EX.NO: 1	R STUDIO DATATYPES

AIM:

To execute the following R-objects

- Vectors
- Lists
- Matrices
- Array
- Factor
- Data Frames

PROCEDURE AND CODE:

Step 1: Open **R studio** version **4.4.2**

Step 2: Write the following code in the console

Vector objects

1. Logical

```
> v<-FALSE
> print(class(v))
```

Output:

```
[1] "numeric"
```

2. Numeric

```
> v<-178.045
> print(class(v))
```

Output:

```
[1] "numeric"
```

3. Integer

```
> v<-20L
> print(class(v))
```

Output:

```
[1] "integer"
```

4. Complex

```
> v<-33i+5
> print(class(v))
```

Output:

```
[1] "complex"
```

5. Character

```
> v<-"String"
> print(class(v))
```

Output:

```
[1] "character"
```

6. Raw

```
> v<-charToRaw("new")
> print(class(v))
```

Output:

```
[1] "raw"
```

VECTORS

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

Output:

```
[1] "red" "green" "yellow"
[1] "character"
```

LIST

```
> list1 <- list(c(2,5,3),21.3,sin)
> print(list1)
```

Output:

```
[[1]]
[1] 2 5 3
[[2]]
[1] 21.3
[[3]] function (x) .Primitive("sin")
```

MATRICES

```
> M = matrix( c('a','a','b','c','b')
, nrow = 2, ncol = 3, byrow = TRUE)

> print(M)
```

Output:

```
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
```

ARRAY

```
> a <- array(c('green','yellow'),
      dim = c(3,3,2))
> print(a)
```

Output:

```
, , 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"
```

FACTOR

```
> apple_colors <-c('green','green',
'yellow','red','red','red','green')
> factor_apple <- factor(apple_colors)
> print(factor_apple)
> print(nlevels(factor_apple))
```

Output:

```
[1] green  green  yellow red
                   red    red    green
[1] 3
```

DATA FRAMES

```
> 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)
```

Output:

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

Step 3: Execute the following code in R studio

RESULT:

Thus, the following R Studio objects are executed successfully.

- Vectors
- Lists
- Matrices
- Array
- Factor
- Data Frames

EX.NO: 2	R STUDIO -- VARIABLES, OPERATORS, FUNCTIONS

AIM:

To execute R studio Variables, Operators, Functions in R studio console.

PROCEDURE AND CODE:

Step 1: Open **R studio** version **4.4.2**
Step 2: Write the following code in the console

VARIABLES:

1. Variable Assignment

```
# Assignment using equal operator.
> var.1 = c(0.5,187,266,33)
> # Assignment using leftward operator.
> var.2 <- c("CS","BS")
> # Assignment using rightward operator.
> c(FALSE,1) -> var.3
> print(var.1)
> cat ("var.1 is ", var.1 ,"\n")
> cat ("var.3 is ", var.3 ,"\n")
```

Output:

```
[1]   0.5 187.0 266.0  33.0
var.1 is  0.5 187 266 33
var.3 is  0 1
```

2. Data Type of a Variable

```
> var_x <- "Hello"
> cat("The class of var_x is
",class(var_x),"\n")
> var_x <- 34.5
> cat(" Now the class of var_x is
",class(var_x),"\n")
> var_x <- 287L
> cat(" Next the class of var_x becomes
",class(var_x),"\n")
```

Output:

```
The class of var_x is character
Now the class of var_x is
numeric
Next the class of var_x becomes

integer
```

3. Finding Variables

```
> print(ls())
```

Output:

```
[1]  "a"                    "BMI"
"colours"           "factor_Lan"
"Lan_names"   "list1"         "M"
"v"          "var.1"      "var.2"
"var.3"      "var_x"
```

4. Deleting Variables

```
rm(var.3)
print(var.3)
```

Output:

```
[1] "var.3"
Error in print(var.3) : object
'var.3' not found
```

OPERATORS:

ARITHMETIC OPERATORS

1. Adding two vectors

```
v <- c( 2,5.5,6)
t <- c(8, 3, 4)
print(v+t)
```

Output:

```
[1] 10.0 8.5 10.0
```

2. Subtracting two vectors

```
v <- c( 2,5.5,6)
t <- c(8, 3, 4)
print(v-t)
```

Output:

```
[1] -6.0 2.5 2.0
```

3. Multiplying two vectors

```
v <- c( 2,5.5,6)
t <- c(8, 3, 4)
print(v*t)
```

Output:

```
[1] 16.0 16.5 24.0
```

4. Dividing two vectors

```
v <- c( 2,5.5,6)
t <- c(8, 3, 4)
print(v/t)
```

Output:

```
[1] 0.250000 1.833333 1.500000
```

5. Remainder of first vector

```
v <- c( 2,5.5,6)
t <- c(8, 3, 4)
print(v%%t)
```

Output:

```
[1] 2.0 2.5 2.0
```

RELATIONAL OPERATORS

1. Greater than

```
v <- c(2,5.5,6,9)
t <- c(8,2.5,14,9)
print(v>t)
```

Output:

```
[1] FALSE TRUE FALSE FALSE
```

2. Less than

```
v <- c(2,5.5,6,9)
t <- c(8,2.5,14,9)
print(v < t)
```

Output:

```
[1] TRUE FALSE TRUE FALSE
```

3. Equal to

```
v <- c(2,5.5,6,9)
t <- c(8,2.5,14,9)
print(v == t)
```

Output:
```
[1] FALSE FALSE FALSE TRUE
```

4. Less than or equal to

```
v <- c(2,5.5,6,9)
t <- c(8,2.5,14,9)
print(v<=t)
```

Output:
```
[1] TRUE FALSE TRUE TRUE
```

5. Greater than or equal to

```
v <- c(2,5.5,6,9)
t <- c(8,2.5,14,9)
print(v>=t)
```

Output:
```
[1] FALSE TRUE FALSE TRUE
```

LOGICAL & ASSIGNMENT OPERATORS

1. Logical AND

```
v <- c(3,0,TRUE,2+2i)
t <- c(1,3,TRUE,2+3i)
print(v&&t)
```

Output:
```
[1] TRUE
```

2. Logical OR

```
v <- c(0,0,TRUE,2+2i)
t <- c(0,3,TRUE,2+3i)
print(v||t)
```

Output:
```
[1] FALSE
```

3. Left assignment

```
v1 <- c(3,1,TRUE,2+3i)
v2 <<- c(3,1,TRUE,2+3i)
v3 = c(3,1,TRUE,2+3i)
print(v1)
print(v2)
print(v3)
```

Output:
```
[1] 3+0i 1+0i 1+0i 2+3i
[1] 3+0i 1+0i 1+0i 2+3i
[1] 3+0i 1+0i 1+0i 2+3i
```

4. Right assignment

```
c(3,1,TRUE,2+3i) -> v1
c(3,1,TRUE,2+3i) ->> v2
print(v1)
print(v2)
```

Output:
```
[1] 3+0i 1+0i 1+0i 2+3i
[1] 3+0i 1+0i 1+0i 2+3i
```

FUNCTIONS

Built-in-function

```
v <- c(3,0,TRUE,2+2i)
t <- c(1,3,TRUE,2+3i)
print(v&&t)
```

Output:
```
[1] TRUE
```

Logical OR

```
v <- c(0,0,TRUE,2+2i)
t <- c(0,3,TRUE,2+3i)
print(v||t)
```

Output:
```
[1] FALSE
```

Left assignment

```
v1 <- c(3,1,TRUE,2+3i)
v2 <<- c(3,1,TRUE,2+3i)
v3 = c(3,1,TRUE,2+3i)
print(v1)
print(v2)
print(v3)
```

Output:
```
[1] 3+0i 1+0i 1+0i 2+3i
[1] 3+0i 1+0i 1+0i 2+3i
[1] 3+0i 1+0i 1+0i 2+3i
```

Right assignment

```
c(3,1,TRUE,2+3i) -> v1
c(3,1,TRUE,2+3i) ->> v2
print(v1)
print(v2)
```

Output:
```
[1] 3+0i 1+0i 1+0i 2+3i
[1] 3+0i 1+0i 1+0i 2+3i
```

Step 3: Execute the following code in R studio

RESULT:

Thus, the following R Studio - Variables, Operators, Functions are executed successfully.

