CHS
Unit-2 Task(R-Programming)

AMAN GUPTA(2021UCS1602) GARIMA(2021UCS1605) TARUN MITTAL(2021UCS1608)

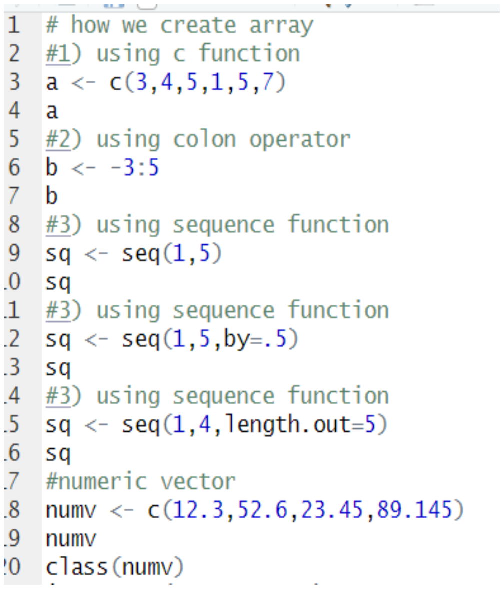
INDEX

Q1) Explain Basic Data Structure in R.

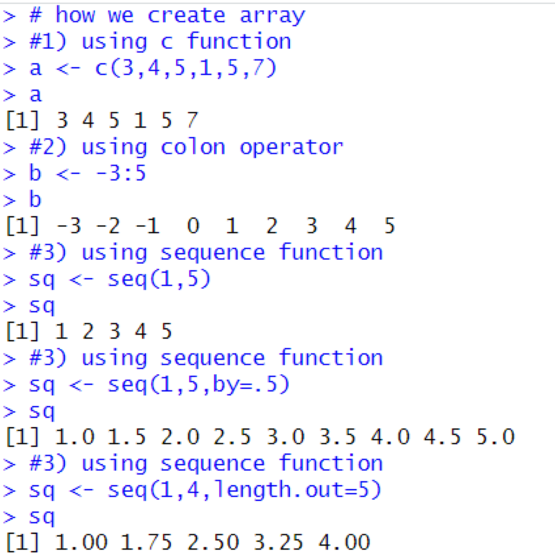
Ans. Various Data Structures in R programming are:-

i) <u>Vector:</u>

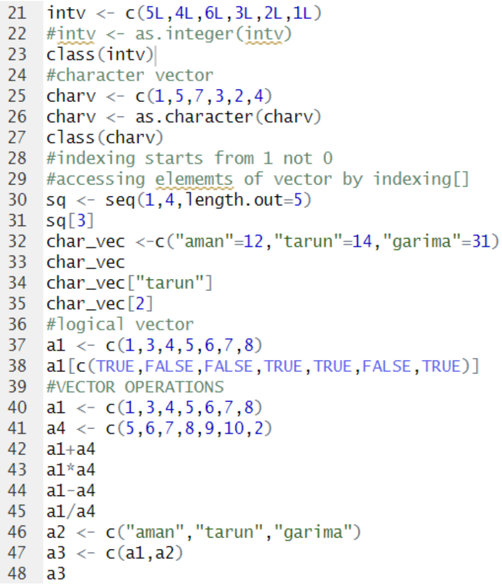
- A vector is a contiguous collection of objects of the same type.
- Common types of vectors include logical, integer, double, and character.
- It is a Collection of Homogeneous data.
- Elements of vectors are known as components.
- length(): No. of elements in vector.
- A vector can be created using the c() function, e.g., my_vector <- c(1, 2, 3).



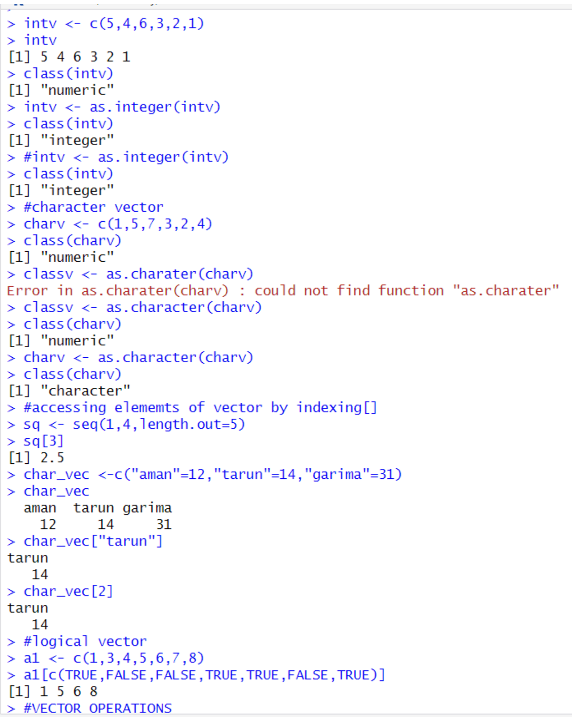
```
# how we create array
#1) using c function
a <- c(3,4,5,1,5,7)
a
#2) using colon operator
b <- -3:5
b
#3) using sequence function
sq <- seq(1,5)
sq
#3) using sequence function
sq <- seq(1,5,by=.5)
sq
#3) using sequence function
sq <- seq(1,4,length.out=5)
sq
#numeric vector
numv <- c(12.3,52.6,23.45,89.145)
numv
class(numv)
```



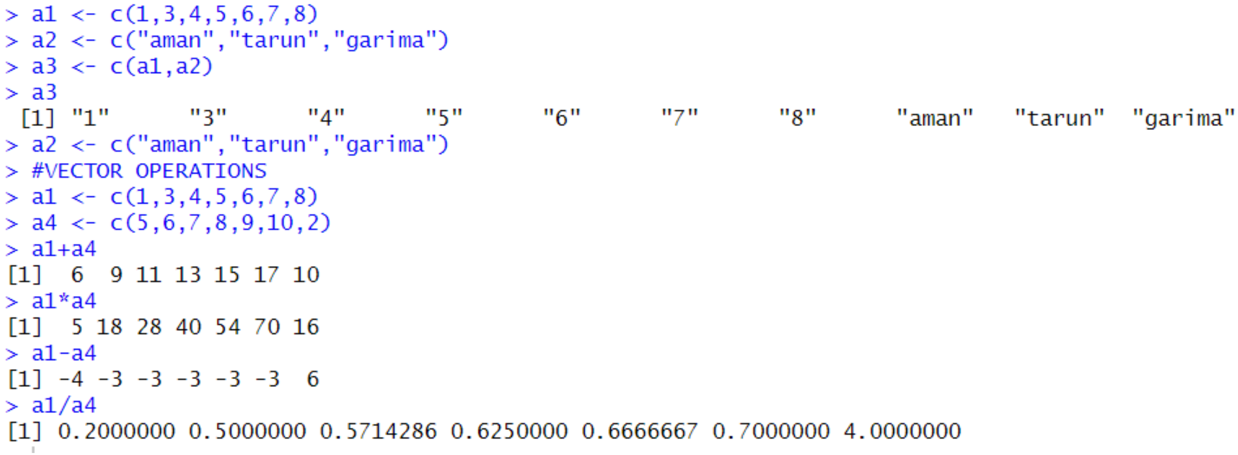
```
> # how we create array
> #1) using c function
> a <- c(3,4,5,1,5,7)
> a
[1] 3 4 5 1 5 7
> #2) using colon operator
> b <- -3:5
> b
[1] -3 -2 -1  0  1  2  3  4  5
> #3) using sequence function
> sq <- seq(1,5)
> sq
[1] 1 2 3 4 5
> #3) using sequence function
> sq <- seq(1,5,by=.5)
> sq
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> #3) using sequence function
> sq <- seq(1,4,length.out=5)
> sq
[1] 1.00 1.75 2.50 3.25 4.00
```



```
intv <- c(5L,4L,6L,3L,2L,1L)
#intv <- as.integer(intv)
class(intv)
#character vector
charv <- c(1,5,7,3,2,4)
charv <- as.character(charv)
class(charv)
#indexing starts from 1 not 0
#accessing elememts of vector by indexing[]
sq <- seq(1,4,length.out=5)
sq[3]
char_vec <-c("aman"=12,"tarun"=14,"garima"=31)
char_vec
char_vec["tarun"]
char_vec[2]
#logical vector
a1 <- c(1,3,4,5,6,7,8)
a1[c(TRUE,FALSE,FALSE,TRUE,TRUE,FALSE,TRUE)]
#VECTOR OPERATIONS
a1 <- c(1,3,4,5,6,7,8)
a4 <- c(5,6,7,8,9,10,2)
a1+a4
a1*a4
a1-a4
a1/a4
a2 <- c("aman","tarun","garima")
a3 <- c(a1,a2)
a3
```



