

Tidyverse Tools in R for Data Science and Statistical Inference

Dr. Jessica Minnier and Dr. Chester Ismay



Slides available at <http://bit.ly/csp-tidy>

PDF slides at <http://bit.ly/csp-tidy-pdf>



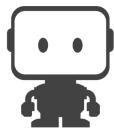
Associate Professor of
Biostatistics



 @datapointier
 @jminnier
 jessicaminnier.com



Data Science Evangelist



DataRobot

 @old_man_chester
 @ismayc
 chester.rbind.io

Table of Contents

Part 1

- Introduction and Setup
- Data Wrangling
- Data Visualization Basics

Part 2

- Sampling
- Inference

Prior Installation

Make sure you have the current R, RStudio, & R packages

- Novice's Guide on ModernDive.com
-

- R (version 4.0.2 or greater)
 - RStudio (version 1.4 or greater)
-

- Run this in the RStudio Console

```
pkgs <- c(  
  "tidyverse", "moderndive", "dslabs",  
  "infer", "janitor"  
)  
install.packages(pkgs)
```

Getting started

1. Open HTML slides: <https://bit.ly/csp-tidy>
2. Open RStudio
3. Download course materials to your preferred location by running this in the RStudio Console:

```
usethis::use_course("bit.ly/csp-tidy")
```

4. Open the Rproj file in the folder that just opened
5. Edit Google Doc to ask and answer questions:
<https://bit.ly/csp-tidy-doc>

R Data Types

Data types review

Vector/variable

- Type of vector (`int`, `num` or `dbl`, `chr`, `lgl`, `date`)

Data frame

- Vectors of (potentially) different types
- Each vector has the same number of rows

Data types review

```
library(tibble) # tibble is the tidyverse data.frame
library(lubridate)
ex1 <- tibble(
  vec1 = c(1980, 1990, 2000, 2010),
  vec2 = c(1L, 2L, 3L, 4L),
  vec3 = c("low", "low", "high", "high"),
  vec4 = c(TRUE, FALSE, FALSE, FALSE),
  vec5 = ymd(c("2017-05-23", "1776/07/04",
              "1983-05/31", "1908/04-01")))
)
ex1
```

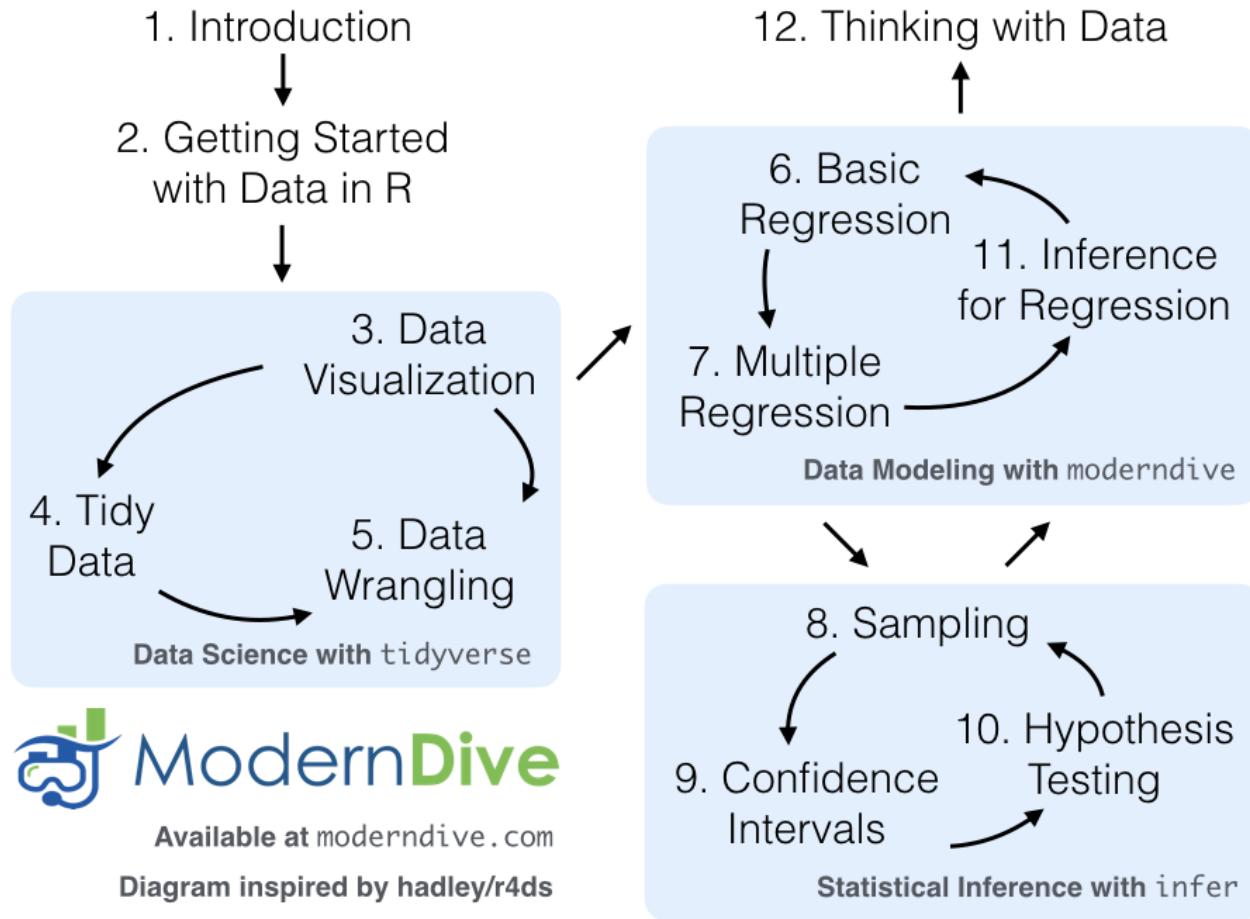
```
# A tibble: 4 x 5
  vec1  vec2  vec3  vec4  vec5
  <dbl> <int> <chr> <lgl> <date>
1 1980      1 low   TRUE  2017-05-23
2 1990      2 low   FALSE 1776-07-04
3 2000      3 high  FALSE 1983-05-31
4 2010      4 high  FALSE 1908-04-01
```

Learning objectives

Part 1

- Distinguish between different `{tidyverse}` packages
- Assess different `{dplyr}` functions for wrangling data
- Discuss the power of "tidy data"
- Develop an intuition behind `{ggplot2}` plotting syntax

Shameless plug



Available at moderndive.com

Diagram inspired by [hadley/r4ds](#)

Getting started with {dplyr}



Alison Horst

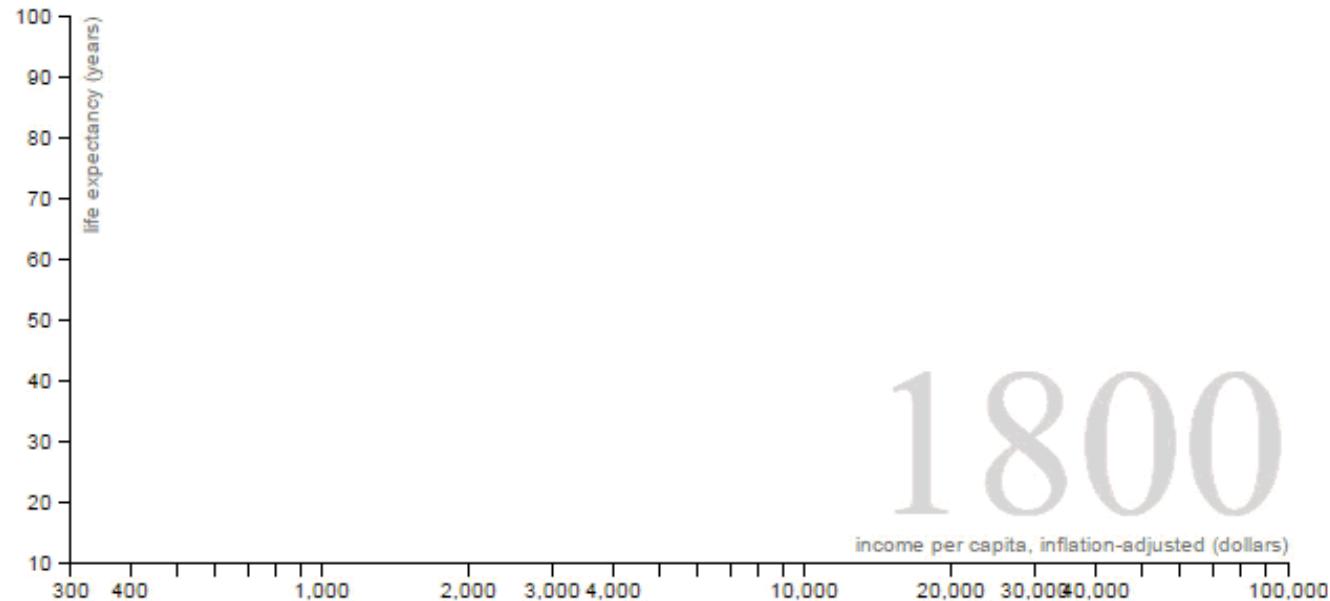
dplyr package

Welcome to the tidyverse!

The {tidyverse} is a collection of R packages that share common philosophies and are designed to work together.



First motivating example for today



- Inspired by the late, great Hans Rosling

The `gapminder` data set in `{dslabs}`

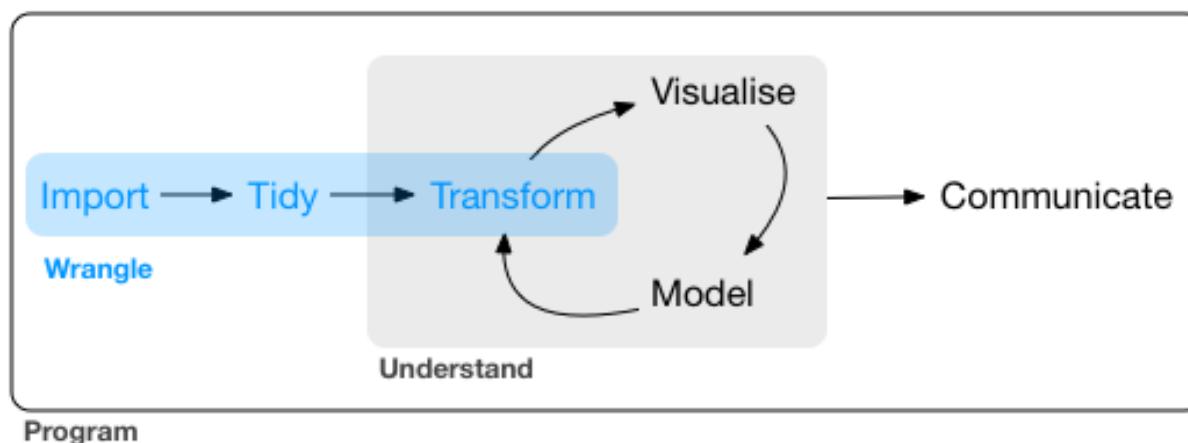
```
library(dslabs)
library(dplyr)
gapminder <- tibble(gapminder)
glimpse(gapminder)
```

```
Rows: 10,545
Columns: 9
$ country           <fct> Albania, Algeria, Angola, Antigua and...
$ year              <int> 1960, 1960, 1960, 1960, 1960, 1960, 1...
$ infant_mortality <dbl> 115.40, 148.20, 208.00, NA, 59.87, NA...
$ life_expectancy   <dbl> 62.87, 47.50, 35.98, 62.97, 65.39, 66...
$ fertility          <dbl> 6.19, 7.65, 7.32, 4.43, 3.11, 4.55, 4...
$ population         <dbl> 1636054, 11124892, 5270844, 54681, 20...
$ gdp                <dbl> NA, 1.3828e+10, NA, NA, 1.0832e+11, N...
$ continent          <fct> Europe, Africa, Africa, Americas, Ame...
$ region             <fct> Southern Europe, Northern Africa, Mid...
```

- Also check out the `{gapminder}` package

What is data wrangling?

