

RProjects

Creating a project-oriented workflow in R

Daniela Palleschi

2024-05-06

Table of contents

Installation requirements	2
Project-oriented workflow	2
Folder structure	2
RProjects	3
Creating a new Project	3
Opening a Project	4
Global RStudio options	4
Spot the differences	7
Spot the differences: RProject vs. None	7
Folder structure	7
data/	10
scripts/	10
Load in the data	10
here-package	12
The problem with <code>setwd()</code>	12
The benefit of <code>here()</code>	12
<code>here::here()</code>	13

Learning Objectives

Today we will...

- learn about project-oriented workflows

- create an RProject
- establish a self-contained project environment with `here`

Installation requirements

- required installations/recent versions of:
 - R
 - * version 4.4.0, “Puppy Cup”
 - * check current version with `R.version`
 - * download/update: <https://cran.r-project.org/bin/macosx/>
 - RStudio
 - * version 2023.12.1.402, “Ocean Storm”
 - * Help > Check for updates
 - * new install: <https://posit.co/download/rstudio-desktop/>

Project-oriented workflow

1. Folder structure:
 - keeping everything related to a project in one place
 - i.e., contained in a single folder, with subfolders as needed
2. Project-relative working directory
 - the project folder should act as your working directory
 - all file paths should be relative to this folder

Folder structure

- a core computer literacy skill
 - keep your Desktop as empty as possible
 - have a sensible folder structure
 - avoid mixing subfolders and files
 - * i.e., if a folder contains subfolders, ideally it should not contain files

RProjects

- in data analysis, using an IDE is beneficial
 - e.g., RStudio
- most IDEs have their own implementation of a Project
- in RStudio, this is the RProject
 - creates a `.Rproj` file in a project folder
 - stores project settings
- you can have several RProjects open simultaneously
 - and run several scripts across projects simultaneously
- most importantly, RProjects (can) centralise a specific project's workflow and file path

Creating a new Project

- when?
 - whenever you're starting a new course or project which will use R
- why?
 - to keep all the relevant materials in one place
- where?
 - somewhere that makes sense, e.g., a folder called `SoSe2024` or `Mastersarbeit`
- how?
 - `File > New Project > New Directory > New Project > [Directory name]`
`> Create Project`

New RProject

Create a new RProject for this course

- `File > New Project > New Directory > New Project > [Directory name]`
`> Create Project`
- make sure you choose a sensible location

Opening a Project

- to open a project, locate its .Rproj file and double-click
- or if you're already in RStudio, you can use the Project (None) drop-down (top right)

