20/1/2020

Unit - 3

1) Hadoop Ecosystem & Components (Internal / In-built)
2) Hadoop Processing Tools (External)

→ Third party - is not closely associated

→ R Programming Language is an Hadoop Processing Tool.

21/1/2020

2. Hadoop Processing Tools

→ 40 tools → 20 tools → 10 tools

Hadoop Processing Tools

→ The various processing tools that support big data and Hadoop to analyse, sort, report, ... etc are

1) Apache Hadoop
2) Cass Andhra
3) Data Wrapper
4) Mongo DB (Database)
5) Lumify
6) Storm
7) Apache SAMOA
8) Rapid Miner
9) Tableau
10) R Programming Language
11) Couch DB
12) Data Cleaner
13) Hive
14) Spark
15) Oracle Data Mining
16) Teradata
17) SILK
18) Impala
19) Mahout
20) FLUME

1) Apache Hadoop :-
The Apache Hadoop Framework is used for clustered filesystem and is used to control Bigdata.

2) CassAndhra :-
→ It is an open source NOSQL Distributed Database Management System (DBMS), which is used to handle large amount of data on various commodity servers.
→ To interact with the database it uses ~~CQSL~~ CSQL (CassAndhra Structured Query Language)

3) Data Wrapper :-
It is an open source platform ~~tool~~ for Data Visualization and it is used to generate simple, precise and embedded charts.

4) Mongo DB :-
→ It is an open source tool for NOSQL Oriented Database. It is written in 'C' or 'C++' or 'Java'.
→ It supports various operating systems.
→ It is used for aggregation, ad hoc queries, indexing, replication, ... etc.

5) Lumify :-

It is an open source tool which is used for data integration, analytics and visualization.

22/1/2020

6) Storm :-

It is a crossplatform computational framework which is used for distributed stream processing and to handle fault tolerance.

7) Apache SAMOA :-

→ SAMOA stands Scalable Advanced Massive Online Analysis.

→ It is an open source platform for Bigdata stream mining and machine learning which run on distributed stream processing engines.

→ It is an alternative to "BigML"

8) Rapid Miner :-

It is a crossplatform tool which provides an integrated environment for DataSaience, Machine learning and Predictive Analysis.

9) Tableau :-
It is used for data visualization and exploration in business intelligence and analytics tools, which are capable of handling all sizes of data.

10) R :-
R Programming Language is used mostly for statistical analysis and graphical representation of data.

11) Couch DB :-
It is an Apache Opensource crossplatform document oriented NOSQL Database which is used to hold scalable architecture.

12) Data Cleaner :-
It is a Python based data quality tool which cleans the datasets and prepares them for analysis and transformation.

13) Hive :-
Apache Hive is a Java based cross platform datawarehouse tool which is used for Data Summarization, Query and analysis.

14) Spark :-
Apache Spark is an open source framework

for Data Analytics, Machine learning algorithms and fast cluster computing.

15) Oracle DataMining :-

This tool is used for DataMining and for Analytics which allows us to create, manage and deploy Oracle data.

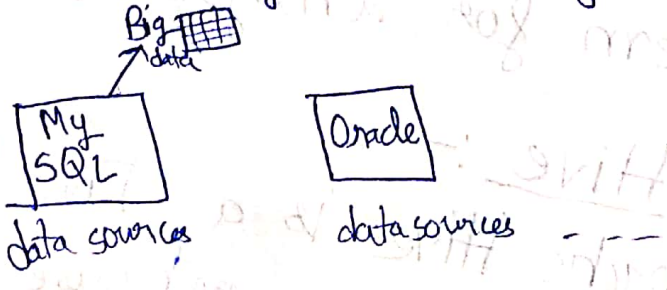
24/1/2020

16) Teradata :-

Teradata provides Data Warehousing products and services like Analytics Platform which combines analytic functions and injects, preferred analytical tools, artificial intelligence technologies and languages and multiple data types in a single workflow.

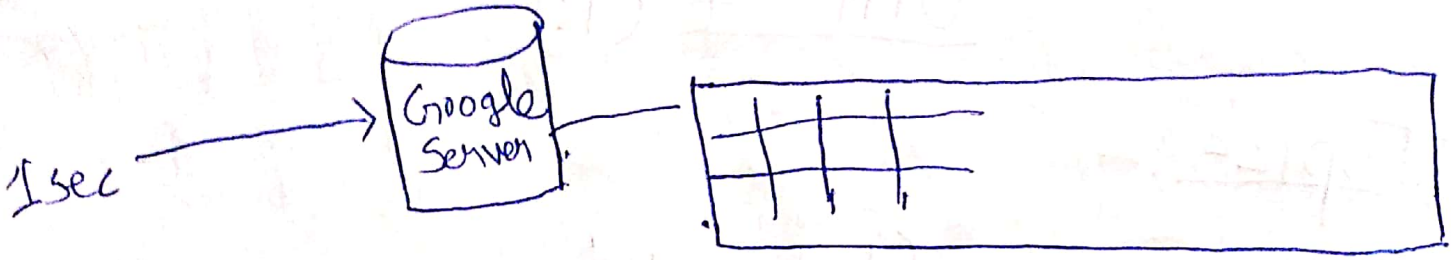
17) SILK :-

It is an open source framework which is used to integrate hetegenous data sources.



18) Impala :-

→ Impala is an MPP (Massive Parallel Processing) SQL Query Engine which runs on Apache Hadoop



→ The Impala Open Source tool combines modern, scalable parallel database technology with Hadoop and enables the users to directly query the data which is stored in HDFS.