- "data janitor work"
- importing data
- cleaning data
- changing shape of data
- fixing errors and poorly formatted data elements
- transforming columns and rows
- filtering, subsetting



Base R versus the {tidyverse}

- The mean life expectancy across all years for Asia

```
# Base R
asia <- gapminder[gapminder$continent == "Asia", ]
mean(asia$life_expectancy)
```

```
[1] 65.901
```

```
library(dplyr)
gapminder %>%
  filter(continent == "Asia") %>%
  summarize(mean_exp = mean(life_expectancy))
```

```
# A tibble: 1 x 1
  mean_exp
  <dbl>
1      65.9
```

The pipe %>%



- A way to chain together commands
- Can be read as "and then" when reading over code

```
library(dplyr)
gapminder %>%
  filter(continent == "Asia") %>%
  summarize(mean_exp = mean(life_expectancy))
```

`filter()` rows that satisfy specified conditions

dplyr:: filter()

KEEP ROWS THAT
satisfy
your CONDITIONS

keep rows from... this data... ONLY IF... type is "otter" AND site is "bay"

```
filter(df, type == "otter" & site == "bay")
```

type	food	site
otter	urchin	bay
Shark	seal	channel
otter	abalone	bay
otter	crab	wharf

@allison_horst

dplyr::filter()

- Arguments are "filters" that you'd like to apply.

```
gap_2014 <- gapminder %>% filter(year == 2014)  
gap_2014
```

```
# A tibble: 185 x 9  
  country      year infant_mortality life_expectancy  
  <fct>        <int>            <dbl>              <dbl>  
1 Albania      2014             12.9               77.9  
2 Algeria      2014              22                 76.3  
3 Angola       2014             98.8               59.2  
4 Antigua and Barbuda 2014          6.1                76.3  
5 Argentina    2014             11.5               76.3  
# ... with 180 more rows, and 5 more variables: fertility <dbl>,  
#   population <dbl>, gdp <dbl>, continent <fct>, region <fct>
```

- Use `==` to compare a variable to a value

Logical operators

- Use `|` to check for any in multiple filters being true:

```
gapminder %>%
  filter(life_expectancy < 50 | fertility > 4) %>%
  slice_sample(n = 8)
```

```
# A tibble: 8 x 9
  country      year infant_mortality life_expectancy fertility
  <fct>        <int>            <dbl>              <dbl>      <dbl>
1 Nepal        1996             73.1               62.5      4.6 
2 Uganda       2010             49.5               57.8      6.16 
3 Nicaragua    1988             54.5               71.4      4.99 
4 Guinea-Bissau 1962            NA                 43.6      5.67 
5 Oman          1985             50.4               68.4      8.22 
6 Comoros       1970             152.                48.6      7.06 
7 Tajikistan    1980             100.                64.3      5.66 
8 Gabon         1990             60.5               59.5      5.42 
# ... with 4 more variables: population <dbl>, gdp <dbl>,
#   continent <fct>, region <fct>
```

Logical operators

- Use `,` to check for all of multiple filters being true:

```
gapminder %>%
  filter(life_expectancy < 50, fertility > 4)
```

```
# A tibble: 8 x 9
  country      year infant_mortality life_expectancy fertility
  <fct>       <int>           <dbl>            <dbl>      <dbl>
1 Algeria     1960            148.             47.5      7.65
2 Angola      1960            208              36.0      7.32
3 Bangladesh  1960            176.             46.2      6.73
4 Benin        1960            187.             38.3      6.28
5 Bhutan       1960            175              35.9      6.67
6 Bolivia      1960            173.             43.8      6.7 
7 Burkina Faso 1960            161.             35.2      6.29
8 Burundi      1960            145.             40.6      6.95
# ... with 4 more variables: population <dbl>, gdp <dbl>,
#   continent <fct>, region <fct>
```

Logical operators

- Use `%in%` to check for any being true
(shortcut to using `|` repeatedly with `==`)

```
gapminder %>%
  filter(country %in% c("Argentina", "Belgium", "Mexico"),
         year %in% c(2012, 2015))
```

```
# A tibble: 6 x 9
  country     year infant_mortality life_expectancy fertility
  <fct>      <int>            <dbl>              <dbl>      <dbl>
1 Argentina   2012             12.3               76.1      2.19
2 Belgium     2012              3.5                80.3      1.85
3 Mexico       2012             13.1               75.7      2.22
4 Argentina   2015             11.1               76.5      2.15
5 Belgium     2015              3.3                80.5      1.86
6 Mexico       2015             11.3               75.9      2.13
# ... with 4 more variables: population <dbl>, gdp <dbl>,
#   continent <fct>, region <fct>
```

Your Turn

Exercise A - 5 minutes

1. Filter observations from either Europe or Africa using the `|`.
2. Filter observations from either Africa or Asia using `%in%`.
3. How many countries had life expectancy greater than 80 years in 1996?

05 : 00

Walk through exercise in RStudio

summarize()

- Any numerical summary that you want to apply to a column of a data frame is specified within `summarize()`.

```
stats_2015 <- gapminder %>%
  filter(year == 2015) %>%
  summarize(
    max_exp = max(life_expectancy),
    sd_exp = sd(life_expectancy)
  )
stats_2015
```

```
# A tibble: 1 x 2
  max_exp  sd_exp
  <dbl>    <dbl>
1     83.7    7.79
```

Combining `summarize()` with `group_by()`

- When you'd like to determine a numerical summary for all levels of a different categorical variable

```
max_exp_2015_by_cont <- gapminder %>%
  filter(year == 2015) %>%
  group_by(continent) %>%
  summarize(max_exp = max(life_expectancy),
            sd_exp = sd(life_expectancy))
max_exp_2015_by_cont
```

```
# A tibble: 5 x 3
  continent max_exp sd_exp
  * <fct>     <dbl>   <dbl>
1 Africa      77.6    6.52
2 Americas    81.7    3.58
3 Asia        83.7    5.12
4 Europe      83.3    3.43
5 Oceania     82.3    7.32
```

`mutate()` changes the data



mutate()

- Most importantly, allows you to create a new variable based on other variables

```
gapminder_plus <- gapminder %>%
  mutate(gdp_per_capita = gdp / population)
slice_sample(gapminder_plus, n = 4)
```

```
# A tibble: 4 x 10
  country      year infant_mortality life_expectancy fertility
  <fct>       <int>            <dbl>             <dbl>        <dbl>
1 Portugal     1972             47.6              69.2        2.88
2 Mauritius    1987             24.4              68.5        2.27
3 South Africa 1994             47.2              62.8        3.2 
4 Tonga        1982              23                67.6        5.48
# ... with 5 more variables: population <dbl>, gdp <dbl>,
#   continent <fct>, region <fct>, gdp_per_capita <dbl>
```

“**TIDY DATA** is a standard way of mapping the meaning of a dataset to its structure.”

—HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

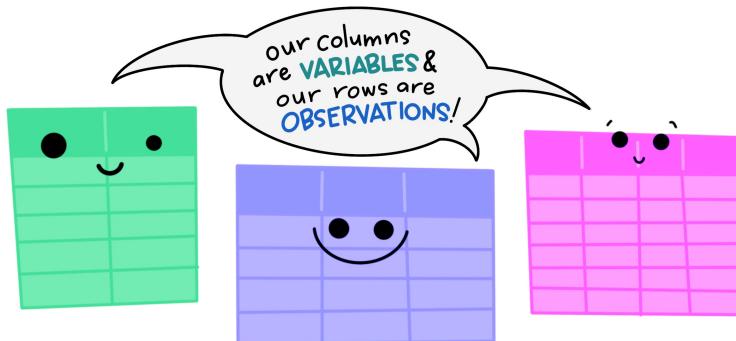
each column a variable

each row an observation

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

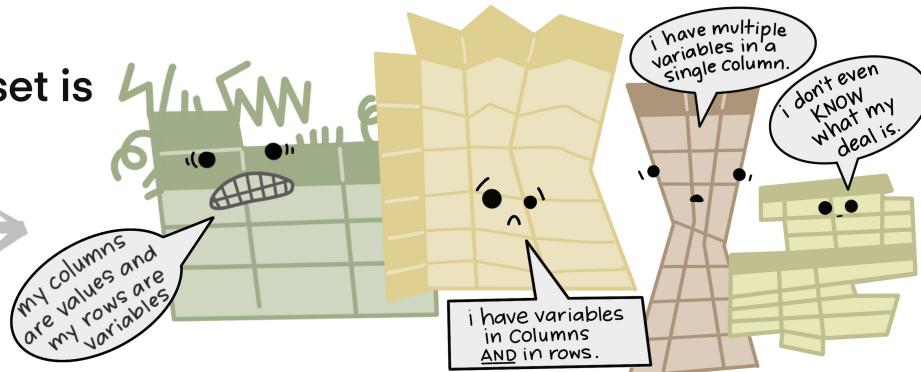
Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

The standard structure of
tidy data means that
“tidy datasets are all alike...”



“...but every messy dataset is
messy in its own way.”

