Getting Started with R

Beomjo Park (beomjop@andrew.cmu.edu)

June 1, 2020

Install R and Rstudio

To install `R` (latest release: 4.0.0), please go to https://www.r-project.org and choose your system. Click the *download R* link in the middle of the page under "Getting Started." Download and install the installer files (executable, pkg, etc) that correspond to your system.

Although you can use `R` without any integrated development environment (IDE), you will need to install **RStudio**, by far the most popular IDE for `R`, for CMSACamp. Basically, it makes your life with R much easier and we will be using it throughout the program. To install RStudio, please go to https://www.rstudio.com/products/rstudio/download/#download and choose your system. The installer is preferred. If you have RStudio installed but not the latest version, just download the latest installer and install.

Intro to R Markdown

An R Markdown(`.Rmd`) file is a dynamic document combining the `R` and the markdown. It contains the reproducible `R` code along with the narration that a reader needs to understand your work. (This file itself is generated by R Markdown.) If you are familiar with the laTeX syntax, math mode works like a charm in almost the same way:

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

The chunks of embedded `R` codes are:

```
#R codes here
print("Hello World")
```

```
## [1] "Hello World"
```

Running a Rmarkdown file and converting it into a reader-friendly documents require the `rmarkdown` and `knitr` package. All the lab documents will be Rmarkdown files and you need to know how to knit them. We recommend to knit as `html` file but if you have LaTex installed, you can knit as PDF.

For more detailed information, **R Markdown Cheatsheet** and the **reference guide** are helpful. (For RStudio users, easily accessible from Help -> Cheatsheets)

Install R packages

`R` performs a wide variety of functions, such as data manipulation, modeling, and visualization. The extensive code base beyond the built-in functions are managed by **packages** created from numerous statisticians and developers. The Comprehensive R Archive Network (CRAN) manages the open-source distribution and the quality control of the `R` packages.

To install a `R` package, using the function `install.packages` and put the package name in the parentheses and the quote. While this is preferred, for those using RStudio, you can also go to "Tools" then "Install Packages" and then input the package name.

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

If in any time you get a message says: "Do you want to install from sources the package which needs compilation?" Choose "No" will tend to bring less troubles. (Note: This happens when the bleeding-edge version package is available, but not yet compiled for each OS distribution. In many case, you can just proceed without the source compilation.)

Each package only need to be installed once. Whenever you want to use functions defined in the package, you need to *load* the package with the statement:

```
library(tidyverse)
```

Here is a list of packages that we may need (but not limited to) in the following lectures and/or labs. Please make sure you can install all of them. If you fail to install any package, please update the `R` and RStudio first and check the error message for any other packages that need to install first.

```
library(tidyverse)
library(devtools)
library(rmarkdown)
library(knitr)
library(stringr)
library(car)
library(randomForest)
library(glmnet)
```

Basic data type and operators

Data type: Vector

The basic unit of `R` is a vector. A vector `v` is a collection of values of the same type and the type could be:

1. numeric (double/integer number): digits with optional decimal point

```
v1 <- c(1, 5, 8.3, 0.02, 99999)
typeof(v1)
```

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

2. character: a string (or word) in double or single quotes, "..." or '...'.

```
v2 <- c("apple", "banana", "3 chairs", "dimension1", ">-<")
typeof(v2)
```

```
## [1] "character"
```

3. logical: TRUE and FALSE

```
v3 <- c(TRUE, FALSE, FALSE)
typeof(v3)
```

```
## [1] "logical"
```

Note: Oftentimes, `factor` is used to encode a character vector into unique numeric vector.

```
player_type = c("Batter", "Batter", "Hitter", "Batter", "Hitter")
player_type = factor(player_type)
str(player_type)
```

```
##  Factor w/ 2 levels "Batter","Hitter": 1 1 2 1 2
```

```
typeof(player_type)
```

```
## [1] "integer"
```

Data type: Lists

Vector can store only single data type:

```
typeof(c(1, TRUE, "apple"))
```

```
## [1] "character"
```

List is a vector of vectors which can store different data types of vectors:

```
roster <- list(
  name = c("Ron", "Pratik", "Beomjo"),
  role = c("Instructor", "TA", "TA"),
  is_TA = c(FALSE, TRUE, TRUE)
)

str(roster)
```

```
## List of 3
##  $ name : chr [1:3] "Ron" "Pratik" "Beomjo"
##  $ role : chr [1:3] "Instructor" "TA" "TA"
##  $ is_TA: logi [1:3] FALSE TRUE TRUE
```

R uses a specific type of list, **data frame**, containing the same number of rows with unique row names.

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...
```

```
typeof(iris)
```

```
## [1] "list"
```

Operators

We can preform element-wise actions on vectors through the operators:

1. Arithmetic: + - * / ^ (and, for integer division, %/% is quotient, %% is remainder)

```
v1 <- c(1,2,3)
v2 <- c(4,5,6)

v1 + v2
```

```
## [1] 5 7 9
```

```
v1 * v2
```

```
## [1]  4 10 18
```

```
v2 %% v1
```

```
## [1] 0 1 0
```

2. Relation: > >= < <= == != (last two are equals and is not equal to)

```
5 > 4
```

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

```
5 <= 4
```

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

```
33 == 22
```

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

```
33 != 22
```

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

3. Logic: !(not) &(and) |(or)

```
(5 > 6) | (2 < 3)
```

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

```
(5 > 6) & (2 < 3)
```

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

```
!(5 > 6) & (2 < 3)
```

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

4. Sequence: from:to (Colon operator)

```
1:5
```

```
## [1] 1 2 3 4 5
```

```
5:1
```

```
## [1] 5 4 3 2 1
```

```
-1:-5
```

```
## [1] -1 -2 -3 -4 -5
```

```
-1:5
```

```
## [1] -1  0  1  2  3  4  5
```

Read the csv files

Most of the data provided to you are in csv format. Please download the csv files first from and put them into the same folder with the R files. Click "Session", then "Set Working Directory" and then "To Source File Location".

Both csv files we provide right now have column names(colnames) for each variable, thus we set `header = TRUE`. Otherwise, remember to set it as `FALSE`.

```
nba <- read.csv("http://www.stat.cmu.edu/cmsac/sure/materials/data/intro_r/nba_2020_player_stats.csv", h
head(nba)
```

Looking for help

If you have any R problem, the best way is to Google and look into the helps. If you have questions for some specific functions or packages, you can use the statement "?" + "function name", for example:

```
?help
```

Double question marks can lead to a more general search.

```
??help
```

Exercise

1. Create four vectors, `v1` and `v2` are numeric vectors, `v3` is a character vector and `v4` is a logic vector. Make sure the length of `v1` and `v2` are the same. (Hint: a way to check the length is to use the function `length()`)

```
#R code here
```

2. Preform add, minus, product and division on `v1` and `v2`.

```
#R code here
```

3. Create four statements with both relation and logic operators, that 2 of them return `TRUE` and 2 of them return `FALSE`.

```
#R code here
```

4. Create 2 sequences with length 20, one in an increasing order and the other in a decreasing order.

```
#R code here
```

5. Following is the `Batting` dataset containing historical MLB statistics from the `Lahman` package. How many of the players are in Double-A league (coded as 'AA' in `lgID`)? How about American League('AL')? Can you neatly summarize the counts for all leagues? (HINT: `table()`)

```
library(Lahman)
data(Batting)

#R code here
```