19) Mahout :-

Apache Mahout is a library of scalable machine learning algorithms implemented by using Map Reduce Paradigm in Hadoop.

20) FLUME :-

→ FLUME is a distributed and reliable service which is used for efficiently collecting, aggregating and moving large amounts of data.

→ FLUME has a simple and flexible architecture based on streaming dataflows.

→ FLUME is robust and fault tolerant.

25/1/2020

Unit - 4 (R)

Topics:

1) R Overview (Not important)
2) Basic Syntax
3) Data Types ——— ✓ implicit
4) Variables
5) Operators ——— ✓
6) Decision Making (Decision Control) } ——— ✓
7) Loops
8) Functions ——— ✓

Decision Making = if, if-else, ~~switch~~ while, ~~do-while~~, for

Functions:

```
int add();
add();

function add()
{
=
=
=
=
}
```

1) R - Overview

→ R was developed by Ross Ihaka and Robert Gentleman in 1993.

→ They belong to Statistics department University of Auckland, New Zealand.

→ R Programming is a scripting language which is used for Statistical Programming in the areas of

1) Statistical Analysis
2) Data Manipulation

→ R Programming is an Interpreted Language (Here we execute the command in the command line and gets interpreted instantly)

→ R Programming incorporates the (includes as it has) features of functional progeramming language and object oriented programming language.

27/1/2020

```
>a=25
>print(a)
 (1) 25

>11+11
 (1)22
>1:100
>100:1
>-20:20
```

```
> b=32
> print(b)
(1) 32
> c=a+b*b
> print(c)
```

```
> 32 -> a
> 33 -> a
> print(a)
  (1) 33
> print(a*b)
  (1) 1056
```

a ≠ 5
a ← 5
5 → a
5 = a
32 → a

Saving the file

R - Editor

```
a <- 55
b <- 66
c = b - a
print(a)
print(b)
print(c)
```

File name : P1.R

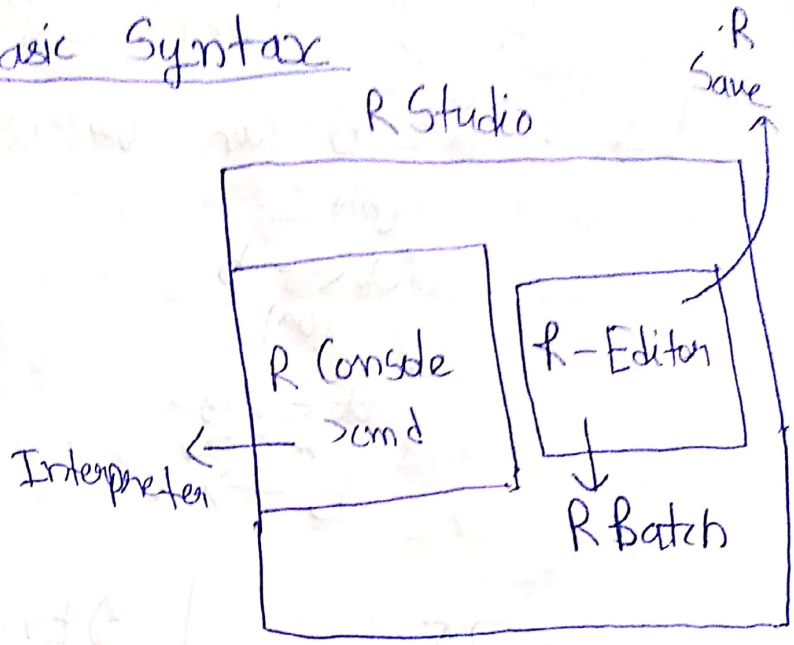
Ctrl + R = Editor or Sends the code to Console

28/1/2020

2. Basic Syntax

→ The R Software majorly consists of 2 modes

They are
- Interactive mode
- Batch Mode



An Interactive mode is represented as R Console in R Programming Language where we can run R Commands and as well as R scripts

A Batch mode in R is represented by an R editor where we can write, save and also execute R programs as R script files. The R script files must be saved with an extension ".R"

Eg :- filename.R

4. Variables

→ Variables in R are used to initialize values by using assignment operator.

→ The various assignment operators in R are "=" (Equi Assignment)

"←" (Left assignment)

" → " (Right assignment)

Eg :- a=5 can be written as

(only)
a ← 5
(or)
~~a 5~~
5 → a

1) a=25
b=5
c ← a+b
print(a)
print(b)
print(c)

2) a ← 25
5 → b
print(a*b)

3) Evaluate $\sqrt{a^2+b^2}$

a=5
~~b~~
b ← 5
c = sqrt((a*a)+(b*b))
print(c)

4) Evaluate $\sqrt{a^6+b^5}$

a = 5 (a ← 5)
b = 6 (b ← 6)
(c ← sqrt(a^6 + b^5))
c = sqrt(a^6 + b^5)
print(c)

5) Initialize strings with variables

x ← "aditya"
print(x)

Multiple strings assigned to a variable

6) x ← "aditya" "degree" "college"

Concatenate

↓R

c (" ")

Concatenation

```
y ← c("aditya", "degree", "college")
print(y)
```

O/P :- aditya degree college

7) Multiple numbers assigned to a variables (values)

```
z ← 5,4,3,2   X
```

```
z ← c(5,4,3,2)
print(z)
```

Comments

The descriptions in R (or) documents can be written by using "#" symbol.

Example :-

```
# Addition
a = 5
b = 6
print(a+b)
```

* 3) Datatypes :- / R objects (10M)

→ R uses implicit data types but does not use explicit datatypes

→ No variable declaration is needed.

