ALGONQUIN COLLEGE

Computer Engineering Technology – Computing Science

Course: Numerical Computing – CST8233
Term: Fall 2021

Lab #2

1. 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 enter input, the data types, the arithmetic and logical operators, creating and manipulating matrices, and some basic data frames.

2. 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.

3. Steps

Step 1. Entering Input

Expressions are typed at the R prompt. R uses the "<-" symbol as the assignment operator.

```
> x <- 1
> print(x)
[1] 1
> x
[1] 1
> msg <- "hello"
```

After a complete expression is entered, it is evaluated and the result of the evaluated expression is returned.

```
> x <- 5  ## nothing printed
> x       ## auto-printing occurs
[1] 5
> print(x)  ## explicit printing
[1] 5
```

Note: "[1] shown in the output indicates that x is a vector and 5 is its first element."

- Try the following expression.

```
> x <- 10:30
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20 21
[13] 22 23 24 25 26 27 28 29 30
```

1. What does [13] indicate?
2. What does ":" operator do?

Step 2. R Classes of Objects

There are six main classes of objects in R language: character, numeric, integer, complex, logical, and raw.

- Character: 'a', 'speed', 'TRUE', '23.5'.
  - fac1 <- 'speed' OR Fac1 <- "Speed""
- Numeric: 12.5, 909.
  - y <- 12.5
- Integer: 2L, 5L 0L.
  - int1 <- 8L
- Complex: 4+2i.
  - com1 <- 6-6i
- Logical: TRUE, FALSE.
  - x <- TRUE
- Raw: "language" is stored as 6c 61 6e 67 75 61 67 65.
  - var <- charToRaw("language")

Try all the previous examples. In order to check the class of each variable, use the **class()** function. For example, class(y).

Step 3. R Objects

R language has many objects, the most widely used are: vectors, lists, matrices, arrays, factors, and data frames.

a. Vectors

To create a vector with more than one element, c() function is used.

```
> x <- c(0.5, 0.6)       ## numeric
> x <- c(TRUE, FALSE)    ## logical
> x <- c(T, F)           ## logical
> x <- c("a", "b", "c")  ## character
> x <- 9:29              ## integer
> x <- c(1+0i, 2+4i)     ## complex
```

In order to initialize vectors, the **vector()** function is used.

```
> x <- vector("numeric", length = 10)
> x
 [1] 0 0 0 0 0 0 0 0 0 0
```

In certain occasions, different classes of R objects get mixed in the same vector.

```
> y <- c(1.7, "a")   ## character
> y <- c(TRUE, 2)    ## numeric
> y <- c("a", TRUE)  ## character
```

- Check the class of each created vector.

These examples show the effect of *implicit coercion*. In other occasions, objects need to be explicitly coerced from one class to another. For this, "as.*" functions are used.

```
> x <- 0:6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

b. *Lists*

Lists are a special type of vector that can contain elements of different classes. Lists are very important data type in R. Lists can be explicitly created using the **list()** function which takes an arbitrary number of arguments.

```
> x <- list(1, "a", TRUE, 1 + 4i)
> x
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```

We can also create an empty list of a prespecified length with the **vector()** function.

```
> x <- vector("list", length = 4)
> x
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL
```

c. *Matrices*

Matrices are vectors with a dimension attribute. The dimension attribute is itself an integer vector of length 2, i.e., the number of rows and number of columns. The **matrix()** function is used to create a matrix.

```
> m <- matrix(nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(m)
[1] 2 3
> attributes(m)
$dim
[1] 2 3
```

Also, a matrix can be created using a vector input to the matrix function.

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

- Observe how the matrix is constructed column-wise starting from the upper left corner and running down the column.
- Instead of using the ":" operator, use **c()** function to create a vector of characters and pass it to the matrix function.

Matrices can also be created from vectors by adding a dimension attribute.

```
> m <- 1:10
> m
 [1]  1  2  3  4  5  6  7  8  9 10
> dim(m) <- c(2, 5)
> m
```

✓ What is the output of the previous commands?

