Introduction to Analytics

UNIT-I: Introduction to Analytics and R programming

- Introduction to R
- RStudio (GUI):
- R Windows Environment
- Introduction to various data types, Numeric, Character ,date, data frame, array, matrix etc.,
- Reading Datasets, Working with different file types .txt,.csv etc.
- Outliers,
- Combining Datasets,
- R Functions and loops.
-

1. Introduction to R

What is R?

R is a flexible and powerful open-source implementation of the language S (for statistics) developed by John Chambers and others at Bell Labs.

Why R?

Five reasons to learn and use R:

- ✓ R is open source and completely free. R community members regularly contribute packages to increase R's functionality.
- ✓ R is as good as commercially available statistical packages like SPSS, SAS, and Minitab.
- ✓ R has extensive statistical and graphing capabilities. R provides hundreds of built-in statistical functions as well as its own built-in programming language.
- ✓ R is used in teaching and performing computational statistics. It is the language of choice for many academics who teach computational statistics.
- ✓ Getting help from the R user community is easy. There are readily available online tutorials, data sets, and discussion forums about R.

R uses:

- ✓ R combines aspects of functional and object-oriented programming.
- ✓ R can use in interactive mode
- ✓ It is an interpreted language rather than a compiled one.
- ✓ Finding and fixing mistakes is typically much easier in R than in many other languages.

R Features:-

- ✓ •Programming language for graphics and statistical computations
- ✓ •Available freely under the GNU public license
- ✓ •Used in data mining and statistical analysis
- ✓ •Included time series analysis, linear and nonlinear modeling among others
- ✓ •Very active community and package contributions
- ✓ •Very little programming language knowledge necessary
- ✓ •Can be downloaded from **http://www.r-project.org/**

What is CRAN?

CRAN abbreviates **Comprehensive R Archive Network** will provide binary files and follow the installation instructions and accepting all defaults. Download from *http://cran.r-project.org/* we can see the R Console window will be in the RGui (graphical user interface). Fig 1 is the sample

R GUI.

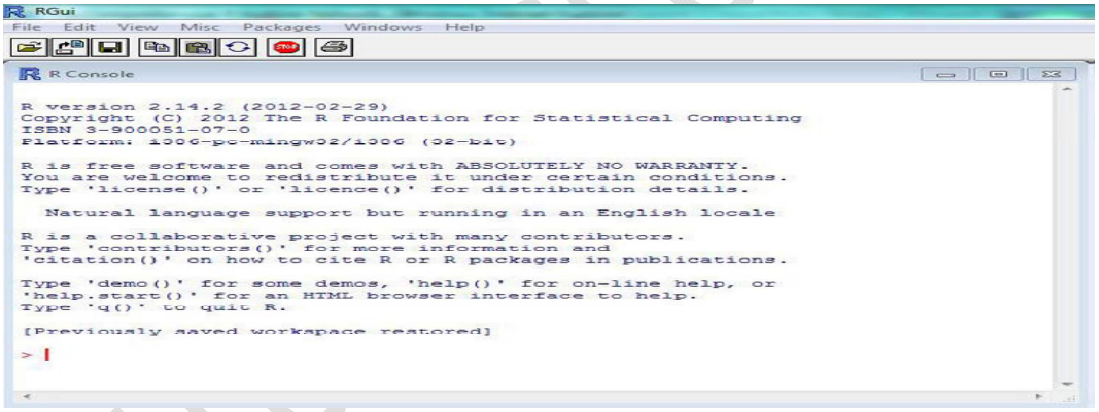


Fig 1: R console

R Studio:

R studio is an IDE for R with advanced and more user-friendly GUI.

Using R as calculator:

R as a calculator, typing commands directly into the R Console. Launch R and type the following code, pressing < Enter > after each command. Technically, everything the user types are an *expression*. R allows 3 assignment operations as: <- or = for assignment and == to test equality.

