Week 9 Cohort 4: R4DS Book Club

Chapters 8, 9, & 10

Workflow: projects; Tibbles

Collin K. Berke

Twitter: @BerkeCollin

Last updated: 2021-02-10

5-minute ice breaker

- What types of data do you work with?

Quick housekeeping/reminders

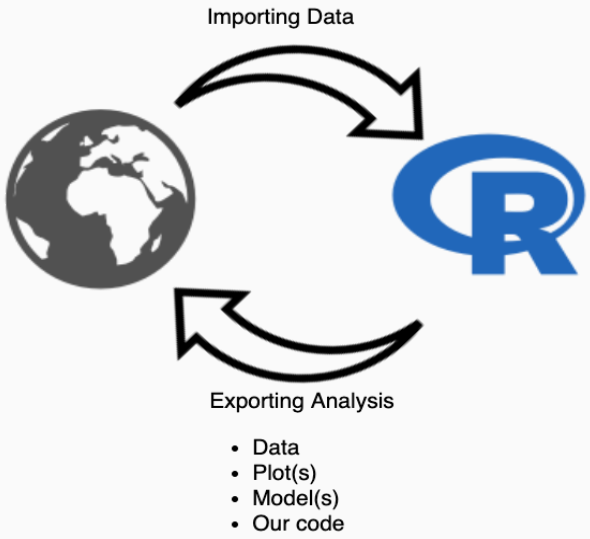
- Video camera is optional, but encouraged.
- If we need to slow down and discuss, let me know.
    - Most likely someone has the same question.
- Take time to learn the theory.
- Please attempt the chapter exercises.
- Please plan on teaching one of the lessons.

Tonight's discussion

- Chapter 8 - Workflow: projects
- Chapter 9 - Data wrangling introduction
- Chapter 10 - Tibbles

Why care about a project-oriented workflow?

> "One day you will need to bring data from the outside world into R and send numerical results and figures from R back into the world. ~Hadley Wickham & Garrett Grolemund (authors)"



Why care about a project-oriented workflow (cont.)?

- Maximize effectiveness
- Reduce frustration
- Jenny Bryan on 'Project-oriented workflows'
- Sharla Gelfand on repeated reporting
    - Some of the concepts covered will be discussed in later chapters

Consider: What is real from our analysis?

> "With your R scripts (and your data files), you can recreate the environment. It's much harder to recreate your R scripts from your environment! ~Hadley Wickham & Garrett Grolemund (authors)"

- Create well-commented, self-contained code and projects to aid replication.
- To facilitate this:
  - Change settings to NOT save your workspace.
  - Restart RStudio and rerun your script to check if everything is present
  - `Cmd/Ctrl + Shift + F10` - Restarts R
  - `Cmd/Ctrl + Shift + S` - Rerun the current script

Where does our analysis take place?

Okay, it's a little more complex.

- Paths and directories (AKA the location of your files).
- Three differences across operating systems (Linux/Mac vs. Windows)
    1. Slashes (e.g., `/`, Mac & Linux) vs. backslashes (e.g. `\\`, Windows).
        - *Backslashes are special (e.g., \\)*, so use slashes
    2. Absolute paths, avoid them
        - `/users/cberke/project-name/data`👎 vs `/project-name/data`👍
        - Other users will have different directories.
        - Use relative paths.
    3. The `~` (AKA the home directory)
        - Windows points to `Documents`

Avoid `setwd()`

- You *_can_* set your working directory by doing something like this: `setwd("/path/to/project/files")`

- > I will come into your office and SET YOUR COMPUTER ON FIRE 🔥 ~Jenny Bryan

  - Path's wont work for another user or on another computer.
  - Assumes all projects are done in one R process.
    - Avoid project leakage 💧.

