Program Name :

Page No. 01

Date :

R programming

Introduction :-

→ R language is a open source program used for performing statistical operations

→ It is a command line driven program and execute one command at a time.

→ It performs statistical operations and generates R data analysis reports in graphical or text formats.

History of R Language :-

→ John chambers and colleagues developed R at bell laboratories.

→ R is an implementation of the S programming language and combines with lexical scoping semantics inspired by Scheme.

R-Studio :-

→ R studio is an integrated development environment (IDE) for R language. R studio is a code editor and development environment, with some nice features that make code development in R easy and fun.

features of R-studio:-

→ code highlighting
→ Automatic bracket matching
→ code completion
→ It is free to Use.

components of R-studio:

1) Source = Multiple lines of code can be entered here.

2) Console = All the interactive work of R programming is performed in this window.

3) Files = where the user can browse folders and files.

4) plots = where R displays the User's plots.

5) Workspace and History = shows overview of workspace, history of the commands.

6) Packages = This is where the user can view a list of all the installed packages.

7) Help = This is where you can browse the built-in Help system of R

→

	X	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	1	5.1	3.5	1.4	0.2	setosa
2	2	4.9	3.0	1.4	0.2	setosa
3	3	4.7	3.2	1.3	0.2	setosa
⋮						
149	149	6.2	3.4	5.4	2.3	virginica
150	150	5.9	3.0	5.1	1.8	virginica

→ "x" "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "species"

→ 'data.frame' : 150 obs of 6 variables

```
$x            : int    1   2   3   4 .....
$Sepal.Length : num    5.1 4.9 4.7 4.6 .....
$Sepal.Width  : num    3.5 3.3 3.2 3.1 .....
$Petal.length : num    1.4 1.4 1.3 1.5 .....
$petal.width  : num    0.2 0.2 0.2 0.2 .....
$species      : Factor w/3 levels "setosa","versicolor",...: 1 1 1 1 .....
```

→ [1] 4.3

→ [1] 7.9

→ [1] 4.3 7.9

→ [1] 5.843333

1) Load the 'iris.csv' file and display the names and type of each column. Find Statistics such as min, max, range, mean, median, variance, standard deviation for each column of data.

```
>iris<-read.csv("c:/Users/Admin/Desktop/iris.csv",
                header=TRUE, sep=",")
>print(iris)

>colnames(iris)

>str(iris)

>print(min(iris$Sepal.Length))

>print(max(iris$Sepal.Length))

>print(range(iris$Sepal.Length))

>print(mean(iris$Sepal.Length))
```

→ [1] 5.8

→ [1] 0.6856935

→ [1] 0.8280661

→ [7] 2

→ [1] 4.4

→ [1] 2.0 4.4

→ [1] 3.057333

→ [1] 3

→ [1] 0.1899794

→ [1] 0.4358663

```
> print(median (iris $ Sepal.Length))

>print(var (iris $ Sepal.Length))

>print( sd(iris $Sepal.Length))

>print(min (iris $Sepal.Width))

>print(max (iris $Sepal.Width))

>print(range (iris $Sepal.Width))

>print(mean (iris $Sepal.Width))

>print(median (iris $Sepal.Width))

>print( var(iris $Sepal.Width))

>print( sd(iris $Sepal.Width))
```

⇒[1] 1

→[1] 6.9

→[1] 1.0 6.9

→[1] 3.758

→[1] 4.35

→[1] 3.116278

→[1] 1.765298

→[1] 0.1

→[1] 2.5

→[1] 0.1 2.5

→[1] 1.199333

→[1] 1.3

→[1] 0.58100063

→[1] 0.7622377

```
>print(min(iris$Petal.Length))
>print(max(iris$Petal.Length))
>print(range(iris$Petal.Length))
>print(mean(iris$Petal.Length))
>print(median(iris$Petal.Length))
>print(var(iris$Petal.Length))
>print(sd(iris$Petal.Length))
>print(min(iris$Petal.Width))
>print(max(iris$Petal.Width))
>print(range(iris$Petal.Width))
>print(mean(iris$Petal.Width))
>print(median(iris$Petal.Width))
>print(var(iris$Petal.Width))
>print(sd(iris$Petal.Width))
```

→

[1] 0.22222222 0.16666667 0.11111111 0.08333333
[5] 0.19444444 0.30555556 0.08333333 0.19444444
[9] 0.02777778 0.16666667 0.30555556 0.13888889
[13] 0.00000000 0.41666666 0.38888889 0.30555556

.....

→

[0] 0.62500000 0.41666667 0.50000000 0.45833333
[5] 0.66666667 0.79166667 0.58333333 0.58333333
[9] 0.37500000 0.45833333 0.70833333 0.41666667
[13] 0.41666667 0.83333333 1.00000000 0.79166667

.....

2) Write R program to normalize the variables into 0 to 1 scale using min-max normalisation.

min-max normalizations:=

Min-max normalization is a normalization strategy which linearly transforms x to y

$y = (x - min)/(max - min)$, where min and max are the minimum and maximum values in x, where x is the set of observed values of x. It can be easily seen that when x=min then y=0 and when x=max, then y=1

This technique is traditionally used with K-Nearest Neighbours (KNN) classifications problems.

```
> x1 <- iris $ Sepal.Length
> Normalized = (x1 - min(x1)) / (max(x1) - min(x1))
> print (normalized)

> x2 <- iris $ Sepal.Width
> normalized = (x2 - min(x2)) / (max(x2) - min(x2))
> print (normalized)
```

→
[1] 0.06779661 0.06779661 0.05084746 0.08474576
[5] 0.06779661 0.11864407 0.06779661 0.08474576
[9] 0.06779661 0.08474576 0.08474576 0.10169492
[13] 0.06779661 0.01694915 0.03389831 0.08474576

→
[1] 0.04166667 0.04166667 0.04166667 0.04166667
[5] 0.12500000 0.08333333 0.04166667 0.04166667
[9] 0.00000000 0.04166667 0.04166667 0.00000000
[13] 0.41667 0.12500000 0.12500000 0.08333333

```
> x3 <- iris$Petal.Length
> normalized = (x3 - min(x3)) / (max(x3) - min(x3))
> print(normalized)
```

```
> x4 <- iris$Petal.Width
> normalized = (x4 - min(x4)) / (max(x4) - min(x3))
> print(normalized)
```

Program Name : 03

Page No. 08

Date :

3) Generate histograms for any one variable (Sepallength / Sepal Width / petal Length / petal Width) and generate scatter plots for every pair of variables showing each species in different color.

Histogram:-

A Histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to barchat but the difference is to groups the values into continuous ranges.

