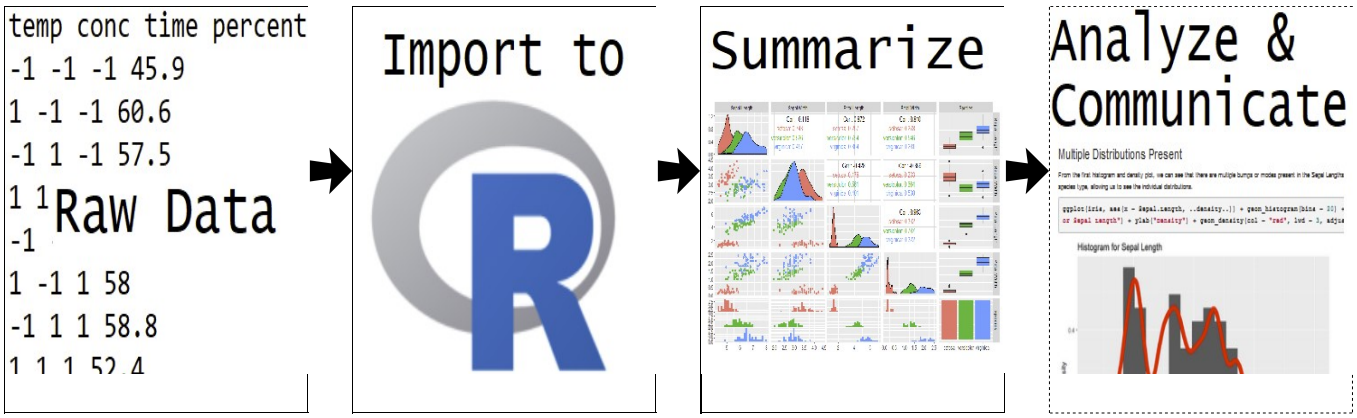
NC STATE UNIVERSITY

Importing Data: Basics & Delimited Data

Justin Post

What is this course about?

Basic use of R for reading, manipulating, and plotting data!



What is this course about?

Basic use of R for reading, manipulating, and plotting data!

- read and write basic R programs
- **import well formatted data into R**
- do basic data manipulation in R
- produce common numerical and graphical summaries in R
- describe a use case of an analysis done in R

Where do we start?

- Common raw data formats
- Comma Separated Value (CSV) files
- Asides: R projects and R packages
- Read ‘clean’ delimited data
- Excel, SAS, & SPSS data
- Resources for JSON, XML, databases, and APIs

Importing Data

How to read in data depends on raw/external data type!

- Lecture focus: Delimited data
  - Delimiter - Character (such as a `,`) that separates data entries

```
Treatment,Sex,Age,Duration,Pain
P,F,68,1,No
B,M,74,16,No
P,F,67,30,No
P,M,66,26,Yes
B,F,67,28,No
B,F,77,16,No
A,F,71,12,No
B,F,72,50,No
B,F,76,9,Yes
A,M,71,17,Yes
```

Comma: usually .csv

```
temp conc time percent
-1 -1 -1 45.9
1 -1 -1 60.6
-1 1 -1 57.5
1 1 -1 58.6
-1 -1 1 53.3
1 -1 1 58
-1 1 1 58.8
```

Space: usually .txt or .dat

```
"color" "spine" "width" "satell"      "weight"      "y"
3       3       28.3    8       3050  1
4       3       22.5    0       1550  0
2       1       26      9       2300  1
4       3       24.8    0       2100  0
4       3       26      4       2600  1
3       3       23.8    0       2100  0
2       1       26.5    0       2350  0
4       2       24.7    0       1900  0
3       1       23.7    0       1950  0
4       3       25.6    0       2150  0
4       3       24.3    0       2150  0
3       3       25.8    0       2650  0
3       3       28.2    11      3050  1
5       2       21      0       1850  0
3       1       26      14      2300  1
2       1       27.1    8       2950  1
3       3       25.2    1       2000  1
3       3       29      1       3000  1
5       3       24.7    0       2200  0
```

