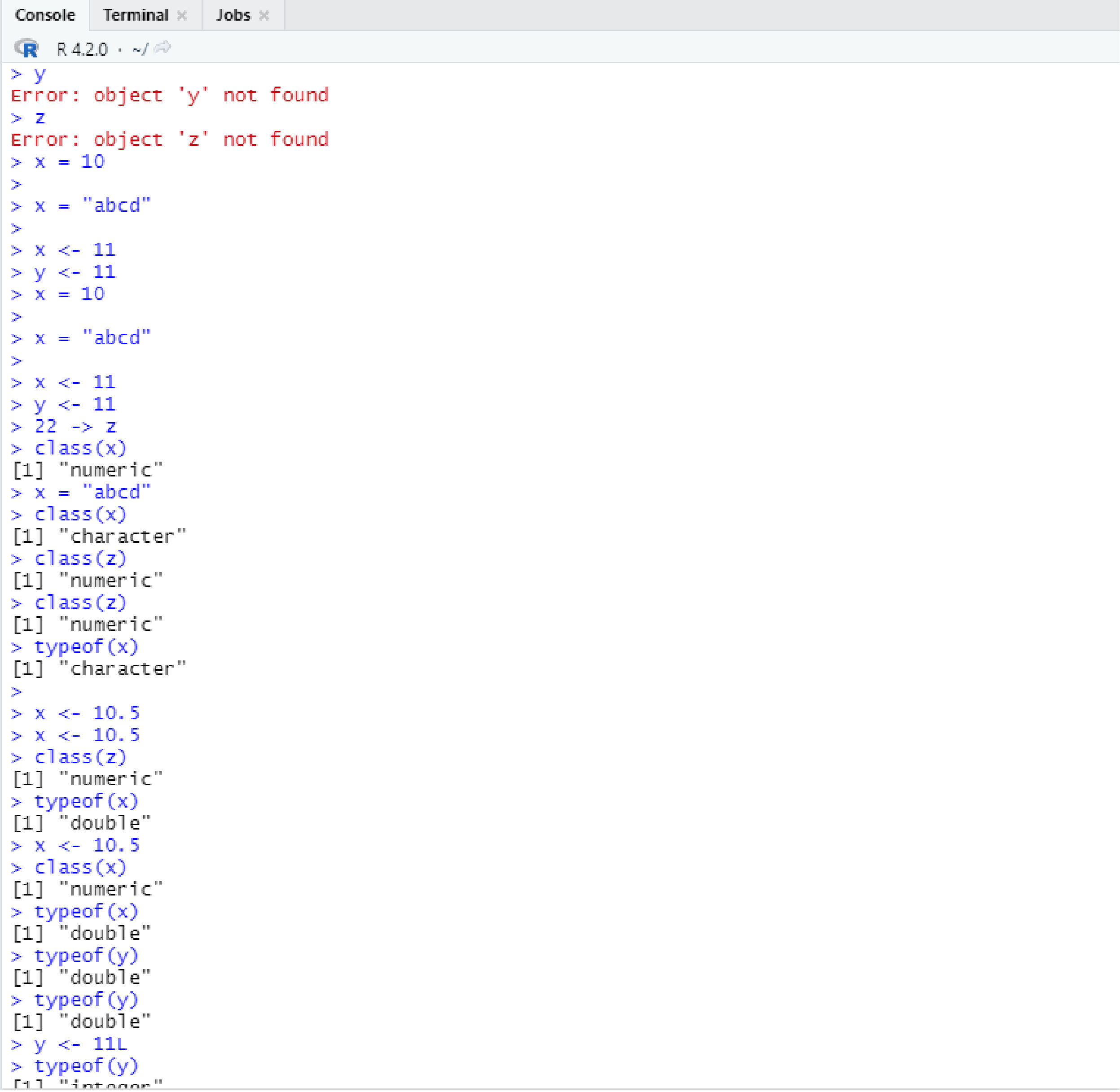
RMySQL
Plotrix
Aplpack
Rvest
dplyr

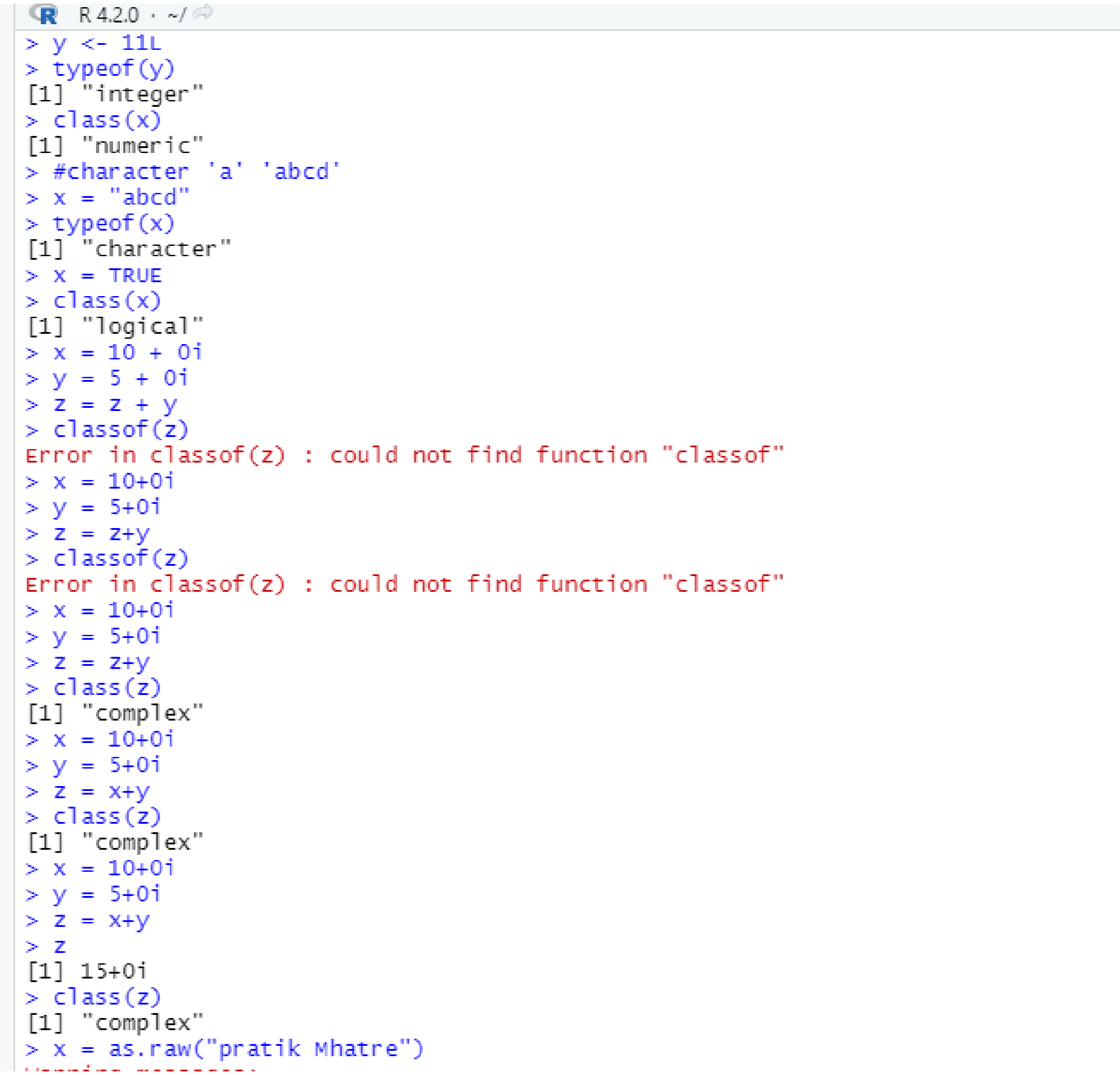
Program:

```
#get current working directory
#get current working directory
getwd()

#set current working directory using command
setwd("D:/aaaaaPratik/R workspace")

#set current working directory using -> set working dir menu option

#R programming doesnt support multiline comment
#trick for using multiline comment
if(False){
  "this is my multiline comment"
  "i hope this works"
}
#end of multiline comment

library(xlsx)
library(package)

#user of (var_name <- value) and value -> var_name operators for assigning
x = 10

x = "abcd"

x <- 11

#integer dataytpes requires 'L'
y <- 11L
typeof(y)

22 -> z

x <- 10.5
class(x)
typeof(y)


class(x)
#character 'a' 'abcd'
x = "abcd"
```

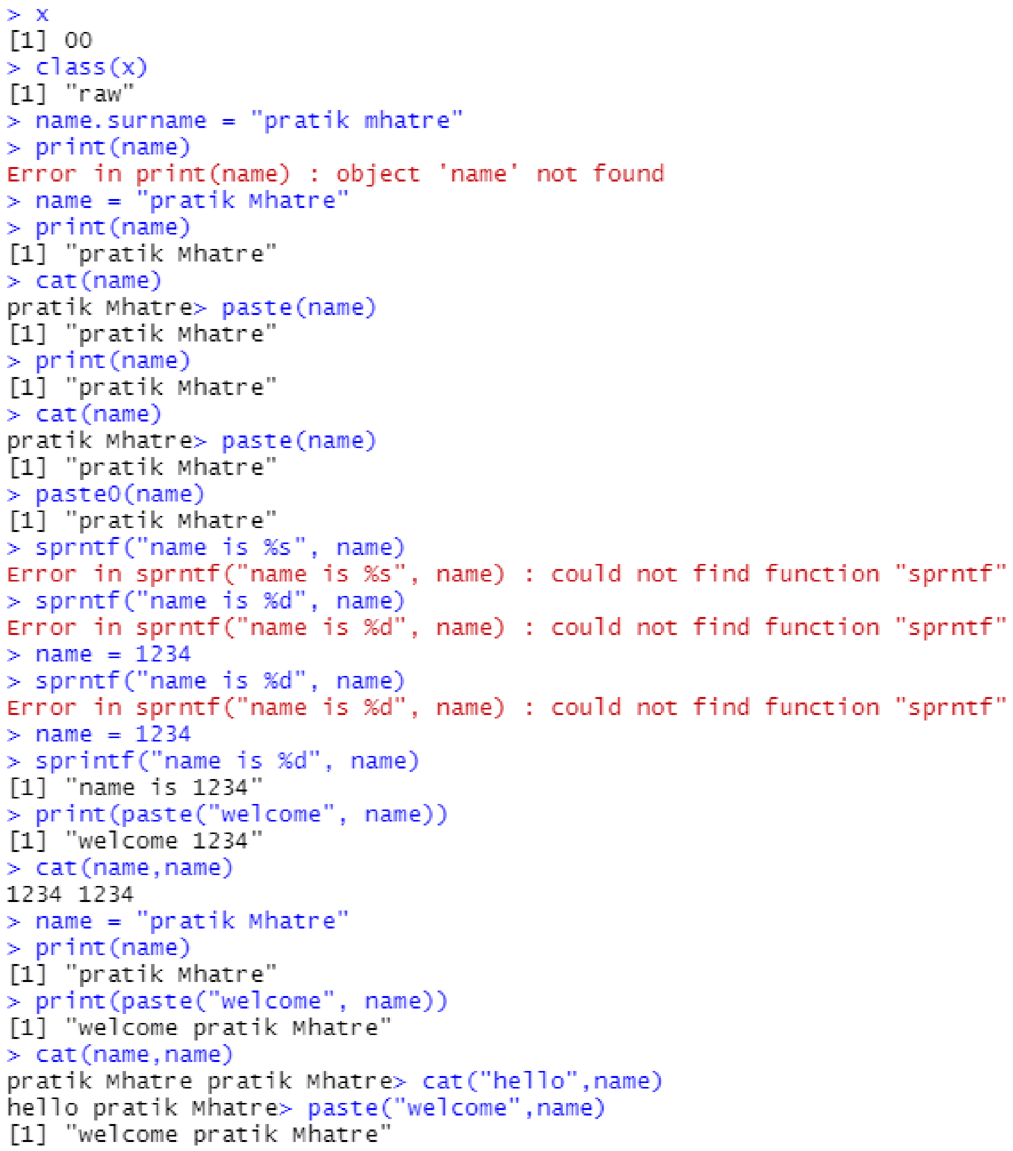
```
typeof(x)

#logical datatype true or false
x = TRUE
class(x)

#complex type
x = 10+0i
y = 5+0i
z = x+y
z
class(z)

#raw type
x = as.raw("pratik Mhatre")
x
class(x)

name.surname = "pratik mhatre"

#output functions in R
name = "pratik Mhatre"
print(name)
print(paste("welcome", name))
cat(name,name)
cat("hello",name)
paste("welcome",name)
paste0("welcome",name)

#d - integer
#s - string
#c - character

name = 1234
sprintf("name is %d", name)
```



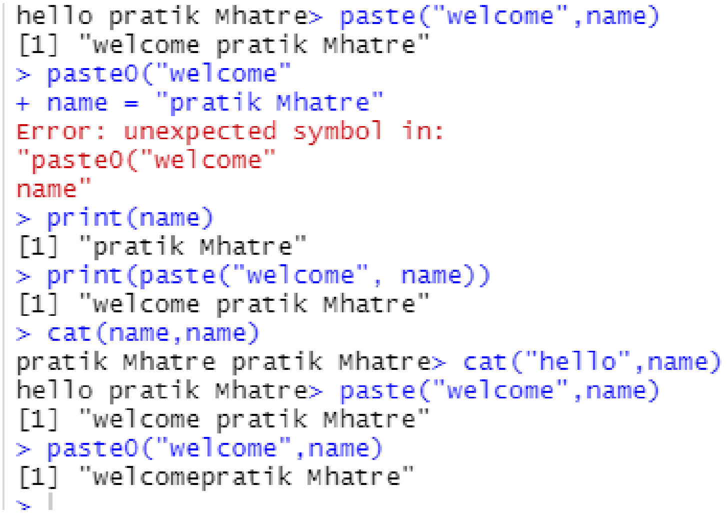
```
> y
Error: object 'y' not found
> z
Error: object 'z' not found
> x = 10
>
> x = "abcd"
>
> x <- 11
> y <- 11
> x = 10
>
> x = "abcd"
>
> x <- 11
> y <- 11
> 22 -> z
> class(x)
[1] "numeric"
> x = "abcd"
> class(x)
[1] "character"
> class(z)
[1] "numeric"
> class(z)
[1] "numeric"
> typeof(x)
[1] "character"
>
> x <- 10.5
> x <- 10.5
> class(z)
[1] "numeric"
> typeof(x)
[1] "double"
> x <- 10.5
> class(x)
[1] "numeric"
> typeof(x)
[1] "double"
> typeof(y)
[1] "double"
> typeof(y)
[1] "double"
> y <- 11L
> typeof(y)
[1] "integer"
```



```
> y <- 11L
> typeof(y)
[1] "integer"
> class(x)
[1] "numeric"
> #character 'a' 'abcd'
> x = "abcd"
> typeof(x)
[1] "character"
> x = TRUE
> class(x)
[1] "logical"
> x = 10 + 0i
> y = 5 + 0i
> z = z + y
> classof(z)
Error in classof(z) : could not find function "classof"
> x = 10+0i
> y = 5+0i
> z = z+y
> classof(z)
Error in classof(z) : could not find function "classof"
> x = 10+0i
> y = 5+0i
> z = z+y
> class(z)
[1] "complex"
> x = 10+0i
> y = 5+0i
> z = x+y
> class(z)
[1] "complex"
> x = 10+0i
> y = 5+0i
> z = x+y
> z
[1] 15+0i
> class(z)
[1] "complex"
> x = as.raw("pratik Mhatre")
Warning messages:
```

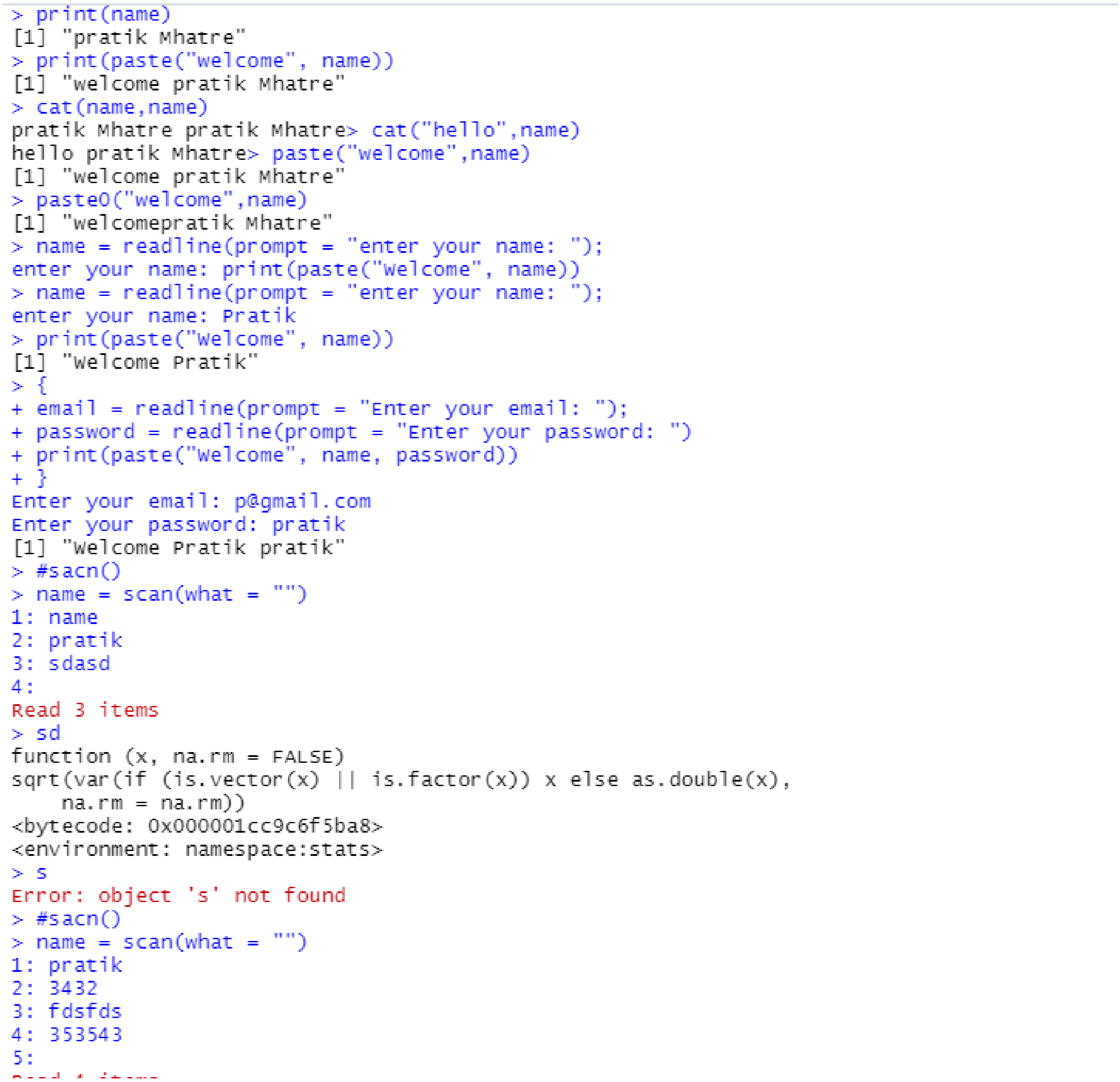


```
> x
[1] 00
> class(x)
[1] "raw"
> name.surname = "pratik mhatre"
> print(name)
Error in print(name) : object 'name' not found
> name = "pratik Mhatre"
> print(name)
[1] "pratik Mhatre"
> cat(name)
pratik Mhatre> paste(name)
[1] "pratik Mhatre"
> print(name)
[1] "pratik Mhatre"
> cat(name)
pratik Mhatre> paste(name)
[1] "pratik Mhatre"
> paste0(name)
[1] "pratik Mhatre"
> sprntf("name is %s", name)
Error in sprntf("name is %s", name) : could not find function "sprntf"
> sprntf("name is %d", name)
Error in sprntf("name is %d", name) : could not find function "sprntf"
> name = 1234
> sprntf("name is %d", name)
Error in sprntf("name is %d", name) : could not find function "sprntf"
> name = 1234
> sprintf("name is %d", name)
[1] "name is 1234"
> print(paste("welcome", name))
[1] "welcome 1234"
> cat(name,name)
1234 1234
> name = "pratik Mhatre"
> print(name)
[1] "pratik Mhatre"
> print(paste("welcome", name))
[1] "welcome pratik Mhatre"
> cat(name,name)
pratik Mhatre pratik Mhatre> cat("hello",name)
hello pratik Mhatre> paste("welcome",name)
[1] "welcome pratik Mhatre"
```