-HADLEY WICKHAM



arrange()

- Reorders the rows in a data frame based on the values of one or more variables

```
gapminder_plus %>%  
  arrange(year, country)
```

```
# A tibble: 10,545 x 10
  country             year infant_mortality life_expectancy
  <fct>            <int>          <dbl>           <dbl>
1 Albania           1960         115.            62.9
2 Algeria           1960         148.            47.5
3 Angola            1960         208              36.0
4 Antigua and Barbuda 1960        NA              63.0
5 Argentina         1960         59.9            65.4
# ... with 10,540 more rows, and 6 more variables:
#   fertility <dbl>, population <dbl>, gdp <dbl>,
#   continent <fct>, region <fct>, gdp_per_capita <dbl>
```

arrange()

- Can also put into descending order

```
gapminder_plus %>%
  filter(year > 2000) %>%
  arrange(desc(life_expectancy))
```

```
# A tibble: 2,960 x 10
  country             year infant_mortality life_expectancy
  <fct>              <int>            <dbl>            <dbl>
1 Hong Kong, China   2016              NA               83.9
2 Hong Kong, China   2015              NA               83.7
3 Hong Kong, China   2014              NA               83.6
4 Hong Kong, China   2013              NA               83.4
5 Iceland            2014              1.6               83.3
# ... with 2,955 more rows, and 6 more variables: fertility <dbl>,
#   population <dbl>, gdp <dbl>, continent <fct>, region <fct>,
#   gdp_per_capita <dbl>
```

Don't mix up `arrange()` and `group_by()`

- `group_by()` is used (mostly) with `summarize()` to calculate summaries over groups
- `arrange()` is used for sorting

Don't mix up `arrange()` and `group_by()`

This doesn't really do anything useful by itself

```
gapminder %>% group_by(country)
```

```
# A tibble: 10,545 x 9
# Groups:   country [185]
  country      year infant_mortality life_expectancy
  <fct>       <int>            <dbl>             <dbl>
1 Albania     1960            115.              62.9
2 Algeria     1960            148.              47.5
3 Angola      1960            208               36.0
4 Antigua and Barbuda 1960          NA              63.0
5 Argentina    1960            59.9              65.4
# ... with 10,540 more rows, and 5 more variables:
#   fertility <dbl>, population <dbl>, gdp <dbl>,
#   continent <fct>, region <fct>
```

Don't mix up `arrange()` and `group_by()`

But this does

```
gapminder %>% arrange(country)
```

```
# A tibble: 10,545 x 9
  country   year infant_mortality life_expectancy fertility
  <fct>     <int>            <dbl>              <dbl>        <dbl>
1 Albania    1960            115.               62.9       6.19
2 Albania    1961            111.               63.9       6.08
3 Albania    1962            106.               64.8       5.96
4 Albania    1963            102.               65.6       5.83
5 Albania    1964             97.9               66.2       5.71
# ... with 10,540 more rows, and 4 more variables:
#   population <dbl>, gdp <dbl>, continent <fct>, region <fct>
```

select()

- Chooses a subset of *columns* (don't mix up with filter())

```
gapminder_plus %>%  
  select(country, region, gdp_per_capita)
```

```
# A tibble: 10,545 x 3  
  country           region      gdp_per_capita  
  <fct>            <fct>          <dbl>  
1 Albania          Southern Europe     NA  
2 Algeria          Northern Africa   1243.  
3 Angola           Middle Africa    NA  
4 Antigua and Barbuda Caribbean     NA  
5 Argentina        South America   5254.  
# ... with 10,540 more rows
```

Your Turn

Exercise B - 5 minutes

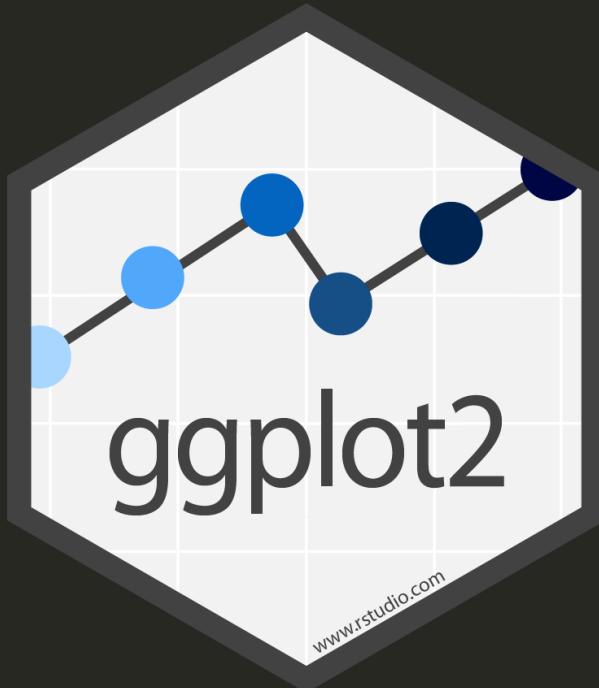
1. Create a data frame that has the median life expectancy by year.
2. Use `arrange()` on this data frame to see the top years with the highest median life expectancy. What about the year with the lowest median life expectancy?
3. Create a new column that is median life expectancy rounded to the nearest integer, and remove the original un-rounded column with `select()`.



05 : 00

Walk through exercise in RStudio

Basics of {ggplot2}



Alison Horst

ggplot2 package

59 / 141

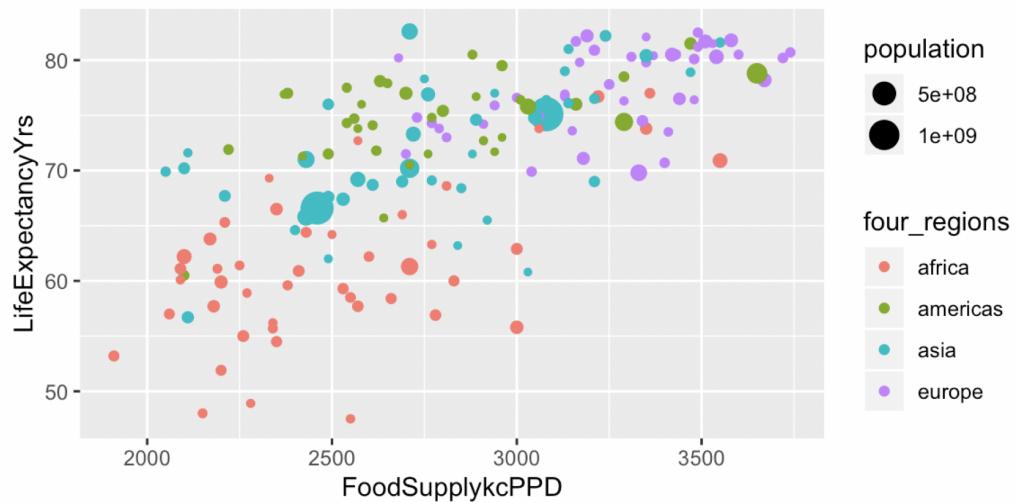
Function

Dataset

```
ggplot(data = gapminder2011,  
       aes(x = FoodSupplykcPPD, y = LifeExpectancyYrs,  
            color = four_regions, size = population)) +  
  geom_point()
```

What kind of plot to make

Which variables to plot



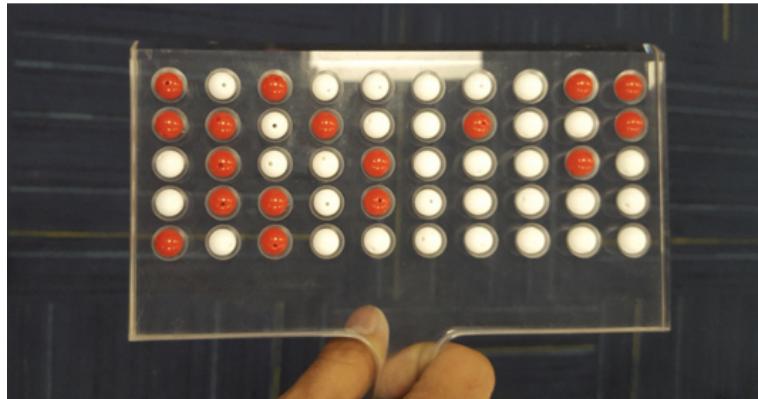
Learning objectives

Part 2

- Build a sampling distribution by repeatedly sampling from a population using the `{moderndive}` package
- Compose a visualization of a sampling distribution
- Construct a bootstrap distributions for a basic confidence interval of a statistic
- Perform tidy hypothesis testing using the `{infer}` package

Extending this {tidyverse} knowledge to something new

- How can we now learn about sampling distributions?



```
library(moderndive)
bowl %>% slice_head(n = 15)
```

```
# A tibble: 15 x 2
  ball_ID color
  <int> <chr>
1 1 white
2 2 white
3 3 white
4 4 red
5 5 white
6 6 white
7 7 red
8 8 white
9 9 red
10 10 white
11 11 white
12 12 white
13 13 white
14 14 white
15 15 red
```

One virtual scoop of 50 balls (one sample)

```
set.seed(8675309)
(jennys_sample <- bowl %>% slice_sample(n = 50))
```

```
# A tibble: 50 x 2
  ball_ID color
  <int> <chr>
1 2260 white
2 970  white
3 1291 white
4 1216 white
5 2217 red
# ... with 45 more rows
```

Proportion that are red

```
jennys_sample %>%
  summarize(prop_red = mean(color == "red")) %>%
  pull()
```

```
[1] 0.32
```

Is this how many are in the full bowl?

Sampling variability

What does `rep_bowl_samples` look like?

