Data Input

Quick reminder - Getting files from downloads

This course will involve moving files around on your computer and downloading files.

If you are new to this - check out these videos.

If you have a PC: https://youtu.be/we6vwB7DsNU

If you have a Mac: https://www.youtube.com/watch?v=Ao9e0cDzMrE

You can find these on the resource page of the website.

R Projects

R Projects

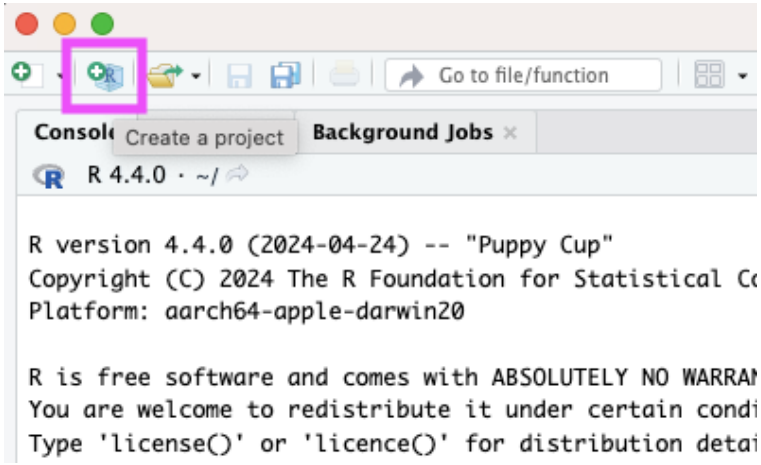
R Projects are a super helpful feature of RStudio. They help you:

- **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.

Let's go over how to create and use an R Project!

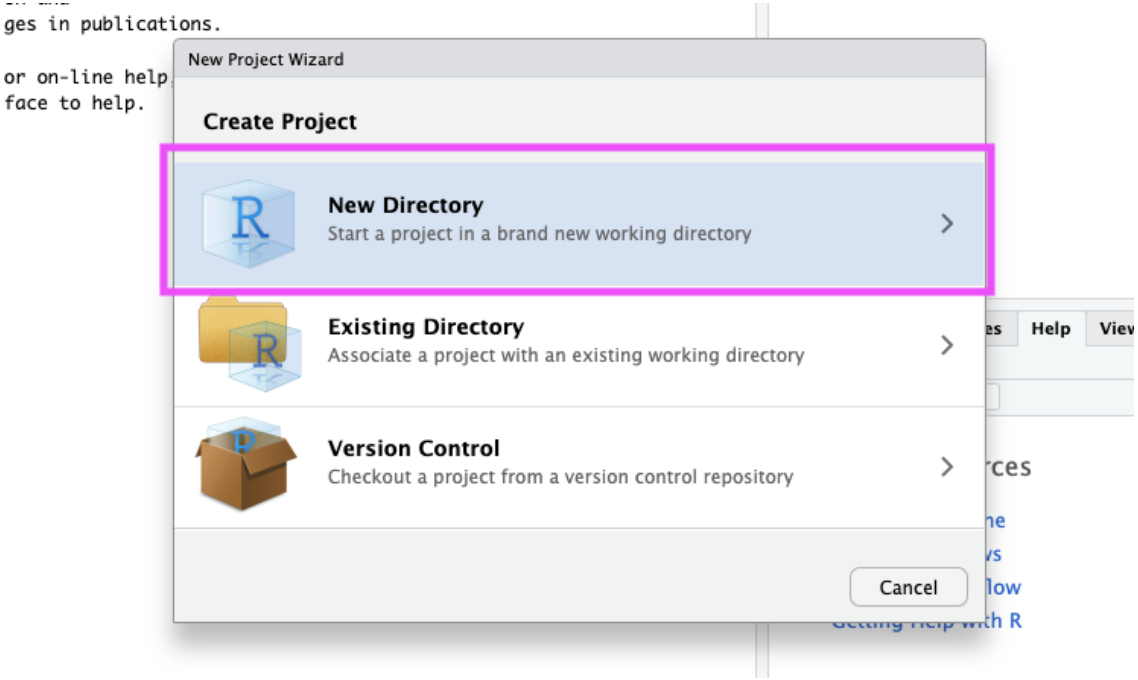
New R Project

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:



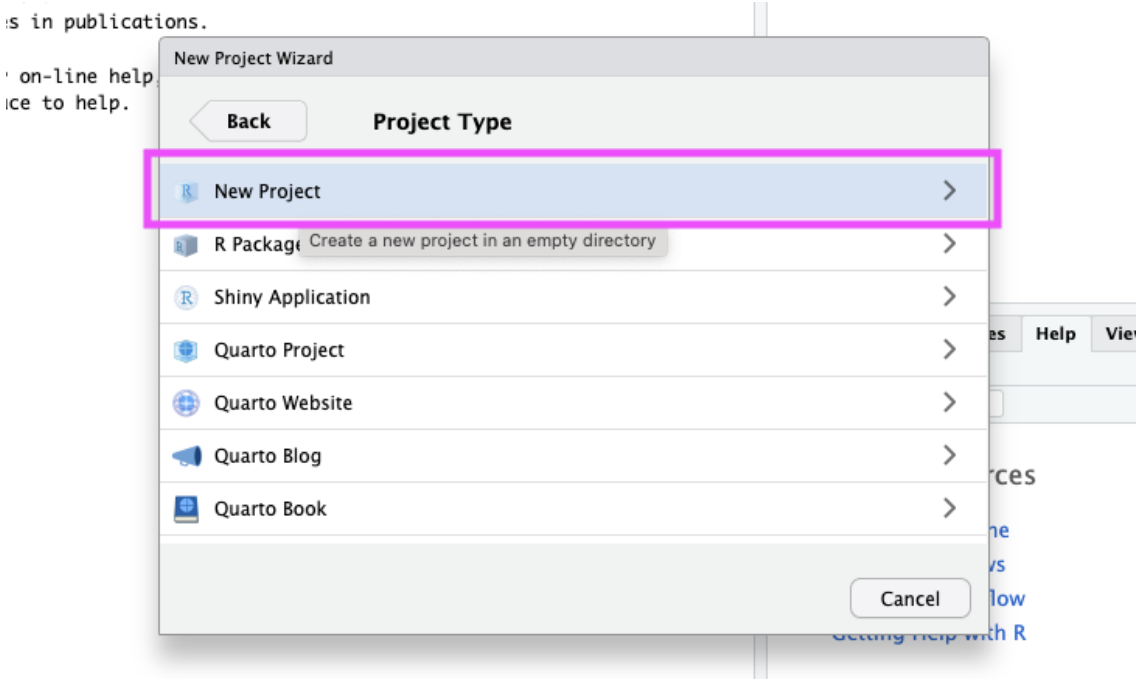
New R Project

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



New R Project

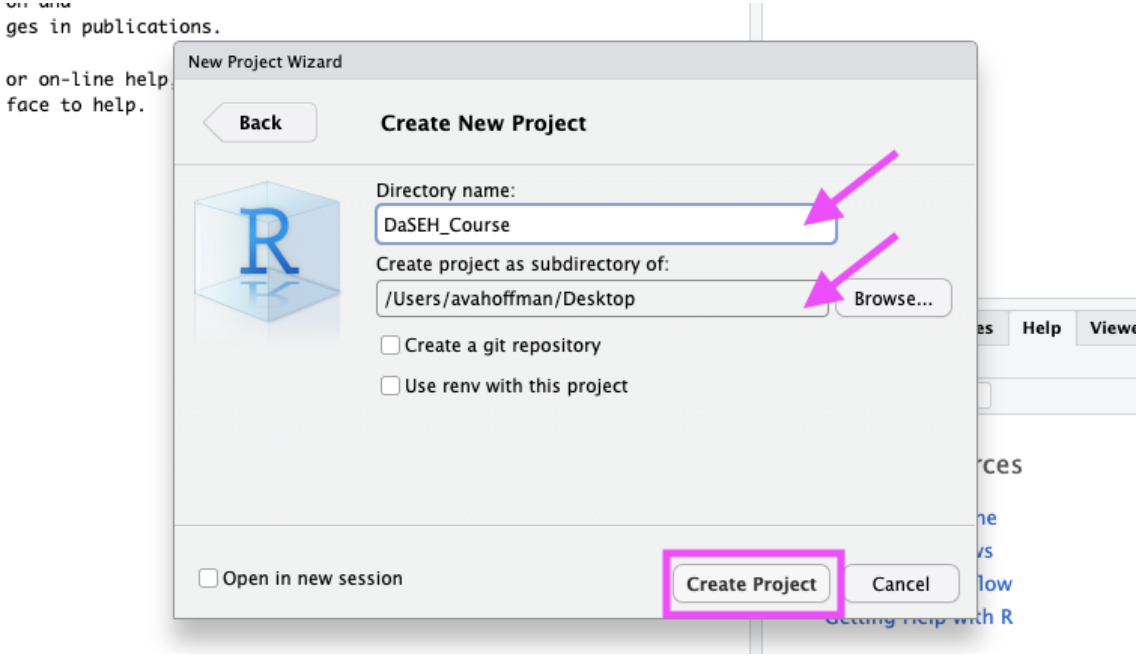
Click “New Project”:



New R Project

Type in a name for your new folder.