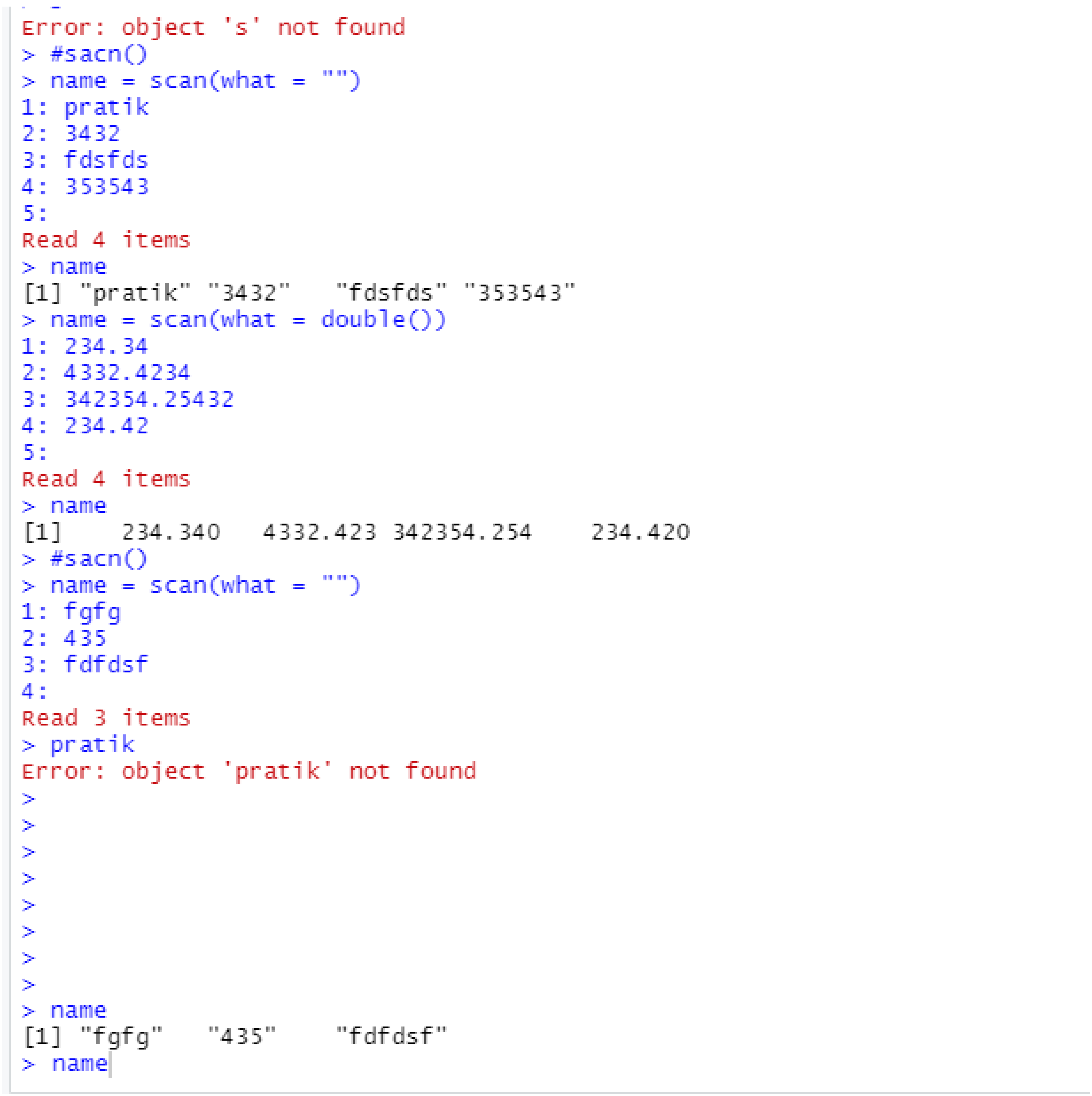


```
hello pratik Mhatre> paste("welcome",name)
[1] "welcome pratik Mhatre"
> paste0("welcome"
+ name = "pratik Mhatre"
Error: unexpected symbol in:
"paste0("welcome"
name"
> print(name)
[1] "pratik Mhatre"
> print(paste("welcome", name))
[1] "welcome pratik Mhatre"
> cat(name,name)
pratik Mhatre pratik Mhatre> cat("hello",name)
hello pratik Mhatre> paste("welcome",name)
[1] "welcome pratik Mhatre"
> paste0("welcome",name)
[1] "welcomepratik Mhatre"
> 
```

```r
#accepting user input from console
#prompt only needs semi colon in R
name = readline(prompt = "enter your name: ");
print(paste("Welcome", name))
{
email = readline(prompt = "Enter your email: ");
password = readline(prompt = "Enter your password: ")
print(paste("Welcome", name, password))
}

#sacn()
name = scan(what = "")
name = scan(what = double())
```



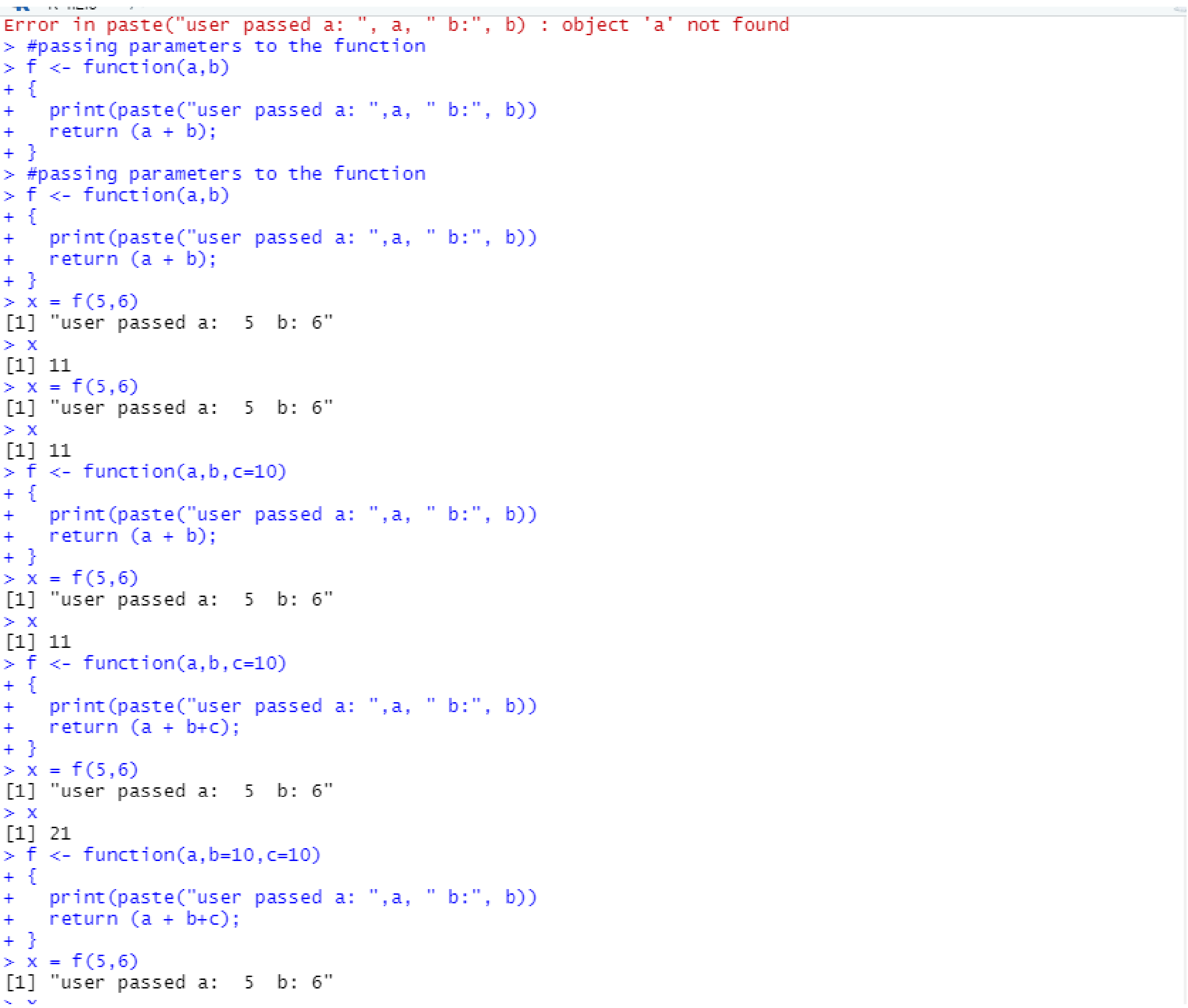
```
> print(name)
[1] "pratik Mhatre"
> print(paste("welcome", name))
[1] "welcome pratik Mhatre"
> cat(name,name)
pratik Mhatre pratik Mhatre> cat("hello",name)
hello pratik Mhatre> paste("welcome",name)
[1] "welcome pratik Mhatre"
> paste0("welcome",name)
[1] "welcomepratik Mhatre"
> name = readline(prompt = "enter your name: ");
enter your name: print(paste("Welcome", name))
> name = readline(prompt = "enter your name: ");
enter your name: Pratik
> print(paste("Welcome", name))
[1] "Welcome Pratik"
> {
+ email = readline(prompt = "Enter your email: ");
+ password = readline(prompt = "Enter your password: ")
+ print(paste("Welcome", name, password))
+ }
Enter your email: p@gmail.com
Enter your password: pratik
[1] "Welcome Pratik pratik"
> #sacn()
> name = scan(what = "")
1: name
2: pratik
3: sdasd
4: 
Read 3 items
> sd
function (x, na.rm = FALSE) 
sqrt(var(if (is.vector(x) || is.factor(x)) x else as.double(x), 
    na.rm = na.rm))
<bytecode: 0x000001cc9c6f5ba8>
<environment: namespace:stats>
> s
Error: object 's' not found
> #sacn()
> name = scan(what = "")
1: pratik
2: 3432
3: fdsfds
4: 353543
5: 
Read 4 items
```
]



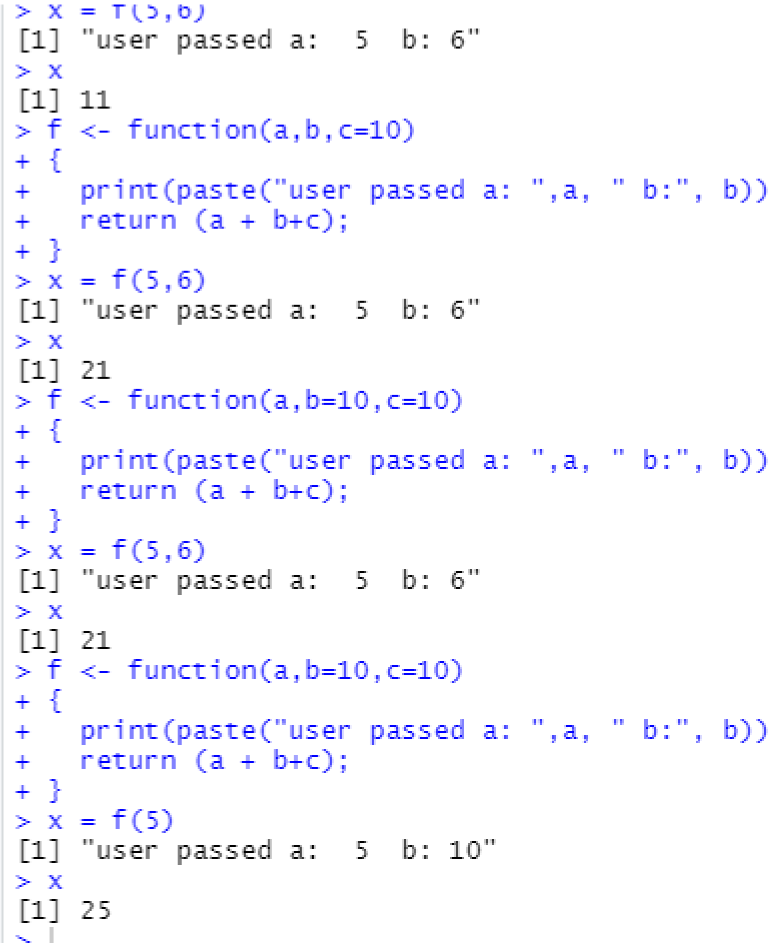
```
Error: object 's' not found
> #sacn()
> name = scan(what = "")
1: pratik
2: 3432
3: fdsfds
4: 353543
5: 
Read 4 items
> name
[1] "pratik" "3432"   "fdsfds" "353543"
> name = scan(what = double())
1: 234.34
2: 4332.4234
3: 342354.25432
4: 234.42
5: 
Read 4 items
> name
[1]    234.340   4332.423 342354.254    234.420
> #sacn()
> name = scan(what = "")
1: fgfg
2: 435
3: fdfdsf
4: 
Read 3 items
> pratik
Error: object 'pratik' not found
> 
> 
> 
> 
> 
> 
> 
> 
> name
[1] "fgfg"   "435"    "fdfdsf"
> name
```

```
#functions in R
#built- in and user defined

# user defined functions
f <- function()
{
  print("normal user defined function")
}
f

#passing parameters to the function
```

```
#function returning value
f <- function(a,b)
{
  print(paste("user passed a: ",a, " b:", b))
  return (a + b);
}
x = f(5,6)
x

f <- function(a,b=10,c=10)
{
  print(paste("user passed a: ",a, " b:", b))
  return (a + b+c);
}
x = f(5)
x
```



```
Error in paste("user passed a: ", a, " b:", b) : object 'a' not found
> #passing parameters to the function
> f <- function(a,b)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b);
+ }
> #passing parameters to the function
> f <- function(a,b)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b);
+ }
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 11
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 11
> f <- function(a,b,c=10)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b);
+ }
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 11
> f <- function(a,b,c=10)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b+c);
+ }
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 21
> f <- function(a,b=10,c=10)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b+c);
+ }
> x = f(5,6)
[1] "user passed a:  5  b: 6"
```



```
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 11
> f <- function(a,b,c=10)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b+c);
+ }
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 21
> f <- function(a,b=10,c=10)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b+c);
+ }
> x = f(5,6)
[1] "user passed a:  5  b: 6"
> x
[1] 21
> f <- function(a,b=10,c=10)
+ {
+   print(paste("user passed a: ",a, " b:", b))
+   return (a + b+c);
+ }
> x = f(5)
[1] "user passed a:  5  b: 10"
> x
[1] 25
> 
```

Functions and Data Structures of R

Program:

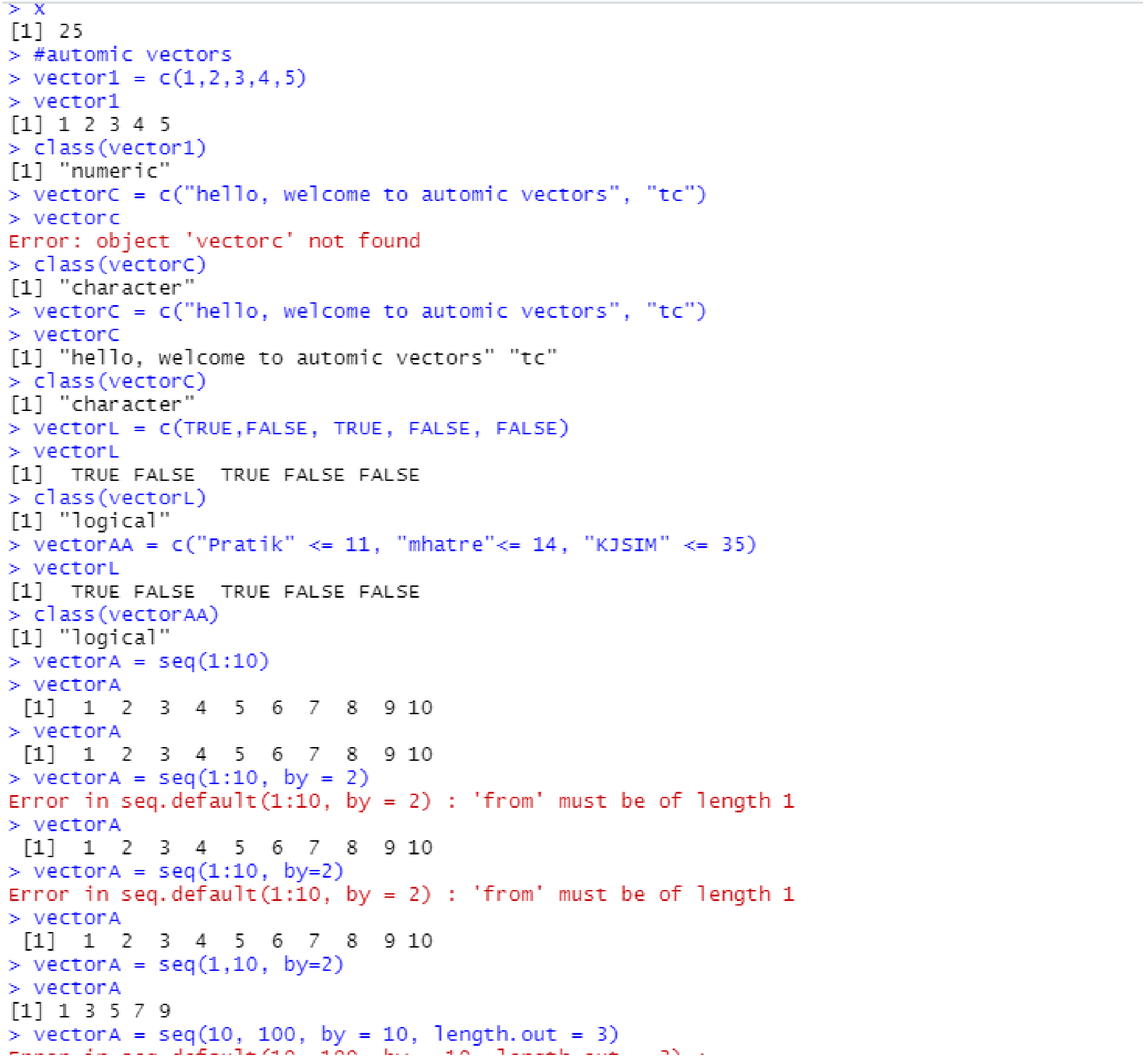
```
#functions in R
#built- in and user defined

# user defined functions
f <- function()
{
  print("normal user defined function")
}
f

#passing parameters to the function
#function returning value
f <- function(a,b)
{
  print(paste("user passed a: ",a, " b:", b))
  return (a + b);
}
x = f(5,6)
x

f <- function(a,b=10,c=10)
{
  print(paste("user passed a: ",a, " b:", b))
  return (a + b+c);
}
x = f(5)
x

#data structures in r
#1: vectors
#2: list
#3: arrays
#4: matrix
#5: data frames

#1.R is obssessed with vectors
#1.1 Automic vectors
#1.2 created with seq()

#automic vectors
#c - c() is called as combine function
```

```
vectorN = c(1,2,3,4,5)
vectorN
class(vectorN)

vectorC = c("hello, welcome to automic vectors", "tc")
vectorC
class(vectorC)

vectorL = c(TRUE,FALSE, TRUE, FALSE, FALSE)
vectorNames = c()
vectorL
class(vectorL)


vectorAA = c("Pratik" <= 11, "mhatre"<= 14, "KJSIM" <= 35)
vectorL
class(vectorAA)

vectorA = seq(1:10)
vectorA

vectorA = seq(1,10, by=2)
vectorA

vectorA = seq(10, 100,length.out = 3)
vectorA = seq(10, 100,length.out = 7)
vectorA

vectorB = seq(20,100,length.out = 7)
vectorB

vectorC = vectorA + vectorB
vectorC

a = c(2,3,4,5)
b = c(6,7,8,9)
d = a + b
d

e = c(a,b)
e

s = c("a", "b", "c")
e = c(a,b,s)
```

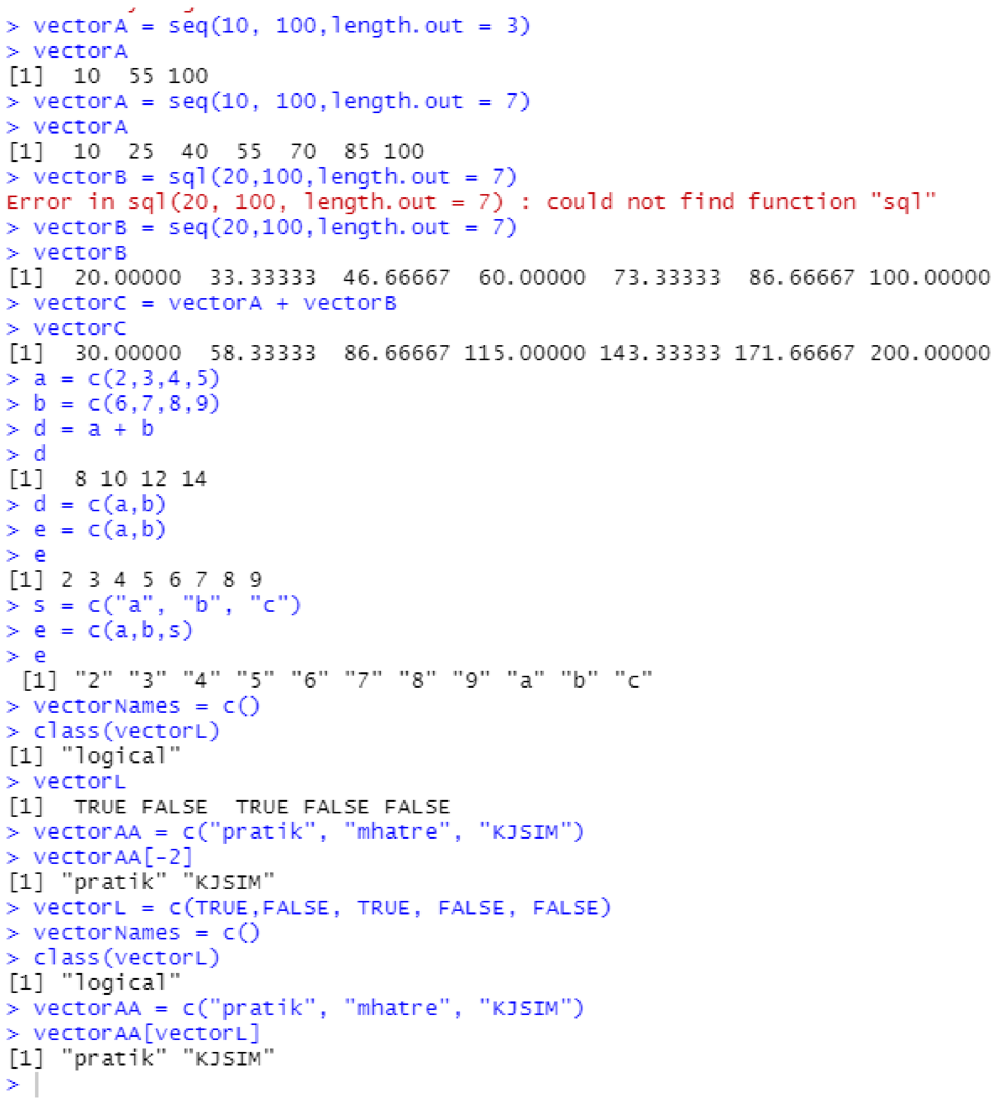
e

```
vectorL = c(TRUE,FALSE, TRUE, FALSE, FALSE)
vectorNames = c()
class(vectorL)

vectorAA = c("pratik", "mhatre", "KJSIM")
#this will print only true values
vectorAA[vectorL]
```

Output:



```
> x
[1] 25
> #automic vectors
> vector1 = c(1,2,3,4,5)
> vector1
[1] 1 2 3 4 5
> class(vector1)
[1] "numeric"
> vectorC = c("hello, welcome to automic vectors", "tc")
> vectorc
Error: object 'vectorc' not found
> class(vectorC)
[1] "character"
> vectorC = c("hello, welcome to automic vectors", "tc")
> vectorC
[1] "hello, welcome to automic vectors" "tc"
> class(vectorC)
[1] "character"
> vectorL = c(TRUE,FALSE, TRUE, FALSE, FALSE)
> vectorL
[1]  TRUE FALSE  TRUE FALSE FALSE
> class(vectorL)
[1] "logical"
> vectorAA = c("Pratik" <= 11, "mhatre"<= 14, "KJSIM" <= 35)
> vectorL
[1]  TRUE FALSE  TRUE FALSE FALSE
> class(vectorAA)
[1] "logical"
> vectorA = seq(1:10)
> vectorA
 [1]  1  2  3  4  5  6  7  8  9 10
> vectorA
 [1]  1  2  3  4  5  6  7  8  9 10
> vectorA = seq(1:10, by = 2)
Error in seq.default(1:10, by = 2) : 'from' must be of length 1
> vectorA
 [1]  1  2  3  4  5  6  7  8  9 10
> vectorA = seq(1:10, by=2)
Error in seq.default(1:10, by = 2) : 'from' must be of length 1
> vectorA
 [1]  1  2  3  4  5  6  7  8  9 10
> vectorA = seq(1,10, by=2)
> vectorA
[1] 1 3 5 7 9
> vectorA = seq(10, 100, by = 10, length.out = 3)
Error in seq.default(10, 100, by = 10, length.out = 3) :
```



```
> vectorA = seq(10, 100,length.out = 3)
> vectorA
[1]  10  55 100
> vectorA = seq(10, 100,length.out = 7)
> vectorA
[1]  10  25  40  55  70  85 100
> vectorB = sql(20,100,length.out = 7)
Error in sql(20, 100, length.out = 7) : could not find function "sql"
> vectorB = seq(20,100,length.out = 7)
> vectorB
[1]  20.00000  33.33333  46.66667  60.00000  73.33333  86.66667 100.00000
> vectorC = vectorA + vectorB
> vectorC
[1]  30.00000  58.33333  86.66667 115.00000 143.33333 171.66667 200.00000
> a = c(2,3,4,5)
> b = c(6,7,8,9)
> d = a + b
> d
[1]  8 10 12 14
> d = c(a,b)
> e = c(a,b)
> e
[1] 2 3 4 5 6 7 8 9
> s = c("a", "b", "c")
> e = c(a,b,s)
> e
 [1] "2" "3" "4" "5" "6" "7" "8" "9" "a" "b" "c"
> vectorNames = c()
> class(vectorL)
[1] "logical"
> vectorL
[1]  TRUE FALSE  TRUE FALSE FALSE
> vectorAA = c("pratik", "mhatre", "KJSIM")
> vectorAA[-2]
[1] "pratik" "KJSIM"
> vectorL = c(TRUE,FALSE, TRUE, FALSE, FALSE)
> vectorNames = c()
> class(vectorL)
[1] "logical"
> vectorAA = c("pratik", "mhatre", "KJSIM")
> vectorAA[vectorL]
[1] "pratik" "KJSIM"
> 
```

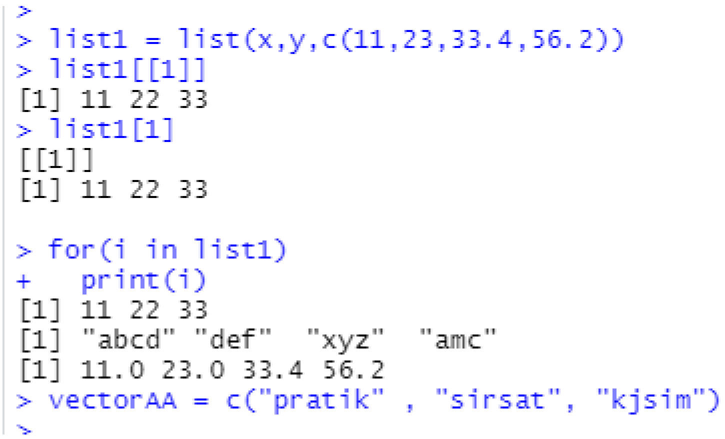
List
Program:

```
#functions in R
#built in and user defined
#user defined functions

vectorAA = c("pratik" , "sirsat", "kjsim")

vectorAA[vectorL]
list1 = list(c(1,2,3), "masters of computer application", c("abc","def"))

x = c(11,22,33)
y = c("abcd", "def", "xyz", "amc")

list1 = list(x,y,c(11,23,33.4,56.2))
list1[[1]] #returning values in vector
list1[1]   #returning first vector as list item


#iterating through list using for loop
for(i in list1)
  print(i)

#for loop on vector
for(i in y)
  print(i)

list1[[4]] = c(2,3,4) #appending value at the end of list using index
list1
list1[[4]] = NULL #deleting list element at the end using index
list1[[1]] = #returning values in vector
list[1] = #returning first vector as list item

x1 = list(101, "pratik", c(34,54,22))
x2 = list(102,"mhare", c("wt", "wt2", "R"))

list2 = list(x1,x2) #merging two lists

#passing names to the list index elements
names(list1) = c("rollno", "names", "marks")
list1

length(list1) #count the last index
```

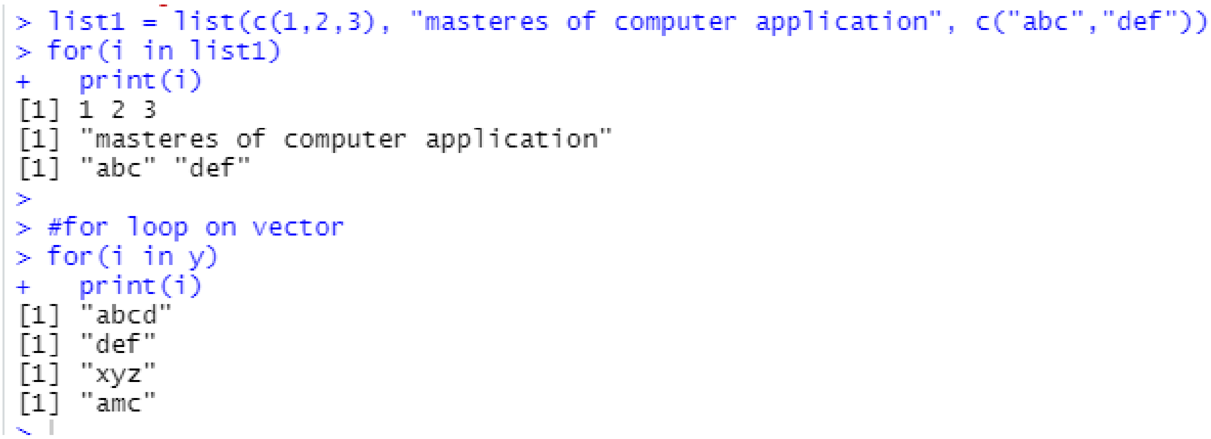
```
#arrays
#array(data, dim = c(rows, col, matrix))
array1 = array(c(11,22,33,44), dim = c(2,2))
array1 = array(c(11,22,33,44,55,66,77), dim = c(3,3,2),
dimnames = list(row_names, col_names, matrix_names))

array1 = array(c(11,22,33,44,55,66,77), dim = c(3,3,3))
array1

array1[,,1]

for (i in array1)
  print(i)

row_names = c("student1", "student2", "student3")
col_names = c("sub1", "sub2", "sub3")
matrix_names=c("div1", "div2")
array()

for (i in array1)
  print(i)


array1[,,1] #matrix1
array1[1,2,2] #

mysum <- function(element)
{
  element = element + 5
}

#apply(data,1-row,2-col,sum)
apply(array1[,,1],1,sum)

mysum <-function(element)
{
  element=element+5
}
apply(array1[,,1],1,mysum)


#apply(data, 1-row, 2-col, sum)
apply(array1[,,1],1,sum)
```

apply(array[,,1],1,mysum)

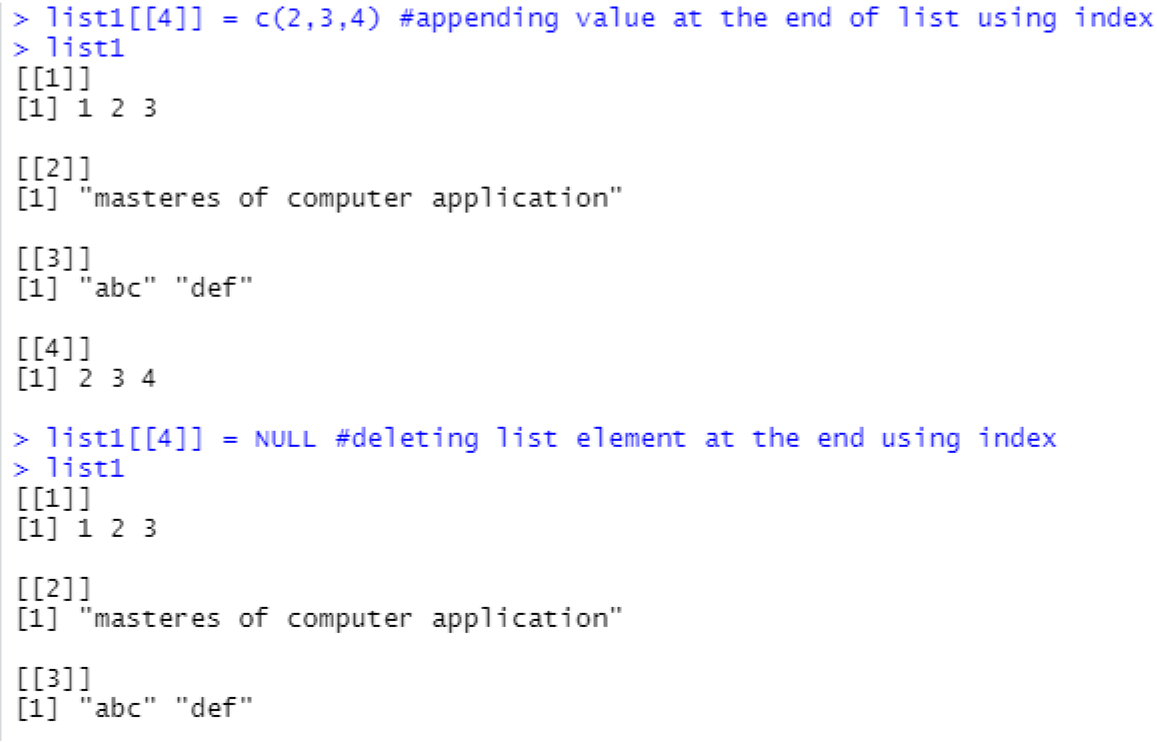
#factors
courses =c("mba","mca","mim","mim","mca","mba")
class(courses)
courses = factor(courses,levels=c("mba","mca","mim"))
courses
is.factor(courses)