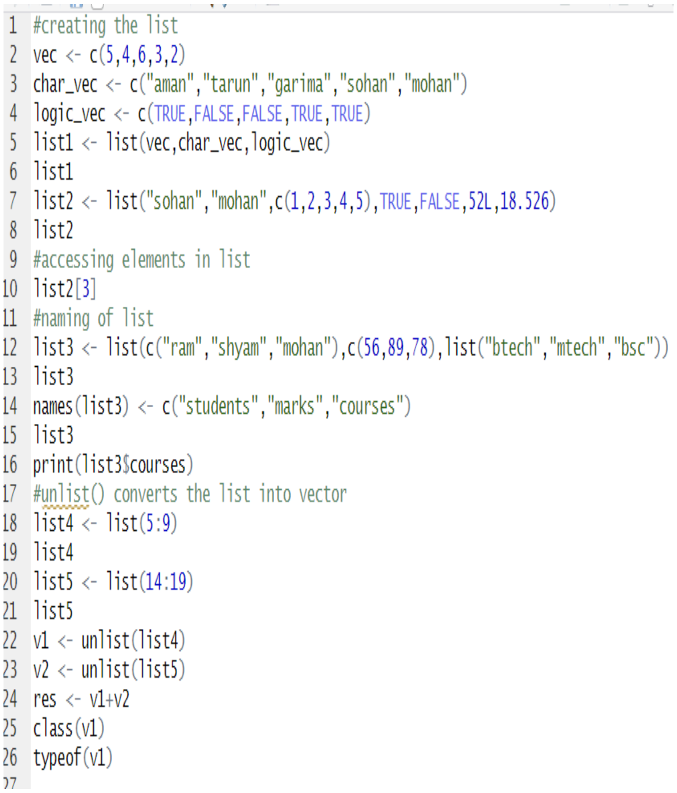
```
> intv <- c(5,4,6,3,2,1)
> intv
[1] 5 4 6 3 2 1
> class(intv)
[1] "numeric"
> intv <- as.integer(intv)
> class(intv)
[1] "integer"
> #intv <- as.integer(intv)
> class(intv)
[1] "integer"
> #character vector
> charv <- c(1,5,7,3,2,4)
> class(charv)
[1] "numeric"
> classv <- as.charater(charv)
Error in as.charater(charv) : could not find function "as.charater"
> classv <- as.character(charv)
> class(charv)
[1] "numeric"
> charv <- as.character(charv)
> class(charv)
[1] "character"
> #accessing elememts of vector by indexing[]
> sq <- seq(1,4,length.out=5)
> sq[3]
[1] 2.5
> char_vec <-c("aman"=12,"tarun"=14,"garima"=31)
> char_vec
  aman  tarun garima
    12     14     31
> char_vec["tarun"]
tarun
   14
> char_vec[2]
tarun
   14
> #logical vector
> a1 <- c(1,3,4,5,6,7,8)
> a1[c(TRUE,FALSE,FALSE,TRUE,TRUE,FALSE,TRUE)]
[1] 1 5 6 8
> #VECTOR OPERATIONS
```



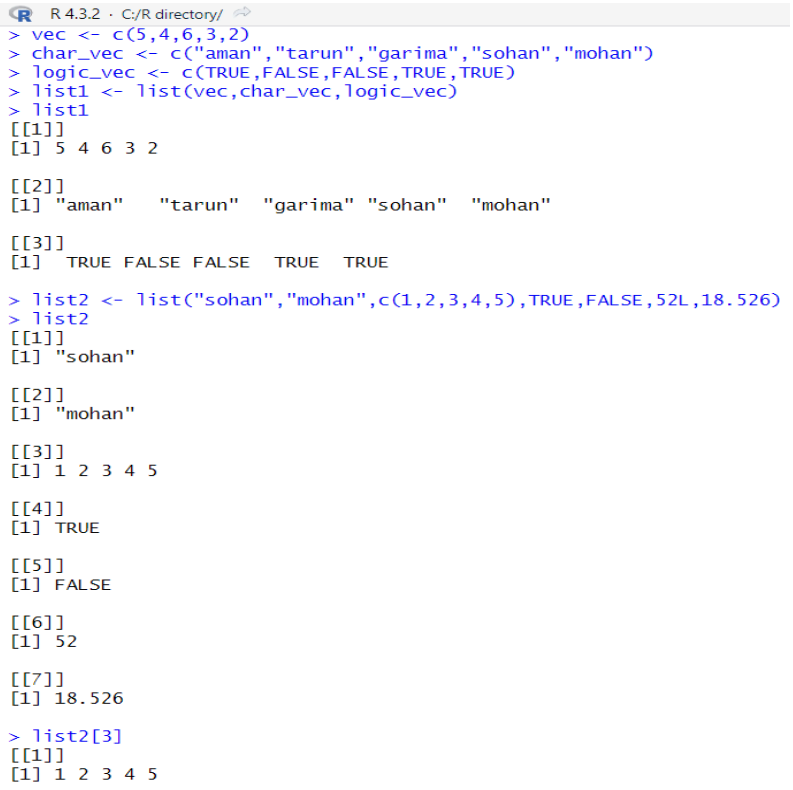
```
> a1 <- c(1,3,4,5,6,7,8)
> a2 <- c("aman","tarun","garima")
> a3 <- c(a1,a2)
> a3
 [1] "1"      "3"      "4"      "5"      "6"      "7"      "8"      "aman"   "tarun"  "garima"
> a2 <- c("aman","tarun","garima")
> #VECTOR OPERATIONS
> a1 <- c(1,3,4,5,6,7,8)
> a4 <- c(5,6,7,8,9,10,2)
> a1+a4
[1]  6  9 11 13 15 17 10
> a1*a4
[1]  5 18 28 40 54 70 16
> a1-a4
[1] -4 -3 -3 -3 -3 -3  6
> a1/a4
[1] 0.2000000 0.5000000 0.5714286 0.6250000 0.6666667 0.7000000 4.0000000
```

ii) List:

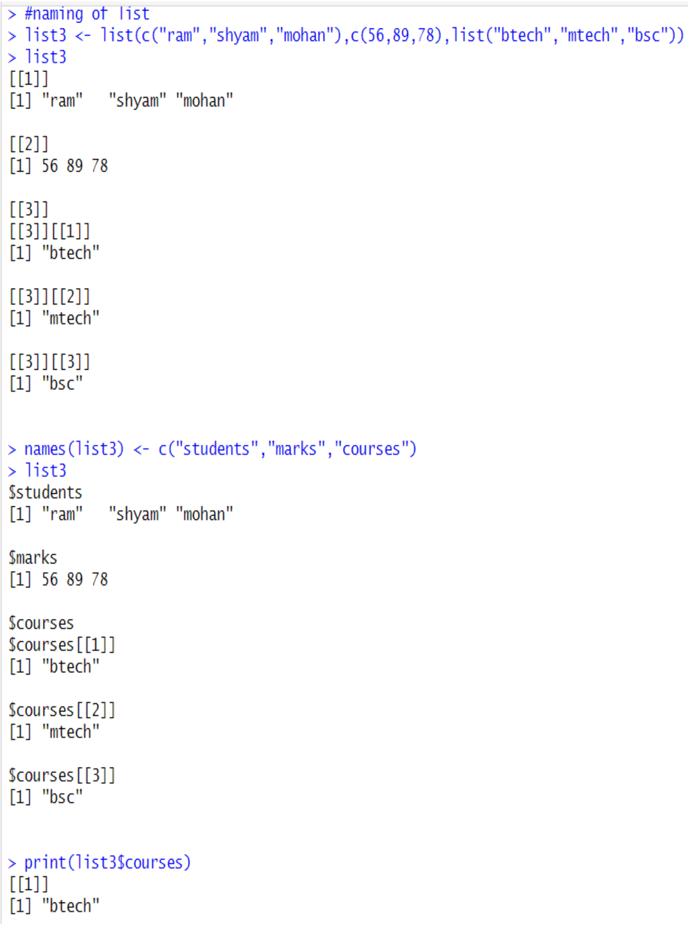
- A list is a versatile data structure that can contain elements of different data types.
- Elements in a list can be vectors, matrices, data frames, or even other lists.
- It is a Collection of Heterogeneous data.
- A List can be created using the list() function, e.g., my_list <- list(1, "hello", c(2.5, 3.0)).



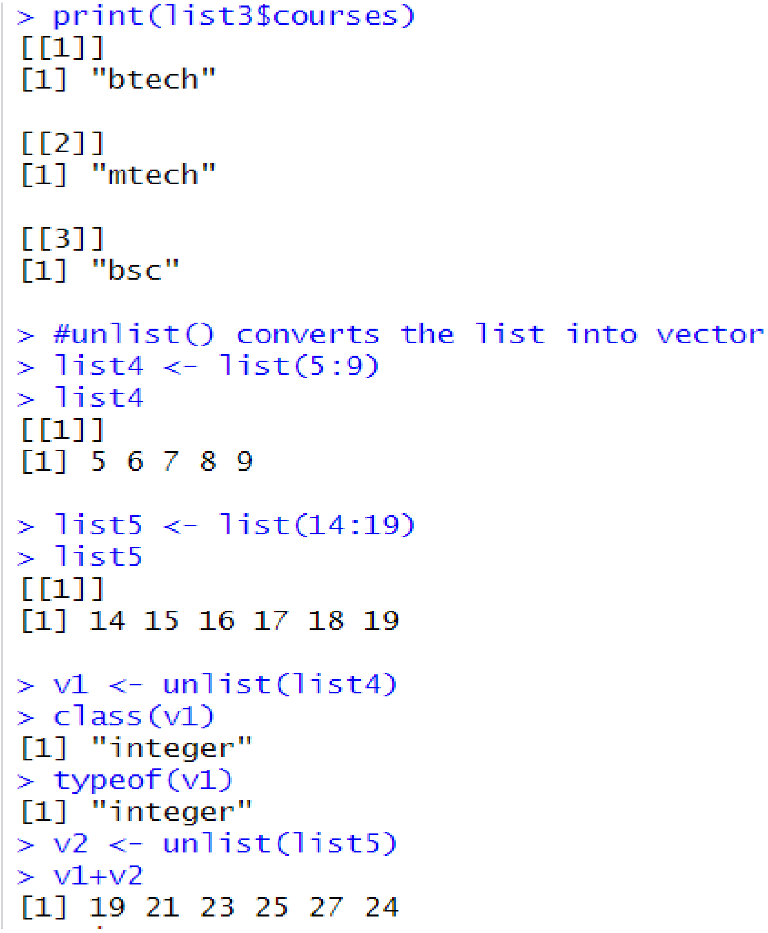
```
#creating the list
vec <- c(5,4,6,3,2)
char_vec <- c("aman","tarun","garima","sohan","mohan")
logic_vec <- c(TRUE,FALSE,FALSE,TRUE,TRUE)
list1 <- list(vec,char_vec,logic_vec)
list1
list2 <- list("sohan","mohan",c(1,2,3,4,5),TRUE,FALSE,52L,18.526)
list2
#accessing elements in list
list2[3]
#naming of list
list3 <- list(c("ram","shyam","mohan"),c(56,89,78),list("btech","mtech","bsc"))
list3
names(list3) <- c("students","marks","courses")
list3
print(list3$courses)
#unlist() converts the list into vector
list4 <- list(5:9)
list4
list5 <- list(14:19)
list5
v1 <- unlist(list4)
v2 <- unlist(list5)
res <- v1+v2
class(v1)
typeof(v1)
```



```
> vec <- c(5,4,6,3,2)
> char_vec <- c("aman","tarun","garima","sohan","mohan")
> logic_vec <- c(TRUE,FALSE,FALSE,TRUE,TRUE)
> list1 <- list(vec,char_vec,logic_vec)
> list1
[[1]]
[1] 5 4 6 3 2

[[2]]
[1] "aman"   "tarun"  "garima" "sohan"  "mohan"

[[3]]
[1]  TRUE FALSE FALSE  TRUE  TRUE

> list2 <- list("sohan","mohan",c(1,2,3,4,5),TRUE,FALSE,52L,18.526)
> list2
[[1]]
[1] "sohan"

[[2]]
[1] "mohan"

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

[[4]]
[1] TRUE

[[5]]
[1] FALSE

[[6]]
[1] 52

[[7]]
[1] 18.526

> list2[3]
[[1]]
[1] 1 2 3 4 5
```



