Programming Practices
Winter Institute in Data Science

Ryan T. Moore

2026-01-05

Style

Style

https://style.tidyverse.org

Good File Names

```
fit_models.R
clean_data.R
```

Good File Names

`fit_models.R`

`clean_data.R`

(not `stuff.R`, `trying.R`)

Good File Names

```
00-read-data.R
01-clean-data.R
```

Good File Names

`00-read-data.R`

`01-clean-data.R`

(not `work.R`, `laterwork.R`)

Good Variable Names

```
janitor::clean_names()
```

Good Internal Organization

1. Start `.R` with a comment preamble (context, data, author?, etc.)
2. Start with

```
library(here)
library(janitor)
library(tidyverse)
```

3. Set off sections in `.R` with Cmd-Shft-R

Good Object Names

- meaningful
- lower case
- numbers
- separate words with _
- short enough
- not common R usages

Good Object Names

- meaningful
- lower case
- numbers
- separate words with _
- short enough
- not common R usages

not

```
T <- FALSE
c <- 10
mean <- function(x){
  sum(x)
}
```

Legibility

1. Comma space as in English

```
df[, 5]
c(1, 2, 3)
```

Legibility

1. Comma space as in English

```
df[, 5]
c(1, 2, 3)
```

2. Parentheses without space

```
mean(x)
```

(not `mean (x)` nor `mean( x )`)

Legibility

1. Comma space as in English

```
df[, 5]
c(1, 2, 3)
```

2. Parentheses without space

```
mean(x)
```

(not `mean (x)` nor `mean( x )`)

3. Space around operators

```
x <- y      4 / 11
```

Legibility

4. No space around high-priority operators

```
1:10
x[6]
```

Legibility

4. No space around high-priority operators

```
1:10
x[6]
```

5. Align delimiters `{` with `}`

Legibility

4. No space around high-priority operators

```
1:10
x[6]
```

5. Align delimiters `{` with `}`
6. Next pipes on new lines

Legibility

4. No space around high-priority operators

```
1:10
x[6]
```

5. Align delimiters `{` with `}`
6. Next pipes on new lines
7. Next args on new lines, when long, or helpful

Usage

1. Use `<-` for assignment.
2. Use `#` for comments.
3. Comments with sentence case, but no period, unless two sentences.

Workflow: Tips and Tricks

Keyboard Tips

Tab completion	speed, accuracy
Up arrow	previous command
Cmd-Return	code ⇝ Console
Cmd-Shift-0	restart R
Cmd-Shift-A	reformat highlit code
Alt-Shift-K	shortcut for shortcuts
Cmd-Shift-S	`source()` a `.R` file
Cmd-Shft-R	make section in `.R`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Avoid

- `install.packages("...")`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Avoid

- `install.packages("...")`
- `setwd("C:Users/me/my/unique/dir/structure/")`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Avoid

- `install.packages("...")`
- `setwd("C:Users/me/my/unique/dir/structure/")`
- `rm(list = ls())`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Avoid

- `install.packages("...")`
- `setwd("C:Users/me/my/unique/dir/structure/")`
- `rm(list = ls())`

Reproducibility: How to Start a File

- Some metadata (what, who, ...)
- `library(fivethirtyeight)`
  `library(tidyverse)`

Avoid

- `install.packages("...")`
- `setwd("C:Users/me/my/unique/dir/structure/")`
- `rm(list = ls())`

Why?