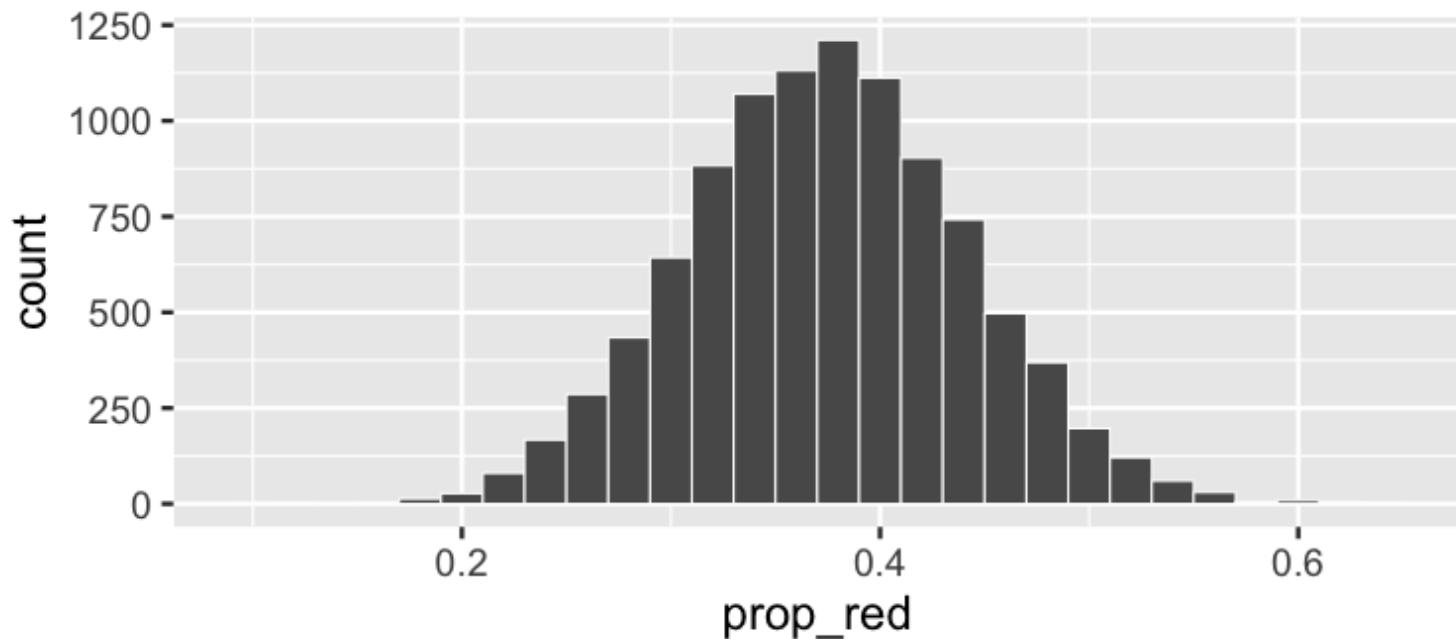
```
library(moderndive)
library(infer)
rep_bowl_samples <- bowl %>%
  rep_slice_sample(n = 50, reps = 10000)
```

How about `bowl_props`?

```
bowl_props <- rep_bowl_samples %>%
  group_by(replicate) %>%
  summarize(prop_red = mean(color == "red"))
```

The sampling distribution

```
ggplot(data = bowl_props, mapping = aes(x = prop_red)) +  
  geom_histogram(binwidth = 0.02, color = "white")
```



Shifting focus

What about if all we had was the one sample of balls (not the whole bowl)?

```
jennys_sample %>% count(color)
```

```
# A tibble: 2 x 2
  color     n
  <chr> <int>
1 red      16
2 white    34
```

How could we use this sample to make a guess about the sampling variability from other samples?

Building up to statistical inference!

```
library(infer)
jennys_sample %>%
  specify(formula = color ~ NULL, success = "red")
```

```
Response: color (factor)
# A tibble: 50 x 1
  color
  <fct>
  1 white
  2 white
  3 white
  4 white
  5 red
# ... with 45 more rows
```

Bootstrapping?

```
library(infer)
(bootstrap_samples <- jennys_sample %>%
  specify(formula = color ~ NULL, success = "red") %>%
  generate(reps = 48, type = "bootstrap"))
```

```
Response: color (factor)
# A tibble: 2,400 x 2
# Groups:   replicate [48]
  replicate color
  <int> <fct>
1       1 red
2       1 white
3       1 white
4       1 white
5       1 white
# ... with 2,395 more rows
```

What does `bootstrap_samples` represent?

- Remember we assumed that all we had was the original sample of 19 red and 31 white to start.
- Hope each selection in `bootstrap_samples` is similar to:



Bootstrap statistics

```
jennys_sample %>%  
  specify(formula = color ~ NULL, success = "red") %>%  
  generate(reps = 48, type = "bootstrap") %>%  
  calculate(stat = "prop")
```

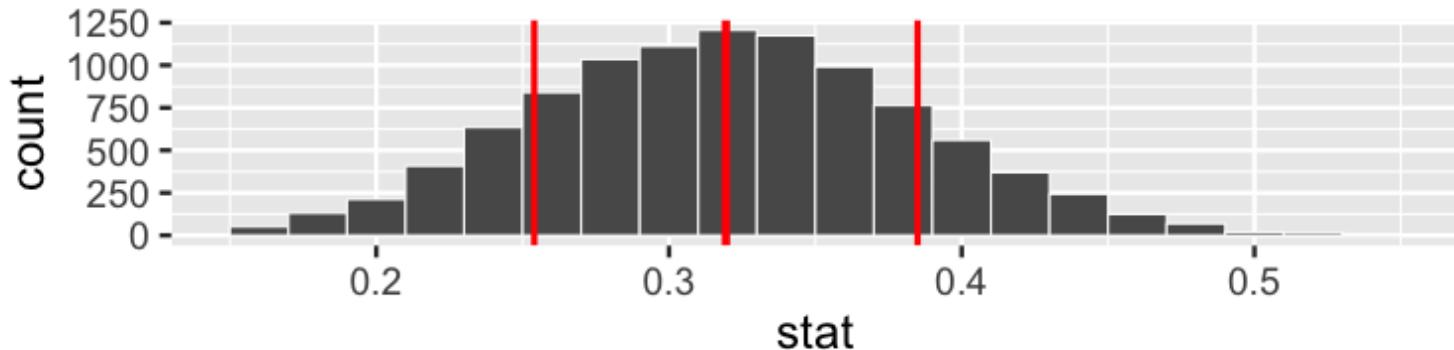
```
# A tibble: 48 x 2  
  replicate  stat  
* <int> <dbl>  
1 1 0.24  
2 2 0.34  
3 3 0.34  
4 4 0.4  
5 5 0.28  
# ... with 43 more rows
```

Do 10,000 reps to get a better sense for variability

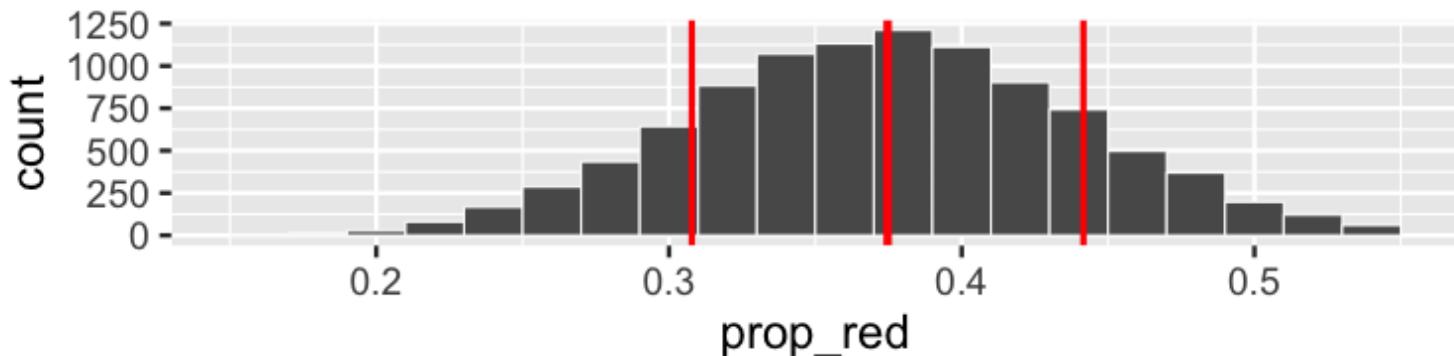
Just as we did with the sampling distribution

```
bootstrap_stats <- jennys_sample %>%
  specify(formula = color ~ NULL, success = "red") %>%
  generate(reps = 10000, type = "bootstrap") %>%
  calculate(stat = "prop")
```

The bootstrap distribution



The sampling distribution



Get a confidence interval

```
get_ci(bootstrap_stats, level = 0.95)
```

```
# A tibble: 1 x 2
  lower_ci upper_ci
  <dbl>    <dbl>
1     0.2     0.44
```

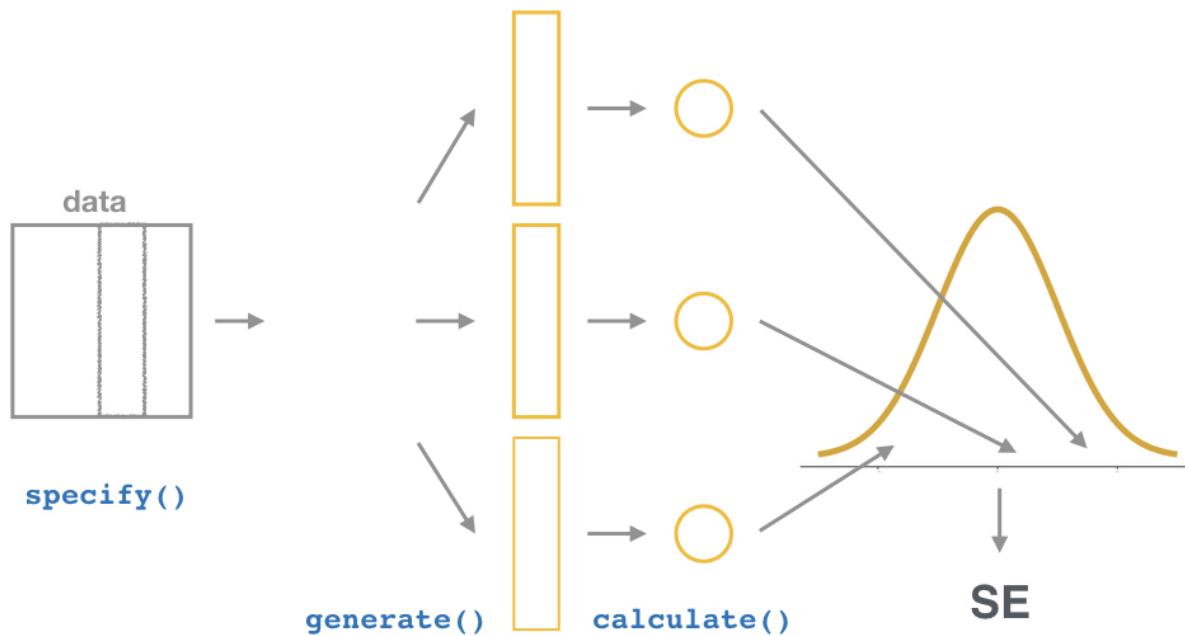
- We are 95% "confident" the true proportion of red balls in the hopper/bowl is between 0.2 and 0.44.
- In the population:

```
mean(bowl$color == "red")
```

```
[1] 0.375
```