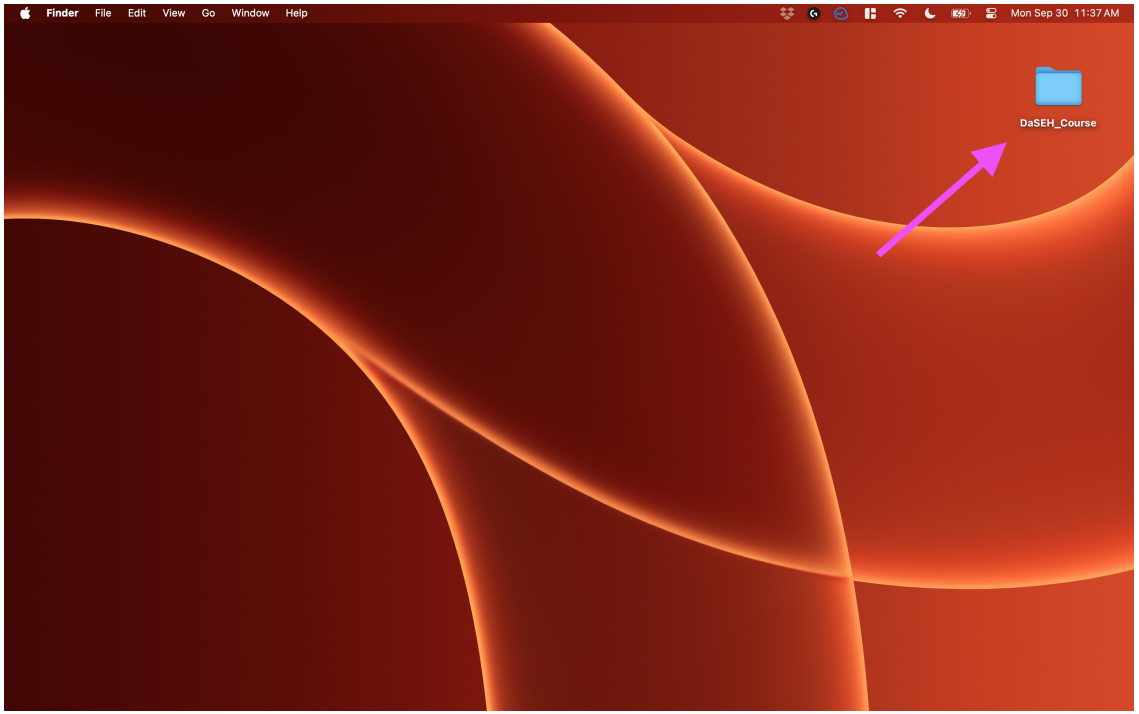
Store it somewhere easy to find, such as your Desktop:



New R Project

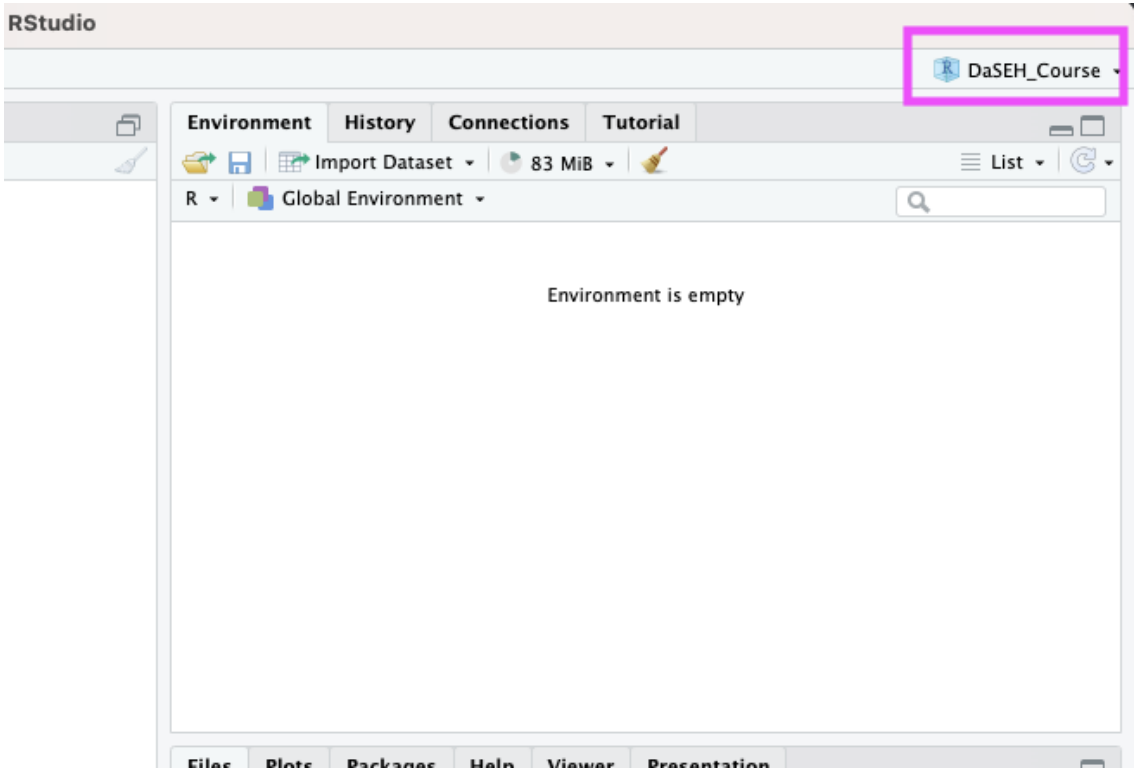
You now have a new R Project folder on your Desktop!

