

# Data I/O, Part 1

Data Wrangling in R

# Explaining output on slides

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

```
print("I'm code")
```

```
[1] "I'm code"
```

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

So `print("I'm code")` is the code chunk and `[1] "I'm code"` is the output.

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

# 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
  - tab delimited (e.g. '.txt')
  - comma separated (e.g. '.csv')
  - Microsoft excel (e.g. '.xlsx')

# Data Input

UFO Sightings via Kaggle.com: “Reports of unidentified flying object reports in the last century”.

“There are two versions of this dataset: scrubbed and complete. The complete data includes entries where the location of the sighting was not found or blank (0.8146%) or have an erroneous or blank time (8.0237%). Since the reports date back to the 20th century, some older data might be obscured. Data contains city, state, time, description, and duration of each sighting.”

<https://www.kaggle.com/NUFORC/ufo-sightings>

# 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
```

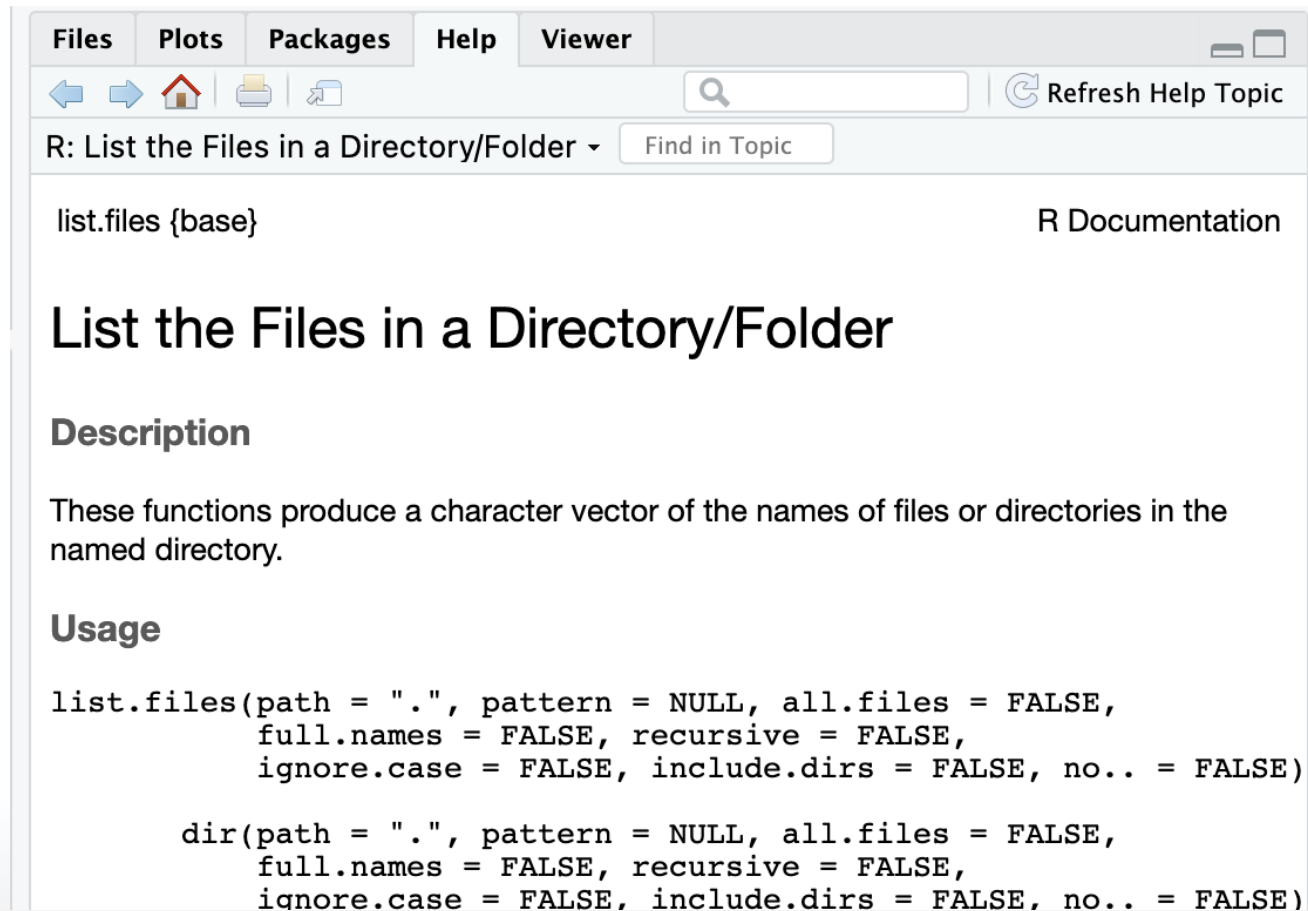
```
x + 2
```

```
[1] 4
```

# Help

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

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