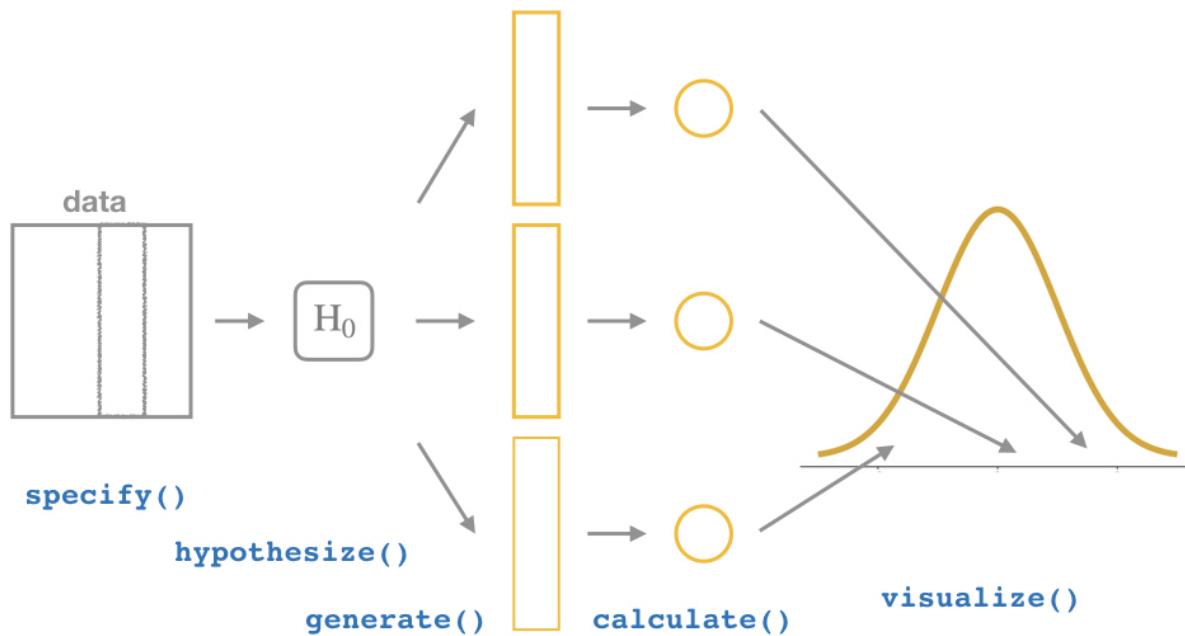
{infer} verbs

Confidence Interval



{infer} verbs

Hypothesis test



Statistical Inference



{infer} hex sticker designs kindly created by [Thomas Mock](#) 84 / 141

Research Question

If you see someone else yawn, are you more likely to yawn? In an episode of the show *Mythbusters*, they tested the myth that yawning is contagious.

- 50 adults who thought they were being considered for an appearance on the show.
- Each participant was interviewed individually by a show recruiter ("confederate") who either yawned or did not.
- Participants then sat by themselves in a large van and were asked to wait.
- While in the van, the Mythbusters watched to see if the unaware participants yawned.

Data

- group
 - 34 saw the confederate yawn (seed)
 - 16 did not see the confederate yawn (control)
- yawn
 - yes - participant yawned
 - no - participant did not yawn

```
library(moderndive)
mythbusters_yawn %>% slice(c(1, 3, 6, 19))
```

```
# A tibble: 4 x 3
  subj group  yawn
  <int> <chr> <chr>
1     1 seed    yes
2     3 seed    no 
3     6 control no 
4    19 control yes
```

Results



```
library(janitor)
mythbusters_yawn %>%
  tabyl(group, yawn) %>%
  adorn_percentages() %>%
  adorn_pct_formatting() %>%
  adorn_ns()
```

group	no	yes
control	75.0% (12)	25.0% (4)
seed	70.6% (24)	29.4% (10)

Finding: CONFIRMED

"Though that's not an enormous increase, since they tested 50 people in the field, the gap was still wide enough for the MythBusters to confirm that yawning is indeed contagious."¹

[1] <http://www.discovery.com/tv-shows/mythbusters/mythbusters-database/yawning-contagious/>

Really? Let's formally check this

- State the hypotheses

Null hypothesis:

There is no difference between the seed and control groups in the proportion of people who yawned.

Alternative hypothesis (directional):

More people (relatively) yawned in the seed group than in the control group.

Test the hypothesis

QUIZ: Which type of hypothesis test would you conduct here?

- A. Independent samples t-test
- B. Two proportion test
- C. Chi-square Goodness of Fit
- D. Analysis of Variance

Two proportion test

$$H_0 : p_{seed} - p_{control} = 0$$

$$H_A : p_{seed} - p_{control} > 0$$

The observed difference

```
library(infer)
obs_diff <- mythbusters_yawn %>%
  specify(yawn ~ group, success = "yes") %>%
  calculate(
    stat = "diff in props",
    order = c("seed", "control")
  )
obs_diff
```

```
# A tibble: 1 x 1
  stat
  <dbl>
1 0.0441
```

Is this difference *meaningful*?

Different question:

Is this difference *significant*?

Modeling the null hypothesis

If...

$$H_0 : p_{seed} = p_{control}$$

is true, then whether or not the participant saw someone else yawn does not matter.

In other words, there is no association between exposure and yawning.



Original universe

subj	group	yawn
1	seed	yes
2	control	yes
3	seed	no
4	seed	yes
5	seed	no
6	control	no
15	seed	no
16	seed	no
17	seed	no
18	seed	no
19	control	yes
20	seed	no

group	no	yes	Total
control	12	4	16
seed	24	10	34
Total	36	14	50

Parallel universe

subj	group	alt_yawn
1	seed	no
2	control	no
3	seed	no
4	seed	no
5	seed	no
6	control	no
15	seed	no
16	seed	yes
17	seed	no
18	seed	yes
19	control	no
20	seed	yes

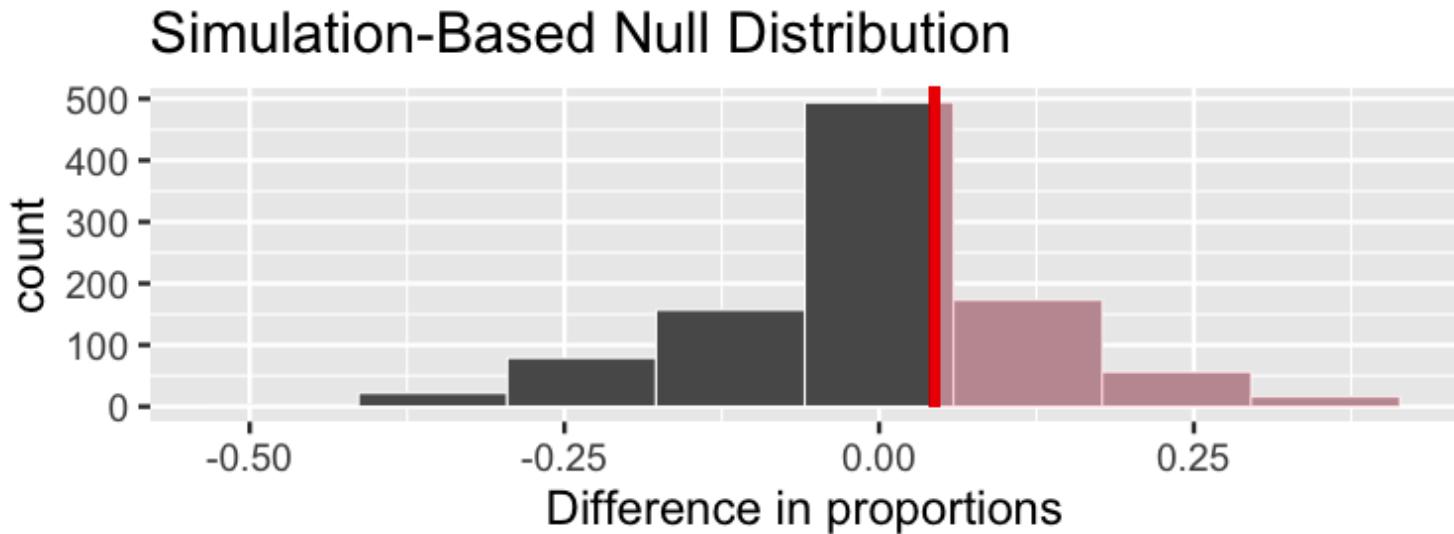
group	no	yes	Total
control	12	4	16
seed	24	10	34
Total	36	14	50

1000 parallel universes

replicate	stat
1	0.22794
2	0.41176
3	0.04412
4	0.04412
5	-0.04779
6	0.13603
7	-0.23162
8	0.22794
9	-0.04779
10	0.04412
11	0.13603
12	0.04412
13	0.04412
14	-0.04779
15	-0.13971

replicate	stat
986	0.04412
987	-0.13971
988	0.04412
989	-0.23162
990	-0.04779
991	-0.13971
992	-0.13971
993	-0.13971
994	0.04412
995	-0.13971
996	-0.23162
997	0.13603
998	-0.04779
999	0.04412
1000	0.04412

The parallel universe distribution



The distribution of 1000 differences in proportions, if the null hypothesis were *true* and yawning was not contagious.

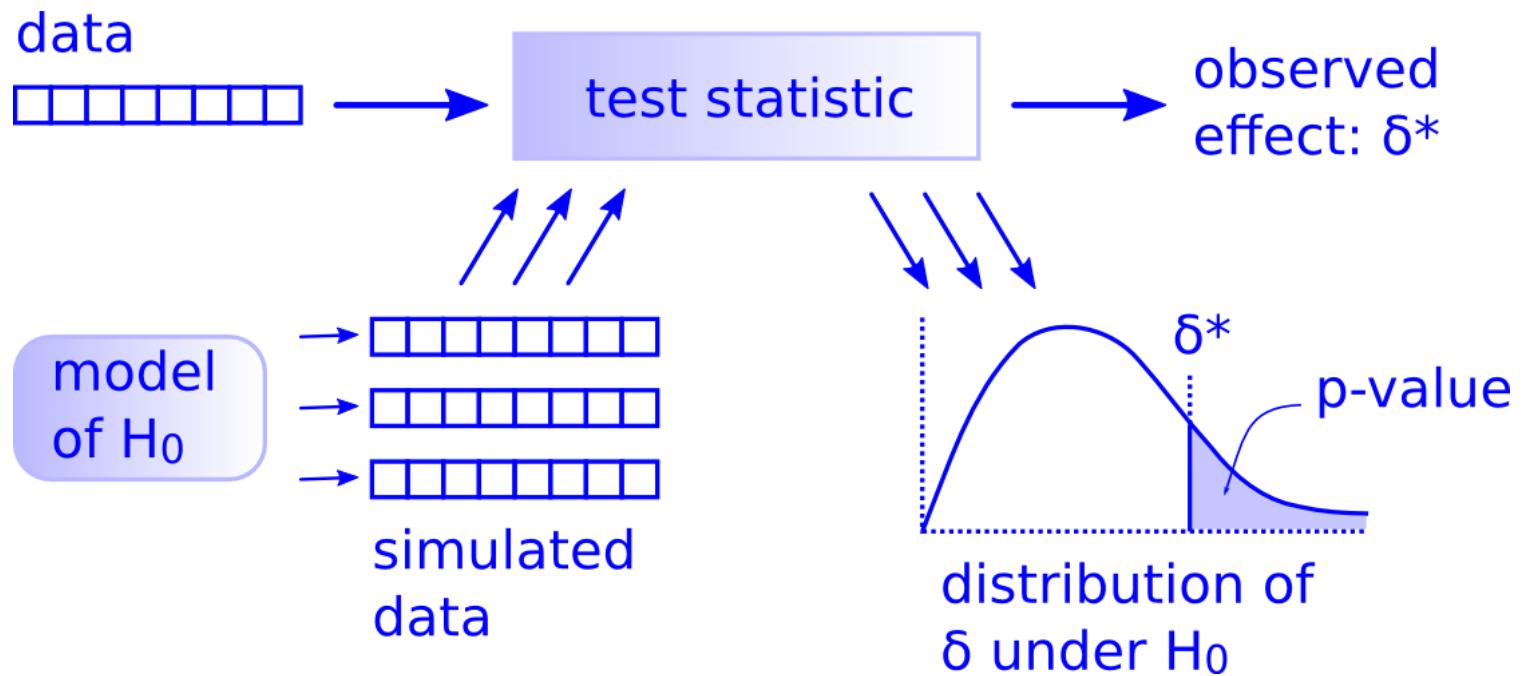
Calculating the p-value

The shaded proportion is the p-value!

```
null_distn %>%
  get_p_value(obs_stat = obs_diff, direction = "right")
```

```
# A tibble: 1 x 1
  p_value
  <dbl>
1 0.493
```

There is Only One Test!



