

Data I/O

Data Wrangling in R

R Basics

Explaining output on slides

In slides, a command (we'll also call them code or a code chunk) will look like this

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

And then directly after it, will be the output of the code.

These slides were made in R using **knitr** and **R Markdown** (covered later today when we discuss reproducible research)

R variables

A few reminders:

- You can create variables from within the R environment and from files on your computer
- Use "<-" to assign values to a variable name
- Variable names are case-sensitive, i.e. X and x are different

```
x <- 2  
x
```

```
[1] 2
```

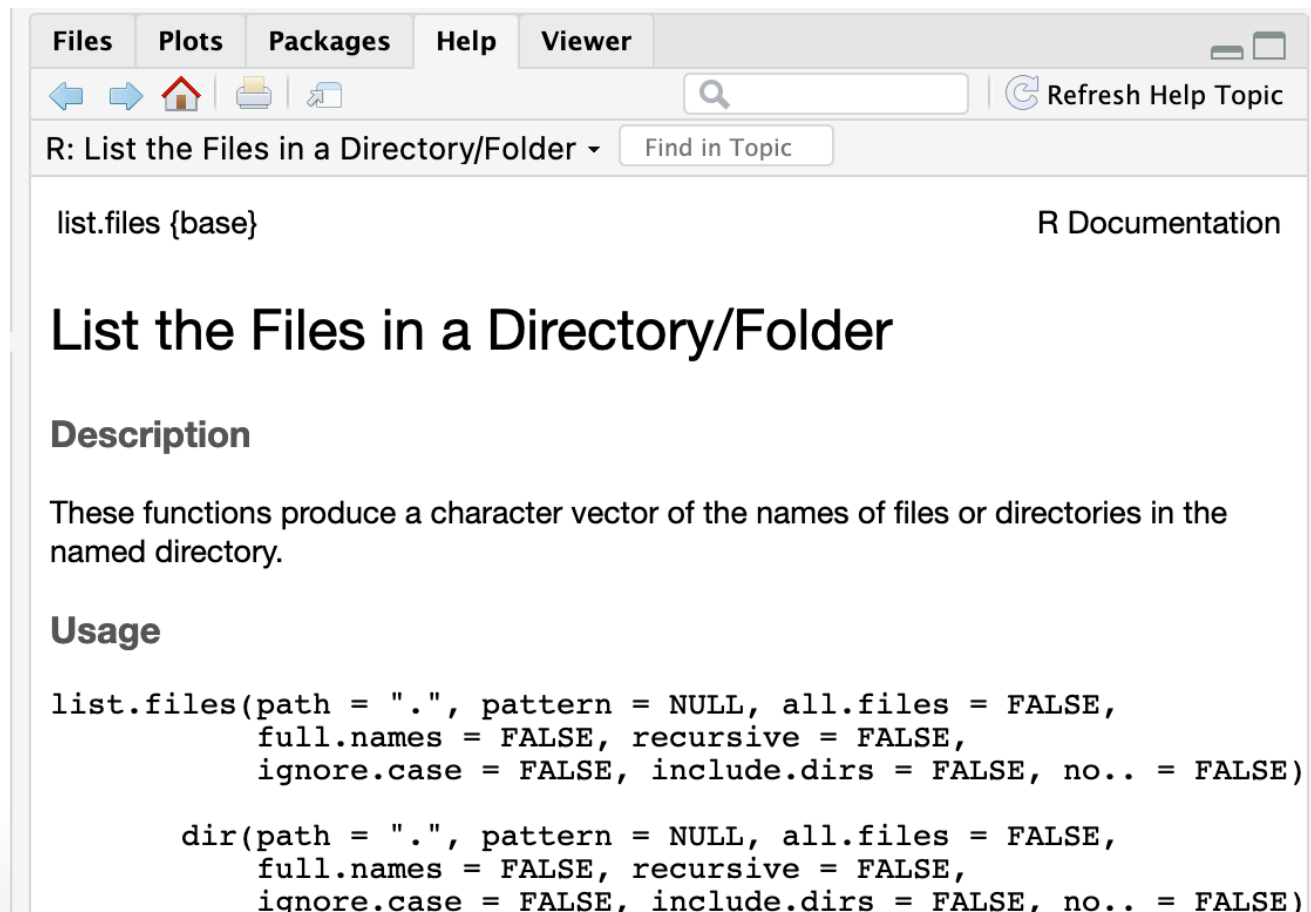
```
x * 4
```

```
[1] 8
```

Help

For any function, you can write `?FUNCTION_NAME`, or `help("FUNCTION_NAME")` to look at the help file:

```
?dir  
help("dir")
```