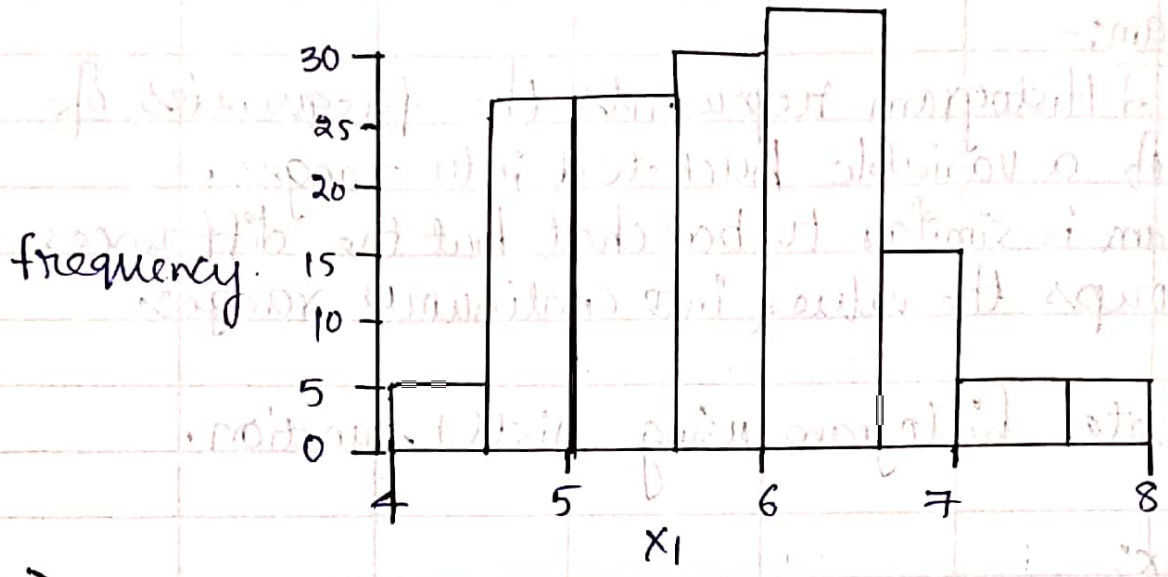
R creates histogram using hist() function.

Syntax:

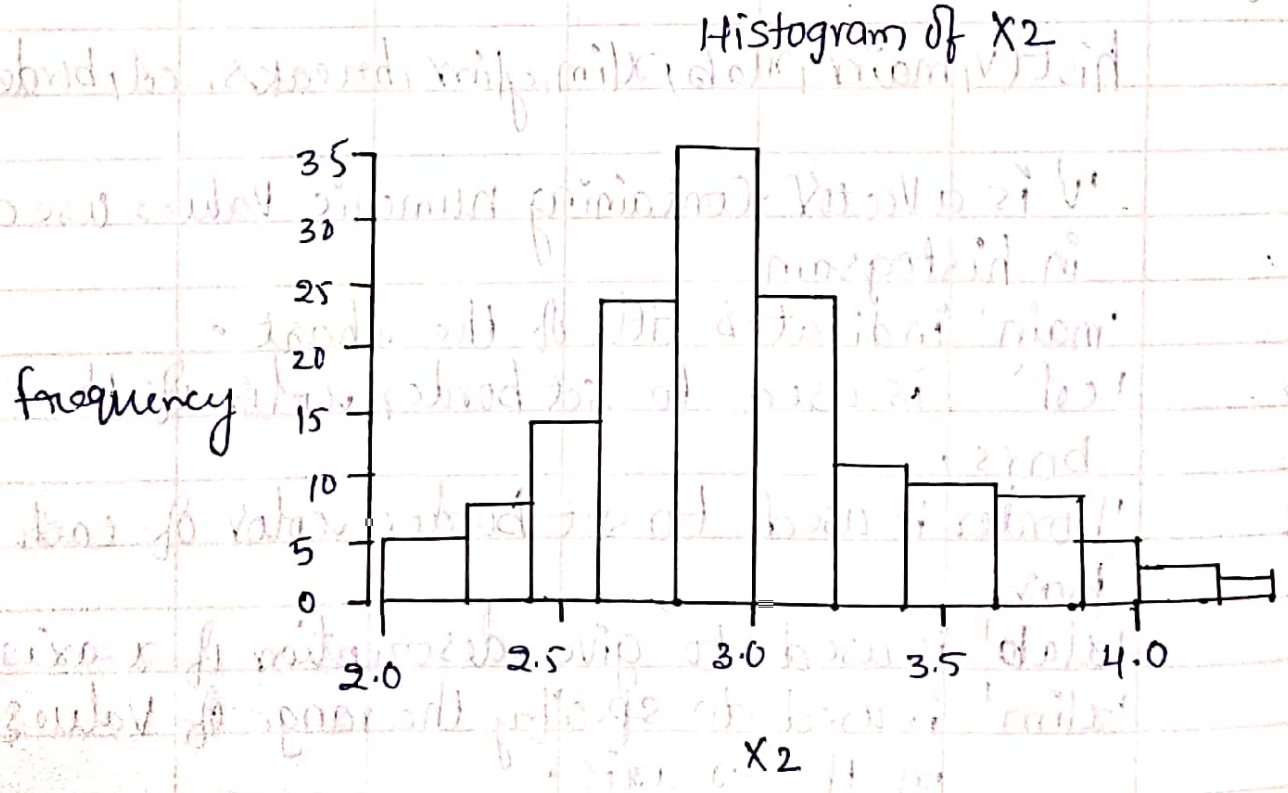
hist(v, main, xlab, xlim, ylim, breaks, col, border)

'v' is a vector containing numeric values used in histogram.
'main' indicates title of the chart.
'col' is used to set border color of the bars.
'border' is used to set border color of each bar.
'xlab' is used to give description of x-axis.
'xlim' is used to specify the range of values on the x-axis.
'ylim' is used to specify the range on the y-axis.

