

Import Data

Jake Thompson

 wjakethompson.com

 /  @wjakethompson



Incredibly boring, or...

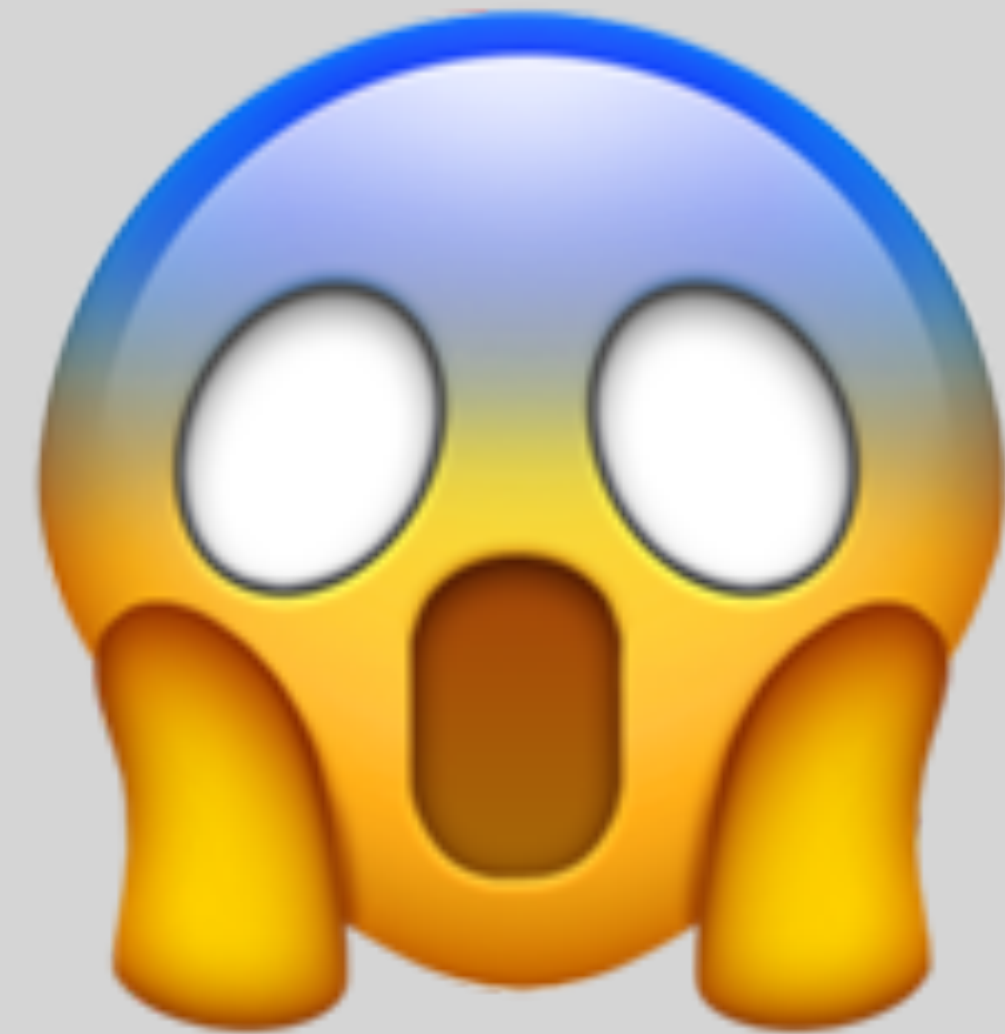


absolutely infuriating.



Your Turn 0

- Open **04-Import.Rmd**
- Run the setup chunk

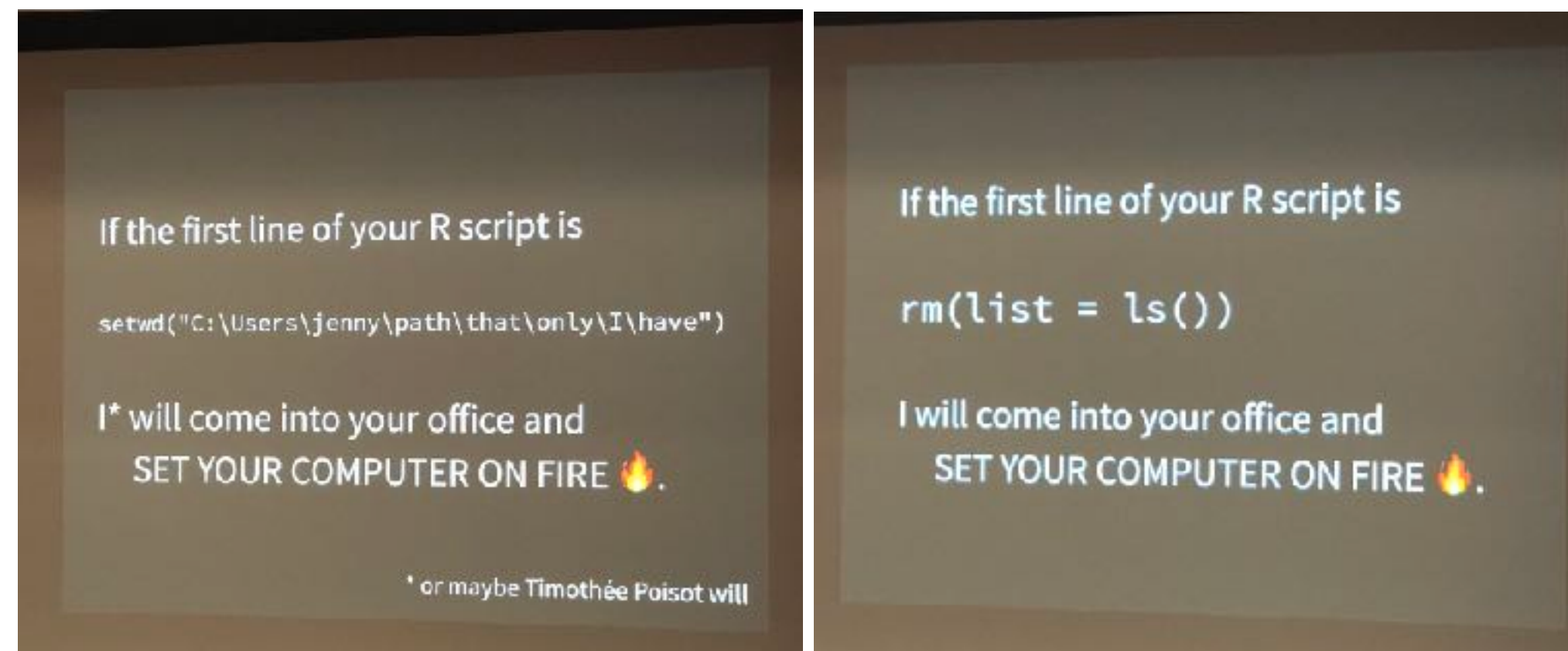


01:00



Hadley Wickham 
@hadleywickham

The only two things that make @JennyBryan 🤔😡🤯. Instead use projects + `here::here()` #rstats