Make sure you add any scripts or data files to this folder as you go through your Intro to R lessons, or work on a new project. This will make sure R is able to “find” your files.



See and change projects

You can see what project you have open in the top right corner.



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’)

Data Input

CO Heat-related ER visits dataset:

We're going to load a dataset from the Colorado Environmental Public Health Tracking Program. This dataset has age-adjusted visit rates and total number of visits for all genders by Colorado county for 2011-2023.

- Check out the data at: https://coepht.colorado.gov/heat-related-illness

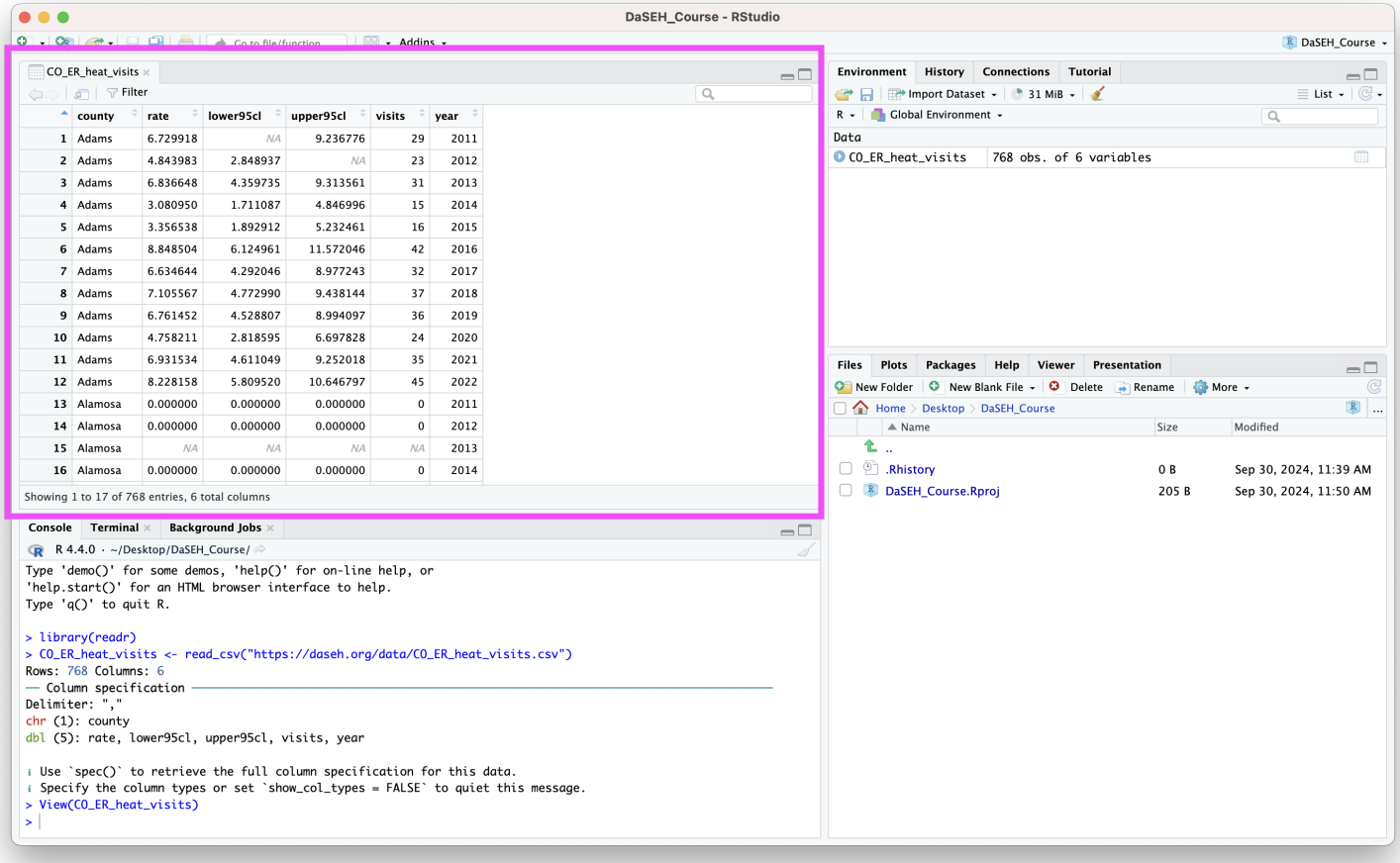
Import Dataset (URL)