→



→



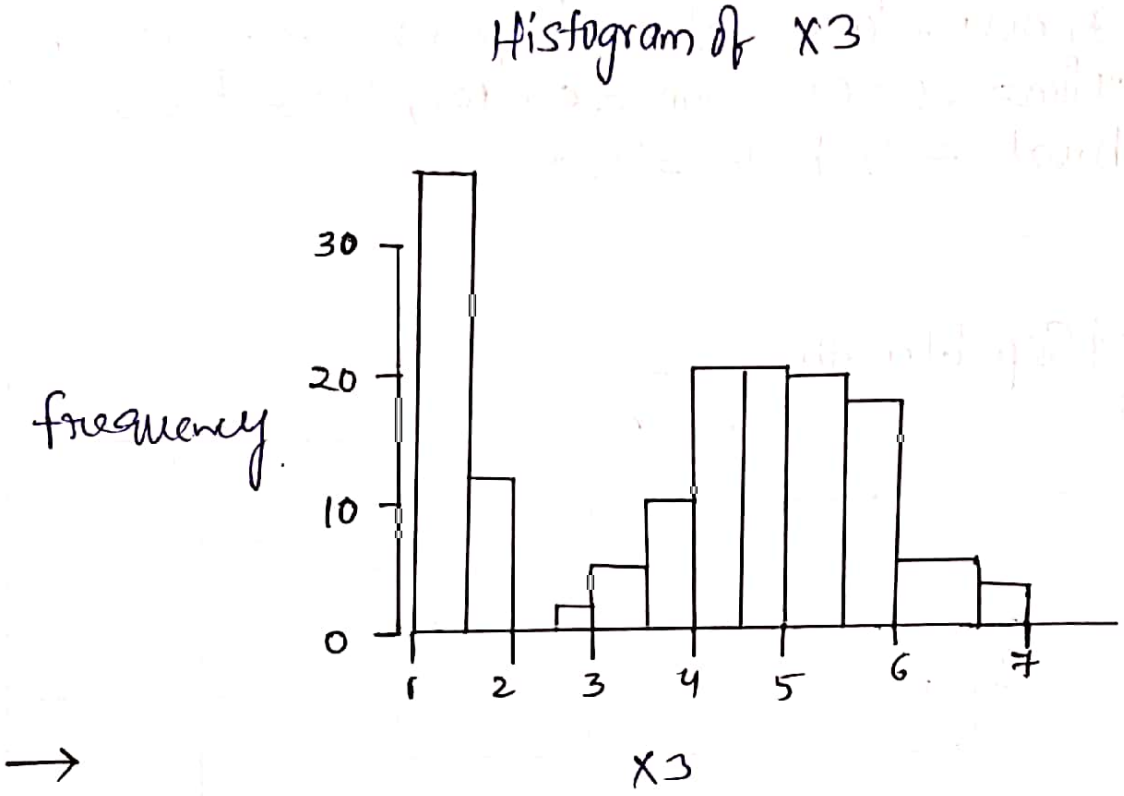
'breaks' is used to mention the width of each bar.

Ex:- hist(x, main = "acme", xlab = "alpha", ylab = "beta", xlim = c(0, 0.4), ylim = c(0, 10), col = "blue", breaks = 10, border = "red")

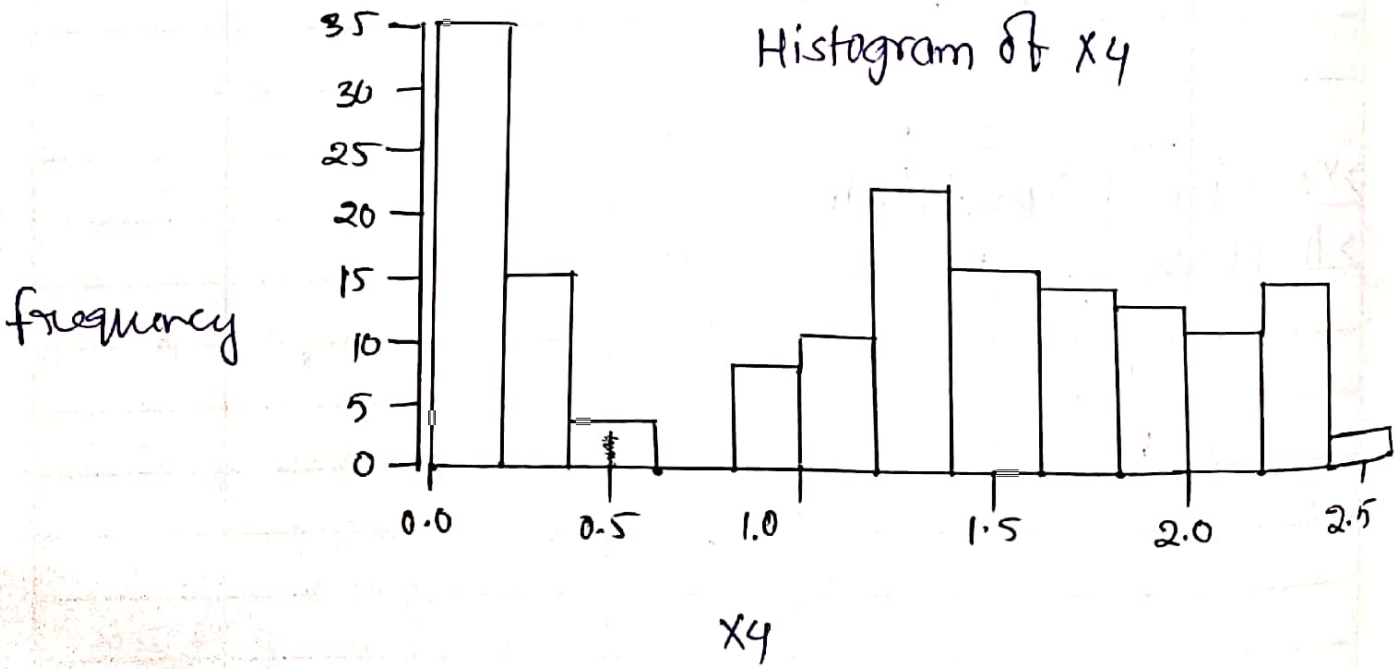
```
> x1 <- iris$Sepal.Length
> hist(x1)
```

```
> x2 <- iris$Sepal.Width
> hist(x2)
```

→



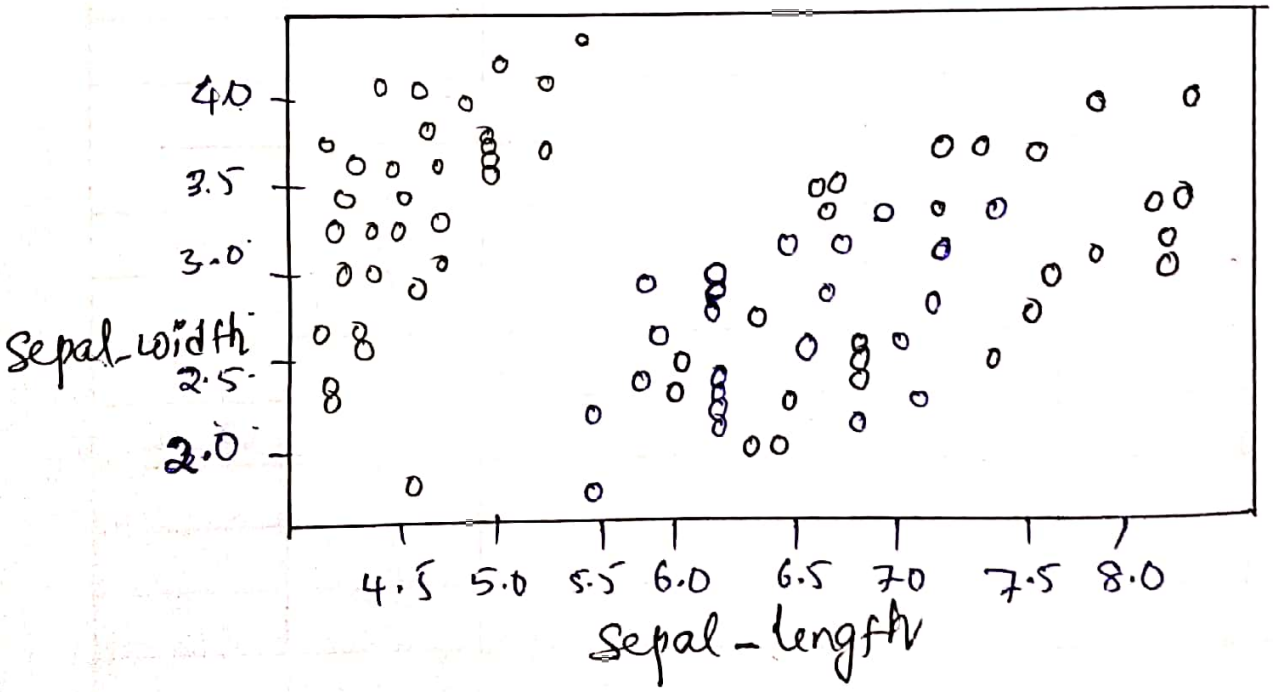
→



```
> x3 <- iris $ Petal.Length
> hist(x3)
```

```
> x4 <- iris $ Petal.Width
> hist(x4)
```