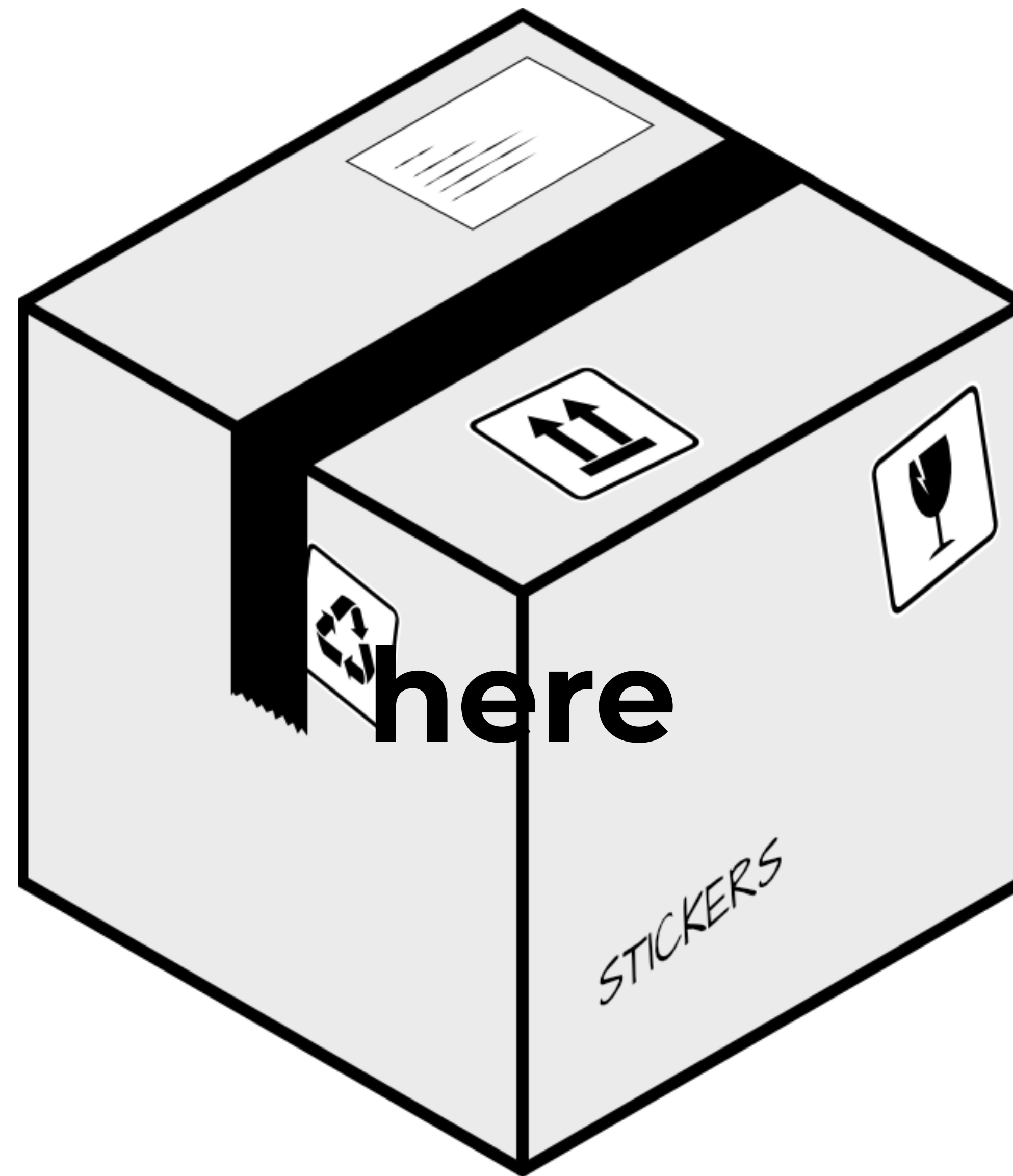
6:50 pm - 10 Dec 2017

Be kind to your collaborators

- Workflow
 - Editor
 - Home directory
 - R code you ran before break
- Product
 - Raw data
 - R code someone else needs to run to replicate results
- **Workflows should not be hardwired into the products**

Projects

- Each analysis as a *project*
 - Folder on computer with all relevant files
- R scripts written with assumption of:
 1. Clean session
 2. Working directory = project directory
- Creates everything it needs, touches nothing it didn't create





In R4DS
Workflow: Projects

here()

Find the project directory and build file paths

```
library(here)
# here() starts at /Users/w449t405/Documents/GIT/courses/tidyds-2019

here()
# [1] "/Users/w449t405/Documents/GIT/courses/tidyds-2019"

here("data", "nimbus.csv")
# [1] "/Users/w449t405/Documents/GIT/courses/tidyds-2019/data/nimbus.csv"
```


here()

Where does **here()** start?

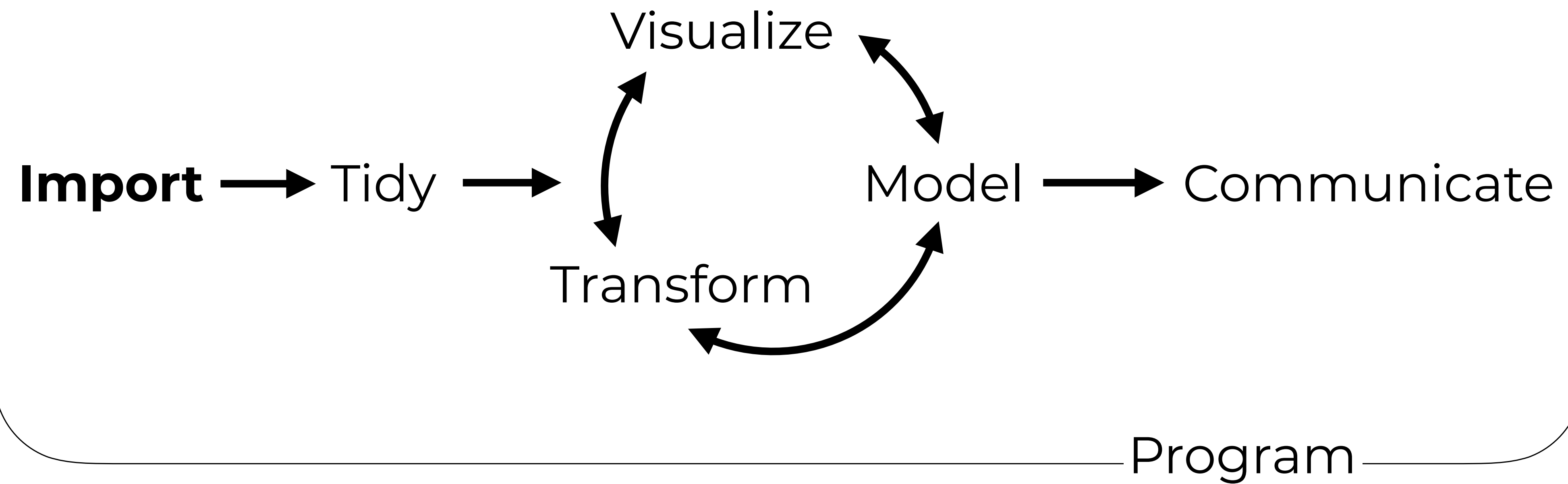
- Is a file named `.here` present?
- Is there a `.Rproj` file (i.e., `tidyds-2019.Rproj`)
- Is there a `.git` or `.svn` directory?

```
dr_here()  
# here() starts at /Users/w449t405/Documents/GIT/courses/tidyds-2019,  
# because it contains a file matching `[.]Rproj$` with contents matching  
# `^Version: ` in the first line
```



