Lab 1: Intro to R

January 21-22, 2020

During the first lab we will introduce the programming language R, which will be used throughout this course. It will be helpful for you to be familiar with the basics of programming. R can be used to do simple calculations, create plots and figures,and to run statistical analyses. In lab, we will provide you with all the programming you need to know for this class.

Downloading and Installing R

We will be using a version of R called RStudio. On the lab computers, which we be using in discussion sessions, you can just open up RStudio. However, to get it on your personal computer it is a two-step process.

1. To install R on your personal computer, go to http://cran.r-project.org
2. Then install RStudio on your personal computer, go to http://www.rstudio.com

Note: You have to install R first before installing RStudio.

The instructions are pretty clear on the website, but if you need help please feel free to ask a TA to assist you during office hours.

Creating a New H Drive Folder & Starting an R Script

It is important that we all have the same set-up for folders/files in our H: Drive, so please follow these steps exactly:

Open your Documents Menu. This can be done by clicking on the windows icon in the bottom left of your screen, and then clicking on the icon of the page labelled "Documents" on the left of that menu.

Click on your H: Drive, which can be found on the lower part of the menu on the left, labeled "(H:) HawkID".

Create a new folder by right-clicking in the blank space on this page, selecting "New", and then selecting "Folder". Title this Folder "BIOS4120Labs" (without the quotations).

Interface: The Layout of RStudio

Look for RStudio in the start menu, and go ahead and open it up.

The first thing you'll want to do is go to File -> New File -> RScript.

This will open a window on the top left of your screen in RStudio where you'll be doing all of your work.

Save this R Script into your newly created folder by doing File -> Save As... -> (H:) -> Bios4120Labs, and call it "Lab-01".

You'll now have four windows open in RStudio:
1. Script (top left)
2. Console (bottom left)
3. Variables (top right)
4. Graphs/Help (bottom right)

Note: To run code, type it in the script, then highlight it and hit Ctrl-Enter to send it to the console to run.

Adding Comments

Often in programming languages, you can provide comments within code that explains what the code does and allows you to leave notes for yourself. In R, to start a comment a # is used and everything on the same line immediately to the right of the # is commented out.

```r
# Example of a comment. This line does NOT get run by the program.
```

In the R Script, comments are green. Remember, the # starts a comment ONLY on the same line.

Basics: R can act as a calculator.

```r
4 + 6 - (24/6)
```

```
## [1] 6
```

```r
(6 - 4) * 3
```

```
## [1] 6
```

```r
5 ^ 2
```

```
## [1] 25
```

Functions:

```r
exp(2) # This is the number e raised to the power within the parentheses
```

```
## [1] 7.389056
```

```r
sqrt(4) # Square root
```

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

```r
abs(-5) # Absolute value
```

```
## [1] 5
```

Sequences

Creating a sequence:

```r
1:5 # Creates a sequence from 1 to 5
```

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

```r
seq(from=1,to=5,by=1) # Constructs the same sequence as above
```

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

Sequences and Computation:

```r
1:5 + 5
```

```
## [1]  6  7  8  9 10
```

```r
1:5 * 2
```

```
## [1]  2  4  6  8 10
```

Storing/Assigning Variables

You can name numeric or character variables any way you want. Spaces are **NOT** allowed in naming variables. It is recommended to only use letters (uppercase or lowercase) and underscores (_) when naming a variable.

Note: R is case-sensitive, which means capital letters are different from lower case letters. Therefore, X would be different from x.

Also note that the assignment operater (<-) is preferred to the equals sign. This is standard R programming practice.

```r
x <- 5 # This assigns the value 5 to the variable x.
# Now we can reference x, and R substitutes in 5.
x
```

```
## [1] 5
```

```r
# This is useful for things like
y <- log(5) + 3/2
y
```

```
## [1] 3.109438
```

You can store sequences as variables too. These types of variables are called *vectors*.

```r
x <- c(2,3,5) # This is an example of storing a vector.
x
```

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

Indexing

Let's say we have a vector, x, as defined above, and we want to extract the second element of vector x and store it in a new variable called b.

```r
b <- x[2]
b
```

```
## [1] 3
```

The square brackets after a vector variable indicate we want to extract a certain position or positions of x. In this case, we extract the second position in x, and store it in a new variable called b.

Reading in Data

All of the datasets for this class will be on the class website, and can be read in using the URL:

```r
lab1.data<-read.delim("http://myweb.uiowa.edu/pbreheny/data/tips.txt")
```

Some basic things you can do with datasets:
(This will be expanded upon throughout the semester.)

```r
head(lab1.data) # Outputs only the first few lines of a dataset
```

```
##   TotBill  Tip Sex Smoker Day  Time Size
## 1   18.29 3.76   M    Yes Sat Night    4
## 2   16.99 1.01   F     No Sun Night    2
## 3   10.34 1.66   M     No Sun Night    3
## 4   21.01 3.50   M     No Sun Night    3
## 5   23.68 3.31   M     No Sun Night    2
## 6   24.59 3.61   F     No Sun Night    4
```

```r
summary(lab1.data) # Gives summary statistics for the dataset
```

```
##     TotBill            Tip          Sex     Smoker     Day        Time    
##  Min.   : 3.07   Min.   : 1.000   F: 87   No :151   Fri:19   Day  : 68  
##  1st Qu.:13.35   1st Qu.: 2.000   M:157   Yes: 93   Sat:87   Night:176  
##  Median :17.80   Median : 2.900                     Sun:76              
##  Mean   :19.79   Mean   : 2.998                     Thu:62              
##  3rd Qu.:24.13   3rd Qu.: 3.562                                         
##  Max.   :50.81   Max.   :10.000                                         
##       Size     
##  Min.   :1.00  
##  1st Qu.:2.00  
##  Median :2.00  
##  Mean   :2.57  
##  3rd Qu.:3.00  
##  Max.   :6.00
```

Let's say that we only want to look at one variable within the dataset. (For this example, we will choose to only look at the tip amounts.) We are able to reference a single variable by implementing a dollar sign and using the form dataset$variable.

```r
tips <- lab1.data$Tip
head(tips)
```

```
## [1] 3.76 1.01 1.66 3.50 3.31 3.61
```

Now we are able to analyze a specific variable from the dataset which will be very useful in future labs. We can use a variety of functions on this new variable as shown below.

```r
max(tips) # Largest Tip
```

```
## [1] 10
```

Help

To access the help documentation on a function you're not sure about, type a question mark before the function. For example, try typing ?seq.

Practice questions (Not graded)

Problem 1

Part A:

Create a sequence from 25 to 300 in increments of 25.

Part B:

Set Part A to variable named partB.

Part C:

Divide the sequence by 25 using the variable created in Part B.

Part D:

Using the partB variable, take the square root of the sequence.

What you should get upon running your code:

```
Part a
 25 50 75 100 125 150 175 200 225 250 275 300
Part b
 Stores internally, doesn't print
Part c
 1 2 3 4 5 6 7 8 9 10 11 12
Part d
 5 7.071068 8.660254 10 11.18034 12.24745 13.22876 14.14214 15 15.81139 16.58312 17.32051
```

Problem 4

Use the "c" function to create a vector called "grades", containing the following numbers representing student grades on an exam: 78.3, 90.1, 99.3, 82.1, 68.0

Problem 5

Look up the help documentation for the "sort" function in R. Use to organize the "grades" vector from highest to lowest. Name this new vector "grades_2"

Hint: use "decreasing = TRUE" in the sort function.

Problem 6

Find the 3rd highest score on the exam.

Hint: Use the [ ] brackets

Problem 7

Find the highest score on the exam using the "max" function.