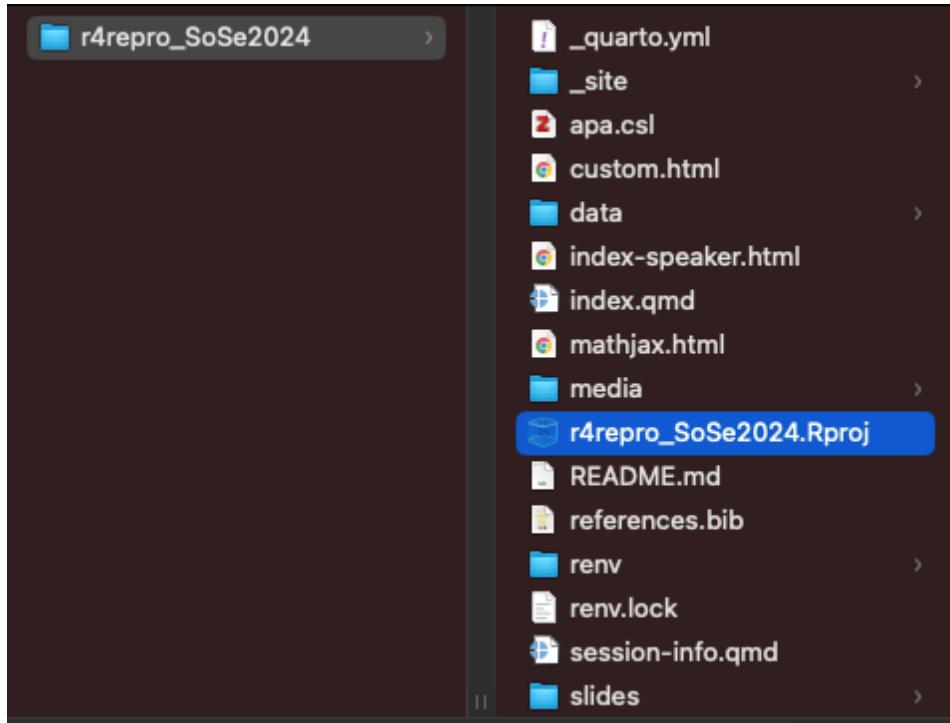


Figure 1: Double-click .Rproj

Global RStudio options

- Tools > Global Options
 - **Workspace:** Restore .RData into workspace at startup: NO
 - Save workspace to .RData on exit: Never
- this will ensure that you are always starting with a clean slate
 - and that your code is not dependent on some package or object you created in another session
- this is also how RMarkdown and Quarto scripts run
 - they start with an empty environment and run the script linearly