In R4DS
Data Import



readr vs. base R

Compared to **read.table()** and friends:

- ~ 10 times faster
- Returns tibbles
- Intuitive defaults (e.g., no strings as factors)

readr functions

function	reads
<code>read_csv()</code>	Comma separated values
<code>read_csv2()</code>	Semi-colon separate values
<code>read_delim()</code>	General delimited files
<code>read_fwf()</code>	Fixed width files
<code>read_log()</code>	Apache log files
<code>read_table()</code>	Space separated files
<code>read_tsv()</code>	Tab delimited values

readr functions

function	reads
<code>read_csv()</code>	Comma separated values
<code>read_csv2()</code>	Semi-colon separate values
<code>read_delim()</code>	General delimited files
<code>read_fwf()</code>	Fixed width files
<code>read_log()</code>	Apache log files
<code>read_table()</code>	Space separated files
<code>read_tsv()</code>	Tab delimited values

nimbus.csv

```
date,longitude,latitude,ozone
1985-10-01T00:00:00Z,-179.375,-73.5,302
1985-10-01T00:00:00Z,-178.125,-73.5,302
1985-10-01T00:00:00Z,-176.875,-73.5,302
1985-10-01T00:00:00Z,-175.625,-73.5,302
1985-10-01T00:00:00Z,-174.375,-73.5,304
1985-10-01T00:00:00Z,-173.125,-73.5,304
1985-10-01T00:00:00Z,-171.875,-73.5,304
1985-10-01T00:00:00Z,-170.625,-73.5,304
1985-10-01T00:00:00Z,-164.375,-73.5,287
1985-10-01T00:00:00Z,-163.125,-73.5,287
1985-10-01T00:00:00Z,-161.875,-73.5,287
1985-10-01T00:00:00Z,-160.625,-73.5,287
```


nimbus.csv

```
date,longitude,latitude,ozone
1985-10-01T00:00:00Z,-179.375,-73.5,302
1985-10-01T00:00:00Z,-178.125,-73.5,302
1985-10-01T00:00:00Z,-176.875,-73.5,302
1985-10-01T00:00:00Z,-175.625,-73.5,302
1985-10-01T00:00:00Z,-174.375,-73.5,304
1985-10-01T00:00:00Z,-173.125,-73.5,304
1985-10-01T00:00:00Z,-171.875,-73.5,304
1985-10-01T00:00:00Z,-170.625,-73.5,304
1985-10-01T00:00:00Z,-164.375,-73.5,287
1985-10-01T00:00:00Z,-163.125,-73.5,287
1985-10-01T00:00:00Z,-161.875,-73.5,287
1985-10-01T00:00:00Z,-160.625,-73.5,287
```

read_csv()

readr functions share a common syntax

```
df <- read_csv("path/to/file.csv", ...)
```

object to
save output
into

path to working
file

read_csv()

readr functions share a common syntax

```
df <- read_csv(here("path", "to", "file.csv"), ...)
```

object to
save output
into

build path to file using
here()

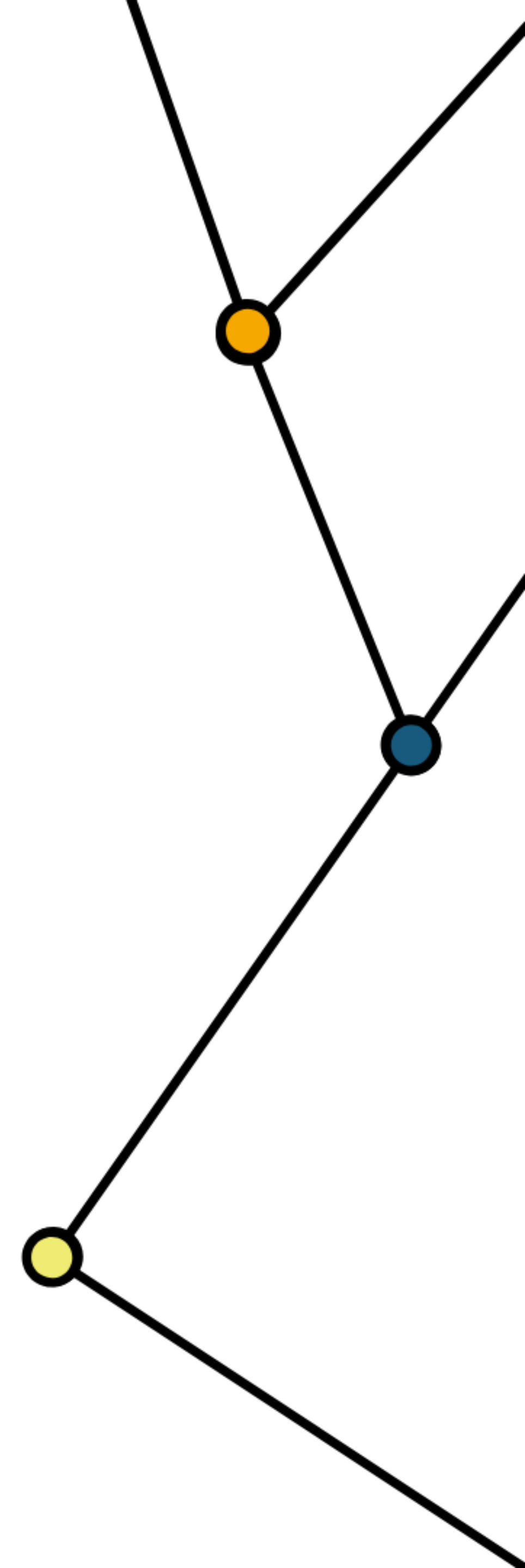
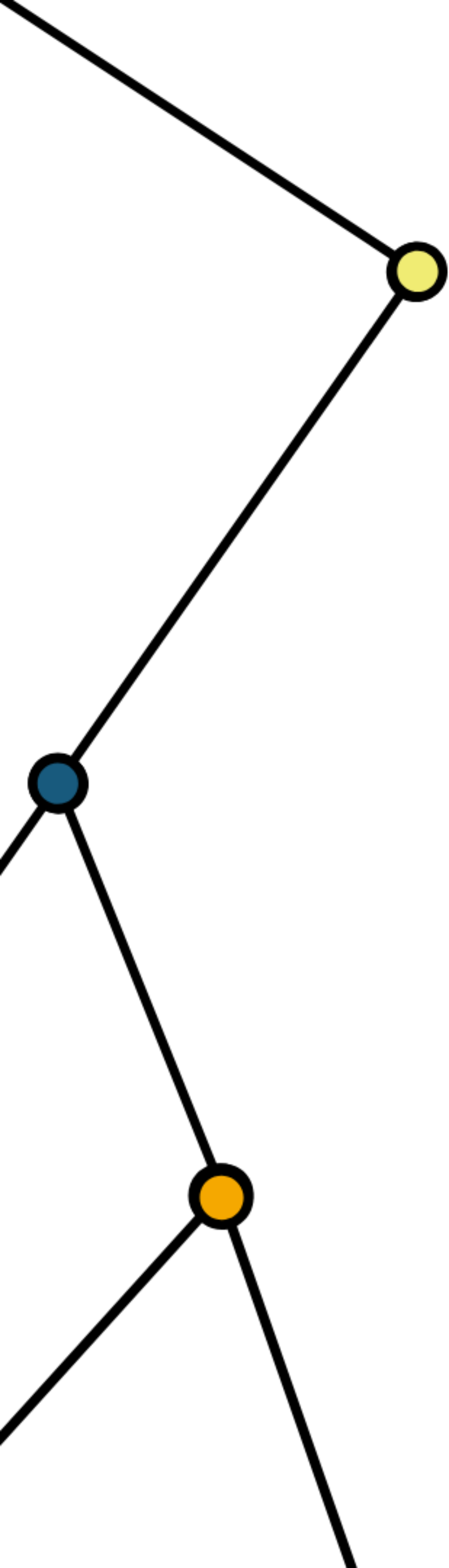
Your Turn 1