Example Exercise1:

> 2 * 2 ## Multiplication

[1] 4

> 2 / 2 ## Division

[1] 1

> 2 + 2 ## addition

[1] 4

> 2 – 2 ## subtraction

[1] 0

> 2 ^ 2 ## exponentiation

[1] 4

> q() ## to quit

> **y <- 3*exp(x)**

> **x <- 3*exp(x)**

 # Declare variables of different types:

my_numeric <- 42

my_character <- "forty-two"

my_logical <- FALSE

 # Check which type these variables have:

R expression:

At > prompt type the R expression and press enter

R output:

R labels each output value with a number in square brackets. As far as R is concerned, an individual number is a one-element vector. The [1] is simply the index of the first element of the vector.

Activity1: Calculate the following using R:

1. Log of 2

2. 2^3 X 3^2

3. e^3

Variable (Object) Names:

Certain variable names are reserved for particular purposes. Some reserved symbols are: **c q t C D F I T**

meaning of c q t C D F I T

? ## to see help document

?c ## c means Combine Values into a Vector or List

?q ## q means Terminate an R Session

?t ## t means Matrix Transpose

?C ## C means sets contrast for a factor

?D ## D means Symbolic and Algorithmic Derivatives of Simple Expressions

?F ## F means logical vector Character strings

c("T", "TRUE", "True", "true") are regarded as true, c("F", "FALSE", "False", "false") as false, and all others as NA.

>F ##[1] FALSE

?I ##Inhibit Interpretation/Conversion of Objects

Getting Help in R:

help(function), or use the ?function shortcut. Either of these will open the R documentation, which is always a good place to start when you have questions. To see a listing of the objects in your workspace, you can use the **ls()** function. To get more detail, use **ls.str()**

> ls()

[1] "A" "acctdata" "address" "B" "b1"

[6] "balance" "c" "CareerSat" "chisquare" "colnames"

Components of R:

Data Types:

There are two types of data classified on very broad level. They are Numeric and Character data.

- Numeric Data: - It includes 0~9, "." and "- ve" sign.
- Character Data: - Everything except Numeric data type is Character. For Example, Names, Gender etc.

Data in analytics is also classified as Quantitative and Qualitative.

For Example, "1,2,3…" are Quantitative Data while "Good", "Bad" etc. are Qualitative Data.

We can convert Qualitative Data into Quantitative Data using Ordinal Values.

For Example, "Good" can be rated as 9 while "Average" can be rated as 5 and "Bad" can be rated as 0.

Data Type	Example	Verify
Logical	TRUE , FALSE	v <- TRUE print(class(v)) [1] "logical"
Numeric	12.3, 5, 999	v <- 23.5 print(class(v)) [1] "numeric"
Integer	2L, 34L, 0L	v <- 2L print(class(v)) [1] "integer"
Complex	3 + 2i	v <- 2+5i print(class(v)) [1] "complex"

Character	'a' , '"good", "TRUE", '23.4'	v <- "TRUE" print(class(v)) [1] "character"
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <- charToRaw("Hello") print(class(v)) [1] "raw"

mode() or Class() is used to know the type of data type assigned.

Assign several different objects to x, and check the mode (storage class) of each object.

o/p:

Declare variables of different types:

> my_numeric <- 42

> my_character <- "forty-two"

> my_logical <- FALSE

> # Check which type these variables have:

> class(my_numeric)

[1] "numeric"

> class(my_character)

[1] "character"

> class(my_logical)

[1] "logical"