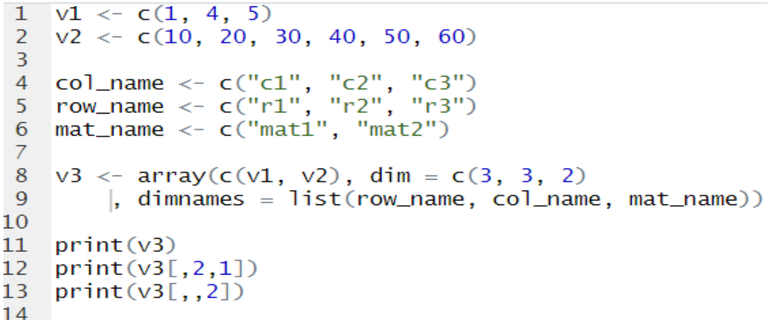
```
> #naming of list
> list3 <- list(c("ram","shyam","mohan"),c(56,89,78),list("btech","mtech","bsc"))
> list3
[[1]]
[1] "ram"   "shyam" "mohan"

[[2]]
[1] 56 89 78

[[3]]
[[3]][[1]]
[1] "btech"

[[3]][[2]]
[1] "mtech"

[[3]][[3]]
[1] "bsc"


> names(list3) <- c("students","marks","courses")
> list3
$students
[1] "ram"   "shyam" "mohan"

$marks
[1] 56 89 78

$courses
$courses[[1]]
[1] "btech"

$courses[[2]]
[1] "mtech"

$courses[[3]]
[1] "bsc"


> print(list3$courses)
[[1]]
[1] "btech"
```



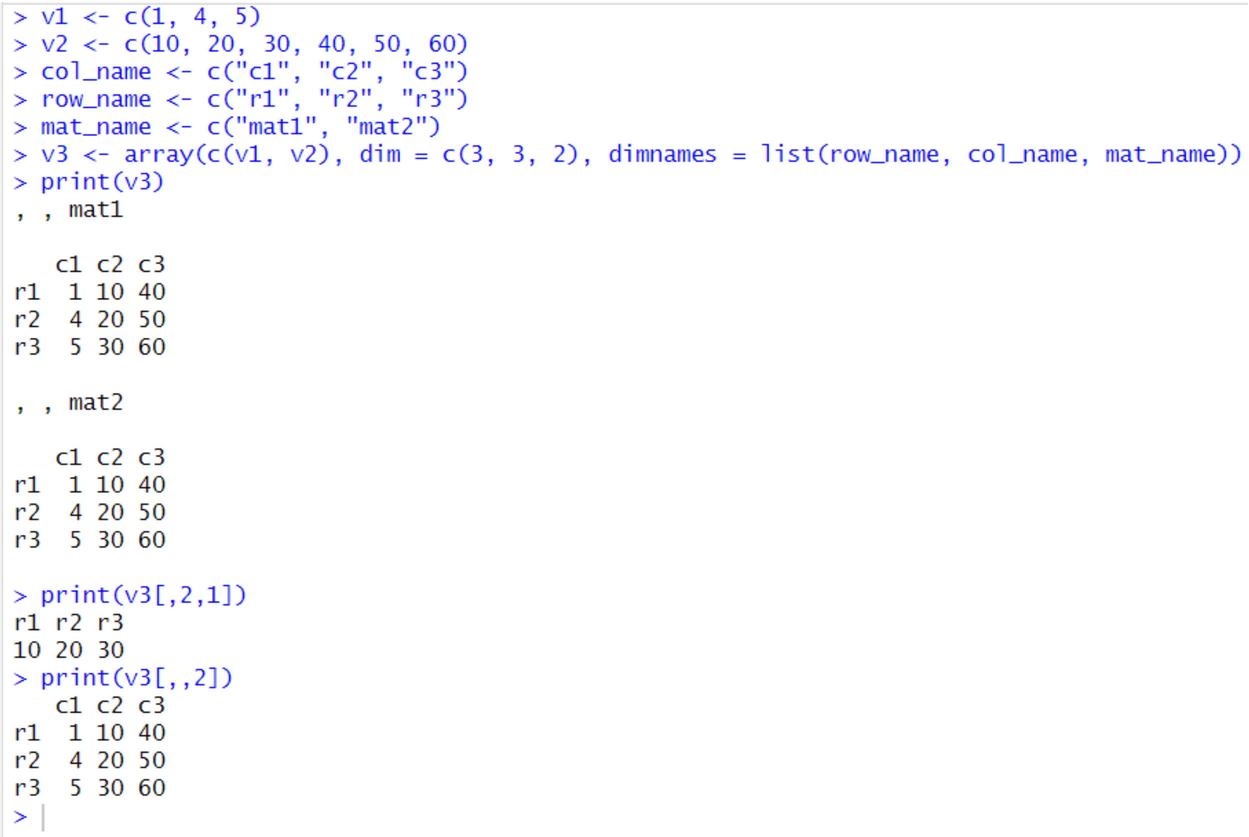
```
> print(list3$courses)
[[1]]
[1] "btech"

[[2]]
[1] "mtech"

[[3]]
[1] "bsc"

> #unlist() converts the list into vector
> list4 <- list(5:9)
> list4
[[1]]
[1] 5 6 7 8 9

> list5 <- list(14:19)
> list5
[[1]]
[1] 14 15 16 17 18 19

> v1 <- unlist(list4)
> class(v1)
[1] "integer"
> typeof(v1)
[1] "integer"
> v2 <- unlist(list5)
> v1+v2
[1] 19 21 23 25 27 24
```

iii) Arrays:

- An array is a multi-dimensional extension of a matrix that can have more than two dimensions.
- An array can be created using the array() function.
- Syntax for creation of an array is:
  variable_name=array(vector,dimension(row,col,no of arrays)
- It can hold elements of the same data type.
- Elements in an array are accessed using square brackets [ ].



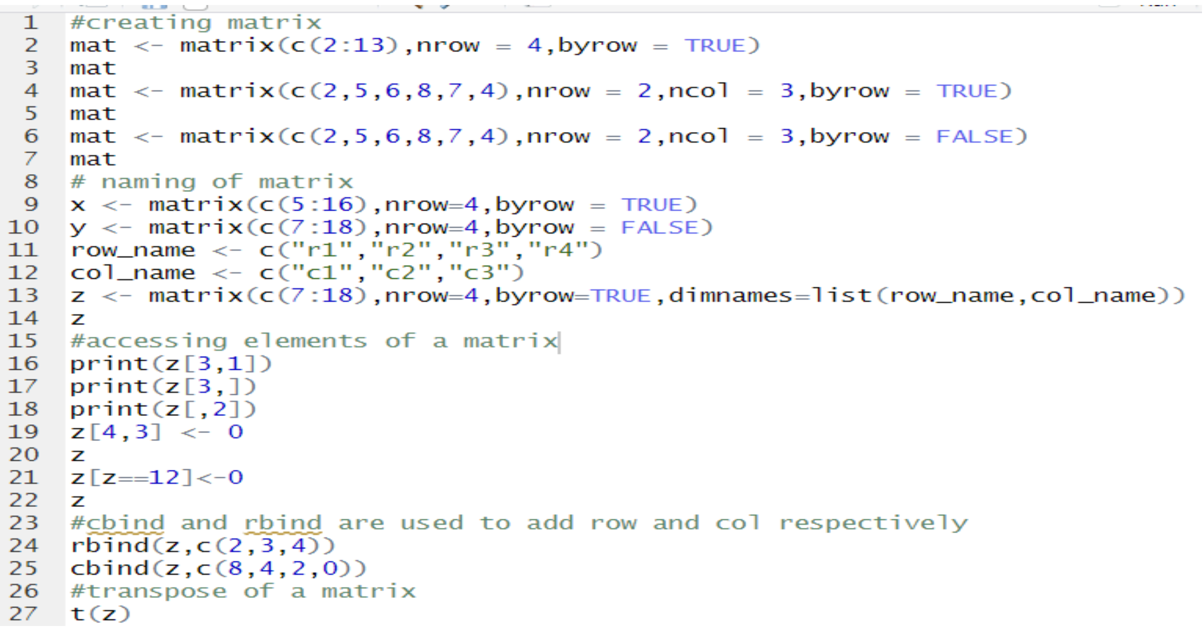
```
v1 <- c(1, 4, 5)
v2 <- c(10, 20, 30, 40, 50, 60)

col_name <- c("c1", "c2", "c3")
row_name <- c("r1", "r2", "r3")
mat_name <- c("mat1", "mat2")

v3 <- array(c(v1, v2), dim = c(3, 3, 2)
    , dimnames = list(row_name, col_name, mat_name))

print(v3)
print(v3[,2,1])
print(v3[,,2])
```