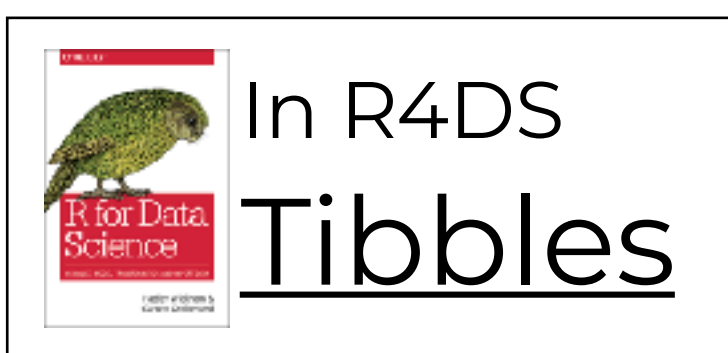
- Find **nimbus.csv** in your project directory
- Read it into an object
- View the results

02 : 00

```
nimbus <- read_csv(here("data", "nimbus.csv"))
nimbus
# A tibble: 18,963 x 4
  date                longitude latitude ozone
  <dtm>                <dbl>     <dbl> <chr>
1 1985-10-01 00:00:00    -179.    -73.5 302
2 1985-10-01 00:00:00    -178.    -73.5 302
3 1985-10-01 00:00:00    -177.    -73.5 302
4 1985-10-01 00:00:00    -176.    -73.5 302
5 1985-10-01 00:00:00    -174.    -73.5 304
6 1985-10-01 00:00:00    -173.    -73.5 304
7 1985-10-01 00:00:00    -172.    -73.5 304
8 1985-10-01 00:00:00    -171.    -73.5 304
9 1985-10-01 00:00:00    -164.    -73.5 287
10 1985-10-01 00:00:00    -163.    -73.5 287
# ... with 18,953 more rows
```