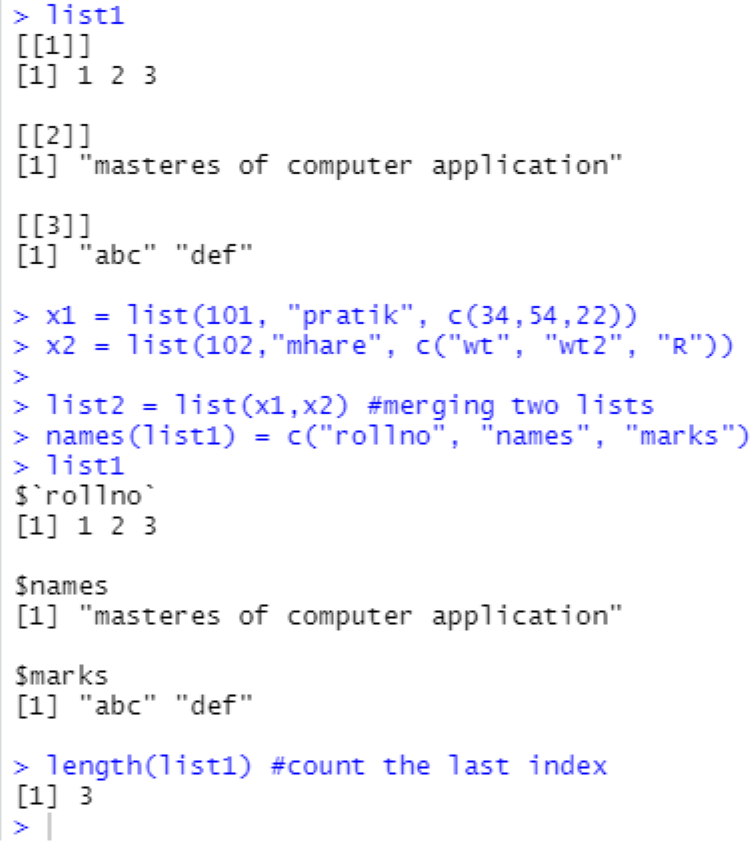
```
>
> list1 = list(x,y,c(11,23,33.4,56.2))
> list1[[1]]
[1] 11 22 33
> list1[1]
[[1]]
[1] 11 22 33

> for(i in list1)
+   print(i)
[1] 11 22 33
[1] "abcd" "def"  "xyz"  "amc"
[1] 11.0 23.0 33.4 56.2
> vectorAA = c("pratik" , "sirsat", "kjsim")
>
```



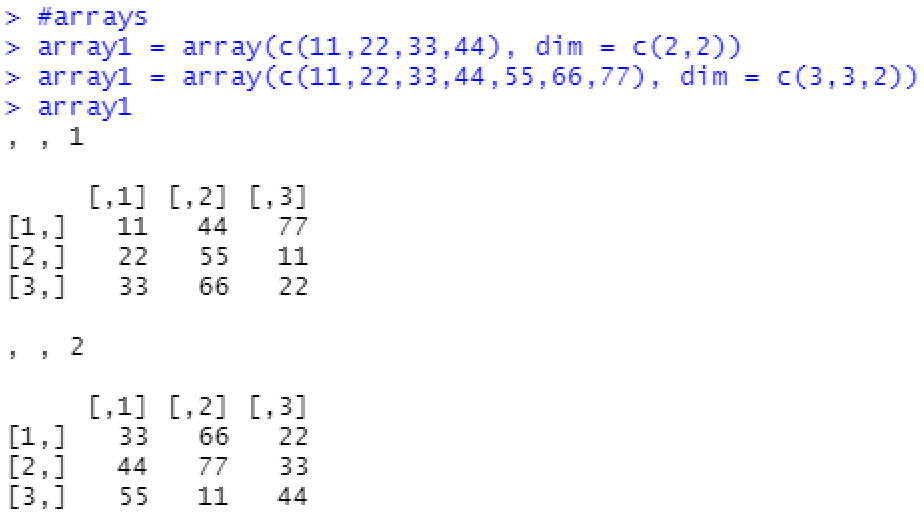
```
> list1 = list(c(1,2,3), "masteres of computer application", c("abc","def"))
> for(i in list1)
+   print(i)
[1] 1 2 3
[1] "masteres of computer application"
[1] "abc" "def"
>
> #for loop on vector
> for(i in y)
+   print(i)
[1] "abcd"
[1] "def"
[1] "xyz"
[1] "amc"
>
```



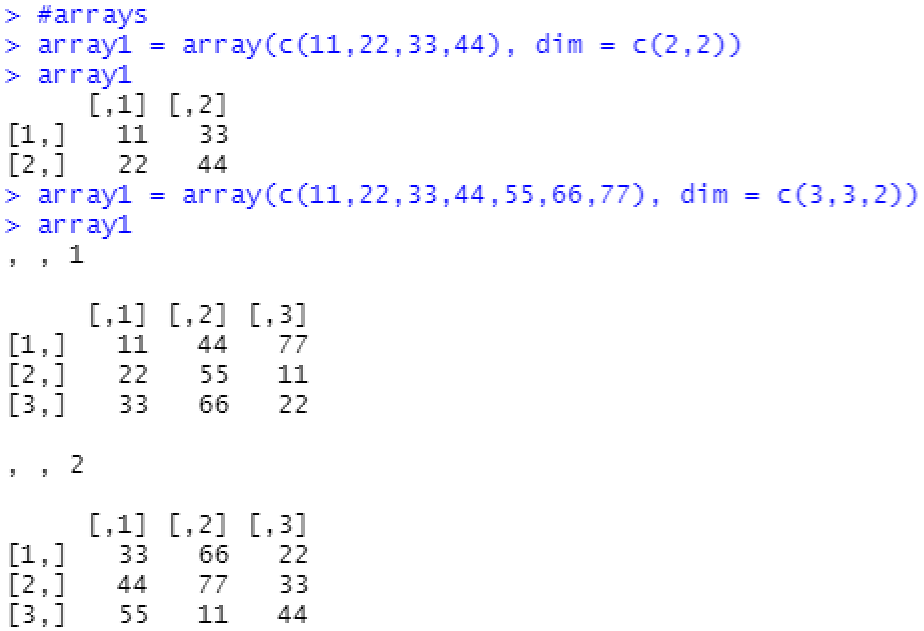
```
> list1[[4]] = c(2,3,4) #appending value at the end of list using index
> list1
[[1]]
[1] 1 2 3

[[2]]
[1] "masteres of computer application"

[[3]]
[1] "abc" "def"

[[4]]
[1] 2 3 4

> list1[[4]] = NULL #deleting list element at the end using index
> list1
[[1]]
[1] 1 2 3

[[2]]
[1] "masteres of computer application"

[[3]]
[1] "abc" "def"
```



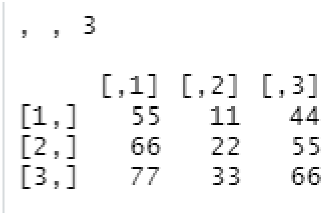
```
> list1
[[1]]
[1] 1 2 3

[[2]]
[1] "masteres of computer application"

[[3]]
[1] "abc" "def"

> x1 = list(101, "pratik", c(34,54,22))
> x2 = list(102,"mhare", c("wt", "wt2", "R"))
>
> list2 = list(x1,x2) #merging two lists
> names(list1) = c("rollno", "names", "marks")
> list1
$`rollno`
[1] 1 2 3

$names
[1] "masteres of computer application"

$marks
[1] "abc" "def"

> length(list1) #count the last index
[1] 3
> 
```



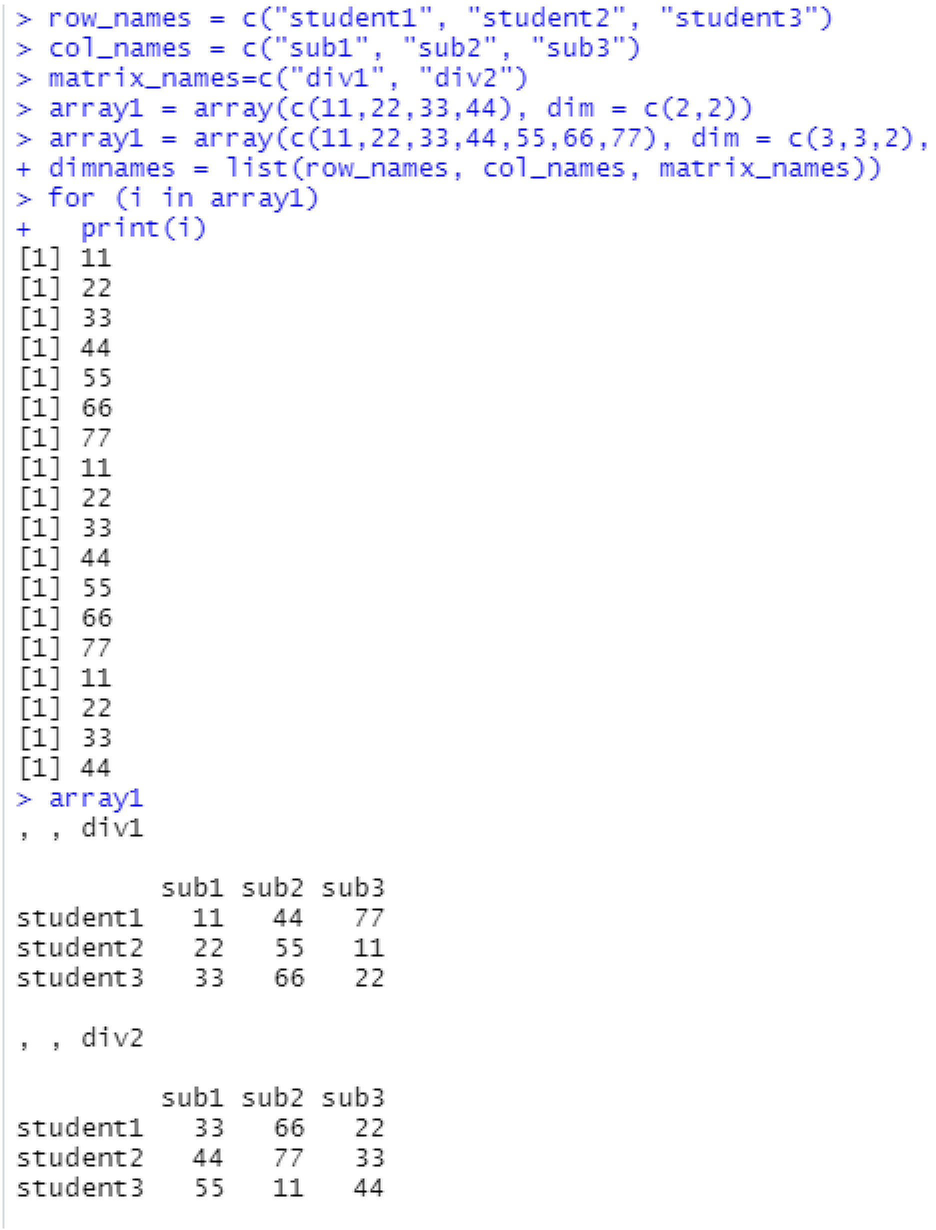
```
> #arrays
> array1 = array(c(11,22,33,44), dim = c(2,2))
> array1 = array(c(11,22,33,44,55,66,77), dim = c(3,3,2))
> array1
, , 1

     [,1] [,2] [,3]
[1,]   11   44   77
[2,]   22   55   11
[3,]   33   66   22

, , 2

     [,1] [,2] [,3]
[1,]   33   66   22
[2,]   44   77   33
[3,]   55   11   44
```



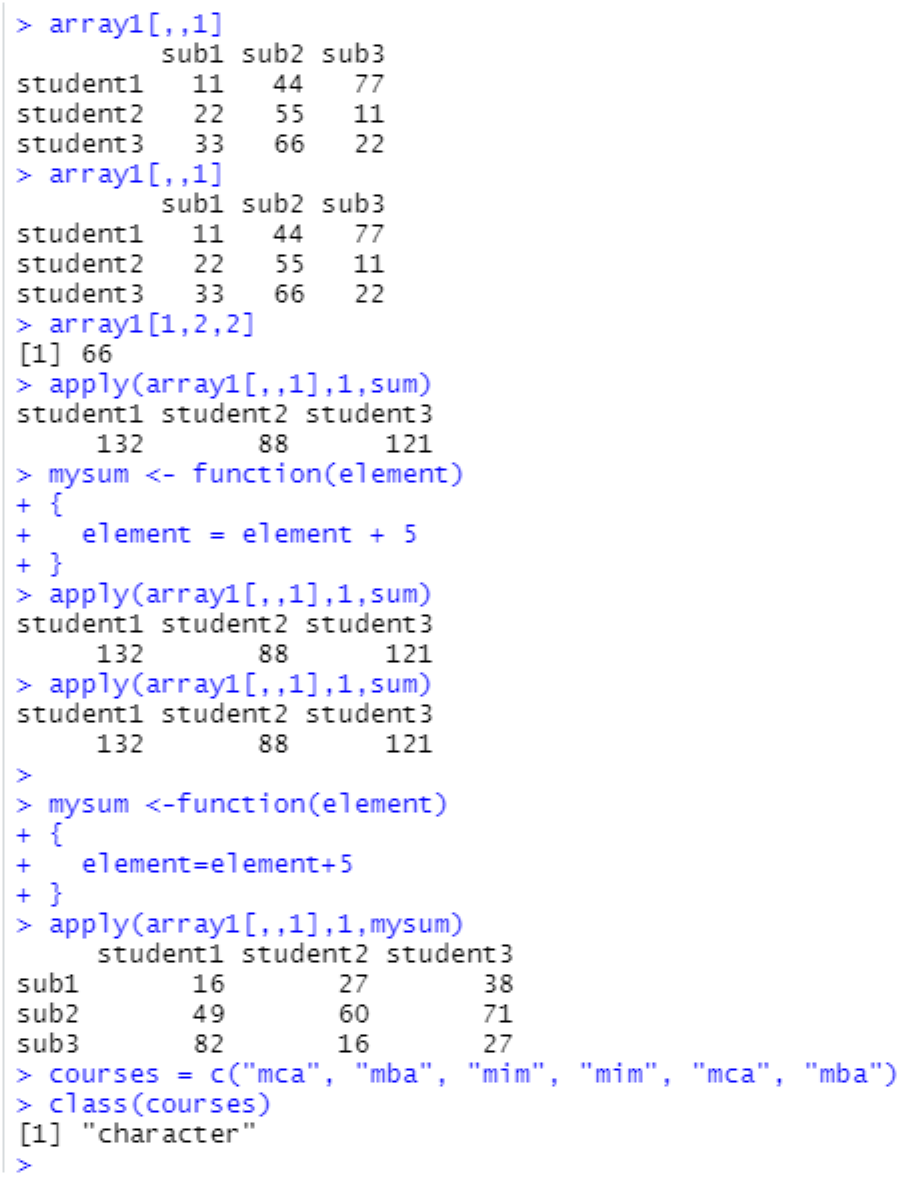
```
> #arrays
> array1 = array(c(11,22,33,44), dim = c(2,2))
> array1
     [,1] [,2]
[1,]   11   33
[2,]   22   44
> array1 = array(c(11,22,33,44,55,66,77), dim = c(3,3,2))
> array1
, , 1

     [,1] [,2] [,3]
[1,]   11   44   77
[2,]   22   55   11
[3,]   33   66   22

, , 2

     [,1] [,2] [,3]
[1,]   33   66   22
[2,]   44   77   33
[3,]   55   11   44
```



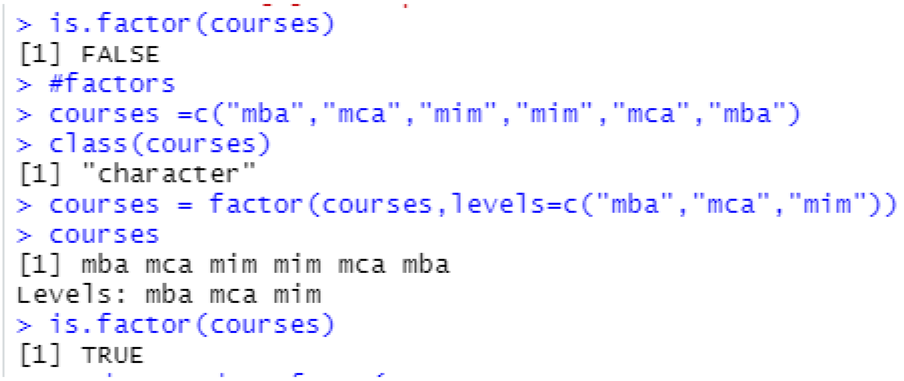
```
, , 3

     [,1] [,2] [,3]
[1,]   55   11   44
[2,]   66   22   55
[3,]   77   33   66
```



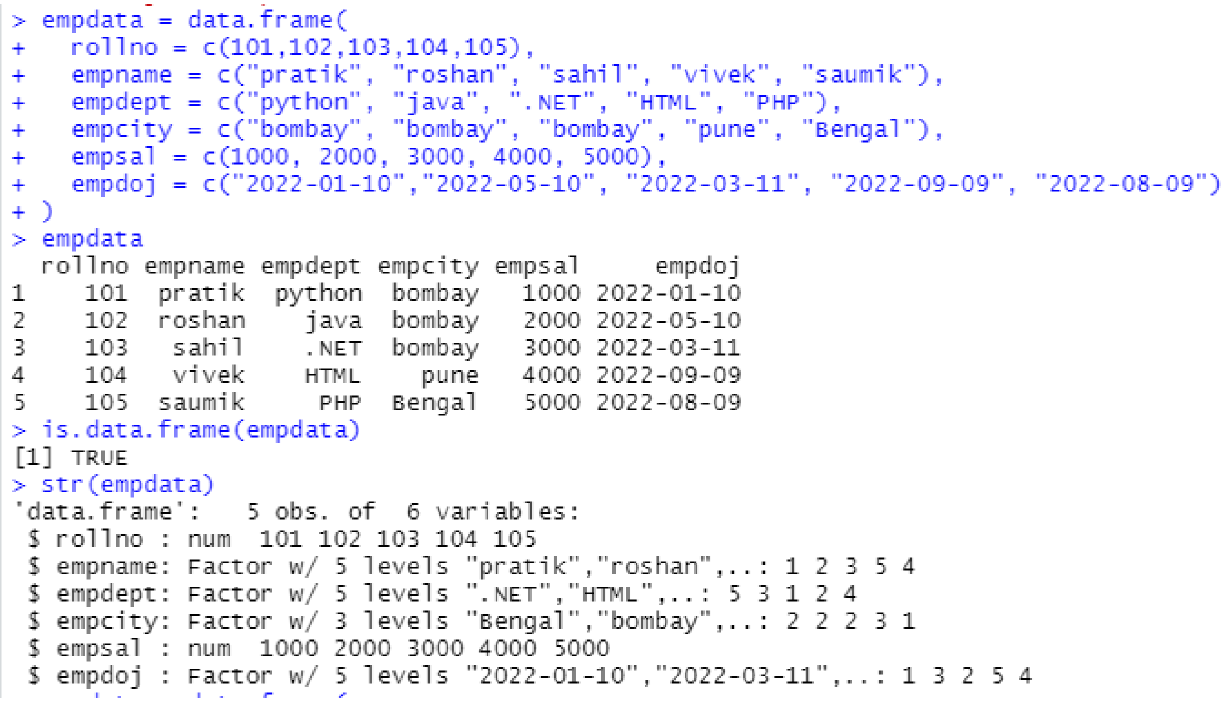
```
> row_names = c("student1", "student2", "student3")
> col_names = c("sub1", "sub2", "sub3")
> matrix_names=c("div1", "div2")
> array1 = array(c(11,22,33,44), dim = c(2,2))
> array1 = array(c(11,22,33,44,55,66,77), dim = c(3,3,2),
+ dimnames = list(row_names, col_names, matrix_names))
> for (i in array1)
+   print(i)
[1] 11
[1] 22
[1] 33
[1] 44
[1] 55
[1] 66
[1] 77
[1] 11
[1] 22
[1] 33
[1] 44
[1] 55
[1] 66
[1] 77
[1] 11
[1] 22
[1] 33
[1] 44
> array1
, , div1

         sub1 sub2 sub3
student1   11   44   77
student2   22   55   11
student3   33   66   22

, , div2

         sub1 sub2 sub3
student1   33   66   22
student2   44   77   33
student3   55   11   44
```



