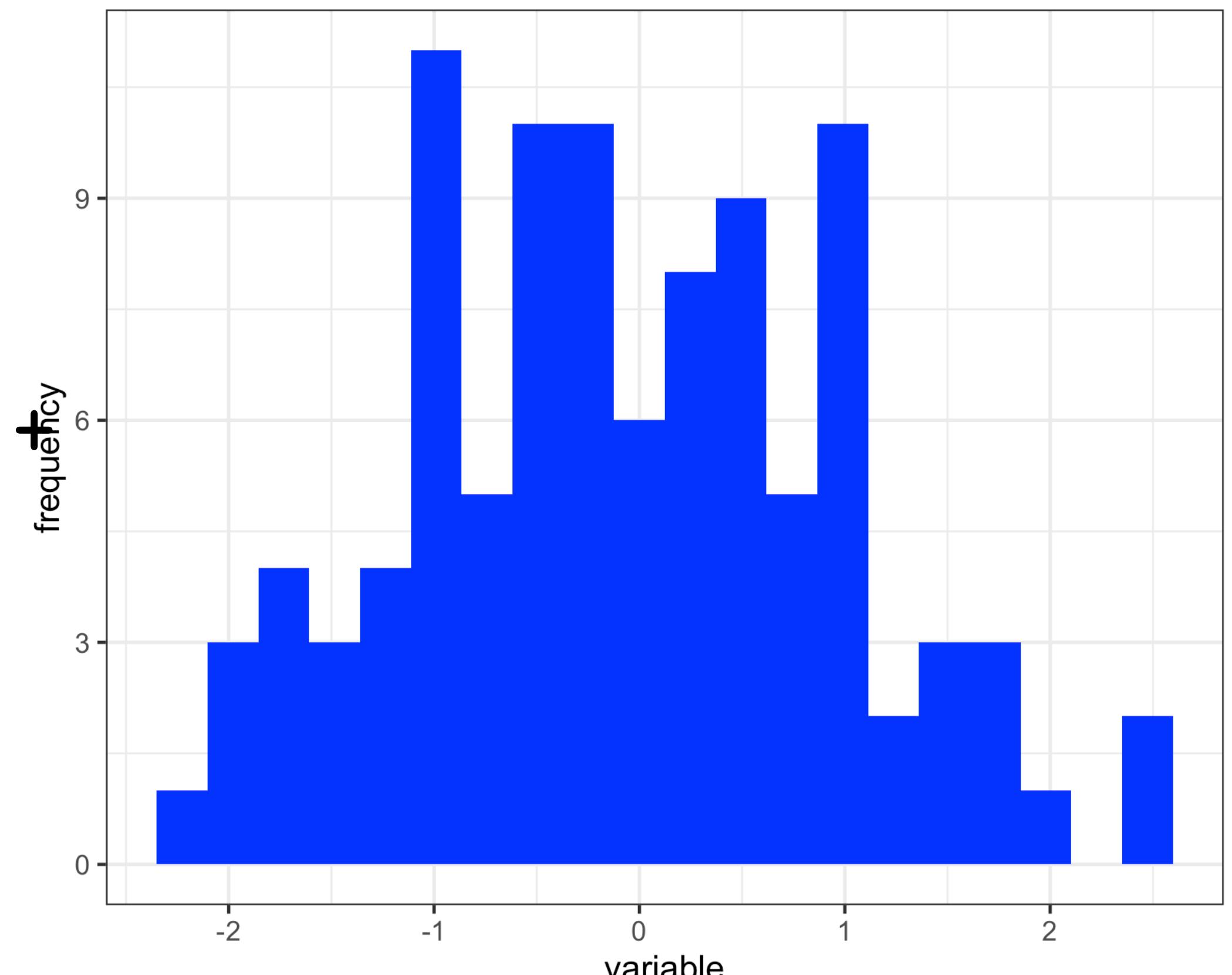
```
# generate NORMAL distribution  
> x <- rnorm(100, mean = 0, sd = 1)  
  
# make a ggplot object  
> ggplot2::ggplot(data.frame(x)) +  
  # add x and y axes  
  ggplot2::aes(x = x) +  
  # add how to plot data  
  ggplot2::geom_histogram(bins = 20)  
  # add pre-set theme  
  ggplot2::theme_bw()
```



Plotting with ggplot2

```
# make a ggplot object  
> ggplot2::ggplot(data.frame(x)) +  
  # add x and y axes  
  ggplot2::aes(x = x) +  
  # add how to plot data  
  ggplot2::geom_histogram(bins = 20,  
                         fill = "blue")  
  # add pre-set theme  
  ggplot2::theme_bw() +  
  # change labels  
  ggplot2::labs(x = "variable",  
                y = "frequency")
```

Unlimited modifications!!!



