

Lab 1

The purpose of this lab is to familiarize students with the R language and R studio (IDE) and help them learn about conditional, iterative statements and how to make new functions in R. They will also learn about various data structures like Vectors, Dataframe.

INTRODUCTION TO R -

R is an open-source statistical computing environment that is becoming increasingly popular among social scientists.

RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging, and workspace management.

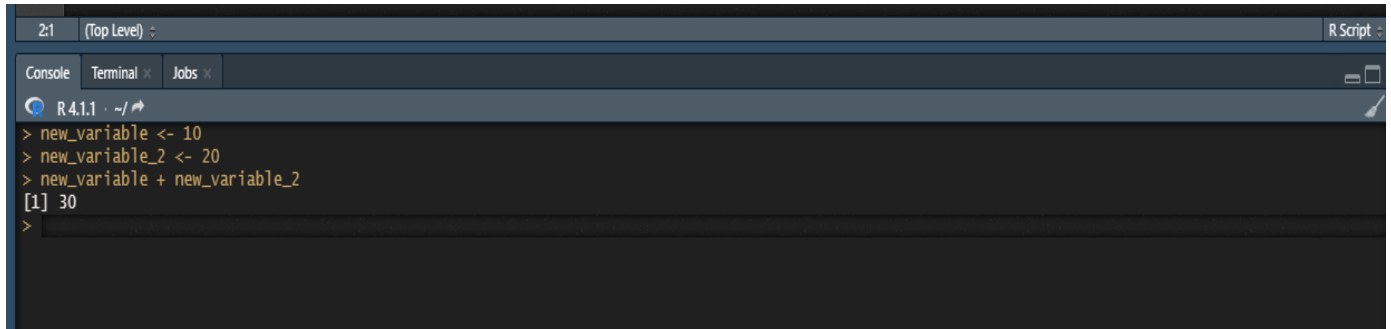
SO LET'S BEGIN !!

Copy-paste the below code in the R studio top left sub-window and then run the code snippet by selecting all the code with your mouse and then using **Ctrl+R** or **the run button at the top**.

Code -

```
new_variable <- 10  
new_variable_2 <- 20  
new_variable + new_variable_2
```

On running the code R-STUDIO is gonna run each line of code **sequentially** and will show the results in the bottom left sub-window, something like this -



```
R 4.1.1 ~ /  
> new_variable <- 10  
> new_variable_2 <- 20  
> new_variable + new_variable_2  
[1] 30  
>
```

The first line of code assigns the value **10** to a variable called **new_variable** while the second line assigns the value **20** to a variable called **new_variable_2** and in the end, the third statement **adds** these variables and **prints the result**.

STATEMENTS -

Each line of code in R is called a **statement** and there are various types of statements in **R language**.

1) Assigning statements ->

These statements are of the form -

Variable <- statement/value

These statements are used to evaluate a statement to the right and assign it to a variable in the left so that the variable can be used later on for other calculations

E.g -

- | | |
|------------------|--|
| 1) x <- 100 | #assigns the value 100 to the variable x |
| 2) y <- 100*4/50 | #R calculates the value of the expression and then assign it to the variable y |
| 3) z <- x*y | #R multiplies the value of x and y and assigns it to z |

NOTE -

1)R supports only one variable on the left and if the same variable is used its previous value will be overridden so be cautious while using similar variable names and

2)make sure all variables used on the left have been assigned a value beforehand or else an error will occur

EX1 - Try finding the mean of the first ten natural numbers using only assignment statements and assign it to a variable called “var”.

Soln.) `var <- (1+2+3+4+5+6+7+8+9+10)/10`

2) Conditional statements ->

Conditional statements are used when a particular set of statements are supposed to be run only if a particular condition is satisfied.