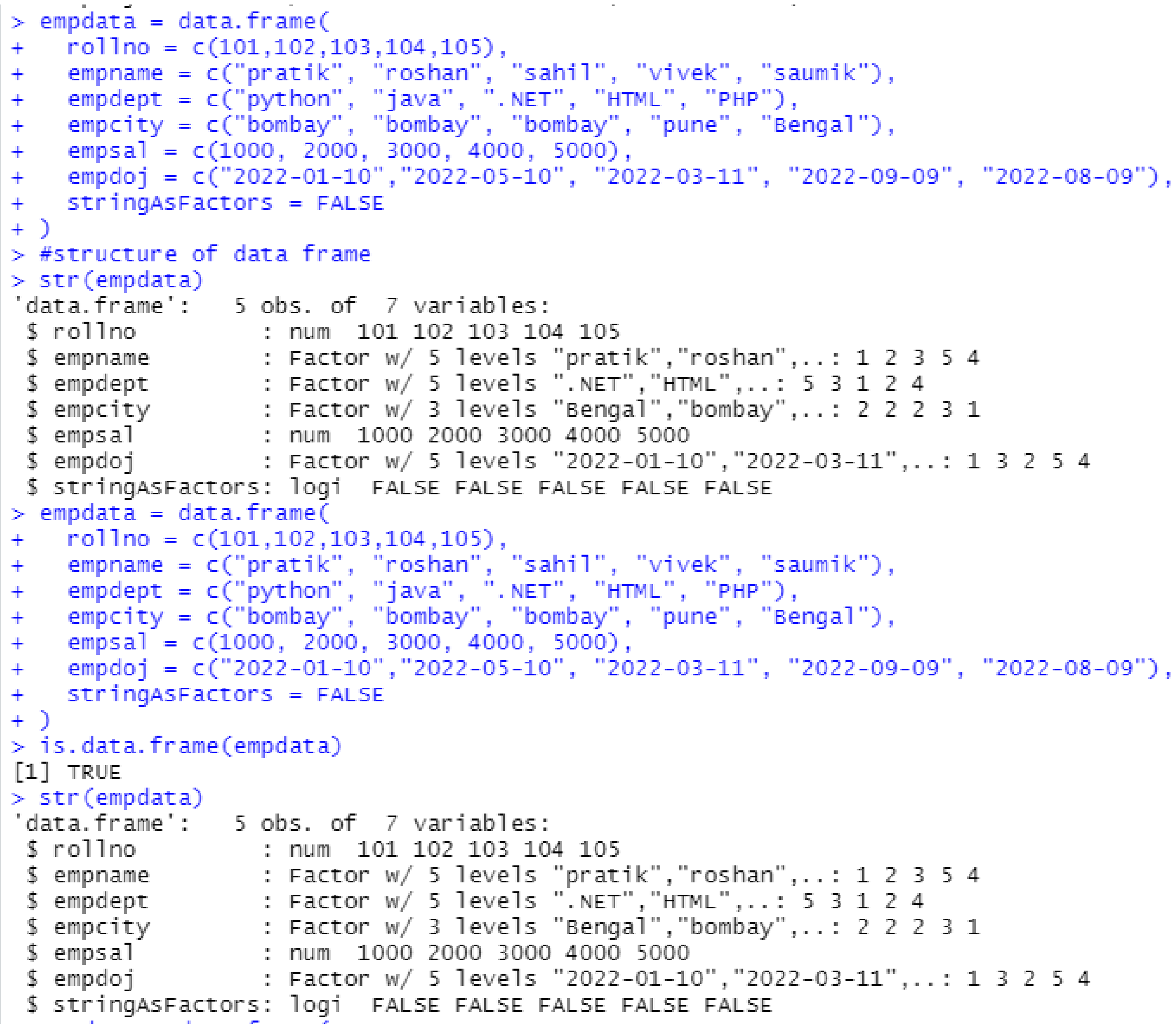
```
> array1[,,1]
         sub1 sub2 sub3
student1   11   44   77
student2   22   55   11
student3   33   66   22
> array1[,,1]
         sub1 sub2 sub3
student1   11   44   77
student2   22   55   11
student3   33   66   22
> array1[1,2,2]
[1] 66
> apply(array1[,,1],1,sum)
student1 student2 student3
     132       88      121
> mysum <- function(element)
+ {
+   element = element + 5
+ }
> apply(array1[,,1],1,sum)
student1 student2 student3
     132       88      121
> apply(array1[,,1],1,sum)
student1 student2 student3
     132       88      121
>
> mysum <-function(element)
+ {
+   element=element+5
+ }
> apply(array1[,,1],1,mysum)
     student1 student2 student3
sub1       16       27       38
sub2       49       60       71
sub3       82       16       27
> courses = c("mca", "mba", "mim", "mim", "mca", "mba")
> class(courses)
[1] "character"
>
```



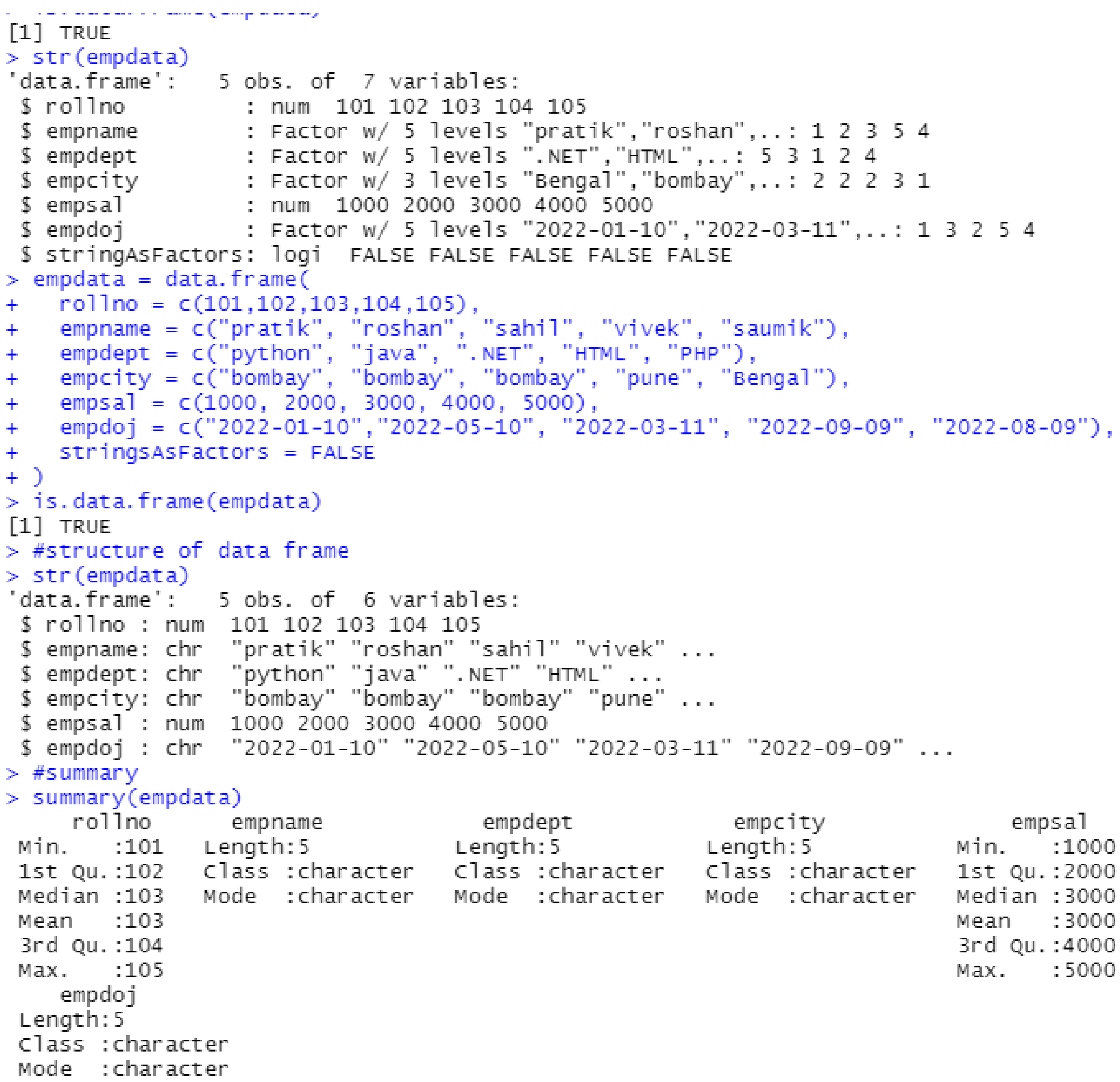
```
> is.factor(courses)
[1] FALSE
> #factors
> courses =c("mba","mca","mim","mim","mca","mba")
> class(courses)
[1] "character"
> courses = factor(courses,levels=c("mba","mca","mim"))
> courses
[1] mba mca mim mim mca mba
Levels: mba mca mim
> is.factor(courses)
[1] TRUE
```



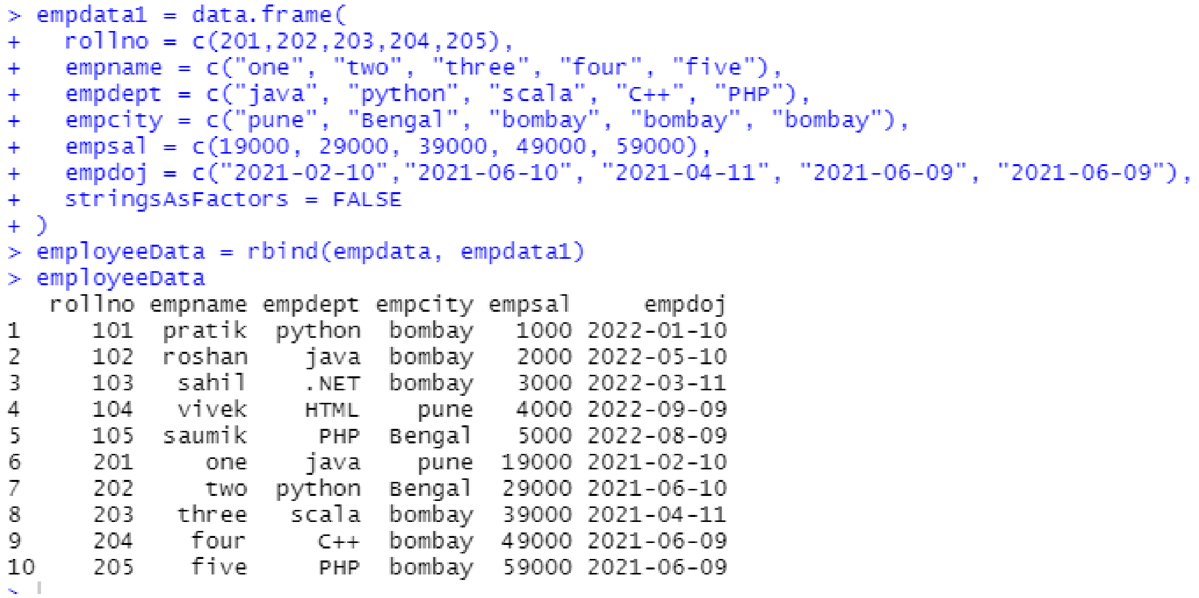
```
> empdata = data.frame(
+   rollno = c(101,102,103,104,105),
+   empname = c("pratik", "roshan", "sahil", "vivek", "saumik"),
+   empdept = c("python", "java", ".NET", "HTML", "PHP"),
+   empcity = c("bombay", "bombay", "bombay", "pune", "Bengal"),
+   empsal = c(1000, 2000, 3000, 4000, 5000),
+   empdoj = c("2022-01-10","2022-05-10", "2022-03-11", "2022-09-09", "2022-08-09")
+ )
> empdata
  rollno empname empdept empcity empsal     empdoj
1    101  pratik  python  bombay   1000 2022-01-10
2    102  roshan    java  bombay   2000 2022-05-10
3    103   sahil    .NET  bombay   3000 2022-03-11
4    104   vivek    HTML    pune   4000 2022-09-09
5    105  saumik     PHP  Bengal   5000 2022-08-09
> is.data.frame(empdata)
[1] TRUE
> str(empdata)
'data.frame':   5 obs. of  6 variables:
 $ rollno : num  101 102 103 104 105
 $ empname: Factor w/ 5 levels "pratik","roshan",..: 1 2 3 5 4
 $ empdept: Factor w/ 5 levels ".NET","HTML",..: 5 3 1 2 4
 $ empcity: Factor w/ 3 levels "Bengal","bombay",..: 2 2 2 3 1
 $ empsal : num  1000 2000 3000 4000 5000
 $ empdoj : Factor w/ 5 levels "2022-01-10","2022-03-11",..: 1 3 2 5 4
```



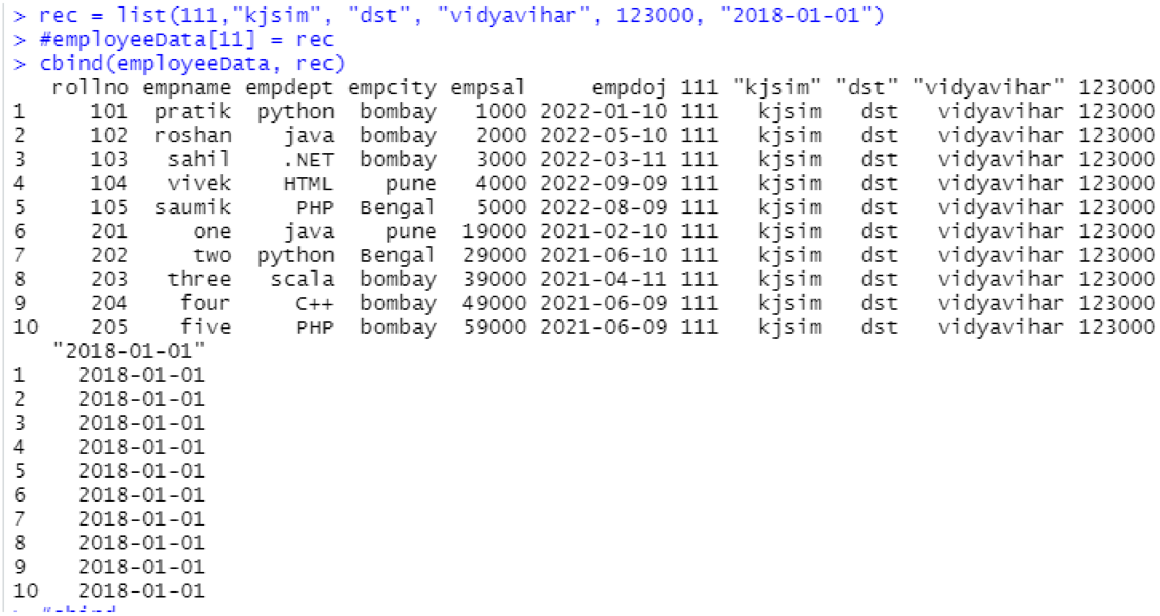
```
> empdata = data.frame(
+   rollno = c(101,102,103,104,105),
+   empname = c("pratik", "roshan", "sahil", "vivek", "saumik"),
+   empdept = c("python", "java", ".NET", "HTML", "PHP"),
+   empcity = c("bombay", "bombay", "bombay", "pune", "Bengal"),
+   empsal = c(1000, 2000, 3000, 4000, 5000),
+   empdoj = c("2022-01-10","2022-05-10", "2022-03-11", "2022-09-09", "2022-08-09"),
+   stringAsFactors = FALSE
+ )
> #structure of data frame
> str(empdata)
'data.frame':   5 obs. of  7 variables:
 $ rollno         : num  101 102 103 104 105
 $ empname        : Factor w/ 5 levels "pratik","roshan",..: 1 2 3 5 4
 $ empdept        : Factor w/ 5 levels ".NET","HTML",..: 5 3 1 2 4
 $ empcity        : Factor w/ 3 levels "Bengal","bombay",..: 2 2 2 3 1
 $ empsal         : num  1000 2000 3000 4000 5000
 $ empdoj         : Factor w/ 5 levels "2022-01-10","2022-03-11",..: 1 3 2 5 4
 $ stringAsFactors: logi  FALSE FALSE FALSE FALSE FALSE
> empdata = data.frame(
+   rollno = c(101,102,103,104,105),
+   empname = c("pratik", "roshan", "sahil", "vivek", "saumik"),
+   empdept = c("python", "java", ".NET", "HTML", "PHP"),
+   empcity = c("bombay", "bombay", "bombay", "pune", "Bengal"),
+   empsal = c(1000, 2000, 3000, 4000, 5000),
+   empdoj = c("2022-01-10","2022-05-10", "2022-03-11", "2022-09-09", "2022-08-09"),
+   stringAsFactors = FALSE
+ )
> is.data.frame(empdata)
[1] TRUE
> str(empdata)
'data.frame':   5 obs. of  7 variables:
 $ rollno         : num  101 102 103 104 105
 $ empname        : Factor w/ 5 levels "pratik","roshan",..: 1 2 3 5 4
 $ empdept        : Factor w/ 5 levels ".NET","HTML",..: 5 3 1 2 4
 $ empcity        : Factor w/ 3 levels "Bengal","bombay",..: 2 2 2 3 1
 $ empsal         : num  1000 2000 3000 4000 5000
 $ empdoj         : Factor w/ 5 levels "2022-01-10","2022-03-11",..: 1 3 2 5 4
 $ stringAsFactors: logi  FALSE FALSE FALSE FALSE FALSE
```