Finally, matrices can be created by column-binding or row-binding using **cbind()** and **rbind()** functions.

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

d. *Arrays*

Arrays, as opposed to matrices that are restricted to two dimensions, can be of any number of dimensions. The array function takes a dim attribute as shown in the below example:

```
> a <- array(c('red','yellow'), dim = c(3,3,2))
> print(a)
, , 1

     [,1]     [,2]     [,3]
[1,] "red"    "yellow" "red"
[2,] "yellow" "red"    "yellow"
[3,] "red"    "yellow" "red"

, , 2

     [,1]     [,2]     [,3]
[1,] "yellow" "red"    "yellow"
[2,] "red"    "yellow" "red"
[3,] "yellow" "red"    "yellow"
```

e. *Data frames*

Data frames are used to store tabular data in R. They are a special type of list where every element of the list has to have the same length. Each element of the list can be thought of as a column and the length of each element od the list is the number of rows.

Notes:

- ✓ Unlike matrices, data frames can store different classes of objects in each column.
- ✓ Data frames have column names which indicate the names of the variables.
- ✓ Data frames have a special attribute called **row.names** which indicate information about each row of the data frame.
- ✓ Data frames can be explicitly created using **data.frame()** function.
- ✓ Data frames are usually created by reading in a dataset using **read.table()** or **read.csv()** functions.

```
> BMI <- data.frame(
+     gender = c("Male", "Male","Female"),
+     height = c(152, 171.5, 165),
+     weight = c(81,93, 78),
+     Age = c(42,38,26)
+ )
> print(BMI)
  gender height weight Age
1   Male  152.0     81  42
2   Male  171.5     93  38
3 Female  165.0     78  26
>
```

Another example:

```
> x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
> x
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Step 4. Arithmetic & Logical Operators

All the arithmetic operators in R are vectorized. The following examples demonstrate subtraction, multiplication, exponentiation, and two kinds of division, as well as remainder after division:

```
c(2, 3, 5, 7, 11, 13) - 2            #subtraction
## [1]  0  1  3  5  9 11
-2:2 * -2:2                          #multiplication
## [1] 4 1 0 1 4
identical(2 ^ 3, 2 ** 3)             #we can use ^ or ** for exponentiation
                                     #though ^ is more common
## [1] TRUE
1:10 / 3                             #floating point division
##  [1] 0.3333 0.6667 1.0000 1.3333 1.6667 2.0000 2.3333 2.6667 3.0000 3.3333
1:10 %/% 3                           #integer division
##  [1] 0 0 1 1 1 2 2 2 3 3
1:10 %% 3                            #remainder after division
##  [1] 1 2 0 1 2 0 1 2 0 1
```

There are three vectorized logical operators in R: “!” is used for not, “&” is used for and, “|” is used for or.

```
> x <- 1:10
> x >= 5
 [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> !x
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> y <- 1:10
> z <- y %% 2
> z == 0
 [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
> x & y
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> x | y
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
>
```

Step 5. Exercises

1. Write a code using R language to take input from the user, name and age, and display the following messages.

   ```
   Input your name: Mike Blacksmith
   Input your age: 18
   My name is Mike Blacksmith and I will be 19 years old next year.
   ```

   Hints: you need to use the **readline()** and **paste()** functions.

2. Write a code using R language to create three vectors v1, v2, and v3 and initialize them with the following integers: v1: 1, 3, 5, v2: 7, 9, 11, and v3: 13, 15, 17. Create a matrix using these three vectors where each column represents one of these vectors. Finally, print this matrix.

3. Write a code using R language to create a data frame which contains details of four students and display them as shown below.

```
> print(Students)
       Name Gender Age Designation NoCourses
1  Michael A      M  18 CET Studnet         5
2 Jennifer R      F  19  CP Student         4
3     Sara B      F  20 SSN Student      <NA>
4    James H      M  22  CS Student         3
>
```

You need to demo this to your lab professor.

> **"In some ways, programming is like painting. You start with a blank canvas and certain basic raw materials. You use a combination of science, art, and craft to determine what to do with them."** - *Andrew Hunt*