ALGONQUIN COLLEGE

Computer Engineering Technology – Computing Science

Course: Numerical Computing – CST8233
Term: Fall 2021

Lab #4

• Objectives

The main objective of this lab is to get familiar with creating and using functions in R. Also, this lab will introduce decision making structures and loops.

• 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. Functions in R

A. Built-In Math Functions

Similar to other programming languages, R provides various mathematical functions to perform mathematical calculations. Examples:

```
x <- -4.5
y <- c(1.2, 2.5,8.1)

# to return the absolute value of an input
print(abs(x))

# to returns the square root of an input
print(sqrt(abs(x)))

# to return the smallest integer which is larger than or equal to an input
print(ceiling(x))

# to return the largest integer, which is smaller than or equal to an input
print(floor(x))

# to return the truncate value of an input
print(trunc(y))
```

```r
# to return round value of an input
z <- 3.142857
print(round(z, 2))

# to return cos(x), sin(x) value of an input
print(sin(z))
print(cos(z))
print(tan(z))

# to returns natural logarithm of an input
print(log(abs(x)))

# to returns common logarithm of an input
print(log10(abs(x)))

# to return the exponent of an input
print(exp(x))
```

B. Built-In String Functions

In addition to math functions, R provides various string functions to perform tasks such as extracting sub-strings, searching patterns, etc.

```r
# to extract sub-strings in a character vector
a <- "Hello World!"
substr(a, 3, 4)

# to search for pattern in an input
st1 <- c('abcd','bdcd','abcdabcd')
pattern<- '^abc'
print(grep(pattern, st1))

# to find pattern in an input and replaces it with replacement (new) text
st1<- "England is beautiful but no the part of EU"
sub("England", "UK", st1)

# to concatenate strings after using "sep" string to separate them
paste('one',2,'three',4,'five')

# to split the elements of character an input vector at split point
a<-"Split all the character"
print(strsplit(a, ""))

# to convert the string into lower case
st1<- "HelLo WorLd"
print(tolower(st1))

# to convert the string into upper case
st1<- "HelLo WorLd"
print(toupper(st1))
```

C. User-Defined Functions

One of the strengths of R is that one can make user-defined functions that add to the already built-in functions. Usually, such functions are written in R script files and saved in the same working directory and then used in any other R script. The general structure of user-defined functions is as follows:

```r
function.name <- function(arguments)
{
  # computations on the arguments
  # some other code
  # use "return" if the function return a value

  return(variable)
}
```

Example: write a function that calculates the surface area of a circle given its radius.

```
Circlesurface <- function (radius)
  return(pi*radius^2)


# calling the function
a <- Circlesurface(8)
```

Step 2. Decision-Making and Loops in R

Programmers use decision making structures and loops in most of their codes. To use a decision-making structure, the programmer needs to specify one or more conditions to be tested along with statement(s) to be executed if these conditions are TRUE. Optionally, other statements can be executed if the conditions are FALSE. On the other and, loops are usually used to execute a block of code several number of times.

A. Decision-Making Structures

There are three decision-making statements in R: if statement, if ... else statement, and switch statement. The syntax of each statement is as follows:

```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true.
}
```

```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true.
} else {
   // statement(s) will execute if the boolean expression is false.
}
```

```
switch(expression, case1, case2, case3....)
```

Examples:

```
# example of if statement
x <- 30L
if(is.integer(x)) {
  print("X is an Integer")
}

# example of if ... else statement
x <- c("what","is","truth")

if("Truth" %in% x) {
  print("Truth is found")
} else {
  print("Truth is not found")
}

# example of switch statement
x <- switch(
  3,
  "first",
  "second",
  "third",
  "fourth"
)
print(x)
```

B. Loops Structures

There are three loop statements in R: repeat, while, and for statement. The syntax of each statement is as follows:

```
repeat {
   commands
   if(condition) {
      break
   }
}
```

```
while (test_expression) {
   statement
}
```

```
for (value in vector) {
   statements
}
```

Examples:

```
# example of repeat loop
v <- c("Hello","loop")
cnt <- 2

repeat {
  print(v)
  cnt <- cnt+1

  if(cnt > 5) {
    break
  }
}

# example of while loop
v <- c("Hello","while loop")
cnt <- 2

while (cnt < 7) {
  print(v)
  cnt = cnt + 1
}

# example of for loop
v <- LETTERS[1:4]
for ( i in v) {
  print(i)
}
```

Step 3. Exercises

A. Write an R function called “vecSqr” that takes a vector of length n and returns another vector where each element is the square of each corresponding element of the passed vector.

B. Write an R function called “first3let” that takes a string and returns the first three characters of that string. If the length of the string is less than 3, then print the following message: “Length of input string is less than 3.”

C. Write an R function that accepts a vector of integers and returns a logical vector that is TRUE whenever the input is even, FALSE whenever the input is odd, and NA whenever the input is non-finite, such as NA, NaN, Inf, and -Inf. Hint: you may use **“is.finite()”** function to check the last case.

D. The factorial of a non-negative integer, n, notated n!, can be algebraically defined as:

$$n! = \prod_{i=0}^{n-1}(n-i) = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$$

In the case where n=0, n! = 1. Write an R function which recursively computes the factorial.

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

> **"The best error message is the one that never shows up."** - *Thomas Fuchs*