Hypothesis test

data

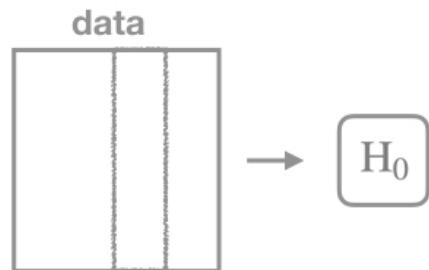
Hypothesis test

data

--	--	--

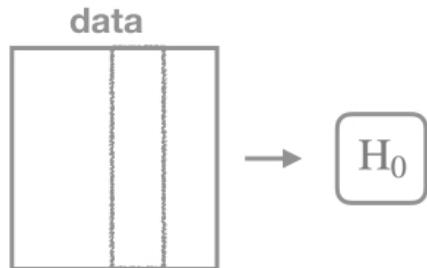
`specify()`

Hypothesis test



`specify()`

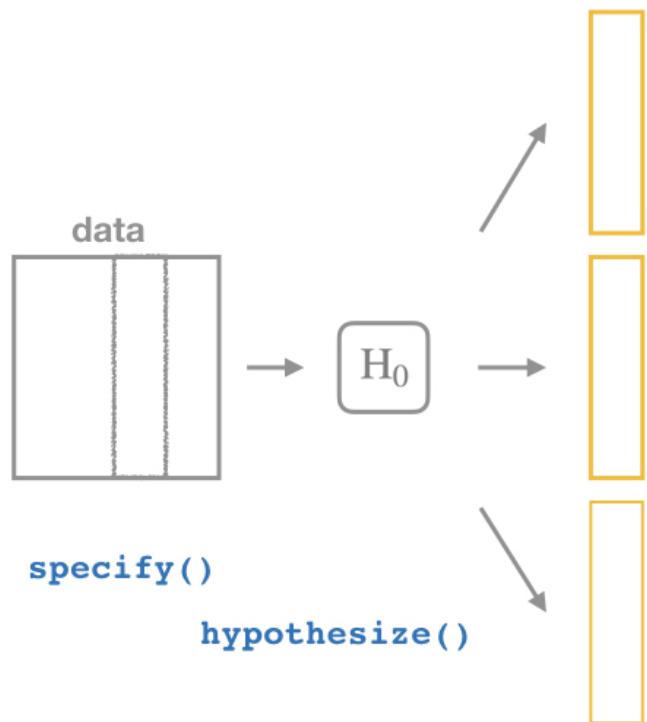
Hypothesis test



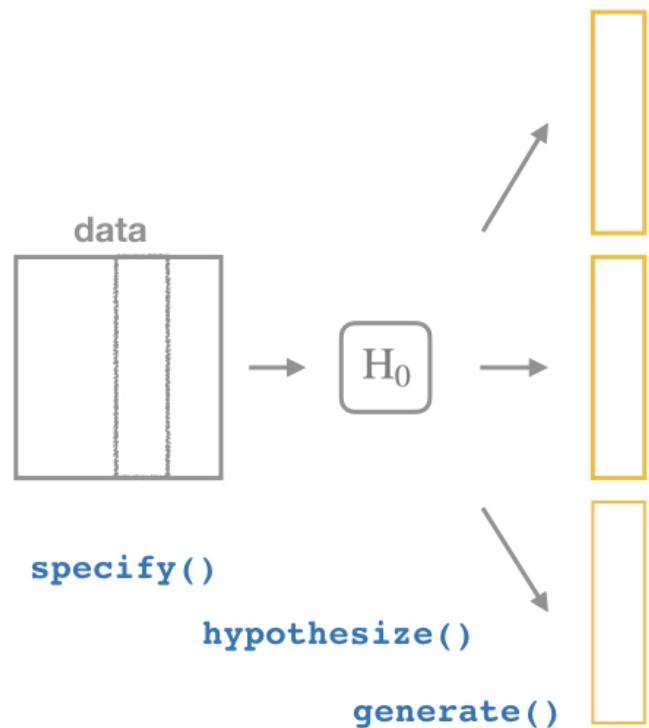
`specify()`

`hypothesize()`

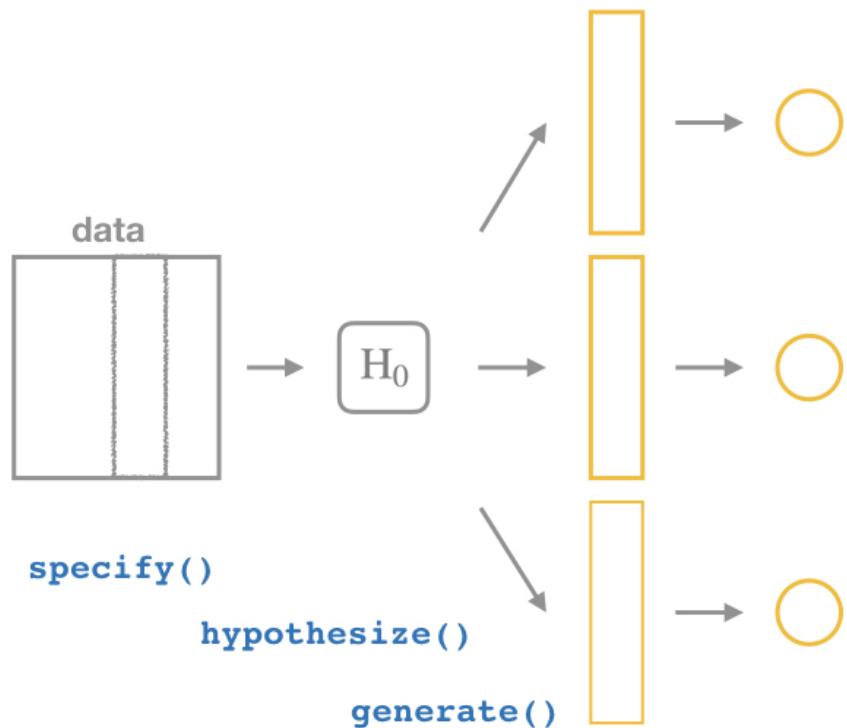
Hypothesis test



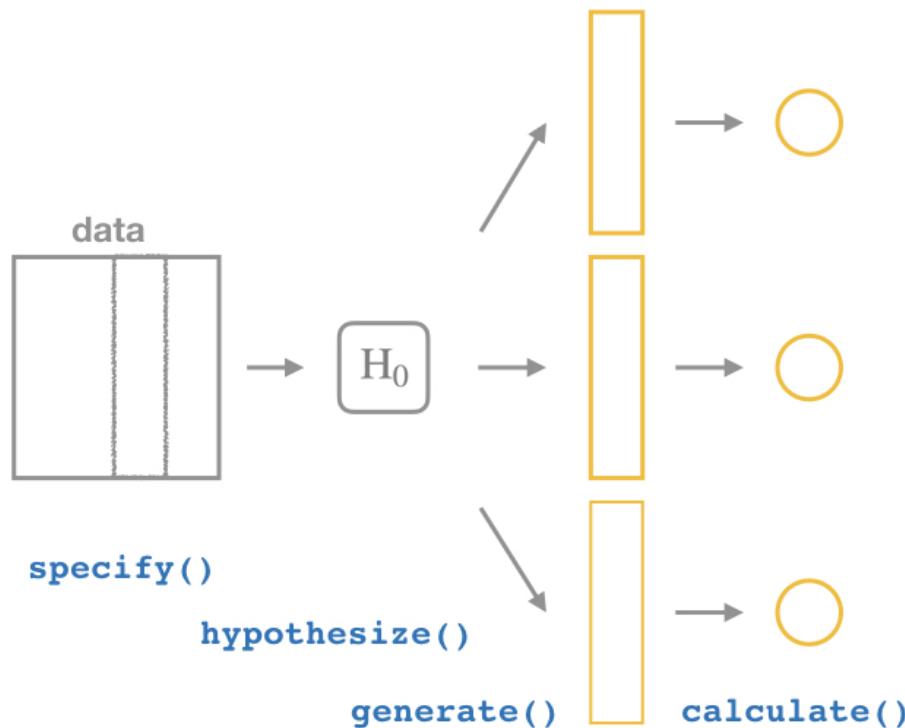
Hypothesis test



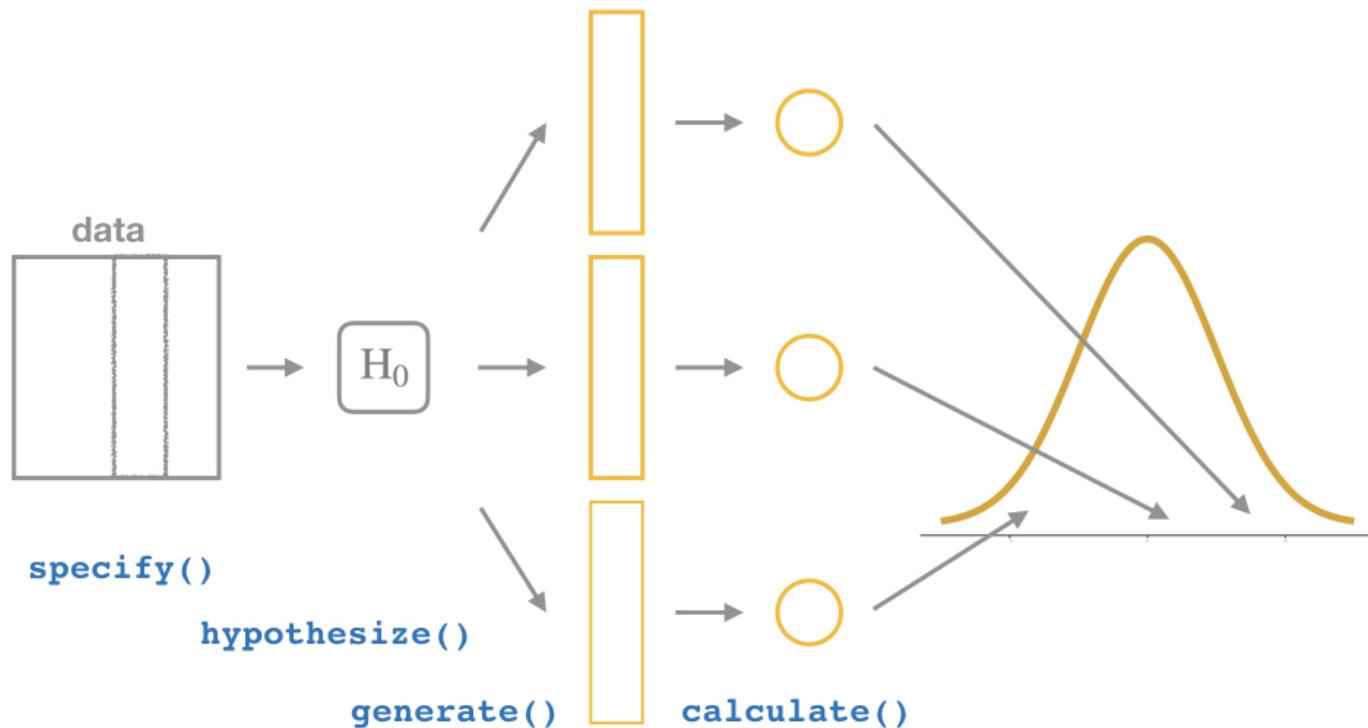
Hypothesis test



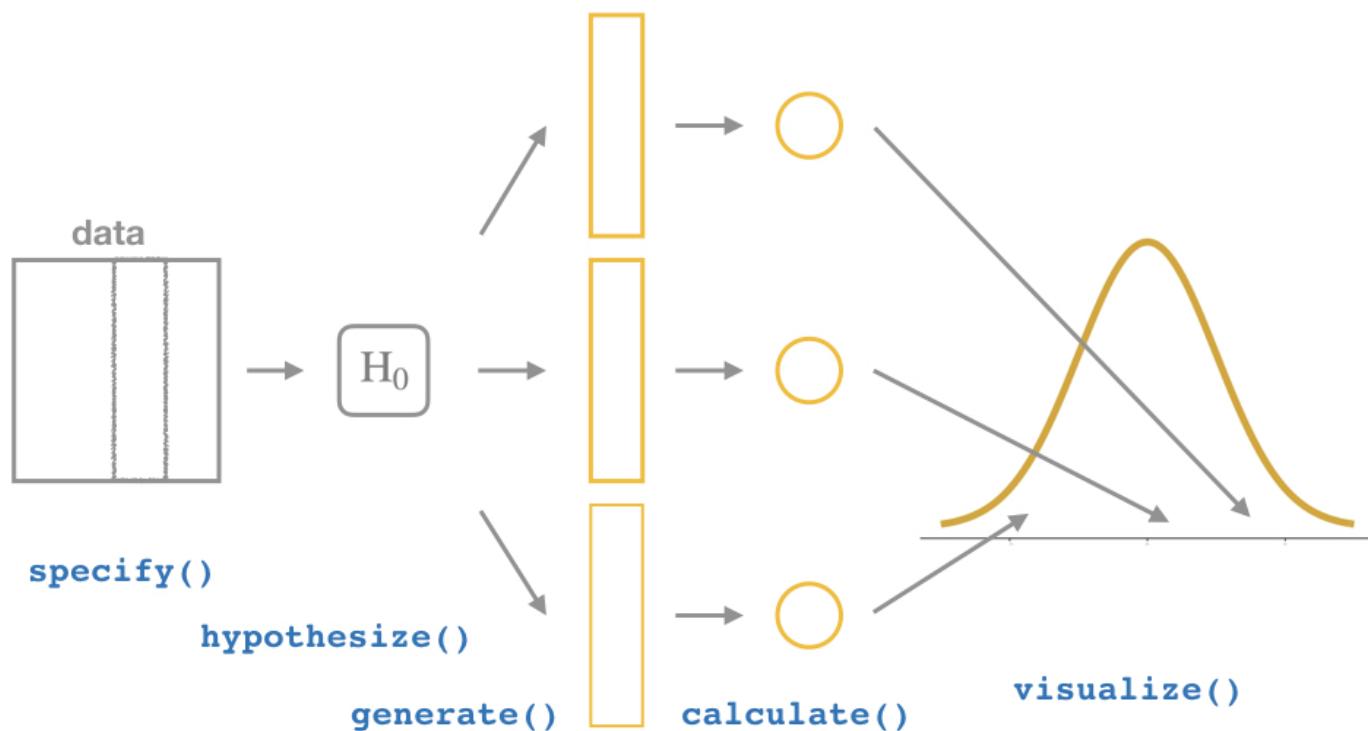
Hypothesis test



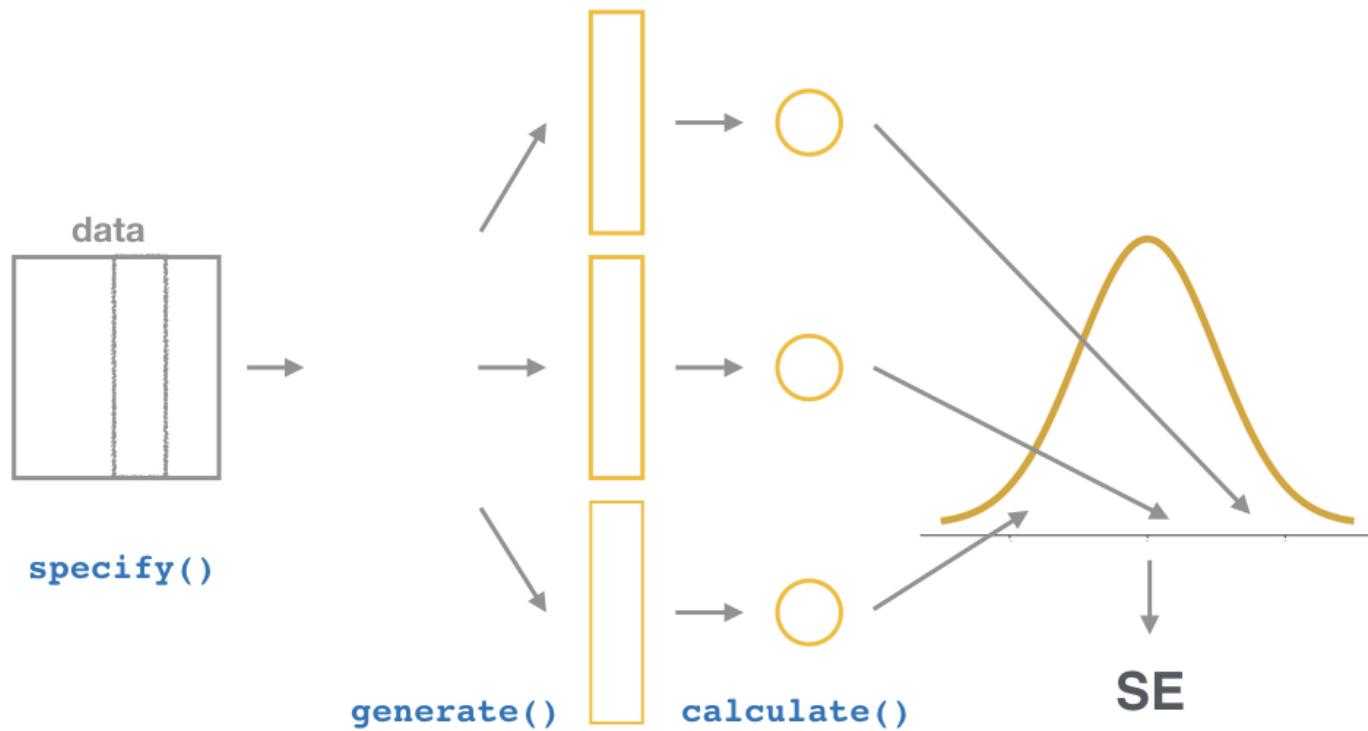
Hypothesis test



Hypothesis test



Confidence Interval



{infer}

Five main functions:

- `specify()`
- `hypothesize()`
- `generate()`
- `calculate()`
- `visualize()`