The screenshot shows the R Help Viewer window. The title bar includes tabs for Files, Plots, Packages, Help, and Viewer. The Help tab is active, displaying the documentation for the 'list.files' function. The window title is 'R: List the Files in a Directory/Folder'. The main content area shows the function signature 'list.files {base}' and the title 'List the Files in a Directory/Folder'. Below the title is the 'Description' section, which states: 'These functions produce a character vector of the names of files or directories in the named directory.' The 'Usage' section follows, showing the function signatures for 'list.files' and 'dir' with their default arguments.

Files Plots Packages Help Viewer

R: List the Files in a Directory/Folder

list.files {base} R Documentation

List the Files in a Directory/Folder

Description

These functions produce a character vector of the names of files or directories in the named directory.

Usage

```
list.files(path = ".", pattern = NULL, all.files = FALSE,  
           full.names = FALSE, recursive = FALSE,  
           ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)  
  
dir(path = ".", pattern = NULL, all.files = FALSE,  
    full.names = FALSE, recursive = FALSE,  
    ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)
```

Packages

Not all packages are available by default.

```
install.packages("tidyverse")  
library(tidyverse)
```

```
install.packages("light")
```



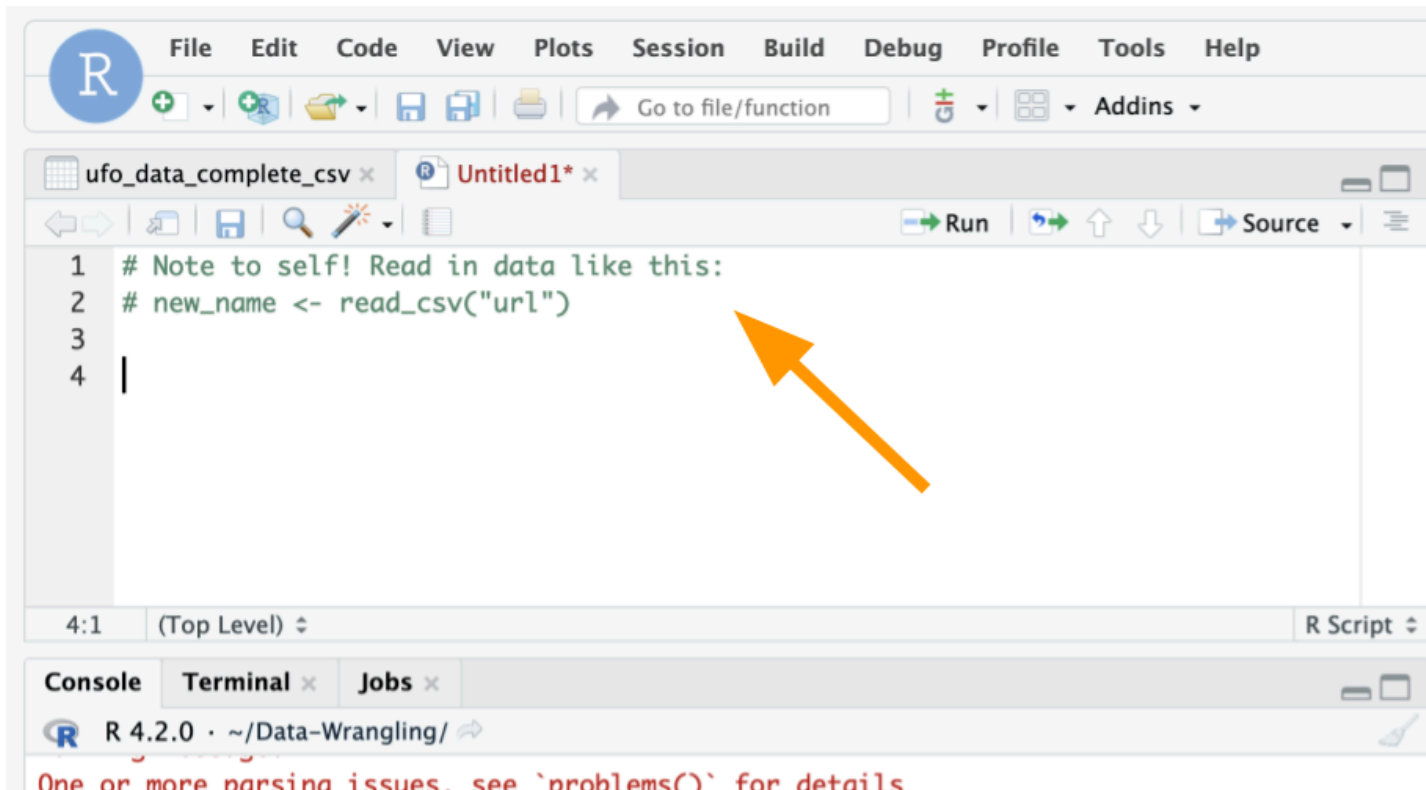
```
library("light")
```



Images sourced from <https://www.wikihow.com/Change-a-Light-Bulb>

Commenting in Scripts

Commenting in code is super important. You should be able to go back to your code years after writing it and figure out exactly what the script is doing. Commenting helps you do this. Also handy for notes!