EX.NO: 3	R STUDIO -- STRING, VECTORS, LIST, MATRICES, ARRAY

AIM:

To execute R studio String, Vectors, List, Matrices, Array in R studio console.

PROCEDURE AND CODE:

Step 1: Open **R studio** version **4.4.2**
Step 2: Write the following code in the console
Step 3: Execute the following code in R studio

R - Strings

1. String manipulation

```
a <- "Hello"
b <- 'How'
c <- "are you? "
print(paste(a,b,c))
print(paste(a,b,c, sep = "-"))
print(paste(a,b,c, sep = "",
collapse = ""))
```

Output:

```
[1] "Hello How are
you? " [1] "Hello-
How-are you? " [1]
"HelloHoware you? "
```

2. Formatting numbers & strings

```
result<- format(23.123456789,digits=9)
print(result)
result<-format(c(6, 13.14521),
scientific = TRUE)
print(result)
```

Output:

```
[1] "23.1234568"
[1] "6.000000e+00"
"1.314521e+01"
[1] "23.47000"
```

3. Counting number of characters in string

```
result <- nchar("Count the number of
characters")
print(result)
```

Output:

```
[1] 30
```

4. Changing the case

```
result <- toupper("Changing To Upper")
print(result)
result <- tolower("Changing To Lower")
print(result)
```

Output:

[1] "CHANGING TO UPPER"
[1] "changing to lower"

5. Extracting parts of a string

```
result <- substring("Extract",5,7)
print(result)
```

Output:

[1] "act"

R vectors

1. Vector Creation

```
print("abc");
print(12.5)
print(63L)
print(TRUE)
print(2+3i)
print(charToRaw('hello'))
```

Output:

```
[1] "abc"
[1] 12.5
[1] 63
[1] TRUE
[1] 2+3i
[1] 68 65 6c 6c 6f
```

2. Accessing Vector Elements

```
t <-c("Sun","Mon","Tue","Wed"
,"Thurs","Fri","Sat")
u <- t[c(2,3,6)]
print(u)
v <- t[c(TRUE,FALSE,FALSE,FALSE
,FALSE,TRUE,FALSE)]
print(v)
x <- t[c(-2,-5)]
print(x)
y <- t[c(0,0,0,0,0,0,1)] print(y)
```

Output:

```
[1] "Mon" "Tue" "Fri"
 [1] "Sun" "Fri"
 [1] "Sun" "Tue"
"Wed" "Fri" "Sat"
 [1] "Sun
```

3. Vector manipulation

```
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)
add.result <- v1+v2
print(add.result)
sub.result <- v1-v2
print(sub.result)
multi.result <- v1*v2
print(multi.result)
divi.result <- v1/v2
print(divi.result)
```

Output:

```
[1]  7 19  4 13  1 13
 [1] -1 -3  4 -3 -1  9
 [1] 12 88  0 40  0 22
 [1] 0.7500000 0.7272727
Inf 0.6250000 0.0000000
5.5000000
```

4. Vector element sorting

```
v <- c(3,8,4,5,0,11, -9, 304)
sort.result <- sort(v)
print(sort.result)
revsort.result <- sort(v,
decreasing = TRUE)
print(revsort.result)
v <- c("Red","Blue","yellow","violet")
sort.result <- sort(v)
print(sort.result)
revsort.result <- sort(v,decreasing =
TRUE)
print(revsort.result)
```

Output:

```
[1] -9  0  3  4  5  8 11 304
 [1] 304 11  8  5  4  3  0 -9
 [1] "Blue"   "Red"    "violet"
"yellow"
 [1] "yellow" "violet" "Red"
"Blue"
```