> x <- 2

> x

[1] 2

> x ^ x

[1] 4

> x ^ 2

[1] 4

> mode(x) ## will returen the storage class of object

[1] "numeric"

```
>  seq(1:10)   ## will create a vector of 1 to sequence numbers
[1] 1 2 3 4 5 6 7 8 9 10
>  x <- c(1:10)   ##vector of 1 to 10 digits
>  x
 [1]  1 2 3 4 5 6 7 8 9 10
> mode(x)
[1] "numeric"
> x <- c("Hello","world","!")
> mode(x)
[1] "character"
> x <- c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE)
> mode(x)
[1] "logical"
> x <- list("R","12345",FALSE)
> x
[[1]]
[1] "R"
[[2]]
[1] "12345"
[[3]]
[1] FALSE
> mode(x)
[1] "list"
```

Vectors:

Vector is the most common data structure in R. Vectors must be homogeneous i.e, the type of data in a given vector must all be the same. Vectors can be numeric, logical, or character. If a vector is mix data types then R forces (**coerces**, if you will) the data into one mode.

Creating a vector:

To create a vector , “concatenate” a list of numbers together to form a vector.

```
x < - c(1, 6, 4, 10, -2)   ## c() to concatenate elements
```

my.vector<- 1:24 ## a numeric vector with 1 to 24 numbers

List of built-in functions to get useful summaries on vectors:

Example1:

> sum(x) ## sums the values in the vector

> length(x) ## produces the number of values in the vector, ie its length

> mean(x) ## the average (mean)

> var(x) ## the sample variance of the values in the vector

> sd(x) ## the sample standard deviation of the values in the vector (square root of the sample variance)

> max(x) ## the largest value in the vector

> min(x) ## the smallest number in the vector

> median(x) ## the sample median

> y < - sort(x) ## the values arranged in ascending order

Example2:

linkedin <- c(16, 9, 13, 5, 2, 17, 14)

> last <- tail(linkedin, 1)

> last

[1] 14

> # Is last under 5 or above 10?

> # Is last between 15 (exclusive) and 20 (inclusive)?

> # Is last between 0 and 5 or between 10 and 15?

> (last > 0 | last < 5)

[1] TRUE

> (last > 0 & last < 5)

[1] FALSE

> (last > 10 & last < 15)

[1] TRUE

Example3: Following are some other possibilities to create vectors

> x <- 1:10

> y <- seq(10) #Create a sequence

> z <- rep(1,10) #Create a repetitive pattern

> x

[1] 1 2 3 4 5 6 7 8 9 10

> y

[1] 1 2 3 4 5 6 7 8 9 10

> z

[1] 1 1 1 1 1 1 1 1 1 1

Adding elements to vector:

> x <- c(x, 11:15)

>x

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Vector Arithmetic:

> x <- c(1:10)

> x

[1] 1 2 3 4 5 6 7 8 9 10

> y <- 10

> x + y

[1] 11 12 13 14 15 16 17 18 19 20

> 2 + 3 * x #Note the order of operations

[1] 5 8 11 14 17 20 23 26 29 32

> (2 + 3) * x #See the difference

[1] 5 10 15 20 25 30 35 40 45 50

> sqrt(x) #Square roots

[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427

[9] 3.000000 3.162278

> x %% 4 #This is the integer divide (modulo) operation

[1] 1 2 3 0 1 2 3 0 1 2

> y <- 3 + 2i #R does complex numbers

> Re(y) #The real part of the complex number

[1] 3

> Im(y) #The imaginary part of the complex number

[1] 2

> x * y

[1] 3+ 2i 6+ 4i 9+ 6i 12+ 8i 15 + 10i 18 + 12i 21 + 14i 24 + 16i 27 + 18i 30 + 20i

Matrices:

A matrix is a two-dimensional rectangular data set. Matrices must be heterogeneous i.e, the type of data in a given vector of all not in the same class.

Matrix can be created using :

- matrix(): A vector input to the matrix function.
- Using rbind() and cbind() functions.
- Using dim() to the existing vector

Creating a matrix using matrix():

```
# Create a matrix.

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"
```

Creating a matrix using rbind() or cbind():

First crate two vectors and then create a matrix using rbind() .It binds the two vectors data into two rows of matrix.