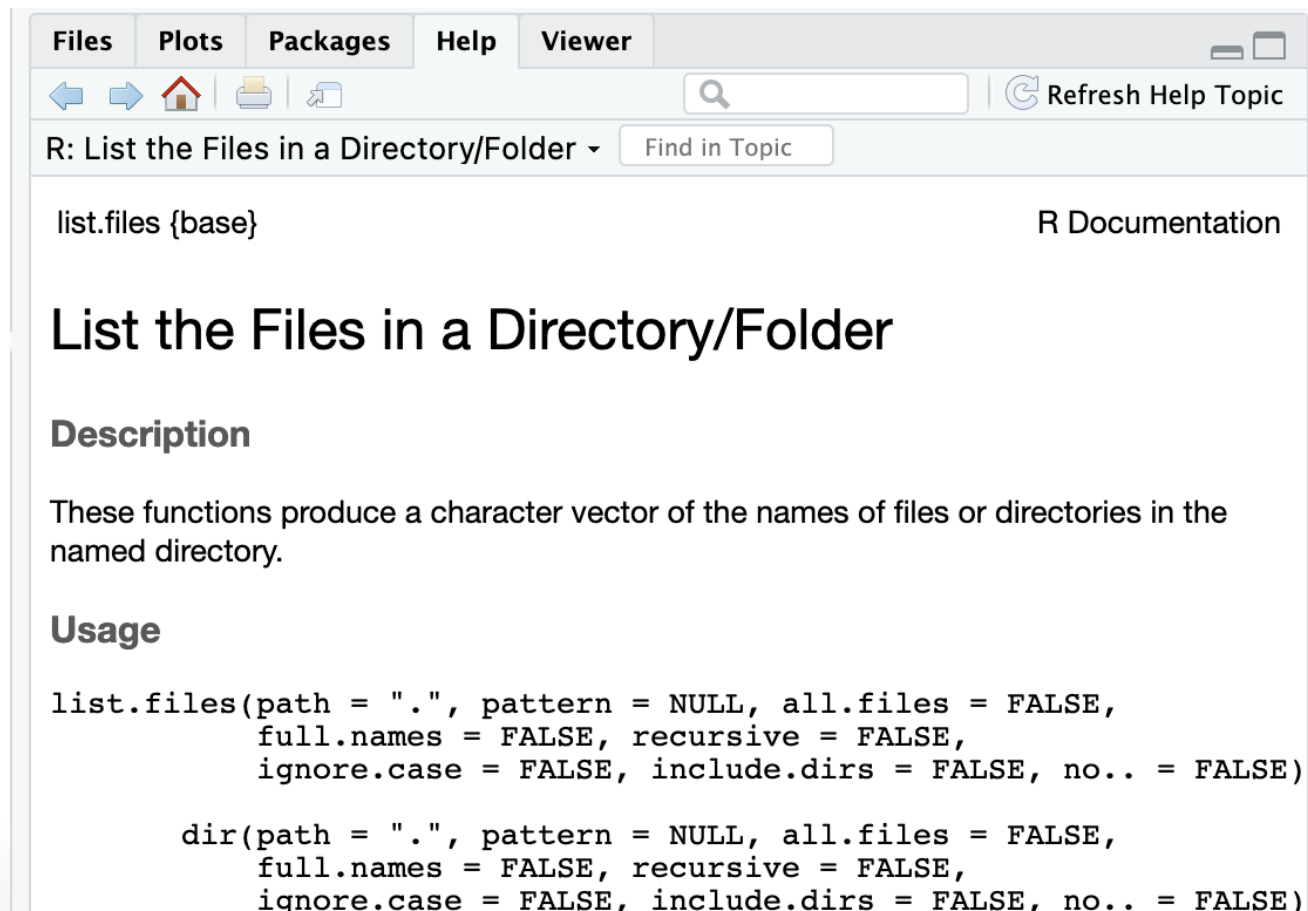
The screenshot shows the R Help Viewer window. The title bar includes tabs for Files, Plots, Packages, Help, and Viewer. Below the tabs is a navigation bar with back, forward, and home icons, a search box, and a 'Refresh Help Topic' button. The main content area displays the title 'R: List the Files in a Directory/Folder' with a 'Find in Topic' button. The text 'list.files {base}' and 'R Documentation' are visible. The main heading is 'List the Files in a Directory/Folder'. Under the 'Description' section, it states: 'These functions produce a character vector of the names of files or directories in the named directory.' Under the 'Usage' section, the following code is shown:

```
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 on AnVIL's RStudio.

```
install.packages("naniar")  
AnVIL::install("naniar") # Alternative  
library(naniar)
```



The screenshot shows the RStudio interface with the 'Packages' tab selected. The main pane displays the help topic 'R: List the Files in a Directory/Folder'. The title bar includes 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. The toolbar has navigation icons, a search bar, and a 'Refresh Help Topic' button. The help content includes the function 'list.files {base}', a description, and usage examples.

