Day 1, Part 3: Import Your Data

Brennan Terhune-Cotter and Matt Dye

Agenda

1. Importing Data
2. Finding your files for import
3. Local paths ¶

Importing Data

Importing data using `readr` or `readxl`

- Importing data from Excel spreadsheets is easy with `read_xlsx()` in the package `readxl::`
- Importing data from csv or text files are easy with `read_...` functions in `readr::`
  - Depends on what the delimiter is:

function	delimiter	typical suffix
`read_table`	white space	.txt
`read_csv`	comma	.csv
`read_csv2`	semicolon	.csv
`read_tsv`	tab	.tsv
`read_delim`	any; must define delimiter	.txt

Cleaning Variable Names

> **Tip**
>
> `janitor::clean_names` will automatically "clean" the names of a dataset to make them more usable!

```
1 eruptions <- readxl::read_xlsx("../../data/holocene_eruptions.xlsx")
2 names(eruptions)
```

```
 [1] "Volcano Number"       "Volcano Name"         "Country"
 [4] "Primary Volcano Type" "Activity Evidence"    "Last Known Eruption"
 [7] "Region"               "Subregion"            "Latitude"
[10] "Longitude"            "Elevation (m)"        "Dominant Rock Type"
[13] "Tectonic Setting"
```

```
1 eruptions <- janitor::clean_names(eruptions)
2 names(eruptions)
```

```
 [1] "volcano_number"       "volcano_name"         "country"
 [4] "primary_volcano_type" "activity_evidence"    "last_known_eruption"
 [7] "region"               "subregion"            "latitude"
[10] "longitude"            "elevation_m"          "dominant_rock_type"
[13] "tectonic_setting"
```

Cleaning Variable Names

- Usually, you will want to rename variables even if they've been cleaned with `janitor::`.
- You do this with the function `rename()` in the `dplyr` package:

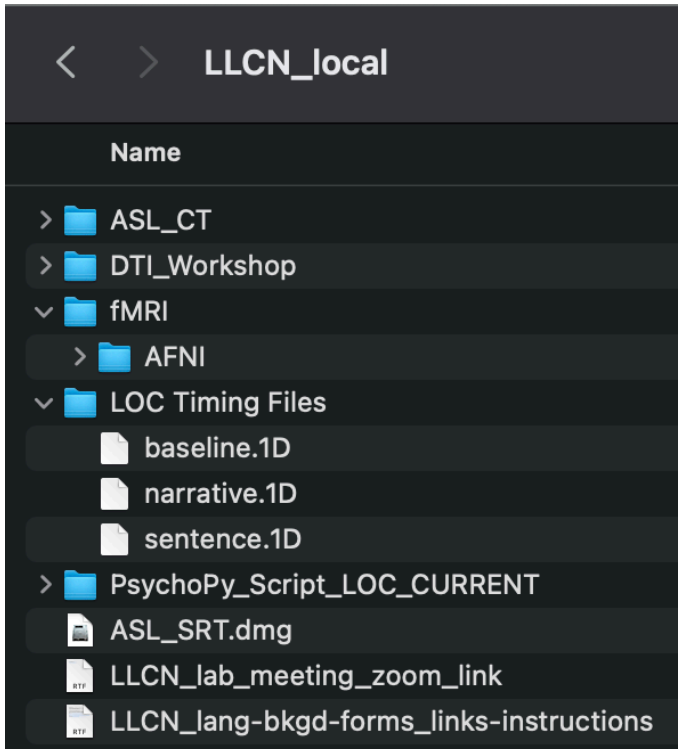
```
new_df <- dplyr::rename(old_df,
                  new_name1 = old_name1,
                  new_name2 = old_name2,
                  new_name3 = old_name3)
```

Finding your files!

Caution

Importing files into your R workspace for the first time can be difficult because you have to **know the current file paths to the files you want!**