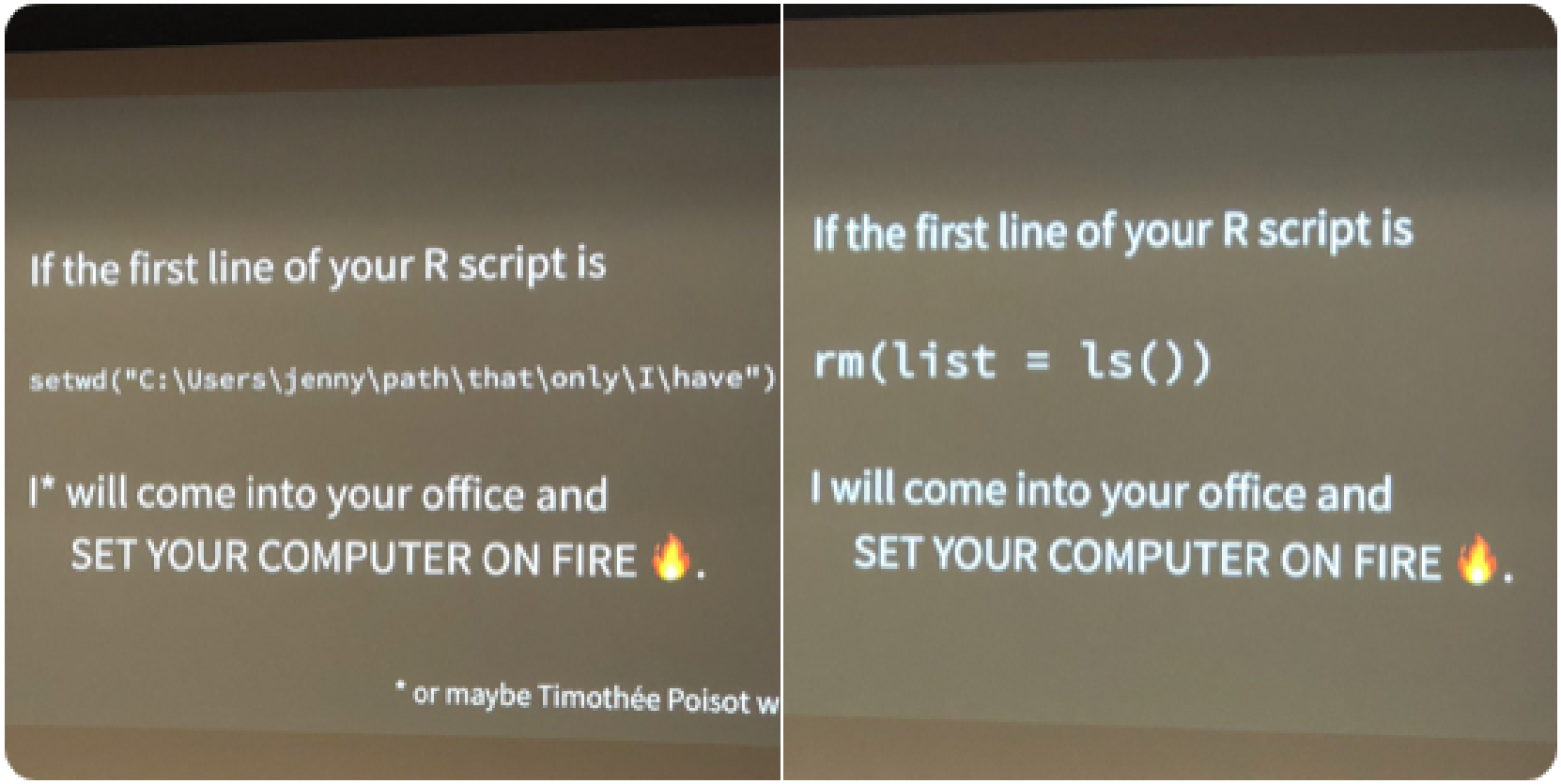
Hadley Wickham 

@hadleywickham

Follow

The only two things that make @JennyBryan 😤😠🤯. Instead use projects + here::here()

#rstats



4:50 PM - 10 Dec 2017

Reproducibility: Starting Workflow

You should start fresh.

Reproducibility: Starting Workflow

You should start fresh.

Turn off "Restore my history/workspace" prefs.