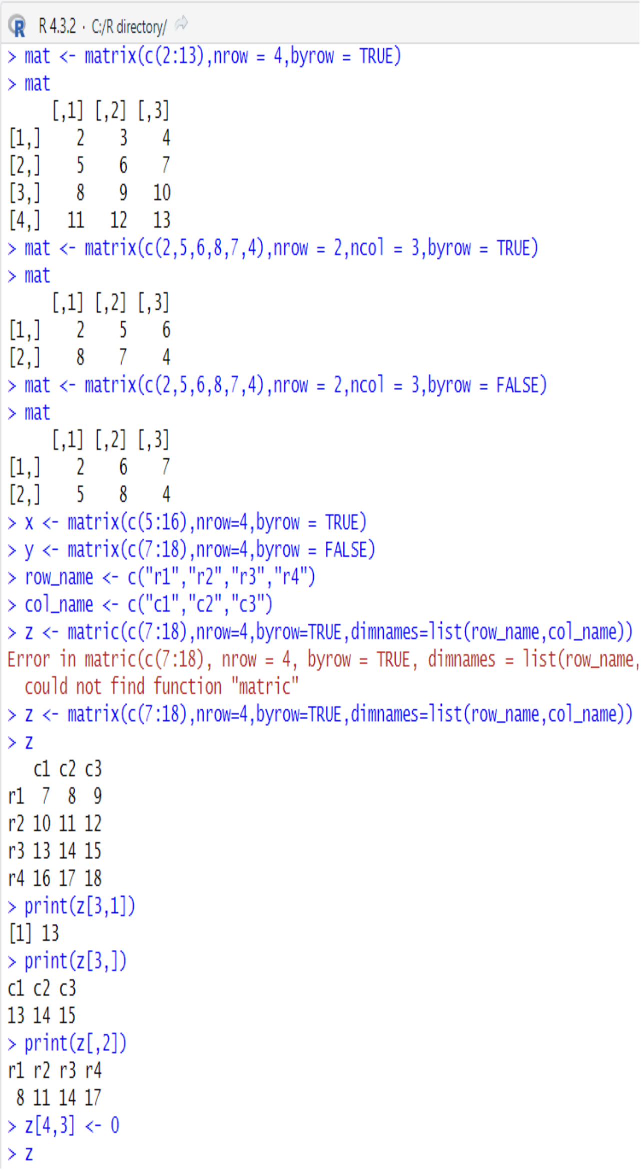
```
> v1 <- c(1, 4, 5)
> v2 <- c(10, 20, 30, 40, 50, 60)
> col_name <- c("c1", "c2", "c3")
> row_name <- c("r1", "r2", "r3")
> mat_name <- c("mat1", "mat2")
> v3 <- array(c(v1, v2), dim = c(3, 3, 2), dimnames = list(row_name, col_name, mat_name))
> print(v3)
, , mat1

   c1 c2 c3
r1  1 10 40
r2  4 20 50
r3  5 30 60

, , mat2

   c1 c2 c3
r1  1 10 40
r2  4 20 50
r3  5 30 60

> print(v3[,2,1])
r1 r2 r3
10 20 30
> print(v3[,,2])
   c1 c2 c3
r1  1 10 40
r2  4 20 50
r3  5 30 60
> 
```

iv) Matrix:

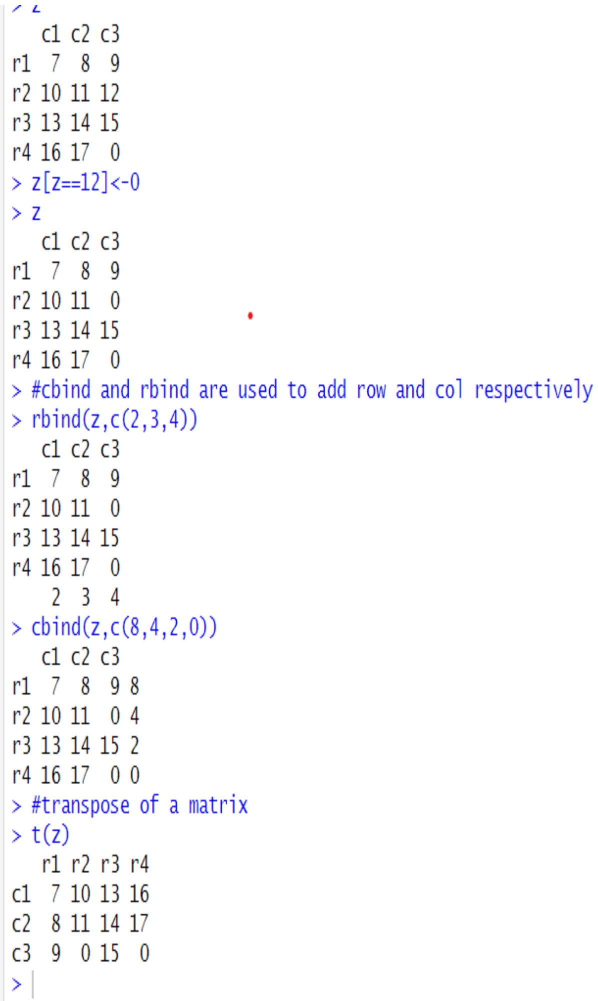
- A matrix is a two-dimensional data structure in R, consisting of rows and columns.
- All elements in a matrix must be of the same data type.
- A matrix can be created using the matrix() function.
- Syntax: matrix(data,nrow,ncol,byrow,dim)



```
#creating matrix
mat <- matrix(c(2:13),nrow = 4,byrow = TRUE)
mat
mat <- matrix(c(2,5,6,8,7,4),nrow = 2,ncol = 3,byrow = TRUE)
mat
mat <- matrix(c(2,5,6,8,7,4),nrow = 2,ncol = 3,byrow = FALSE)
mat
# naming of matrix
x <- matrix(c(5:16),nrow=4,byrow = TRUE)
y <- matrix(c(7:18),nrow=4,byrow = FALSE)
row_name <- c("r1","r2","r3","r4")
col_name <- c("c1","c2","c3")
z <- matrix(c(7:18),nrow=4,byrow=TRUE,dimnames=list(row_name,col_name))
z
#accessing elements of a matrix
print(z[3,1])
print(z[3,])
print(z[,2])
z[4,3] <- 0
z
z[z==12]<-0
z
#cbind and rbind are used to add row and col respectively
rbind(z,c(2,3,4))
cbind(z,c(8,4,2,0))
#transpose of a matrix
t(z)
```



