Chapter 1 - R Language Fundamentals

1 R Language

R is a programming language and environment for statistical computing and graphics. R features

- Effective data handling and storage facility
- Provides a suite of operators for calculations on arrays, list, vectors and matrices
- Provides large, coherent and integrated collection of intermediate tools for data analysis
- Graphical facilities for data analysis and display for either on-screen or in print copies
- Well-developed, simple and effective language which includes conditionals, loops, user-defined recursive functions and input and output facilities

1.1 Basics

As it is convention with learning any programming language to start with an "Hello World" program, let's start with one in R.

```
myString <- "Hello World"
print(myString)
```

the output is

```
[1] "Hello World"
```

R is a language known for its ability to handle statistical computing and graphics pertaining to statistical computation. So how easy is it to generate and graph the frequencies of some random numbers from a Normal Distribution? Let's look at the code

```
n <- floor(rnorm(100, mean = 50, sd = 10))
t <- table(n)
barplot(t, xlab = "Numbers", ylab = "Frequencies")
```

The breakdown of the lines of code are as follows:

1. Using the `rnorm()` function, we generate 100 random values from a normal distribution then those values are rounded down to the nearest whole number regardless of how many decimal points it has. Store the results in `n`.
2. Using the `table()` function, we count the frequency of each number. Store the results in `t`.
3. Using a bar graph, the frequencies of the are plotted and the graph labeled.

With just 3 lines of code, we have plotted the frequencies of some random numbers (refer to figure 1 below).

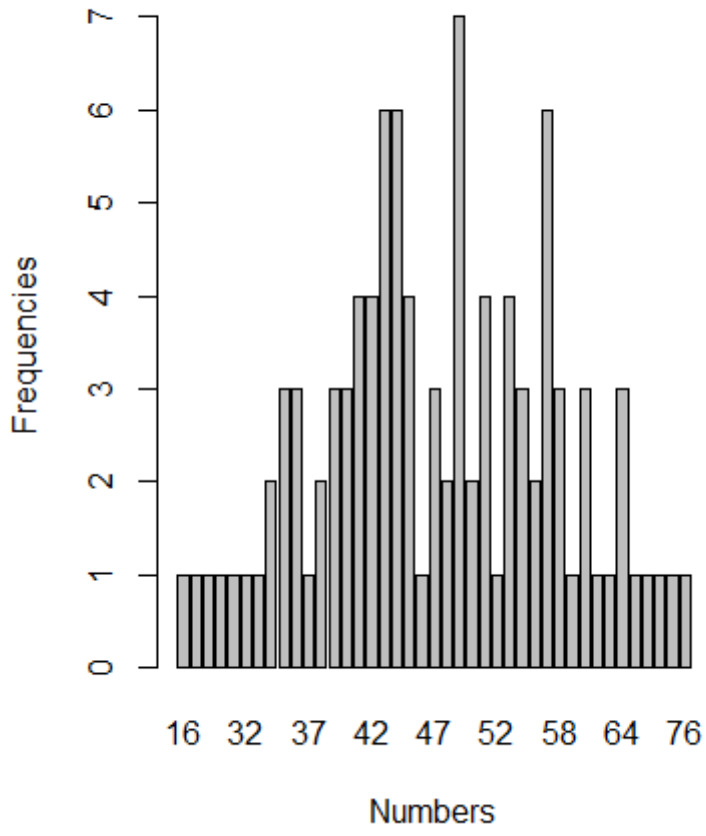

Figure 1: Barplot of the 100 normally distributed randomly generated values.

1.2 Commenting in R

Single line commenting in R uses the hash (`#`) character but R does not support multi-line comments. However, we are able to multi-line comments so long as they are enclosed within either single (' ') or double(" ") quotes.

```
# This is a single line comment

"This is a trick for
multi-line
comment"
```

1.3 Data Types

In general most programming languages have variables to store different types of information. However in R, the variables are stored in R-Objects thus the data type of the R-Object becomes the data type of the variable. The most commonly used R-Objects are

- **Vectors** - consist of 6 classes namely logical, integer, real, complex, string (or character) and raw. It uses a `c()` to create a vector.

- **Lists** - able to store vectors, functions and other lists. The elements within the lists do not have to be all of the same type.
- **Matrices** - a 2 dimensional data set that stores information of the same data type
- **Arrays** - multi-dimensional data set
- **Factors** - are created from vectors and the elements in the vectors are used as labels. Labels are always characters regardless of whichever data type is used to create the input vector.
- **Data Frames** - are more generalized versions of matrix that stores information in a tabular way

The 6 different classes of vectors are shown in the table below

Vector Class	Example	code Example
Logical	TRUE, FALSE	`v <- True`
Numeric	2,7,8,99.9	`v <- 25.36`
Integer	5L,8L,0L	`v <- 2L`
Complex	5+1i	`v <- 5+1i`
Character	'a', "hello"	`v <- "hello"`
Raw	"hello" stored as ASCII hex numbers (68 65 6c 6c 6f)	`v <- charToRaw("hello")`

Examples of the common data types of R

```r
# vectors
apple <- c('red','green',"yellow")

# list (vector, number, function)
list1 <- list(c(2,5,3),21.3,sin)

# matrices
mat <- matrix(c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)

# arrays
a <- array(c('green','yellow'),dim = c(3,3,2))

# factors
factor_apple <- factor(apple)

# data frame
person <- data.frame(
   name = c("John", "Rob","Anna"),
   gender = c("Male", "Male", "Female"),
   Age = c(42,38,26)
)
```

1.4 Variables

Variables gives us a named storage of the data that the program can manipulate. The rules of naming variables in R are as follows

- Can consists of letters, numbers and dots or underscores characters
- Able to start with a letter or a dot but for dots, it has to be followed by a letter. Variables starting with a dot are hidden.