{infer}

- `specify()` the response and explanatory variables (`y ~ x`)
- `hypothesize()` what the null hypothesis is (here, independence of `y` and `x`)
- `generate()` new samples from parallel universes under the null hypothesis model:
 - Resample from our original data without replacement, each time shuffling the `group` (`type = "permute"`)
 - Do this a ton of times (`reps = 1000`)
- `calculate()` the statistic (`stat = "diff in props"`) for each of the `reps`

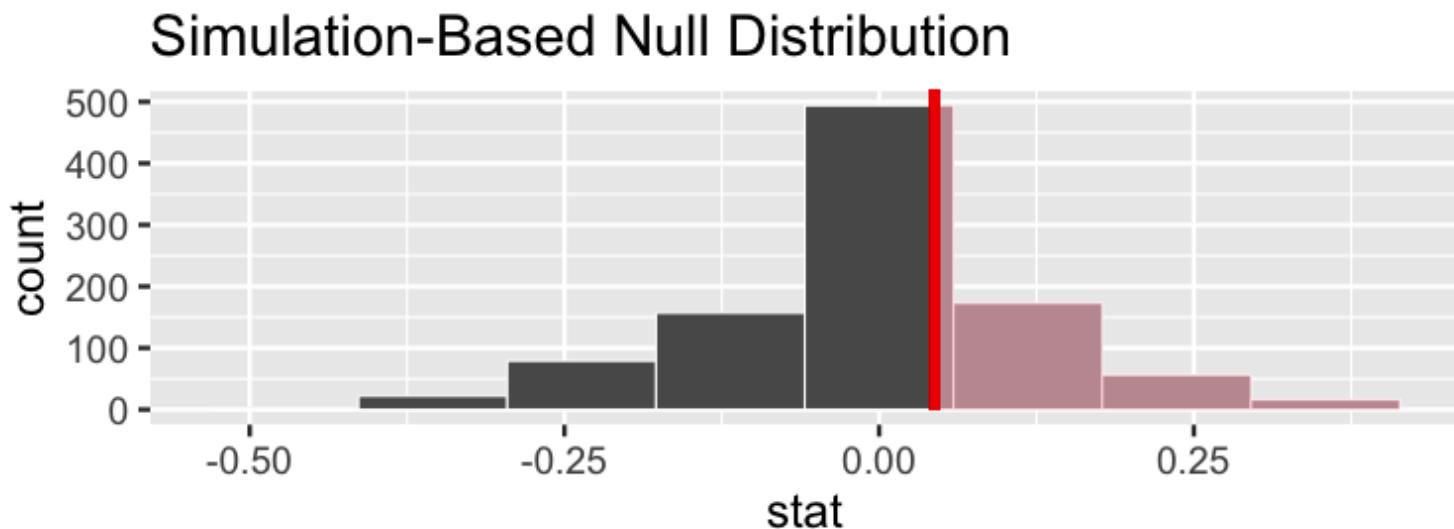
{infer} example

```
set.seed(8)
null_distn <- mythbusters_yawn %>%
  specify(
    formula = yawn ~ group,
    success = "yes"
  ) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(
    stat = "diff in props",
    order = c("seed", "control")
  )
```