**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)
```

# Data Input

The Easy way: import text datasets using the “File -> Import Dataset -> From Text (readr)” command. Selecting this will bring up a new screen that lets you specify the formatting of your text file.

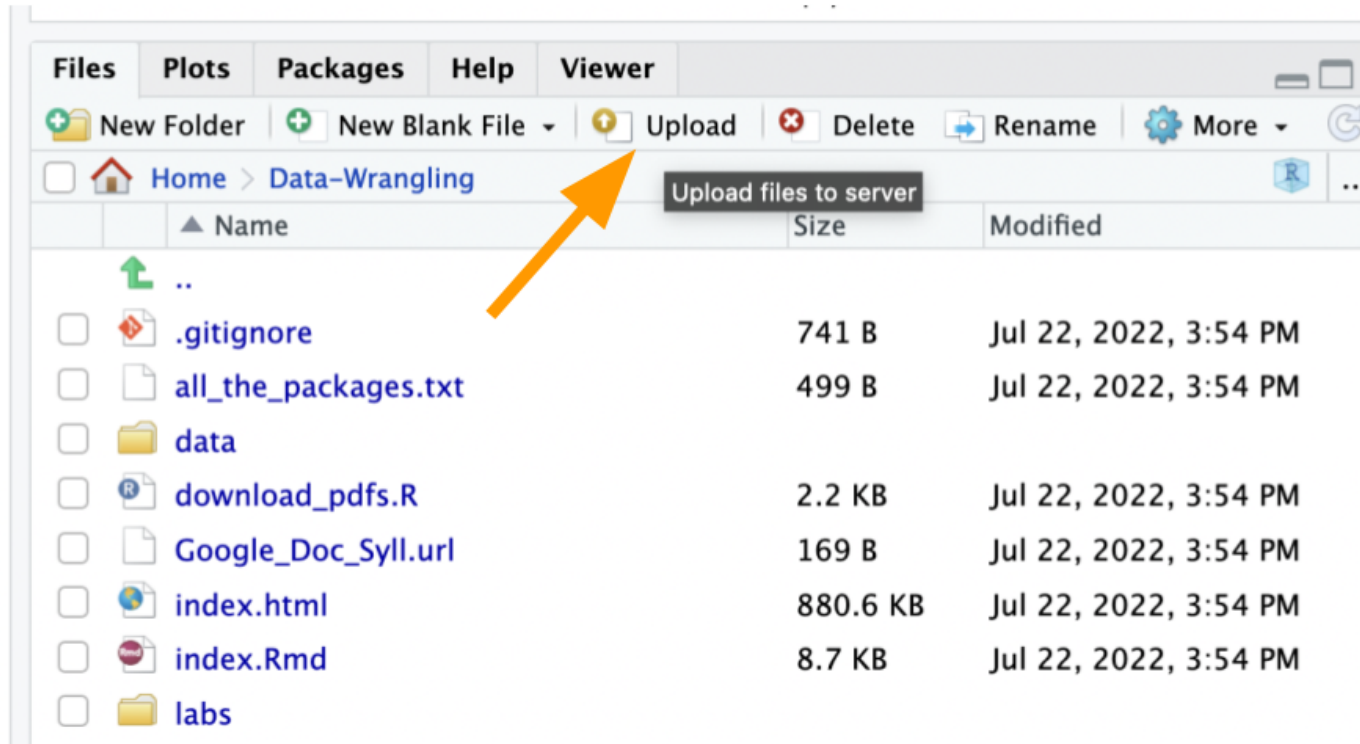
Going through this process enters the corresponding R commands in the console (you can copy these for later!)



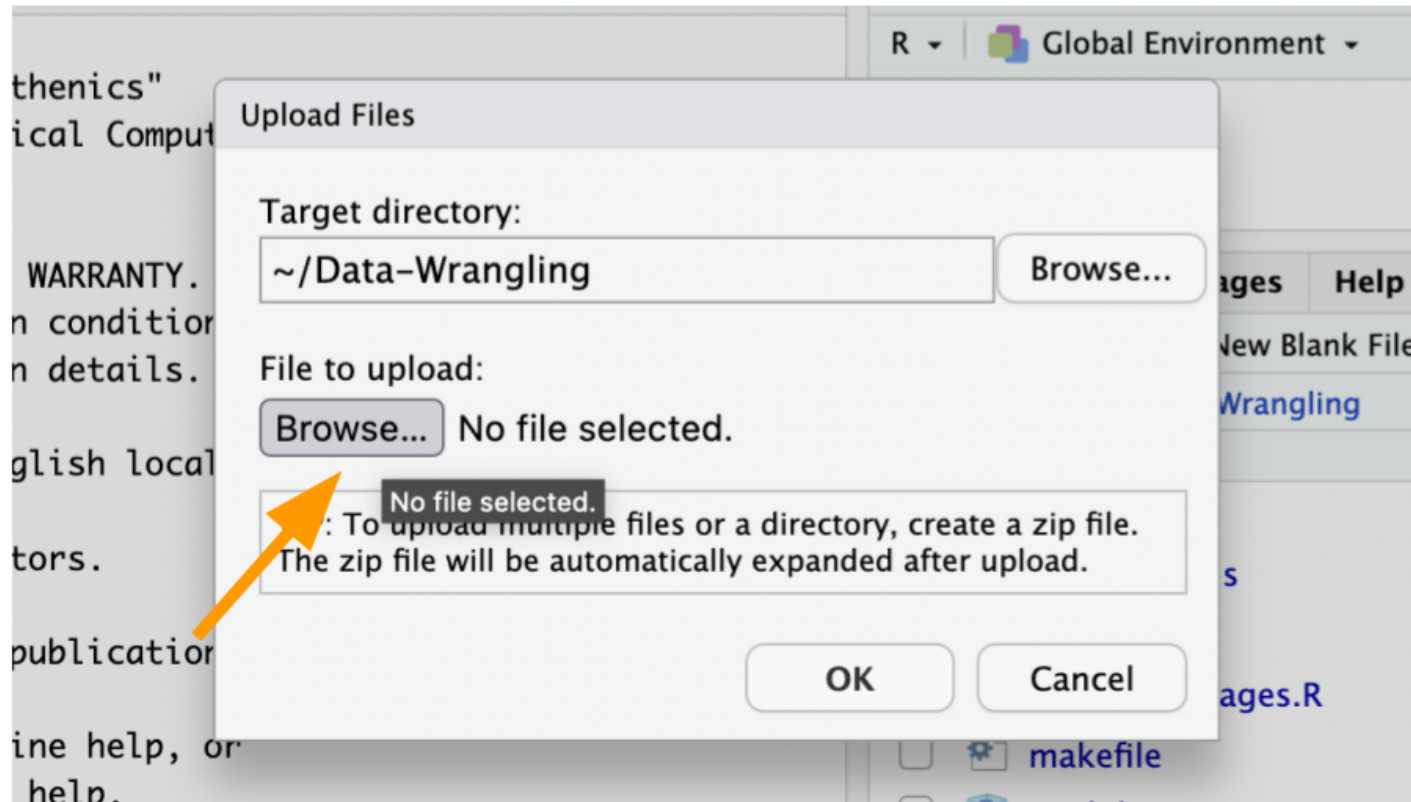
## Approach #1

- Download data onto your machine from [http://sisbid.github.io/Data-Wrangling/data/ufo/ufo\\_data\\_complete.csv.gz](http://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv.gz) (right-click this link)