→ Sepal_length Vs sepal_width



Date :

Scatter plots :-

Scatter plots show many points plotted in the cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

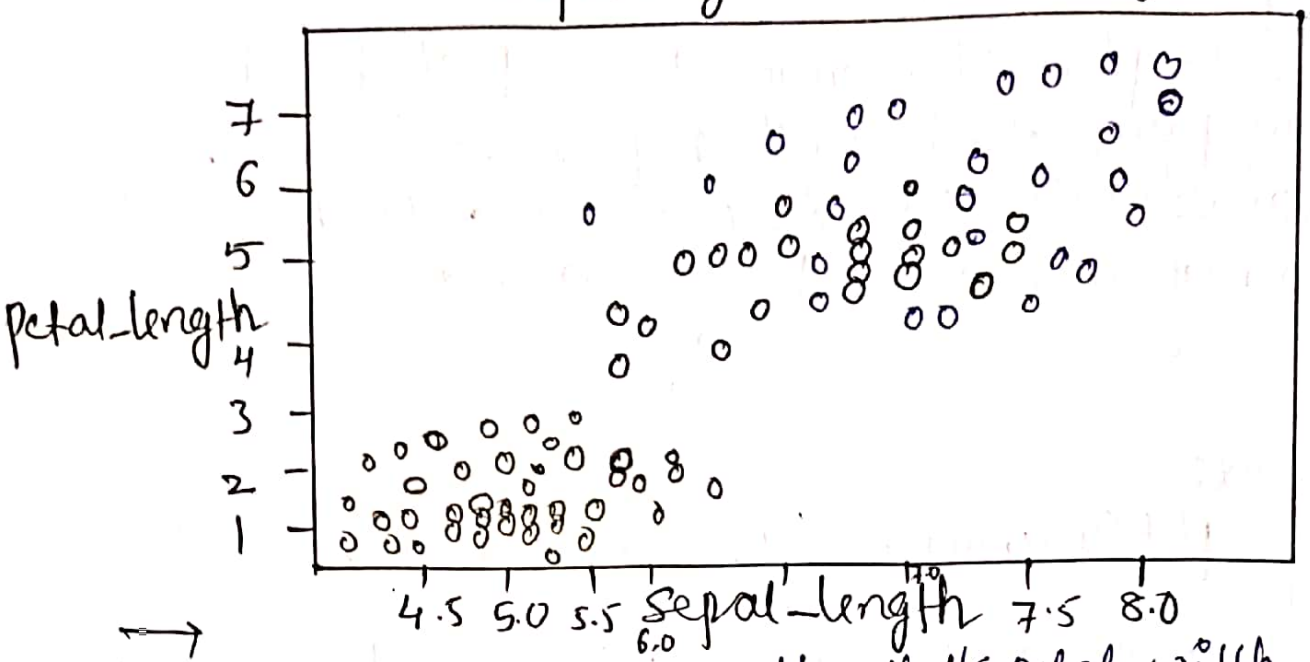
Syntax :-

plot(x, y, main, xlab, ylab, xlim, ylim, axes)

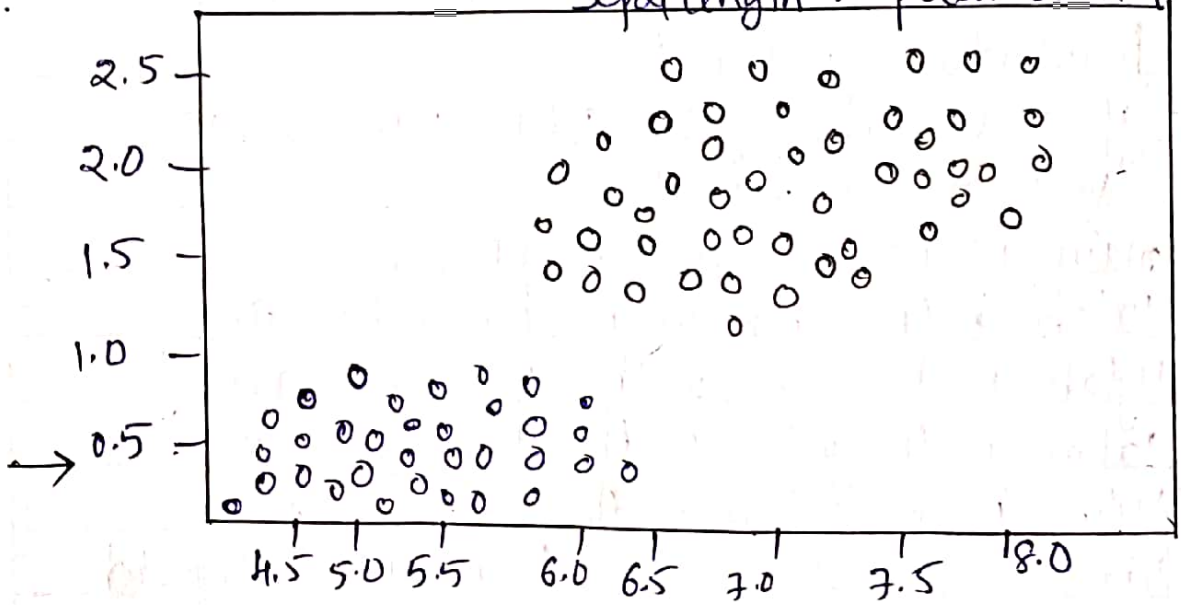
'x' is the data set whose values are the horizontal coordinates
'y' is the data set whose values are vertical coordinates
'main' is title of the graph
'xlab' is the label in the horizontal axis
'ylab' is the label in the vertical axis
'xlim' is the limits of the values of x.
'ylim' is the limits of the values of y.
'axes' indicates whether both axes should be drawn on the plot.

```
>plot(x=x1, y=x2, col=c("red","blue","green")[iris$Species],
    xlab="Sepal.Length", ylab="Sepal.width",
    main="Sepal.Length Vs Sepal.width")
```