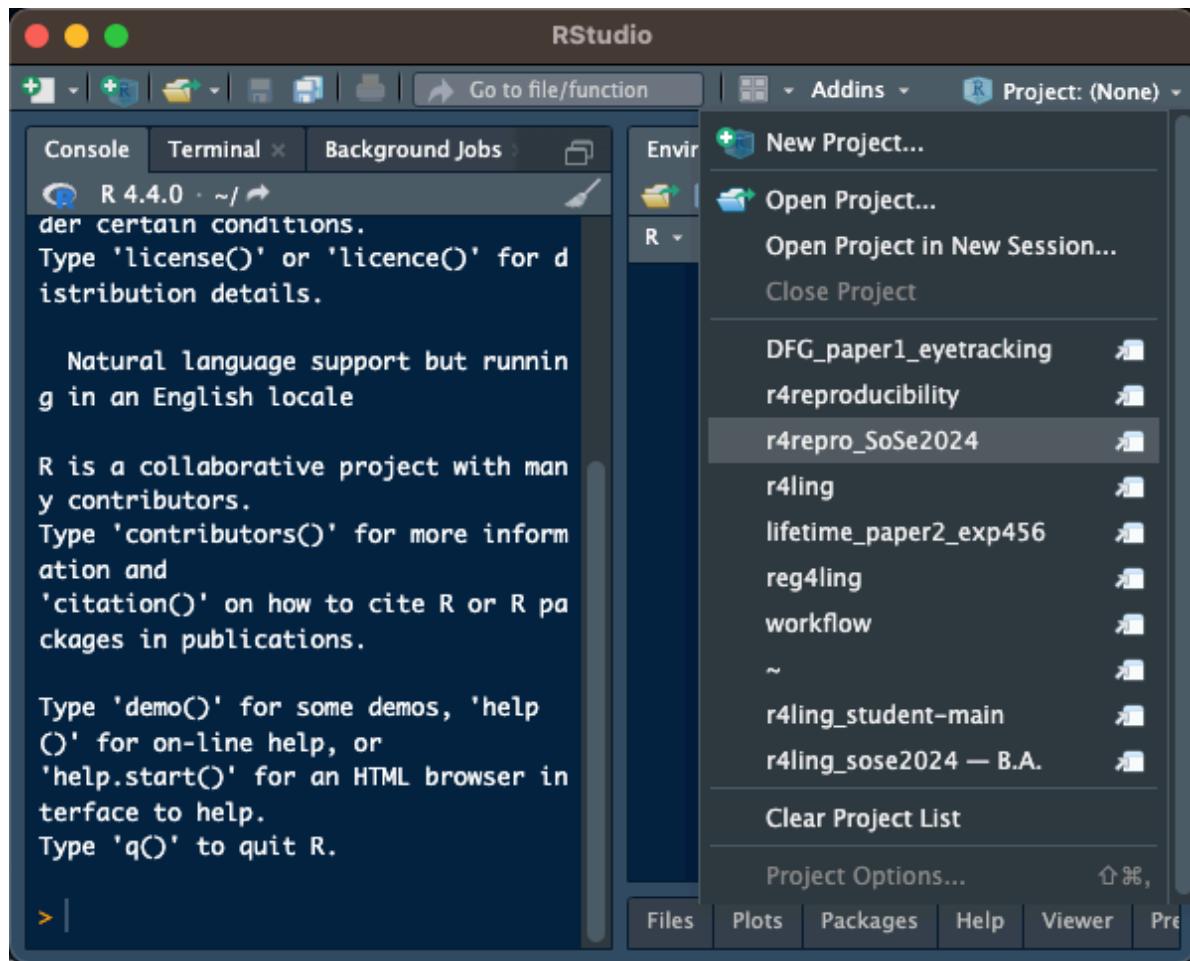


Figure 2: Open from RStudio

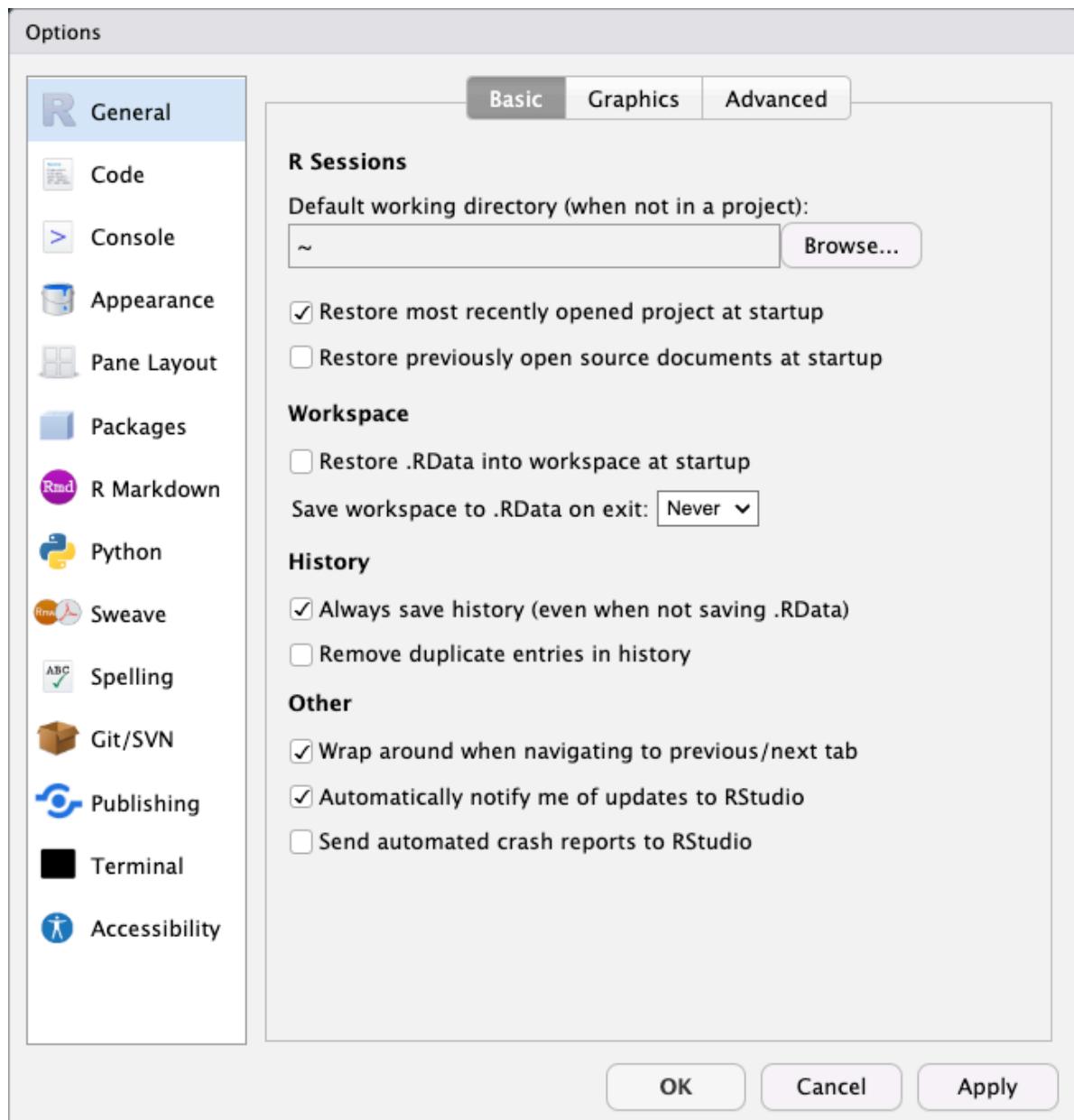


Figure 3: RStudio settings for reproducibility

💡 Global settings

Change your Global Options so that

- **Workspace:** Restore .RData into workspace at startup: NO
- Save workspace to .RData on exit: Never