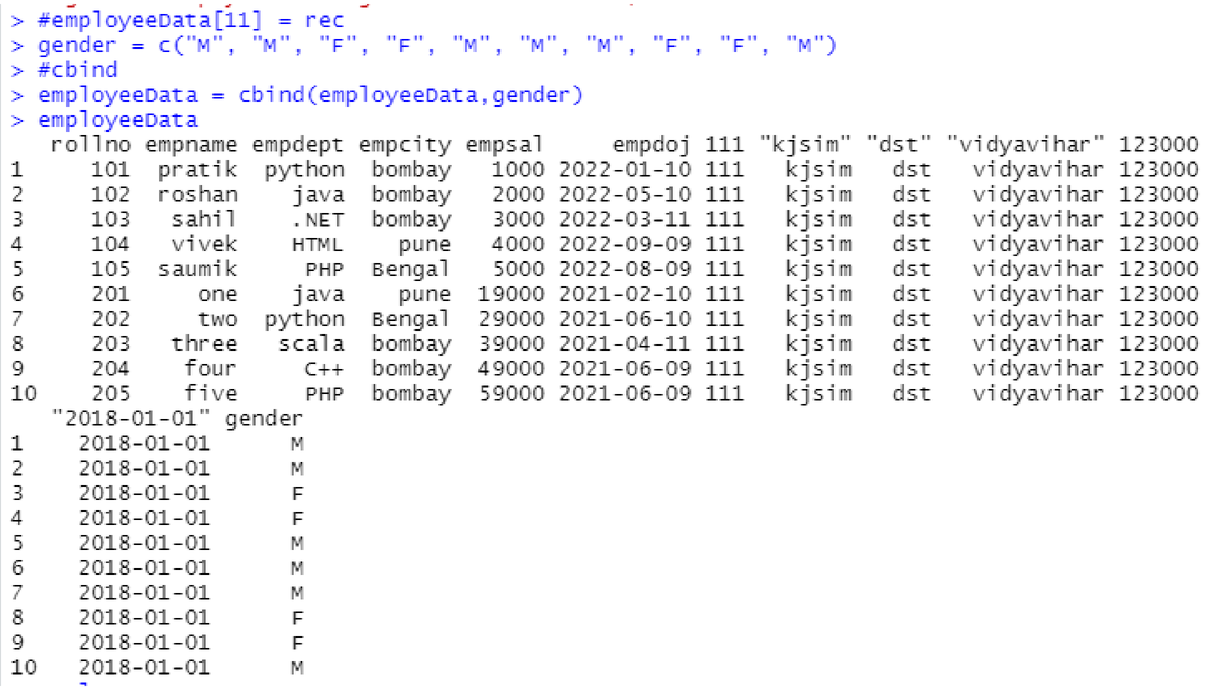
```
> is.data.frame(empdata)
[1] TRUE
> str(empdata)
'data.frame':	5 obs. of  7 variables:
 $ rollno         : num  101 102 103 104 105
 $ empname        : Factor w/ 5 levels "pratik","roshan",..: 1 2 3 5 4
 $ empdept        : Factor w/ 5 levels ".NET","HTML",..: 5 3 1 2 4
 $ empcity        : Factor w/ 3 levels "Bengal","bombay",..: 2 2 2 3 1
 $ empsal         : num  1000 2000 3000 4000 5000
 $ empdoj         : Factor w/ 5 levels "2022-01-10","2022-03-11",..: 1 3 2 5 4
 $ stringAsFactors: logi  FALSE FALSE FALSE FALSE FALSE
> empdata = data.frame(
+   rollno = c(101,102,103,104,105),
+   empname = c("pratik", "roshan", "sahil", "vivek", "saumik"),
+   empdept = c("python", "java", ".NET", "HTML", "PHP"),
+   empcity = c("bombay", "bombay", "bombay", "pune", "Bengal"),
+   empsal = c(1000, 2000, 3000, 4000, 5000),
+   empdoj = c("2022-01-10","2022-05-10", "2022-03-11", "2022-09-09", "2022-08-09"),
+   stringsAsFactors = FALSE
+ )
> is.data.frame(empdata)
[1] TRUE
> #structure of data frame
> str(empdata)
'data.frame':	5 obs. of  6 variables:
 $ rollno : num  101 102 103 104 105
 $ empname: chr  "pratik" "roshan" "sahil" "vivek" ...
 $ empdept: chr  "python" "java" ".NET" "HTML" ...
 $ empcity: chr  "bombay" "bombay" "bombay" "pune" ...
 $ empsal : num  1000 2000 3000 4000 5000
 $ empdoj : chr  "2022-01-10" "2022-05-10" "2022-03-11" "2022-09-09" ...
> #summary
> summary(empdata)
     rollno      empname            empdept            empcity              empsal    
 Min.   :101   Length:5           Length:5           Length:5           Min.   :1000  
 1st Qu.:102   Class :character   Class :character   Class :character   1st Qu.:2000  
 Median :103   Mode  :character   Mode  :character   Mode  :character   Median :3000  
 Mean   :103                                                            Mean   :3000  
 3rd Qu.:104                                                            3rd Qu.:4000  
 Max.   :105                                                            Max.   :5000  
    empdoj         
 Length:5          
 Class :character  
 Mode  :character  
```



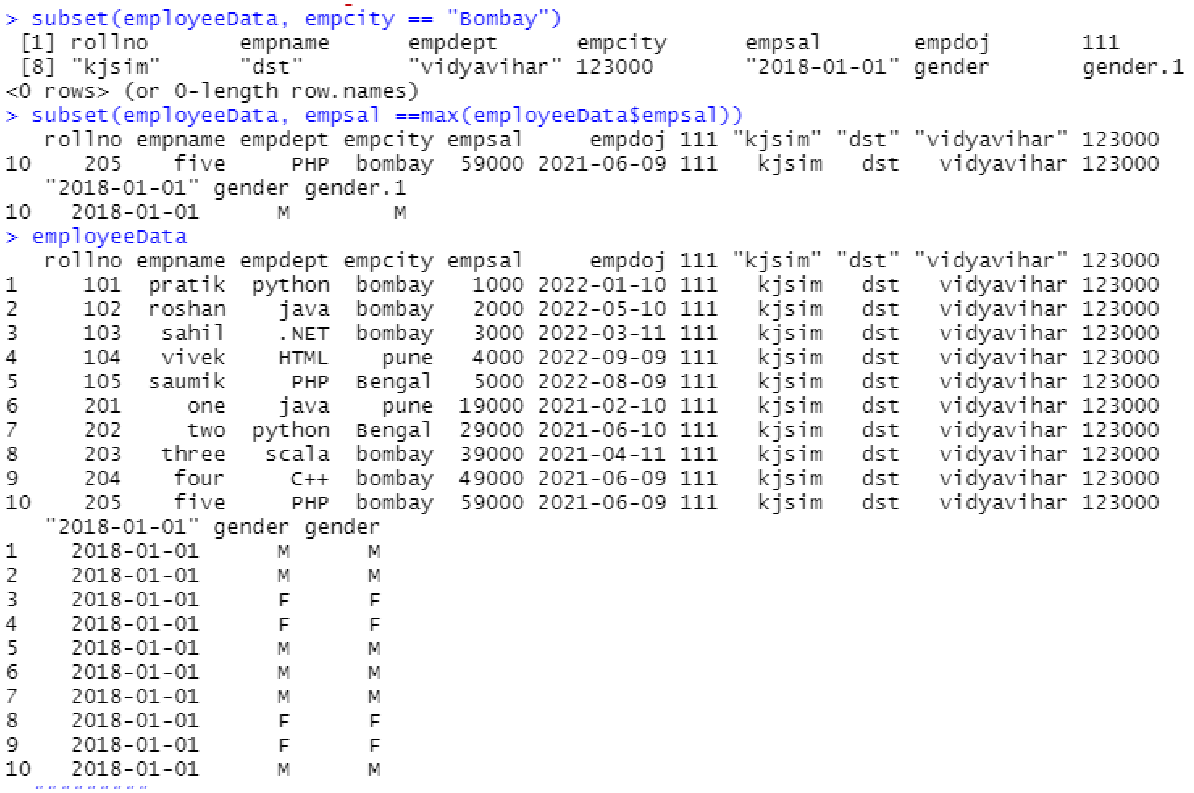
```
> empdata1 = data.frame(
+   rollno = c(201,202,203,204,205),
+   empname = c("one", "two", "three", "four", "five"),
+   empdept = c("java", "python", "scala", "C++", "PHP"),
+   empcity = c("pune", "Bengal", "bombay", "bombay", "bombay"),
+   empsal = c(19000, 29000, 39000, 49000, 59000),
+   empdoj = c("2021-02-10","2021-06-10", "2021-04-11", "2021-06-09", "2021-06-09"),
+   stringsAsFactors = FALSE
+ )
> employeeData = rbind(empdata, empdata1)
> employeeData
   rollno empname empdept empcity empsal     empdoj
1     101  pratik  python  bombay   1000 2022-01-10
2     102  roshan    java  bombay   2000 2022-05-10
3     103   sahil    .NET  bombay   3000 2022-03-11
4     104   vivek    HTML    pune   4000 2022-09-09
5     105  saumik     PHP  Bengal   5000 2022-08-09
6     201     one    java    pune  19000 2021-02-10
7     202     two  python  Bengal  29000 2021-06-10
8     203   three   scala  bombay  39000 2021-04-11
9     204    four     C++  bombay  49000 2021-06-09
10    205    five     PHP  bombay  59000 2021-06-09
```



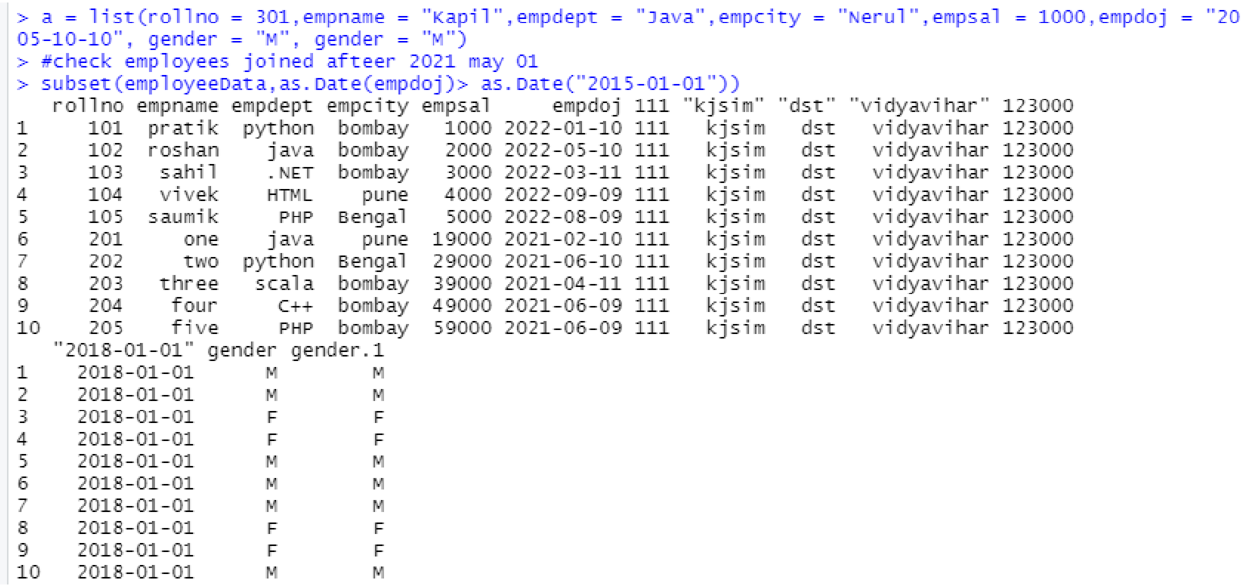
```
> rec = list(111,"kjsim", "dst", "vidyavihar", 123000, "2018-01-01")
> #employeeData[11] = rec
> cbind(employeeData, rec)
   rollno empname empdept empcity empsal     empdoj 111 "kjsim" "dst" "vidyavihar" 123000
1     101  pratik  python  bombay   1000 2022-01-10 111   kjsim   dst   vidyavihar 123000
2     102  roshan    java  bombay   2000 2022-05-10 111   kjsim   dst   vidyavihar 123000
3     103   sahil    .NET  bombay   3000 2022-03-11 111   kjsim   dst   vidyavihar 123000
4     104   vivek    HTML    pune   4000 2022-09-09 111   kjsim   dst   vidyavihar 123000
5     105  saumik     PHP  Bengal   5000 2022-08-09 111   kjsim   dst   vidyavihar 123000
6     201     one    java    pune  19000 2021-02-10 111   kjsim   dst   vidyavihar 123000
7     202     two  python  Bengal  29000 2021-06-10 111   kjsim   dst   vidyavihar 123000
8     203   three   scala  bombay  39000 2021-04-11 111   kjsim   dst   vidyavihar 123000
9     204    four     C++  bombay  49000 2021-06-09 111   kjsim   dst   vidyavihar 123000
10    205    five     PHP  bombay  59000 2021-06-09 111   kjsim   dst   vidyavihar 123000
   "2018-01-01"
1    2018-01-01
2    2018-01-01
3    2018-01-01
4    2018-01-01
5    2018-01-01
6    2018-01-01
7    2018-01-01
8    2018-01-01
9    2018-01-01
10   2018-01-01
```



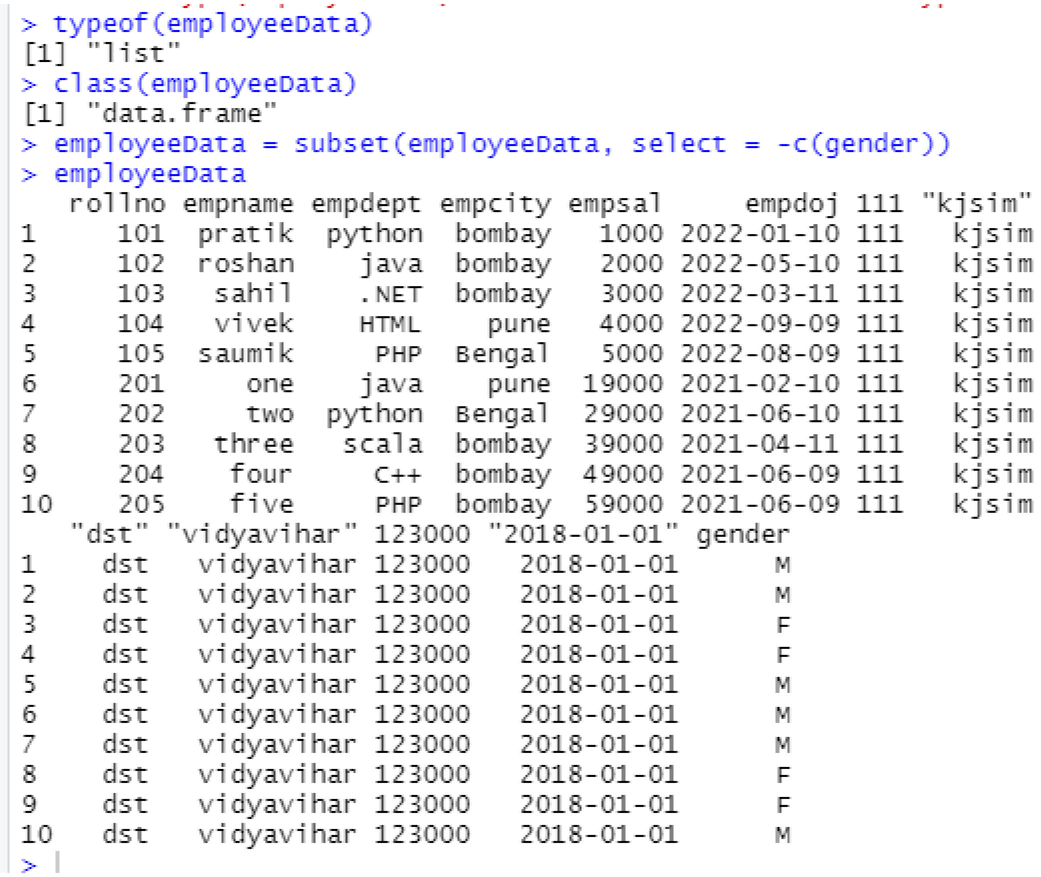
```
> #employeeData[11] = rec
> gender = c("M", "M", "F", "F", "M", "M", "M", "F", "F", "M")
> #cbind
> employeeData = cbind(employeeData,gender)
> employeeData
   rollno empname empdept empcity empsal     empdoj 111 "kjsim" "dst" "vidyavihar" 123000
1     101  pratik  python  bombay   1000 2022-01-10 111   kjsim   dst   vidyavihar 123000
2     102  roshan    java  bombay   2000 2022-05-10 111   kjsim   dst   vidyavihar 123000
3     103   sahil    .NET  bombay   3000 2022-03-11 111   kjsim   dst   vidyavihar 123000
4     104   vivek    HTML    pune   4000 2022-09-09 111   kjsim   dst   vidyavihar 123000
5     105  saumik     PHP  Bengal   5000 2022-08-09 111   kjsim   dst   vidyavihar 123000
6     201     one    java    pune  19000 2021-02-10 111   kjsim   dst   vidyavihar 123000
7     202     two  python  Bengal  29000 2021-06-10 111   kjsim   dst   vidyavihar 123000
8     203   three   scala  bombay  39000 2021-04-11 111   kjsim   dst   vidyavihar 123000
9     204    four     C++  bombay  49000 2021-06-09 111   kjsim   dst   vidyavihar 123000
10    205    five     PHP  bombay  59000 2021-06-09 111   kjsim   dst   vidyavihar 123000
   "2018-01-01" gender
1    2018-01-01      M
2    2018-01-01      M
3    2018-01-01      F
4    2018-01-01      F
5    2018-01-01      M
6    2018-01-01      M
7    2018-01-01      M
8    2018-01-01      F
9    2018-01-01      F
10   2018-01-01      M
```