The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor window has two tabs: 'ufo_data_complete_csv' and 'Untitled1*'. The 'Untitled1*' tab is active, showing a script with four lines of code. Line 1 is a comment: '# Note to self! Read in data like this:'. Line 2 is a comment: '# new_name <- read_csv("url")'. Line 3 is empty. Line 4 has a single vertical bar character '|'. An orange arrow points from the right side of the editor towards the comment on line 2. The status bar at the bottom shows '4:1 (Top Level)' and 'R Script'. Below the editor is a console window with tabs for Console, Terminal, and Jobs. The console shows the R version 'R 4.2.0' and the path '~/Data-Wrangling/'. A red message at the bottom of the console reads: 'One or more parsing issues. see `problems()` for details'.

```
1 # Note to self! Read in data like this:
2 # new_name <- read_csv("url")
3
4 |
```

Commenting in Scripts



avahoffman Add code to save discarded outliers in a csv

1 contributor

127 lines (108 sloc) 4.16 KB

```
1 # Search for outliers among biomass subplots in preparation for the rest of the analysis
2 #####
3 library(dplyr)
4 library(ggplot2)
5 library(cowplot)
6
7 # Useful information here: http://r-statistics.co/Outlier-Treatment-With-R.html
8 #####
9
10 make_outlier_plot <-
11   function(d) {
12     # This function will test for chi-square scores that are outside the
13     # percentile cutoff, and color them blue.
14     # For best results, use only on a specific site-category-treatment subset
15     # Probably best for viz only!!
16     ggplot() +
17       geom_point(aes(
18         x = as.numeric(rownames(d)),
19         y = d$biomass
```


Data Input

Outline

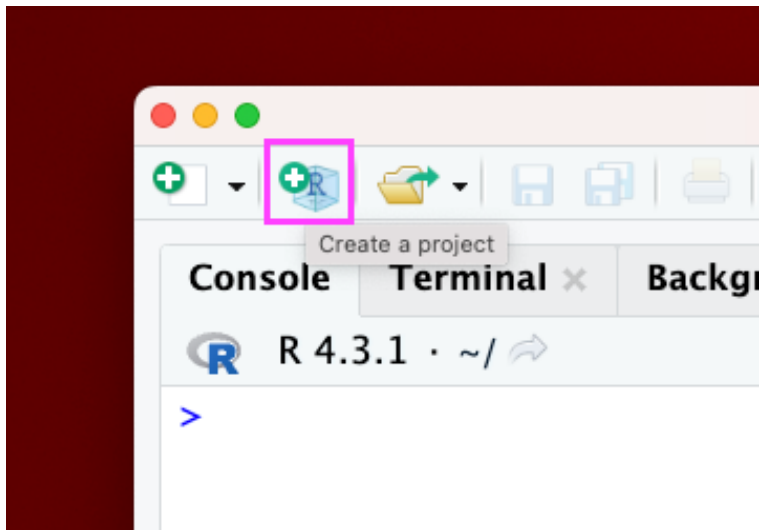
- Part 0: A little bit of set up!
- Part 1: reading in manually (point and click)
- Part 2: reading in directly & working directories
- Part 3: checking data & multiple file formats