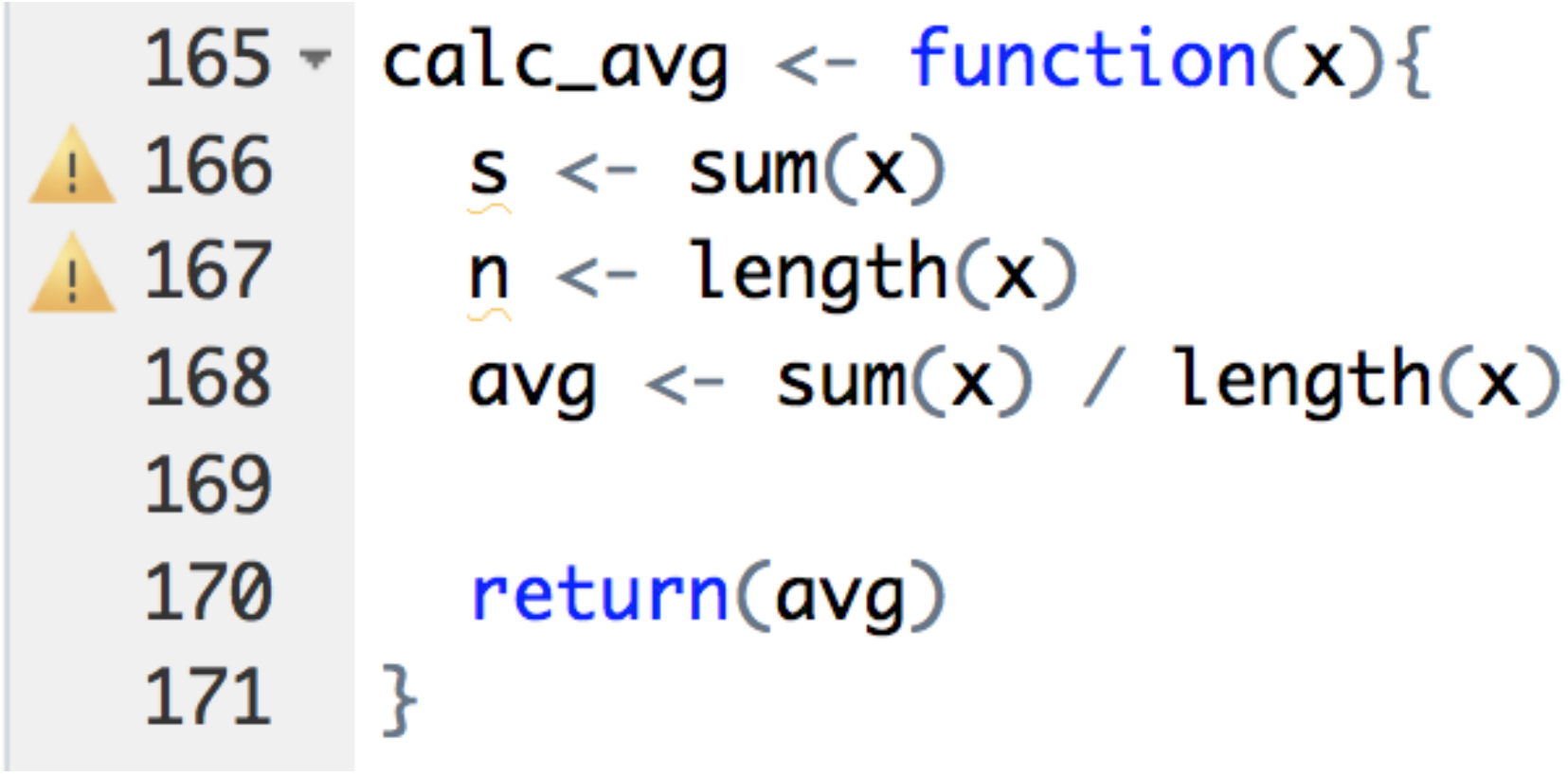
Reproducibility: Starting Workflow

You should start fresh.

