IST 687: Using R to manipulate data

Corey Jackson

2021-01-20 19:16:29

Agenda

- Announcements
- Review of Week 1 (Async; Chapters 1-3)
- Breakout I (Complete Lab 2)
- Homework 2 Tips
- Next week's agenda
- Final Project Group Meeting

Announcements

- More course information
  - Handling the asynchronous materials
  - Office Hours: 5-6pm CT Wednesday
- Homework and lab 1 answers are linked in the syllabus
- Questions/issues submitting homework and Labs using RMarkdown?
- Communicating via SLACK. Please join #homework and #lab channels

Review of Week 1

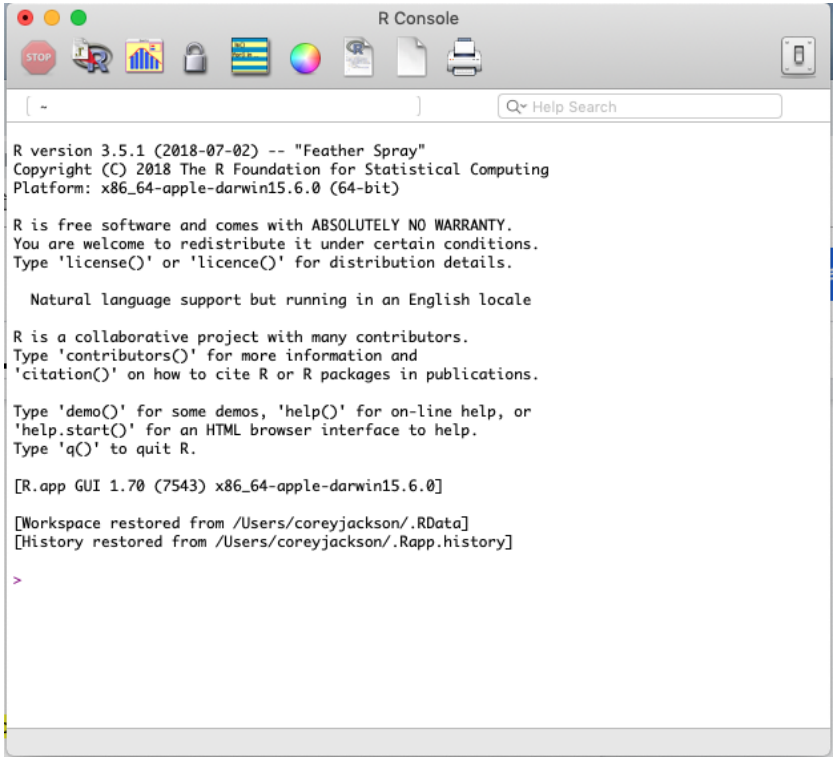
Overview of Week 1

- R, RStudio, and RMarkdown
- Programming in R
  - Data Structures
  - Functions
  - Objects and Variables
  - Conditionals