Part 0: Setup - R Project

New R Project

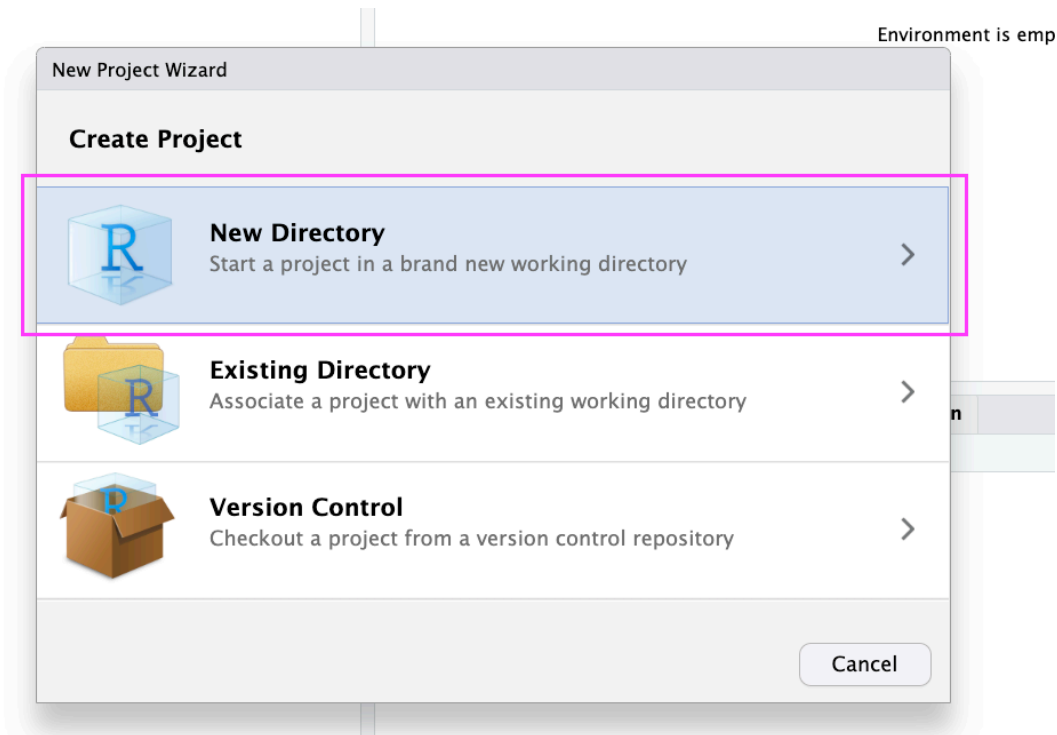
Let's make an R Project so we can stay organized in the next steps.

Click the new R Project button at the top left of RStudio:



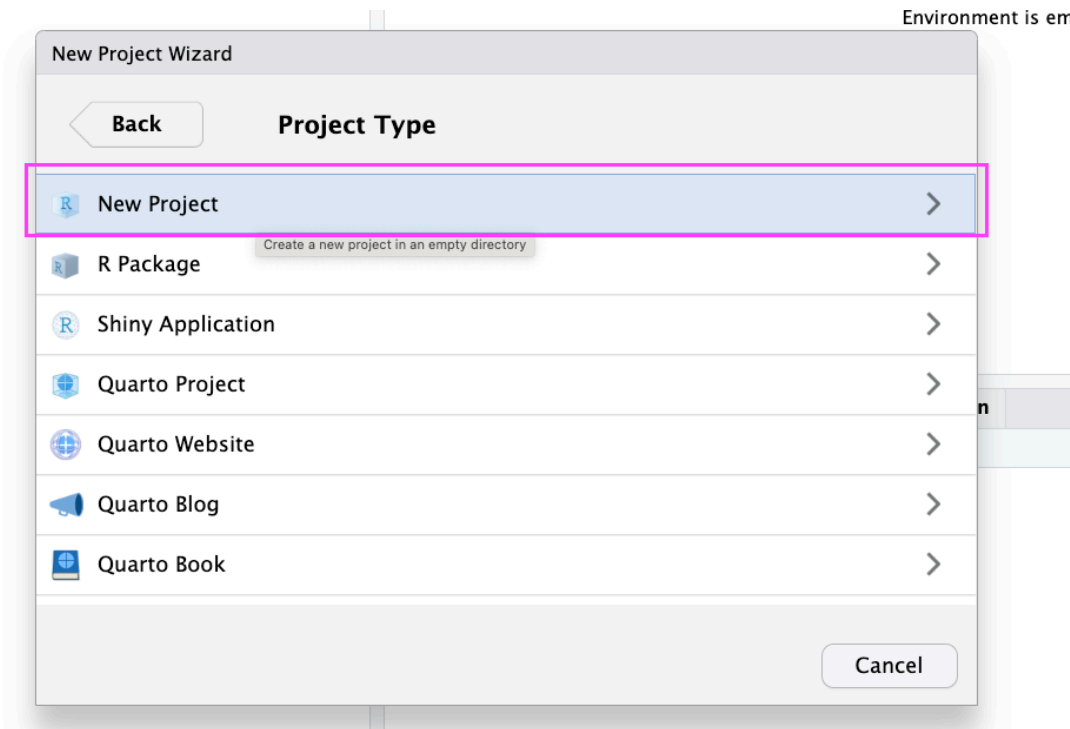
New R Project

In the New Project Wizard, click “New Directory”:



New R Project

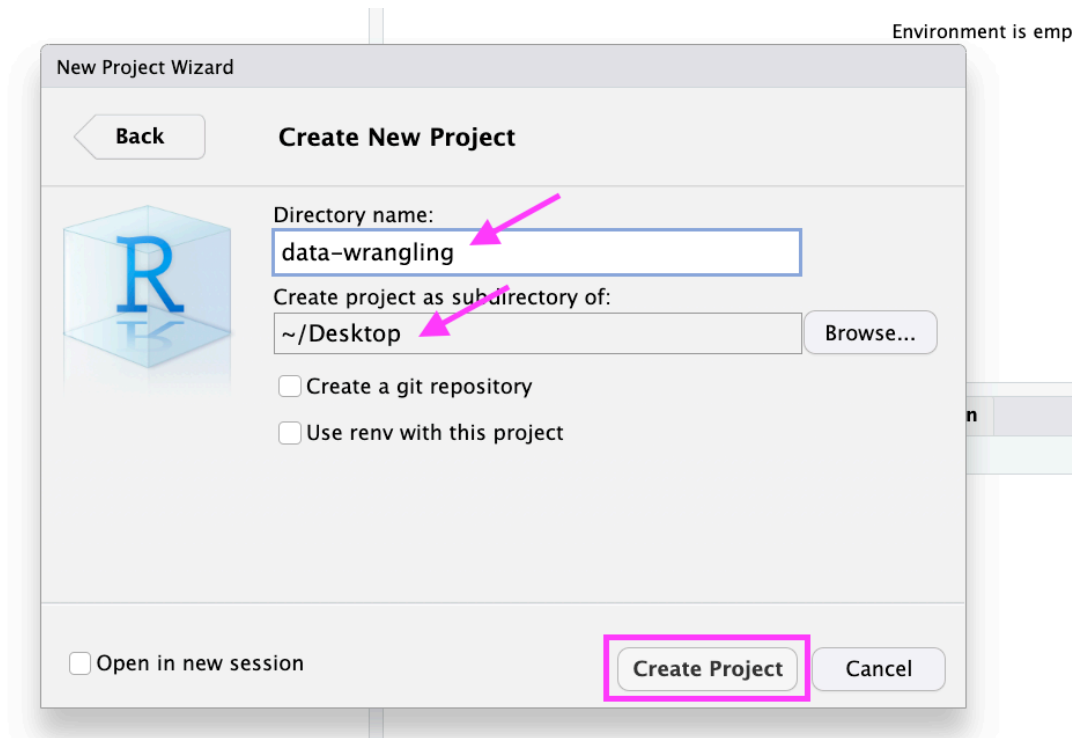
Click “New Project”:



New R Project

Type in a name for your new folder.