Example: To create a matrix havind the data as: 6,2 ,10 & 1, 3, -2

Step1:create two vectors as xr1,xr2

```
> xr1 <- c( 6, 2, 10)
> xr2 <- c(1, 3, -2)
> x <- rbind (xr1, xr2) ## binds the vectors into rows of a matrix(2X3)
> x
    [,1] [,2] [,3]
xr1    6    2   10
xr2    1    3   -2
```

First crate two vectors and then create a matrix using rbind() .It binds the two vectors data into two rows of matrix.

Example: To create a matrix havind the data as: 6,2 ,10 & 1, 3, -2

create two vectors as xr1,xr2

```
> y <- cbind(xr1, xr2) ## binds the same vectors into columns of a matrix(3X2)
> y
    xr1 xr2
[1,]   6   1
[2,]   2   3
[3,]  10  -2
```

Matrix operations:

```
> A <- matrix(c( 6, 1, 0, -3, -1, 2),3, 2, byrow = TRUE)
> B <- matrix(c( 4, 2, 0, 1, -5, -1),3, 2, byrow = TRUE)
```

>A (with output)

>B (with output)

```
> A + B
     [,1]  [,2]
[1,] 10    3
[2,]  0    -2
[3,]  -6    1
> A - B
      [,1] [,2]
[1,]  2   -1
[2,]  0   -4
[3,]  4   3
> A * B # this is component-by-component multiplication, not matrix multiplication
     [,1] [,2]
[1,]  24  2
[2,]  0  -3
[3,]  5  -2
> t(A) ## Transpose of a matrix
[,1] [,2] [,3]
```

[1,] 6 0 -1

[2,] 1 -3 2

Alternative method to create a matrix using dim():

Create a vector and add the dimensions using the **dim ( )** function.It's especially useful if you have your data already in a vector.

Example: A vector with the numbers 1 through 24, like this:

>my.vector<- 1:24

You can easily convert that vector to an array exactly like my.array simply by assigning the dimensions, like this:

> dim(my.vector) <- c(3,4)

Array:

Arrays can be of any number of dimensions. The array function takes a **dim** attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

Creating an array:

>my.array< - array(1:24, dim=c(3,4,2))

In the above example, "my.array" is the name of the array we have given. There are 24 units in this array mentioned as "1:24" and are divided in three dimensions "(3, 4, 2)".

Alternative: with existing vector and using dim()

> my.vector<- 1:24

To convert my.vector vector to an array exactly like my.array simply by assigning the dimensions, like this:

> dim(my.vector) <- c(3,4,2)

Activity 2:Create an Array with name “MySales” with 30 observations using following methods:

1. Defining the dimensions of the array as 3, 5 and 2.
2. By using Vector method.

Lists:

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

```
>list1 <- list(c(2,5,3),21.3,sin)   # Create a list.
>print(list1)     # Print the list.
 [[1]]
[1] 2 5 3
[[2]]
[1] 21.3
[[3]]
function (x)  .Primitive("sin")
```

Data Frames:

Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length. Data Frames are created using the data.frame() function. It displays data along with header information.

To retrieve data in a particular cell: Enter its row and column coordinates in the single square bracket "[ ]" operator.

Example: To retrieve the cell value from the first row, second column of mtcars.

```
>mtcars[1,2]
mtcars[row,column]
# Create the data frame.
>BMI <- data.frame(gender = c("Male", "Male","Female"),          height = c(152, 171.5, 165),
weight = c(81,93, 78),Age =c(42,38,26))
```

>print(BMI)

	gender	height	weight	Age
1	Male	152.0	81	42
2	Male	171.5	93	38
3	Female	165.0	78	26

Data1 : Height GPA

66	3.80
62	3.78
63	3.88
70	3.72
74	3.69

> student.ht <- c(66, 62, 63, 70, 74)