Overview of Week 1: R

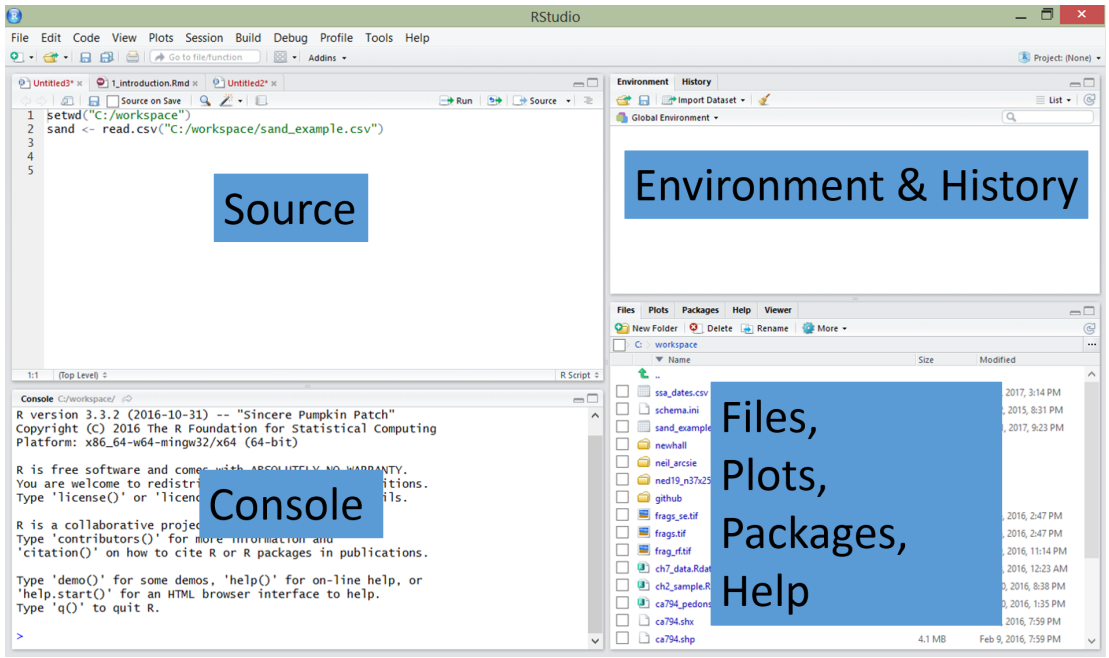
R is a free, open-source *software* and *programming language* developed as an environment for statistical computing and graphics.

R offers numerous advantages, such as: free, community, learning resources, and one of the most popular tools for doing data science.



Overview of Week 1: RStudio

The RStudio integrated development environment (IDE) Cheatsheet



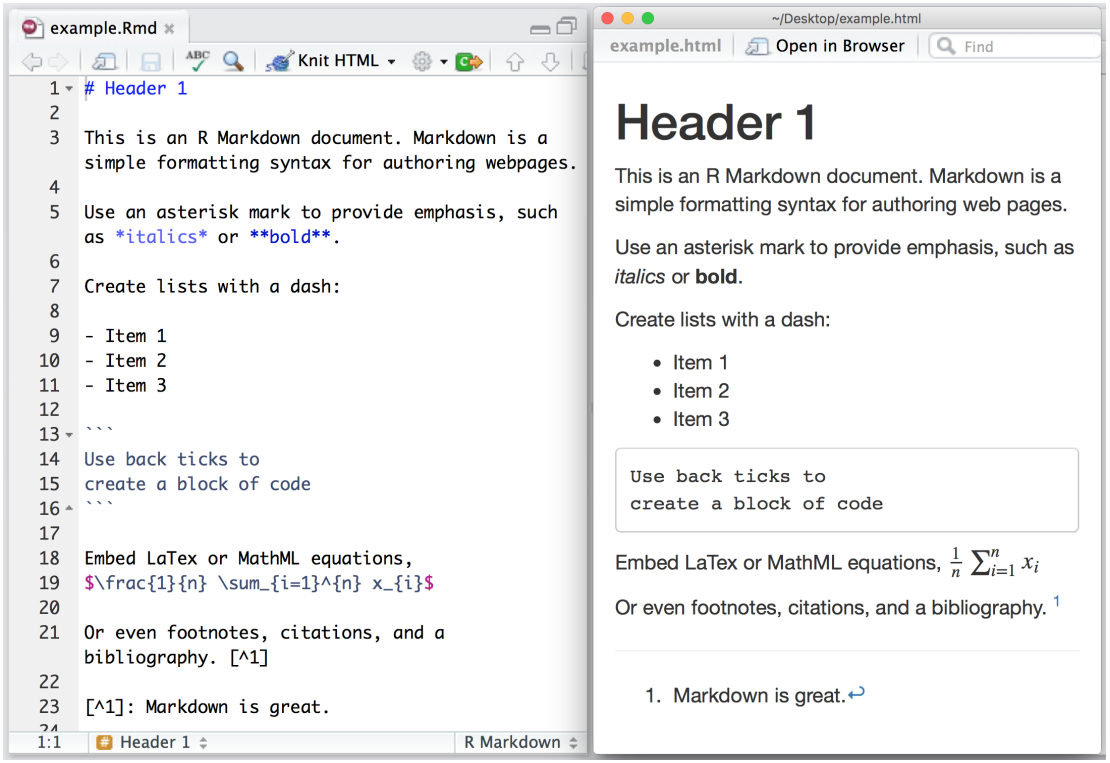
- **Source**: type/edit R scripts
- **Console**: directly enter commands for immediate execution by R interpreter
- **Environment & History**: view R objects (e.g., dataframes, functions), past commands, establish external connections
- **Notebook**: file explorer to navigate local folders, view plots, packages, and help documentation