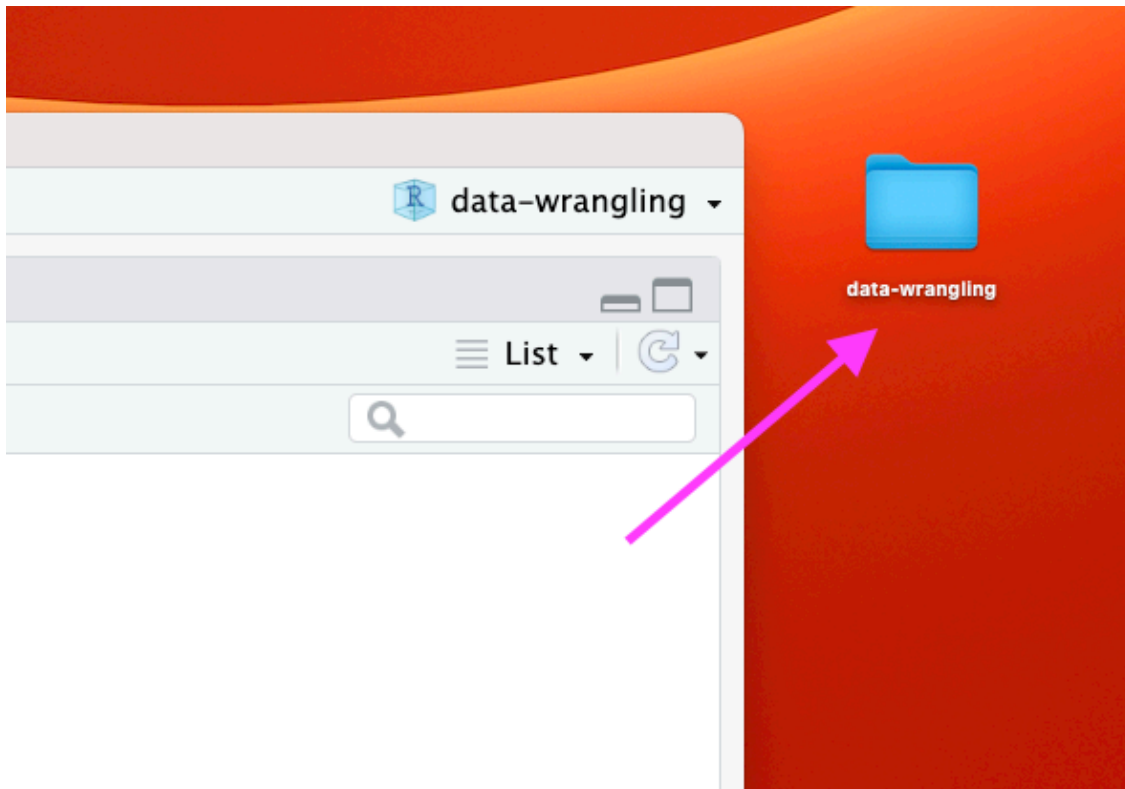
Store it somewhere easy to find, such as your Desktop:



New R Project

You now have a new R Project folder on your Desktop!

Make sure you add any scripts or data files to this folder as we go through today's lesson. This will make sure R is able to "find" your files.



Why Projects?

R Projects are a super helpful feature of RStudio. They help you:

- **Stay organized.** R Projects help in organizing your work into self-contained directories (folders), where all related scripts, data, and outputs are stored together. This organization simplifies file management and makes it easier to locate and manage files associated with your analysis or project.
- **Find the right files.** When you open an R Project, RStudio automatically sets the working directory to the project's directory. This is where RStudio “looks” for files. Because it's always the Project folder, it can help avoid common issues with file paths.
- **Be more reproducible.** You can share the entire project directory with others, and they can replicate your environment and analysis without much hassle.

Part 1: Getting data into R (manual/point and click)

Data Input

- 'Reading in' data is the first step of any real project/analysis
- R can read almost any file format, especially via add-on packages
- We are going to focus on simple delimited files first
 - comma separated (e.g. '.csv')
 - tab delimited (e.g. '.txt')
 - Microsoft Excel (e.g. '.xlsx')

Data Input

UFO dataset:

“This dataset contains over 80,000 reports of UFO sightings over the last century. Inspiration includes What areas of the country are most likely to have UFO sightings? Are there any trends in UFO sightings over time? Do they tend to be clustered or seasonal? Do clusters of UFO sightings correlate with landmarks, such as airports or government research centers? What are the most common UFO descriptions?”