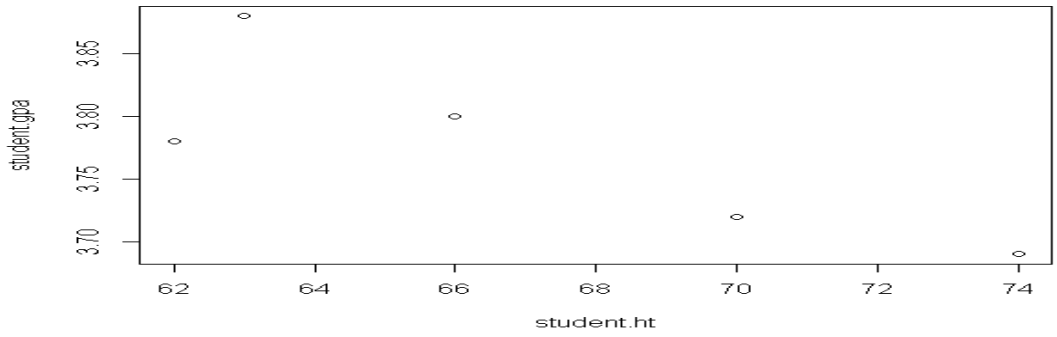
> student.gpa < - c(3.80, 3.78, 3.88, 3.72, 3.69)

> student.data1 < - data.frame(student.ht, student.gpa)

> student.data

	student.ht	student.gpa
1	66	3.80
2	62	3.78
3	63	3.88
4	70	3.72
5	74	3.69

> plot(student.ht, student.gpa)



Reading Database using R

We can import Datasets from various sources having various file types :

Example:

- **.csv format**
- **Big data tool – Impala**
- **CSV File**

The sample data can also be in comma separated values (CSV) format. Each cell inside such data file is separated by a special character, which usually is a comma, although other characters can be used as well.The first row of the data file should contain the column names instead of the actual data. Here is a sample of the expected format.

Col1,Col2,Col3

100,a1,b1

200,a2,b2

300,a3,b3

After we copy and paste the data above in a file named "mydata.csv" with a text editor, we can read the data with the function read.csv.

> mydata = read.csv("mydata.csv") # read csv file

> mydata

Date in R:

Sys.Date() : The current system date.

Notice that this function returns a Date object.

class(Date)

A string in this format is treated as a character unless cast to a Date type.

class(“2010-06-16″)

class(as.Date(“2010-06-16″))

You can also pass in dates in other formats and cast them as strings by specifying the format in use.

as.Date('02/03/2004′,'%m/%d/%Y')

To format date information in a wide variety of string formats, use the strftime function.

strftime(Sys.Date(),'%A: %B %d, %Y (Day %j of %Y)')

This returns the string "Tuesday: June 15, 2010 (Day 166 of 2010)"

Dates can be manipulated arithmetically.

To return the next ten days…

seq(Sys.Date(),Sys.Date() + 10,1)

…or the last ten days…

seq(Sys.Date(),Sys.Date() – 10,-1)

Introduction to Analytics

UNIT-I: Lab Activity

1) R as calculator:

> 4+6

Use ***** for multiply.

Use ^ for raised to the power of.

Use parentheses to ensure that it understands what you are trying to compute.

The order of doing arithmetic operations is (left [done first] to right [done last])

^ / * - +

Built-in Functions:

pi

exp(3) ## provides the cube of e

log(1.4) ## provides the natural logarithm of the number 1.4

log10(1.4) ## provides the log to the base of 10

sqrt(16) ## provides the square root of 16

2) Assignment Statements:

Just like in algebra, we often want to store a computation under some variable name.

The result is assigned to a variable with the symbols <- which is formed by the "less than" symbol followed immediately by a hyphen.

> **x <- 2.5**

When you want to know what is in a variable simply ask by typing the variable name.

> **x**

We can store a computation under a new variable name or change the current value in an old variable.