Overview of Week 1: RStudio Tips

- R is case sensitive. Make sure your spelling and capitalization are correct
- Access the history of code in the console using the ↑ when working in the console
- R objects can be opend in the source pane (double-click the object in environment pane)
- Get help with function and other R objects Help!! `??[function name]` e.g., to know more about the mtcars dataset type `??mtcars` in the console

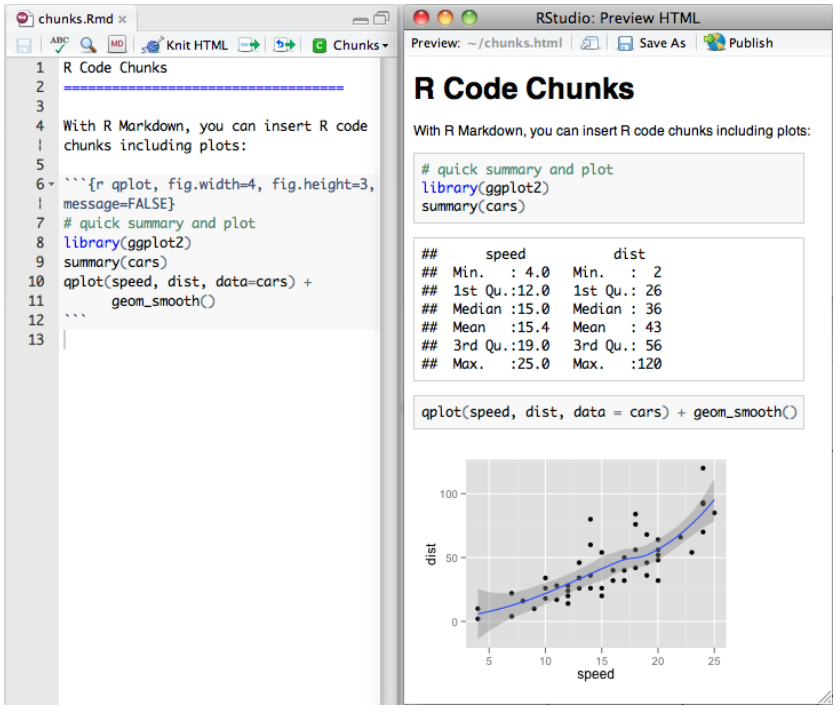
Overview of Week 1: RMarkdown

- RMarkdown a lightweight markup language converted to other formats like HTML or pdf. Example RMarkdown documents.



Overview of Week 1: RMarkdown

- Work inside code chuncks (i.e., gray areas in the .Rmd file). This is how R knows that is code to be executed as opposed to regular text



- To render output, "Knit" the markdown to an html, pdf, docx file type

To learn about formatting: RMarkdown Cheetsheet

Overview of Week 1: Programming

- Data Structures/Types
- Calling Functions
- Variables
- Conditionals

Overview of Week 1: Data Structures

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

In this class, we'll focus on:

Atomic vectors: a sequence of data components of the same basic type which can be logical, integer, double, character, complex or raw

`c(59,60,61,58,67,72,70)` and `c("Hello","world")` are both vectors

Data frames: a list of vectors of equal length (think tabular data in spreadsheets)

Data Structures: Atomic vectors

Atomic vectors have three common properties:

- Type, `typeof()`, what it is.
- Length, `length()`, how many elements it contains.
- Attributes, `attributes()`, additional arbitrary metadata.