- > File
- > Import Dataset
- > From Text (`readr`)
- > paste the url (https://daseh.org/data/CO_ER_heat_visits.csv)
- > click “Update” and “Import”

Saves data in memory, not to hard drive

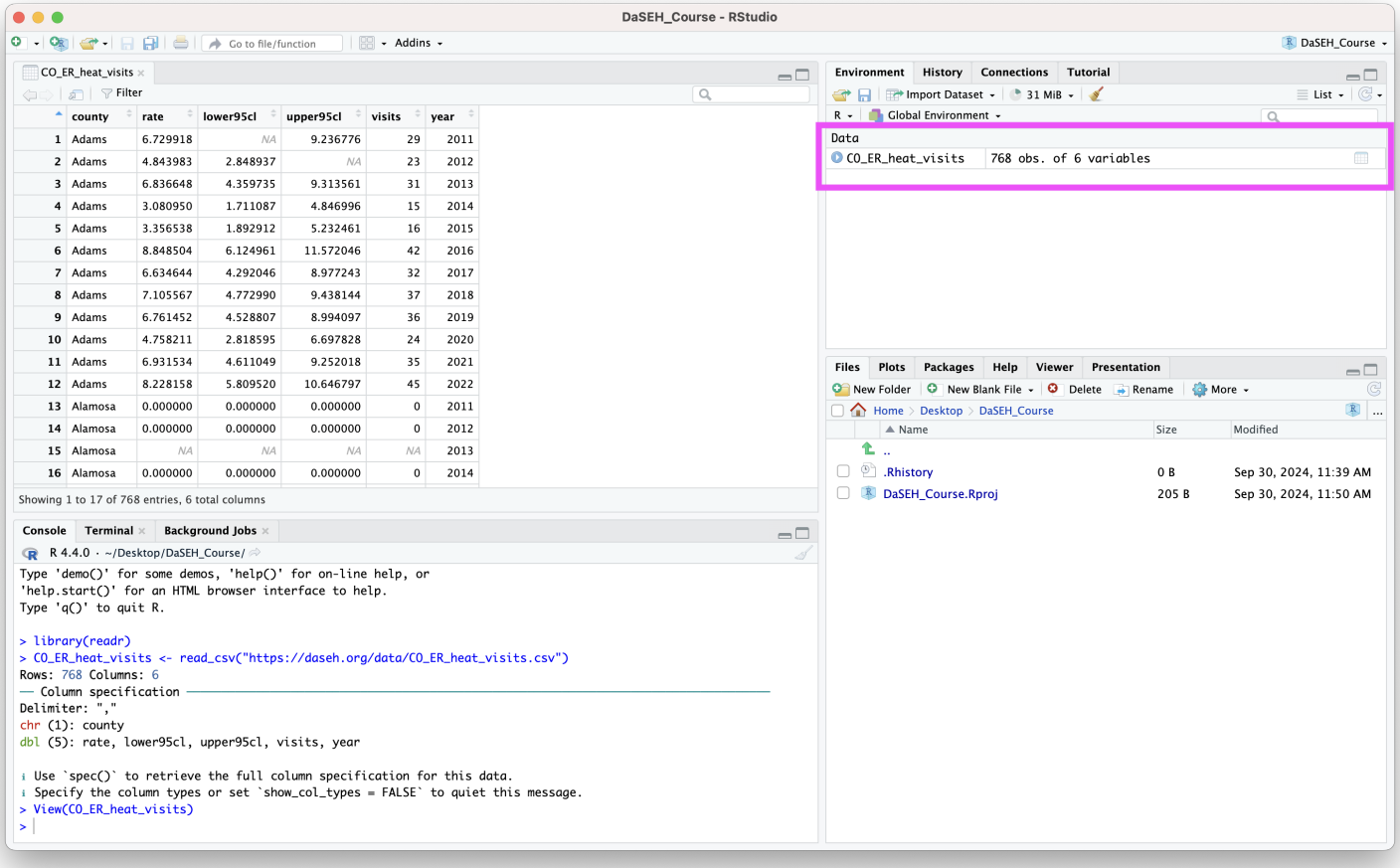
What Just Happened?

You see a preview of the data on the top left pane.