SYNTAX -

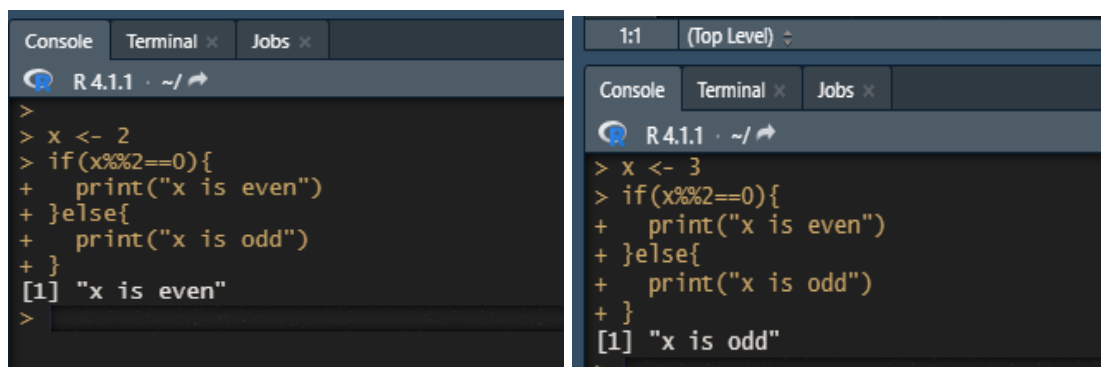
```
If (Condition 1){
  ...set of statements to run if condition 1 is true
}else if(Condition 2){
  ... set of statements to run if condition 2 is true
}else{
  .. set of statements to run if all conditions are false
}
```

Let's write an R code for testing if the value of a particular variable is even or not

Code ->

```
x <- 2
if(x%%2==0){
  print("x is even")
}else{
  print("x is odd")
}
```

Copy-paste the above code and run it in the editor and then replace 2 with 3 in the first statement and run it again do you see any difference.



```
R 4.1.1 · ~/
>
> x <- 2
> if(x%%2==0){
+   print("x is even")
+ }else{
+   print("x is odd")
+ }
[1] "x is even"
>

R 4.1.1 · ~/
1:1 (Top Level)
> x <- 3
> if(x%%2==0){
+   print("x is even")
+ }else{
+   print("x is odd")
+ }
[1] "x is odd"
>
```

Since in the First case x is even so the condition is true so statements in the if block is run while in the second case the else block is run.

EX2 - Use the knowledge of R you have learnt till now and create a programme that first assigns an age to a variable “age” and then checks if in that particular age a human is an infant(<5 years) or child(<12 yrs) or teenager(<18 years) or an adult

SOLUTION -

```
age <- 21
```

```
if(age>=18){  
    print("adult")  
}else if(age>12){  
    print("teenager")  
}else if(age>5){  
    print("child")  
}else{  
    print("infant")  
}
```

NOTE - There is always more than one way to code a problem and you're solution doesn't need to match exactly with the given solutions.

3) Iterative statements ->

Are used when you want to run a particular set of statements repeatedly for many times or until a certain condition is met. For example, if you want to print "Hello," 100 times writing print("Hello") 100 times doesn't make sense and hence here we will use iterative statements here.

A) While statement -

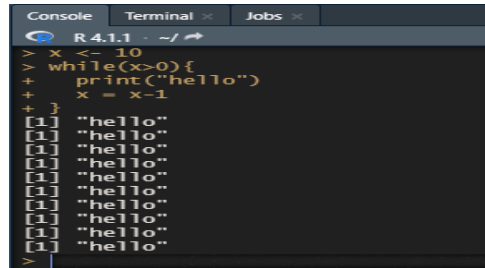
Used to run a set of statements until a particular condition is met and that condition is met before and after the set of instructions has run.

SYNTAX -

```
while(Condition){  
    ...set of statements to run until the condition is met  
}
```

Example -

```
x <- 10
while(x>0){
  print("hello")
  x = x-1}
```

A screenshot of an R 4.1.1 console window. The window has tabs for 'Console', 'Terminal', and 'Jobs'. The console shows the following code being entered: `> x <- 10`, `> while(x>0){`, `+ print("hello")`, `+ x = x-1`, and `+ }`. The output shows ten lines, each starting with `[1]` followed by `"hello"`, indicating the loop executed ten times.

This code assigns the value 10 to x and then compares the value of x to 0 and prints “hello” if greater than 0 and then the value of x is decreased by 1 and the condition is checked again and the process repeats itself till x becomes 0 and the condition becomes false thus resulting in printing “hello” 10 times.

B) FOR LOOP -

A FOR loop is used to run a particular set of instructions for a finite number of times

SYNTAX -

```
for(x in vector){
  ... set of instruction to repeat
}
```

NOTE - We will learn what a vector is later till then use **seq(number of times to repeat)** instead

Example -

```
for(x in seq(10)){
  print("hello")
}
```

DATATYPES IN R

- 1) **LOGICAL** - R supports the boolean values True and False and can be assigned to a variable using conditions or using TRUE, FALSE respectively.