> **y <- 3*exp(x)**

> **x <- 3*exp(x)**

 # Declare variables of different types:

```
my_numeric <- 42
my_character <- "forty-two"
my_logical <- FALSE
```
 # Check which type these variables have:
```
class(my_numeric)
class(my_character)
class(my_logical)
```
o/p:
```
# Declare variables of different types:
> my_numeric <- 42
> my_character <- "forty-two"
> my_logical <- FALSE
> # Check which type these variables have:
> class(my_numeric)
[1] "numeric"
> class(my_character)
[1] "character"
> class(my_logical)
[1] "logical"
```
Creating a Vector:

To create a vector we “concatenate” a list of numbers together to form a vector.

```
> x < - c(1, 6, 4, 10, -2)
```

Once we have a vector of numbers we can apply certain built-in functions to them to get useful s ummaries.

Example1:
```
> sum(x) ## sums the values in the vector
> length(x) ## produces the number of values in the vector, ie its length
> mean(x) ## the average (mean)
> var(x) ## the sample variance of the values in the vector (has n-1 in denominator)
> sd(x) ## the sample standard deviation of the values in the vector (square root of the sample va
riance)
> max(x) ## the largest value in the vector
> min(x) ## the smallest number in the vector
> median(x) ## the sample median
> y < - sort(x) ## the values arranged in ascending order
```

Example2:
```
linkedin <- c(16, 9, 13, 5, 2, 17, 14)
> last <- tail(linkedin, 1)
```

```
> last
[1] 14
```

> # Is last under 5 or above 10?
> # Is last between 15 (exclusive) and 20 (inclusive)?
> # Is last between 0 and 5 or between 10 and 15?

```
> (last > 0 | last < 5)
[1] TRUE
> (last > 0 & last < 5)
[1] FALSE
> (last > 10 & last < 15)
[1] TRUE
```

Variable (Object) Names:

Certain variable names are reserved for particular purposes. Some reserved symbols are: **c q t C D F I T**

Creating a matrix:

matrix data as: 6,2 ,10 & 1, 3, -2

You could form the rows and then bind them together to form the matrix.

```
> xr1 <- c( 6, 2, 10)
> xr2 <- c(1, 3, -2)
> x <- rbind(xr1, xr2) ## binds the vectors into rows of a matrix
> x
    [,1] [,2] [,3]
xr1    6    2   10
xr2    1    3   -2
> y <- cbind(xr1, xr2) ## binds the same vectors into columns of a matrix
> y
     xr1 xr2
[1,]   6   1
[2,]   2   3
[3,]  10  -2
> t(x) ## this is the transpose of the matrix x
     xr1 xr2
[1,]   6   1
[2,]   2   3
[3,]  10  -2
> x
    [,1] [,2] [,3]
xr1    6    2   10
xr2    1    3   -2
```

```
> x[2,1]##The elements in a matrix are indexed by their (Row, Column) positions
xr2
 1
```

Creating a Data Frame:

```
Data :  Height GPA
        66     3.80
        62     3.78
        63     3.88
        70     3.72
        74     3.69
> student.ht <- c( 66, 62, 63, 70, 74)
> student.gpa < - c( 3.80, 3.78, 3.88, 3.72, 3.69)
> student.data < - data.frame(student.ht, student.gpa)
> student.data
> plot(student.ht, student.gpa)
```

Descriptive:

1. What is R?
2. What are the R features while comparing with other programming languages?
3. How to set up the R environment?
4. Explain R as calculator using basic operations and inbuilt functions with suitable example?
5. Write expression in R

   a)e^4+log2 b)2^4X5^3 c)$\log_2 10$ c)$\log_{10} 2$
6. What are the different forms of data types and how to test the data type in R ? Give one example for each
7. What are different basic components in R?
8. Explain how to create a vector and characteristics of vectors
9. What are the inbuilt functions to get summaries on vector?
10. Give the list of reserved variables in R and explain the functionalities of each.
11. How to create a array in multiple dimensions give suitable example
12. How many ways we can create a matrix in R explain with example
13. **Create an Array with name "MySales" with 30 observations using following methods:**

    a) By using the array with dimensions 3, 5 and 2.

