ALGONQUIN COLLEGE

Computer Engineering Technology – Computing Science

Course: Numerical Computing – CST8233
Term: Fall 2021

Lab #3

• Objectives

The main objective of this lab is to get familiar with the main programming components of R language. Namely, you will learn how to use variable names and how to manipulate vectors, matrices, and lists.

• Earning

This lab worth 1% of your final course mark. Each student should complete this lab and demo the codes of the exercises to the lab professor during the lab session.

• Steps

Step 1. Variables

In R, variables provide us with named storage that can be manipulated by our program. A valid variable name consists of letters, numbers, and the dot and the underscore characters. Variable names can only start with a letter or a dot NOT followed by a number.

- Which of the following can be used as a variable name?
  - Integer_1
  - Employee.1
  - Integer%
  - 1Integer
  - .Monthly_Salary
  - .4Compound_Rate
  - _Yearly_Rate

To assign a value to a variable, we can use one of the following methods:

- Leftward assignment
- Rightward assignment
- Equal to operator

The values of the variables can be printed using **print()** and **cat()** functions.

- ***Example:***

Enter the following code snippet and run each line and observe the output.

```
# Assignment using equal operator.
vect.1 = c(0,1,2,3)

# Assignment using leftward operator.
vect.2 <- c("Hello","World")

# Assignment using rightward operator.
c(TRUE,1) -> vect.3

print(vect.1)
cat ("vect.1 is ", vect.1 ,"\n")
cat ("vect.2 is ", vect.2 ,"\n")
cat ("vect.3 is ", vect.3 ,"\n")
```

In order to find the data type of a variable, **class()** function is used. Find the data type of each of the previous variables.

To know all the variables currently used in your program, you can use **ls()** function. Note that variables that start with a dot "." are hidden. To show these variables, you need to use "all.name = TRUE" option → ls(all.name = TRUE)

Variables can be deleted using remove function **rm()**.

Step 2. More on R Objects

Manipulation of matrices and vectors is one of the most common tasks you will undertake in R.

A. Manipulating Matrices

Create three matrices using three different methods as follows:

```
mat1 <- matrix(c(1,2,3,4,5,6,7,8,9), nrow = 3)
mat2 <- matrix(1:9, ncol = 3)
vect1 <- 1:9
mat3 <- matrix(vect1, nrow =3)
```

You can add, subtract, multiply (element-wise and algebra), and divide matrices as follows. Also, you can find the transpose of a matrix using **t()** function.

```
# adding two matrices
mat1 + mat2
# the product of linear algebra matrix multiplication
mat1 %*% mat2
# element by element multiplication
mat1 * mat2
# to find the transpose of a matrix (switch columns and rows)
t(mat1)
```

To access an element in a matrix, use the indices of the matrix. Also, you can remove any row, column of any matrix or change the value of an element as shown in the below code snippet.

```
# to access element in row 1 and column 3
mat1[1,3]
# to access all elements in row 2
mat1[2,]
# to access all elements in column 2
mat1[,2]
# to remove column 1
mat2[,-1]
# to remove row 1
mat3[-1,]
# to remove row 1 and column 2
mat1[-1,-2]
# to change a value of an element in a matrix
mat1[1,2] <- 15
```

- Change the values of all elements of the first row of mat1 to 15.
- Change the values of all elements of the first and second columns of mat2 to 6.

Demo your answer to the lab professor.

Manipulating data based on the value

In the previous examples, the value of the element is manipulated based on the indices. It is useful to manipulate the element based on the value.

```
mat1 <- matrix(c(1,2,3,4,5,6,7,8,9), nrow = 3)
# to find all elements that have values less than 6
mat1[mat1 < 6]
# to find all elements that have values greater than or equal 6
mat1[mat1 >= 6]
# to see the Boolean result of each element
mat1 > 4
```

