**Different analytics**

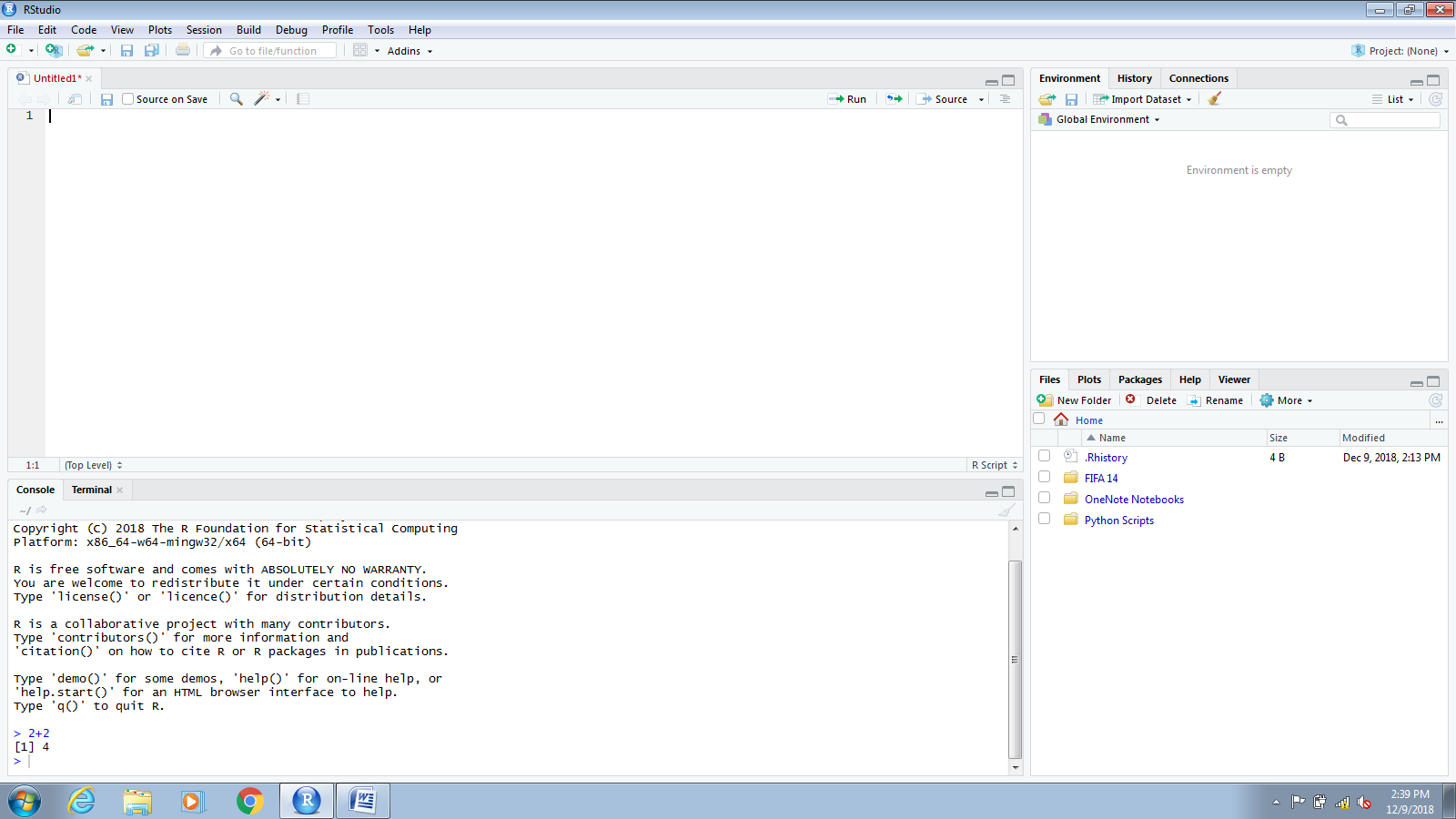
1. Descriptive - history, what has happened, AS-IS
2. Diagnosis - root cause, why it happened
3. Predictive- what is likely to happen, modelling, recomending
4. Prescriptive - Action on the recomendations provided
5. Congitive - Artificial intelligence

Different types of dataset we work on

Structured - Dataset with header, label, Its in the form of rows and columns

Semi structured - ( Jason, XML) Has the lable but can't be identified easily because of nesting

Unstructured- Free flowing text(Word, log)



Environment

Console





Script - Set of instructions

**In script**

If the code or statement has to be turned as message we use # (EX: ' in macrocs)

Whatever is written in console window cannot be saved.