Rstudio projects 🎉

- Demo Rstudio project set up.
- Create notebook.
    - Highlight inserting a code chunk

Your turn 👩‍💻

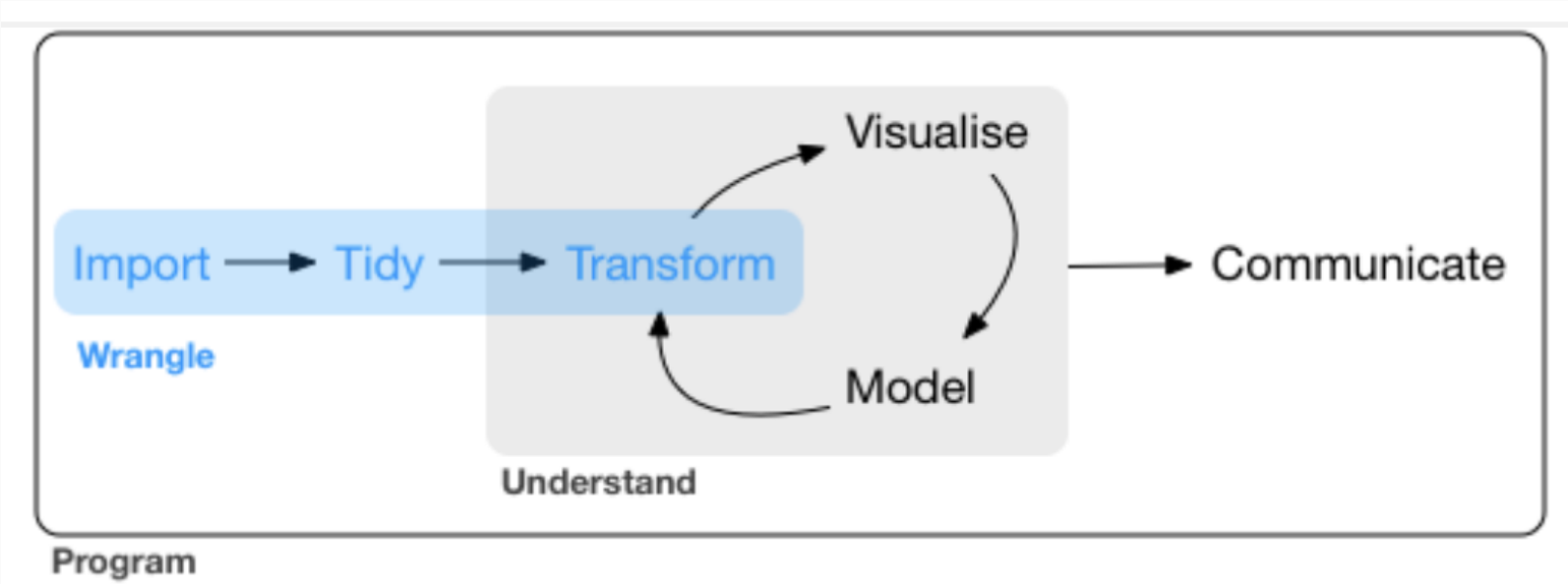
1. Open Rstudio and run `getwd()` (Close a project if you have one open)
    - What file path is returned?
2. Create a Rstudio project
3. Run `getwd()` again
    - What file path is returned?
4. Create a Rnotebook
5. Enter and run the following in a code chunk:

```
library(tidyverse)

# **This will write a file to your computer**
write_csv(mtcars, "mtcars.csv")
```

6. Can you find the file using your file explorer?

Chapter 9: Intro to data wrangling



Data science workflow ~Hadley Wickham & Garrett Grolemund (authors)

What does this section of the book cover?