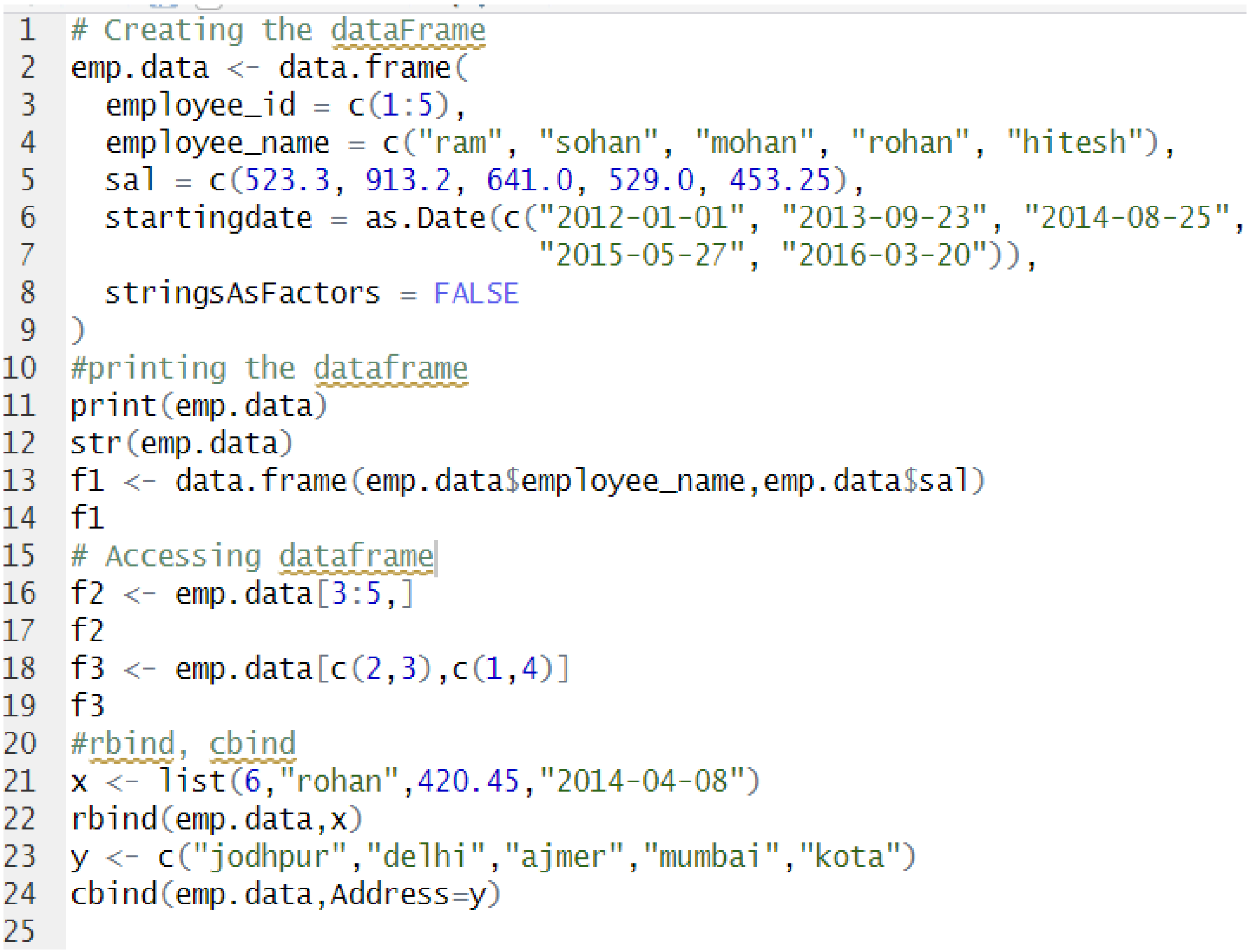
```
> mat <- matrix(c(2:13),nrow = 4,byrow = TRUE)
> mat
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    5    6    7
[3,]    8    9   10
[4,]   11   12   13
> mat <- matrix(c(2,5,6,8,7,4),nrow = 2,ncol = 3,byrow = TRUE)
> mat
     [,1] [,2] [,3]
[1,]    2    5    6
[2,]    8    7    4
> mat <- matrix(c(2,5,6,8,7,4),nrow = 2,ncol = 3,byrow = FALSE)
> mat
     [,1] [,2] [,3]
[1,]    2    6    7
[2,]    5    8    4
> x <- matrix(c(5:16),nrow=4,byrow = TRUE)
> y <- matrix(c(7:18),nrow=4,byrow = FALSE)
> row_name <- c("r1","r2","r3","r4")
> col_name <- c("c1","c2","c3")
> z <- matric(c(7:18),nrow=4,byrow=TRUE,dimnames=list(row_name,col_name))
Error in matric(c(7:18), nrow = 4, byrow = TRUE, dimnames = list(row_name,
  could not find function "matric"
> z <- matrix(c(7:18),nrow=4,byrow=TRUE,dimnames=list(row_name,col_name))
> z
   c1 c2 c3
r1  7  8  9
r2 10 11 12
r3 13 14 15
r4 16 17 18
> print(z[3,1])
[1] 13
> print(z[3,])
c1 c2 c3
13 14 15
> print(z[,2])
r1 r2 r3 r4
 8 11 14 17
> z[4,3] <- 0
> z
```



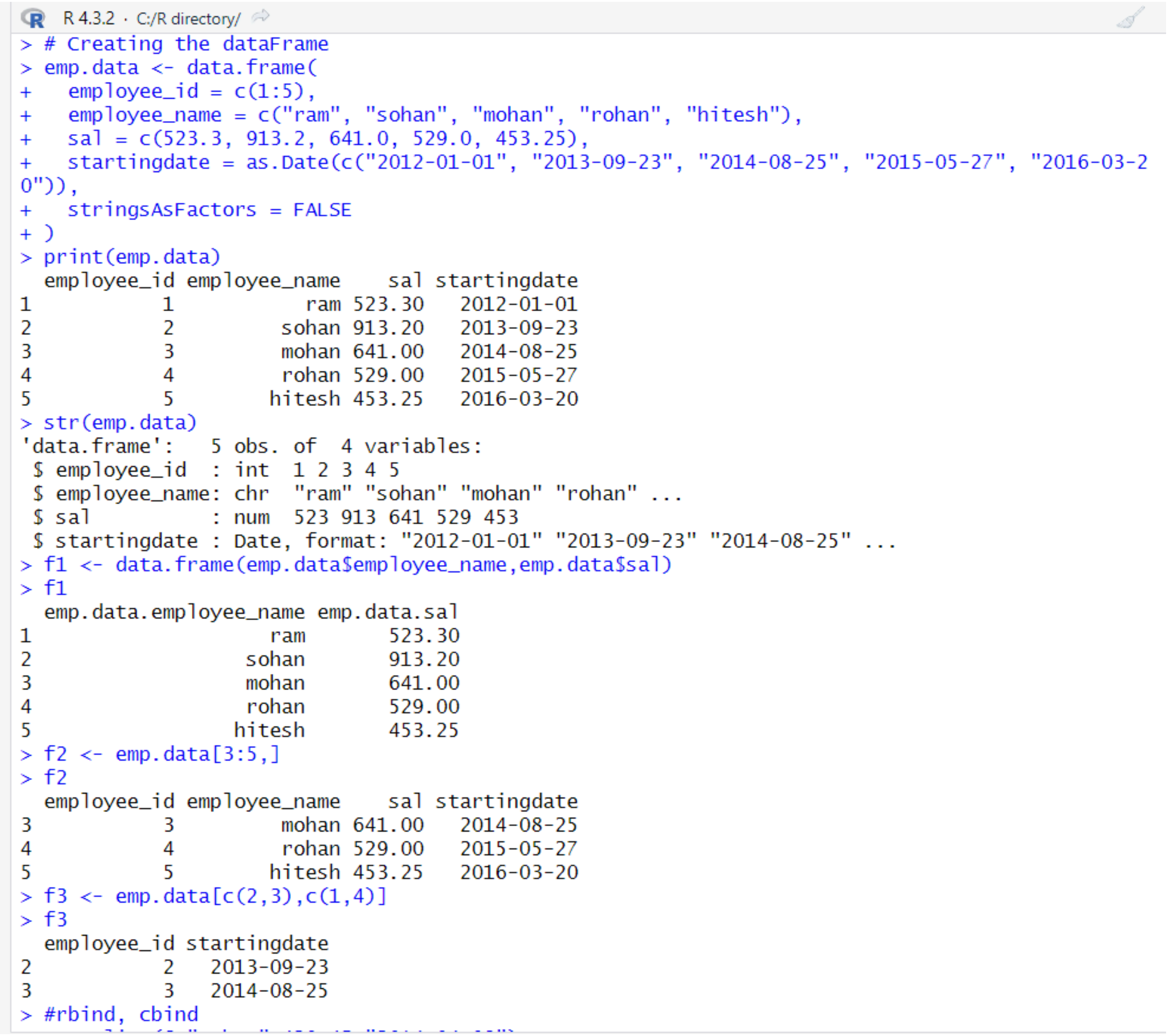
```
> z
   c1 c2 c3
r1  7  8  9
r2 10 11 12
r3 13 14 15
r4 16 17  0
> z[z==12]<-0
> z
   c1 c2 c3
r1  7  8  9
r2 10 11  0
r3 13 14 15
r4 16 17  0
> #cbind and rbind are used to add row and col respectively
> rbind(z,c(2,3,4))
   c1 c2 c3
r1  7  8  9
r2 10 11  0
r3 13 14 15
r4 16 17  0
    2  3  4
> cbind(z,c(8,4,2,0))
   c1 c2 c3
r1  7  8  9 8
r2 10 11  0 4
r3 13 14 15 2
r4 16 17  0 0
> #transpose of a matrix
> t(z)
   r1 r2 r3 r4
c1  7 10 13 16
c2  8 11 14 17
c3  9  0 15  0
>
```

v) DataFrame:

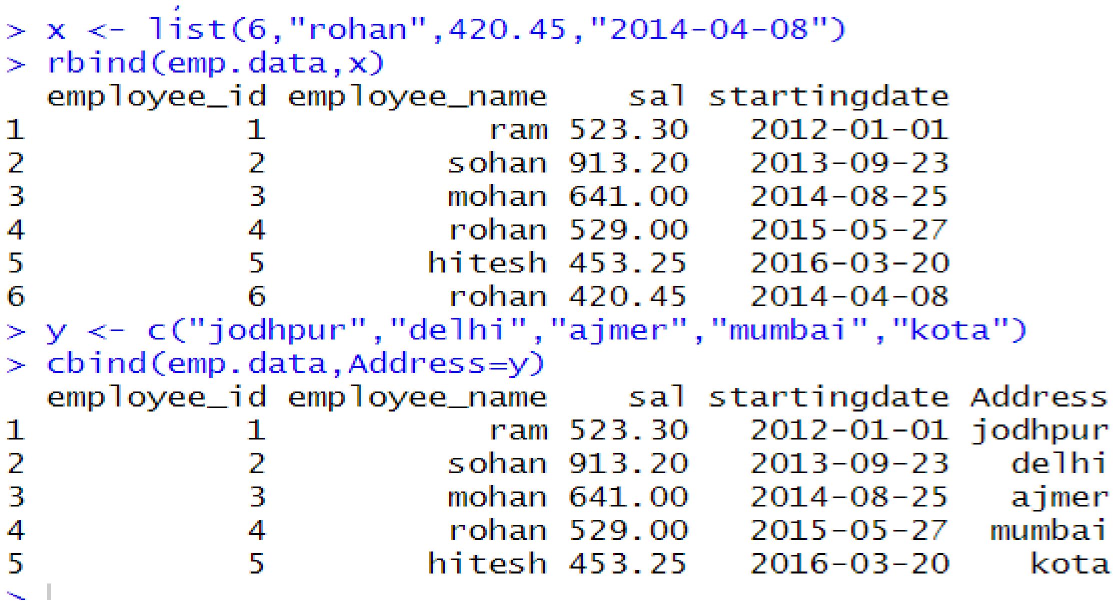
- A data frame is a two-dimensional table-like structure similar to a matrix, but with more flexibility.
- Columns in a data frame can be of different data types.
- Data frames are often used to store datasets.
- You can create a data frame using the data.frame() function.



```r
# Creating the dataFrame
emp.data <- data.frame(
  employee_id = c(1:5),
  employee_name = c("ram", "sohan", "mohan", "rohan", "hitesh"),
  sal = c(523.3, 913.2, 641.0, 529.0, 453.25),
  startingdate = as.Date(c("2012-01-01", "2013-09-23", "2014-08-25",
                           "2015-05-27", "2016-03-20")),
  stringsAsFactors = FALSE
)
#printing the dataframe
print(emp.data)
str(emp.data)
f1 <- data.frame(emp.data$employee_name,emp.data$sal)
f1
# Accessing dataframe
f2 <- emp.data[3:5,]
f2
f3 <- emp.data[c(2,3),c(1,4)]
f3
#rbind, cbind
x <- list(6,"rohan",420.45,"2014-04-08")
rbind(emp.data,x)
y <- c("jodhpur","delhi","ajmer","mumbai","kota")
cbind(emp.data,Address=y)
```