Turn off “Restore my history/workspace” prefs.

Cmd-Shft-0 regularly.

RStudio Diagnostics (turn them on)



R Projects

What is real?

What is real?

The code is real.

What is not real?

What is not real?

Objects (currently) in the workspace

How could this possibly matter?

How could this possibly matter?

It does.

Recently, at The Lab ...

```
full_df <- read_csv("...")
t_out <- t.test(outcome ~ treatment, data = full_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Recently, at The Lab ...

```
full_df <- read_csv("...")
t_out <- t.test(outcome ~ treatment, data = full_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Then, time pressure ...a meeting ...

Recently, at The Lab ...

```
full_df <- read_csv("...")
t_out <- t.test(outcome ~ treatment, data = full_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Then, time pressure ...a meeting ...

```
subset_df <- filter(full_data, <condition>)
t_out <- t.test(outcome ~ treatment, data = subset_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Recently, at The Lab ...

```
full_df <- read_csv("...")
t_out <- t.test(outcome ~ treatment, data = full_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Then, time pressure ...a meeting ...

```
subset_df <- filter(full_data, <condition>)
t_out <- t.test(outcome ~ treatment, data = subset_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Then, redo first plot adding aesthetics, error bars, ...

Recently, at The Lab ...

```
full_df <- read_csv("...")
t_out <- t.test(outcome ~ treatment, data = full_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Then, time pressure ...a meeting ...

```
subset_df <- filter(full_data, <condition>)
t_out <- t.test(outcome ~ treatment, data = subset_df)
# Process into tmp df ...
ggplot(df, aes(treatment, prop)) + geom_bar()
```

Then, redo first plot adding aesthetics, error bars, ...

What is real `t_out`? `df`? `gg <- ggplot()` object?

The source code is real.

The source code is real.

The objects are realizations of the source code. Source for EVERY user modified object is placed in a particular directory or directories, for later editing and retrieval.

– the ESS manual

The Notebook Problem

There are notebooks.

The Notebook Problem

There are notebooks.

- Joel Grus, author of *Data Science from Scratch*

The Notebook Problem

There are notebooks.

- Joel Grus, author of *Data Science from Scratch*
- Talk at JupyterCon 2018: http://j.mp/2QHxjHB

The Notebook Problem

There are notebooks.

- Joel Grus, author of *Data Science from Scratch*
- Talk at JupyterCon 2018: http://j.mp/2QHxjHB
- "hidden state" problem

The Notebook Problem

There are notebooks.

- Joel Grus, author of *Data Science from Scratch*
- Talk at JupyterCon 2018: http://j.mp/2QHxjHB
- "hidden state" problem
  - "state": contents of memory locs at pt in time

Ensure that Code Captures Reality

- Start fresh (Cmd-Shft-0)
- Turn off “Restore history/workspace” prefs
- Run the file (Cmd-Shift-S)
- Use relative paths from the working dir

How to Behave (featuring Jenny Bryan)

- Adopt a "project-oriented workflow'': https://www.tidyverse.org/articles/2017/12/workflow-vs-script/

(a.k.a., avoid arson)

- About pkg `here`: https://github.com/jennybc/here_here

Use `here` (not `setwd()`)

The fine print

`here::here()` figures out the top-level of your current project using some sane heuristics. It looks at working directory, checks a criterion and, if not satisfied, moves up to parent directory and checks again. Lather, rinse, repeat.

Here are the criteria. The order doesn't really matter because all of them are checked for each directory before moving up to the parent directory:

- Is a file named `.here` present?
- Is this an RStudio Project? Literally, can I find a file named something like `foo.Rproj`?
- Is this an R package? Does it have a `DESCRIPTION` file?
- Is this a remake project? Does it have a file named `remake.yml`?
- Is this a projectile project? Does it have a file named `.projectile`?
- Is this a checkout from a version control system? Does it have a directory named `.git` or `.svn`? Currently, only Git and Subversion are supported.

Use `here` — not `setwd()`