Tab: usually .tsv or .txt

```
2012>4>12>MIN>LAA>D.J. Reyburn
2012>4>12>SD>ARI>Marty Foster
2012>4>12>WSH>CIN>Mike Everitt
2012>4>12>PHI>MIA>Jeff Nelson
2012>4>12>CHC>MIL>Fieldin Culbreth
2012>4>12>LAD>PIT>Wally Bell
2012>4>12>TEX>SEA>Doug Eddings
2012>4>12>COL>SF>Ron Kulpa
2012>4>12>DET>TB>Mark Carlson
2012>4>13>NYY>LAA>Mike DiMuro
2012>4>13>COL>ARI>Mark Wegner
```

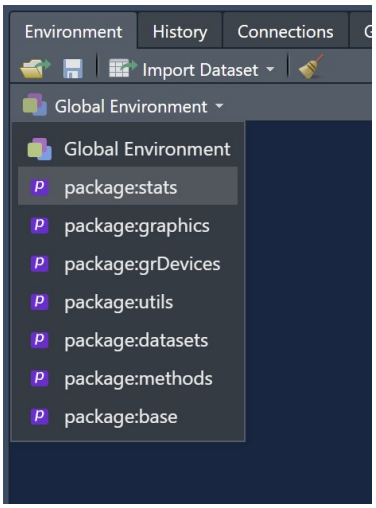
General: usually .txt or .dat

Importing Delimited Data: Standard R Methods

- When you open R a few `packages` are loaded
- R package
  - Collection of functions/datasets/etc. in one place
  - Packages exist to do almost anything
  - List of CRAN approved packages on R's website
  - Plenty of other packages on places like GitHub

Importing Delimited Data: Standard R Methods

- When you open R a few `packages` are loaded



- `utils` package has *family* of `read.` functions ready for use!

Importing Delimited Data: Standard R Methods

Function and purpose:

Type of Delimeter	Function
Comma	`read.csv()`
Semicolon (`,` for decimal)	`read.csv2()`
Tab	`read.delim()`
White Space/General	`read.table(sep = "")`

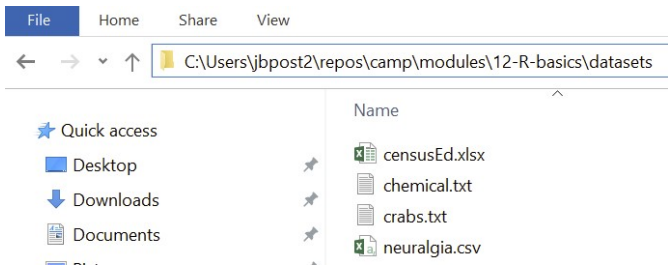
- Each function requires a **path** to the file

Reading a .csv File

- Let's read in the 'neuralgia.csv' file
- How does R locate the file?

Path to File

- Let's read in the 'neuralgia.csv' file
- How does R locate the file?
  - Can give *full path name*
    - ex: C:/Users/jbpost2/repos/camp/modules/12-R-basics/datasets/neuralgia.csv
    - ex: C:\\Users\\jbpost2\\repos\\camp\\modules\\12-R-basics\\datasets\\neuralgia.csv



Reading a .csv File

- Let's read in the 'neuralgia.csv' file
- Use full local path

```
neuralgiaData <- read.csv(
           "C:/Users/jbpost2/repos/camp/modules/12-R-basics/datasets/neuralgia.csv"
           )
```

```
head(neuralgiaData)
```

```
##   Treatment Sex Age Duration Pain
## 1         P   F  68        1   No
## 2         B   M  74       16   No
## 3         P   F  67       30   No
## 4         P   M  66       26  Yes
## 5         B   F  67       28   No
## 6         B   F  77       16   No
```

Working Directory

- Let's read in the 'neuralgia.csv' file
- Using full local path not recommended!
    - Can't share code without changing path…

Working Directory

- Let's read in the 'neuralgia.csv' file
- Using full local path not recommended!
  - Can't share code without changing path…
- Can change *working directory*
  - Folder where R 'looks' for files
  - Supply **relative** path

Working Directory

- Let's read in the 'neuralgia.csv' file
- Using full local path not recommended!
  - Can't share code without changing path…
- Can change *working directory*
  - Folder where R 'looks' for files
  - Supply **relative** path

```
getwd()
```

```
## [1] "C:/Users/jbpost2/repos/camp/modules/12-R-basics/Module2_ImportingData"
```

