**Introduction**

The goal of this assignment is to familiarize you with creating and manipulating objects in R. The assignment will also introduce you very handy base R functions that will make both of these tasks more efficient (less typing by hand!).

**Concepts covered in this assignment include:**

1. Creating objects in R
2. Indexing values in objects
3. Applying operators to objects

**Resources/Files needed for this assignment:**

* None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Run the R code given below and consider the vector and the two matrices:

a <- seq(2,12,by=2); a

A <- matrix(a,4,2); A

B <- matrix(a,3,4); B

* 1. What is the length of the vector? How does the length of the vector compare to the dimensions of the two matrices? How did R handle any discrepancies? Did it differ between the two matrices – if so, how?
  2. Using the information obtained in part (a) provide the R code to create the matrix given below using the following approaches:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | [,1] | [,2] | [,3] | [,4] | [,5] |
| [1,] | -2 | -1 | 0 | 1 | 2 |
| [2,] | -2 | -1 | 0 | 1 | 2 |
| [3,] | -2 | -1 | 0 | 1 | 2 |

* + 1. Using the c() and matrix() functions (i.e. the long way).
    2. Using the cbind() or the rbind() function.
    3. Using vectorized programming with the matrix() function (i.e. using information obtained in part (a)).

1. Run the R code given below and consider the two matrices:

A <- matrix(2,2,2); A

B <- matrix(1:4,2,2); B

* 1. Submit the following commands to R: A\*B and A%\*%B. Explain the difference between the two commands (i.e. \* vs. %\*%).
  2. Should the output of A\*B equal the output of B\*A? Explain.
  3. Should the output of A%\*%B equal the output of B%\*%A? Explain.

1. Consider the data set given below. It lists 5 students’ scores on three tests and on a final exam in a course:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test1 | Test2 | Test3 | Final |
| Student1 | 20 | 23 | 18 | 48 |
| Student2 | 16 | 15 | 18 | 36 |
| Student3 | 25 | 20 | 22 | 40 |
| Student4 | 14 | 19 | 18 | 42 |
| Student5 | 10 | 15 | 14 | 30 |

* 1. Create a data object that contains the same information as the data set shown above. Call the data object grades. Make sure that the mode of each data element is appropriate. Print the grades object.
  2. Suppose that Student 1 actually received a 21 on Test 3. Update the grades data object to reflect the correct score. Print the updated grades object.
  3. Print all the grades for …
     1. Test 3
     2. Student 4
     3. Test 1 – 3 but not for the final exam.
  4. Print the test scores for students who scored above 16 on Test 1.
  5. Find the mean score for each student across all tests (including the final).
  6. Find the range of scores for each test (including the final).
  7. Add a variable that denotes whether a student passed or failed the final. A fail is defined as a score below 40. Indicate a pass with a P and a fail with an F. Name the new variable PF\_Final. Print the updated grades object.

1. Run the R code given below and consider the vector and the two matrices:

a <- c(-2,4,-3,0); a

A <- matrix(c(2,5,3,0,9,-1,-4,-7,6),3,3); A

B <- matrix(c(2,5,-3,10,9,-1,9,-7,6),3,3); B

* 1. Apply the length() function to the vector and the first matrix. Explain what the function returns for each data object.
  2. Using the information obtained in part (a), determine if the two matrices are element-wise equal. If not, find the largest absolute element-wise difference between the two matrices.

1. The diag() function is very flexible and can be a useful tool with working the matrices. Run the R code below:

a <- 1:5; a

diag(a)

A <- matrix(c(-5,9,-14,1,3,-12,6,-6,13),3,3); A

diag(A)

* 1. Explain the difference between applying the diag() function to vector vs. applying the diag() function to a matrix.
  2. Using the information obtained in part (a) …
     1. Find the smallest on diagonal element of matrix A.
     2. Find the smallest off diagonal element of matrix A.

