R Programming _Basic_Task 1

1. Vector recycling

Recycling occurs when R encounters two vectors of different lengths in a binary operation, R will replicate (recycle) the smaller vector until it is the same length as the longest vector, then it does the operation.

To replicate the smaller vector, say between addition of two vectors c(1,5,9) and c(7,3,10,4), the leftover units in the longer vector will add with the first unit of the smaller vector.

Example: c(1,5,9) and c(7,3,10,4)
c(1,5,9) + c(7,3,10,4)
Output:

```
> c(1,5,9) + c(7,3,10,4)
[1]  8  8 19  5
Warning message:
In c(1, 5, 9) + c(7, 3, 10, 4) :
  longer object length is not a multiple of shorter object length
```

// we can see that the fourth unit of the longer vector has used the first unit of the shorter vector to be use for addition 4+1=5

```
> c(1,5,9,1) + c(7,3,10,4)
[1]  8  8  19  5
```

2. Inner multiplication

Inner multiplication refers to the multiplication between matrices. It is a useful matrix operation that is widely used in areas such as network theory, transformation of coordinates and many more uses. A matrix in R can be created using *matrix()* function and this function takes input vector, nrow, ncol, byrow, and dimnames as arguments.

The multiplication operator * is used.

When two matrices are multiplied, all the corresponding elements of both matrices will be multiplied under the condition that both matrices will be of the same dimension.

Example:

```r
# creating two matrices
m <- matrix(2:7, nrow=2)
n <- matrix(11:16, nrow=2)

# multiply matrices using * operator
print(m*n)
```

Output:

```
> m <- matrix(2:7, nrow=2)
> n <- matrix(11:16, nrow=2)
> print(m*n)
     [,1] [,2] [,3]
[1,]   22   52   90
[2,]   36   70  112
```

// if two matrices are of different dimension, say m matrix uses nrow and n matrix uses ncol, an error will be shown as
*Error in m*n : non-comfortable arrays*

3. Outer multiplication

Outer multiplication refers to the multiplication between arrays. It uses *outer()* function and multiply array *x* elements with array *y* elements.

Example:

```r
# two arrays
x <- c(11,4,17,11,2)
y <- c(3,8)

# use outer() function
outer(x,y,"*")
```

Output:

```
> # two arrays
> x <- c(11,4,17,11,2)
> y <- c(3,8)
> # use outer() function
> outer(x,y,"*")
     [,1] [,2]
[1,]   33   88
[2,]   12   32
[3,]   51  136
[4,]   33   88
[5,]    6   16
```

4. Functions

a. sample()

sample() function takes a sample of the specified size from the elements of *x* using either with or without replacement. Sample with replacement allows repeated elements from the same vector, while sample without replacement is the default function which does not allow repeated elements.

Example:

```
x <- 1:20
sample(x)     # sample of all elements
sample(x,5)   # sample of 5 elements
sample(x,5,replace=TRUE)     # sample of 5 elements that allow repeated elements
```

Output:

```
> x <- 1:20
> sample(x)     # sample of all elements
 [1]  7 10 13  3 20  5 14 11 16  8  2  9  4 18  6 17  1 15 12 19
> sample(x,5)   # sample of 5 elements
[1]  7 17 20 12  6
> sample(x,5,replace=TRUE)     # sample of 5 elements that allow repeated elements
[1]  4  4 19 11 18
```

Warning: *if x only has one value [4], sample() function will take it as a vector with elements (1,2,3,4)*

```
> x = 4
> sample(x)
[1] 3 2 1 4
```

b. seq()

seq() function generates a sequence of numbers.

Example:

sequence of numbers

```
> seq(-5,3)
[1] -5 -4 -3 -2 -1  0  1  2  3
```

from -5 to 3, step=2

```
> seq(-5,3, by=2)
[1] -5 -3 -1  1  3
```

7 evenly distributed numbers from -1 to 1 (length == length.out)

```
> seq(-1,1, length=7)
[1] -1.0000000 -0.6666667 -0.3333333  0.0000000  0.3333333  0.6666667  1.0000000
```

c. rep()

rep() function replicates the value of *x*.

Example:

repeat 1 to 3 three times

```
> rep(1:3, 3)
[1] 1 2 3 1 2 3 1 2 3
```

replicates each elements two times

```
> rep(1:3, each=2)
[1] 1 1 2 2 3 3
```

limit the length of return elements

```
> rep(1:3, length=5)
[1] 1 2 3 1 2
```

d. round()

round() function round a number

Example:

round a number

```
> round(4.5)
[1] 4
```

round a number with 2 digits after decimal point

```
> round(4.56382, digits=2)
[1] 4.56
```

e. factorial()

factorial() function calculates factorial.

Example:

factorial of 4

```
> factorial(4)
[1] 24
```

f. is()

is() function is to test the inheritance relationship between an object and a class or between two classes.

Example:

```
animal <- c("cat", "cow", "parrot", "lion", "mice")
animal
is.integer(animal)
is.vector(animal)
is.character(animal)
is.array(animal)
```

Output:

```
> animal <- c("cat", "cow", "parrot", "lion", "mice")
> animal
[1] "cat"    "cow"    "parrot" "lion"   "mice"
> is.integer(animal)
[1] FALSE
> is.vector(animal)
[1] TRUE
> is.character(animal)
[1] TRUE
> is.array(animal)
[1] FALSE
```

g. mean()

mean() function is used to calculate by taking the sum of the values and dividing with the number of values in the data series.

Example:

```
x <- c(3,4,5)
mean(x)
```

Output:

```
> x <- c(3,4,5)
> mean(x)
[1] 4
```

h. set.seed()

set.seed() function is useful for simulations or random objects that can be reproduced.

Example:

```
set.seed(20)
sample(1:12)
```

Output:

```
> set.seed(20)
> sample(1:12)
 [1]  6 11  8  2  1  9  5 12  3  4 10  7
```

5. Subset

Subset can be used to select and filter variables and observations. It returns subsets of vectors, matrices, or data frames which meet conditions.

6. Write a program to calculate the BMI rate?Get the user input & Result should be in integer.

- BMI rate: a measure that use your height and weight to work out if your weight is healthy

Program:

```
# user input
h <- readline(prompt="Enter height(m): ")
w <- readline(prompt="Enter weight(kg): ")

# convert into integer
h <- as.numeric(h)
w <- as.integer(w)

print(as.integer(w/h^2))
```

Output:

```
> # user input
> h <- readline(prompt="Enter height(m): ")
Enter height(m): 1.67
> w <- readline(prompt="Enter weight(kg): ")
Enter weight(kg): 55
> # convert into integer
> h <- as.numeric(h)
> w <- as.integer(w)
> print(as.integer(w/h^2))
[1] 19
```

7. Create a function to calculate the BMI Rating? Result should be in integer?

Program:

```
bmi = function(h,w){
  return (as.numeric(w/h^2))
}
print(as.integer(bmi(1.67,55)))
```

Output:

```
> bmi = function(h,w){
+   return (as.numeric(w/h^2))
+ }
> print(as.integer(bmi(1.67,55)))
[1] 19
```