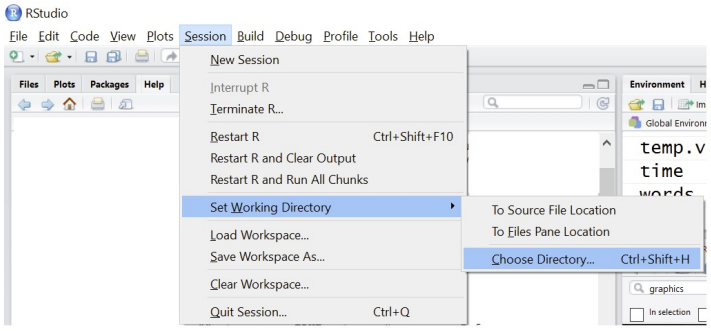
Working Directory

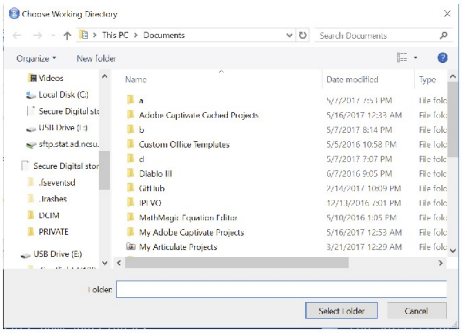
- Can change *working directory*
  - Via code

```
setwd("C:/Users/jbpost2/repos/camp/modules/12-R-basics/datasets")
#or
setwd("C:\\Users\\jbpost2\\repos\\camp\\modules\\12-R-basics\\datasets")
```

Working Directory

- Can change *working directory*
  - Via code
  - Via menus





Reading a .csv File

- Let's read in the 'neuralgia.csv' file
- Use relative path (`../` drops down a folder)

```
neuralgiaData <- read.csv("../datasets/neuralgia.csv")
```

- Working directory: "…/12-R-basics/Module2_ImportingData"
- File location: "…/12-R-basics/datasets/neuralgia.csv
- As long others have the same folder structure, can share code with no path change needed!

Reading a .csv File

- Let's read in the 'neuralgia.csv' file
- R can pull from URLs as well!

```
neuralgiaData <- read.csv("https://www4.stat.ncsu.edu/~online/datasets/neuralgia.csv")
head(neuralgiaData)
```

```
##   Treatment Sex Age Duration Pain
## 1         P   F  68        1   No
## 2         B   M  74       16   No
## 3         P   F  67       30   No
## 4         P   M  66       26  Yes
## 5         B   F  67       28   No
## 6         B   F  77       16   No
```

Reading a .csv File

`read.csv()` function

Tell R where the file lives via:

- a full local path (not recommended)
- a relative path
  - can set the working directory with `setwd()`
- pulling from URL

Aside: RStudio Project

- Often have many files associated with an analysis
- With multiple analyses things get cluttered…

Aside: RStudio Project

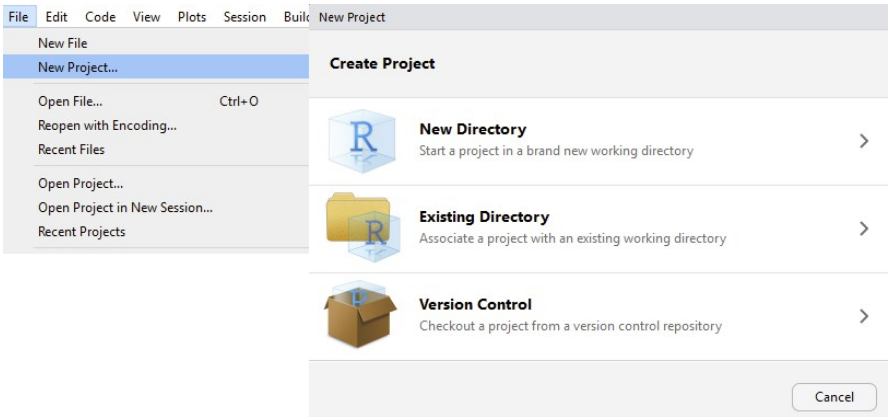
- Often have many files associated with an analysis
- With multiple analyses things get cluttered…
- Want to associate different
  - environments
  - histories
  - working directories
  - source documents

with each analysis

- Can use “Project” feature in R Studio

Aside: RStudio - Project

- Easy to create! Use an existing folder or create one:



- Easily switch between analyses!
- Create one for today's lesson
- Swap between projects using menu in top right

Reading Delimited Data

- Functions from `read.` family work well
- Concerns:
  - (formerly, prior to R 4.0) poor default function behavior
    - strings were read as `factors`

Reading Delimited Data

- Functions from `read.` family work well
- Concerns:
  - poor default function behavior
    - (formerly, prior to R 4.0) strings are read as `factors`
    - row & column names can be troublesome
  - Slower processing
  - (Slightly) different behavior on different computers

Aside: R Packages

- R package
  - Collection of functions in one place
  - Packages exist to do almost anything
  - List of CRAN approved packages on R's website
  - Plenty of other packages on places like GitHub
- "TidyVerse" - collection of R packages that share common philosophies and are designed to work together!

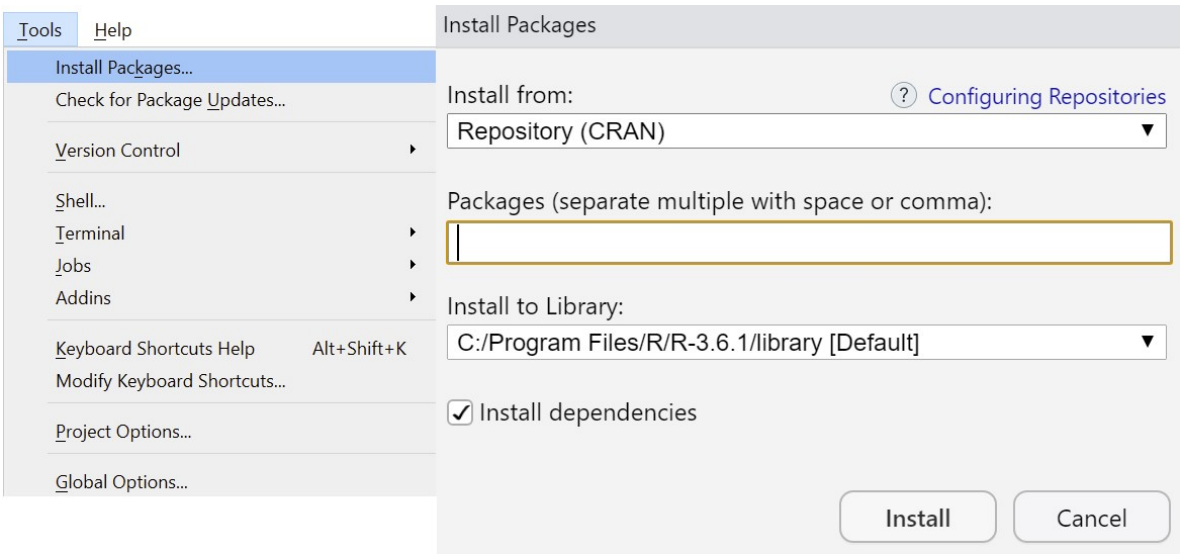
Aside: R Packages

- First time using a package
  - Must install package (download files)
  - Can use code or menus

```
install.packages("readr")
#can do multiple packages at once
install.packages(c("readr", "readxl", "haven", "DBI", "httr"))
```

Aside: R Packages

- First time using a package
  - Must install package (download files)
  - Can use code or menus



Aside: R Packages

- Only install once!
- **Each session**: read in package using `library()` or `require()`

```
library("readr")
require("haven")
```

Aside: R Packages

- Difference - if no package
  - `library()` throws an error
  - `require()` returns FALSE

```
library("notAPackage")
```

```
## Error in library("notAPackage"): there is no package called 'notAPackage'
```

```
require("notAPackage")
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'notAPackage'
```

Aside: R Packages

- Many packages to do things in R
- How to choose?
  - Want ‘fast’ code
  - Want ‘easy’ syntax
  - Good default settings on functions
  - Nice set of examples and vignettes
- Enter: TidyVerse