① The various datatypes in R programming Language referred to as "R objects", they are

- Vectors
- lists
- arrays
- Matrices
- factors
- Data frames

29/1/2020

Vectors :-

① The Vectors in R are classified into 6 classes, they are :-

1) Numeric
2) Integer
3) Character
4) Complex
5) Bodean
6) Raw

1) Numeric :- A Numberic class is used to store real values (float) and

also integer values.

→ It is the default class where if no representation is provided to a value, it implicity considers the value as numeric.

Eg:-
```
"Numeric" { a ← 5.5
          { b ← 6
          { c ← -6.5
Print(a)
Print(b)
Print(c)
ca ← class(a)      Numeric
cb ← class(b)      Numeric
cc ← class(c)      Numeric
print(ca)
print(cb)
print(cc)
```

O/P :- 5.5
6
6.5

O/P : "Numeric"
"Numeric"
"Numeric"

→ The class() is used to display the datatype associated with a variable

2) Integer :-

→ Integer class is used to store positive (or) negative integer numbers.
(55, -65... etc)

→ To initialize an integer value to a variable, the value must be ended with "L" (55L, -65L.... Etc)

Eg:-

```
x ← 25L          | cx ← class(x)
y ← 25L          | cy ← class(y)
z ← x + y        | cz ← class(z)
print(x)         | print(cx)
print(y)         | print(cy)
print(z)         | print(cz)
```

30/1/2020

3) Complex :-

real + imaginary
↓ ↓
+ i

⇒A complex class is used to store a complex number, which is a combination of real value and imaginary value.

→ The imaginary value is represented with "i" at the end of the value (7i, 6i, ... etc)

→ The real value and imaginary value are concatenated with '+' symbol

Eg:-

```
x ← 3 + 4i
y ← 5 + 6i
z ← x + y
```

Numeric x ← 5
5.5

integer x ← 5L

Complex

Character

B|l

R

```
print (x)
print (y)
print (z)
print ( class (x))
print ( class (y))
print( class (z))
```

4) Character :-

→ The character class is used to store single characters (or) multicharacters (strings). The single characters must be denoted with single quotes ' ' and the multi characters must be denoted with double quotes " ".

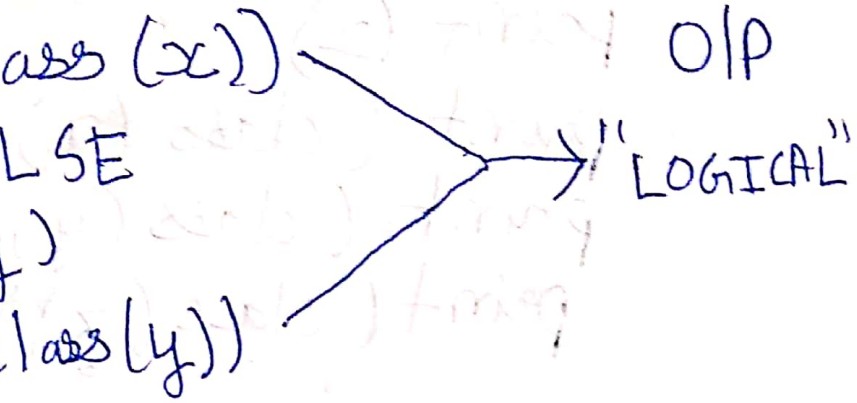
Eg :-

```
x ← 'a'
print(a)
print ( class (a))
y ← "aditya"
print (y)
print ( class (y))
```

5) Boolean / Logical :-

The Logical class is used to store Boolean values. The Boolean value 1 is represented as TRUE and the Boolean value 0 is represented as FALSE

Eg :- x ← TRUE
print(x)
print(class(x))
y ← FALSE
print(y)
print(class(y))



6) Raw :-

→ Raw class is used to store the data in hexadecimal format.

→ To convert character data into hexadecimal format, we use charToRaw().

→ The charToRaw() converts only character data or string data into hexadecimal value.

Eg :- x ← "aditya"
print(x)
y ← charToRaw(x)
print(y)
~~print(class~~
z ← class(y)
print(z)

31/1/2020

Applications of R :-
(Areas)

The various applications (or) areas in R are :-

1) **Finance & Banking**
It is used for making decisions in Finance and to detect fraud transactions in Banking

2) **Bio Informatics :-**
It is used to analyse genetic sequences and to perform computational neuroscience.

3) **Social Media :-**
It is used to analyze social media information

4) **E-Commerce area :-**
It is used to analyze the purchases made by the customer and to predict the feedbacks

5) **Statistics apps :-**

6) **Data Analysis :-** R is capable of analysing any format of data

7) **Probability distribution :-**

8) **Linear Regression :-**

R-Programming

1)
```
x = 5
print(x)
```
Output :- [1] 5

2)
```
x = 5
print(class(x))
```
Output :- numeric

3)
```
x = 5L
print(class(x))
```
Output :- integer

4)
```
x = 5i
print(class(x))
```
O/P :- complex

6) x = 5i
print (class(x))
print(x)

OIP :-
[1] complex
(1) 0+5i

7) x = 4+5i
print(class(x))
print(x)

OIP :-
(1) complex
(2) 4+5i

7) x = "aditya"
print(x)
print(class(x))

OIP :-
[1] "aditya"
[1] "characters"

8) x = TRUE
print(x)
print (class (x)
(1) TRUE
OIP :- [1] logical

9) x = "aditya"
y = charToRaw(x)
print(x) print(y)
print(class(y))

Additional functions

c(.)