- **Data wrangling:**
  - Tibbles
  - Data import
  - Tidy data
- **Data transformation:**
  - Relational data
  - Strings
  - Factors
  - Dates and times

Chapter 10: Tibbles

- What is a `tibble`?
  - We've already worked with them!
  - A data structure adapted from Base R's `data.frame`.
- How are `tibbles` different from a `data.frame`?
  - > Tibbles are data frames, but they tweak some older behaviors to make life a little easier. ~Hadley Wickham & Garrett Grolemund (authors)
  - > tibbles provide stricter checking and better formatting than the traditional data frame. ~Hadley Wickham & Garrett Grolemund (authors)
  - > The general ethos is that tibbles are lazy and surly: they do less and complain more than the base `data.frame`s. ~ `tbl_df-class` documentation.
  - Tibbles are opinionated

Let's observe these properties

- Example 1:

```
mtcars # A data.frame
class(mtcars) # Check the class
tbl_sum(mtcars)
```

- Example 2:

```
mtcars_tbl ← as_tibble(mtcars) # using as_tibble() to create a tibble
mtcars_tbl
class(mtcars)
tbl_sum(mtcars)
```

- What differences do you see?

Creating tibbles

- Have a data.frame? Use `as_tibble()`.

```
(iris ← as_tibble(iris))
```

- Creating a tibble from vectors.

```
# Notice the recycling rules of inputs of length 1
tibble(
  x = 1:5,
  y = 1,
  z = x ^ 2 + y
)
```

- Use `tribble()` for data entry in code

```
tribble(
  ~x, ~y, ~z,
  #--|--|----
  "a", 2, 3.6,
  "b", 1, 8.5
)
```

When creating/using tibbles, know that...

- `tibble()` never changes the type of the inputs
- `tibble()` never changes the names of variables
- `tibble()` never creates row names
- `tibble()` allows the use of non-syntactic names

```
(tb ← tibble(
  `:)` = "smile",
  ` ` = "space",
  `2000` = "number"
))
```

```
## # A tibble: 1 x 3
##   `:)`  ` `   `2000`
##   <chr> <chr> <chr>
## 1 smile space number
```

- tibbles are at the core of the tidyverse, so get comfortable using them

Tibbles vs. data.frame

1. **Printing**
    - Tibbles have a refined print method.
    - Rows and columns fit the screen.
    - Data types are reported.
    - You have flexibility to change the output, `package?tibble`
2. **Subsetting**
    - New tools: `$` and `[[`

Tibble subsetting

```
# Create a tibble
tibble_df ← tibble(
  abc = runif(5),
  xyz = rnorm(5)
)

# Create a data.frame
df ← data.frame(
  abc = runif(5),
  xyz = rnorm(5)
)
```

Tibble subsetting (cont.)

```
# Using the $, exact match
tibble_df$abc
```

```
## [1] 0.79014970 0.21065367 0.13461285 0.57002023 0.06602784
```

```
df$abc
```

```
## [1] 0.96659906 0.72045568 0.16218410 0.50333947 0.02519313
```

```
# Using the $, partial match won't work for tibbles
tibble_df$a
```

```
## Warning: Unknown or uninitialised column: `a`.
```

```
## NULL
```

```
# Partial match does for data.frames
df$a
```

```
## [1] 0.96659906 0.72045568 0.16218410 0.50333947 0.02519313
```

Tibble subsetting (cont.)

```
# Using [[]], by position and name
tibble_df[[1]]
```

```
## [1] 0.79014970 0.21065367 0.13461285 0.57002023 0.06602784
```

```
tibble_df[["abc"]]
```

```
## [1] 0.79014970 0.21065367 0.13461285 0.57002023 0.06602784
```

```
df[[1]]
```

```
## [1] 0.96659906 0.72045568 0.16218410 0.50333947 0.02519313
```

```
df[["abc"]]
```

```
## [1] 0.96659906 0.72045568 0.16218410 0.50333947 0.02519313
```

Tibble subsetting (cont.)

```
# Using the pipe, you need the `.` place holder
tibble_df %>% .$abc
```

```
## [1] 0.79014970 0.21065367 0.13461285 0.57002023 0.06602784
```

```
tibble_df %>% .[["xyz"]]
```

```
## [1] -0.44601287  0.02497698  0.19212898  0.37532214 -0.83882670
```

Converting back to a data.frame

- Older functions may not work well with tibbles.
  - Use `as.data.frame()`

```
class(as.data.frame(tibble_df))
```

```
## [1] "data.frame"
```

Questions/Discussion