Spot the differences

Spot the differences: RProject vs. None

Figure 1: RStudio Session A

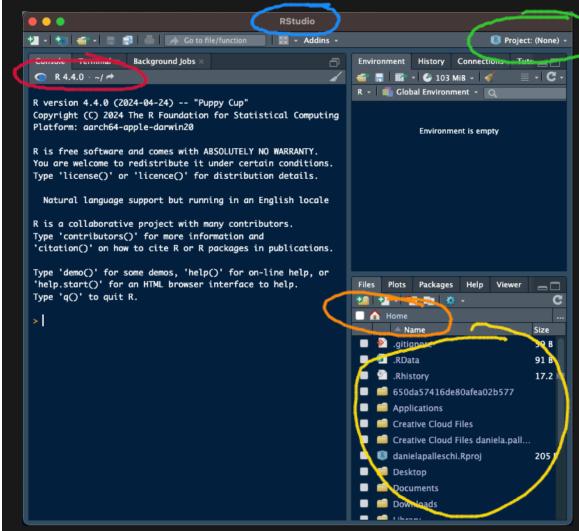
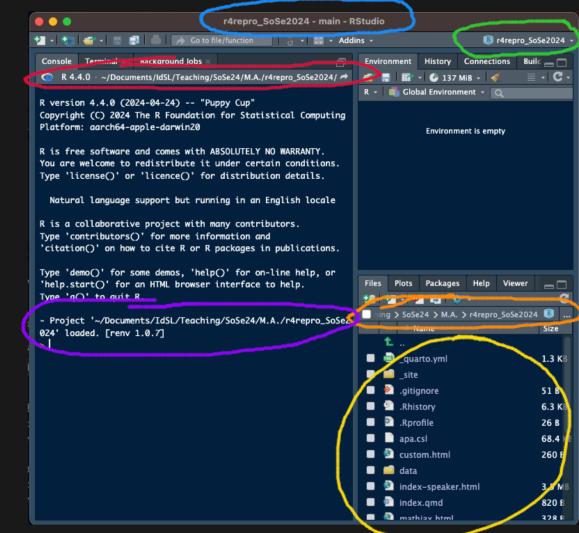


Figure 2: RStudio Session B



Folder structure

- some folders you'll typically want to have:
 - **data:** containing your dataset(s)
 - **scripts** (or **analyses**, etc.): containing any analysis scripts
 - **manuscript:** containing any write-ups of your results
 - **materials:** containing relevant experiment materials (e.g., stimuli)
- let's just create the first 2 (**data** and **scripts**)