The filepaths behind your Finder window



/Desktop/LLCN_local

/Desktop/LLCN_local/ASL_CT

/Desktop/LLCN_local/DTI_Workshop

/Desktop/LLCN_local/fMRI

/Desktop/LLCN_local/fMRI/AFNI

/Desktop/LLCN_local/LOC Timing Files

/Desktop/LLCN_local/LOC Timing Files/baseline.1D

/Desktop/LLCN_local/LOC Timing Files/narrative.1D

/Desktop/LLCN_local/LOC Timing Files/sentence.1D

/Desktop/LLCN_local/PsychoPyScript_LOC_CURRENT

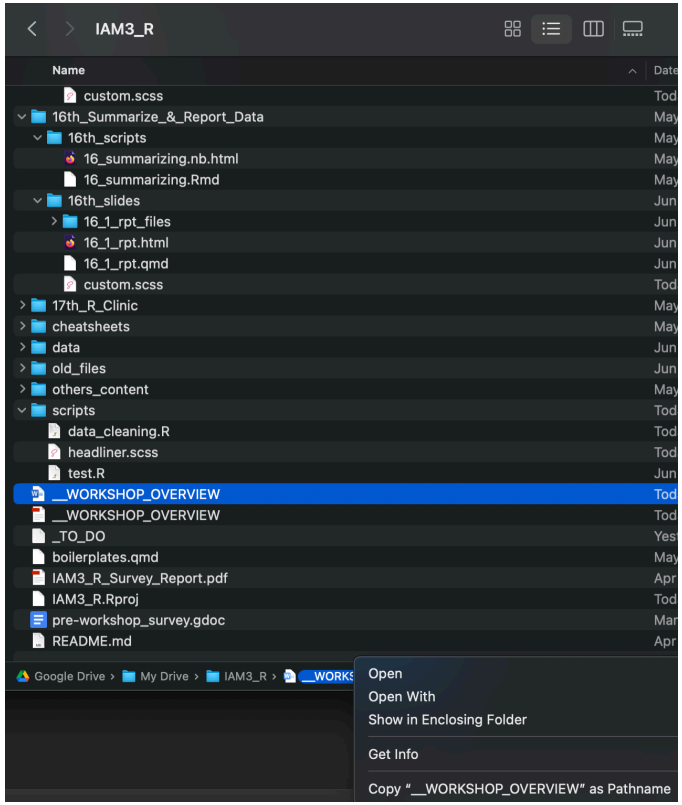
/Desktop/LLCN_local/ASL_SRT.dmg

/Desktop/LLCN_local/LLCN_lab_meeting_zoom_link.rtf

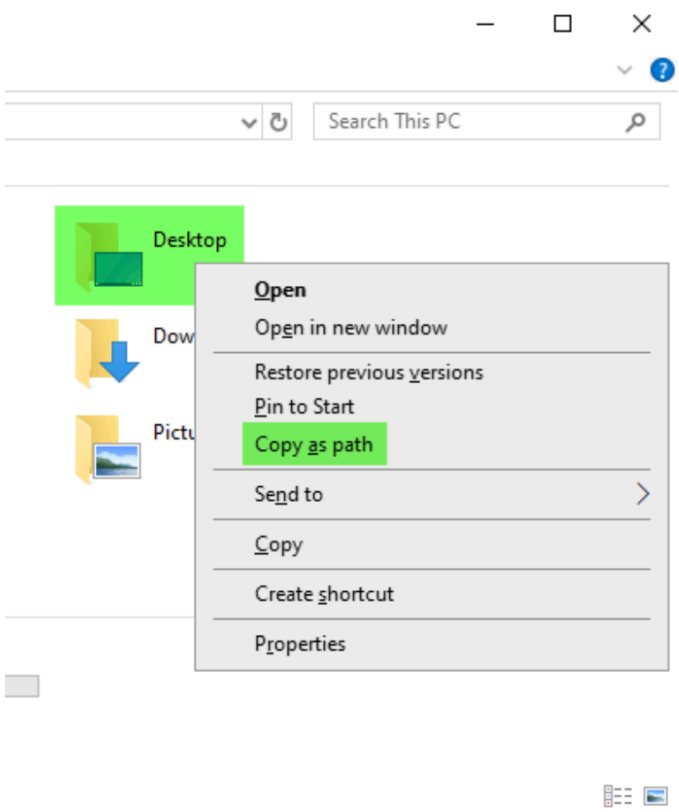
/Desktop/LLCN_local/LLCN_lang-bkgd-forms_links-instructions.rtf

Finding the absolute file path of a file

There are shortcuts to finding the absolute file path of a file and copying it!



in Finder, select file & right-click on its name in the bottom of the window



in Explorer, right-click path and “copy as path”

Mac vs. PC

Macs and PCs format file paths differently.