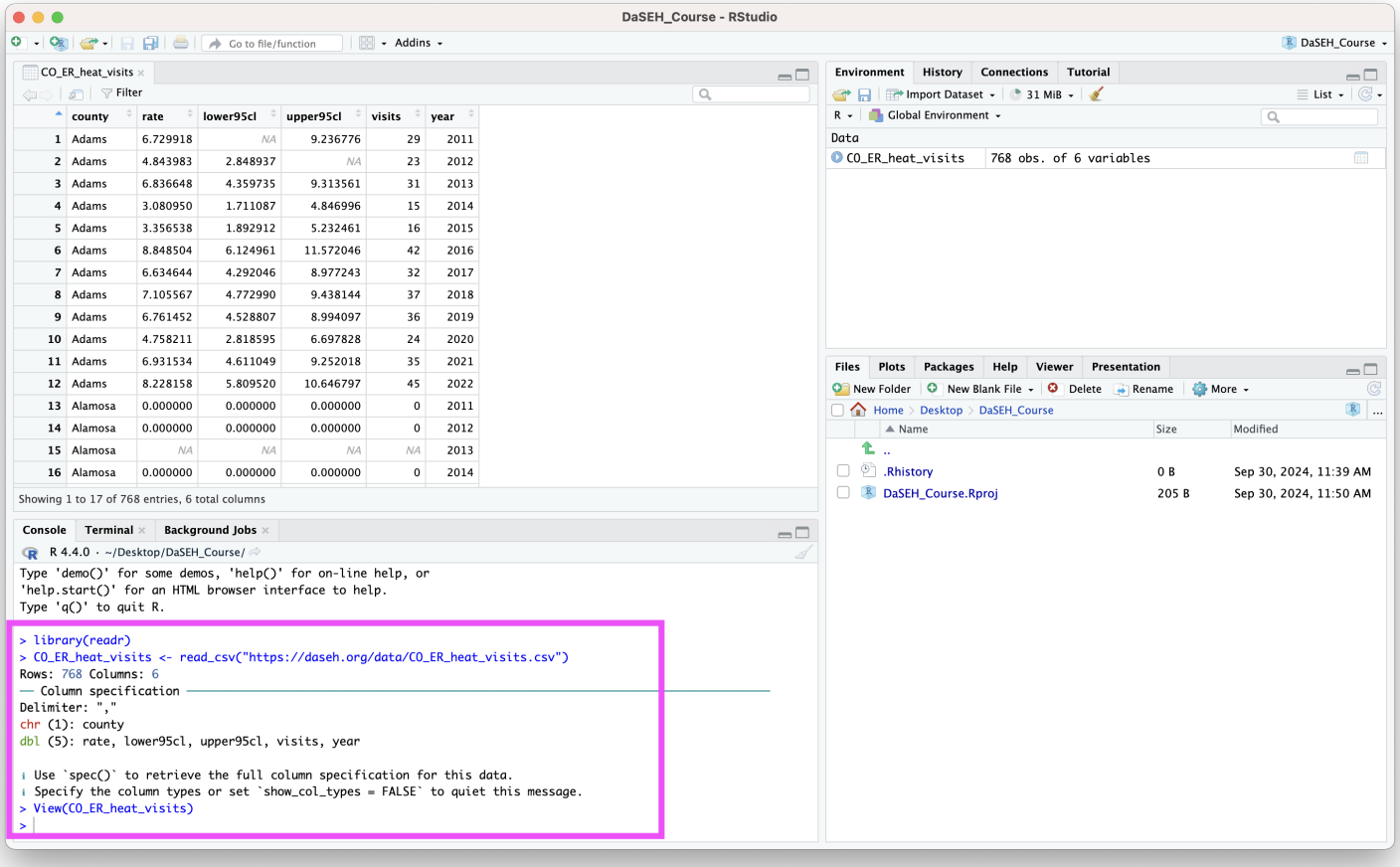
What Just Happened?

You see a new object called `CO_ER_heat_visits` in your environment pane (top right). The table button opens the data for you to view.