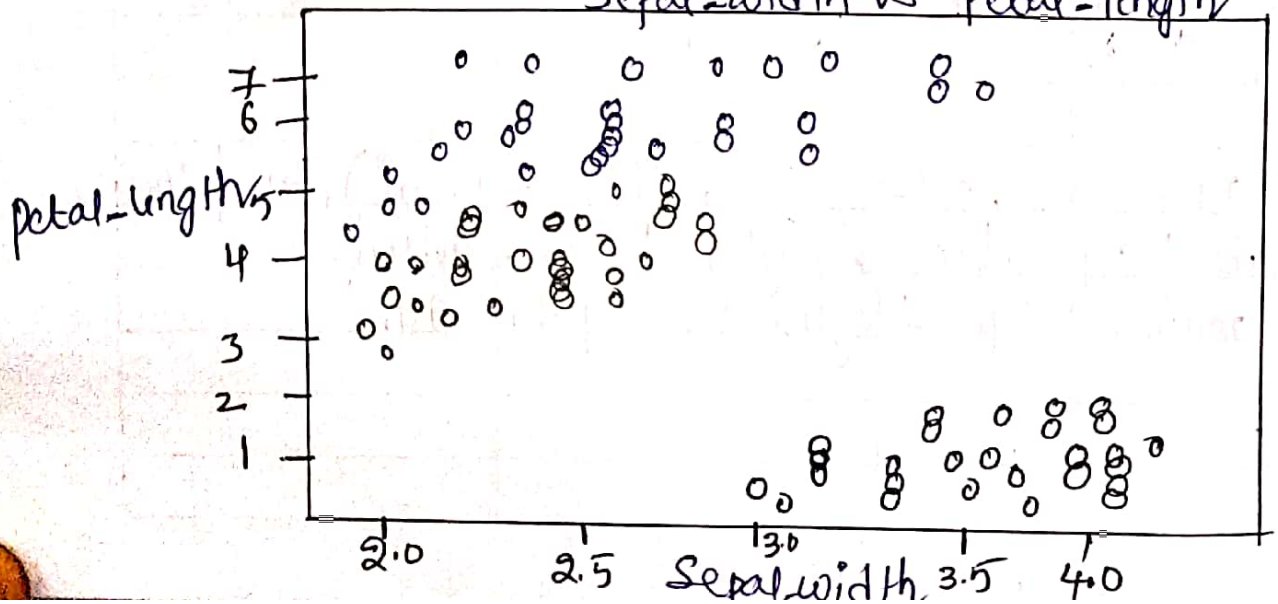
→ Sepal length Vs Petal-length



→ Sepal length Vs petal-width



Sepal-width Vs Petal-length

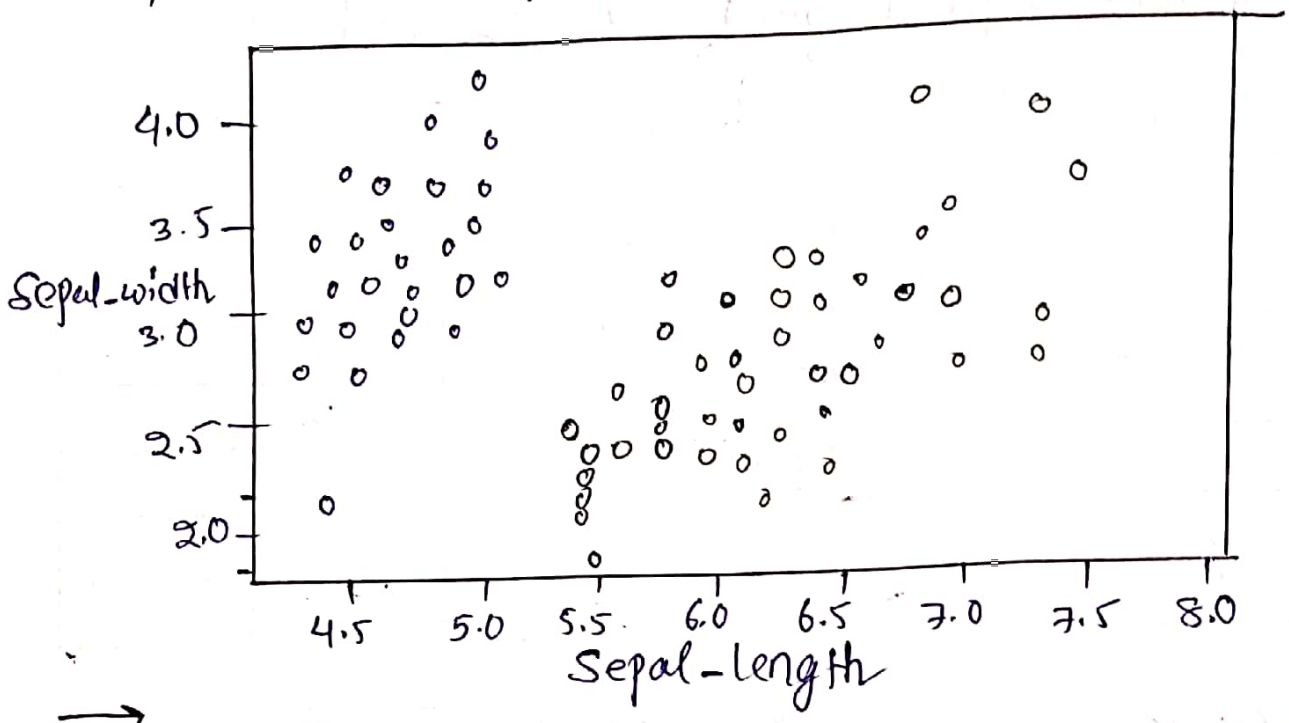


```
>plot(x=x1, y=x3, col=c("red","blue","green")[iris$Species],
     xlab="sepal-length", ylab="petal-length",
     main="sepal-length Vs petal-length")
```