Macs use slashes: /

PCs use backslashes: \

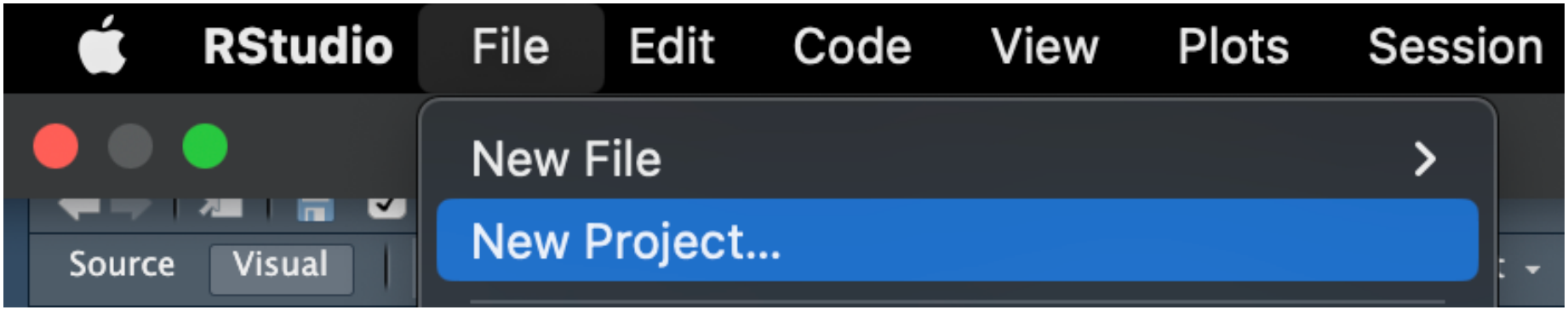
I will be using Mac convention as I have a Mac :)

PC users, please remember to use backslashes instead of slashes!

R Projects and Local Paths

R Projects

- An R project is a file which saves your **workspace**
- It is useful to have an .Rproj file for each project you have
- When you open the .Rproj, everything will look like the last time you saved it.
- Instead of opening RStudio or individual R files (like any other application):
  1. Open your .Rproj
  2. Open files from that .Rproj ¶



Using local paths to stay within a folder

- All project-specific code should be written to work within the project's folder *no matter where the folder is*
- This means using local paths to load data and refer to other files within the folder
  - No slash in the beginning = local path (starts from working directory)
  - Slash in beginning = absolute path (starts from home directory)
  - `./` = current directory
  - `../` = go one directory "up"
- That project file (or analysis file that your scripts are in) will be your *working directory ¶*

```
localFilePath = "scripts/data_cleaning.R"
absoluteFilePath = "/Users/brennanwork/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My
```

Working Directories

- Your *working directory* is where you run your scripts "from".
- You can temporarily change your working directory with `setwd()` ¶

```
1 getwd() # check your current working directory
```

```
[1] "/Users/brennanwork/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My
Drive/IAM3_R/IAM3_2023_R/12th_Introduction/12th_slides"
```

```
1 setwd("../") # change your working directory ("." refers to current directory)
2 getwd() # check it again
```

```
[1] "/Users/brennanwork/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My
Drive/IAM3_R/IAM3_2023_R/12th_Introduction"
```

Which folder is my working directory?

- The answer depends on how you run your code and how you opened the session:
  - If you run your code directly, and…
    - if you opened an .Rproj, then **it is that .Rproj's folder**
    - if you opened a new session of R or opened a .R file directly, then **it is your home folder** :(
  - If you run the code in a script by using `source()`, then **it is that script's folder ¶**

> If you are just starting out with R, just save your .Rproj and all .R scripts to one folder. That folder will always be your working directory if you open R via your .Rproj :)
>
> Don't worry about using `source()` at this point!

Writing Local Paths

- project_folder
  - data
    - data.xlsx
  - analysis
    - project_analysis.R
    - project.Rproj

In the script `project_analysis.R`, opened in `project.Rproj`, how would you open `data.xlsx`?

```
1 open_xlsx("../data/data.xlsx")
```

The working directory would be `project_folder/analysis`, so you would go *up* one folder then back down into the data folder.

Writing Local Paths

- project_folder
  - project.Rproj
  - data
    - data.xlsx
  - analysis
    - project_analysis.R

In the script `project_analysis.R`, opened in `project.Rproj`, how would you open `data.xlsx`?

```
# if you run code directly
open_xlsx("./data/data.xlsx") # OR
open_xlsx("data/data.xlsx")

# if you run the script in a source() call
open_xlsx("../data/data.xlsx")
```

The working directory would be in the same folder as project.Rproj when running code within the project.

If you source the script, the working directory would be in the `analysis folder`.

Next up... Lab 1 and OYOLab!

In Lab 1:

1. Follow the instructions in 12_1_begin.Rmd!

In OYOLab:

1. Create a folder (or folders) where you will put your R scripts.
2. Create an .Rproj for each folder.
3. Open a new R Script file.
4. Copy and paste the code for importing your data from 12_1_begin.Rmd and try running it.
    - Don't forget to fix the filepath and/or move the file with your data to the right place!
5. Save your R script.
6. You now have your first (or second, third, or hundredth) R script! ¶