Introduction to R

homework problems

Jason Thomas

November 6th, 2018

R

1. What is CRAN?
2. What are the different data structures in `R`? Describe a useful way to classify these data structures (along 2 dimensions).
3. What does missing data look like in `R` (i.e., how do you know if a certain observation is missing)?
4. Describe the functions `dir.create()` and `file.exists()` – what do they do, what arguments do they take, and do they return anything?
5. What is going on below, why does R return an error for the last line?

```
> n <- 3
> n
> n <- n + n
> n
> 3 + N
```

6. Why does the following code give a Warning message?

```
> M1 <- matrix(1:100, nrow=10, byrow=TRUE)
> y <- 1:3
> cbind(M1, y)
```

7. *Conditional* statements in R only run *if* a certain condition is true. Here is an example

```
> x <- 4; y <- 3
> if (x > y) {
+     print("x is bigger than y")
+     print("y is smaller than x")
+ }
```

```
## [1] "x is bigger than y"
## [1] "y is smaller than x"
```

THE + SIGNS THAT APPEAR IN THE ABOVE BLOCK OF CODE INDICATE THAT R CONSIDERS THE ENTIRE BLOCK AS A SINGLE COMMAND. THE + IS NOT SOMETHING YOU SHOULD TYPE AT THE PROMPT*.

Will the following block of code change values equal to -9 into R's way of indicating missing values? Why not?

```
> age <- c(19, -9)
>
> if (age[1] == -9) {
+
+     age[1] <- NA
+
+     if (age[2] == -9) {
+
+         age[2] <- NA
+     }
+ }
```

8. *For Loops* repeatedly execute the same block of code that appears between curly braces. To specify the number of times that R will evalute (the same) code, we choose a variable and then give a vector of values that the variable will take on

```
> for (i in 1:3) { # i is a variable that will take values 1, 2, 3
+   print(i)
+ }
```

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

First, note that `1:3` is a vector of 3 values: 1, 2, & 3. R will set `i = 1`, then evaluate the code: print(1). Then, R will set `i = 2`, then evaluate all of the code between the curly braces: print(2); etc. Fix the following for loop so that it recodes the value -9 into `NA`.

```
age <- c(28, 88, 2, -9, 73, 14, -9, 24, -9)
for (i in 1:7) {

    if (age[i] = -9) {

        age[i] <- NA
}
age
```

Vectors

1. How many elements are in the following vector?

```
> x <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
```

   (please make R do the counting)

2. The `rep()` function can be useful. Use it to create (1) a vector that contains five of the letter "a"; (2) a vector that looks like (1, 1, 1, 2, 2, 2, 3, 3, 3); and (3) a vector that looks like (1, 2, 2, 3, 3, 3).

3. Create the following vector: `z <- c(1, 4, NA, 3, 9)`. Now, have R calculate the mean (please ignore missing values so that your answer is 4.25).

4. Create a `factor` in R. For example, a factor called education with 151 observations/individuals, where 55 are "low education", 28 are "medium education", and 68 are "high education". But come up with your own example (doesn't have to be 151 observations, although that would be good practice).

5. Give an example of an *ordered* `factor`.

Matrices and Data Frames

1. Create a 5 by 8 matrix (i.e., 5 rows and 8 columns) that is full of random numbers. Have R provide a numerical `summary` of each column.

2. From the matrix in the previous problem, make R tell you what the following elements are (a) 4th row and 6th column; (b) 3rd row and 8th column; (c) all of the elements in the last row; (d) all of the elements in the 2nd column.

3. With the same matrix in the previous problem, explain what the following code is doing:

```
> X == X
> table( X == X )
```

4. What is the difference between a matrix and a data frame?

5. If I use the command `read.csv()` to read data into R, will what type of object will it create?

Functions

1. Write an R function that takes a matrix as its argument, and returns a new matrix where each element is divided by the row total. For example:

```
> inputMat <- cbind(c(1, 1, 1),  ## input to your function
+                   c(5, 5, 0)
+                   )
>
> ## your function should create:
> outputMat <- cbind(c(.33, .33, .33),
+                    c(.50, .50, .00)
+                    )
> ## so y <- myFunction(inputMat)
> ## and y should be very similar to outputMat
```

2. Explain what is going on with the following code:

```
> twoThree <- function(x = 2) {
+     if (x != 2) {
+         return(3)
+     }
+     return(x)
+ }
> twoThree()
> twoThree(3)
> twoThree(4)
```

3. Use the following steps to create a function that simulates random causes of death.

   a. Create a vectors that contains 4 elements: "COPD", "Heart Disease", "Accident", and "other"

   b. Create another vector that contains the probability of dying from each cause – the probabilities should sum to 1.

   c. Create a function that randomly chooses the cause of death according to the probabilities you created.

      - Users should be able to give your function an interger and your function will return that many random causes.
      - `sample()` might be a useful function to include

   d. Change your function to include 6 causes of death.