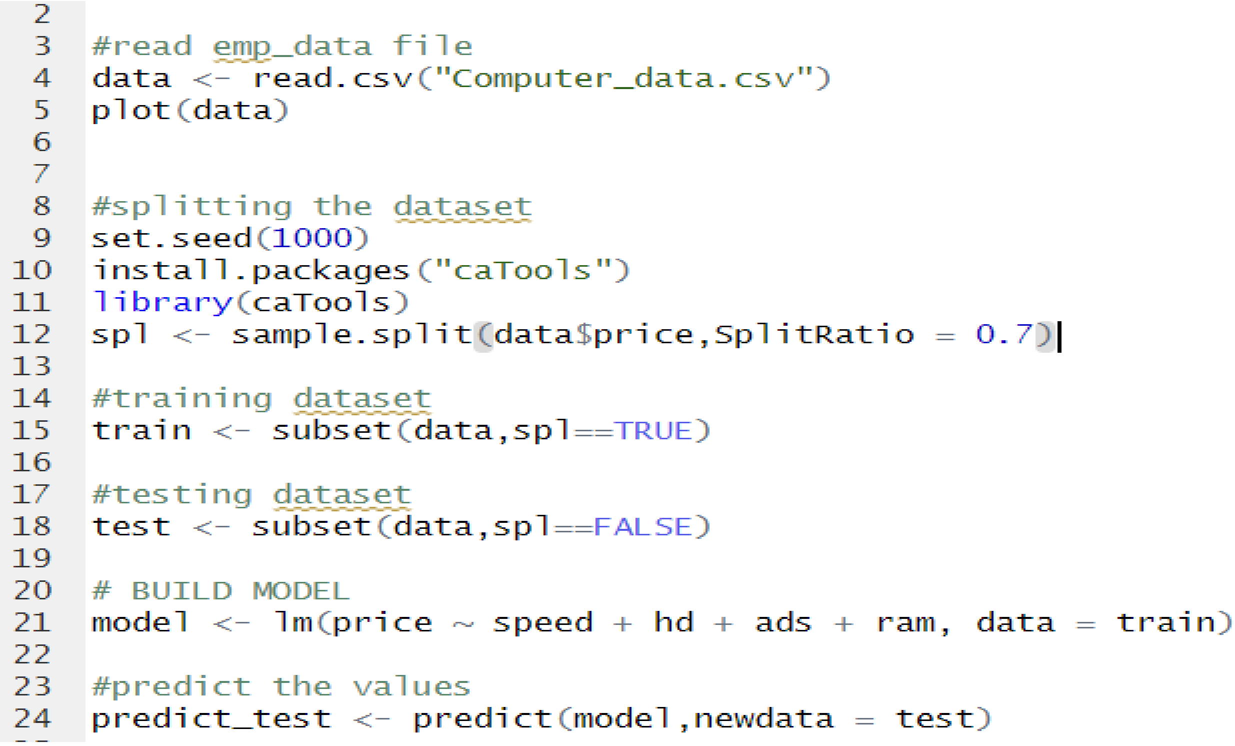


```
> # Creating the dataFrame
> emp.data <- data.frame(
+   employee_id = c(1:5),
+   employee_name = c("ram", "sohan", "mohan", "rohan", "hitesh"),
+   sal = c(523.3, 913.2, 641.0, 529.0, 453.25),
+   startingdate = as.Date(c("2012-01-01", "2013-09-23", "2014-08-25", "2015-05-27", "2016-03-2
0")),
+   stringsAsFactors = FALSE
+ )
> print(emp.data)
  employee_id employee_name    sal startingdate
1           1           ram 523.30   2012-01-01
2           2         sohan 913.20   2013-09-23
3           3         mohan 641.00   2014-08-25
4           4         rohan 529.00   2015-05-27
5           5        hitesh 453.25   2016-03-20
> str(emp.data)
'data.frame':	5 obs. of  4 variables:
 $ employee_id  : int  1 2 3 4 5
 $ employee_name: chr  "ram" "sohan" "mohan" "rohan" ...
 $ sal          : num  523 913 641 529 453
 $ startingdate : Date, format: "2012-01-01" "2013-09-23" "2014-08-25" ...
> f1 <- data.frame(emp.data$employee_name,emp.data$sal)
> f1
  emp.data.employee_name emp.data.sal
1                    ram       523.30
2                  sohan       913.20
3                  mohan       641.00
4                  rohan       529.00
5                 hitesh       453.25
> f2 <- emp.data[3:5,]
> f2
  employee_id employee_name    sal startingdate
3           3         mohan 641.00   2014-08-25
4           4         rohan 529.00   2015-05-27
5           5        hitesh 453.25   2016-03-20
> f3 <- emp.data[c(2,3),c(1,4)]
> f3
  employee_id startingdate
2           2   2013-09-23
3           3   2014-08-25
> #rbind, cbind
```

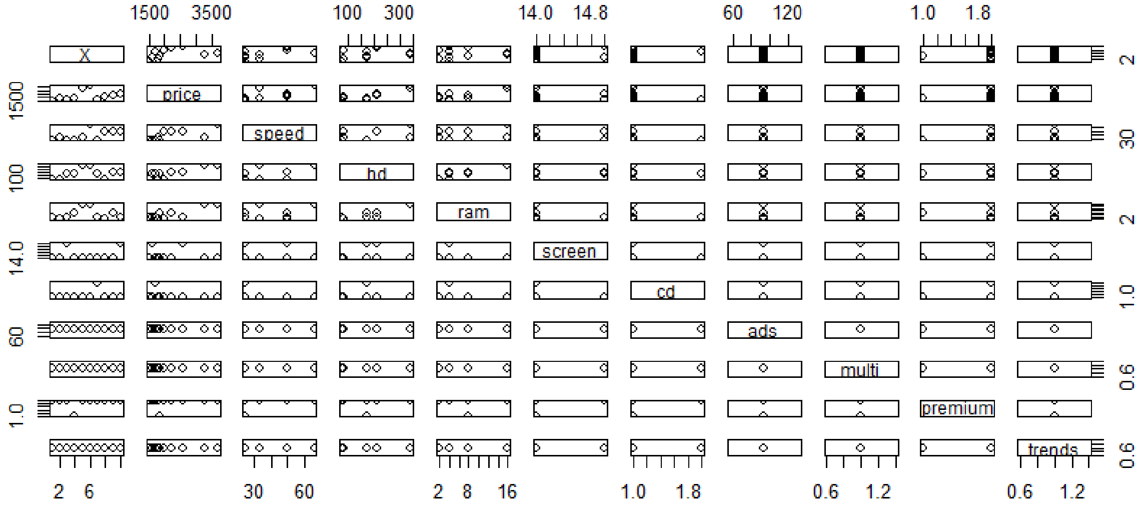


```
> x <- list(6,"rohan",420.45,"2014-04-08")
> rbind(emp.data,x)
  employee_id employee_name    sal startingdate
1           1           ram 523.30   2012-01-01
2           2         sohan 913.20   2013-09-23
3           3         mohan 641.00   2014-08-25
4           4         rohan 529.00   2015-05-27
5           5        hitesh 453.25   2016-03-20
6           6         rohan 420.45   2014-04-08
> y <- c("jodhpur","delhi","ajmer","mumbai","kota")
> cbind(emp.data,Address=y)
  employee_id employee_name    sal startingdate Address
1           1           ram 523.30   2012-01-01 jodhpur
2           2         sohan 913.20   2013-09-23   delhi
3           3         mohan 641.00   2014-08-25   ajmer
4           4         rohan 529.00   2015-05-27  mumbai
5           5        hitesh 453.25   2016-03-20    kota
>
```

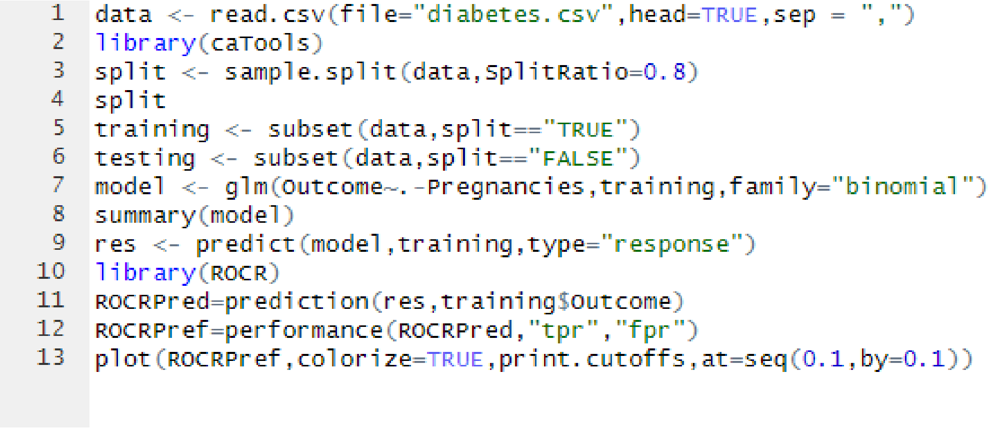
Q2) Implement Linear Regression in R and Visualize the results.



