Data Types and Basic Variable Manipulation

http://datascience.tntlab.org

Module 2

Today's Agenda

- About Outside Resources
- Using DataCamp Effectively
- Highlights from *Introduction to R*
  - Variable assignments and comparators
  - Data types (atomic classes) vs. variable types
  - Functions
  - Comments
- New Skills for This Week's Project

About Outside Resources

- Cheatsheets
  - https://www.rstudio.com/resources/cheatsheets/
  - http://github.com/rstudio/cheatsheets/raw/master/source/pdfs/base-r.pdf

- Websites
  - Not now, but later

Using DataCamp Effectively

- Demonstration
  - Remember that R is running "beneath the hood"

Variable Assignment and Comparators

- <- or =
  - These are both called the “assignment operator”
  - <- only works as a standalone command, not within a function
  - It is pronounced “gets”: x = 4

- **Comparators**
  - <
  - >
  - ==
  - !=

- ***Output does not get assigned to a variable unless you tell it so.***

Data Types (also called *atomic classes*)

- **Numeric**
  - Floating-point: 4.5
  - Integer: 7 (sometimes indicated with L, e.g., 7L)
- **Logical (Boolean)**
  - Always all-caps and are *reserved words*
  - TRUE
  - FALSE
- **Factor**
  - Used for categorical data and their labels, usually characters
  - Note: this is not really a data type in the same way as the others
- **Character**
  - Also called *strings*
  - Always surrounded by quotes (either ' ' or " ")
  - “ “ is not the same as " “
- **Complex** (which we won’t use)

Variable Types

- **Vector**
  - Even single numbers are usually vectors: x <- 5 is a one-item vector
  - Can only contain a single data type (integer, character, etc.)
  - You can subset with either numbers or ranges and single brackets, e.g.,
    - x[1]
    - x[5:10]
    - x[c(1,2,3,4,5)]
- **Matrix**
  - **A multi-dimensional vector**
  - Always remember: down then across
  - You can subset by row, by column, by order, or by intersection, e.g.,
    - m[4,]
    - m[,4]
    - m[4]
    - m[4,4]
  - Common functions: rowSums, colSums, rowMeans, colMeans, rbind, cbind

Variable Types

- **Factors**
  - Not very common unless you're doing factor-related analysis, e.g., ANOVA
  - Involves redefining a vector of words as a factor
  - Communicates to R what sort of analysis are permitted given this variable

- **Lists**
  - Combinations of data types into a single data structure
  - Note [ ] vs [[ ]]: Single-brackets subset whereas double-brackets extract
    - If a list contains a matrix as its first item, [1] will return a single-item list containing the matrix whereas [[1]] will return a matrix
  - You really don't want to create lists unless you need to keep dissimilar data tied together, which is why…

Variable Types

- **Data Frames**
  - A special type of list where:
    - All list elements are vectors (which we colloquially call "variables")
    - List-element/vectors/variables are (usually) named, and no two can have the same name
    - All list element/vectors/variables must be of the same length
  - Because of the length and type restrictions, they look similar to matrices, and many matrix functions work with them… but they are technically lists
    - Common matrix functions also here: rowSums, colSums, rowMeans, colMeans, rbind, cbind
  - This means you need to worry about the [ ] (subset) vs [[ ]] (component) difference
    - Notice the difference between mtcars[1] and mtcars[[1]]
    - You can also extract components with $ and variable names, which is essentially the same as [[ ]]

Functions

- You've been using functions already, although we haven't referred to them that way.

- rowSums(parameter)
  - This is an example of the function called "rowSums"
  - If you just type *rowSums* into R, it will give you the code for that function
  - If you type *rowSums()* into R, it will execute that function without any parameters
  - If you type *rowSums(my_df)* into R, it will execute that function passing the variable *my_df* as a parameter to it

- Parameters are separated by commands and are named. If you omit names, they will be processed in the order specified by the person who wrote the function. For example:
  - newSums <- rowSums(my_df, na.rm=TRUE)

Comments

- You saw lots of comments but probably never actually wrote one.

- Just use # before whatever you want to comment
- You can use # in the middle of a line, e.g.,
  - summary() # this creates a summary table

- If you need to comment multiple lines at once in R Studio, you can use Ctrl+Shift+C

New Skills for This Module's Project

- R Studio
  - Create Project
  - Create Subdirectories (see https://nicercode.github.io/blog/2013-04-05-projects/)
    - R, data, output, docs, figures
  - Put Files in Correct Places
  - Refer to Files by Relative Paths
  - Create Archive