Chapter 1.5: Introduction to R

Learning Outcomes: For our first introduction to R, we are going to go over the following:

- See the difference between R and RStudio
- Writing commands in a script vs directly in the console.
- Opening an R Markdown file.
- Check and set the directory that we are working in.
- Adding, subtracting, multiplying, dividing and averaging values.
- Assigning values to a variable.
- Vectors.

Difference between R and RStudio

R is a programming language used for statistical computing while RStudio uses the R language to develop statistical programs. RStudio is an example of an integrated development environment or IDE.

You can use R without RStudio but you cannot use RStudio without R (since RStudio uses R).

The Advantage of RStudio: RStudio has a much more user-friendly interface which makes it nice especially for beginning programmers. Additionally, RStudio allows users to develop and edit programs in R by supporting a large number of statistical packages, higher quality graphics, the ability to manage your workspace, and conveniently has a way to present your code and output as a pdf, HTML, or Github document (just a few examples).

Writing Commands in the Console vs a Script:

You can type commands directly into the console, each command will start with a > symbol, followed by the command.

Once you press enter, the command will be executed. For example, type into the console 5+2, press enter, and then type 6-3, and press enter:

To open a script in RStudio, click on File $->$ New File $->$ R Script.

The script will open above the console. In a script you type the commands each on their own line. Notice that if you press enter in the script, the command is not executed.

For example, type into the script 5+2, press enter, and then type 6-3, and press enter:

If we want to execute the commands in the script, we need to **Run** the code. You can do this in a few different ways:

1. Click on the line in the script with the command that you wish to execute. Then click on $->$Run.

2. Highlight the line in the script the command that you wish to execute. Then click on $->$Run.

3. Click on the line in the script with the command that you wish to execute. Then (for Macs) press Command+Enter or (for PC) press Ctrl+Enter.

Notice: When you run the code from a script, the output is shown in the console, not in the script itself.

Question: Why not just type all of our commands into the console since typing it into a script involves an extra step to run the code?

Answer:

Using R Markdown

R Markdown is a feature of RStudio which allows you to combine your script and output all in the same document. To open up a new R Markdown click on File $->$ New File $->$ R Markdown.

In this course, when using R Markdown, we will only be creating documents. You can type in the title of your document and also select the format of your document (either HTML, PDF or Word).

Note 1: If you wish for your document output to be in PDF format, you will need to download something called LaTeX which is a free Math programming language (I use it to write all of your notes). You can download it at: https://www.latex-project.org/get/

Note 2: You do not need to download LaTeX if you want your document to be an HTML or Word file.

Advantage of R Markdown:

You will be learning how to use R Markdown in your first lab on Wednesday January 20th.

Checking and Setting your Directory:

If is a good idea to create a file dedicated to your Stat 123 R assignments/labs. For example, I've created a folder called R Stat123 which is where I will save all content related to assignments and labs in this course.

You will need to tell R what file you want it to look in. For example, if you are given a data set to download and you save it to your RStat123 folder. You will need to tell R to find that data file in that folder.

Set your Working Directory in R Studio: In order to set your directory using R Studio, click on Session $->$ Set Working Directory $->$ Choose Directory. Then select the folder that you've designated for R assignments.

If you want to double check that you've set your working directory correctly, you can type in the command:

```
getwd()
```

If you want to set your directory manually you can use the command:

setwd(" your directory ") or use RStudio menu.

Adding, Subtracting, Multiplying, Dividing and Averaging in R:

You already saw that we can directly perform arithmetic operations in R by using the keys:

+ , − , * , / ^ * exponentiation 2^5 ⇒ R 2^5

There are also R commands that can add or multiply many numbers:

e.g. 2+3+5+4+10

2*3*5*4*10

2*(3+4)

2.(3+4)

To find the average, we must add up all of the values and then divide by the number of values:

(1+2+3)/3

Question: Is there a better way to do this?

Answer: Yes, with vectors.

In R, a vector is a list of values (which could be numerical values, TRUE/FALSE statements, characters, etc..).

Before we can talk about vectors, we must first learn how to assign a value to a variable:

Assigning a value to a variable:

In R, we can use either $<-$ or $=$ to assign a value to a variable. For example:

After assigning a value to a variable, if we want to see the value of that variable (or print the value of that variable) we just type the variable name as the command and run that line of code.

Types of Variables: Here are the basic types of variables in R:

- character e.g. "yes" yes is not a character string.
  "t", "fish", "1"
- numeric 8, -21, -6.5324, 3/4
- integer -2, 0, 8, 10
  -2L, 0L, 8L, 10L ← forces R to interpret this as integer.
- logical TRUE (or T), FALSE (or F)

Vectors: To create a vector in R, we use the command c( ).

For example, the vector V containing the numbers (6, 12, -3, 2, 51) can be assigned using the command:

V ← c(6, 12, -3, 2, 51)
(←)

The vector C containing the characters (Apple, Orange, Grape, Other) can be assigned using the command:

C ← c("Apple", "Orange", "Grape", "Other")

If we want to assign a sequence of numbers (each separated by 1) then we can use a colon:

For example, the code c(4 : 9) = 4 5 6 7 8 9

c(4,5,6,7,8,9)

We can also write a sequence of numbers each separated by an amount k by using the command seq(a, b, k).

For example, the code seq(4, 9.0.5) = seq(4, 9, 0.5)

try seq(4, 8.7, 0.5)

Combining this together, to get a vector which contains a sequence of numbers from 6 to 20, each separated by 0.2, we would use the command:

```
seq(6, 20, 0.2)
```

Question: Why are vectors useful?

Answer: We can perform arithmetic operations using vectors which is much easier than performing those operations by individually typing in values .

Examples: Define a vector called *days* which contains the days of the week (starting from Sunday) and another vector called *classes* which contains the numbers $(0, 2, 4, 5, 4, 3, 0)$ which represent the number of hours of classes a student has each day of the week.

```
days <- c("Sunday", "Monday", ..., "Saturday")
```

1. We can add up the total number of classes in a week:

```
sum(classes)
```

```
classes <- c(0, 2, 4, 5, 4, 3, 0)
```

sum( ) → are function

position 1 ... position 7

2. We can determine the average number of hours of classes each day of the week:

```
mean(classes)
```

mean( )

3. If we want to only access some days of the week, we can specify which item of the vector *classes* we want to look at. For example, we can look at only the hours in the 3rd day of the week:

```
classes[3]
```

R index starts at 1.

4. We can look at only the hours from the weekdays (day 2 to day 6 of the week):

```
classes[2:6]
```

5. We could assign a variable called *weekdays* to the vector which contains only the hours of classes during the week:

```
weekdays <- classes[2:6]
```

6. We could then take the average number of hours of classes each weekday:

```
mean(weekdays)
```

7. We can assign the *days* vector to be the **names** of the *classes* vector elements:

```
names(classes) = days
```

8. So if we want to access Thursdays class hours we have two options:

```
classes[5]
classes["Thursday"]   <- just a number
```

DataCamp Assignment: The datacamp course: Introduction to R have been assigned in datacamp as an assignment. Students in this course should have received an email from datacamp inviting them to sign up for free. Once you sign in to your account, check the due date for various assignments.