Aside: R Packages

- Install the `tidyverse` package

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

Aside: R Packages

- Install the `tidyverse` package

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

- Load library

```
library(tidyverse)
```

- Once library loaded, check `help(filter)`

Aside: R Packages

- Can call functions without loading full library with `::`
- If not specified, most recently loaded package takes precedent

```
dplyr::filter(neuralgiaData, Treatment == "P")
```

```
##    Treatment Sex Age Duration Pain
## 1          P   F  68        1   No
## 2          P   F  67       30   No
## 3          P   M  66       26  Yes
## 4          P   F  64        1  Yes
## 5          P   M  74        4   No
## 6          P   M  70        1  Yes
## 7          P   M  83        1  Yes
## 8          P   M  77       29  Yes
## 9          P   F  79       20  Yes
## 10         P   M  78       12  Yes
## 11         P   M  66        4  Yes
## 12         P   F  65       29   No
## 13         P   M  60       26  Yes
## 14         P   F  72       27   No
## 15         P   F  70       13  Yes
## 16         P   F  68       27  Yes
## 17         P   M  68       11  Yes
## 18         P   M  67       17  Yes
## 19         P   F  67        1  Yes
## 20         P   F  72       11  Yes
```

Aside: R Packages

Install packages first (download it)

- Can do more than one at a time

Load package with `require()` or `library()`

- Call without loading using `::`

Reading Delimited Data

We'll use the `tidyverse`!

Function and purpose:

Type of Delimeter	`utils` Function	`readr` Function
Comma	`read.csv()`	`read_csv()`
Semicolon (`,` for decimal)	`read.csv2()`	`read_csv2()`
Tab	`read.delim()`	`read_tsv()`
General	`read.table(sep = "")`	`read_delim()`
White Space	`read.table(sep = "")`	`read_table()` `read_table2()`

Reading Delimited Data

- Let's read in the 'neuralgia.csv' file

```
neuralgiaData2 <- readr::read_csv("https://www4.stat.ncsu.edu/~online/datasets/neuralgia.csv")
```

```
## Parsed with column specification:
## cols(
##   Treatment = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   Duration = col_double(),
##   Pain = col_character()
## )
```

Reading Delimited Data

- Let's read in the 'neuralgia.csv' file

```
neuralgiaData2
```

```
## # A tibble: 60 x 5
##   Treatment Sex     Age Duration Pain
##   <chr>     <chr> <dbl>    <dbl> <chr>
## 1 P         F        68        1 No
## 2 B         M        74       16 No
## 3 P         F        67       30 No
## 4 P         M        66       26 Yes
## 5 B         F        67       28 No
## # … with 55 more rows
```

Reading Delimited Data

- Notice: fancy printing!
- Checking column type is a basic data validation step
- `tidyverse` data frames are called `tibbles`

```
class(neuralgiaData2)
```

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

tibbles

- Behavior slightly different than a standard `data frame`. No simplification!

```
neuralgiaData[,1]
```

```
##  [1] P B P P B B A B B A A A B A P A P A P B B A A A B P B B P P A A B B B A P B
## [39] B P P P A B A P P A B P P P B A P A P A B A
## Levels: P B A
```

```
neuralgiaData2[,1]
```

```
## # A tibble: 60 x 1
##   Treatment
##   <chr>
## 1 P
## 2 B
## 3 P
## 4 P
## 5 B
## # … with 55 more rows
```

tibbles

- Behavior slightly different than a standard `data frame`. No simplification!
- Use either `pull()` or `$`

```
pull(neuralgiaData2, 1) #or pull(neuralgiaData2, Treatment)
```

```
##  [1] "P" "B" "P" "P" "B" "B" "A" "B" "B" "A" "A" "A" "B" "A" "P" "A" "P" "A" "P"
## [20] "B" "B" "A" "A" "A" "B" "P" "B" "B" "P" "P" "A" "A" "B" "B" "B" "A" "P" "B"
## [39] "B" "P" "P" "P" "A" "B" "A" "P" "P" "A" "B" "P" "P" "P" "B" "A" "P" "A" "P"
## [58] "A" "B" "A"
```

```
neuralgiaData2$Treatment
```

```
##  [1] "P" "B" "P" "P" "B" "B" "A" "B" "B" "A" "A" "A" "B" "A" "P" "A" "P" "A" "P"
## [20] "B" "B" "A" "A" "A" "B" "P" "B" "B" "P" "P" "A" "A" "B" "B" "B" "A" "P" "B"
## [39] "B" "P" "P" "P" "A" "B" "A" "P" "P" "A" "B" "P" "P" "P" "B" "A" "P" "A" "P"
## [58] "A" "B" "A"
```