tibbles

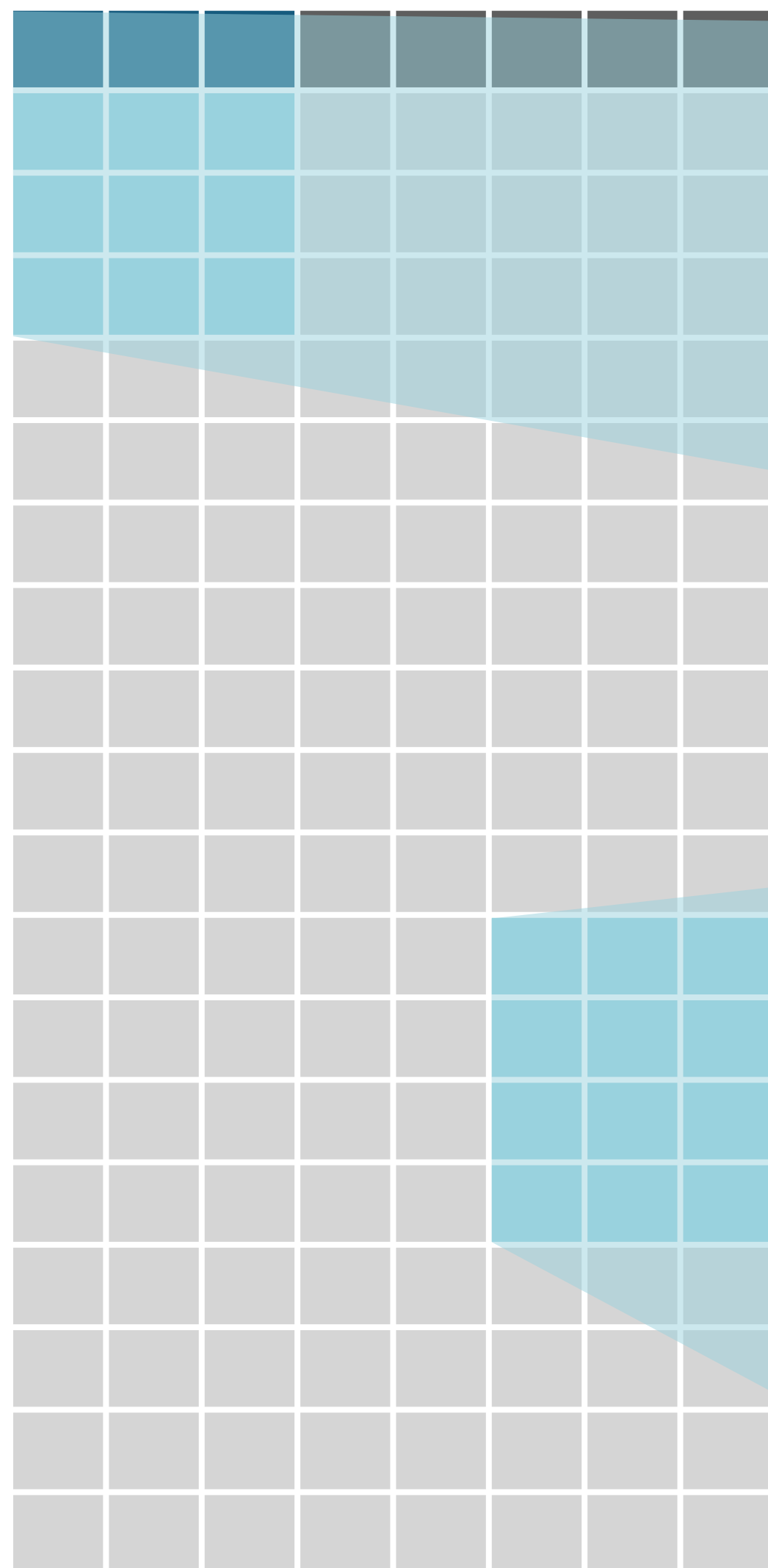




read.csv() vs. read_csv()

```
Console Terminal Jobs
~/Documents/GIT/courses/tidyds-2019/
210 1985-10-01T00:00:00Z -93.125 -72.5 204
211 1985-10-01T00:00:00Z -91.875 -72.5 204
212 1985-10-01T00:00:00Z -90.625 -72.5 204
213 1985-10-01T00:00:00Z -89.375 -72.5 209
214 1985-10-01T00:00:00Z -88.125 -72.5 209
215 1985-10-01T00:00:00Z -86.875 -72.5 209
216 1985-10-01T00:00:00Z -85.625 -72.5 209
217 1985-10-01T00:00:00Z -84.375 -72.5 206
218 1985-10-01T00:00:00Z -83.125 -72.5 206
219 1985-10-01T00:00:00Z -81.875 -72.5 206
220 1985-10-01T00:00:00Z -80.625 -72.5 206
221 1985-10-01T00:00:00Z -79.375 -72.5 208
222 1985-10-01T00:00:00Z -78.125 -72.5 208
223 1985-10-01T00:00:00Z -76.875 -72.5 208
224 1985-10-01T00:00:00Z -75.625 -72.5 208
225 1985-10-01T00:00:00Z -74.375 -72.5 205
226 1985-10-01T00:00:00Z -73.125 -72.5 205
227 1985-10-01T00:00:00Z -71.875 -72.5 205
228 1985-10-01T00:00:00Z -70.625 -72.5 205
229 1985-10-01T00:00:00Z -69.375 -72.5 208
230 1985-10-01T00:00:00Z -68.125 -72.5 208
231 1985-10-01T00:00:00Z -66.875 -72.5 208
232 1985-10-01T00:00:00Z -65.625 -72.5 208
233 1985-10-01T00:00:00Z -64.375 -72.5 223
234 1985-10-01T00:00:00Z -63.125 -72.5 223
235 1985-10-01T00:00:00Z -61.875 -72.5 223
236 1985-10-01T00:00:00Z -60.625 -72.5 223
237 1985-10-01T00:00:00Z -59.375 -72.5 232
238 1985-10-01T00:00:00Z -58.125 -72.5 232
239 1985-10-01T00:00:00Z -56.875 -72.5 232
240 1985-10-01T00:00:00Z -55.625 -72.5 232
241 1985-10-01T00:00:00Z -54.375 -72.5 238
242 1985-10-01T00:00:00Z -53.125 -72.5 238
243 1985-10-01T00:00:00Z -51.875 -72.5 238
244 1985-10-01T00:00:00Z -50.625 -72.5 238
245 1985-10-01T00:00:00Z -49.375 -72.5 243
246 1985-10-01T00:00:00Z -48.125 -72.5 243
247 1985-10-01T00:00:00Z -46.875 -72.5 243
248 1985-10-01T00:00:00Z -45.625 -72.5 243
249 1985-10-01T00:00:00Z -44.375 -72.5 232
250 1985-10-01T00:00:00Z -43.125 -72.5 232
[ reached 'max' / getOption("max.print") -- omitted 18713 rows ]
>
```

```
Console Terminal Jobs
~/Documents/GIT/courses/tidyds-2019/
> nimbus
# A tibble: 18,963 x 4
  date                longitude latitude ozone
  <dtm>              <dbl>    <dbl> <chr>
1 1985-10-01 00:00:00 -179.    -73.5 302
2 1985-10-01 00:00:00 -178.    -73.5 302
3 1985-10-01 00:00:00 -177.    -73.5 302
4 1985-10-01 00:00:00 -176.    -73.5 302
5 1985-10-01 00:00:00 -174.    -73.5 304
6 1985-10-01 00:00:00 -173.    -73.5 304
7 1985-10-01 00:00:00 -172.    -73.5 304
8 1985-10-01 00:00:00 -171.    -73.5 304
9 1985-10-01 00:00:00 -164.    -73.5 287
10 1985-10-01 00:00:00 -163.    -73.5 287
# ... with 18,953 more rows
> |
```



```
# A tibble: 87 x 13
  name height mass hair_color skin_color eye_color birth_year gender
<chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr>
1 Luke... 172 77 blond fair blue 19 male
2 C-3P0 167 75 NA gold yellow 112 NA
3 R2-D2 96 32 NA white, bl... red 33 NA
4 Dart... 202 136 none white yellow 41.9 male
5 Leia... 150 49 brown light brown 19 female
6 Owen... 178 120 brown, gr... light blue 52 male
7 Beru... 165 75 brown light blue 47 female
8 R5-D4 97 32 NA white, red red NA NA
9 Bigg... 183 84 black light brown 24 male
10 Obi-... 182 77 auburn, w... fair blue-gray 57 male
# ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
# films <list>, vehicles <list>, starships <list>
```

tibble display

```
48
49
50
51
52
53
54
55 Jedi starfighter
56
57 Naboo fighter
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
[ reached 'max' / getOption("max.print") -- omitted 11 rows ]
```

data frame display

tibbles

A type of data frame common throughout the tidyverse packages

Tibbles enhance data frames in three ways:

1. **Subsetting** - ``[`` always returns a new tibble, ``[[`` and ``$`` always return a new vector
2. **No partial matching** - You must use full column names when subsetting
3. **Display** - When you print a nibble, R provides a concise view of the data that fits on one screen

tibble helpers

function

does

`as_tibble()`

convert a data frame to a tibble

`as.data.frame()`

convert a tibble to a data frame

`tribble()`

make a tibble (transversed)

```
tribble(  
  ~x, ~y,  
  1, "a",  
  2, "b",  
  3, "c")
```



x	y
1	a
2	b
3	c

Tibbles - an enhanced data frame

The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve three behaviors:

- **Subsetting** - `[` always returns a new tibble, `[]` and `$` always return a vector.
- **No partial matching** - You must use full column names when subsetting
- **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen

tibble display

A large table to display

- Control the default appearance with options:
`options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)`
- View full data set with **View()** or **glimpse()**
- Revert to data frame with **as.data.frame()**

CONSTRUCT A TIBBLE IN TWO WAYS

tibble(...)
Construct by columns.
`tibble(x = 1:3, y = c("a", "b", "c"))`

tribble(...)
Construct by rows.
`tribble(~x, ~y, 1, "a", 2, "b", 3, "c")`

as_tibble(...) Convert data frame to tibble.
`enframe(x, name = "name", value = "value")`
Convert named vector to a tibble
`is_tibble(x)` Test whether x is a tibble.

Tidy Data with tidyr

Tidy data is a way to organize tabular data. It provides a consistent data structure across packages. A table is tidy if:

- Each **variable** is in its own **column**
- Each **observation**, or **case**, is in its own **row**

Tidy data:

- Makes variables easy to access as vectors
- Preserves cases during vectorized operations

Reshape Data

Use **gather()** and **spread()** to reorganize the values of a table into a new layout.

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE)
gather() moves column names into a key column, gathering the column values into a single value column.

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)
spread() moves the unique values of a key column into the column names, spreading the values of a value column across the new columns.

Handle Missing Values

drop_na(data, ...)
Drop rows containing NA's in ... columns.

fill(data, ..., direction = c("down", "up"), fill = NA)
Fill in NA's in ... columns with most recent non-NA values.

replace_na(data, replace = list(...))
Replace NA's by column.