B. Manipulating Vectors

Similar to the matrices, element of vectors can be accessed using the square brackets “[ ]” In addition to using numbers as indices, TRUE/FALSE and 0/1 can also be used. Input the following snippet in your RStudio and observe the output.

```
# Accessing vector elements using position.
t <- c("Sun","Mon","Tue","Wed","Thurs","Fri","Sat")
u <- t[c(2,3,6)]
print(u)

# Accessing vector elements using logical indexing.
v <- t[c(TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE)]
print(v)

# Accessing vector elements using negative indexing.
x <- t[c(-2,-5)]
print(x)

# Accessing vector elements using 0/1 indexing.
y <- t[c(0,0,0,0,0,0,1)]
print(y)
```

Two vectors can be added, subtracted, multiplied and divided by each other.

```r
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)

# Vector addition.
add.result <- v1+v2
print(add.result)

# Vector subtraction.
sub.result <- v1-v2
print(sub.result)

# Vector multiplication.
multi.result <- v1*v2
print(multi.result)

# Vector division.
divi.result <- v1/v2
print(divi.result)
```

In order to sort the elements of a vector, **sort()** function is used as follows:

```r
v <- c(13,8,41,2,0,11,-9)

# Sort the elements of the vector.
sort.result <- sort(v)
print(sort.result)

# Sort the elements in the reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result)

# Sorting character vectors.
v <- c("green","Blue","Yellow","violet", "Green")
sort.result <- sort(v)
print(sort.result)

# Sorting character vectors in reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result)
```

C. Manipulating Lists

As shown in Lab #2, lists can contain elements of different types, such as numbers, strings, vectors, and even another list or a matrix. The elements of the list can be named, use **names()** function, so they can be accessed by their names. The operator "**$**" is used to specify the name of the element of the list. Enter the following snippet and run it. Observe the output.

```r
# Create a list containing a vector, a matrix and a list.
listA <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
                list("green",12.3))

# Give names to the elements in the list.
names(listA) <- c("1st Quarter", "A_Matrix", "An_Inner_list")

# Show the list.
print(listA)
# Access the first element of the list.
print(listA[1])
# Access the third element. As it is also a list, all its
# elements will be printed.
print(listA[3])
# Access the list element using the name of the element.
print(listA$An_Inner_list)
```

Step 3. Exercises

A. Write a program using R language to vector vec1 from 0.1 to 1 with 0.1 increment. Find the length of this vector. Then, create a 5x2 matrix called mat1 using vec1. Name each row and column as row1, col1, etc. Then, change the value of the element located in the 3rd row and 2nd column to 10. Finally, print both the 3rd row and the 2nd column.

Hints:

- Use **seq()** function to generate a sequence of numbers with certain increment.
- Search in R Documentation for **matrix()** function to see how to give names to rows and columns. Basically, create two vectors and initialize them with the names and pass these two vectors as a list to **matrix()** function as the value of "dimnames".

Usage

```
Matrix(data=NA, nrow=1, ncol=1, byrow=FALSE, dimnames=NULL,
       sparse = NULL, doDiag = TRUE, forceCheck = FALSE)
```

B. Write a program using R language to create the following data frame (refer to Lab #2) and call it examDF:

	name	score	attempts	qualify
1	Anastasia	12.5	1	yes
2	Dima	9.0	3	no
3	Katherine	16.5	2	yes
4	James	12.0	3	no
5	Emily	9.0	2	no
6	Michael	20.0	3	yes
7	Matthew	14.5	1	yes
8	Laura	13.5	1	no
9	Kevin	8.0	2	no
10	Jonas	19.0	1	yes

- Create another data frame from examDF by dropping the first two columns and call it examRes.
- Create a second data frame from examDF by dropping the following rows: 2, 4, 5, 8, and 9 and call the new data frame as examQual.

Hints:

- Use **subset()** function to return a subset from the data frame. Search R Documentation to see how this function can be used to select vectors which meet certain conditions.

C. Write a program using R language to convert temperatures measurements from Fahrenheit F to Celsius C using the following equation: C = 5/9 (F-32). Test your program using the following vector: (45, 77, 19, 101, 120)

<u>You need to demo this to your lab professor.</u>

> **"Programming isn't about what you know; it's about what you can figure out."** - *Chris Pine*