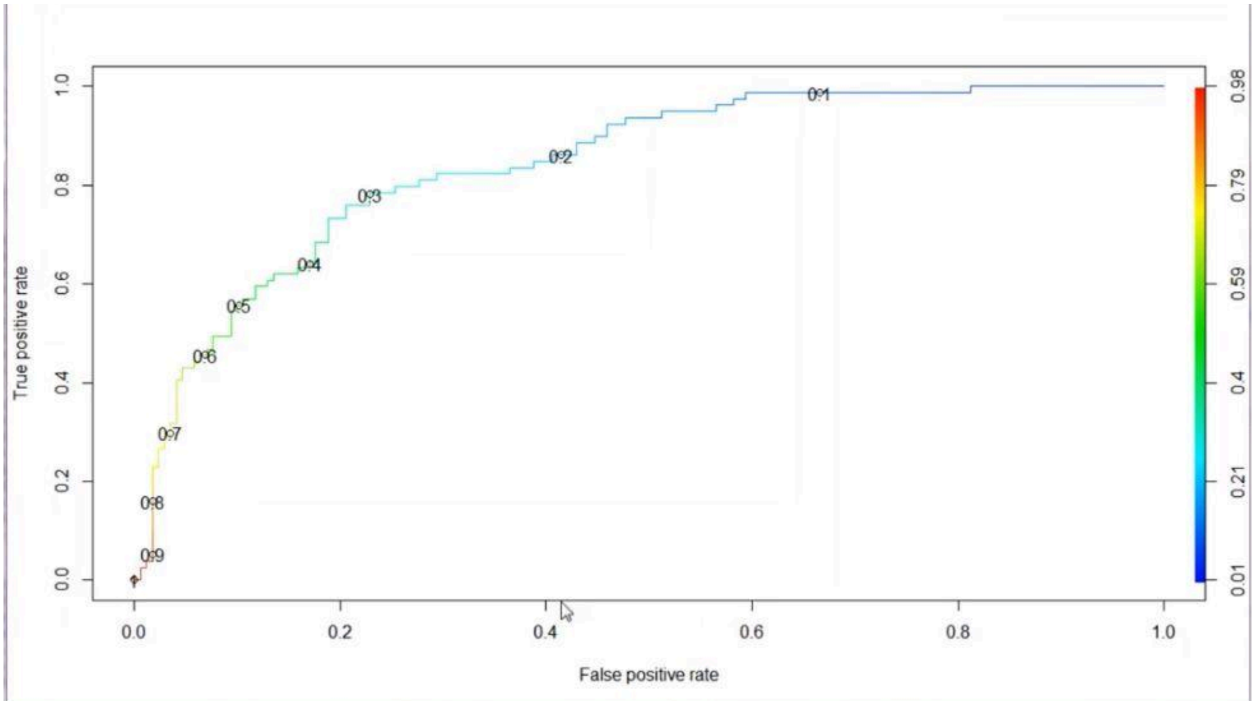
```

#read emp_data file
data <- read.csv("Computer_data.csv")
plot(data)


#splitting the dataset
set.seed(1000)
install.packages("caTools")
library(caTools)
spl <- sample.split(data$price,SplitRatio = 0.7)

#training dataset
train <- subset(data,spl==TRUE)

#testing dataset
test <- subset(data,spl==FALSE)

# BUILD MODEL
model <- lm(price ~ speed + hd + ads + ram, data = train)

#predict the values
predict_test <- predict(model,newdata = test)
```



Q3) Implement Logistic Regression in R and Visualize the results.

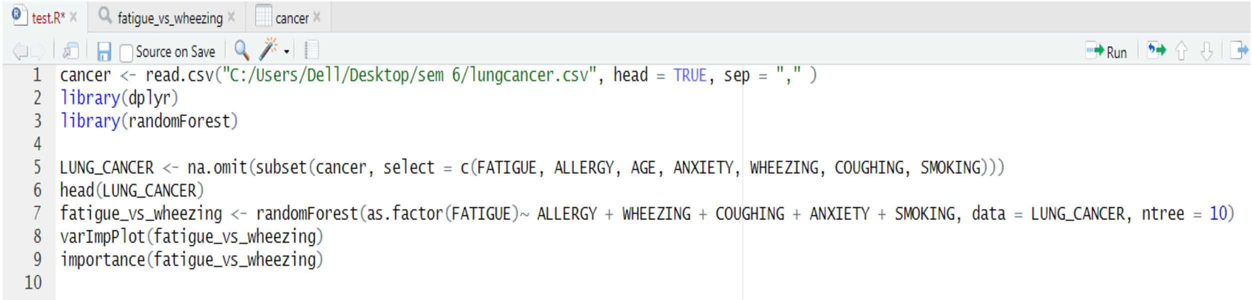


```r
data <- read.csv(file="diabetes.csv",head=TRUE,sep = ",")
library(caTools)
split <- sample.split(data,SplitRatio=0.8)
split
training <- subset(data,split=="TRUE")
testing <- subset(data,split=="FALSE")
model <- glm(Outcome~.-Pregnancies,training,family="binomial")
summary(model)
res <- predict(model,training,type="response")
library(ROCR)
ROCRPred=prediction(res,training$Outcome)
ROCRPref=performance(ROCRPred,"tpr","fpr")
plot(ROCRPref,colorize=TRUE,print.cutoffs,at=seq(0.1,by=0.1))
```



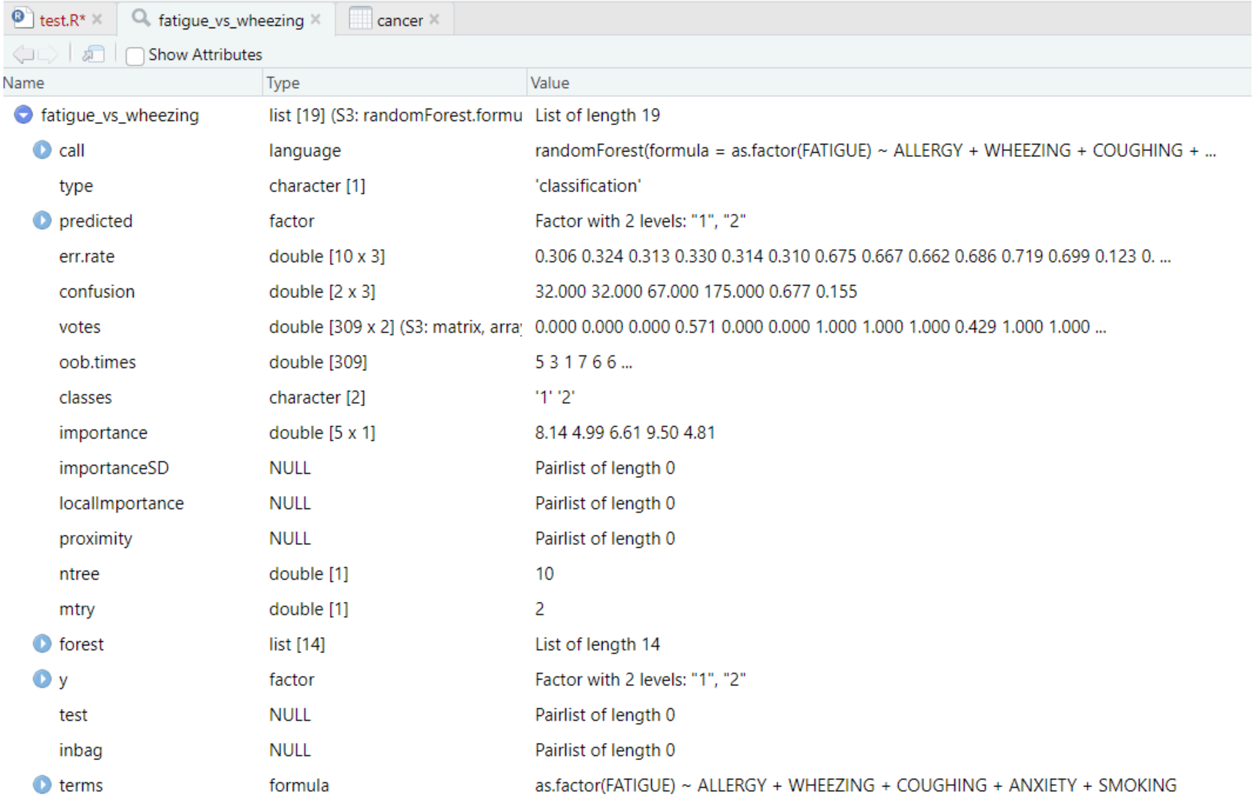
Q4) Implement any Machine learning Algorithm along with feature selection and data visualization.