Expand Tables

quickly create tables with combinations of values

complete(data, ..., fill = list())
Adds to the data missing combinations of the values of the variables listed in ...
`complete(mtcars, cyl, gear, carb)`

expand(data, ...)
Create new tibble with all possible combinations of the values of the variables listed in ...
`expand(mtcars, cyl, gear, carb)`

Split Cells

Use these functions to split or combine cells into individual, isolated values.

separate(data, col, into, sep = "[[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...)
Separate each cell in a column to make several columns.

separate_rows(data, ..., sep = "[[:alnum:]]+", convert = FALSE)
Separate each cell in a column to make several rows. Also **separate_rows_()**.

unite(data, col, ..., sep = "", remove = TRUE)
Collapse cells across several columns to make a single column.

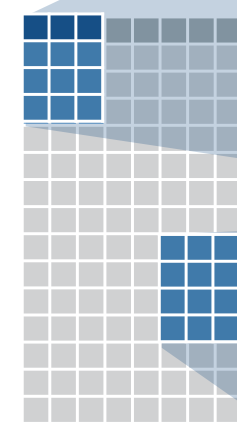
RStudio is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1222 • rstudio.com • Learn more at tidymasterclass.com • readr 1.1.0 • tibble 1.2.12 • dplyr 0.8.0 • Updated: 2017-01

Tibbles - an enhanced data frame



The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve three behaviors:

- **Subsetting** - `[` always returns a new tibble, `[]` and `$` always return a vector.
- **No partial matching** - You must use full column names when subsetting
- **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen



A large table to display

```
# A tibble: 234 × 6
  manufacturer model displ
<chr> <chr> <dbl>
1 audi a4 1.8
2 audi a4 1.8
3 audi a4 2.0
4 audi a4 2.0
5 audi a4 2.0
6 audi a4 2.0
7 audi a4 3.1
8 audi a4 quattro 1.8
9 audi a4 quattro 1.8
10 ... with 224 more rows, and 3
#> more variables: year <int>,
#> cyl <int>, trans <chr>
```

tibble display

```
156 1999 6 auto{l4}
157 1999 6 auto{l4}
158 2008 6 auto{l4}
159 2008 6 auto{l4}
160 2008 4 manual{m5}
161 1999 4 auto{l4}
162 2008 4 manual{m5}
163 2008 4 manual{m5}
164 2008 4 auto{l4}
165 2008 4 auto{l4}
166 1999 4 auto{l4}
-- reached getOption("max.print")
-- omitted 68 rows --
```

data frame display

- Control the default appearance with options:
options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)
- View full data set with **View()** or **glimpse()**
- Revert to data frame with **as.data.frame()**

CONSTRUCT A TIBBLE IN TWO WAYS

tibble(...)
Construct by columns.
`tibble(x = 1:3, y = c("a", "b", "c"))`

Both make this tibble

tribble(...)
Construct by rows.
`tribble(~x, ~y, 1, "a", 2, "b", 3, "c")`

```
# A tibble: 3 × 2
  x     y
<int> <chr>
1     1 a
2     2 b
3     3 c
```

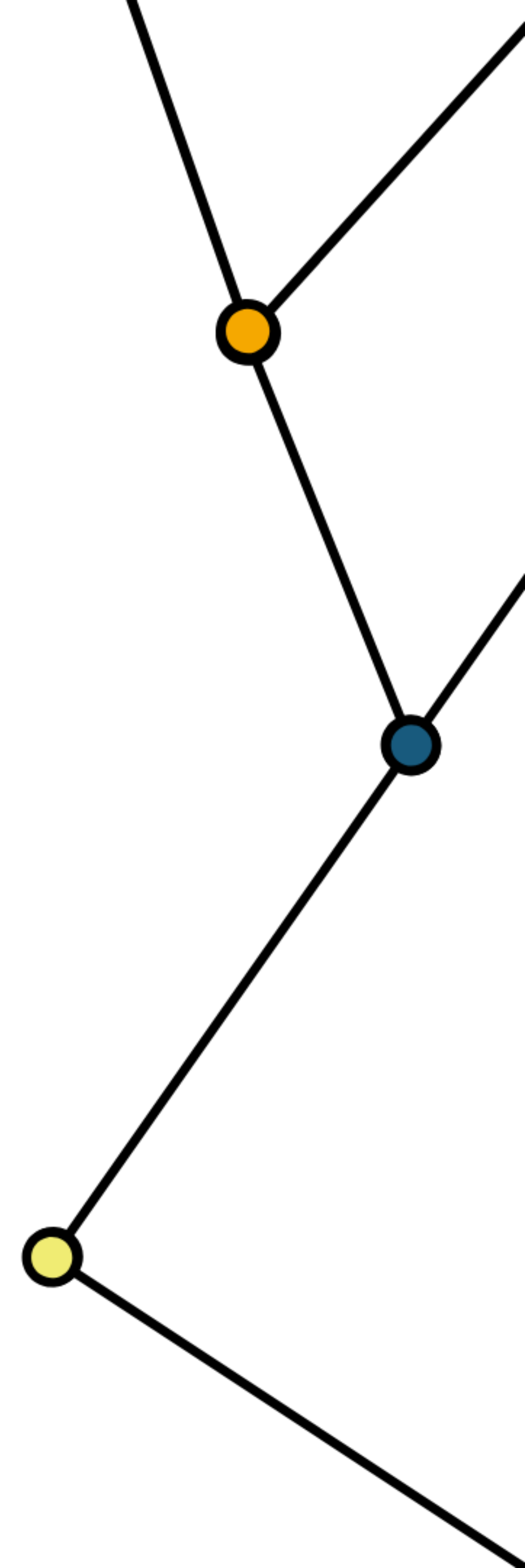
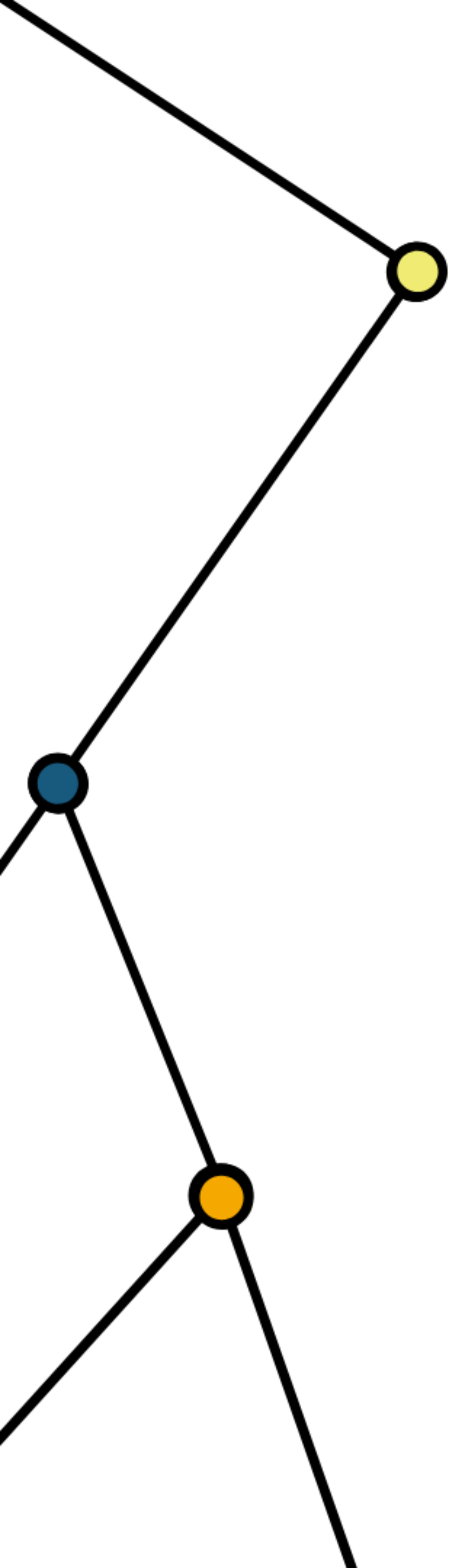
as_tibble(x, ...) Convert data frame to tibble.

enframe(x, name = "name", value = "value")
Convert named vector to a tibble

is_tibble(x) Test whether x is a tibble.