R Lists

1. List creation

```
list_data <- list("Red", "Green",
c(21,32,11), TRUE, 51.23, 119.1)
 print(list_data)
```

Output:

```
[[1]] [1] "Red"
[[2]] [1] "Green"
[[3]] [1] 21 32 11
[[4]] [1] TRUE
[[5]] [1] 51.23
[[6]] [1] 119.1
```

2. Naming List Elements

```
list_data <- list(c("Jan","Feb","Mar")
, matrix(c(3,9,5,1,-2,8), nrow = 2),
list("green",12.3))
names(list_data) <- c("1st Quarter",
"A_Matrix", "A Inner list")
print(list_data)
```

Output:

```
$`1st_Quarter`
 [1] "Jan" "Feb" "Mar"
 $A_Matrix
      [,1] [,2] [,3]
 [1,]    3    5   -2
 [2,]    9    1    8
 $A_Inner_list
 $A_Inner_list[[1]]
 [1] "green"
 $A_Inner_list[[2]]
 [1] 12.3
```

3. List manipulation

```
list_data <- list(c("Jan","Feb","Mar")
, matrix(c(3,9,5,1,-2,8), nrow = 2),
list("green",12.3))
names(list_data) <- c("1st Quarter",
"A_Matrix", "A Inner list")
list_data[4] <- "New element"
print(list_data[4])
list_data[4] <- NULL
print(list_data[4])
list_data[3] <- "updated element"
print(list_data[3])
```

Output:

```
[1]][1] "New element"
$<NA>
NULL
$`A Inner list`
[1] "updated element"
```

4. Converting list to vector

```
list1 <- list(1:5)
print(list1)
list2 <-list(10:13)
print(list2)
v1 <- unlist(list1)
v2 <- unlist(list2)
print(v1)
print(v2)
result <- v1+v2
print(result)
```

Output:

```
[[1]][1] 1 2 3 4 5
[[1]][1] 10 11 12 13
[1] 1 2 3 4 5
[1] 10 11 12 13 14
[1] 11 13 15 17 19
```

R matrices

1. **Matrix creation**

```
M <- matrix(c(3:14), nrow = 4, byrow =
TRUE)
print(M)
N <- matrix(c(3:14), nrow = 4, byrow =
FALSE)
print(N)
rownames = c("row1", "row2", "row3",
"row4")
colnames = c("col1", "col2", "col3")
P <- matrix(c(3:14), nrow = 4, byrow =
TRUE, dimnames = list(rownames, colnames))
print(P)
```

Output:

```
[,1] [,2] [,3]
 [1,]    3    4    5
 [2,]    6    7    8
 [3,]    9   10   11
 [4,]   12   13   14
     [,1] [,2] [,3]
 [1,]    3    7   11
 [2,]    4    8   12
 [3,]    5    9   13
 [4,]    6   10   14
     col1 col2 col3
 row1    3    4    5
 row2    6    7    8
 row3    9   10   11
 row4   12   13   14
```

2. **Accessing elements of matrix**

```
rownames = c("row1", "row2", "row3",
"row4")
colnames = c("col1", "col2", "col3")
P <- matrix(c(3:14), nrow = 4, byrow =
TRUE, dimnames = list(rownames,
colnames))
print(P[1,3])
print(P[4,2])
print(P[2,])
print(P[,3])
```

Output:

```
[1] 5
[1] 13
 col1 col2 col3
    6    7    8
row1 row2 row3 row4
   5    8   11   14
```

3. **Matrix Computations**

```
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6),
nrow = 2)
print(matrix1)
matrix2 <- matrix(c(5, 2, 0, 9, 3, 4),
nrow = 2)
print(matrix2)
result <- matrix1 + matrix2
cat("Result of addition","\n")
print(result)
result <- matrix1 - matrix2
cat("Result of subtraction","\n")
print(result)
```