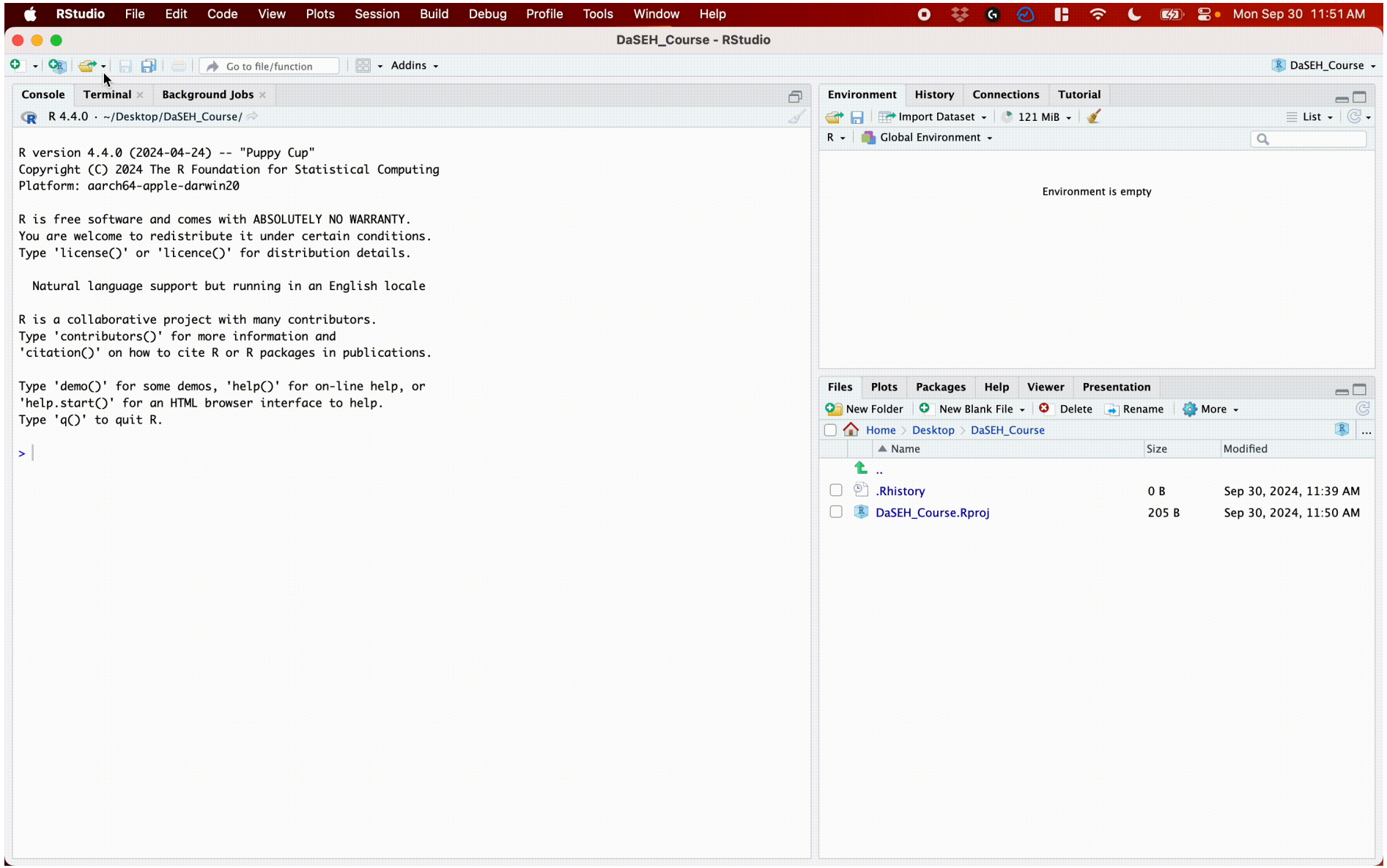


What Just Happened?

R ran some code in the console (bottom left).



Import Dataset



Import Dataset (file)

- > *Download the data*
- > *Put data in the project folder*
- > File, Import Dataset, From Text (`readr`)
- > browse for the file
- > click “Update” and “Import”

GUT CHECK!

How can we get data into R?

A. From a URL

B. From a file we downloaded

C. Both of these!

Lab - Part 1

- Class Website
- Data Input Lab

Manual Import: Pros and Cons

Pros: easy!!

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

Getting data into R (directly)

Data Input: Read in Directly

The `tidyverse` contains a package `readr` that is handy for importing data.

```
library(tidyverse)
dat <- read_csv(
  file = "https://daseh.org/data/CO_ER_heat_visits.csv"
)

# `head` displays first few rows of a data frame. `tail()` works the same way.
head(dat, n = 5)

# A tibble: 5 × 6
  county  rate lower95cl upper95cl visits  year
  <chr>  <dbl>     <dbl>     <dbl>  <dbl> <dbl>
1 Adams   6.73     NA         9.24     29  2011
2 Adams   4.84      2.85     NA        23  2012
3 Adams   6.84      4.36      9.31     31  2013
4 Adams   3.08      1.71      4.85     15  2014
5 Adams   3.36      1.89      5.23     16  2015
```

Data Input: Declaring Arguments

```
dat <- read_csv(
  file = "https://daseh.org/data/CO_ER_heat_visits.csv"
)
# EQUIVALENT TO
dat <- read_csv(
  "https://daseh.org/data/CO_ER_heat_visits.csv"
)
```

Data Input: Read in Directly

`read_csv()` needs an argument `file =` **in quotation marks**.

- can be path to a file on a website (URL)
- can be **path** in your local computer – absolute file path or relative file path

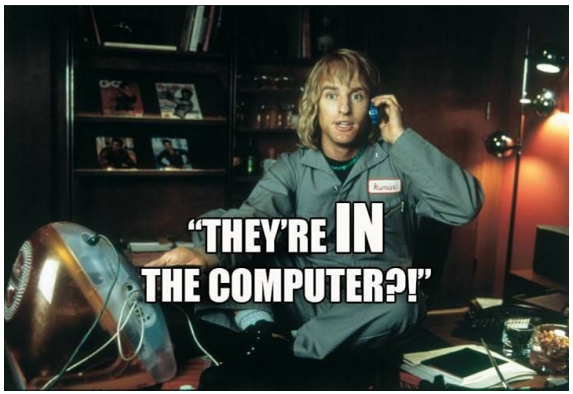
```
# URL
dat <- read_csv("https://daseh.org/data/CO_ER_heat_visits.csv")

# In project folder
dat <- read_csv("CO_ER_heat_visits.csv")
```

The working directory

When we work in an R Project, our project folder is our **working directory**.