Valid	Invalid
`var_str`	`name%`
`var_int8`	`8iname`
`int.five`	`.9_var`
`.int_five`	`_var`

To assign a value to a variable in R, we can use the `->`, `<-` and the `=` operators. Printing of information is done using the `print()` or the `cat()` functions. The difference between them is that the `print()` function is used more to print single variables whereas the `cat()` function is able to combine multiple items in a continuous output.

```
# vector
apple <- c('red','green',"yellow")

# using the cat() function
cat("Vectors: ", apple, "\n")
# using the print() function
print(apple)
```

the output is

```
Vectors:  red green yellow
[1] "red"    "green"  "yellow"
```

Each time a variable is created in the workspace, R keeps a record of it. To find out all the variables in your current workspace, use the `ls()` function and to show all variables (including hidden variables), use `ls(all.name = True)`. Variables can also be deleted from the workspace using the `rm()` function.

1.5 Operators

Operators are special symbols that tell the compiler to perform specific mathematical or logical computations. R has the following types of operators

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Miscellaneous Operators

1.5.1 Arithmetic Operators

The Arithmetic operator acts on **each element** of the vector. Assume that the variables **v = c(1,2,3)** and **t = c(4,5,6)**.

Operator	Description	Example
+	Adds two vectors.	`v+t` = (5,7,9)
-	Subtracts second vector from the first.	`t-v` = (3,3,3)
*	Multiplies both vectors.	`v*t` = (4,10,18)
/	Divide the first vector with the second.	`v/t` = (4,2.5,2)
%%	Give the remainder (modulus) of the first vector with the second.	`v%%t` = (0,1,0)
%/%	The integer result of division of first vector with second (quotient).	`v%/%t` = (4,2,2)
^	The first vector raised to the exponent of second vector.	`v^t` = (4,25,216)

1.5.2 Relational Operators

Relational operators are used to compare **each element** in the vector with the corresponding element in the second vector. Assume that the variables **v = c(9,2,3)** and **t = c(4,5,2)**.

Operator	Description	Example
>	Checks if each element of the first vector is greater than the corresponding element of the second vector.	`v>t` results in TRUE FALSE TRUE
<	Checks if each element of the first vector is less than the corresponding element of the second vector.	`v<t` results in FALSE TRUE FALSE
==	Checks if each element of the first vector is equal to the corresponding element of the second vector.	`v==t` results in FALSE FALSE FALSE
>=	Checks if each element of the first vector is greater than or equal to the corresponding element of the second vector.	`v>=t` results in TRUE FALSE TRUE
<=	Checks if each element of the first vector is less than or equal to the corresponding element of the second vector.	`v<=t` results in FALSE TRUE FALSE
!=	Checks if each element of the first vector is unequal to the corresponding element of the second vector.	`v!=t` results in TRUE TRUE TRUE

Note that if there is a mismatch of the operand's vector lengths, the shorter operand's elements are are recycled in a cyclic manner to match the length of the longer vector. The short vector's length needs to be a multiple of the longer vector's length otherwise R will return a warning.

1.5.3 Logical Operators

There are 2 types of logical operators, one type works on every element of a vector and another type that works on only the first element of a vector. These operators only work on vectors of the type logical, numeric and complex. Note that **all** numbers greater than 1 is considered TRUE.

For the first type of **element-wise** logical operators. Assume that the variables **v = c(3, TRUE, 5+8i)** and **t = c(9, FALSE, 5+8i)**.

Operator	Description	Example
&	Element-wise Logical AND operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if both the elements are TRUE.	`v&t` results in TRUE FALSE TRUE
|	Element-wise Logical OR operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if one the elements is TRUE.	`v|t` results in TRUE TRUE TRUE
!	Element-wise Logical NOT operator. Takes each element of the vector and gives the opposite logical value.	`!t` results in FALSE TRUE FALSE

For the second type of logical operators where the comparison is **only** between the first elements of both vectors. Assume that the variables **v = c(9, TRUE, 5+8i)** and **t = c(0, FALSE, 5+8i)**.

Operator	Description	Example
&&	Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.	`v&&t` results in FALSE
||	Logical OR operator. Takes first element of both the vectors and gives the TRUE if one of them is TRUE.	`v||t` results in TRUE

1.5.4 Assignment Operators

These assign values to the vectors

Operator	Description	Example
<- or = or <<-	Assign the value to the left operand.	`v <<- c(4,5,6)`
-> or ->>	Assign the value to the right operand.	`c(8,5,4) ->> v`

1.5.5 Miscellaneous Operators

These operators have a specific purpose and are not used for any general mathematical or logical computation.

Operator	Description
: (colon)	Colon operator. It creates the series of numbers in sequence for a vector.
%in%	Used to identify if an element belongs to a vector.
%*%	Used to multiply a matrix with its transpose.

1.6 References

1. Medeiros, 2018, Variable Types and Data Structures in R Programming Fundamentals, Packt Publishing
2. Learn R Programming, https://www.datamentor.io/r-programming/
3. R Tutorial, https://www.tutorialspoint.com/r/index.htm