10) x ← c(5, 6, 7)
print(x)
O/P :- 5 6 7

11) x = 5, 6, 7
print(x)

~~x = 6~~
O/P :-
Error: unexpected ',' in "x=5,"
Execution halted

12) x = c(5L, 6L)
print(x)
O/P :- [1] 5 6

13) x = c(5i, 6i)
print(x)
o/p :- 0+5i 0+6i

14) x = c(5i, 6L)
print(x)

O/P :- [1] 0+5i 6+0i

15) x = c(3, 4L)
print(x)
O/P :- 3 4

16) x = c(3i, 4L)
print(x)
O/P :- [1] 0+3i 4+0i

17) x = c[3L, 4i)
print(x)

O/P :- [1] 3+0i 0+4i

18) 1 : 10
O/P :- [1] 1 2 3 4 5 6 7 8 9
10

Additional function

1) c() :-

→ the concatenate function is used to combine two or more vectors. The function used is c()

Eg :- (i) x ← c(5,6,9,7)
print(x)

O/P :- # 5 6 9 7

(ii) y ← c(5L, 6L, 7L)
print(y)

(iii) z ← c(5i, 3i)
print(z)

(c ⇐ r + i)
1 ↓ ↓
Complex real imaginary
number number numb

c ⇐ r + i
0 ↓

O/P :- 0+5i . 0+3i

(iv) str1 ← c("aditya", "colkge")
print(str1)

str2 ← c("5", "5L", "5i", "5a")
print(str2)

2) Indexing :- [ ]

Index = referring something

→ Indexing is used to refer a particular value in a list. We indicate the indexing sequence by using [ ]

Eg :- x ← c(11, 22, 33, 44, 55)

x[1] x[3] x[5]

x[2] x[4]

print(x[3]) #33

print(x[2:4]) #22 33 44

x[234]

x[2] x[3] x[4]

22 33 44

(In C, C++, Java counter will be starting from 0, but in R counter will be starting from 1)

3) Colon operator :-

→ Colon operator is used to acquire a range of values

Eg :- >1:5 #1 2 3 4 5

>3:1 #3 2 1

>1:3 + 3:1 #4 4 4

1 2 3

+ 3 2 1

4 4 4

```
> 1 : 3 * 3 : 1   # 3 4 3
> x <- 1 : 3
  y <- 3 : 1
  print(x+y)
# 6 2 10
```

4) Sequence function :-

→ The Sequence function seq() is used to display a sequence of values in a particular range. The ~~sequence funct~~ seq() attributes are from, to, by

Eg :- seq(from = 1, to = 10, by = 2)

```
      1  3  5
# 1  3  5  7  9
```

odd no (1-10)

```
# 1  3  5  7  9
```

seq(from 2, to = 10, by = 2)

```
# 2  4  6  8  10
```

5) any() & all()

→ any() is used to check for a condition and returns a Boolean value

Eg:- x ← 1 : 10

any (x > 4) # TRUE

1 2 3 4 5 6 7 8 9 10
F F F F T T T T T T

all (x > 4) # FALSE

1 : 10

6) Cumulative functions :-

There are two cumulative functions cumsum() for cumulative sum and cusprod() for cumulative product.

~~1 : 10~~

Eg:- x ← c (5, 1, 2, 3)

y ← c (2, 3, 1, 4)

cumsum(x) # 5 6 8 11

cumprod(y) # 2 6 6 24

~~cumprod(z) # 120~~

7) Compare operator :- >, <, ----

Eg:- x ← c (5, 1, 2, 3)

y ← c (2, 3, 2, 4)

T F F F

(x > y) # TRUE F F F

(x < y) # F T F T

8) repeat function :-

→ To repeat multiple statements or single statement in R we use rep()

Syntax :- rep(variable, count)

Eg :- rep("aditya", 3)

O/P :- #aditya aditya aditya

9) 't' operator :-

Eg :- x ← c(2, 3, 4)

y ← c(4, 3, 2)

x+y #6 6 6

x ← c(2, 3, 4)

y ← c(4, 3,)

x+y # 6 6 4

x+(2) #4 5 6

(234)

7/2/2020

2) Lists :-

Lists ~~have~~ in "R" can be implemented by using list(). The list() is used to concatenate two or more elements with different datatypes.

Eg:- x<-list(5, 6L, 7i)

y<-list("aditya", 6L)

print(x)

z<-list(c(5,6,7), c(5L,6L), c("aditya", "Pragathi"))

print(z)

Output:-

5 6 7

5L 6L

"aditya" "pragathi"

x<-c(5,6,7)

print(x) # 5 6 7

x<-list(5,6,7)

print(x) 5 6 7

Eg:-(2) x<-list(2, c(3,4))

y<-list("aditya", c(5i))

z<-list(x, y)

print(z)

Output:- # 2 3 4 "aditya" 0+5i

3) Matrices (or) Arrays :-

Matrix
Represents the value in row by column

→ Matrices in R can be implemented by using matrix()

→ The matrix() is used to store the elements in a row by column format

→ The matrix() has two attributes nrow, ncol which represents number of rows and number of columns.