Ex.

```
var <- TRUE    #assigns true to var
var <- FALSE   #assigns value false to var
var <- 10==10  #right condition evaluates to true and assigns it to var
```

Conditional operators -

- 1) >= Greater than or equal to
 - 2) > Greater than
 - 3) < Lesser than
 - 4) <= Lesser than equal to
 - 5) == Equality
 - 6) != Not Equal
 - 7) && Logical and
 - 8) || Logical or
 - 9) ! logical not
- 2) **NUMERIC** - Simple numeric values like 10,20,3.2 are stored as numeric value data type .

Ex.

```
var <- 2.3 # assigns value 2.3 to var can be used for calculations later
```

- 3) **COMPLEX** - R also supports complex numbers and enables various operations on these numbers

Ex.

```
var <- 3+2i #assigns the complex number 3+2i to var
```

- 4) **CHARACTER** - R supports all characters to be stored in a variable. but note these strings are always represented by double codes on both sides else R will confuse it with a variable name.

```
Var <- "hello"  
Var <- "learning R is fun"  
Var <- "enjoying R is fun"
```

VECTORS -

It is the main data structure used in R. It is basically a group of **similar data types** and is assigned by the function **c**.

Ex.

```
Var <- c(1,2,3,4,5,6)  
Var <- c("hello","my name is dave","R is fun")  
Var <- c(True,False,True)  
Var <- c(3+2i,4+3i,5+6i)
```

To access a particular value of a vector use ->

Var_name[index]

For example

if

```
Var <- c(1,2,3,5,6,7,10) then var[1] = 1,var[2]=2 and so on till var[7]=10
```


Lists

A list is an R-object that can contain many different types of elements inside it like vectors, functions, and even another list inside it.

Ex.

```
# Create a list.  
list1 <- list(c(2,5,3),21.3)  
# Print the list.  
print(list1)
```

Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

Ex.

```
# Create a matrix.  
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow =  
TRUE)  
print(M)
```

```
> M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)  
> print(M)  
  [,1] [,2] [,3]  
[1,] "a"  "a"  "b"  
[2,] "c"  "b"  "a"  
>
```

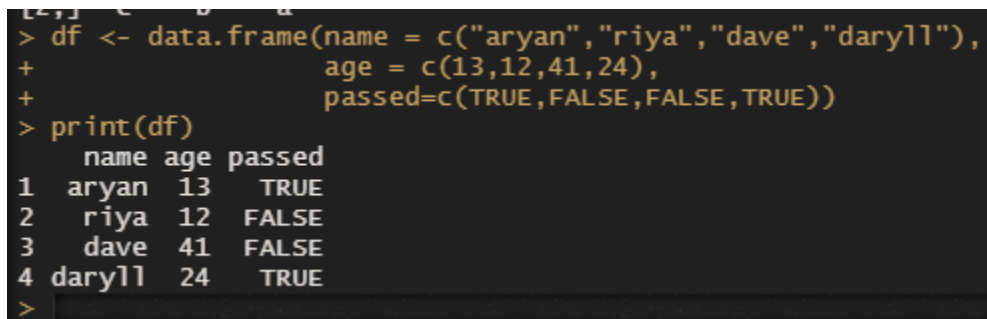
Data Frames

Data frames are tabular data objects. Unlike a matrix in a data frame, each column can contain different modes of data. The first column can be numeric while the second column can be character and the third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the **data.frame()** function.

Ex.

```
df <- data.frame(name = c("aryan","riya","dave","daryll"),
                 age = c(13,12,41,24),
                 passed=c(TRUE,FALSE,FALSE,TRUE))
print(df)
```



```
> df <- data.frame(name = c("aryan","riya","dave","daryll"),
+                 age = c(13,12,41,24),
+                 passed=c(TRUE,FALSE,FALSE,TRUE))
> print(df)
  name age passed
1  aryan  13   TRUE
2   riya  12  FALSE
3   dave  41  FALSE
4 daryll  24   TRUE
```

PRACTICE EXERCISES

- 1) Write a programme that checks the age of a person and then prints whether he or she is eligible to vote or not?
- 2) Write a programme that prints the first 1000 numbers with the odd number printed only once while the even numbers are printed twice?

Like -

1

2

2

3

4

4

.

.

So on

- 3) Take a vector of numeric values and find the maximum and minimum of the vector. NOTE - use **max(vector)** and **min(vector)** to find the maximum and minimum of the vector

- 4) Given a vector of strings write a programme that prints only even length strings and ignores the odd letter ones. Note use the **nchar()** function to get the length of a particular string.

- 5) Make a 3X3 matrix having different values and then double each even value in the matrix and half each odd value.
Note in the case for matrix each element is accessed by **matrix[row,column]**