Working directory is a folder (directory) that RStudio will use to find files.



Checking the working directory

Run the `getwd()` function to determine your working directory.

```
# Get the working directory
getwd()
```

Setting the working directory

You can set the working directory manually with the `setwd()` function. But it's easier to set up a project :)

```
# set the working directory
setwd("/Users/avahoffman/Desktop")
```

Now what? Checking data & Other formats

Data Input: Checking the data

- the `View()` function shows your data in a new tab, in spreadsheet format
- be careful if your data is big!

```
View(dat)
```

Data Input: Other delimiters

`read_tsv()` can read tab delimited (separated) files.

`read_delim()` can be used to specify the delimiter.

- `file` is the path to your file, in quotes
- `delim` is what separates the fields within a record

```
## Examples
dat2 <- read_tsv(file = "table1.tsv", delim = "\t")

dat3 <- read_delim(file = "data.txt", delim = ":")
```

Data input: other file types

- `readxl` package can read excel files
- `haven` package has functions to read SAS, SPSS, Stata formats
- There are also resources for REDCap : `REDCapR`

WARNING! `read.csv` is * base R *

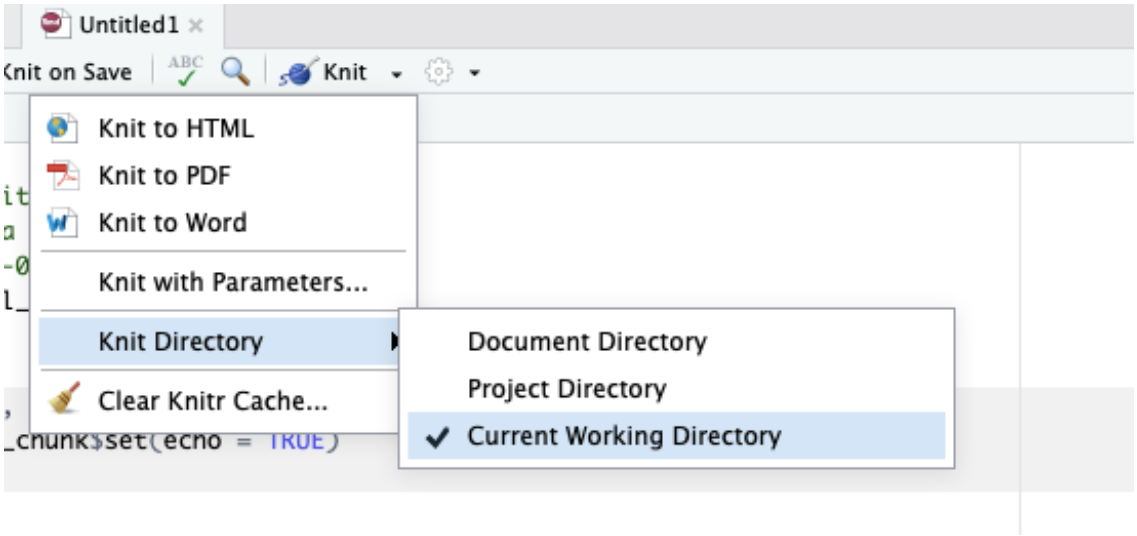
There are also data importing functions provided in base R (rather than the `readr` package), like `read.delim()` and `read.csv()`.

These functions have slightly different syntax for reading in data (e.g. `header` argument).

However, while many online resources use the base R tools, the latest version of RStudio switched to use these new `readr` data import tools, so we will use them in the class for slides. They are also up to two times faster for reading in large datasets, and have a progress bar which is nice.

TROUBLESHOOTING: Setting the working directory

If you are trying to knit your work, it might help to set the knit directory to the "Current Working Directory":



Other Useful Functions

- The `str()` function can tell you about data/objects.
- We will also discuss the `glimpse()` function later, which does something very similar.
- `head()` shows first few rows
- `tail()` shows the last few rows

Summary

R Projects can make it easier to find files.

Importing data manually:

- File > Import Dataset > From Text (`readr`)
- Paste the url / browse
- Click “Update” and “Import”
- Review the process: `https://youtu.be/LEkNfJgpunQ`

Importing data programmatically:

- `read_csv()` function from tidyverse (`readr`) package
- Use `getwd()` to check your working directory, where R looks for your data files

Summary - Part 2

Look at your data!

- Check the environment for a data object
- `View()` gives you a preview of the data in a new tab

Other file types

- `readr` package: `read_delim()` for general delimited files
- other packages for more complicated files.

Don't forget to use `<-` to assign your data to an object!

Lab - Part 2

- Class Website
- Data Input Lab
- Posit's Data Import Cheatsheet
- Day 2 Cheatsheet

Image by Gerd Altmann from Pixabay