Week 2 Lab problems

EEB 429
Bhaskar Kumawat

HW Doubts

> ## 5. Data exploration
>
> *Anurag is interested in the species that show extreme defense trait values. Write a function that takes as its arguments: your milkweed dataframe, the name of the defense trait you want to investigate, and an arbitrary cutoff value (above which you consider trait values to be "extreme"). The function should return the names of the species with extreme values.*

A. Data-types

1. What is the data-type of the variables x, y, and z defined as below? (You don't need to run it in R)

```
x <- 42.0
y <- 42
z <- "42"
```

2. Define a variable "a" as below in a new R-script

```
a <- "9.999"
```

3. Convert the variable "a" to numeric, store it in a variable "b", and then display it.
4. Convert the variable "b" to integer, store it in a variable "c", and then display it.
5. Use the function `round()` on variable "b", store it in a variable "d", and then display it.
6. How does variable "d" differ from variable "c" (i.e., how does converting a numeric to integer directly differ from rounding it using a function)?

Write the answer to parts 1 and 6, and submit your inputs **and outputs** for parts 2-5 to earn full credit for this question.

B. Vectors and Matrices

1. Create a vector "v1" that is the number 4 repeated 7 times.
2. Create a vector "v2" that is a sequence from 1 to 20 with a step-size of 3.
3. Create a vector "v3" that consists of elements 8, 10, 11, 12, 13, 14, 6.
4. Create a matrix "A" with v1, v2, and v3 as its rows (and display it)
5. Create a matrix "B" with v1, v2, and v3 as its columns (and display it)
6. Add a new row with values 100, 200, and 300 to matrix "B" (and display it)
7. Print the dimensions of matrices A and B.

Submit your inputs and outputs for part 1-7 to earn full credit for this question.

C. Indexing (and plotting)

In a new R-script, do the following:

Vectors

1. Create a vector with the following elements: "L", "O", "W", "D", "E", "R", "G", "H", "I"
2. Index the vector (i.e., pick elements from it) to spell “HELLOWORLD” (in one line of code.)

Matrix

3. Create a 3x3 matrix from the above vector using the matrix() function
4. Once again, index the matrix to spell “HELLOWORLD” in one line of code.
   Hint: You can specify a “coordinate” in the matrix using c(x,y) and then bind all coordinates together using rbind()

Data Frames

5. “mtcars” is a pre-loaded data-frame in R. Make sure you can see it when you execute “mtcars”
6. Display the first column of this data-frame. (By number and not name)
7. Display the third row of this data-frame. (By number and not name)
8. Display the “gear” column of the data-frame (By name and not number)
9. Display the 3rd through 10th elements of the “cyl” column of the data-frame.
10. Use plot() command to plot the mpg of a car in the dataset against the weight (wt) of the car. (You can use ?plot to see how the function works)

Lists

11. Create a list containing the above three objects with labels “v”, “m”, and “d”.
12. Display the first object in the list.
13. Display the object labeled “d” in the list.
14. Display the letter “H” from the matrix inside the list. (You can either access the matrix by number or by label)

Submit the inputs and outputs for 1-14 to get full credit for this problem (you can just execute the entire R-script and copy-paste the results from the console).

(Optional extra reading on indexing: https://csu-r.github.io/Module1/indexing.html)

C. Creating data-frames

Consider the following data-table:

country	cases	population	year
Afghanistan	745	19987	1999
Afghanistan	2666	20595	2000
Brazil	37737	172006	1999
Brazil	80488	174504	2000

To recreate this table in R as a data-frame, we can use one of three methods:

A. We can input it into a .csv file and import it into R.
B. We can recreate each column as a vector in R, and then create a data-frame from them.
C. We can create an *empty* data-frame, and then add the data row-by-row.

In a new R-script file:

1. Use method A to recreate the data-frame and display it in R (you can use Excel or Google Sheets to input the data and save it as a .csv file).
2. Use method B to recreate the data-frame and display it in R. (you will have to use the function data.frame() for this)
3. Use method C to recreate the data-frame and display it in R. This is a bit more complicated and requires you to know the following commands.

   Creating an empty data-frame

   ```
   # This creates an empty data-frame called “testdata” with two columns named “name” and “birthyear”
   testdata <- data.frame(name=character(), birthyear=integer())
   ```

   Note how we need to specify the data-type stored in each column, R needs to know what you’re going to store in each column beforehand!

   Adding a new row to a data-frame

   ```
   # This adds the name “john” and birth-year “1956” to the data-frame “testdata”
   new_row <- data.frame(name=”john”, birthyear=1956)
   testdata <- rbind(testdata,new_row)
   ```

Submit your inputs and outputs for part 1-3 to earn full credit for this question.