- Check out the data at: <https://www.kaggle.com/datasets/NUFORC/ufo-sightings>

Data Input: Dataset Location

Dataset is located at https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv

- Download data by clicking the above link
 - Safari - if a file loads in your browser, choose File → Save As, select, Format “Page Source” and save

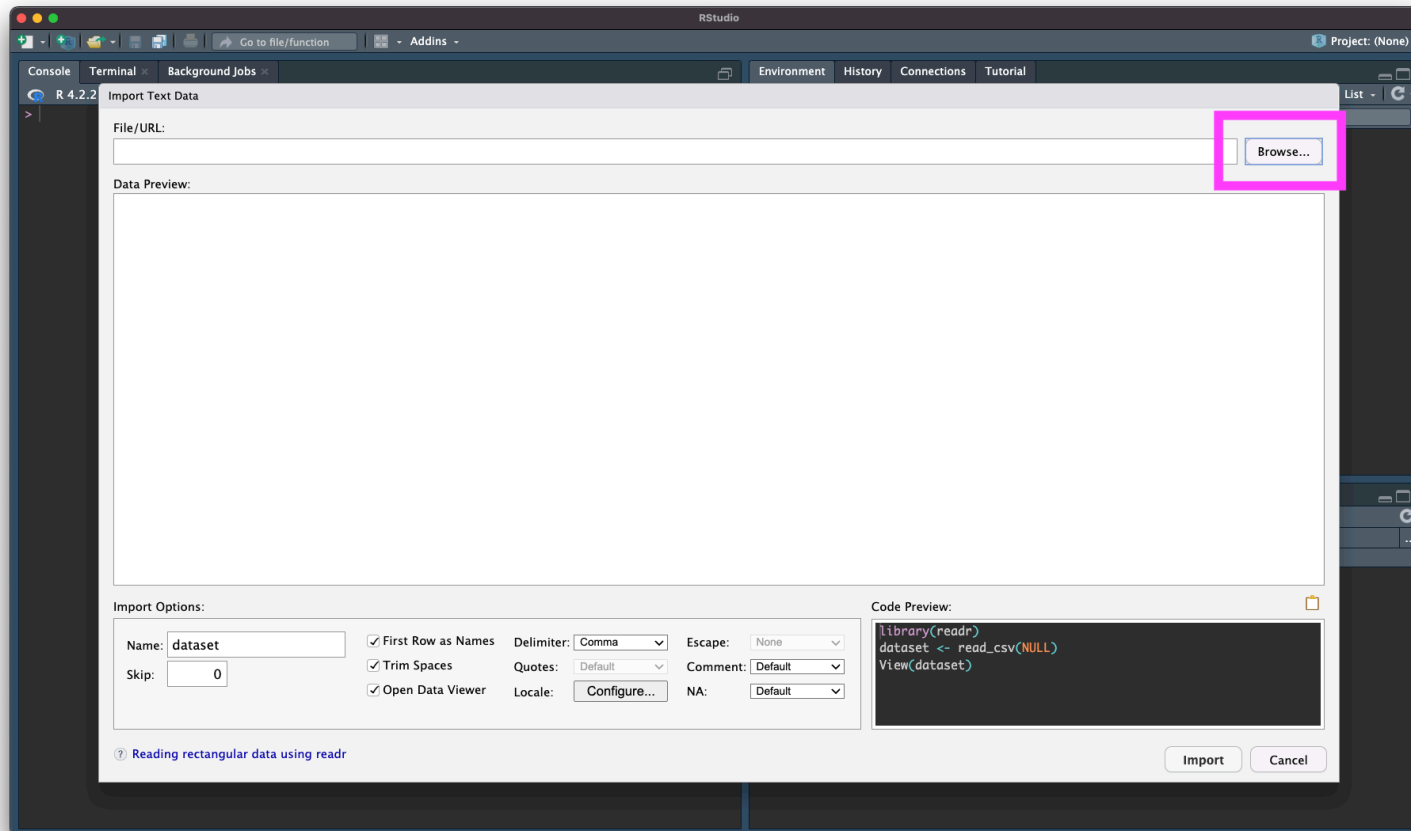
Import Dataset

- > File
- > Import Dataset
- > From Text (readr)
- > paste the url (https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv)
- > click "Update" and "Import"