```
library(here)
```

Use `here` — not `setwd()`

```
library(here)
```

Use `here` — not `setwd()`

```r
library(here)
```

```r
path <- here("data", "anes_pilot_2016.csv")
path
```

Use `here` — not `setwd()`

```
library(here)
```

```
path <- here("data", "anes_pilot_2016.csv")
path
```

```
[1] "/Users/ryanmoore/Documents/github/winter-inst/data/anes_pilot_2016.csv"
```

Use `here` — not `setwd()`

```
library(here)
```

```
path <- here("data", "anes_pilot_2016.csv")
path
```

```
[1] "/Users/ryanmoore/Documents/github/winter-inst/data/anes_pilot_2016.csv"
```

```
anes <- read_csv(path)
head(anes, 2)
```

```
# A tibble: 2 x 594
  version caseid weight weight_spss follow turnout12 tu
  <chr>    <dbl>  <dbl>       <dbl>  <dbl>     <dbl>
1 ANES 2~      1  0.951       0.542      1         1
2 ANES 2~      2  2.67        1.52       2         2
# i 585 more variables: meet <dbl>, givefut <dbl>, info
#   sign <dbl>, give12mo <dbl>, compromise <dbl>, ftoba
#   ftwhite <dbl>, fthisp <dbl>, ftgay <dbl>, ftjeb <dl
```

`.Rproj` File

In top-level directory, create a `.Rproj` file for each project

`.Rproj` File

In top-level directory, create a `.Rproj` file for each project

Then, always start work by opening the `.Rproj` file.

`.Rproj` File

In top-level directory, create a `.Rproj` file for each project

Then, always start work by opening the `.Rproj` file.

- Starts fresh R
- Sets wd to project directory
- Opens files you were working on
- Restores other settings (prefs, data, etc.)

`.Rproj` File

`library(here)` will look for your `.Rproj` file.

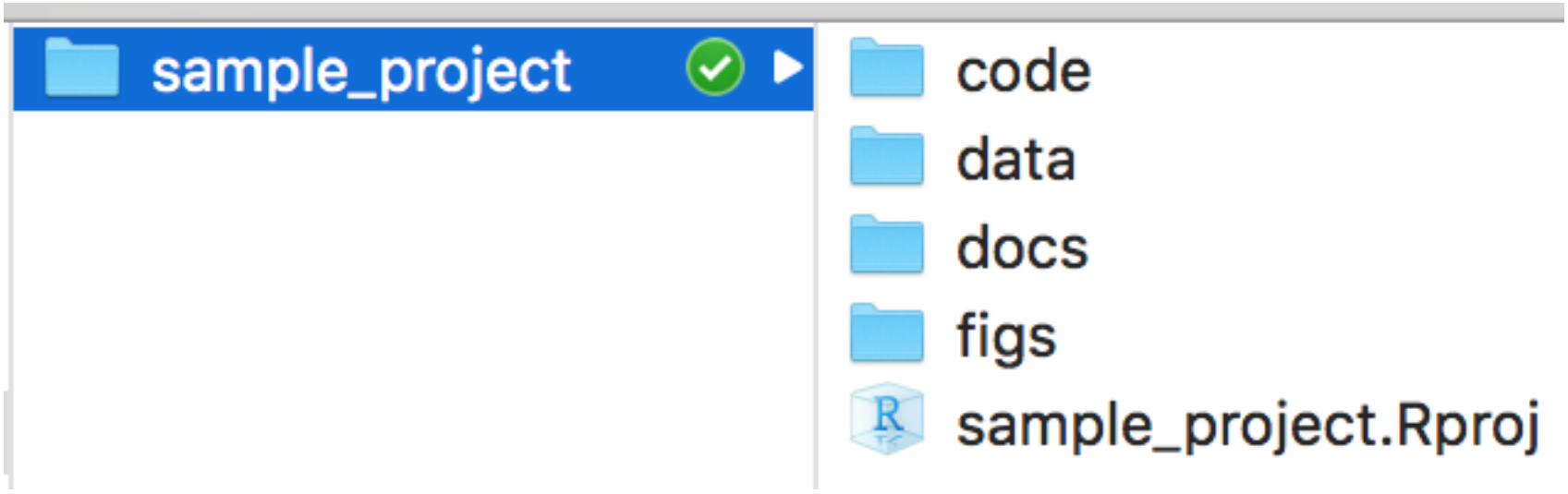
`.Rproj` File

`library(here)` will look for your `.Rproj` file.