Visualize the null distribution

- `visualize()` the distribution of the `stat` (here, `diff in props`)

```
null_distn %>%
  visualize(bins = 8) +
  shade_p_value(obs_stat = obs_diff, direction = "right")
```



Classical inference

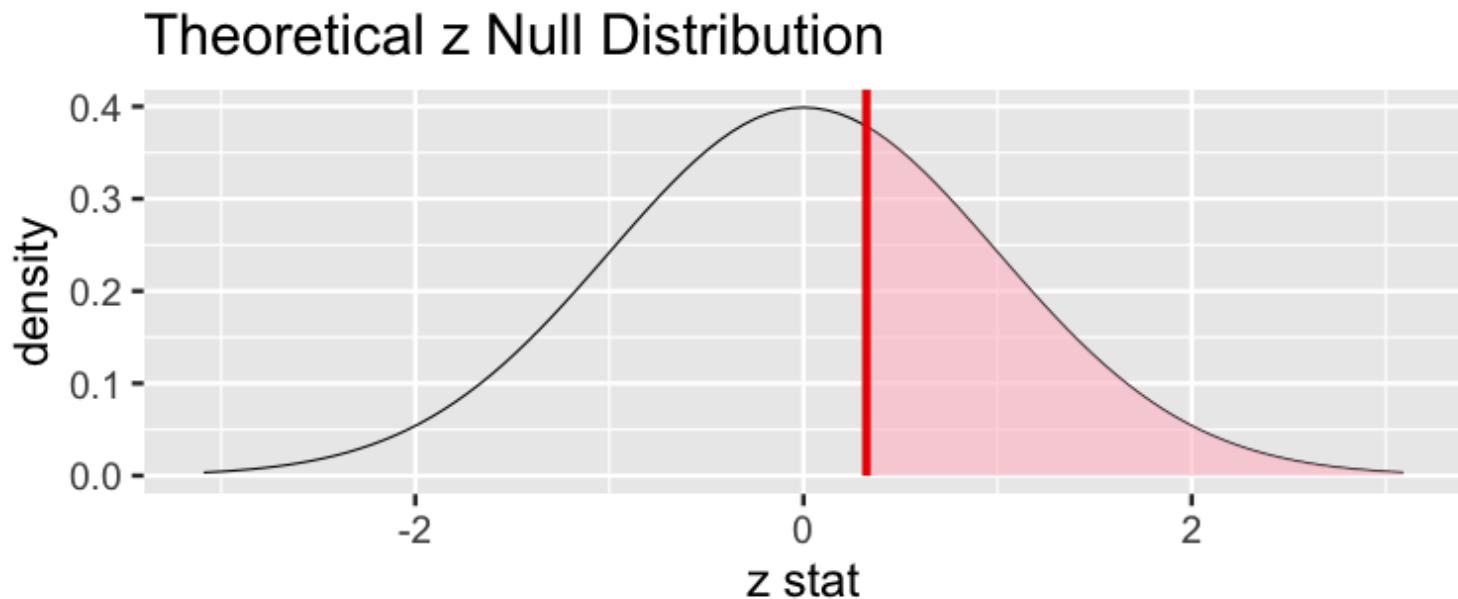
Rely on theory to tell us what the null distribution looks like.

```
obs_z <- mythbusters_yawn %>%
  specify(yawn ~ group, success = "yes") %>%
  calculate(stat = "z", order = c("seed", "control"))

mythbusters_yawn %>%
  specify(yawn ~ group, success = "yes") %>%
  hypothesize(null = "independence") %>%
  # generate() is not needed since we are not simulating
  calculate(stat = "z", order = c("seed", "control")) %>%
  visualize(method = "theoretical")
  shade_p_value(obs_stat = obs_z, direction = "right")
```

Classical inference

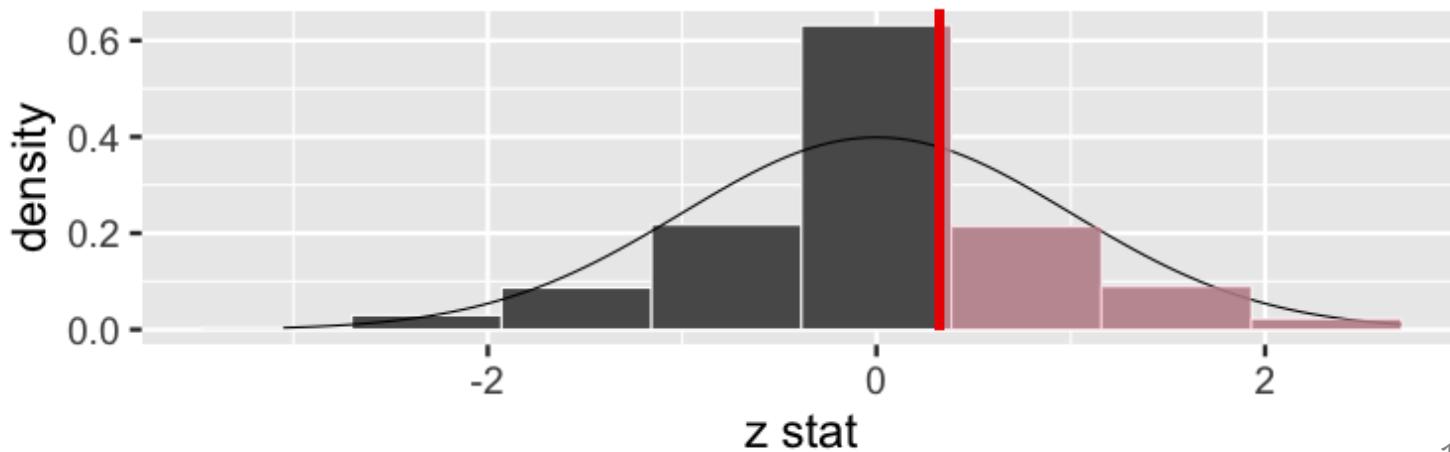
Warning: Check to make sure the conditions have been met **for** the theoretical method. `{infer}` currently does not check these **for** you.



Simulation-based vs Classical (`stat = "z"`)

```
mythbusters_yawn %>%
  specify(yawn ~ group, success = "yes") %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "z", order = c("seed", "control")) %>%
  visualize(method = "both", bins = 8) +
  shade_p_value(obs_stat = obs_z, direction = "right")
```

Simulation-Based and Theoretical z Null Distributions



More info and resources

- <https://infer.tidymodels.org/>
 - Many examples under Articles
 - Discussed in www.ModernDive.com
 - Sign up to the mailing list for updates
- Learn the tidyverse



Any questions?

- Slides created via the R package [xaringan](#) by Yihui Xie
- Slides' source code at <https://github.com/ismayc/talks/>
- R code from throughout the slides as an R script [here](#)

Thanks!

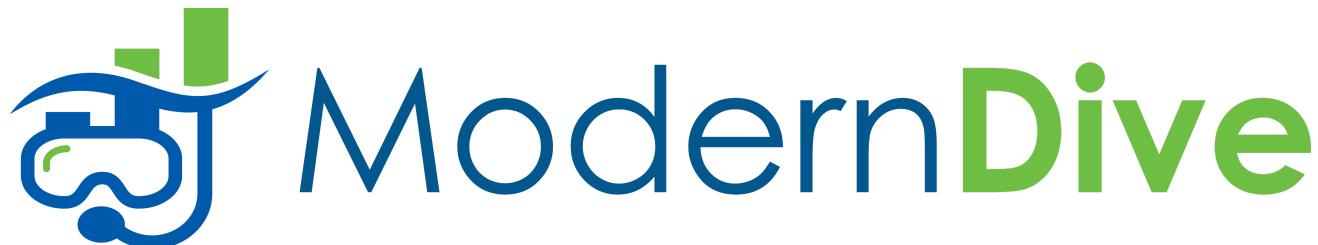


 [@datapointier](https://twitter.com/datapointier)
 [@jminnier](https://github.com/jminnier)
 jessicaminnier.com

 [@old_man_chester](https://twitter.com/old_man_chester)
 [@ismayc](https://github.com/ismayc)
 chester.rbind.io

Appendix

Freely available information



Statistical Inference via Data Science A ModernDive into R and the Tidyverse

- Webpage: <https://moderndive.com>
- Developmental version: <https://moderndive.netlify.app>
- [GitHub Repo](#)
- Please [signup](#) for our mailing list!

Good practices in RStudio

Use projects ([read this](#))

- Create an RStudio project for each data analysis project
- A project is associated with a directory folder
 - Keep data files there
 - Keep scripts there; edit them, run them in bits or as a whole
 - Save your outputs (plots and cleaned data) there
- Only use relative paths, never absolute paths
 - relative (good): `read_csv("data/mydata.csv")`
 - absolute (bad):
`read_csv("/home/yourname/Documents/stuff/mydata.csv")`

Advantages of using RStudio projects

- standardize file paths
- keep everything together
- a whole folder can be shared and run on another computer

Useful keyboard shortcuts

action	mac	windows/linux
run code in script	cmd + enter	ctrl + enter
<-	option + -	alt + -
%>%	cmd + shift + m	ctrl + shift + m

Try typing (with shortcut)
and running

```
y <- 5  
y
```

Now, in the console, press
the up arrow.

Other keyboard shortcuts: ([see full list](#))

action	mac	windows/linux
interrupt currently executing command	esc	esc
in console, go to previously run code	up/down	up/down
keyboard shortcut help	option + shift + k	alt + shift + k