Syntax :- matrix(vectors_list, nrow =val, ncol =val)

Eg :-

$$x \leftarrow \begin{bmatrix} 4 & 1 \\ 6 & 9 \end{bmatrix}_{2\times 2}$$

```
x <- matrix(c(4,1,6,9), nrow=2, ncol=2)
```

c(4,1,6,9)

4 → { 1×4 (r c), 4×1, 2×2 }

```
print(x)
```

Q Write a program for addition of two matrices with values

```
x <- matrix(c(4,1,6,9), nrow=2, ncol=2)
y <- matrix(c(2,5,3,7), nrow=2, ncol=2)
print(x+y)
```

8/2/2020

→ Matrix can handle 2 D arrays

4) Arrays :-

→ Arrays in R are used to store matrix values of more than two dimensions.

→ The function used to represent arrays in R is array().

→ The attribute used in array() is dim where the values are to be assigned in row, column and dimension format

Syn : array (Vector_list, dim = n, c, d)

Eg :- x <- array (c(1,2,3,4), dim=c(2,2,2))

~~y <- array(1:9)~~

```
y <- array(1:9, dim = c(3,3,2))
z <- array(1:7, dim=c(3,3,2))
print(x)
print(y)
print(z)
```

Output :-

$$\begin{bmatrix}1 & 3\\ 2 & 4\end{bmatrix}_{2\times2} \quad \begin{bmatrix}1 & 3\\ 2 & 4\end{bmatrix}_{2\times2} \leftarrow x$$

$$\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}_{3\times3} \quad \begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}_{3\times3} \leftarrow y$$

$$\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 11\\ 3 & 6 & 2\end{bmatrix}_{3\times3} \quad \begin{bmatrix}3 & 6 & 2\\ 4 & 7 & 3\\ 5 & 1 & 4\end{bmatrix}_{3\times3} \leftarrow z$$

2) x ← 1:4
y ← 1:9
z ← array (x, y, dim = c(3,3,2))
print(z)

10/2/2020

5) Factors :- (5M)

Factors in R can be implemented by using factor(). The factor() is used to distinguish different elements in a vector list.

→ To Identify the number of different elements in a vector can be done by using nlevels()

Eg :- x ← c("Bus", "Car", "Bus", "Train", "Car")
y ← factor(x)
print(y) # Bus Car Train

```
z ← nlevels(y)
print(z)  # 3
```

6) data frames** :- (5M)

2D	3D
matrix( )	array( )
ncol	dim
nrow	+
+	~~Ver~~ V
V	Column X (No column

→ data frames in R can be implemented by using data.frame( )

→ data.frame( ) is used to represent a list of values in a tabular matrix format.

Output :-

sid	sname	sage	sgender
S01	Ramu	20	M
S02	Rani	20	F
S03	John	20	M
S04	Hari	21	M

```
x ← data.frame(
    sid ← c("S01", "S02", "S03", "S04"),
    sname ← c("Ramu", "Rani", "John", "Hari"),
    sage ← c(20, 20, 20, 21),
    sgender ← c('M', 'F', 'M', 'M')
)
```

11/2/2020

5. Operators (10M)

The Operators in R are classified into 4 categories, they are :-

1) Assignment operators } values
2) Arithmetic operators }
3) Relational operators } Boolean. TRUE/FALSE
4) Logical operators }

1) Assignment operators :-

→ The Assignment operators in R are used to assign values (literals) to a variable by using =, ←, <<-, →, →>

Eg :-

= (Equal) (or) ← (or) <<- } Left Assignment operator	x = 5 (or) x ← 5 (or) x <<- 5
→ (or) →> } Right Assignment	5 → x (or) 5 →> x

2) Arithmetic operators

→ The Arithmetic operators are used to perform arithmetic calculations for a regular expression. The various arithmetic operators are :-

+, −, *, /, %%, ^

addition (+)	x ← c(2,3,4) y ← c(1,3,2) print (x+y) Output :- 3 6 6
Substraction (−)	x ← c(2,3,4) y ← c(1,3,2) print (x−y) Output :- 1 0 2
Multiplication (*)	x ← c(2,3,4) y ← c(4,5,6) print(x*y) Output :- 8 15 24
Division (/) Quotient	x ← c(8,4,7) y ← c(4,2,1) print (x/y) Output :- 2 2 7

Modulo (%%) Remainder	x ← c(8, 5, 1) y ← c(4, 2, 1) print (x %% y) Output :- 0 1 0
Power (^)	x ← c(2, 5, 1) y ← c(4, 2, 1) print(x^y) Output :- 16 25 1

3) Relational Operators :-

The Relational operators are used to compare values. The various relational operators are :-

<, >, <=, >=, ==, !=

Less than (<)	x ← c(5, 1, 2) y ← c(3, 1, 3) print (x < y) Output :- FALSE FALSE TRUE
Greater than (>)	x ← c(5, 1, 2) y ← c(3, 1, 3) print (x > y) Output :- TRUE FALSE FALSE

Less than or Equal to (<=)	x ← c(5, 1, 2) y ← c(3, 1, 3) print(x <= y) Output :- FALSE TRUE TRUE
Greater than or Equal to (>=)	x ← c(5, 1, 2) y ← c(3, 1, 3) print(x <= y) Output :- TRUE TRUE FALSE
Equal to (==)	x ← c(5, 1, 2) y ← c(3, 1, 3) print ~~(x <= y~~ (x == y) Output :- FALSE TRUE FALSE
Not Equal to (!=)	x ← c(5, 1, 2) y ← c(3, 1, 3) print(x != y) Output :- TRUE FALSE TRUE

13/2/2020

4) Logical

→ The various Logical operators are :-

& , | , && , || , !

vector — scalar

Eg :-

(i) & (and)

```
x ← c(5, 6L, 0, 7i)
y ← c(6, 5L, 1, 6i)
print(x & y)
```

O/P :- "TRUE" "TRUE" "FALSE" "TRUE"