Reading Delimited Data

- How did R determine the column types?

```
help(read_csv)
```

- Other useful inputs:
  - `skip = 0`
  - `col_names = TRUE`
  - `na = c("", "NA")`

Reading Delimited Data

- Reading *clean* delimited data pretty easy!
- Let's read in the 'chemical.txt' file (space delimited)
- `read_table2()` allows multiple white space characaters between entries

Reading Delimited Data

- Reading *clean* delimited data pretty easy!
- Let's read in the 'chemical.txt' file (space delimited)
- `read_table2()` allows multiple white space characaters between entries

```
read_table2("https://www4.stat.ncsu.edu/~online/datasets/chemical.txt")
```

```
## # A tibble: 19 x 4
##     temp  conc  time percent
##    <dbl> <dbl> <dbl>   <dbl>
##  1  -1    -1    -1      45.9
##  2   1    -1    -1      60.6
##  3  -1     1    -1      57.5
##  4   1     1    -1      58.6
##  5  -1    -1     1      53.3
##  6   1    -1     1      58
##  7  -1     1     1      58.8
##  8   1     1     1      52.4
##  9  -2     0     0      46.9
## 10   2     0     0      55.4
## 11   0    -2     0      55
## 12   0     2     0      57.5
## 13   0     0    -2      56.3
## 14   0     0     2      58.9
## 15   0     0     0      56.9
## 16   2    -3     0      61.1
## 17   2    -3     0      62.9
## 18  -1.4   2.6   0.7    60
## 19  -1.4   2.6   0.7    60.6
```

Reading Delimited Data

- Reading *clean* delimited data pretty easy!
- Let's read in the 'crabs.txt' file (tab delimited)

Reading Delimited Data

- Reading *clean* delimited data pretty easy!
- Let's read in the 'crabs.txt' file (tab delimited)

```
read_tsv("https://www4.stat.ncsu.edu/~online/datasets/crabs.txt")
```

```
## # A tibble: 173 x 6
##   color spine width satell weight     y
##   <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
## 1     3     3  28.3      8   3050     1
## 2     4     3  22.5      0   1550     0
## 3     2     1  26        9   2300     1
## 4     4     3  24.8      0   2100     0
## 5     4     3  26        4   2600     1
## # ... with 168 more rows
```

Reading Delimited Data

- Reading *clean* delimited data pretty easy!
- Let's read in the 'umps2012.txt' file ('>' delimited)
- Notice no column names provided
    - Year Month Day Home Away HPUmpire

Reading Delimited Data

- Reading *clean* delimited data pretty easy!
- Let's read in the 'umps2012.txt' file ('>' delimited)
- Notice no column names provided
    - Year Month Day Home Away HPUmpire

```
read_delim("https://www4.stat.ncsu.edu/~online/datasets/umps2012.txt", delim = ">",
           col_names = c("Year", "Month", "Day", "Home", "Away", "HPUmpire"))
```

```
## # A tibble: 2,359 x 6
##    Year Month   Day Home  Away  HPUmpire
##   <dbl> <dbl> <dbl> <chr> <chr> <chr>
## 1  2012     4    12 MIN   LAA   D.J. Reyburn
## 2  2012     4    12 SD    ARI   Marty Foster
## 3  2012     4    12 WSH   CIN   Mike Everitt
## 4  2012     4    12 PHI   MIA   Jeff Nelson
## 5  2012     4    12 CHC   MIL   Fieldin Culbreth
## # ... with 2,354 more rows
```

Reading Fixed Field & Tricky Non-Standard Data

- read_fwf()
  - reads in data where entries are very structured
- read_file()
  - reads an entire file into a single string
- read_lines()
  - reads a file into a character vector with one element per line
- Usually parse the last two with `regular expressions` :(

Next Up!

- Read data from other sources

Type of file	Package	Function
Delimited	`readr`	`read_csv()`, `read_tsv()`, `read_table()`, `read_delim()`
Excel (.xls,.xlsx)	`readxl`	`read_excel()`
SAS (.sas7bdat)	`haven`	`read_sas()`
SPSS (.sav)	`haven`	`read_spss()`

- Resources for JSON, XML, databases, and APIs