That's (part) of how it knows the top-level dir.

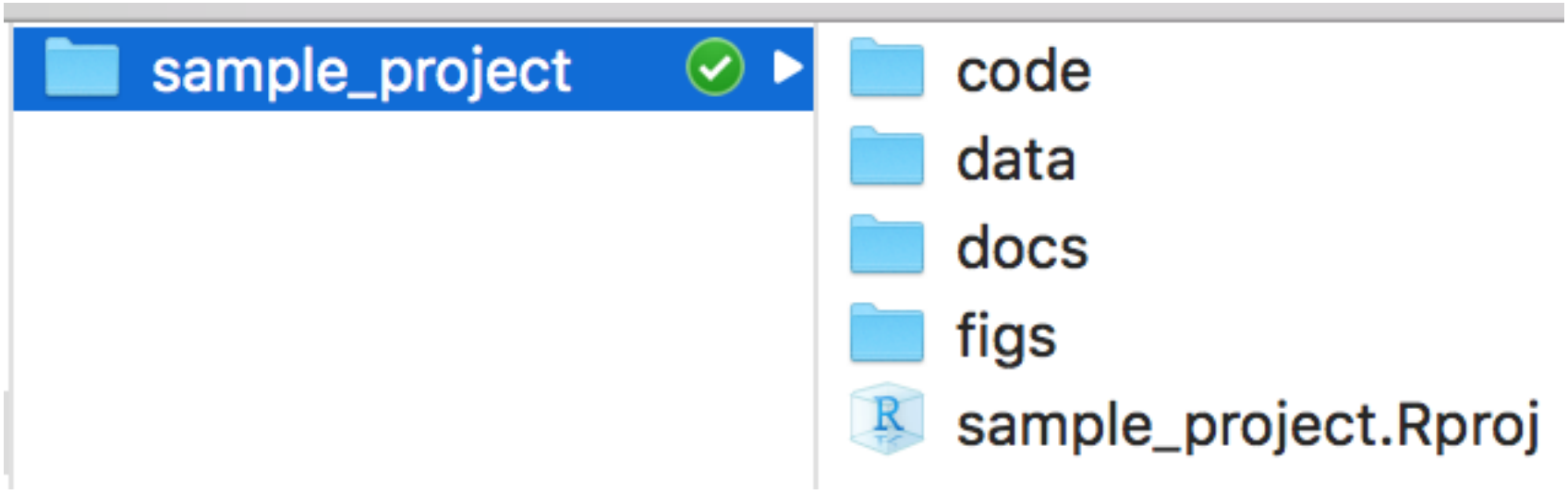
Structuring a Project

- Main project dir, subdirs:



Structuring a Project

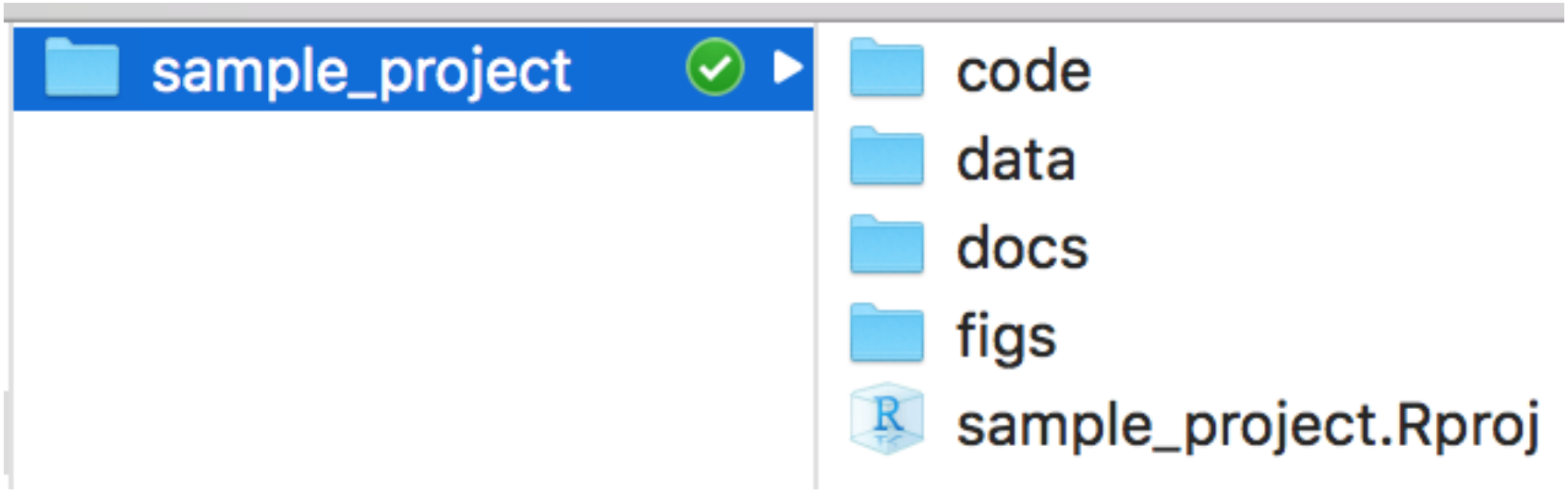
- Main project dir, subdirs:



- Click/`open` the `.Rproj` file in main project dir

Structuring a Project

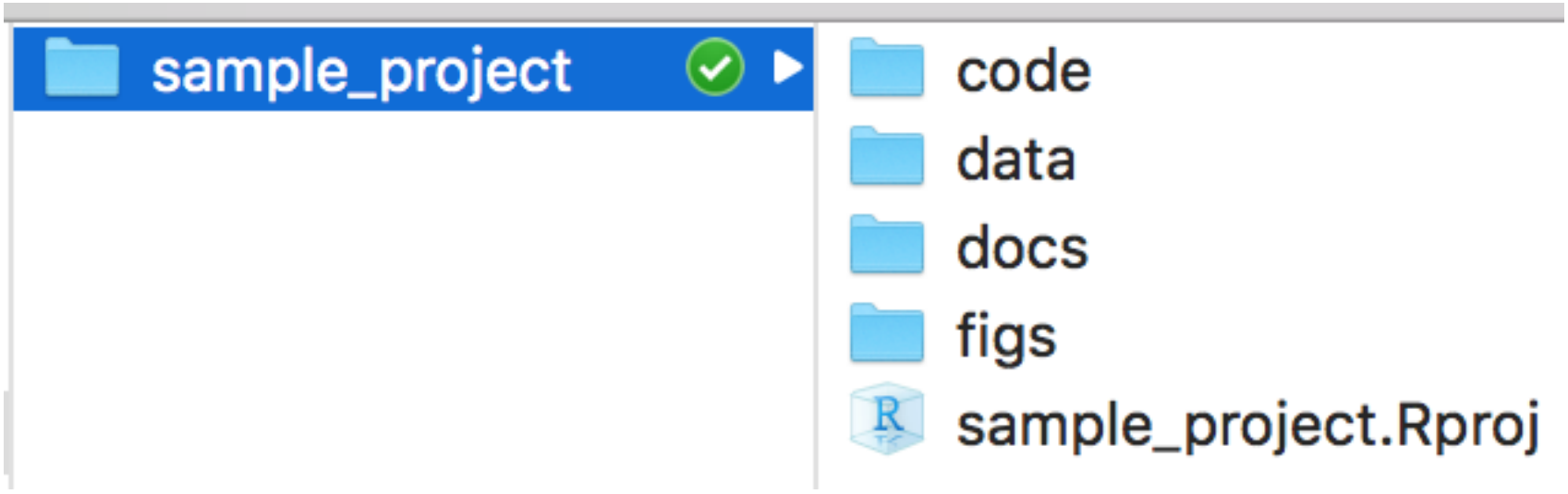
- Main project dir, subdirs:



- Click/`open` the `.Rproj` file in main project dir
- Opens files you had open

Structuring a Project

- Main project dir, subdirs:



- Click/`open` the `.Rproj` file in main project dir
- Opens files you had open
- Sets `.Rproj` dir as working dir

Structuring a Project

- In code files, paths are relative
- Best strategy: open `.Rproj`, use `here`

Structuring a Project

- In code files, paths are relative
- Best strategy: open `.Rproj`, use `here`