Using Random Forest :

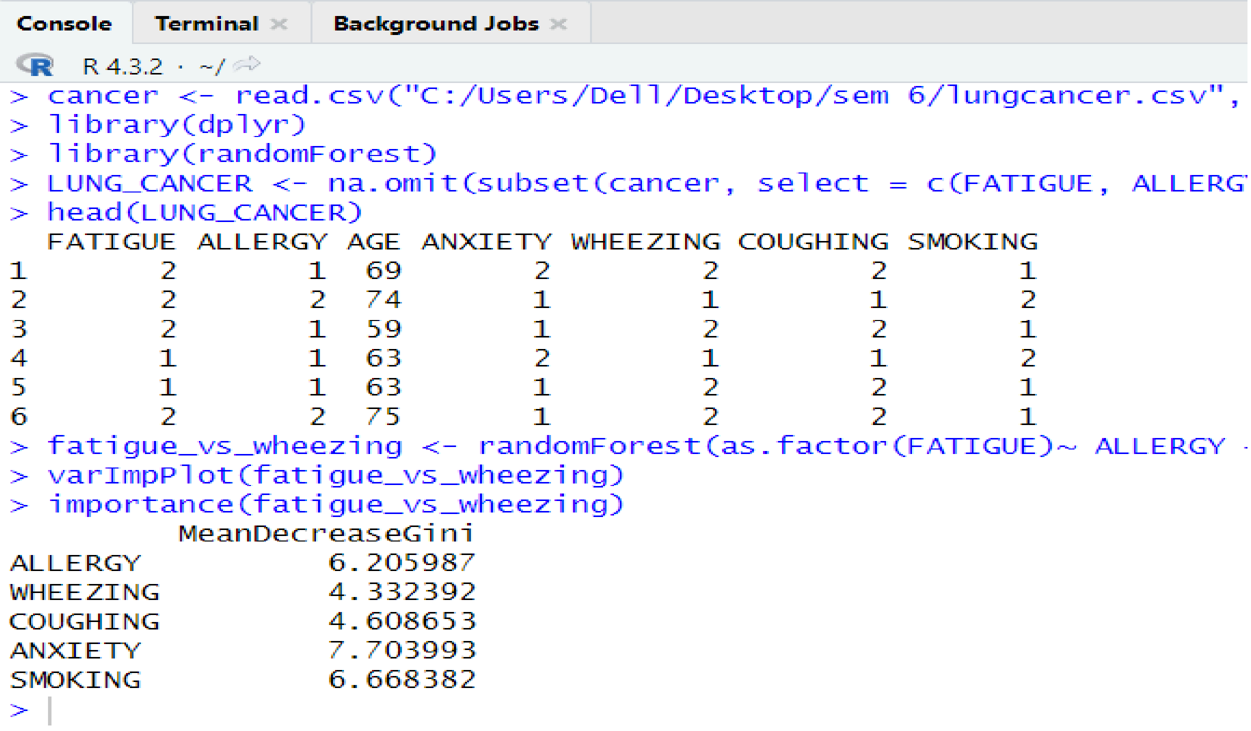


```r
cancer <- read.csv("C:/Users/Dell/Desktop/sem 6/lungcancer.csv", head = TRUE, sep = "," )
library(dplyr)
library(randomForest)

LUNG_CANCER <- na.omit(subset(cancer, select = c(FATIGUE, ALLERGY, AGE, ANXIETY, WHEEZING, COUGHING, SMOKING)))
head(LUNG_CANCER)
fatigue_vs_wheezing <- randomForest(as.factor(FATIGUE)~ ALLERGY + WHEEZING + COUGHING + ANXIETY + SMOKING, data = LUNG_CANCER, ntree = 10)
varImpPlot(fatigue_vs_wheezing)
importance(fatigue_vs_wheezing)
```

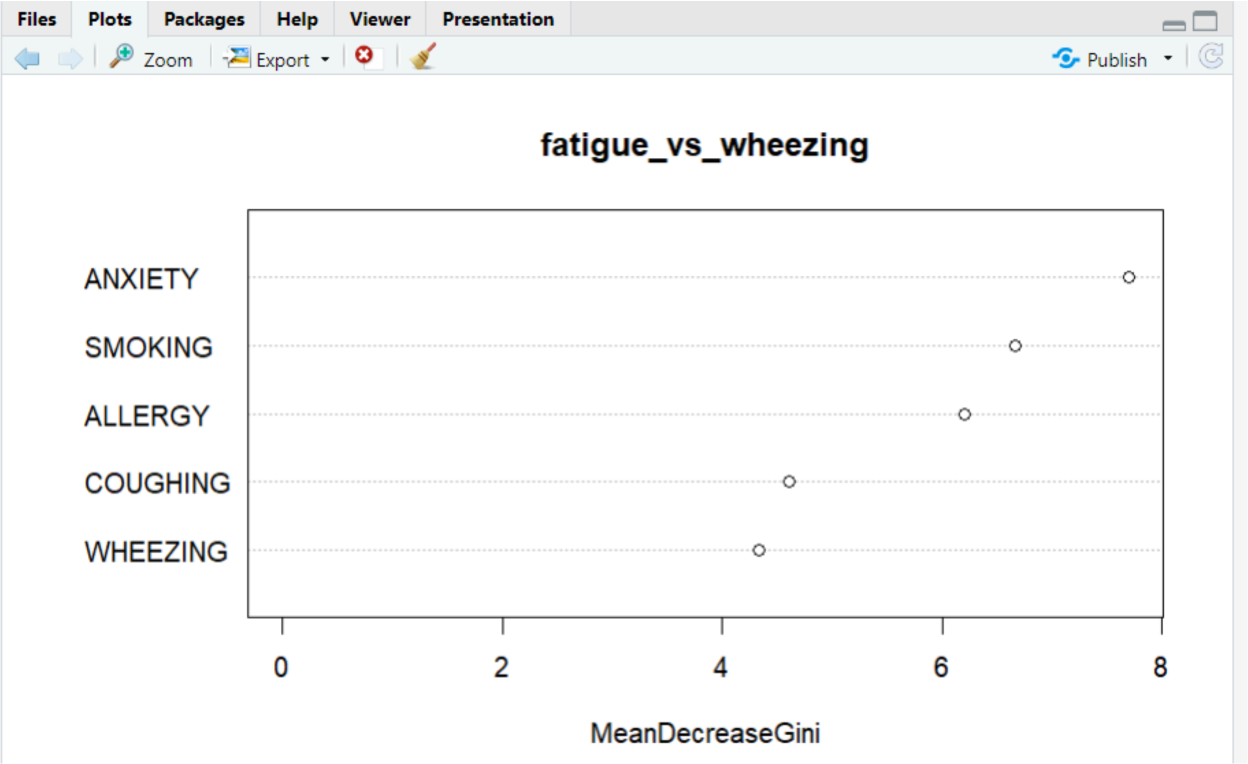
Random Forest Tree



Name	Type	Value
fatigue_vs_wheezing	list [19] (S3: randomForest.formu	List of length 19
call	language	randomForest(formula = as.factor(FATIGUE) ~ ALLERGY + WHEEZING + COUGHING + ...
type	character [1]	'classification'
predicted	factor	Factor with 2 levels: "1", "2"
err.rate	double [10 x 3]	0.306 0.324 0.313 0.330 0.314 0.310 0.675 0.667 0.662 0.686 0.719 0.699 0.123 0. ...
confusion	double [2 x 3]	32.000 32.000 67.000 175.000 0.677 0.155
votes	double [309 x 2] (S3: matrix, array	0.000 0.000 0.000 0.571 0.000 0.000 1.000 1.000 1.000 0.429 1.000 1.000 ...
oob.times	double [309]	5 3 1 7 6 6 ...
classes	character [2]	'1' '2'
importance	double [5 x 1]	8.14 4.99 6.61 9.50 4.81
importanceSD	NULL	Pairlist of length 0
localImportance	NULL	Pairlist of length 0
proximity	NULL	Pairlist of length 0
ntree	double [1]	10
mtry	double [1]	2
forest	list [14]	List of length 14
y	factor	Factor with 2 levels: "1", "2"
test	NULL	Pairlist of length 0
inbag	NULL	Pairlist of length 0
terms	formula	as.factor(FATIGUE) ~ ALLERGY + WHEEZING + COUGHING + ANXIETY + SMOKING



```
> cancer <- read.csv("C:/Users/Dell/Desktop/sem 6/lungcancer.csv",
> library(dplyr)
> library(randomForest)
> LUNG_CANCER <- na.omit(subset(cancer, select = c(FATIGUE, ALLERG
> head(LUNG_CANCER)
  FATIGUE ALLERGY AGE ANXIETY WHEEZING COUGHING SMOKING
1       2       1  69       2        2        2       1
2       2       2  74       1        1        1       2
3       2       1  59       1        2        2       1
4       1       1  63       2        1        1       2
5       1       1  63       1        2        2       1
6       2       2  75       1        2        2       1
> fatigue_vs_wheezing <- randomForest(as.factor(FATIGUE)~ ALLERGY 
> varImpPlot(fatigue_vs_wheezing)
> importance(fatigue_vs_wheezing)
         MeanDecreaseGini
ALLERGY          6.205987
WHEEZING         4.332392
COUGHING         4.608653
ANXIETY          7.703993
SMOKING          6.668382
> 
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
fatigue_vs_wheezing
ANXIETY
SMOKING
ALLERGY
COUGHING
WHEEZING
MeanDecreaseGini