- Upload the data to AnVIL
- Read into RStudio's environment

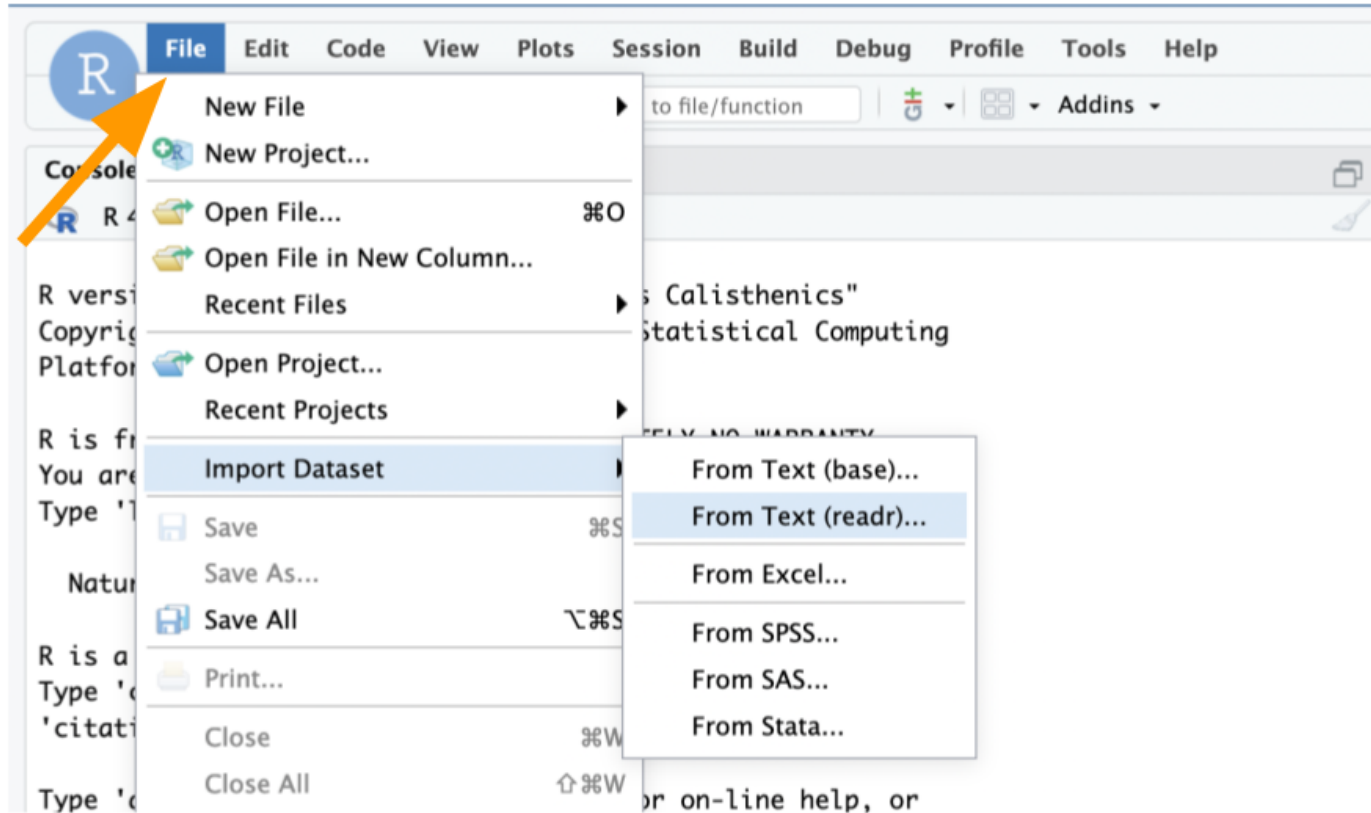
# Upload the data to AnVIL



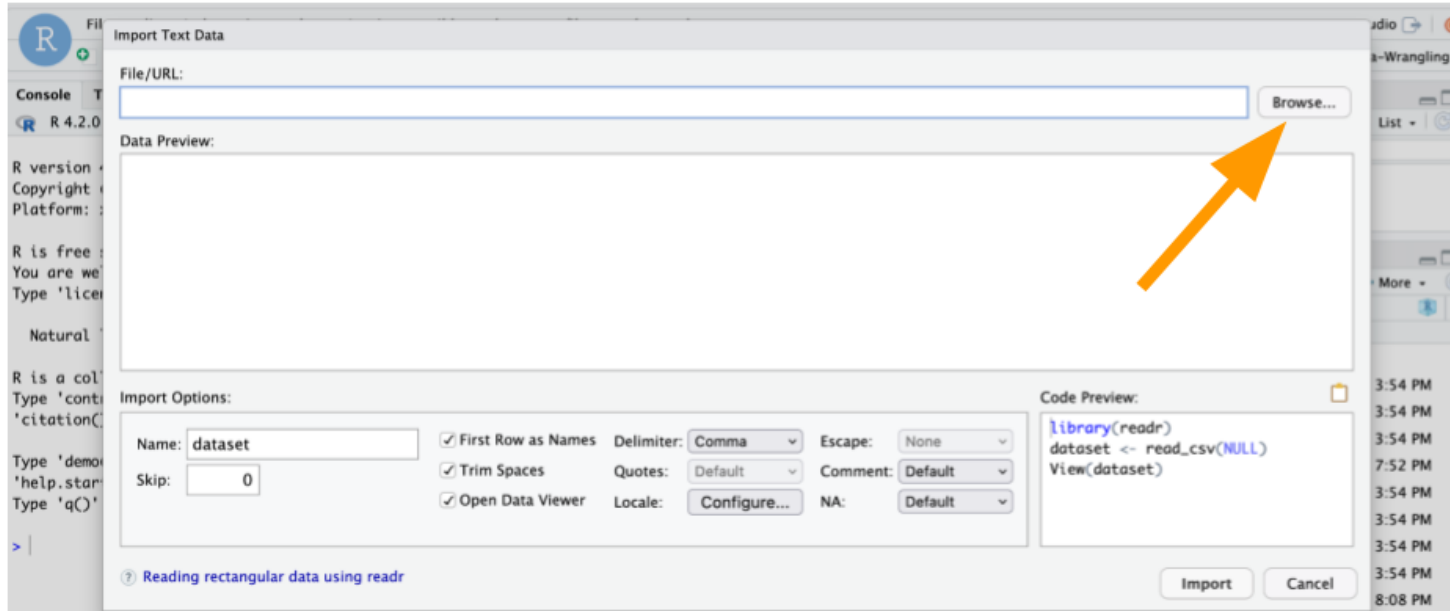
# Upload the data to AnVIL



# Read into RStudio's environment



# Read into RStudio's environment



# Read into RStudio's environment

Import Text Data

File/URL:  
~/Data-Wrangling/ufo\_data\_complete.csv.gz Browse...

Data Preview:

datetime (character)	city (character)	state (character)	country (character)	shape (character)	duration (seconds) (double)	duration (hours/min) (character)	comments
