

Session 1: Introduction to R

Welcome!

TA: Jennifer Lin

MMSS 211: Institutions, Rules, & Models in Social Science

2022-03-25

Goals

1. Become familiar with the "lay of the land" of R and R Studio.
2. Understand the basic data structures for statistical computing and how to translate this theory to practice in R.
3. Introduce techniques to organize code and to troubleshoot issues.

Overview of R Session Structure

- Before class, I will post, to Canvas, the following:
 - Slides (PDF)
 - Handout (PDF)
 - Code (R Script and R Markdown)
- *Slides* are these current slides in PDF format
- *Handouts* are the narration to the slides that provide more detail about each topic
- *Code* files are for you to use during class sessions. You can choose between the R Script or R Markdown format. Both contain the same information.

My Philosophy

- The best way to learn R is to apply what you learn to novel situations that interest you.
- Class will integrate lecture with hands on exercises that have a "choose your adventure" component so you can pick topics of interest to apply the code.
- Learning R is collaborative so section will be rather interactive in nature.

What is R?

R

```
R version 4.1.1 (2021-08-10) -- "Kick Things"  
Copyright (C) 2021 The R Foundation for Statistical Computing  
Platform: x86_64-apple-darwin17.0 (64-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.
```

```
  Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

R is...

- free software and comes with ABSOLUTELY NO WARRANTY.
- a collaborative project with many contributors.

Difference between R and R Studio

You need to write a paper for a class. There are many ways to do it.

- Microsoft Word
- Pages
- Google Docs

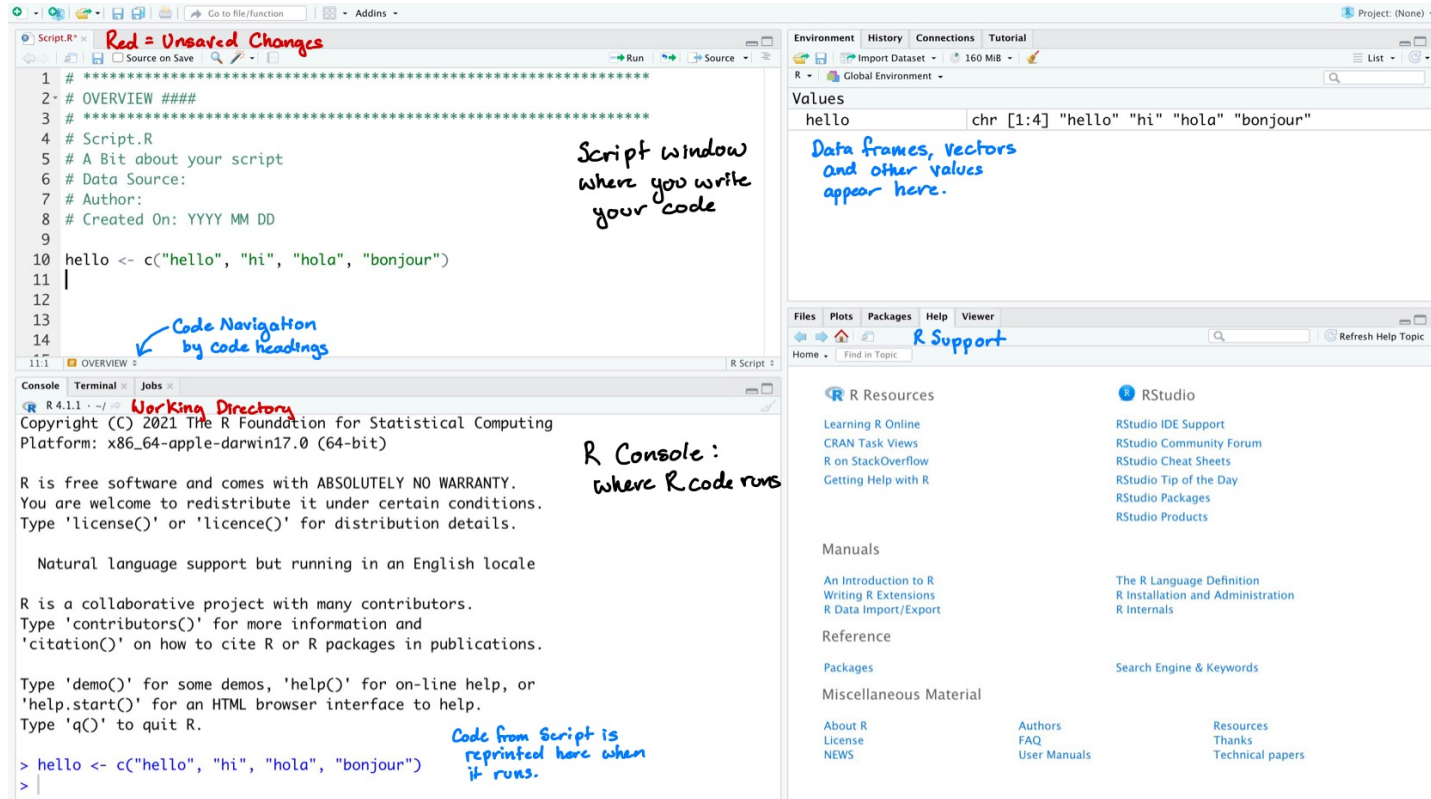