Parsing



Quiz

- Look at the **nimbus** data
- What class (data type) is ozone?

```
nimbus %>%  
  pull(ozone) %>%  
  class()
```



```

nimbus %>%
  pull(ozone) %>%
  class()
# [1] "character"

nimbus %>%
  pull(ozone) %>%
  unique()
# [1] "302" "304" "287" "274" "264" "242" "211" "195"
# [9] "197" "196" "198" "193" "187" "190" "199" "194"
# [17] "213" "218" "221" "229" "209" "186" "188" "191"
# [25] "189" "184" "180" ". " "215" "312" "319" "320"
# [33] "311" "300" "290" "267" "226" "210" "200" "203"
# [41] "201" "192" "204" "206" "208" "205" "223" "232"
# [49] "238" "243" "220" "202" "185" "219" "222" "216"

```

NA values

```
nimbus %>%  
  filter(ozone == ".")  
# A tibble: 155 x 4  
  date                longitude latitude ozone  
  <dtm>                <dbl>    <dbl> <chr>  
1 1985-10-01 00:00:00      70.6    -73.5 .  
2 1985-10-01 00:00:00      71.9    -73.5 .  
3 1985-10-01 00:00:00      73.1    -73.5 .  
4 1985-10-01 00:00:00      74.4    -73.5 .  
5 1985-10-01 00:00:00      75.6    -73.5 .  
6 1985-10-01 00:00:00      76.9    -73.5 .  
7 1985-10-01 00:00:00      78.1    -73.5 .  
8 1985-10-01 00:00:00      79.4    -73.5 .  
9 1985-10-01 00:00:00      65.6    -72.5 .  
10 1985-10-01 00:00:00      66.9    -72.5 .  
# ... with 145 more rows
```

Define missing values

readr functions share a common syntax

```
df <- read_csv(here("data", "nimbus.csv"), na = ".")
```

object to
save output
into

build path to file using
here()

Values to
convert to **NA**

Your Turn 2

- Read in **nimbus.csv**
- Set values of `.` to **NA**

02 : 00


```
nimbus <- read_csv(here("data", "nimbus.csv"), na = ".")
```

```
nimbus
```

```
# A tibble: 18,963 x 4
```

	date	longitude	latitude	ozone
	<dtm>	<dbl>	<dbl>	<dbl>
1	1985-10-01 00:00:00	-179.	-73.5	302
2	1985-10-01 00:00:00	-178.	-73.5	302
3	1985-10-01 00:00:00	-177.	-73.5	302
4	1985-10-01 00:00:00	-176.	-73.5	302
5	1985-10-01 00:00:00	-174.	-73.5	304
6	1985-10-01 00:00:00	-173.	-73.5	304
7	1985-10-01 00:00:00	-172.	-73.5	304
8	1985-10-01 00:00:00	-171.	-73.5	304
9	1985-10-01 00:00:00	-164.	-73.5	287
10	1985-10-01 00:00:00	-163.	-73.5	287

```
# ... with 18,953 more rows
```

**<dbl> stand for
double
(decimal number)**

```
nimbus <- read_csv(here("data", "nimbus.csv"))
nimbus
# A tibble: 18,963 x 4
  date                longitude latitude ozone
  <dtm>                <dbl>     <dbl> <chr>
1 1985-10-01 00:00:00    -179.    -73.5 302
2 1985-10-01 00:00:00    -178.    -73.5 302
3 1985-10-01 00:00:00    -177.    -73.5 302
4 1985-10-01 00:00:00    -176.    -73.5 302
5 1985-10-01 00:00:00    -174.    -73.5 304
6 1985-10-01 00:00:00    -173.    -73.5 304
7 1985-10-01 00:00:00    -172.    -73.5 304
8 1985-10-01 00:00:00    -171.    -73.5 304
9 1985-10-01 00:00:00    -164.    -73.5 287
10 1985-10-01 00:00:00    -163.    -73.5 287
# ... with 18,953 more rows
```

**<chr> stand for
character string
(not a number)**

Specify column types

readr functions share a common syntax

```
df <- read_csv(here("path", "to", "file.csv"), na = ".",  
               col_types = cols(ozone = col_integer()))
```

Manually
specify
column types

cols()

Column
name

Column type
function

Column types

type function	data type
<code>col_character()</code>	character
<code>col_date()</code>	Date
<code>col_datetime()</code>	POSIXct (date-time)
<code>col_double()</code>	double (numeric)
<code>col_factor()</code>	factor
<code>col_guess()</code>	let readr guess (default)
<code>col_integer()</code>	integer
<code>col_logical()</code>	logical
<code>col_number()</code>	numbers mixed with non-number characters
<code>col_numeric()</code>	double or integer
<code>col_skip()</code>	do not read
<code>col_time()</code>	time