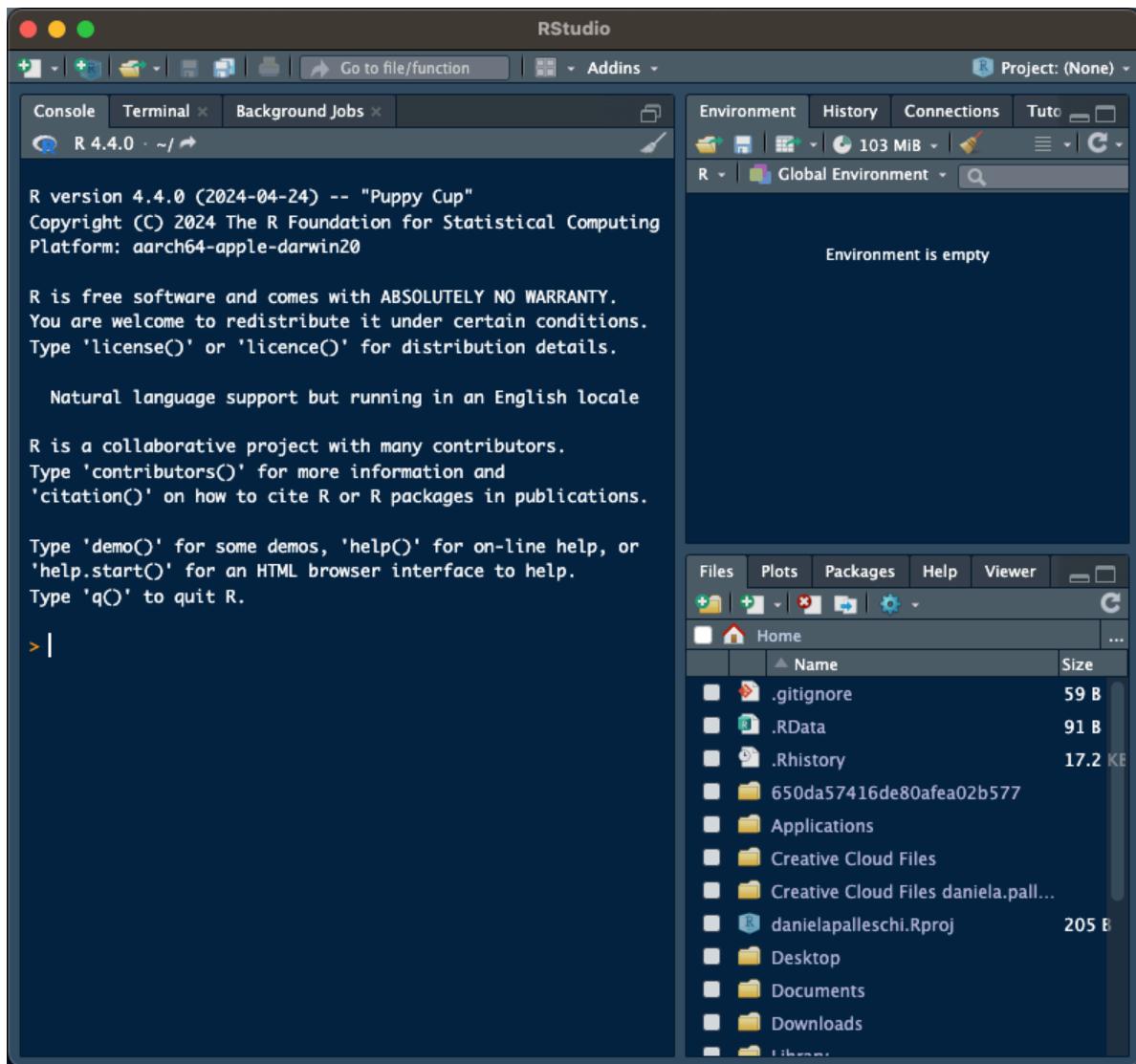


Figure 4: RStudio Session A

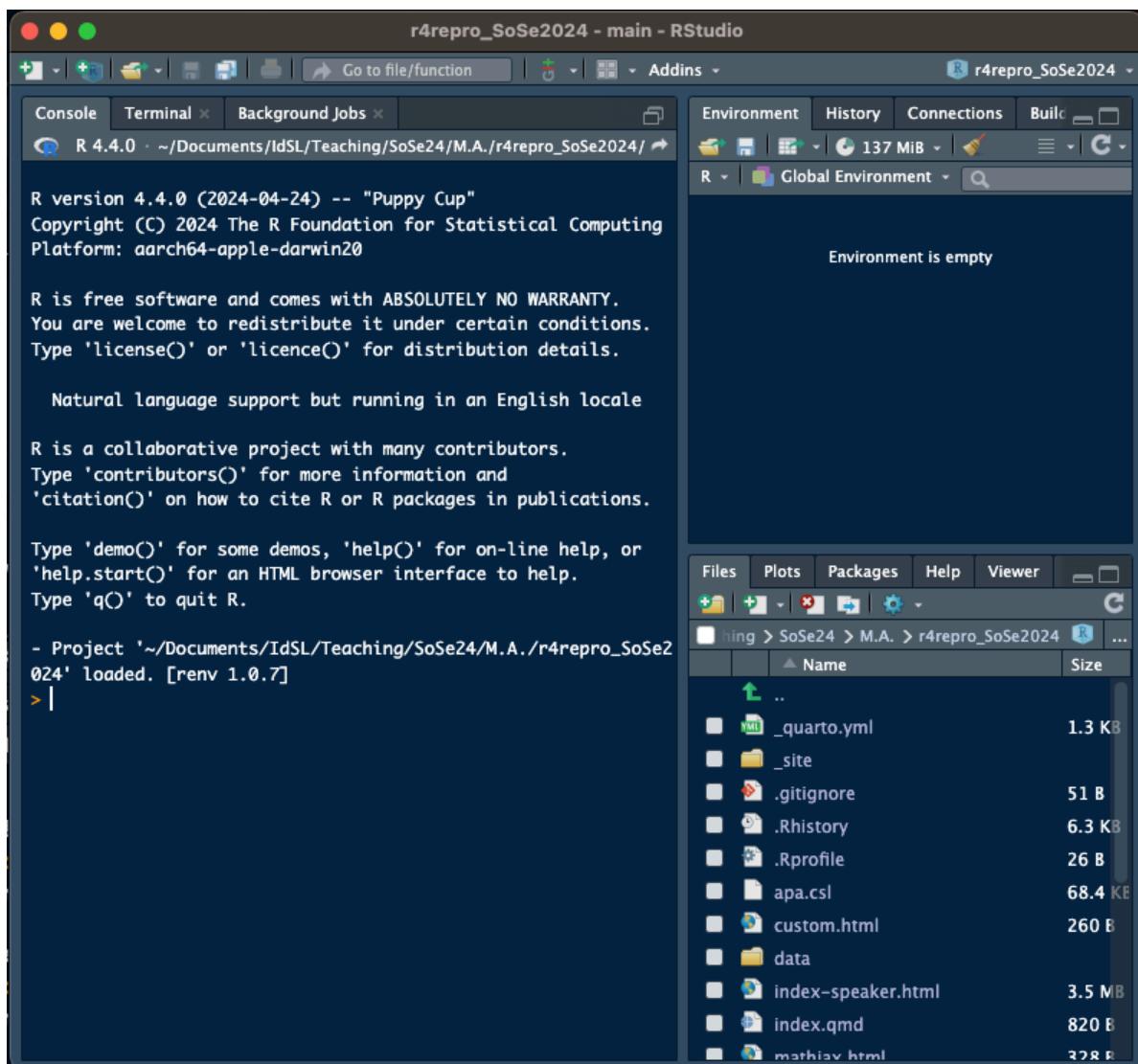


Figure 5: RStudio Session B

data/

- do you have “raw”, i.e., pre-processed data?
 - if so, you might want to create a `raw` sub-folder
 - and any other relevant sub-folders (e.g., `processed` or `tidy`)
- download the dataset on this week’s Moodle section
 - *or*, move a dataset of your own to this folder

scripts/

- try to create a single script for each “product”
 - e.g., anonymised data, ‘cleaned’ data, data exploration, visualisation, analyses, etc.
- you can create sub-folders as the project develops and move scripts around
 - for now, let’s create a new script to take a look at our data

New script

Create a new Quarto script:

1. `File > New File > Quarto Document`
2. Add a title
3. Uncheck the `Use Visual Editor` box
4. Click `Create`
5. Save it in your `scripts/` folder: `File > Save as...`

Load in the data

- load in the data however you normally would
 - e.g., `readr::read_csv()`



Figure 6: Illustration by [Allison Horst](#)

here-package

- `here` package (Müller, 2020) enables file referencing
 - avoids the use of `setwd()`

The problem with `setwd()`

If the first line of your R script is

```
setwd("C:\Users\jenny\path\that\only\I\have")
```

I will come into your office and SET YOUR COMPUTER ON FIRE .

— [Jenny Bryan](#)

- `setwd()` depends on your entire machine's folder structure
- `setwd()` breaks when you
 - send your project folder to a collaborator
 - make your analyses open
 - change the location of your project folder
- using slashes is also dependent on your operating system

The benefit of `here()`