    b) By using Vector method.
14. What is data frame and how to create a data frame using the following data:

Height	GPA
66	3.80
62	3.78
63	3.88
70	3.72
74	3.69

15. How to get system date in R?
16. Generate sequence of previous and coming 10 dates.
17. What is the R function to get system date and time
18. Explain following.
19. a)vector b)data frame c)matrix d)list
20. Write the differences between
21. vector() and list()
22. what are the different ways to read the dataset?
23. How to create and rename a variable in R?
24. What are the read write methods available in R and explain?
25. What is the difference between data frames and matrices? Give examples
26. How will you identify & treat the following in R.?
27. a) Missing value b) Outliers
28. How to combine the two datasets in R?
29. Explain control functions in R with suitable example
30. Explain control loops in R with suitable example
31. Write the Rscript for the following:
32. Import the csv file studentmarks.txt having sno,sem_no,m1.m2,m3,m4,m5,m6,L1,L2 using R
33. Export the Data Frame Smarks to CSV files .
34. Create a new variable Totalmarks and avgmarks with sum and avg of m1,m2,m3,m4,m5,m6.
35. Check for Outliers in the studentmarks data data frame.
36. Create a new data set by combining 2 data sets.
37. vi)Check for missing data in studentmarks and remove missing data.
38. Differences between rbind and cbind

Objectives:

1.) If I have a data.frame df <- data.frame(a = c(1, 2, 3), b = c(4, 5, 6), c(7, 8, 9))

1a.) How do I select the c(4, 5, 6)?
1b.) How do I select the 1?
1c.) How do I select the 5?
1d.) What is df[, 3]?
1e.) What is df[1,]?
1f.) What is df[2, 2]?

2 a.) If I concatenate a number and a character together, what will the class of the resulting vector be?

2b.) What if I concatenate a number and a logical?

2c.) What if I concatenate a number and NA?

3.) What is the output of seq(4) ?

4) The R language is a dialect of which of the following programming languages?

a) c b)java c) statistical d)pearl

5) From the following choose atomic data type in R

a) logical b)numeric c)integer d)all

6) If x <- 4 in R, what is the class of the object `x' ?

7) What is the class of the object x defined by x <- c(4, TRUE)?

8) If I have two vectors x <- c(1,3, 5) and y <- c(3, 2, 10), what is size of the expression rbind(x, y) output?

8) _____is the key property of vectors in R

9) If x is defined as x <- list(2, "a", "b", TRUE). What does x[[1]] ?

10) If x is defined as x <- list(2, "a", "b", TRUE). What does x[[2]] give me?

11) if x ,y are two vectors x <- 1:4 and y <- 2:3. What is produced by the expression x + y?

12) if x ,y are two vectors, x <- 1:4 and a vector y <- 2. What is produced by the expression x + y?

13 If a vector x <- c(17, 14, 4, 5, 13, 12, 10) and I want to set all elements of this vector that are greater than 10 to be equal to 4. What R code achieves this?

14 _____function is used to know the column names of the dataset?

15 _____is used to extract the first 2 rows of the iris dataset.

16 _____is used to know the no of observations are in this data frame

17 mtcars[3,3] will produce _____on console.

18 _____is used to know the value of 23^{rd} row and 2^{nd} coloumn value in iris dataframe.

19 _____is used to know missing values in dataset

20 _____is used to know the undefined values in dataset

21 _____is used to take the mean of coloumn sepal_length having missed values

22 _____is used to create subset of data having sepal_lenth>3

23 _____ is used to combine two datasets horizontally

24 _____is used to know the maximum value of vector x<-c(2,4,9,10)