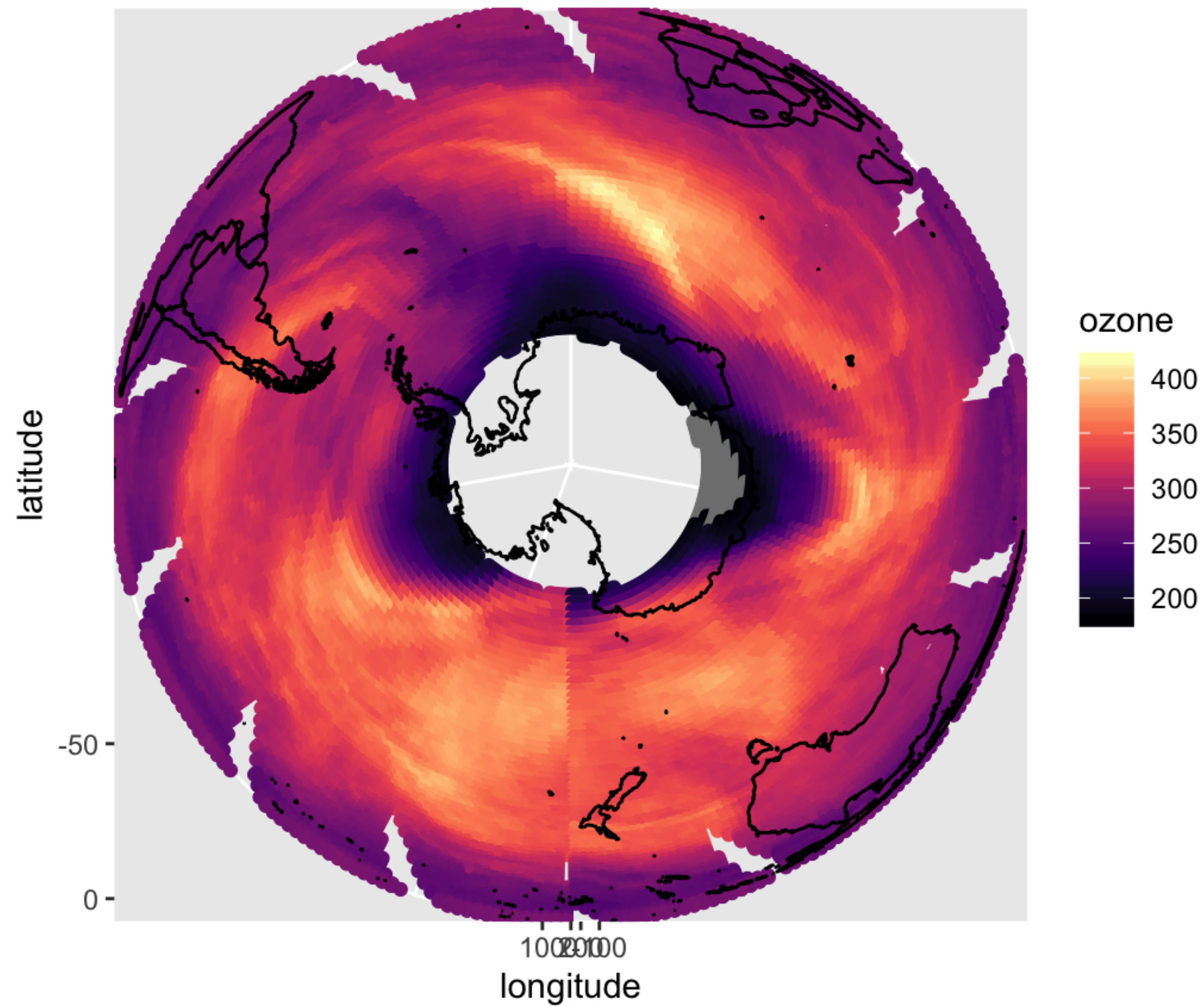
Your Turn 3

- Modify the code below
- Specify **ozone** as integer values

```
nimbus <- read_csv(here("data", "nimbus.csv"), na = ".")
```

02 : 00


```
nimbus <- read_csv(here("data", "nimbus.csv"), na = ".",  
                  col_types = cols(ozone = col_integer()))  
  
library(viridis)  
world <- map_data(map = "world")  
ggplot(data = nimbus) +  
  geom_point(mapping = aes(x = longitude, y = latitude, color = ozone)) +  
  geom_path(data = world, mapping = aes(x = long, y = lat, group = group)) +  
  coord_map("ortho", orientation = c(-90, 0, 0)) +  
  scale_color_viridis(option = "A")
```



Other Data Types



Excel Files (.xls and .xlsx)



Data from other statistical software (SPSS, Stata, and SAS)



Google Sheets and other files in Google Drive



Efficient data sharing between R and Python

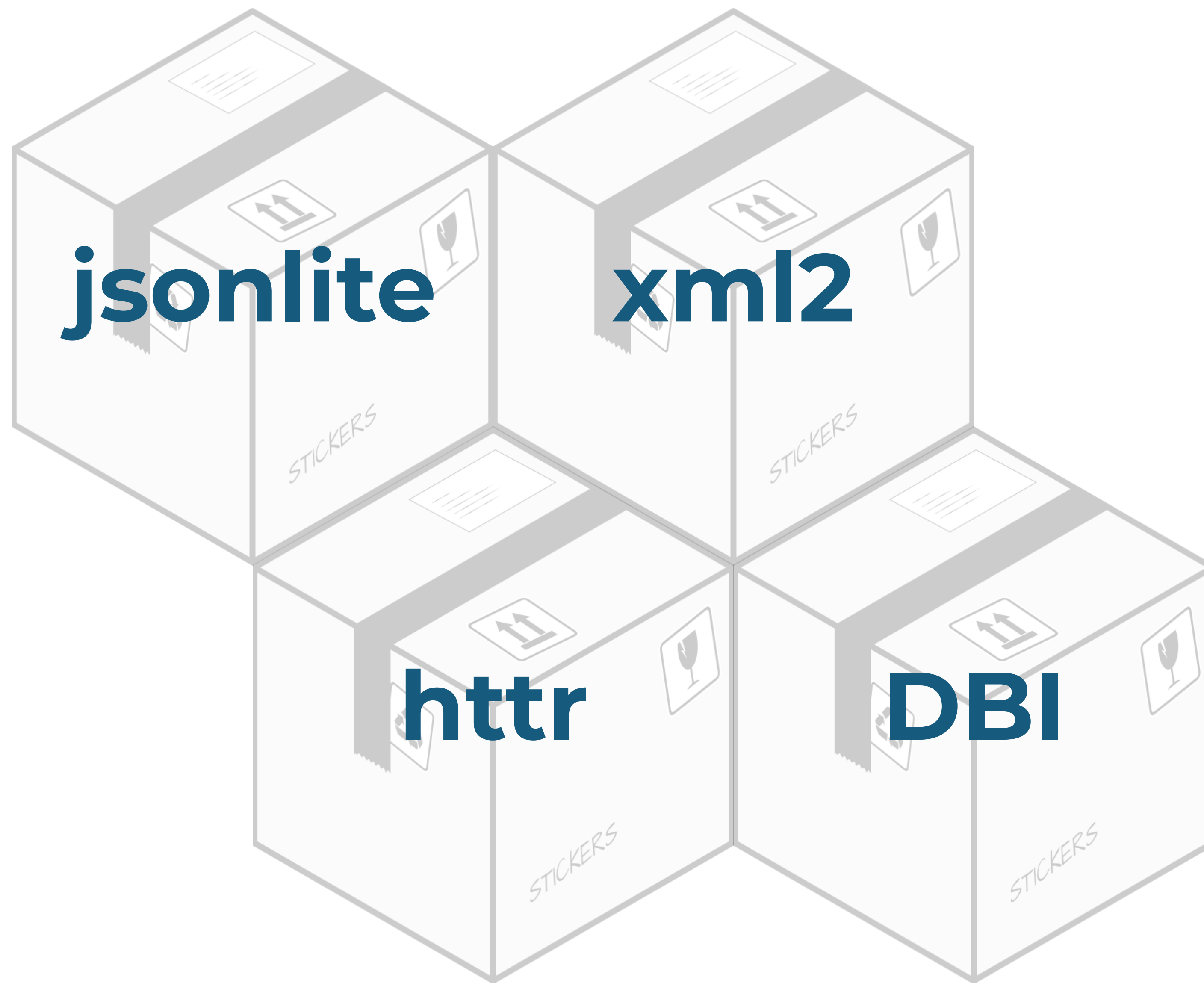


Web Pages (web scraping)



Data loaded into spark





Import Data