All access the same underlying software structure (your computer) to generate a document with words on virtual paper.

R and R Studio operate similarly

Difference between R and R Studio

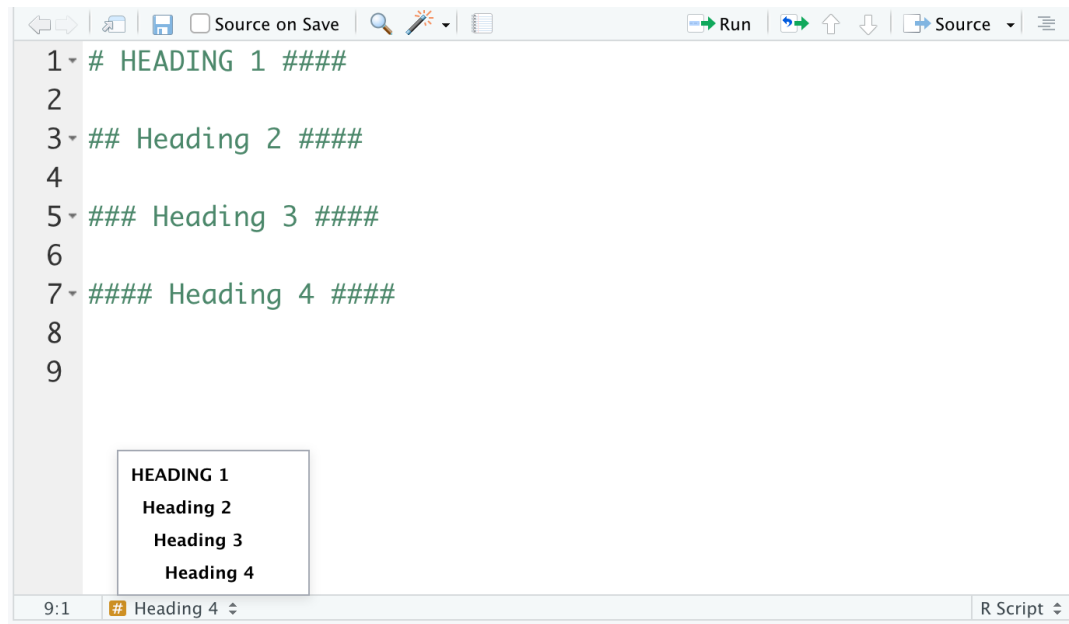
- R is the software for statistical computing
- R Studio is the means for you to do it -- it is the "Microsoft Word" interface for code writing
 - There are Pages/Google Docs equivalents for writing R code
 - But I do not recommend looking into these unless you are super comfortable in R Studio

Layout of R Studio



Using R Scripts

- Insert comments using a # sign -- Tells R to ignore the text that follows this
- Anything not preceded by a # will be treated as code
- Section R scripts using multiple #'s after the heading.



The screenshot shows an R script editor window. The script contains four headings, each preceded by a specific number of hash symbols (#):

```
1 # HEADING 1 ####  
2  
3 ## Heading 2 ####  
4  
5 ### Heading 3 ####  
6  
7 #### Heading 4 ####  
8  
9
```

A preview window is open, showing the rendered headings:

```
HEADING 1  
Heading 2  
Heading 3  
Heading 4
```

The status bar at the bottom indicates the current line is 9:1 and the heading is Heading 4.

A Note About Writing Code

1. ALWAYS comment your code
2. Develop a taste for good code structure and adopt it for your code going forward

Good Code Structure Examples

- Proper indentation of code
- Limit the length of each line
- Create sections on your code

Your future self will thank you for a clean code script!

Introduction to Markdown

The screenshot shows an RStudio editor window titled 'Untitled1'. The top toolbar includes 'Spell Check', 'Knit', and 'Run' buttons. The document content is as follows:

```
1 ---
2 title: "Introduction to R Markdown"
3 output: html_document
4 ---
5
6 {r setup, include=FALSE}
7 knitr::opts_chunk$set(echo = TRUE)
8
9
10 ## R Markdown
11
12 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
13
14 When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
15
16 {r cars}
17 summary(cars)
18
19
20 ## Including Plots
21
22 You can also embed plots, for example:
23
24 {r pressure, echo=FALSE}
```

Handwritten annotations in the image include:

- Red text: "Use to compile" pointing to the Knit button.
- Blue text: "Add new code block (Defaults to R)" pointing to the '+' icon in the toolbar.
- Blue text: "This chunk sets the code preferences for Document: Default is to show all code (echo = TRUE)" pointing to the R code chunk header.
- Blue text: "Here is a standard line of R code in R Markdown" pointing to the `summary(cars)` line.
- Red text: "Denotes R chunk and Names it 'cars'" pointing to the `{r cars}` header.
- Red text: "File will compile as HTML - can change to PDF or Word" pointing to the `output: html_document` line.
- Black text: "YAML -> Document Properties" pointing to the YAML front matter.
- Black text: "R Code" pointing to the R code chunk.
- Black text: "Markdown Text" pointing to the text block.

What is Markdown?

- Plain text -- text formats through markings rather than by clicking things on a toolbar
- Markup language -- Include markings next to text to denote what is bold, italic, and so on

Markdown Syntax

Headings

You can specify headings and subheadings in your document using the # before text that is meant to be a heading.

Heading 1

Heading 2

Heading 3

Markdown Syntax

Bold, Italic, Code Text

You can **BOLD** text by inserting two asterisks before and after
`**text that should be bolded**`

You can *italicize* text by inserting one asterisk before and after
`*text that should be italicized*`

Markdown Syntax

Math Equations

We can also render math in markdown using LaTeX syntax. Inline math equations need to be enclosed in dollar signs such that $ax^2 + bx + c = 0$ is typed as `$ax^2 + bx + c = 0$` and

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

as

```
$$  
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}  
$$
```

Oh and back ticks give us verbatim text.

R Markdown

- Serves as a place to write R code and description text
- Can be rendered as a PDF, HTML or Word document (You will need LaTeX installed to do PDF compiles)
- Compiles with "Knit" button on top of screen -- Careful!
This runs all of the code that you write each time!

R Markdown Code Chunks

- Runs R code like the R Scripts

```
```{r, warning=FALSE, message=FALSE}  
library(dplyr)
library(ggplot2)
library(tidyr)
```
```

- Code must be in the grey chunks and can be shown or hidden
 - `echo` = TRUE shows and evaluates code
 - `eval` = TRUE evaluates code, FALSE does not
 - `include` = TRUE includes the code in the output
 - `warning` = FALSE suppresses warnings
 - `message` = FALSE suppresses messages

Programming in R

The Mindset

- R is an **object oriented programming language**. This means that
 - Objects are the core of R
 - Everything in R is an object
- What are Objects?
 - Functions
 - Data sets
 - Packages
 - Anything you can point an arrow to
- Assign objects with the `<-`, which is the *assignment arrow*

The Grammar of R

In English: Verb(Noun, Adjective)

In Colloquial terms: Do Something(To What, How So)

Formally, `function(data, arguments)`

Example:

```
data <- c(1, 2, 3, 4, 5, 6)
mean(data, na.rm = TRUE)
```

Packages and Libraries

Earlier, we discussed how "R is a collaborative project with many contributors."

What is a Package?

It is an app for your statistics needs

| On Smartphone | In R |
|---------------|---|
| Download app | <code>install.packages("app_name")</code> |
| Open app | <code>library(app_name)</code> |

R's package library is the **Comprehensive R Archive Network (CRAN)**, which houses most packages.

Data Structures

Types of Data

| Type | Example |
|----------------|-------------------------|
| Character | "dog", "cat", "fish" |
| Double/Numeric | -1, 2.8, 3, 4.5, 5, 6.4 |
| Integer | 1, 2, 3, -4, 5 |
| Logical | TRUE, FALSE |
| Complex | $1+4i$ |

Simple Data Formats

Atomic Vectors

| | | |
|------|----------|------|
| 12.2 | "Python" | TRUE |
| 15.6 | "R" | |
| | "Stata" | |

Data Frame

| ID | Gender | Age |
|----|--------|-----|
| 1 | Male | 42 |
| 2 | Female | 37 |
| 3 | Female | 51 |
| 4 | Male | 22 |

Matrix

| | | | |
|---|---|---|---|
| 1 | 1 | 3 | 0 |
| 0 | 1 | 4 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |

(A Bit More) Complex Data Formats

List

| A | B | C |
|---|---|---|
| | | |

Vector (1, 2, 5, 9, 7)

Function $Y = mx + b$

Another list...

- Lists include all sorts of basic data structures
- They can be useful but also a hassle to work with

Vectors

Make a Vector

- Vectors are a collection of values that are tied to an index that starts with 1.
- The function `c()` (concatenate) makes vectors for you. Put all vector elements within `c()`
- Access index by appending `[]` to vector name