```
df_1 <- read_csv(here("data", "first_data.csv"))
df_2 <- read_csv(here("data", "second_data.csv"))
ggplot(df, aes(x, y)) + geom_point()
ggsave(here("figs", "myplot.pdf"))
```

Structuring a Project

- In code files, paths are relative

Structuring a Project

- In code files, paths are relative
- Why `here`? If open `.Rproj` or file at *same* level as `/data/`:

Structuring a Project

- In code files, paths are relative
- Why `here`? If open `.Rproj` or file at *same* level as `/data/`:

```
df_1 <- read_csv("data/first_data.csv")
df_2 <- read_csv("data/second_data.csv")
ggplot(df, aes(x, y)) + geom_point()
ggsave("figs/myplot.pdf")
```

Structuring a Project

- In code files, paths are relative
- Why `here`? If no `.Rproj`, open `.R` file from `/code/` (same-level dir):

Structuring a Project

- In code files, paths are relative
- Why `here`? If no `.Rproj`, open `.R` file from `/code/` (same-level dir):

```
df_1 <- read_csv("../data/first_data.csv")
df_2 <- read_csv("../data/second_data.csv")
ggplot(df, aes(x, y)) + geom_point()
ggsave("../figs/myplot.pdf")
```

Opening the `.Rproj` File

From http://j.mp/2QHeKDt:

When a project is opened within RStudio the following actions are taken:

- A new R session (process) is started
- The .Rprofile file in the project's main directory (if any) is sourced by R
- The .RData file in the project's main directory is loaded (if project options indicate that it should be loaded).
- The .Rhistory file in the project's main directory is loaded into the RStudio History pane (and used for Console Up/Down arrow command history).
- The current working directory is set to the project directory.
- Previously edited source documents are restored into editor tabs
- Other RStudio settings (e.g. active tabs, splitter positions, etc.) are restored to where they were the last time the project was closed.

`sample_project` Directory

- Walk through **`sample_project`** code in my **`research/sample_project/`**
- Create **`.Rproj`**
- Make, save, load data
- Create figures

But which code version is real?

But which code version is real?

My desktop?

But which code version is real?

My desktop? My laptop?

But which code version is real?

My desktop? My laptop?
Your desktop?

But which code version is real?

My desktop? My laptop?
Your desktop? ...?

The `HEAD` of `origin`, on GitHub?

The `HEAD` of `origin`, on GitHub?

Distributed version control.
(Agree on `origin`.)

For next time, see https://happygitwithr.com/index.html

2. Read §1
3. Create a (free) GitHub account, share login name w/ instructor §4
4. Complete §6 “Install Git”
5. Complete §7.0 “Introduce yourself” first paragraph
6. Follow §9 or §10 to set up either HTTPS PAT or SSH keys

(You can stop there.)