		O/P
T	T	T
T	F	F
F	T	F
F	F	F

(ii) | (OR)

```
x ← c(5, 6L, 0, 7i)
y ← c(6, 5L, 1, 6i)
print(x | y)
```

O/P :- T T T T

Truth Table

		O/P
T	T	T
T	F	T
F	T	T
F	F	F

(iii) Logical and (&&)

```
x ← c(5, 6L, 0, 7i)
y ← c(6, 5L, 1, 6i)
print(x && y)
```

O/P :- "TRUE"

→ Logical and results a scalar output

(iv) Logical Or (||)	x ← c(5, 6L, 0, 7i) y ← c(6, 5L, 1, 6i) print (x || y) o/p :- "TRUE"
(v) Logical Not (!)	x ← c(5, 6L, 0, 7i) y ← c(6, 5L, 1, 6i) print (! x) #F F T F print (! y) #F F F F

Control Statements

Conditional control

1) if
2) if - else
3) Nested if else
4) Nested if
5) Switch

Loop control

1) while
2) ~~do-while~~
3) ~~for~~
2) for
3) next, break

(next is similar to continue)

5) Decision making (10M)

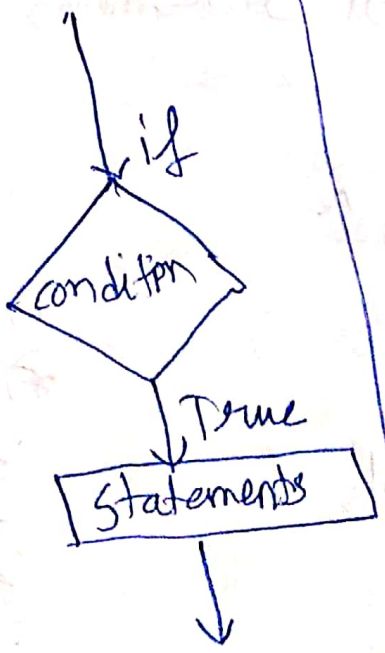
The various decision making statements can also be called as Conditional control statements, they are :-

① if statement
② if else statement
③ switch statement

① if statement :-

→ If statement is used to evaluate Boolean expression. If the condition is true, it returns a single statement or compound statement.

Flowchart :-



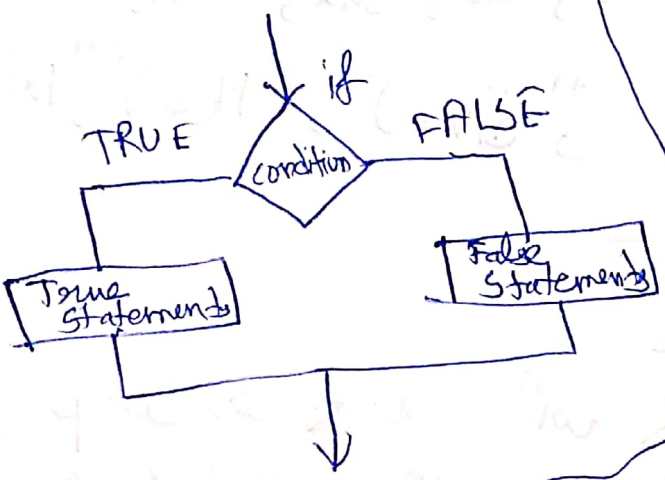
Eg :-1)
```
x ← 5
if (x == 5)
print ("TRUE")
```

2)
```
y ← 6L
if (is.integer (y))
print ("value is an Integer")
```

② if else statement :-

→ If else condition statement is used to evaluate Boolean expression, where a set of statements gets individually executed both for true or false conditions.

Flowchart :-



Eg :-

```
y ← 6i
if (is.integer(y))
print ("value is an Integer")
else
print ("value is Not an Integer")
```

14/2/2020

Eg 2 :-

```
z ← c ("aditya", "pragathi", "ideal")
if ("aditya" in z)
print ("you are aditya student")
else
print ("you are not aditya student")
```

③ switch :-

→ A switch statement is used to check the condition and redirects to an appropriate case.

Syn :- switch (Expression, case1, case2, ... casen)

Eg:- x ← 1
y ← switch (x, "aditya", "Ideal", "Pragathi")
print(y)

Output :- # aditya

Eg :- z ← switch(x, "add()", "sub()", "Mul()", "div()", "Mod()", "Pow()")

7. Loops

The Loops in R will work as Loop control statements. The various loop control statements are :-
while, for, next, repeat, break

1) while statement :-
→ A while statement is used to repeat a set of statements until a condition is not satisfied.
→ We implement it by using while key word, incrementation and initialization.