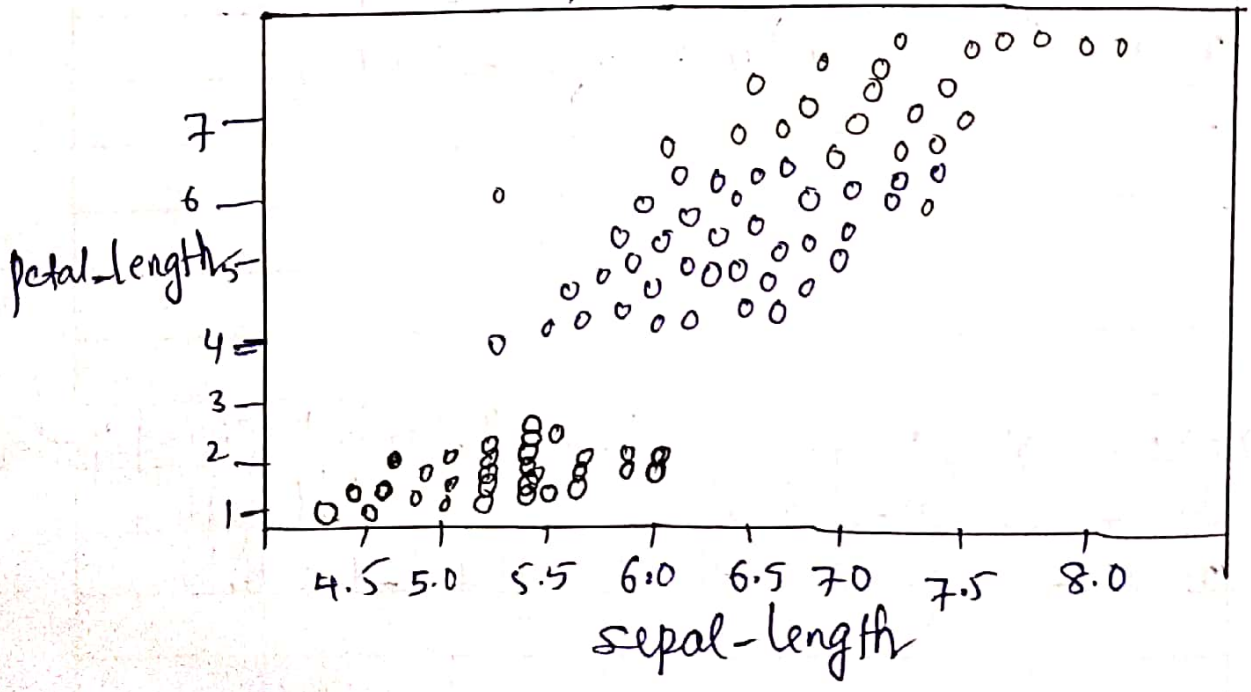
```
>plot(x=x1, y=x4, col=("red","blue","green")[iris$Species],
     xlab="sepal-length", ylab="petal-width",
     main="sepal-length Vs petal-width")
```

```
>plot(x=x2, y=x3, col=c("red","blue","green")[iris$Species],
     xlab="sepal-width", ylab="petal-length",
     main="sepal-width Vs petal-length")
```

→ Sepal-length Vs sepal-width



→ Sepal-length Vs Petal-length



```
> plot(x=x2, y=x4, col= c("red","blue","green")[iris$Species]
     xlab="sepal_width", ylab="petal_width",
     main="sepal_width Vs petal_width").
```

```
> plot(x=x3, y=x4, col= c("red","blue","green")[iris$Species],
     xlab="petal_length", ylab="petal_width",
     main="petal_length Vs petal_width")
```

(4) Generate box plots for each of the numeric attributes. Identify the attribute with the highest variance.

Boxplots :=

Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

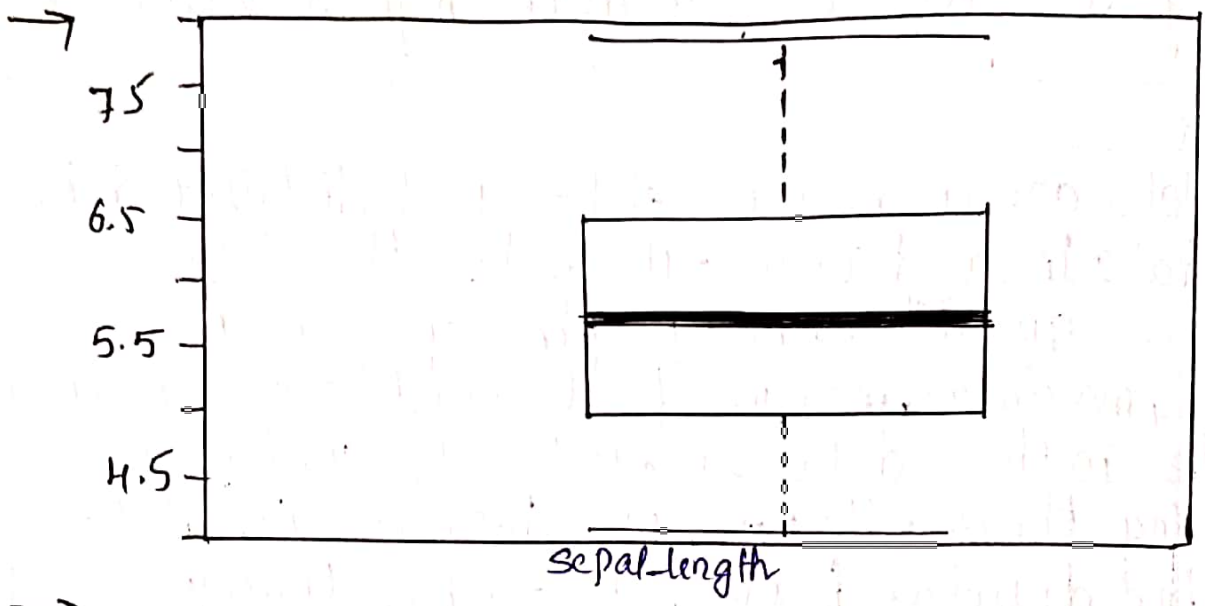
Syntax :=

boxplot(x, data, notch, varwidth, names, main)

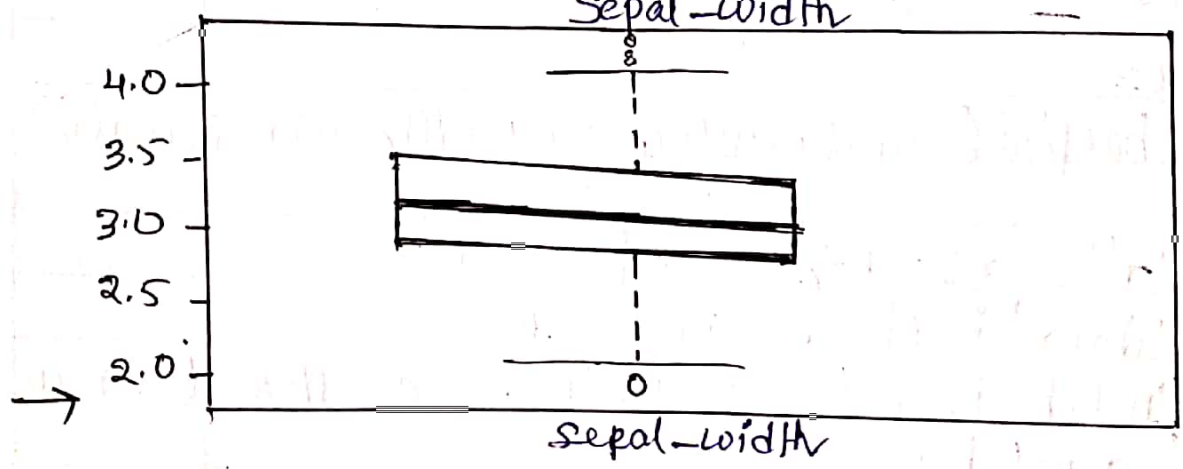
'x' is a vector or a formula.
'data' is the data frame
'notch' is a logical value. set as TRUE to draw a notch.
'varwidth' is a logical value. set as true to draw width of the box proportionate to the sample size.
'names' are the group labels which will be printed under each boxplot.
'main' is used to give a title to the graph.

```
> x1 <- iris$Sepal.Length
> boxplot(x1, xlab=
```