```
stats_programs <- c("Python", "R", "Stata", "SPSS", "SAS")  
stats_programs[2]
```

```
## [1] "R"
```

Vectors

Extracting Values

```
fibonacci <- c(  
  1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144)
```

- Let's get the first 10 numbers of the Fibonacci Sequence

```
fibonacci[1:10]
```

```
## [1] 1 1 2 3 5 8 13 21 34 55
```

- Let's get the first, third, and tenth number in the Sequence

```
fibonacci[c(1, 3, 10)]
```

```
## [1] 1 2 55
```

Vectors

Editing Values

- Whatever math we do to the vector is applied to each element

```
fibonacci + 1
```

```
## [1] 2 2 3 4 6 9 14 22 35 56 90 145
```

```
fibonacci * 4
```

```
## [1] 4 4 8 12 20 32 52 84 136 220 356 576
```

Vectors

Editing Values

- You can edit a value using the index of a vector
- CAUTION: NEVER Write over the original data -- actions cannot be undone

```
new_fibonacci <- fibonacci  
new_fibonacci[2] <- 100  
  
fibonacci
```

```
## [1] 1 1 2 3 5 8 13 21 34 55 89 144
```

```
new_fibonacci
```

```
## [1] 1 100 2 3 5 8 13 21 34 55 89 144
```

Data Frames

Base R

```
state_abbrev <- c("AL", "AK", "AZ", "AR", "CA")
state_fips <- c(1, 2, 4, 5, 6)
state_names <- c("Alabama", "Alaska", "Arizona", "Arkansas", "Cal-
```

```
states <- data.frame(
  state_abbrev,
  state_fips,
  state_names
)
```


Data Frames

Tidyverse

```
library(dplyr)

state_data <- tibble(
  state_abbv,
  state_fips,
  state_names
)
```

| state_abbv | state_fips | state_names |
|------------|------------|-------------|
| AL | 1 | Alabama |
| AK | 2 | Alaska |
| AZ | 4 | Arizona |
| AR | 5 | Arkansas |
| CA | 6 | California |

Data from packages

```
library(tidycensus)
```

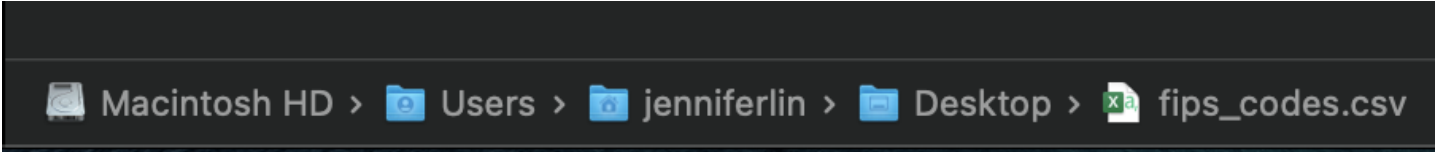
```
data(fips_codes)
```

| state | state_code | state_name | county_code | county |
|-------|------------|------------|-------------|----------------|
| AL | 01 | Alabama | 001 | Autauga County |
| AL | 01 | Alabama | 003 | Baldwin County |
| AL | 01 | Alabama | 005 | Barbour County |
| AL | 01 | Alabama | 007 | Bibb County |
| AL | 01 | Alabama | 009 | Blount County |
| AL | 01 | Alabama | 011 | Bullock County |

Loading External Data

A Note on File Paths

- File paths are a string containing folder names and end with the name of the particular file of interest.
- You can find them in your documents folder.

A screenshot of a Mac OS Finder window showing a file path. The path is displayed as a sequence of icons and text: a hard drive icon for 'Macintosh HD', a folder icon for 'Users', a folder icon for 'jenniferlin', a folder icon for 'Desktop', and a CSV file icon for 'fips_codes.csv'. The path is separated by greater-than symbols (>).

Macintosh HD > Users > jenniferlin > Desktop > fips_codes.csv

- Being able to locate your file path is important for locating files
- If you put an R script and a data file in the same folder and open R from that script, you should be able to skip this step. Otherwise, you need to tell R where your data file is.

Loading External Data

Reading Different File Types -- CSV