Four common types of atomic vectors: logical, integer, double or numeric, and character.

Note: Data types need not be defined a priori.

Data Structures: Atomic vectors

Logical: A logical value `log_var <- c(TRUE, FALSE, T, F)`
Integer: A whole number `int_var <- c(1L, 6L, 10L)`
Double: A decimal number `dbl_var <- c(1, 2.5, 4.5)`
Character: A text or character string `c("Hello", "world")`

Note: R decides on whether to assign class double or integer.

Data Structures: Atomic vectors

We can assign values to a vector using the `c()` function, which stands for concatenate/combine `height <- c(179.26,182.88,185.42,172.72,200.66,281.44,213.36)`. Anytime, `height` is typed in the console, the values in that vector are retrieved.

```
typeof(height)
```

```
## [1] "double"
```

```
length(height)
```

```
## [1] 7
```

```
attributes(height)
```

```
## NULL
```

Data Structures: Dataframes

- A data frame is the most common way of storing data in R and is composed of lists of equal-length vectors
- Data frames are tabular objects (think spreadsheets)

Dataframes three common properties:

- Names `names()`, names of vectors that comprise the data frame
- Column names `colnames()`, names of vectors
- Row names `rownames()`, names of rows

Data Structures: Dataframes

You create a data frame using `data.frame()`, which takes named vectors as input:

```
weight <- c(150,140,180,220,160,140,130)
measurements <- data.frame(height,weight)
measurements
```

```
##   height weight
## 1 179.26    150
## 2 182.88    140
## 3 185.42    180
## 4 172.72    220
## 5 200.66    160
## 6 281.44    140
## 7 213.36    130
```

Data Structures: Dataframes

Exploring the properties of our `measurements` dataframe:

`str(measurements)` - allows us to examine the structure of the dataframe

```
## 'data.frame':    7 obs. of  2 variables:
##  $ height: num  179 183 185 173 201 ...
##  $ weight: num  150 140 180 220 160 140 130
```

```
colnames(measurements)
```

```
## [1] "height" "weight"
```

```
rownames(measurements)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7"
```

```
length(measurements)
```

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

Calling Functions

Functions are a set of statements organized together to perform a specific task. Calling R functions requires take the form:

```
f(argument1, ...)
```

Computing the mean of `height` object. We can use the mean function with syntax:

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

where `mean` is the name of the function, `x` is the first argument a vector, and two others.

```
mean(height)
```

Note: R has many built in functions, and you can access more by installing new packages so we don't do much functional programming in 687.

Variables

To do useful and interesting things, we need to assign values to variables. Assign values to variables using: `=`, `<-`, `<<-`

Pointers about naming variables. You want your variable names shoudl be explicit and not too long. They cannot start with a number (2x is not valid, but x2 is). R is case sensitive (e.g., Ages is different from ages). There are reserved words (type `?Reserved` in the console).

Relational and Logical operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

Conditionals: Using conditionals with operators

```
ifelse(measurements$height >= 175.4,"AA","BA")
```

```
## [1] "AA" "AA" "AA" "BA" "AA" "AA" "AA"
```

```
measurements$status <- ifelse(measurements$height >= 175.4 |
measurements$weight <= ,"BA","AA")
```

```
##   height weight status
## 1 179.26    150     BA
## 2 182.88    140     BA
## 3 185.42    180     BA
## 4 172.72    220     AA
## 5 200.66    160     BA
## 6 281.44    140     BA
## 7 213.36    130     BA
```

Questions on Week 1

- Review appendix at the end of the slide deck for additional material
- Practice programming with Swirl.

```
install.packages("swirl") # install package
library(swirl) # load package
swirl() # use package
```

Lab

Lab 2: Manipulating Data frames

Goal: Explore dataframes in R

- Creating new variables
- Combining multiple vectors into a dataframe
- Examining the dataset

Lab 2: Data Frames & sorting (30 minutes)

Lab Description

Goal: using R expressions and functions to obtain summary information about a dataset and create new variables from existing data

Instructions: With your pair programming partner complete today's lab assignment. The person having the number highlighted next to their name for the week should have RStudio open, download the markdown file, submit the lab, send to partner, and submit on the LMS

Final Project

Final Project Overview

Goal: Put course learning into practice by completing a data science research project.