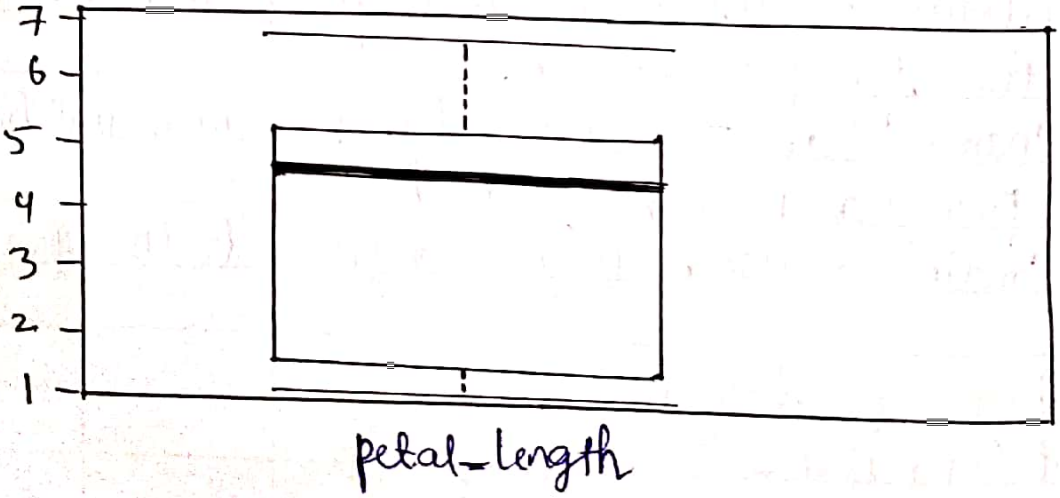
Sepal-length



Sepal-width



Petal-length



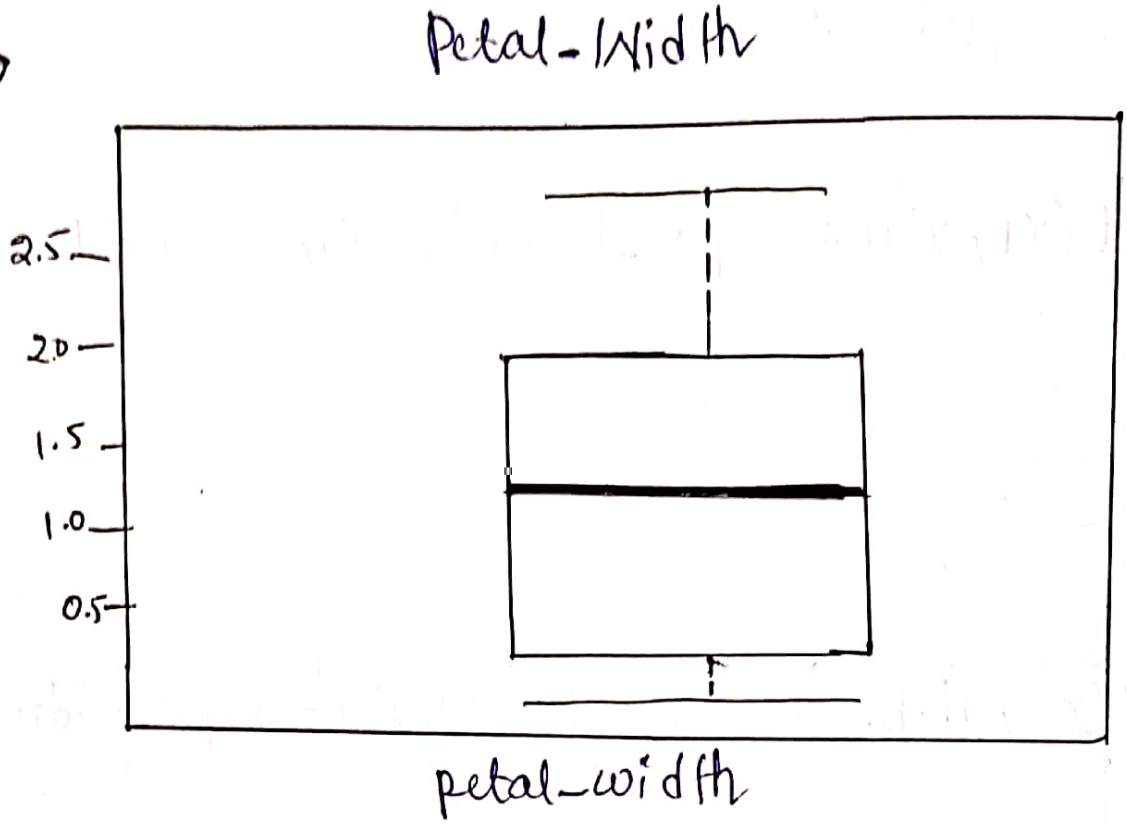
```
x2 <- iris$Sepal.Width
x3 <- iris$Petal.Length
x4 <- iris$Petal.Width
```

```
> boxplot(x1, xlab="sepal_length", main="sepal_length")
```

```
> boxplot(x2, xlab="sepal_width", main="sepal_width")
```

```
> boxplot(x3, xlab="petal_length", main="petal_length")
```

→



→

[1] 3.1162278

```
> boxplot(x4, xlab="petal-width", main="petal-width")

> var1 = var(x1)
> var2 = var(x2)
> var3 = var(x3)
> var4 = var(x4)

> max1 = max(cc(var1, var2, var3, var4))

> print(max1)
```

→ [1] "red" "green" "yellow"

→ [1] "1" "5" "6" "1" . "8" "11" "a"

→ [1] 5

→ [1] "2" "3" "5" "aa" "bb" "cc" "dd" "ee"

→ [1] 2 5 9 15 10

⑤ Study of homogeneous and heterogeneous iris structures such as vectors, matrix, array, list, data frame in R

Vectors :-

When we want to create vector with more than one element, we should use c() function which means to combine the elements into a vector.

```
> apple <- c("red","green","yellow")
> print(apple)

> num <- c(1,5,6,1,8,11,"a")
> print(num)

> length(c("aa","bb","cc","dd","ee"))

> n = c(2,3,5)
> s = c("aa","bb","cc","dd","ee")
> c(n,s)

> a = c(1,3,5,7)
> b = c(1,2,4,8,9)
> a+b
```

→

	[,1]	[,2]	[,3]
[1,]	2	4	3
[2,]	1	5	7

→

	[,1]	[,2]	[,3]
[1,]	2	3	5
[2,]	4	1	7

→ [1] 7

→ [1] 4 1 7

→ [1] 5 7

⟶

	[,1]	[,2]
[1,]	2	5
[2,]	4	7

⟶

	[,1]	[,2]
[1,]	2	4
[2,]	3	1
[3,]	5	7

Matrices :