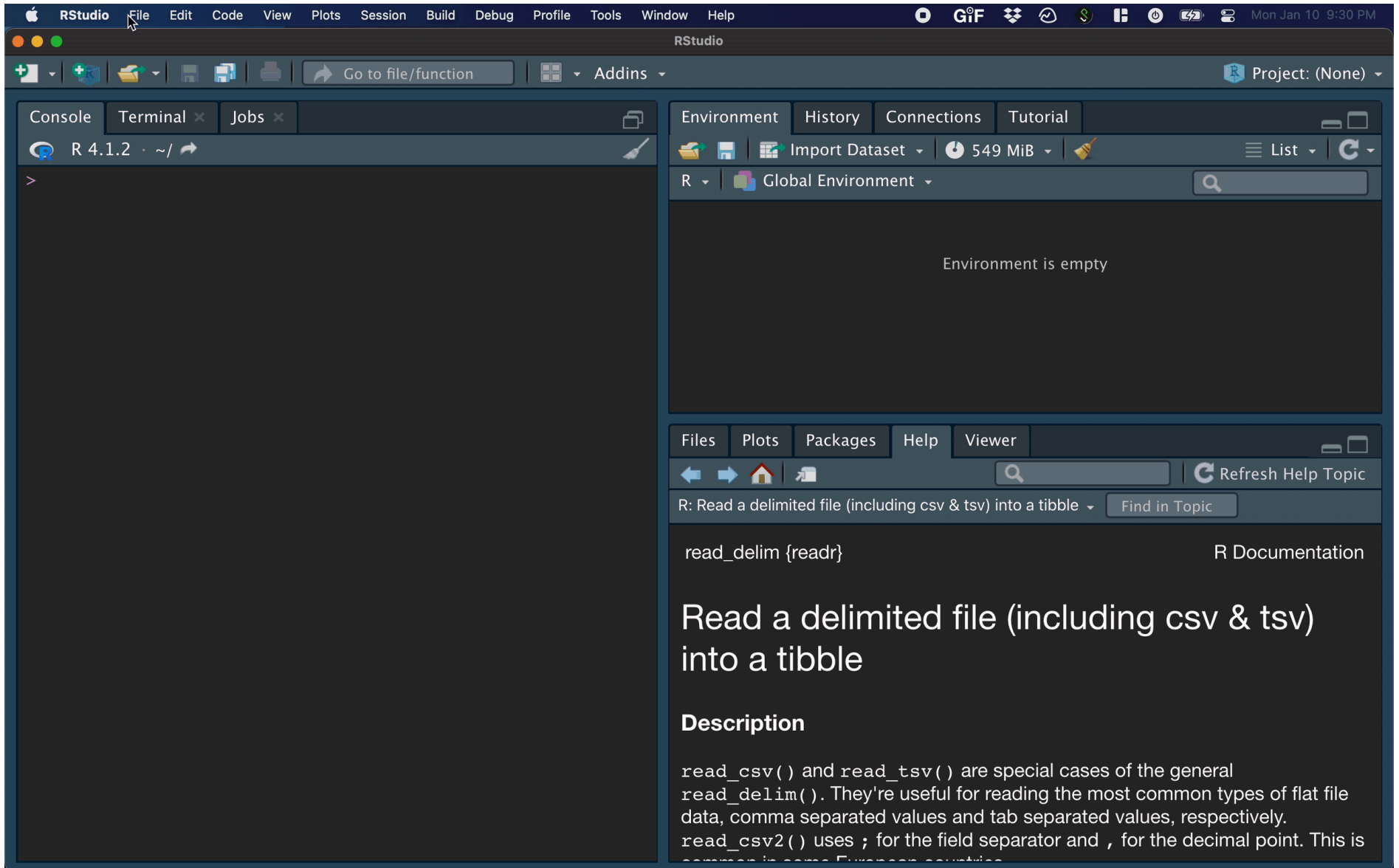
What Just Happened?

- You see a preview of the data on the top left pane.
- You see a new object called `ufo_data_complete` in your environment pane (top right). The table button opens the data for you to view.
- R ran some code in the console (bottom left).

Browsing for Data on Your Machine (not URL)



Watch the process (recap)



Example 2: Delimiters

- > File
- > Import Dataset
- > From Text (readr)
- > paste the url (<https://sisbid.github.io/Data-Wrangling/data/dropouts.txt>)
- > select delimiter
- > click “Update” and “Import”

Example 3: Excel

```
library(readxl)
```

- > File
- > Import Dataset
- > From Excel
- > paste the url (<https://sisbid.github.io/Data-Wrangling/data/asthma.xlsx>)
- > click “Update” and “Import”

Manual Import: Pros and Cons

Pros: easy!!

Cons: obscures some of what's happening, others will have difficulty running your code

Summary & Lab

Review the process: <https://youtu.be/LEkNfJgpunQ>

- > File
- > Import Dataset
- > From Text (`readr`) / From Excel
- > paste the url / browse
- > click “Update” and “Import”

https://sisbid.github.io/Data-Wrangling/02_Data_IO/lab/data-io-lab-part1.Rmd