10/10/1949 20:30	san marcos	tx	us	cylinder	2700	45 minutes	This event took place in early fall
10/10/1949 21:00	lackland afb	tx	NA	light	7200	1-2 hrs	1949 Lackland AFB&#44 TX. Lig
10/10/1955 17:00	chester (uk/england)	NA	gb	circle	20	20 seconds	Green/Orange circular disc over
10/10/1956 21:00	edna	tx	us	circle	20	1/2 hour	My older brother and twin sister

Previewing first 50 entries. 592 parsing errors.

Import Options:

Name:  ☒ First Row as Names Delimiter:  Escape:

Skip:  ☒ Trim Spaces Quotes:  Comment:

☒ Open Data Viewer Locale:  NA:

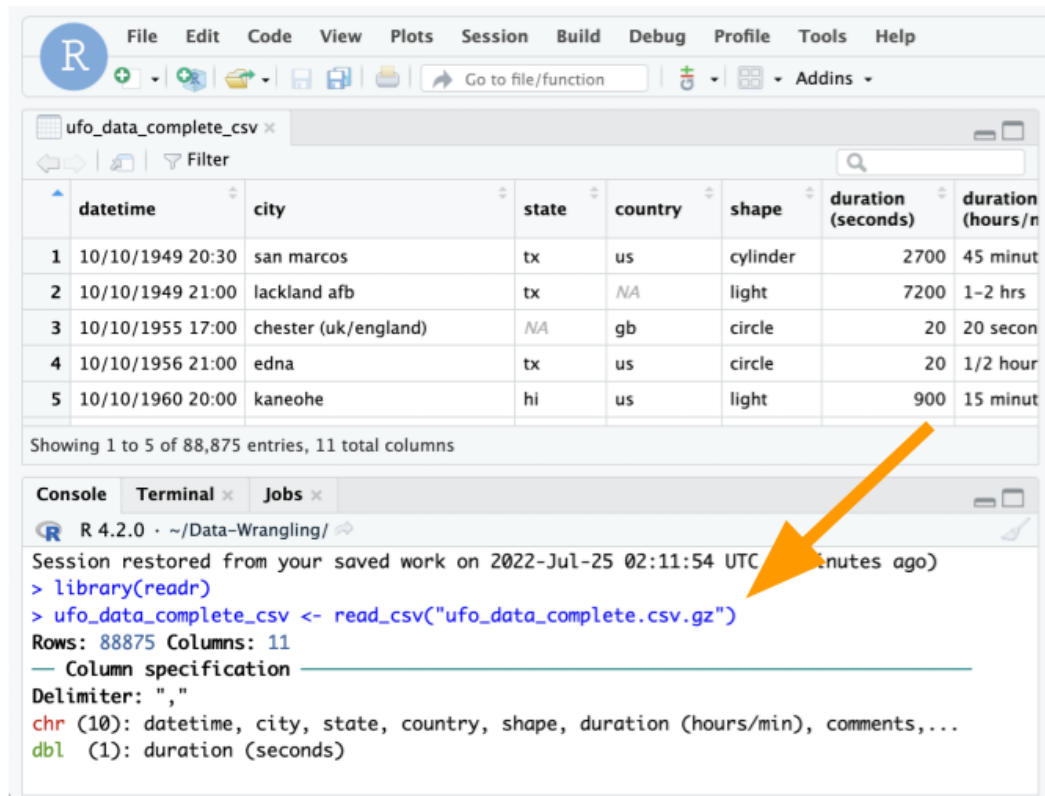
Code Preview:

```
library(readr)
ufo_data_complete_csv <- read_csv(
  "ufo_data_complete.csv.gz")
View(ufo_data_complete_csv)
```

Import Cancel

? Reading rectangular data using readr

# Check out the commands!



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with various icons. The main window displays a data frame named `ufo_data_complete_csv` with 88,875 entries and 11 columns. The first five rows are visible:

	datetime	city	state	country	shape	duration (seconds)	duration (hours/n
1	10/10/1949 20:30	san marcos	tx	us	cylinder	2700	45 minut
2	10/10/1949 21:00	lackland afb	tx	NA	light	7200	1-2 hrs
3	10/10/1955 17:00	chester (uk/england)	NA	gb	circle	20	20 secon
4	10/10/1956 21:00	edna	tx	us	circle	20	1/2 hour
5	10/10/1960 20:00	kaneohe	hi	us	light	900	15 minut

Below the data frame, it says "Showing 1 to 5 of 88,875 entries, 11 total columns". The bottom panel shows the Console with the following output:

```
R 4.2.0 · ~/Data-Wrangling/
Session restored from your saved work on 2022-Jul-25 02:11:54 UTC (minutes ago)
> library(readr)
> ufo_data_complete_csv <- read_csv("ufo_data_complete.csv.gz")
Rows: 88875 Columns: 11
— Column specification —
Delimiter: ","
chr (10): datetime, city, state, country, shape, duration (hours/min), comments,...
dbl (1): duration (seconds)
```

An orange arrow points from the console output to the data frame.

## Approach #2

- Read into RStudio's environment directly