Output:

```
[,1] [,2] [,3]
 [1,]    3   -1    2
 [2,]    9    4    6
     [,1] [,2] [,3]
 [1,]    5    0    3
 [2,]    2    9    4
 Result of addition
     [,1] [,2] [,3]
 [1,]    8   -1    5
 [2,]   11   13   10
 Result of subtraction
```

```
     [,1] [,2] [,3]
[1,]   -2   -1   -1
[2,]    7   -5    2
```

R Arrays

1. Naming column and rows

```
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")
result <- array(c(vector1,vector2),dim =
c(3,3,2),dimnames = list(row.names,colum
matrix.names))
print(result)
```

Output:

```
, , Matrix1
     COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   15
, , Matrix2
     COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   15
```

2. Accessing elements of array

```
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")
result <- array(c(vector1,vector2),dim =
c(3,3,2),dimnames = list(row.names,
column.names, matrix.names))
print(result[3,,2])
print(result[1,3,1])
print(result[,,2]
```

Output:

```
COL1 COL2 COL3
   3   12   15
[1] 13
     COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   1
```

RESULT:

Thus, the following R Studio - string, vectors, list, matrices, array are executed successfully.

EX.NO: 4	R STUDIO -- FACTORS, DATA FRAMES, PACKAGES, DATA RESHAPING

AIM:

To execute R studio factors, data frames, packages and data reshaping in R studio console.

PROCEDURE AND CODE:

Step 1: Open **R studio** version **4.4.2**
Step 2: Write the following code in the console
Step 3: Execute the following code in R studio

R – Factors

1. **Factor creation**

```
data <- c("East","West","East","North",
"North","East","West","West")
print(data)
print(is.factor(data))
factor_data <- factor(data)
print(factor_data)
print(is.factor(factor_data))
```

Output:

```
[1] "East"  "West"
"East"  "North" "North"
"East"  "West"  "West"
[1] FALSE
[1] East  West  East
North North East  West
Levels: East North West
[1] TRUE
```

2. **Factors in data frame**

```
height <- c(132,151,162)
weight <- c(48,49,66)
gender <- c("male","male","female")
input_data <- data.frame
height,weight,gender) print(input_data)
print(is.factor(input_data$gender))
print(input_data$gender)
```

Output:

```
  height weight gender
 1    132     48   male
 2    151     49   male
 3    162     66 female
[1] TRUE
[1] male male female
Levels: female male
```

3. **Changing the order of levels**

```
data <- c("East","West","East",”North”)
factor_data <- factor(data)
print(factor_data)
new_order_data <- factor(factor_data,
levels = c("East","West","North"))
print(new_order_data)
```

Output:

```
[1] East West East North
Levels: East North West
[1] East West East North
Levels: East West North
```

4. **Generating factor levels**

```
v <- gl(3, 4, labels = c("Tampa",
"Seattle","Boston"))
 print(v)
```

Output:

```
Tampa   Tampa   Tampa
Tampa   Seattle Seattle
Seattle Seattle Boston
[10] Boston  Boston
Boston
Levels: Tampa Seattle
Boston
```

R- Data Frames

1. **Data Frames creation**

```
emp.data <- data.frame(emp_name =
c("Rick","Dan","Mic"), salary =
c(6230,515.2,611.0),
start_date = as.Date(c("2012-1-1", "2013-9-
2", "2014-3-5")), stringsAsFactors = FALSE)
print(emp.data)
```

Output:

```
emp_name salary
start_date
1 Rick 6230    2012-1-1
2 Dan 515.20   2013-9-2
3 Mic 611.00   2014-3-5
```

2. **Structure of data frame**

```
height <- c(132,151,162)
weight <- c(48,49,66)
gender <- c("male","male","female")
input_data <- data.frame
height,weight,gender) print(input_data)
print(is.factor(input_data$gender))
print(input_data$gender)
```

Output:

```
  height weight gender
 1    132     48   male
 2    151     49   male
 3    162     66 female
[1] TRUE
[1] male male female
Levels: female male
```