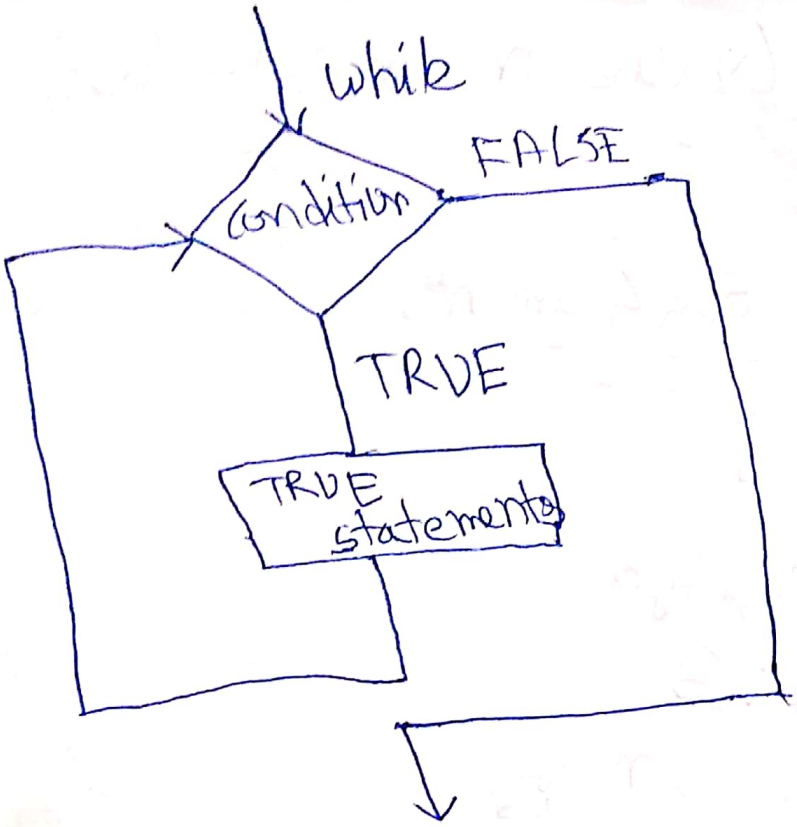


Fig:- Flowchart

Eg :- x ← "adityq"

```
i ← 1
while (i <= 3)
{ print(x)
  i = i + 1
}
```

2) for statement :-

→ A for statement is a loop control structure which is used to execute a set of statements repeatedly based on the ~~con~~ condition

→ for loop in R uses "in" keyword. for keywords.

Syntax :- for(value in vector - list)

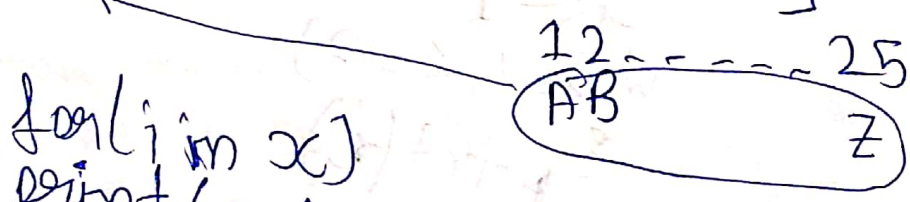
```
{
  - - - - - -
  - - - - - -
  for Statements
  - - - - - -
  - - - - - -
}
```

Eg :-

```
x <- "aditya"
y <- 1:3
for(i in y)
{
  print(x)
}
```

Eg2 :-

```
x <- LETTERS[1:26]
for(i in x)
print(i)
```




```
y <- letters[1:26]
for (i in y)
print(i)
```

15/2/2020

repeat, break & next Statements

repeat statement is an infinite loop.

→ A repeat statement is a loop control statement which never exits

from the loop.

→ If we need to exit from the loop, we use break statement along with a condition.

Syntax :-

```
repeat
{
  ------
  ------
  Statements
  ------
  ------
  if (condn)
  {
    break
  }
}
```

Eg :-

```
repeat
{
  a<-3
  b<-2
  if (a=b)
  {
    break
  }
}
```

Eg :-

```
x <- "aditya"
i<-1
repeat
{
  print(x)
  i=i+1
  if (i>3)
  {
    break
  }
}
```

```
#aditya
aditya
aditya
```

next statement :-

→ next is used to skip one iteration

→ next statement is used to skip the current iteration of a loop without termination.

x ← LETTERS[1:6]

ABCDEF

for (i in x)
{
if (i == "D")
{
next
}
print(i)
}

A
B
C
E
F

Letters of Index 1 = A]
LETTER[1] = A

2 = B

26 = Z

8. Functions :- (5M or 10M)

→ A function is a subprogram which is used to perform a specific task.

→ A function has majorly 4 components :-

1) function name
2) function body / definition (or) definition of a function
3) arguments
4) return value

1) **function name :-** It is the name given to a function and stored as an R object.

2) **function definition :-**
It contains a set of statements to perform a specific task.
(collection (or) group of statements)

3) **arguments :-**
arguments are values used in a function call as actual parameters and

in function definition as formal parameters.

4) return value :- It is the value returned by the function definition.

17/2/2020

functions Examples

1) Eg :-

```
> add <- function()
      x <- 5
      y <- 6
      print(x+y)
  }
```

Function body / definition

```
>add()   #F call
```

```
main()
{
  ...
  add();
}
```

```
>add <- function()
{
  ...
}
```

2) Eg :-

```
>pow <- function()
  { a<-2
    b<-3
    print(a^b)
  }
>pow()
```

3) Eg :-

```
pow <- function(a,b)
  {print(a^b)
  }
```

(2 → a, 3 → b)

```
>pow(2,3)   # with arguments
```

4) Eg :- default values as arguments

```
fun<- function(a, b=2)
  {print(a^b)
  }
```

~~>fun(3)~~ >fun(3,3)

> fun(3)

5) Eg :- Squares of numbers upto a given range

```
> sqn <- function(x)
                  ↳5
{
  for(i in 1:x)
                  1:5
  {               1 2 3 4 5
    y <- i^2   (i^3 = for cubes)
    print(y)
  }
}
```

> sqn(5)

Unit-5

3) — Vector
4) — Strings
5) — Lists
6) — Matrices
7) — Dataframes

1) Graphics
2) Packages

18/2/2020

Unit-5 (HADOOP & R Language)

2. Strings

→ A string in R is used to store multiple characters in an R object.