Data Visualization with ggplot2 :: CHEAT SHEET

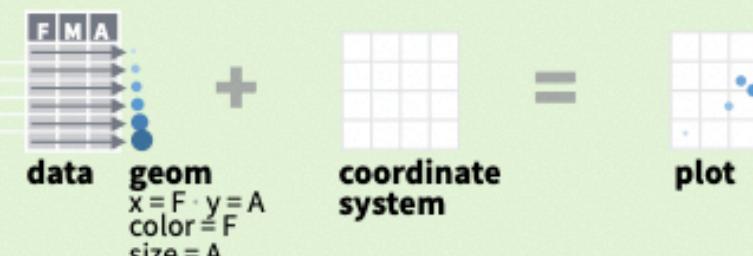


Basics

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 = z)) - x, xend, y, yend,
alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round",
linemitre = 1) x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax =
long + 1, ymax = lat + 1)) - xmax, xmin, ymax,
ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900,
ymax = unemploy + 900)) - x, ymax, ymin,
alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group,
linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha,
color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha,
color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(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,
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

C **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

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 = USArests\$Murder,
state = tolower(rownames(USArests)))
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

R Markdown

(Open a new R Markdown file)

```
1  ---
2  title: "test"
3  author: "Katie Evans"
4  date: "7/23/2018"
5  output: html_document
6  ---
7
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. For more details on using R Markdown see <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 ```
21
22 ## Including Plots
23
24 You can also embed plots, for example:
25
26 ```{r pressure, echo=FALSE}
27 plot(pressure)
28 ```
29
30 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
31
```

test

Katie Evans

7/23/2018

R Markdown

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>.

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:

HTML or PDF

Great for data analysis summary

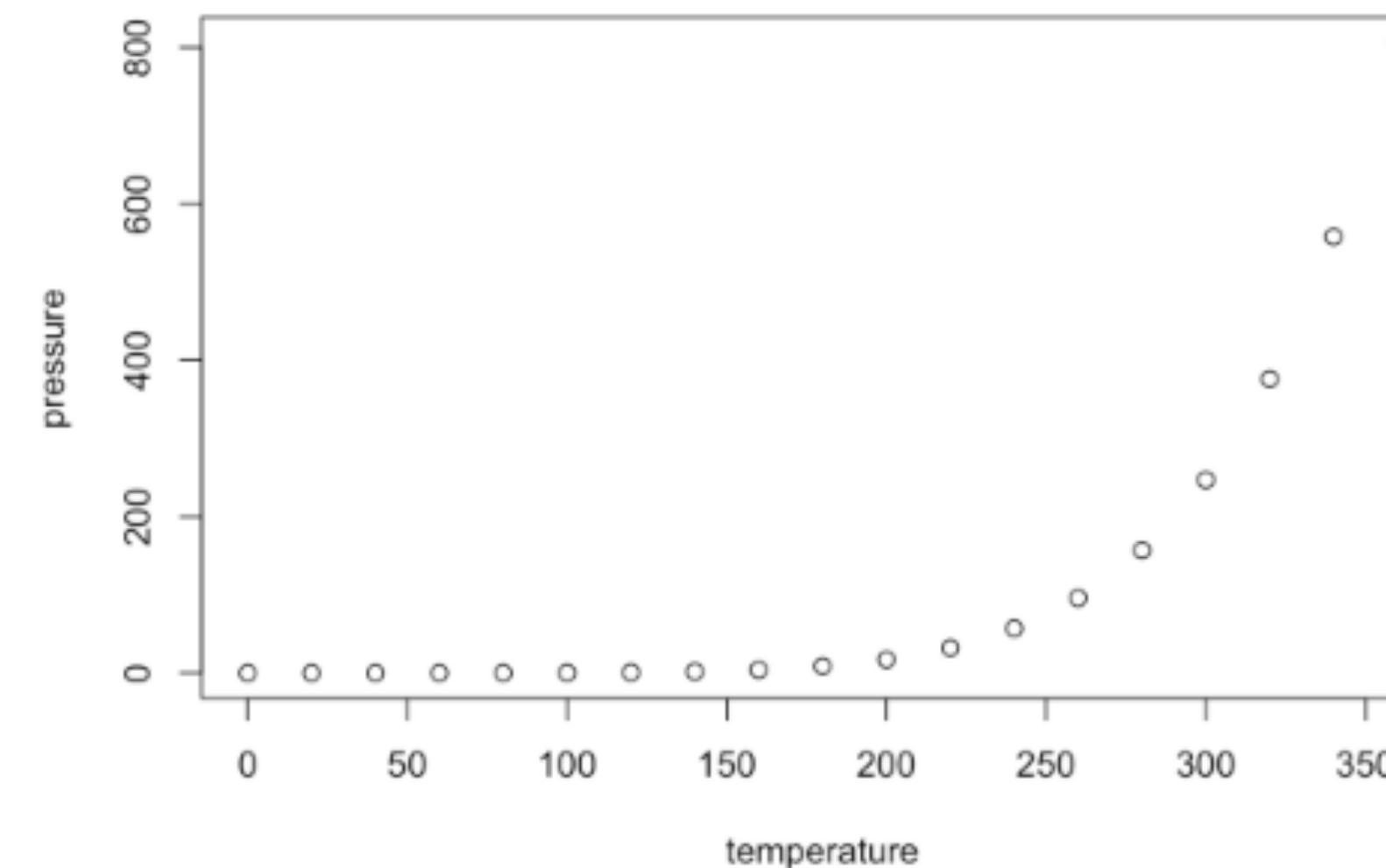
Try pressing “knit” at the top of your script

```
summary(cars)
```

```
##      speed      dist
## Min.   : 4.0   Min.   :  2.00
## 1st Qu.:12.0   1st Qu.: 26.00
## Median :15.0   Median : 36.00
## Mean   :15.4   Mean   : 42.98
## 3rd Qu.:19.0   3rd Qu.: 56.00
## Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.