Instructions: (1) Acquire a dataset to work on throughout the semester, (2) Develop research questions (3) Answer those RQs using exploratory data analaysis and advanced data science methods e.g., regression, text mining

Deliverables: Presentation communicating the results of your research and a project summary document describing the research design and results in greater detail.

Team Assignments

Project Updates

A 10 minute meeting during live session in Weeks 3,7 and 10

Goal: Receive feedback from instructor on final project

Deliverables: Two deliverables to be submitted the night before the live session project updates: (1) Kanban Board and (2) Project update document

During the second project update, you will required to submit a Team Process Agreement (more on this in Project Update 2).

Project Update I

Goal: Brainstorm ideas for the final project. Select a dataset and research questions. Add these to the project update document

Homework 2 Tips

Assignment: Manipulating Data Frames

Step 3: You should write pseudo code not R code.

Step 4: Which car has "best"" car combination of mpg and hp, where mpg and hp must be given equal weight?

- Explore the `scale()` function
- Get information about scale by using the help documentation in R `??scale`
- Use programming sites: Stackoverflow, Rdocumentation.org, and r-bloggers.com

Next Week

Next Week's Agenda

- Asynchronous
  - Week 3: Descriptive Statistics & Functions; Chapters 7-9
  - Submit HW 2 and Lab 2
  - Project Update I
- Live Session
  - Lab 3: Descriptive Stats & Functions
  - Group Project Meeting (Project Update in Week 3)

Note: Brush up on your programming skills using Swirl

Appendix for Week 1

Week 1: Expressions, Variables, and Functions

Functions

- "a set of statements organized together to perform a specific task."
- Functions can be built-in or user-defined (Week 4)
- A typical expression for calling functions: `f(argument1, argument2, ...)`

```
mean(ages) # returns the mean of an object called ages
min(ages) # returns the minimum value in an object called ages
seq(1, 21,5) # returns values from 1 to 21 incremented by 5
```

R programming language contains built-in functions and some created by other developers which we can import
by installing packages

Week 1: Operators and Conditionals

Arithmetic and logical operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Figure 1: Arithmetic and logical

Control structures

```
if(condition) true_expression else false_expression
```

- If `person` is Corey and `birthdate` is 1987 print "Yes" otherwise print "No"

```
person <- "Corey"
birthyear <- 1987
if (person == "Corey" & birthyear == 1982) "Yes"  else "No"
```

Week 1: Commenting your code

- A good practice is using the # to comment your code
- Improves learning and reproducability of your code

```
mean(ages) # the expression computes the mean of an object names ages
```

Confusing operators and expressions

- long vs. short form operators: `&&` vs. `&`

```
ages <- c(20,25,10,15) ages
```

```
## [1] 20 25 10 15
```

```
ages >= 15 & ages == 20
```

```
## [1]  TRUE FALSE FALSE FALSE
```

```
ages >= 15 && ages == 20
```

```
## [1] TRUE
```

- Assigning values to objects using: `=`, `<-`, `<<-`

```
name <- "Corey"
name <<- "Corey"
name = "Corey"`
```

Confusing operators

- updating objects

```
ages <- c(20,25,10,15); ages
```

```
## [1] 20 25 10 15
```

```
ages + 5
```

```
## [1] 25 30 15 20
```

```
ages
```

```
## [1] 20 25 10 15
```

- Remember to "re-assign" the updated values to the object `ages <- ages + 5; ages`

```
## [1] 25 30 15 20
```