3. **Changing the order of levels**

```
data <- c("East","West","East",”North”)
factor_data <- factor(data)
print(factor_data)
new_order_data <- factor(factor_data,levels
= c("East","West","North"))
print(new_order_data)
```

Output:

```
[1] East West East North
Levels: East North West
[1] East West East North
Levels: East West North
```

4. **Generating factor levels**

Output:

```
v <- gl(3, 4, labels = c("Tampa",
"Seattle","Boston"))
 print(v)
```

```
Tampa   Tampa   Tampa
Tampa   Seattle Seattle
Seattle Seattle Boston
[10] Boston  Boston
Boston
Levels: Tampa Seattle
Boston
```

R Packages

1. Check Available R Packages

```
.libPaths()
```

Output:

```
[2] "C:/Program
Files/R/R-3.2.2/library"
```

2. Get the list of all the packages installed

```
library()
```

```
Base: The R Base Package
Boot: Bootstrap Functions
Class: Functions for Classification
cluster: "Finding Groups in Data":
Cluster Analysis Extended
Rousseeuw et al. --- etc---
```

R- data Reshaping

1. Joining Columns and Rows in a Data Frame

```
 city <- c("Tampa","Seattle")
 state <- c("FL","WA")
 zipcode <- c(33602,98104)
 addresses <- cbind(city,state,zipcode)
 cat("# # # # The First data frame\n")
 print(addresses)
 new.address <- data.frame(
   city = c("Lowry"),
   state = c("CO"),
   zipcode = c("80230"),
   stringsAsFactors = FALSE
 )
 cat("# # # The Second data frame\n")
 print(new.address)
 all.addresses <-
rbind(addresses,new.address)
 cat("# # # The combined data frame\n")
```

Output:

```
###The First data frame
city  state  zipcode
"Tampa" "FL" "33602"
"Seattle" "WA"  "98104"

###The Second data frame
city  state  zipcode
 1 Lowry CO  80230

###The combined data frame
city  state  zipcode
 1  Tampa     FL    33602
 2 Seattle   WA    98104
 3 Lowry     CO    80230
```

```
print(all.addresses)
```

2. Melt the Data

```
molten.ships <- melt(ships, id =
c("type","year"))
 print(molten.ships)
```

Output:

```
type year variable value 1
A 60 period 60 2 A 60
period 75 3 A 65 period 60
4 A 65 period 75
.......... ..........  9
B 60 period 60 10 B 60
period 75 11 B 65 period 60
12 B 65 period 75 13 B 70
period 60 ..........
```

3. Cast the Molten Data

```
recasted.ship <- cast(molten.ships,
type+year~variable,sum)
 print(recasted.ship)
```

Output:

```
type year period service
incidents
 1   A   60   135   190     0
 2   A   65   135   2190    7
 3   A   70   135   4865    24
 4   A   75   135   2244    11
 5   B   60   135   62058   68
 6   B   65   135   48979   111
 7   B   70   135   20163   56
```

RESULT:

Thus, the following R studio - factors, data frames, packages and data reshaping are executed successfully.

EX.NO: 5	R STUDIO – MEAN, MEDIAN, MODE, LINEAR REGRESSION, MULTIPLE REGRESSION

AIM:

To execute R studio factors, data frames, packages and data reshaping in R studio console.

PROCEDURE AND CODE:

Step 1: Open **R studio** version **4.4.2**
Step 2: Write the following code in the console
Step 3: Execute the following code in R studio

MEAN, MEDIAN, MODE

1. **Mean finding**

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
result.mean <- mean(x)
print(result.mean)
```

Output:
```
[1] 8.22
```

2. **Applying Trim Option**

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
result.mean <- mean(x,trim = 0.3)
print(result.mean)
```

Output:
```
[1] 5.55
```

3. **Applying NA Option**

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
result.mean <-  mean(x)
print(result.mean)
result.mean <- mean(x,na.rm = TRUE)
print(result.mean)
```

Output:
```
[1] NA
[1] 8.22
```

4. **Median**

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
median.result <- median(x)
print(median.result))
```