- uses the top-level directory of your project as the working directory
- can separate folder names with a comma

💡 here

Load the dataset using `here`

1. Install `here` (e.g., `install.packages("here")`)
2. Load `here` at the beginning of your package
 - or use `here:::` before calling a function
3. Use the `here()` function to load in your data
4. Inspect the dataset however you usually would (e.g., `summary()`, `names()`, etc.)
5. Save your script

```
here::here()
```

- install package

Listing 1 In the Console

```
install.packages("here")
```

- load package and call the `here` function

```
# load package
library(here)

# read in data
df_data <- read.csv(here("data", "data_lifetime_pilot.csv"))
```

- or directly call the `here` function without loading the package

```
# read in data without loading here
df_data <- read.csv(here::here("data", "data_lifetime_pilot.csv"))
```



Reproduce your analysis

1. Make sure you save your script, then close your Rproject.
2. Re-open the project. Can you re-run the script?

Learning objectives

Today we learned...

- learn about project-oriented workflows
- create an RProject
- establish a self-contained project environment with `here`

References

- Bryan, J., Hester, J., Pileggi, S., & Aja, D. E. (n.d.). *What They Forgot to Teach You About R*. <https://rstats.wtf/>.
- Bryan, J., & TAs, T. S. 545. (n.d.). *Chapter 2 R basics and workflows / STAT 545*.
- Müller, K. (2020). *Here: A simpler way to find your files*. <https://CRAN.R-project.org/package=here>
- Using RStudio Projects. (2024). In *Posit Support*. <https://support.posit.co/hc/en-us/articles/200526207-Using-RStudio-Projects>.