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

```
#matrix
A = matrix(
+ c(2,4,3,1,5,7), # data elements
+ nrow=2,          # number of rows
+ ncol=3,          # number of columns
+ byrow=TRUE)      # fill matrix by rows

>A = matrix(c(2,4,3,1,5,7), nrow=2, ncol=3, byrow=TRUE)

>A = matrix(c(2,4,3,1,5,7), nrow=2, ncol=3)

>A[2,3]

>A[2,]

>A[,3]

>A[,c(1,3)]

>t(A)
```

→

, , 1

	[, 1]	[, 2]	[, 3]
[1,]	"green"	"yellow"	"green"
[2,]	"yellow"	"green"	"yellow"
[3,]	"green"	"yellow	"green"

, , 2

	[, 1]	[, 2]	[, 3]
[1,]	"yellow"	"green"	"yellow"
[2,]	"green"	"yellow"	"green"
[3,]	"yellow"	"green"	"yellow"

→

[1]

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

Arrays :-

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimensions.

```
a <- array( c("green","yellow"), dim= c(3,3,2))
```

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.

```
BMI <- data.frame(gender = c("Male","Male","Female"),
        height = c(132,171.5,165), weight = c(81,93,78),
        Age = c(42,38,26))
print(BMI)
```

→

[[1]]
[1] 2 5
[[2]]
[1] "red" "green" "yellow"
[[3]]
[1] 21.3
[[4]]

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

Lists :-

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

```
eglist <- list(c(2,5), c("red","green","yellow"), 21.3,
          BMI), mat
print(eglist)
```

→

[1] -0.89767384 -1.13920048 -1.36072709 -1.50149039

[5] -1.01843718 -0.95538397 -1.50149039 -1.01843718

[9] -1.74301699 -1.13920048 -0.95538397 -1.25996379

[13] -1.25996379 -1.86378030 -0.05233076 -0.17301407

⑥ Write R program using "apply" group of functions to create and apply normalization function on each of the numeric variables/columns of iris dataset to transform them into a value around 0 with z-score normalization

Z-Score :-

Z-Score (aka, a standard score) indicates how many standard deviations an element is from the mean. A Z-score can be calculated from the following formula.

$$z = (x - \mu)/\sigma$$

where z is the z-score,
x is the value of the element,
μ is the population mean
σ is the standard deviation

```
> x1 <- iris$Sepal.Length
> zscr1 <- (x1 - mean(x1)) / sd(x1)
> zscr1
```

→ [1] 1.01560199 −0.13153881 0.32731751 0.09768935
[5] 1.24503015 1.93331463 0.78617383 0.78617383
[9] −0.36096697 0.09788935 1.47445831 0.78617383

→ [1] −1.33575163 −1.33575163 −1.39239929 −1.27910398
[5] −1.33575163 −1.16580868 −1.33575163 −1.27910398
[9] −1.33575163 −1.27910398 −1.27910398 −1.22245633

→ [1] −1.31105214 82 −1.31105214 82 −1.31105214 82 −1.31105214 82
[5] −1.31105214 82 −1.94866679 50 −1.17985947 16 −1.31105214 82
[9] −1.31105214 82 −1.44224482 48 −1.31105214 82 −1.31105214 82

```
>X2 <- iris$Sepal.Width
>ZSCR2 <- (X2 - mean(X2))/ sd(X2)
>ZSCR2
```

```
>X3 <- iris$Petal.Length
>ZSCR3 <- (X3 - mean(X3))/sd(X3)
>ZSCR3
```

```
>X4 <- iris$Petal.Width
>ZSCR4 <- (X4 - mean(X4))/sd(X4)
>ZSCR4
```

a) Use R to apply linear regression to predict evaporation coefficient in terms of air velocity using the iris given below.

Air Velocity (cm/sec)	20, 60, 100, 140, 180, 220, 260, 300 340, 380
Evaporation Coefficient (sq mm/sec)	0.18, 0.37, 0.35, 0.78, 0.56, 0.75 1.18, 1.36, 1.17, 1.65

Regression:

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In linear Regression these two variables are related through an equation, where exponent of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

→

	airvelocity	evaporation coefficient
1	20	0.18
2	60	0.37
3	100	0.35
4	140	0.78
5	180	0.56
6	220	0.75
7	260	1.18
8	300	1.36
9	340	1.17
10	380	1.65

→ Call:
lm (formula = AV $ airvelocity ~ AV $ evaporation coefficient)

Coefficients:

(Intercept)	AV $ evaporation coefficient
2.564	236.450

Linear regression, $y = ax + b$

y is the response variable
x is the predictor variable
a and b are constants which are called the coeffients.

lm() function
This function creates the relationship model between the predictor and the response variable.

syntax:

lm(formula, data)
- data → vector on which formula will be applied.
- formula → Symbol presenting the relation between x and y

```
> AV <- data.frame(airvelocity = c(20,60,100,140,180,220,260,
300,340,380), evaporationcoefficient = c(0.18,0.37,
0.35,0.78,0.56,0.75,1.18,1.36,1.17,1.65)
> print(AV)

> model <- lm(AV$airvelocity ~ AV$evaporationcoefficient)
> print(model)
```

→ Call:
lm(formula = AV$airVelocity ~ AV$evaporationCoefficient)

Residuals:

Min	1Q	Median	3Q	Max
-46.99	-24.88	-17.14	33.74	60.79

Coefficients:

	Estimated	Std. Error	t value	Pr(>|t|)	
(Intercept)	2.564	25.804	0.099	0.923	
AV$evaporationCoefficient	236.450	27.035	8.746	2.29e-05	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 39.53 on 8 degrees of Freedom
Multiple R-squared: 0.9053, Adjusted R-squared: 0.8935
F-statistic: 76.49 on 1 and 8 DF, p-value: 2.28e-05

→ [1] 0.9514814

→ [1] 0.9514814

b) Analyze the significance of residual standard-error value, R-squared value, F-statistic. Find the correlation coefficient for this data and analyze the significance of the correlation value.

Residual Standard error is measure of the quality of a linear regression fit... The Residual Standard Error is the average amount of that the response (dist) will deviate from the true regression line.

R-squared statistic provides a measure of how well the model is fitting the actual data.

F-statistic is a good indicator of whether there is a relationship between our predictor and the response variable.

Is it true that we can have a F value indicating a strong relationship that is NON LINEAR. SO that our RSE is high and our R squared is low

```
> Summary(model)
> cor(AV$airVelocity, AV$evaporationCoefficient)
> y <- cor(AV$airVelocity, AV$evaporationCoefficient)
> y
```

→ [1] 2.995732 4.094345 4.605170 4.941642 5.192957
5.393628 5.560682 5.703782 5.828946
5.940171

→ Call:
lm(formula = x ~ AV$evaporation coefficient)

Coefficients:

(Intercept)	AV$evaporation coefficient
3678	1.614

(c) Perform a log transformation on the 'Air Velocity' column, perform linear regression again and analyze all the relevant values.

```
> x <- log(AV$airvelocity)
> m <- lm(X ~ AV$evaporation Coefficient)
> m
```