Output:
```
[1] 5.6
```

5. **Mode**

```
getmode <- function(v){uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v,
uniqv)))]}
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
result <- getmode(v)
```

Output:
```
[1] 2
[1] "it"
```

```
print(result)
charv <- c("o","it","the","it","it")
result <- getmode(charv)
print(result)
```

R – LINEAR REGRESSION

1. Create Relationship Model

```
x <- c(151, 174, 138, 186, 128)
y <- c(63, 81, 56, 91, 47)
relation <- lm(y~x)
print(relation)
```

Output:

```
Call:
lm(formula = y ~ x)
Coefficients:
(Intercept)      x
-38.4551     0.6746
```

2. predict() Function

```
x <- c(151, 174, 138, 186)
y <- c(63, 81, 56, 91)
relation <- lm(y~x)
a <- data.frame(x = 170)
result <-  predict(relation,a)
print(result)
```

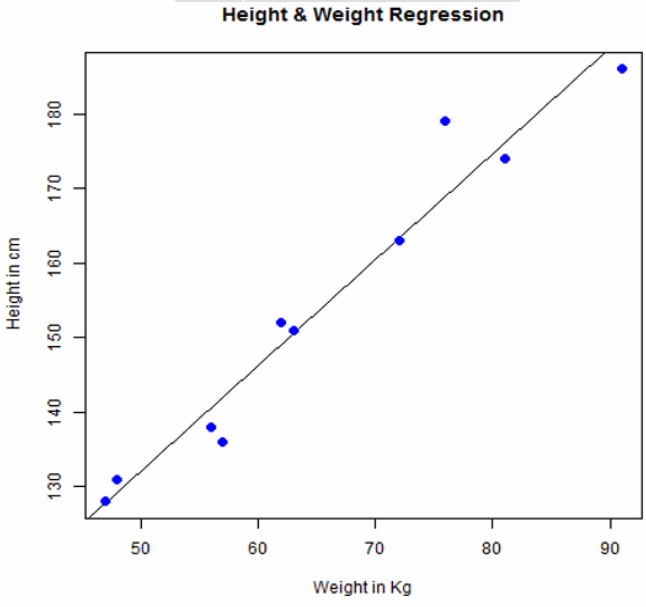
Output:

```
  1
76.22869
```

3. Visualize the Regression Graphically

```
x <- c(151, 174, 138)
y <- c(63, 81, 56, 91)
relation <- lm(y~x)
png(file = "linearregression.png")
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab =
"Height in cm")
dev.off()
```

Output:



R – MULTIPLE REGRESSION

1. Input data

```
input <-mtcars[,c("mpg","disp"
,"hp")] print(head(input))
```

Output:

```
                  mpg   disp   hp
Mazda RX4         21.0  160    110
Mazda RX4 21.0 160    110
Datsun   710   22.8   108
Hornet   21.4  258    110
Hornet   18.7  360    175
Valiant  18.1  225    105
```

2. Create Relationship Model & get the Coefficients

```
input <-mtcars[,c("mpg","disp",
"hp","wt")]
model<-lm(mpg~disp+hp+wt,data=input)
print(model)
cat("# # # # The Coefficient Values
# # # ","\n")
a <- coef(model)[1]
print(a)
Xdisp <- coef(model)[2]
Xhp <- coef(model)[3]
Xwt <- coef(model)[4]
print(Xdisp)
print(Xhp)
print(Xwt)
```

Output:

```
Call:
lm(formula = mpg ~ disp + hp
+ wt, data = input)
Coefficients:
(Intercept)  disp  hp   wt
37.10  -0.0009 -0.031 -3.800
###The Coefficient Values###
(Intercept)
37.10551
disp -0.0009370091
hp -0.03115655
wt -3.800891
```

RESULT:

Thus, the following r studio – mean, median, mode, Linear regression, multiple regression are executed successfully.