Anything which is written in script can be saved.

Environment - Shows all the variables / inputs

Under environment

Data - Dataset

Values - Variables / inputs

History

It shows the history of the command which is executed

- It gets erased as soon as the session is closed

Connections

It can be connected to DB interface (spark, SAP etc) and use to the codes (Cloud database or server database)

Plots

This is the place where we see all the plots with data

Installing packages

Always check install dependencies

In the code (Console)

Install.packages("ggplot2")

To set the directory - setwd("E:/Acadgild/R")

to get directory - getwd()

**operators**

Airthematic

* + - Sum
* - Subratction
* multiplication
* / Division
* % remainder (Modulus)
* %/% Modulus integer

Logical

* And
* or
* Not

Relations

< , > , = , <= , >=

Assignments

**Data structures and types**

Charecter = "a"," abc"

Numeric = 1, 2, 3, 4

Integer = 1,2,3,4

logical = TRUE/FALSE

Complex = 1+4i

Vector = C["a", "b", 1,2,3]

> x=c(1,2,3,"a")

> class(x)

[1] "character"

>

> y=c(1,2,3)

> class(y)

[1] "numeric"

>

> z=c("a","b","c")

> class(z)

[1] "character"

>

> a=c("1","2","3")

> class(a)

[1] "character"

Where C= concatenate and defines what is the type of data present in the vector

> f=list("1","2","3")

> class(f)

[1] "list"

**Mathematical operations**

**Mean**

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

mean(x)

Ans

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

> mean(x)

[1] 5

**Minimum**

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

min(y)

Ans

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

> min(y)

[1] 1

**Maximum**

z=c(1,2,3,4,5,6,7,8,9)

max(z)

Ans

> z=c(1,2,3,4,5,6,7,8,9)

> max(z)

[1] 9

**Median**

a=c(1,2,3,4,5,6,7,8,9)

median(a)

Ans

> a=c(1,2,3,4,5,6,7,8,9)

> median(a)

[1] 5

**SUM**

b=c(1,2,3,4,5,6,7,8,9)

sum(b)

Ans

> b=c(1,2,3,4,5,6,7,8,9)

> sum(b)

[1] 45

**Range**

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

range(c)

Ans

> c=c(1,2,3,4,5,6,7,8,9)

> range(c)

[1] 1 9

**Sort**

d=c(4,5,7,3,2,1)

sort(d)

> d=c(4,5,7,3,2,1)

> sort(d)

[1] 1 2 3 4 5 7

**Rank**

d=c(4,5,7,3,2,1,9,14,15)

rank(d)

Ans

> d=c(4,5,7,3,2,1,9,14,15)

> rank(d)

[1] 4 5 6 3 2 1 7 8 9

d=c(4,5,7,3,2,1,9,14,15)

4 5 6 3 2 1 7 8 9 - It ranks the position on each string in the vector

e=c("adc","ef","rt","az")

rank(e)

> e=c("adc","ef","rt","az")

> rank(e)

[1] 1 3 4 2

("adc","ef","rt","az")

1 3 4 2 - It ranks the position on each string in the vector

**Matrix**

matrix(1:20,nrow = 4, ncol = 5)

Ans

[,1] [,2] [,3] [,4] [,5]

[1,] 1 5 9 13 17

[2,] 2 6 10 14 18

[3,] 3 7 11 15 19

[4,] 4 8 12 16 20

matrix(1:20,nrow = 4, ncol = 5, byrow = TRUE)

matrix(1:20,nrow = 4, ncol = 5, byrow = TRUE)

[,1] [,2] [,3] [,4] [,5]

[1,] 1 2 3 4 5

[2,] 6 7 8 9 10

[3,] 11 12 13 14 15

[4,] 16 17 18 19 20

If by row is given as true, then the values are populated in rows

**Column Bind**

x=1:5

y=4:12

cbind(x,y)

Ans

> x=1:5

> y=4:8

> cbind(x,y)

x y

[1,] 1 4

[2,] 2 5

[3,] 3 6

[4,] 4 7

[5,] 5 8

x=1:5

y=4:12

cbind(x,y)

> x=1:5

> y=4:12

> cbind(x,y)

x y

[1,] 1 4

[2,] 2 5

[3,] 3 6

[4,] 4 7

[5,] 5 8

[6,] 1 9

[7,] 2 10

[8,] 3 11

[9,] 4 12

Warning message:

In cbind(x, y) :

number of rows of result is not a multiple of vector length (arg 1)

**Row bind**

x=1:5

y=4:8

rbind(x,y)

Ans

> x=1:5

> y=4:8

> rbind(x,y)

[,1] [,2] [,3] [,4] [,5]

x 1 2 3 4 5

y 4 5 6 7 8

**Change of dimension**

y=4:12

dim(y)=c(3,3)

y

Ans

> y=4:12

> dim(y)=c(3,3)

> y

[,1] [,2] [,3]

[1,] 4 7 10

[2,] 5 8 11

[3,] 6 9 12

**Naming rows and columns**

x=matrix(1:20,nrow = 4, ncol = 5, byrow = TRUE,dimnames = list(c("r1","r2","r3","r4"),c("c1","c2","c3","c4","c5")))

x

Ans

> x=matrix(1:20,nrow = 4, ncol = 5, byrow = TRUE,dimnames = list(c("r1","r2","r3","r4"),c("c1","c2","c3","c4","c5")))

> x

c1 c2 c3 c4 c5

r1 1 2 3 4 5

r2 6 7 8 9 10

r3 11 12 13 14 15

r4 16 17 18 19 20

**Error: unexpected symbol in: brackets missing**

Data frames

It is called as special form of matrix

f=1:4

y=5:8

z=data.frame(f,y)

z

Ans

f=1:4

> y=5:8

> z=data.frame(f,y)

> z

f y

1 1 5

2 2 6

3 3 7

4 4 8

f=1:4

y=5:8

z=data.frame(f,y)

z

head(z)

> f=1:4

> y=5:8

> z=data.frame(f,y)

> z

f y

1 1 5

2 2 6

3 3 7

4 4 8

> head(z)

f y

1 1 5

2 2 6

3 3 7

4 4 8

f=1:4

y=c("dog","cat","ball","apple")

z=data.frame(f,y)

z

str(z)

Ans

> f=1:4

> y=c("dog","cat","ball","apple")

> z=data.frame(f,y)

> z

f y

1 1 dog

2 2 cat

3 3 ball

4 4 apple

>

> str(z)

'data.frame': 4 obs. of 2 variables:

$ f: int 1 2 3 4

$ y: Factor w/ 4 levels "apple","ball",..: 4 3 2 1

**AS factor**

f=1:4

y=c("ball","cat","dog","apple")

z=data.frame(f,y)

z

sort(y)

str(z)

as.factor(y)

ans

> f=1:4

> y=c("ball","cat","dog","apple")

> z=data.frame(f,y)

> z

f y

1 1 ball

2 2 cat

3 3 dog

4 4 apple

> sort(y)

[1] "apple" "ball" "cat" "dog"

> str(z)

'data.frame': 4 obs. of 2 variables:

$ f: int 1 2 3 4

$ y: Factor w/ 4 levels "apple","ball",..: 2 3 4 1

> as.factor(y)

[1] ball cat dog apple

Levels: apple ball cat dog

x=c(5,6,7,8,9)

x[c(1,2)] - Selecting(Keep)

x[-c(1,2)] - Not selecting(Drop)

x[c(TRUE,TRUE,FALSE,FALSE)] - Logical

x[x>5] - Conditional

> x=c(5,6,7,8,9)

> x[c(1,2)]

[1] 5 6 - It is displaying the first two values of the string

> x[-c(1,2)]

[1] 7 8 9 - It is displaying the other values than what we have described

> x[c(TRUE,TRUE,FALSE,FALSE)]

[1] 5 6 9 - It displays on the values in the position of true

> x[x>5]

[1] 6 7 8 9 - It displays the values as per the condition

**Read CSV**

read.csv(file = "E:/Acadgild/R/Book1.csv" ,header = TRUE,sep = ",", quote = "\"")

File = "file location"

Header - True(First row is header), False(No header is considered)

Sep = Delimiter

Quote = If any cell has comma in it the it will consider that whole cell as one

Fill = Ture (Blank cells shows as NA), False(Vlank cells will be blank)

x=read.csv(file = "E:/Acadgild/R/Book1.csv" ,header = TRUE,sep = ",", quote = "\"")

x

x[c("a","d")] - TO keep only column a & d

x[,-c(2)] - TO remove column 2

x[c(TRUE,FALSE,TRUE,FALSE)] Based on logic