```
ufo_data_complete_csv <-  
  read_csv("http://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv.gz")
```

Warning: One or more parsing issues, see `problems()` for details

Rows: 88875 Columns: 11

— Column specification —

Delimiter: ",",

chr (10): datetime, city, state, country, shape, duration (hours/min), comments, date posted, latitude, longitude

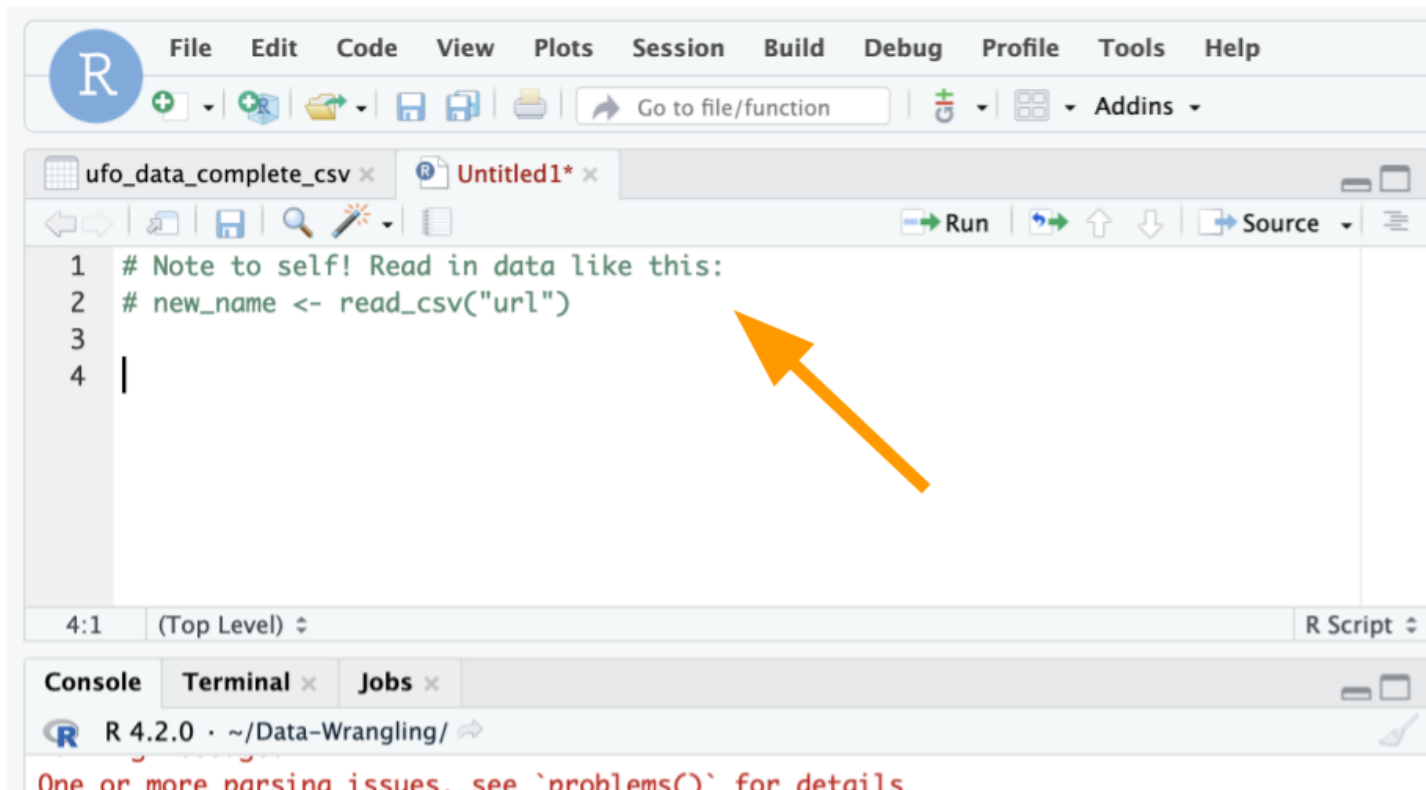
dbl (1): duration (seconds)

- i Use `spec()` to retrieve the full column specification for this data.
- i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.



# 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 bar is a toolbar with icons for adding files, saving, and other functions. 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 to the comment on line 2. The status bar at the bottom shows '4:1 (Top Level)' and 'R Script'. Below the editor window is a console window with the text 'R 4.2.0 · ~/Data-Wrangling/' and a red message: '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: readr

`read_delim()` and `read_csv()` from the `readr` package

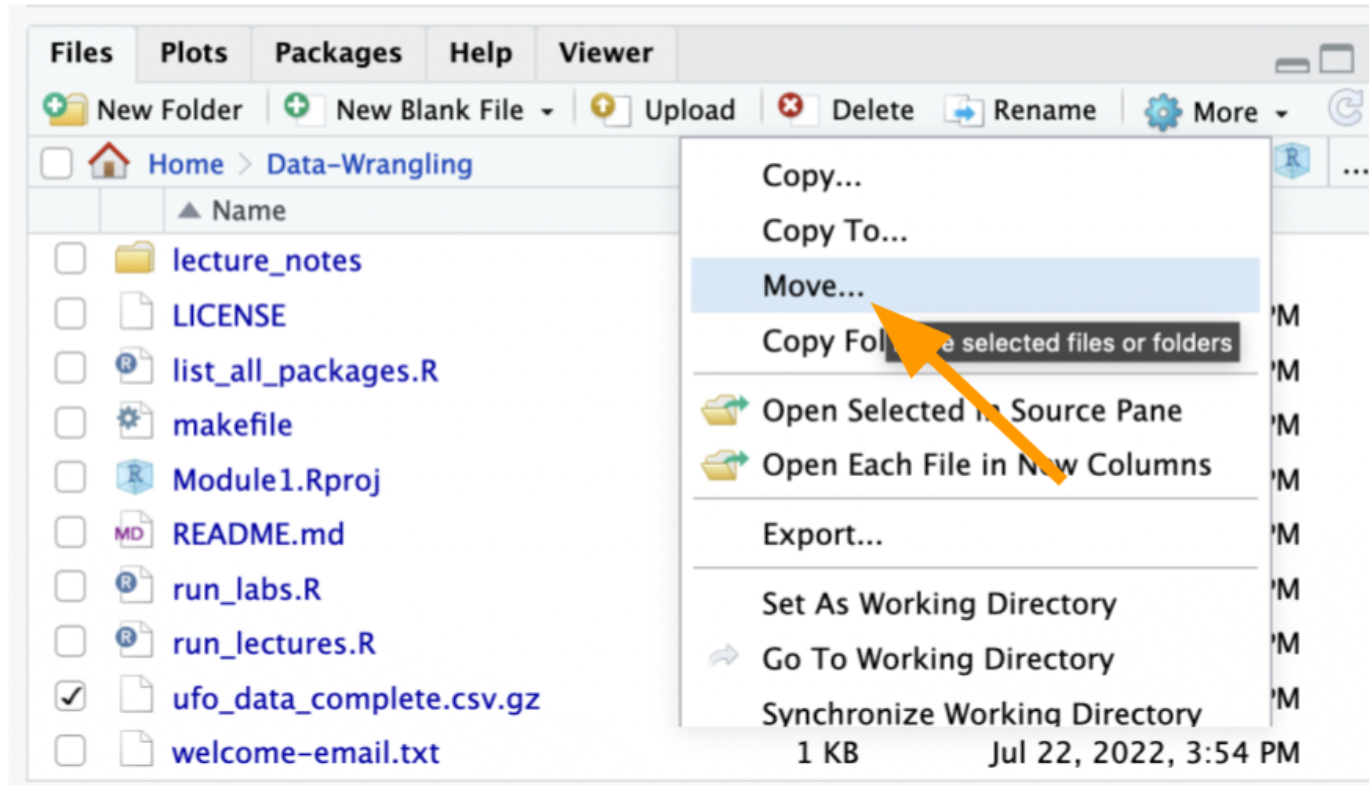
```
# example:  
read_delim(file = "file.txt", delim = "\t")  
read_csv("file.csv")
```

# Data Input

- The filename is the path to your file, in quotes
- The function will look in your “working directory” if no absolute file path is given
- Note that the filename can also be a path to a file on a website (e.g. ‘`www.someurl.com/table1.txt`’)

# Data Input

What if your file is in the “Home” directory?



# Data Input

Backtrack using the relative path with `../` like:

```
ufo <- read_csv("../ufo_data_complete.csv.gz")
```

# Data Input

Or, read in from a subfolder:

```
ufo <- read_csv("data/ufo/ufo_data_complete.csv")
```

Warning: One or more parsing issues, see `problems()` for details

Rows: 88875 Columns: 11

— Column specification —

Delimiter: ",",

chr (10): datetime, city, state, country, shape, duration (hours/min), comment

dbl (1): duration (seconds)

**i** Use `spec()` to retrieve the full column specification for this data.

**i** Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Data Input

The `read_delim()` and related functions return a “tibble” is a `data.frame` with special printing, which is the primary data format for most data cleaning and analyses.

```
class(ufo)
```

```
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```



# Data Input

ufo

```
# A tibble: 88,875 × 11
  datetime      city state country shape `duration (sec...` `duration (hou...
  <chr>         <chr> <chr> <chr>   <chr>   <dbl> <chr>
1 10/10/1949 20:30 san ... tx      us      cyli...    2700 45 minutes
2 10/10/1949 21:00 lack... tx      <NA>     light    7200 1-2 hrs
3 10/10/1955 17:00 ches... <NA>    gb      circ...     20 20 seconds
4 10/10/1956 21:00 edna   tx      us      circ...     20 1/2 hour
5 10/10/1960 20:00 kane... hi      us      light    900 15 minutes
6 10/10/1961 19:00 bris... tn      us      sphe...    300 5 minutes
7 10/10/1965 21:00 pena... <NA>    gb      circ...    180 about 3 mins
8 10/10/1965 23:45 norw... ct      us      disk    1200 20 minutes
9 10/10/1966 20:00 pell... al      us      disk    180 3 minutes
10 10/10/1966 21:00 live... fl      us      disk    120 several minutes
# ... with 88,865 more rows
```

# Data Input

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, like `header` and `as.is`.

However, while many online resources use the base R tools, recent versions 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.

## Data input: **readr** highlights

- Modern, improved tools from `readr` R package: `read_delim()`, `read_csv()`
  - needs a file path to be provided
  - parses the file into rows/columns, determines column type
  - returns a data frame
- Some functions to look at a data frame:
  - `head()` shows first few rows
  - `tail()` shows the last few rows
  - `View()` shows the data as a spreadsheet
  - `spec()` gives specification of column types

## Data input: other file types

- From `readr` package:
  - `read_delim()`: general delimited files
  - `read_csv()`: comma separated (CSV) files
  - `read_tsv()`: tab separated files
  - others
- For reading Excel files, you can do one of:
  - use `read_excel()` function from `readxl` package
  - use other packages: `xlsx`, `openxlsx`

## Data input: other file types

- haven package has functions to read SAS, SPSS, Stata formats

```
library(haven)

# SAS
read_sas(file = "mtcars.sas7bdat")

# SPSS
read_sav(file = "mtcars.sav")

# Stata
read_dta(file = "mtcars.dta")
```