→ A string can be enclosed in single quotes or double quotes (quotations)

Eg :- x ← "aditya"
y ← 'college'

→ R can handle several string manipulation functions, they are :-

1) Paste() :-

Eg :-
```
x ← "Good"
y ← "Morning"
z ← 'Students'
a ← Paste(x, y, z)
print(a)        # Good Morning Students
```

Eg :- 2)
```
x ← "aditya"
y ← "college"
z ← Paste (x, y, sep = " ")
print(z)        # aditya college
```

→ Sep stands for separator

Q) Write a program to display strings

3) d ← 18
m ← 02
y ← 20
date ← paste (d, m, y, sep = "/")
print (date)

o/p :- 18/2/20

2) format() :-

→ format() is used to format strings and also numbers.

Eg:- 1) x ← 12.3456
print(x) # 12.3456
y ← format (x, digits = 2)
o/p :- 12

print(y)

Eg 2 :- x ← 34.56789
print(format(x, digits = 4))
#34.56

3) nchar() :-

→ nchar() is used to display the number of characters in a string.

Eg :- ① x ← "aditya"
y ← nchar(x)
print(y)

② x ← "abcdef"
y ← nchar(x)
print(y)

4) tolower(), toupper()

Eg :- ① x ← "SMALL"
~~y ← tochar(~~
y ← tolower(x)
print(y)

② x ← "aditya"
y ← toupper(x)
print(y)

③ x ← "ADITYA"
y ← tolower(x)
print(y)

5) substring()

→ A substring() in R is used to extract a substring from the main string.

Syntax :- substring(value, 1st position, Last position)

Eg :- x ← "aditya Degree College"

y ← substring (x, 8, 13)

O/p :- Degree

print(y)

19/2/2020

Packages

Package is a collection of functions

→ Packages in R is a collection of relevant R functions which are stored under a directory called ~~dirss~~ "library" (or) "C:R/library"

→ By default, R installs only a few set of packages during installation

→ To know the location of where the R packages are installed, we use ".libPaths()" (gives path)

→ To display the list of all packages that are installed can be done using "library()".

→ To display all the packages that are currently loaded in the environment. is done by using "search()".

* Install packages manually:

a) To install a package (user defined or pre defined) manually, we use "install.packages()". In the above function we must give the compressed package (zip file) along with its location.

ex: install.packages("C:|R|RPack1.zip")

b) * Remove packages manually: To remove a package manually, we use "remove.packages()" → here no need of location.

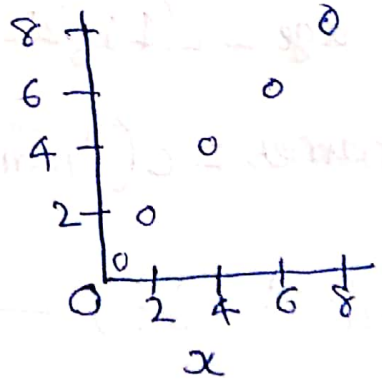
Ex: remove.packages("RPack1")

20/2/20

Graphics → we need x & y axis for 2D.

```
x = c(1,2,3,4,5,6,7,8,9)
y = c(1,2,3,4,5,6,7,8,9)
```

- plot(x,y)
(Ctrl+A)
(Ctrl+R)



plot(x, y, type="o") → plots will be connected using type2()

```
x = c(1,2,3,4,5,6,7,8,9)
y = c(11,53,15,25,39,28)
plot(x,y, type="o")
```

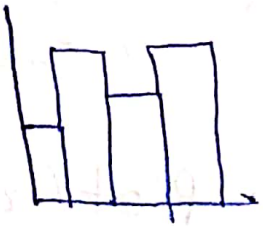
plot(x,y,type="o", pch=2 → pch will change plot symbol for every number

pch=2 → △

pch=3 → +

hist :-

hist(y) functionality of histogram



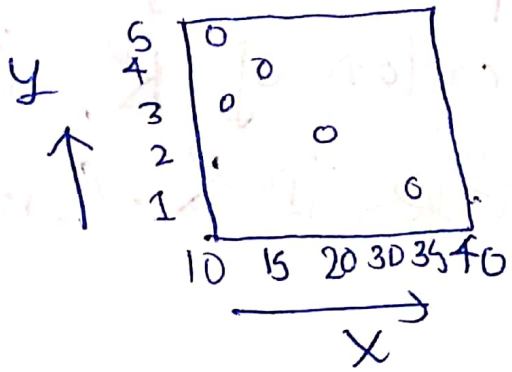
```
df ← data.frame(
    age = c(11,22,33,43,12)
names = c("ramu","John","Hari","Anil","Joe")
```

print(df) → o/p

age	names
11	ramu
22	John
33	Hari
⋮	⋮

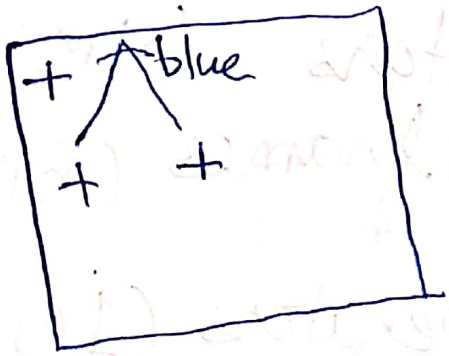
plot (df) → x, y age names

O/P :



plot (df, pch = 3, col = "blue")

O/P :-



change into names from points

points (25, 3, "Aditya") → O/P