```
fips_csv <- read.csv("fips_codes.csv")
```

| X | state | state_code | state_name | county_code | county |
|---|-------|------------|------------|-------------|----------------|
| 1 | AL | 1 | Alabama | 1 | Autauga County |
| 2 | AL | 1 | Alabama | 3 | Baldwin County |
| 3 | AL | 1 | Alabama | 5 | Barbour County |
| 4 | AL | 1 | Alabama | 7 | Bibb County |
| 5 | AL | 1 | Alabama | 9 | Blount County |
| 6 | AL | 1 | Alabama | 11 | Bullock County |

Loading External Data

Reading Different File Types -- CSV (`readr`)

```
library(readr)
fips_csv <- read_csv("fips_codes.csv")
```

| ...1 | state | state_code | state_name | county_code | county |
|------|-------|------------|------------|-------------|----------------|
| 1 | AL | 01 | Alabama | 001 | Autauga County |
| 2 | AL | 01 | Alabama | 003 | Baldwin County |
| 3 | AL | 01 | Alabama | 005 | Barbour County |
| 4 | AL | 01 | Alabama | 007 | Bibb County |
| 5 | AL | 01 | Alabama | 009 | Blount County |
| 6 | AL | 01 | Alabama | 011 | Bullock County |

Loading External Data

Reading Different File Types -- R Data

```
load("fips_codes.RData")
```

| state | state_code | state_name | county_code | county |
|-------|------------|------------|-------------|----------------|
| AL | 01 | Alabama | 001 | Autauga County |
| AL | 01 | Alabama | 003 | Baldwin County |
| AL | 01 | Alabama | 005 | Barbour County |
| AL | 01 | Alabama | 007 | Bibb County |
| AL | 01 | Alabama | 009 | Blount County |
| AL | 01 | Alabama | 011 | Bullock County |

Loading External Data

Reading Different File Types -- JSON

```
library(rjson)
```

```
fips_json <- fromJSON(file = "fips_codes.json") %>%  
  as.data.frame()
```

| state | state_code | state_name | county_code | county |
|-------|------------|------------|-------------|----------------|
| AL | 01 | Alabama | 001 | Autauga County |
| AL | 01 | Alabama | 003 | Baldwin County |
| AL | 01 | Alabama | 005 | Barbour County |
| AL | 01 | Alabama | 007 | Bibb County |
| AL | 01 | Alabama | 009 | Blount County |
| AL | 01 | Alabama | 011 | Bullock County |

Loading External Data

Reading Different File Types -- Stata

```
library(haven)

fips_stata <- read_dta("fips_codes.dta")
```

| state | state_code | state_name | county_code | county |
|-------|------------|------------|-------------|----------------|
| AL | 01 | Alabama | 001 | Autauga County |
| AL | 01 | Alabama | 003 | Baldwin County |
| AL | 01 | Alabama | 005 | Barbour County |
| AL | 01 | Alabama | 007 | Bibb County |
| AL | 01 | Alabama | 009 | Blount County |
| AL | 01 | Alabama | 011 | Bullock County |

Accessing Variables

- You can access variables in a data frame using the \$
- \$ means "at"

```
fips_codes[1:3,1:5]
```

```
##      state state_code state_name county_code
## 1      AL           01    Alabama         001
## 2      AL           01    Alabama         003
## 3      AL           01    Alabama         005
##
##           county
## 1 Autauga County
## 2 Baldwin County
## 3 Barbour County
```

```
head(fips_codes$county)
```

```
## [1] "Autauga County" "Baldwin County" "Barbour County"
## [4] "Bibb County"    "Blount County"  "Bullock County"
```

Troubleshooting R Code

Step 1: Check Your Code

Check your code for

- Typos in object references or function names
- Missing open or close parentheses
- Missing + or %>%
- Upper or lower case -- R is case sensitive!

Step 2: Look at the R Help Page

Use `?package::function()` to load the R help page

Read the argument descriptions and examples

Step 3: Google!

1. Draw (on paper) your desired end result and find words around that
2. Use these words to craft a Google search
3. See what Stack Overflow has to say
4. If nothing useful comes up, take advantage of related searches

Step 4: Schedule a Meeting with Me

I am always happy to help!

Email me at jenniferlin2025@u.northwestern.edu with questions.