1. Consider the vector v given in the R code below:

> v <- 1:10; v

[1] 1 2 3 4 5 6 7 8 9 10

R is a vector-based or “vectorized” language. As such, vectors are very flexible data objects in R. We have seen an example of this in the lecture slides where vectors were re-shaped to create matrices and arrays. This works because R views vectors as “dimension-less” objects; instead vectors have length. To see an example of this, run the R code given below:

length(v)

dim(v)

* 1. The function t() finds the transpose of a matrix. What happens when this function is applied to the data object called v? Does it fail to execute? If so, why? Does it execute? If so, does it return a matrix? If so, why does this make sense? Find an R function that confirms whether the output of t(v) is actually viewed as a matrix by R.
     1. Hint: Google is your friend!
  2. What type of object does the R code t(v)%\*%v return – a scalar, vector, or matrix? (Test yourself – try to determine BEFORE running the code.) Confirm your answer. What is the name given to this object in linear algebra?
  3. What type of object does the R code v%\*%t(v) return – a scalar, vector, or matrix? (Test yourself – try to determine BEFORE running the code.) What is the name given to this object in linear algebra?
  4. Using vectors and matrix algebra, find the mean of the elements in the data object called v. Make sure your code is automated and would work properly for ANY vector v. Confirm your answer using a function available in R.
     1. Hint: You may have to create additional vectors to answer this question.
  5. Using vectors and matrix algebra, find the standard deviation of the elements in the data object called v. Make sure your code is automated and would work properly for ANY vector v. Confirm your answer using a function available in R.
     1. Hint: You may have to create additional vectors to answer this question.
  6. Using the code developed in part (d) and part (e), find the mean and standard deviation of the following vector: v <- seq(-6,6,by=2). Confirm your answer using functions available in R.

1. A student is tasked with simulating (generating data using R based on assumed probability distributions) the blood glucose levels of 15 patients with type 1 diabetes or type 2 diabetes. The following data was simulated: the glucose1 measurement is a fasting blood sugar measurement, the glucose2 measurement is a blood sugar measurement after a low carb lunch, and the glucose3 measurement is a blood sugar measurement after a high carb dinner. He uses the following code to generate the first test data set:

patientID <- 1:10

set.seed(15)

age <- runif(n=10,min=25,max=85)

type <- sample(x=c("Type1","Type2"),size=10,

replace=TRUE,prob=c(0.5,0.5))

glucose1 <- round(runif(n=10,min=50,max=120))

glucose2 <- round(runif(n=10,min=100,max=250))

glucose3 <- round(runif(n=10,min=180,max=450))

sugar<-cbind(patientID,type,glucose1,glucose2,glucose3); sugar

Note: The sample() function randomly draws from vector x, to generate a vector the length specified with the size input, with or without replacement, and assigns the probabilities of drawing each value of in x, according to the vector, prob. The runif() function is used to generate a vector of n randomly drawn numbers from a uniform distribution with the provided minimum and maximum. The default of the round() function, rounds an numeric object (scalar, vector, matrix, etc), to the nearest whole number, and the set.seed() function ensures that the random numbers generated are the same each time the code is executed. For more details, remember you can access the R help pages!

* 1. The code above produces the following output:

patientID type glucose1 glucose2 glucose3

[1,] "1" "Type2" "106" "175" "427"

[2,] "2" "Type1" "106" "139" "304"

[3,] "3" "Type1" "75" "174" "429"

[4,] "4" "Type1" "54" "118" "250"

[5,] "5" "Type1" "90" "177" "273"

[6,] "6" "Type1" "96" "199" "296"

[7,] "7" "Type2" "57" "118" "269"

[8,] "8" "Type1" "60" "177" "383"

[9,] "9" "Type2" "115" "145" "408"

[10,] "10" "Type1" "83" "214" "183"

Will the student be able to perform numerical analyses on the glucose measurements using the object called sugar? If so, simply state so. If not, explain why. Could you modify the R code given above so that numerical analyses could be performed? If so, provide the modified R code.

* 1. The student wants to compare the average glucose values across all fasting conditions (fasting, before meal, and after meal) between patients. Add a variable called gluc.mean to the data object sugar that contains this value. Round the answers to the nearest whole number.
     1. Perform this task using the rowMeans() function.
     2. Perform this task again using the apply() function.
  2. The student thinks it might be useful to sort the data by the value of gluc.mean from highest to lowest to examine which patients tend to have poor blood glucose regulation. Perform this task using one of the following R functions: the sort(), order(), or rank() function. Make sure to not overwrite the data object sugar when performing this task; instead, create a new data object called sugar2.
  3. Ideally, fasting blood sugars should be above 60 in diabetics. Falling substantially below this cutoff puts a patient at risk for hypoglycemic shock and many diabetics have trouble avoiding this overnight if they take medications in the evening. The student wants to create a new variable called hypo.risk that is zero if the patient is not at risk and 1 if the patient is at risk. Provide code that creates this variable and adds it to the data object called sugar. To verify your code print all patients with glucose1 less than 60 and glucose1 greater than 60 separately.