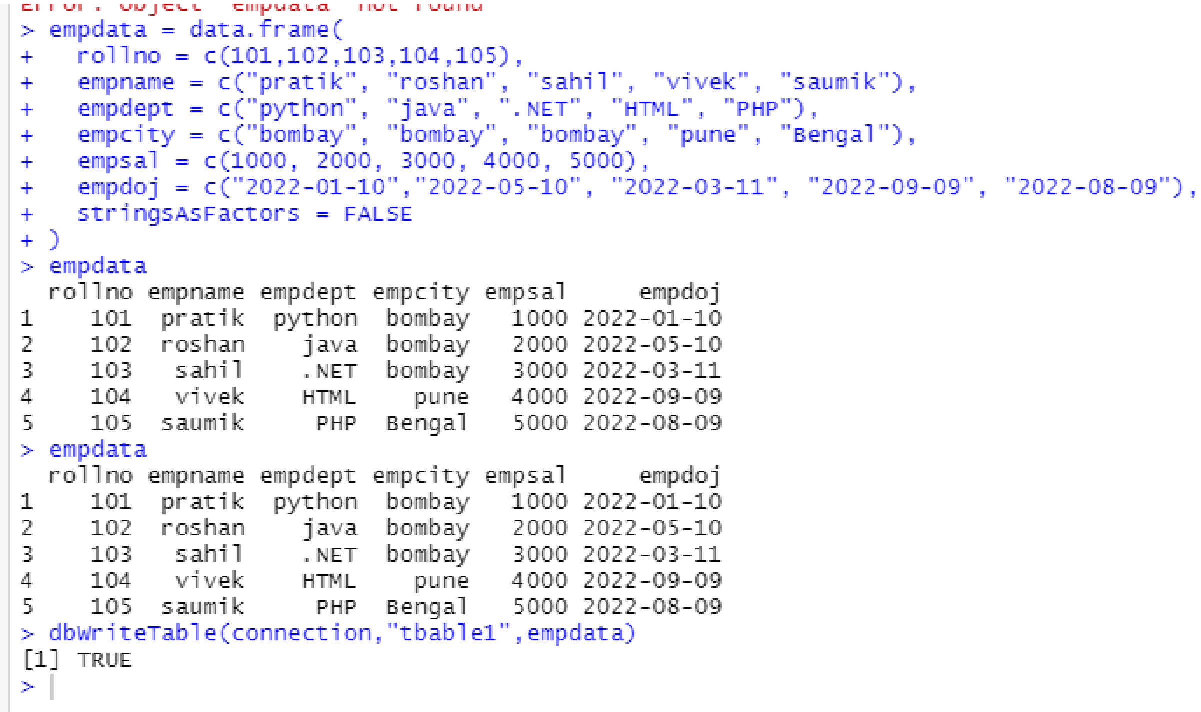
```
> subset(employeeData, empcity == "Bombay")
 [1] rollno       empname      empdept      empcity      empsal       empdoj       111
 [8] "kjsim"      "dst"        "vidyavihar" 123000       "2018-01-01" gender       gender.1
<0 rows> (or 0-length row.names)
> subset(employeeData, empsal ==max(employeeData$empsal))
   rollno empname empdept empcity empsal     empdoj 111 "kjsim" "dst" "vidyavihar" 123000
10    205    five     PHP  bombay  59000 2021-06-09 111   kjsim   dst   vidyavihar 123000
   "2018-01-01" gender gender.1
10   2018-01-01      M        M
> employeeData
   rollno empname empdept empcity empsal     empdoj 111 "kjsim" "dst" "vidyavihar" 123000
1     101  pratik  python  bombay   1000 2022-01-10 111   kjsim   dst   vidyavihar 123000
2     102  roshan    java  bombay   2000 2022-05-10 111   kjsim   dst   vidyavihar 123000
3     103   sahil    .NET  bombay   3000 2022-03-11 111   kjsim   dst   vidyavihar 123000
4     104   vivek    HTML    pune   4000 2022-09-09 111   kjsim   dst   vidyavihar 123000
5     105  saumik     PHP  Bengal   5000 2022-08-09 111   kjsim   dst   vidyavihar 123000
6     201     one    java    pune  19000 2021-02-10 111   kjsim   dst   vidyavihar 123000
7     202     two  python  Bengal  29000 2021-06-10 111   kjsim   dst   vidyavihar 123000
8     203   three   scala  bombay  39000 2021-04-11 111   kjsim   dst   vidyavihar 123000
9     204    four     C++  bombay  49000 2021-06-09 111   kjsim   dst   vidyavihar 123000
10    205    five     PHP  bombay  59000 2021-06-09 111   kjsim   dst   vidyavihar 123000
   "2018-01-01" gender gender
1    2018-01-01      M      M
2    2018-01-01      M      M
3    2018-01-01      F      F
4    2018-01-01      F      F
5    2018-01-01      M      M
6    2018-01-01      M      M
7    2018-01-01      M      M
8    2018-01-01      F      F
9    2018-01-01      F      F
10   2018-01-01      M      M
```



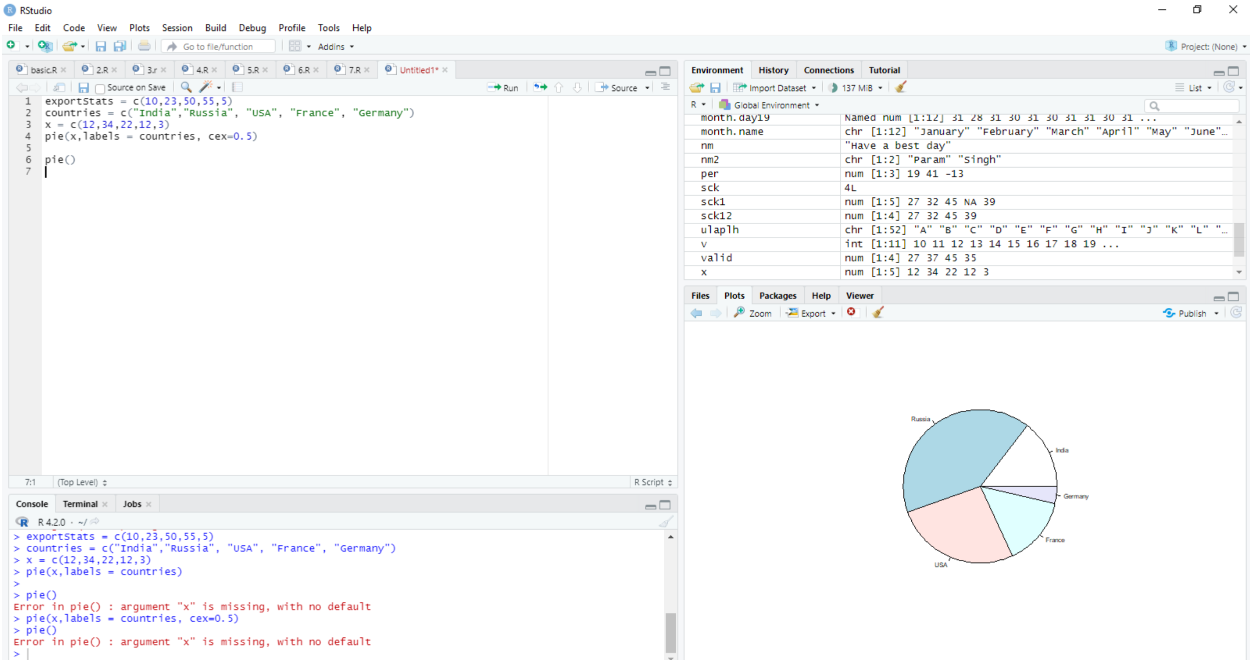
```
> a = list(rollno = 301,empname = "Kapil",empdept = "Java",empcity = "Nerul",empsal = 1000,empdoj = "20
05-10-10", gender = "M", gender = "M")
> #check employees joined afteer 2021 may 01
> subset(employeeData,as.Date(empdoj)> as.Date("2015-01-01"))
   rollno empname empdept empcity empsal     empdoj 111 "kjsim" "dst" "vidyavihar" 123000
1     101  pratik  python  bombay   1000 2022-01-10 111   kjsim   dst   vidyavihar 123000
2     102  roshan    java  bombay   2000 2022-05-10 111   kjsim   dst   vidyavihar 123000
3     103   sahil    .NET  bombay   3000 2022-03-11 111   kjsim   dst   vidyavihar 123000
4     104   vivek    HTML    pune   4000 2022-09-09 111   kjsim   dst   vidyavihar 123000
5     105  saumik     PHP  Bengal   5000 2022-08-09 111   kjsim   dst   vidyavihar 123000
6     201     one    java    pune  19000 2021-02-10 111   kjsim   dst   vidyavihar 123000
7     202     two  python  Bengal  29000 2021-06-10 111   kjsim   dst   vidyavihar 123000
8     203   three   scala  bombay  39000 2021-04-11 111   kjsim   dst   vidyavihar 123000
9     204    four     C++  bombay  49000 2021-06-09 111   kjsim   dst   vidyavihar 123000
10    205    five     PHP  bombay  59000 2021-06-09 111   kjsim   dst   vidyavihar 123000
   "2018-01-01" gender gender.1
1    2018-01-01      M        M
2    2018-01-01      M        M
3    2018-01-01      F        F
4    2018-01-01      F        F
5    2018-01-01      M        M
6    2018-01-01      M        M
7    2018-01-01      M        M
8    2018-01-01      F        F
9    2018-01-01      F        F
10   2018-01-01      M        M
```



```
> typeof(employeeData)
[1] "list"
> class(employeeData)
[1] "data.frame"
> employeeData = subset(employeeData, select = -c(gender))
> employeeData
   rollno empname empdept empcity empsal     empdoj 111 "kjsim"
1     101  pratik  python  bombay   1000 2022-01-10 111   kjsim
2     102  roshan    java  bombay   2000 2022-05-10 111   kjsim
3     103   sahil    .NET  bombay   3000 2022-03-11 111   kjsim
4     104   vivek    HTML    pune   4000 2022-09-09 111   kjsim
5     105  saumik     PHP  Bengal   5000 2022-08-09 111   kjsim
6     201     one    java    pune  19000 2021-02-10 111   kjsim
7     202     two  python  Bengal  29000 2021-06-10 111   kjsim
8     203   three   scala  bombay  39000 2021-04-11 111   kjsim
9     204    four     C++  bombay  49000 2021-06-09 111   kjsim
10    205    five     PHP  bombay  59000 2021-06-09 111   kjsim
   "dst" "vidyavihar" 123000 "2018-01-01" gender
1    dst   vidyavihar 123000   2018-01-01      M
2    dst   vidyavihar 123000   2018-01-01      M
3    dst   vidyavihar 123000   2018-01-01      F
4    dst   vidyavihar 123000   2018-01-01      F
5    dst   vidyavihar 123000   2018-01-01      M
6    dst   vidyavihar 123000   2018-01-01      M
7    dst   vidyavihar 123000   2018-01-01      M
8    dst   vidyavihar 123000   2018-01-01      F
9    dst   vidyavihar 123000   2018-01-01      F
10   dst   vidyavihar 123000   2018-01-01      M
> 
```



```
Error: object 'empdata' not found
> empdata = data.frame(
+   rollno = c(101,102,103,104,105),
+   empname = c("pratik", "roshan", "sahil", "vivek", "saumik"),
+   empdept = c("python", "java", ".NET", "HTML", "PHP"),
+   empcity = c("bombay", "bombay", "bombay", "pune", "Bengal"),
+   empsal = c(1000, 2000, 3000, 4000, 5000),
+   empdoj = c("2022-01-10","2022-05-10","2022-03-11", "2022-09-09", "2022-08-09"),
+   stringsAsFactors = FALSE
+ )
> empdata
  rollno empname empdept empcity empsal     empdoj
1    101  pratik  python  bombay   1000 2022-01-10
2    102  roshan    java  bombay   2000 2022-05-10
3    103   sahil    .NET  bombay   3000 2022-03-11
4    104   vivek    HTML    pune   4000 2022-09-09
5    105  saumik     PHP  Bengal   5000 2022-08-09
> empdata
  rollno empname empdept empcity empsal     empdoj
1    101  pratik  python  bombay   1000 2022-01-10
2    102  roshan    java  bombay   2000 2022-05-10
3    103   sahil    .NET  bombay   3000 2022-03-11
4    104   vivek    HTML    pune   4000 2022-09-09
5    105  saumik     PHP  Bengal   5000 2022-08-09
> dbWriteTable(connection,"tbable1",empdata)
[1] TRUE
> 
```



```
exportStats = c(10,23,50,55,5)
countries = c("India","Russia", "USA", "France", "Germany")
x = c(12,34,22,12,3)
pie(x,labels = countries, cex=0.5)

pie()
```

```
> exportStats = c(10,23,50,55,5)
> countries = c("India","Russia", "USA", "France", "Germany")
> x = c(12,34,22,12,3)
> pie(x,labels = countries)
>
> pie()
Error in pie() : argument "x" is missing, with no default
> pie(x,labels = countries, cex=0.5)
> pie()
Error in pie() : argument "x" is missing, with no default
> 
```

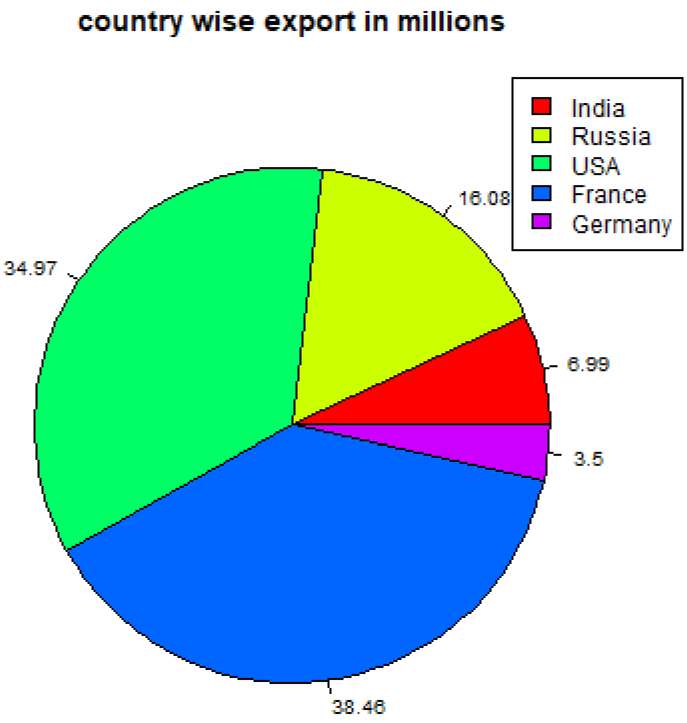
```
month.day19     Named num [1:12] 31 28 31 30 31 30 31 31 30 31 ...
month.name      chr [1:12] "January" "February" "March" "April" "May" "June"…
nm              "Have a best day"
nm2             chr [1:2] "Param" "Singh"
per             num [1:3] 19 41 -13
sck             4L
sck1            num [1:5] 27 32 45 NA 39
sck12           num [1:4] 27 32 45 39
ulaplh          chr [1:52] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "…
v               int [1:11] 10 11 12 13 14 15 16 17 18 19 ...
valid           num [1:4] 27 37 45 35
x               num [1:5] 12 34 22 12 3
```

Data Visualization in R

Program:

```
exportStats = c(10,23,50,55,5)
countries = c("India","Russia", "USA", "France", "Germany")
png("p2.jpeg")
percent = round(100*exportStats/sum(exportStats),2)
#pie(exportStats,labels = percent, cex=0.8)




pie(exportStats,main ="country wise export in millions", labels = percent,
cex=0.8,col=rainbow(length(exportStats)))

legend("topright", legend=countries,cex=1,fill=rainbow(length(exportStats)))

#saving the file
dev.off()
getwd()
```

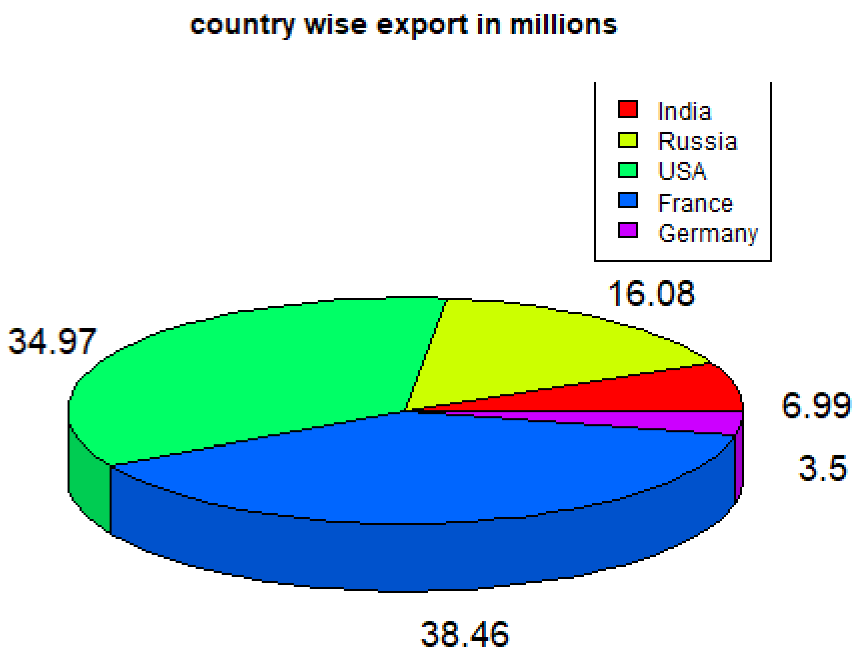
Output:

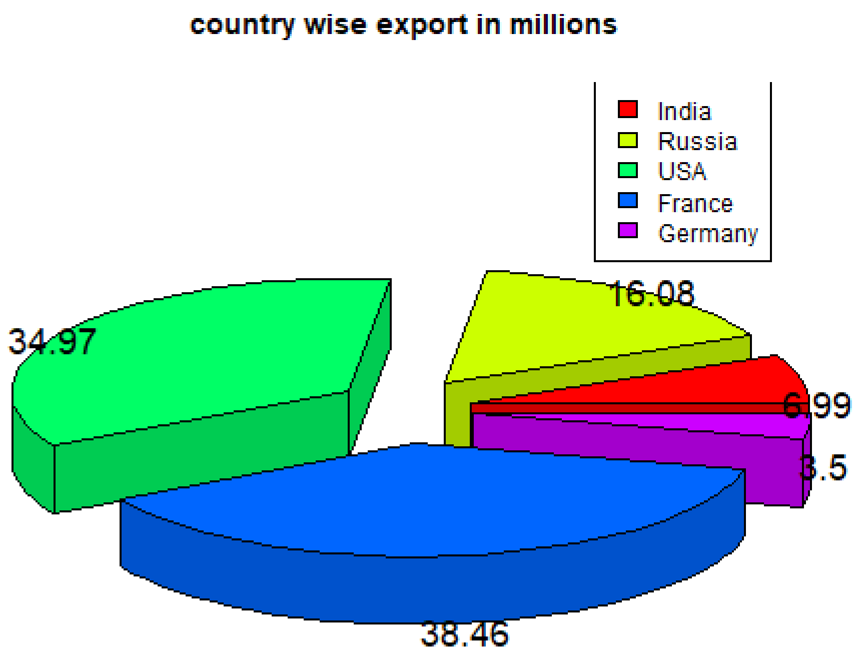


Program

```
exportStats = c(10,23,50,55,5)
countries = c("India","Russia", "USA", "France", "Germany")
png("p4.jpeg")
percent = round(100*exportStats/sum(exportStats),2)
#pie(exportStats,labels = percent, cex=0.8)




#pie(exportStats,main ="country wise export in millions", labels = percent,
cex=0.8,col=rainbow(length(exportStats)))

pie3D(exportStats,explode=0.2,main ="country wise export in millions", labels = percent,
cex=0.5,col=rainbow(length(exportStats)))

legend("topright", legend=countries,cex=1,fill=rainbow(length(exportStats)))

#saving the file
```

```
dev.off()
getwd()
```

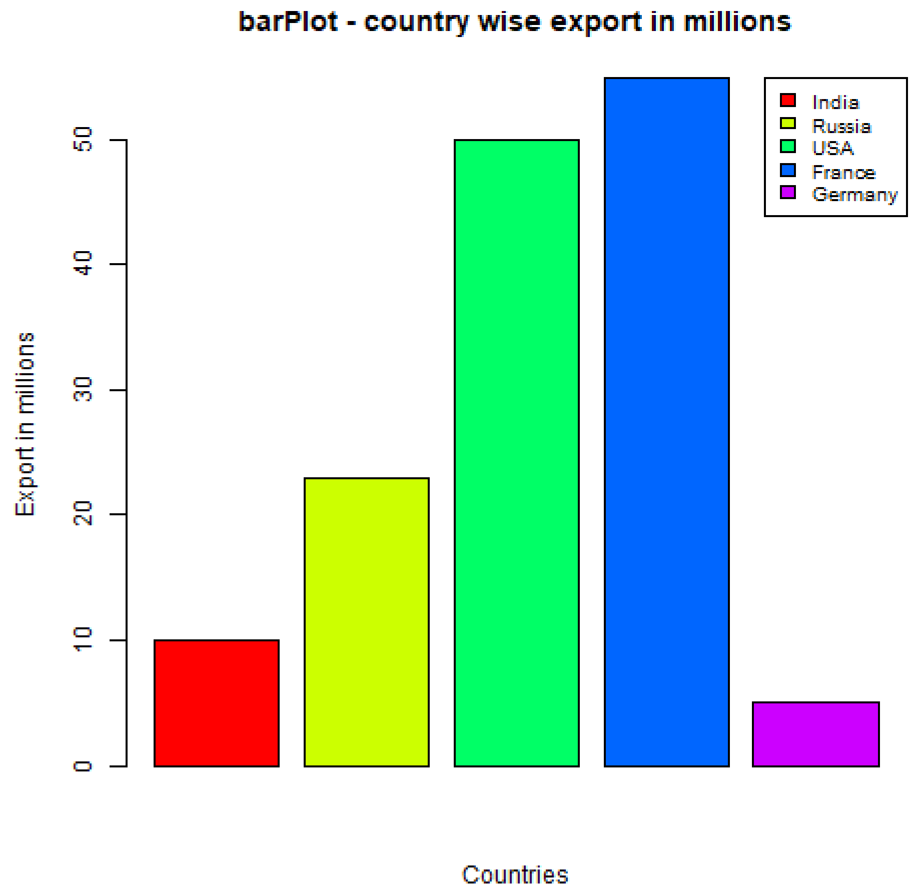




Program:

```
#Barplot

png("barPlot1.jpeg")
emportStats = c(10,23,50,55,5)
countries = c("India","Russia", "USA", "France", "Germany")

barplot(exportStats,xlab = "Countries" , ylab = "Export in millions",
        main = "barPlot - country wise export in millions",
        col = rainbow(length(exportStats))
)

legend("topright",legend = countries,cex=0.8,fill = rainbow(length(exportStats)))

dev.off()
```

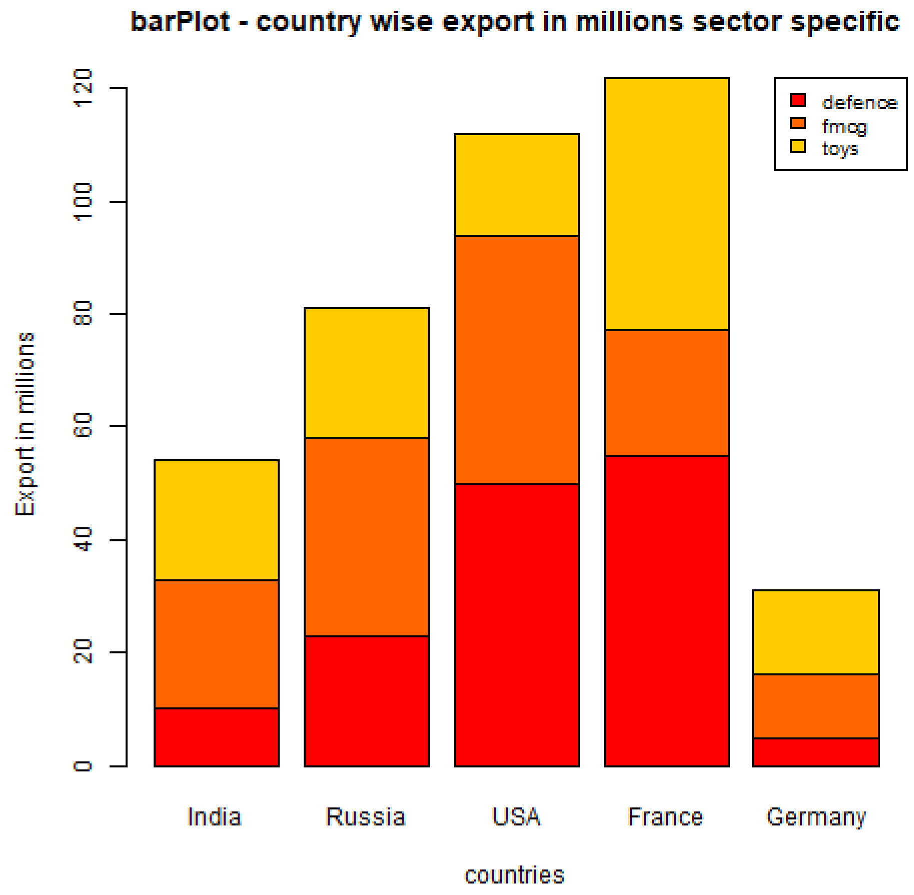
Output:



Program

```
#stacked bar graph
png("barpot2.png")
exportStats = matrix(c(10,23,50,55,5,23,35,44,22,11,21,23,18,45,15),
                     nrow = 3, ncol=5, byrow=TRUE)
countries = c("India","Russia", "USA", "France", "Germany")
sectors=c("defence","fmcg","toys")
barplot(exportStats,names.arg=countries, xlab ="countries", ylab = "Export in millions",
        main="barPlot - country wise export in millions sector specific",
        col=rainbow(length(exportStats)))
legend("topright",legend = sectors,cex=0.8,fill = rainbow(length(exportStats)))

dev.off()
```

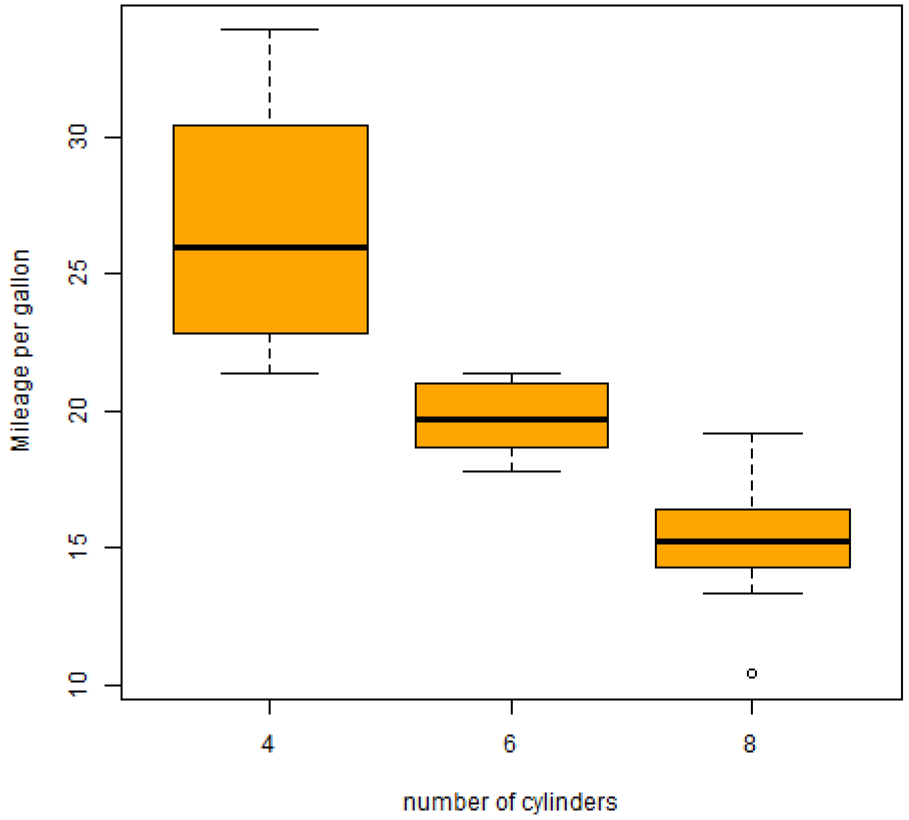
Output:



Program:

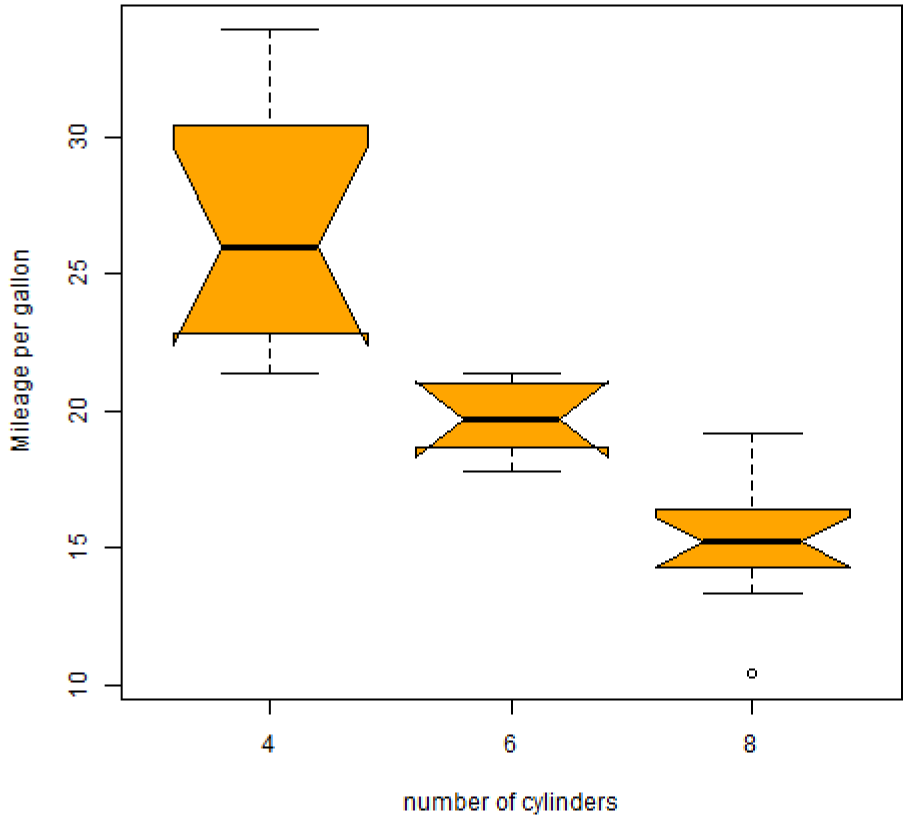
```
#box plot
mtcars

png("boxPlot.png")
boxplot(mtcars$mpg~mtcars$cyl, xlab="number of cylinders", ylab="Mileage per gallon",
col="orange")
#boxplot()

dev.off()
```

Output:



Program:

```
png("boxPlot.png")
boxplot(mtcars$mpg~mtcars$cyl, xlab="number of cylinders", ylab="Mileage per gallon",
col="orange",notch=TRUE)
#boxplot()

dev.off()
```

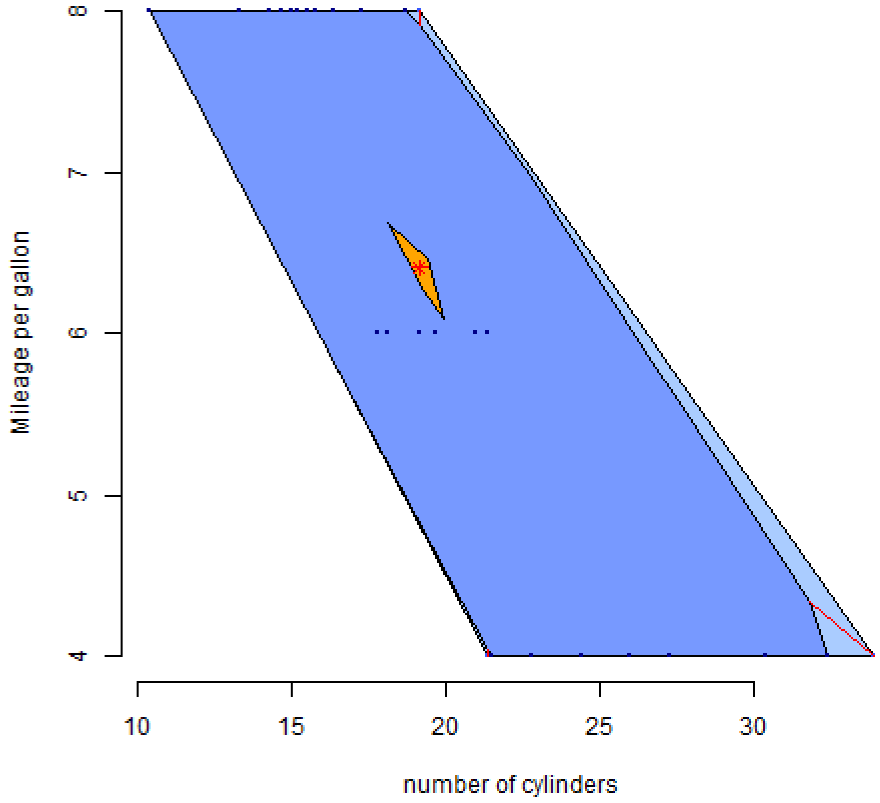


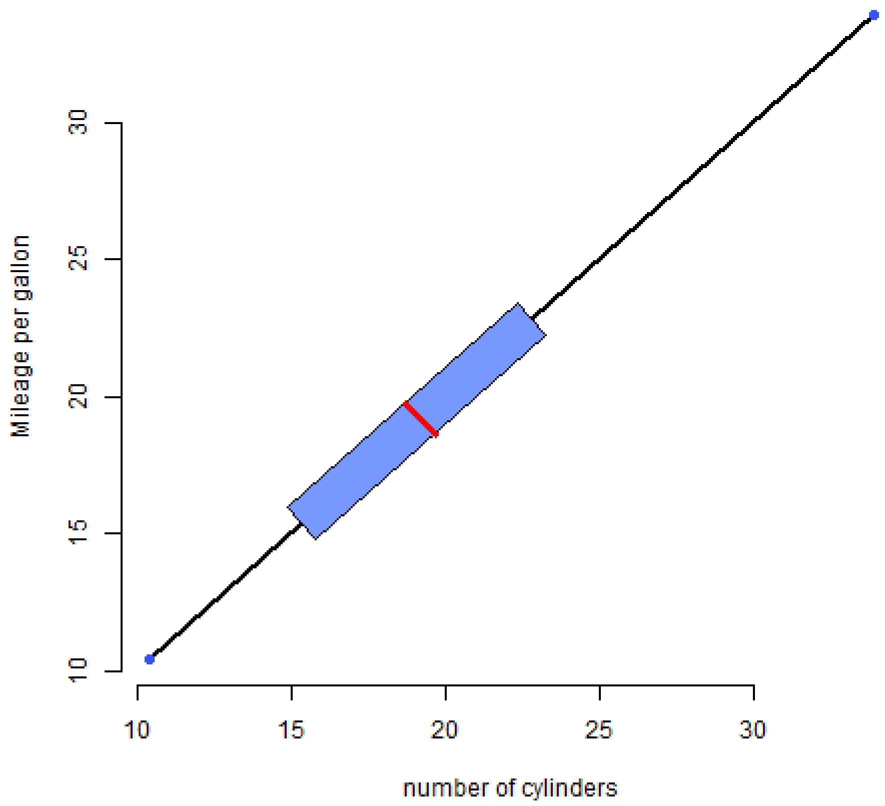
BoxPlot
Program:

```
#box plot
mtcars

png("boxPlot.png")
boxplot(mtcars$mpg~mtcars$cyl,
        xlab="number of cylinders",
        ylab="Mileage per gallon",
        col="orange",
        notch=TRUE,
        horizontal = TRUE)
#boxplot()

dev.off()

library(aplpack)
png("2DboxPlot.png")
```

```
bagplot(mtcars$mpg,mtcars$cyl,
        xlab = "number of cylinders", ylab = "Mileage per gallon")
dev.off()

png("2DboxPlot1.png")
bagplot(mtcars$mpg,mtcars$mpg,
        xlab = "number of cylinders", ylab = "Mileage per gallon")
dev.off()
```

Output:

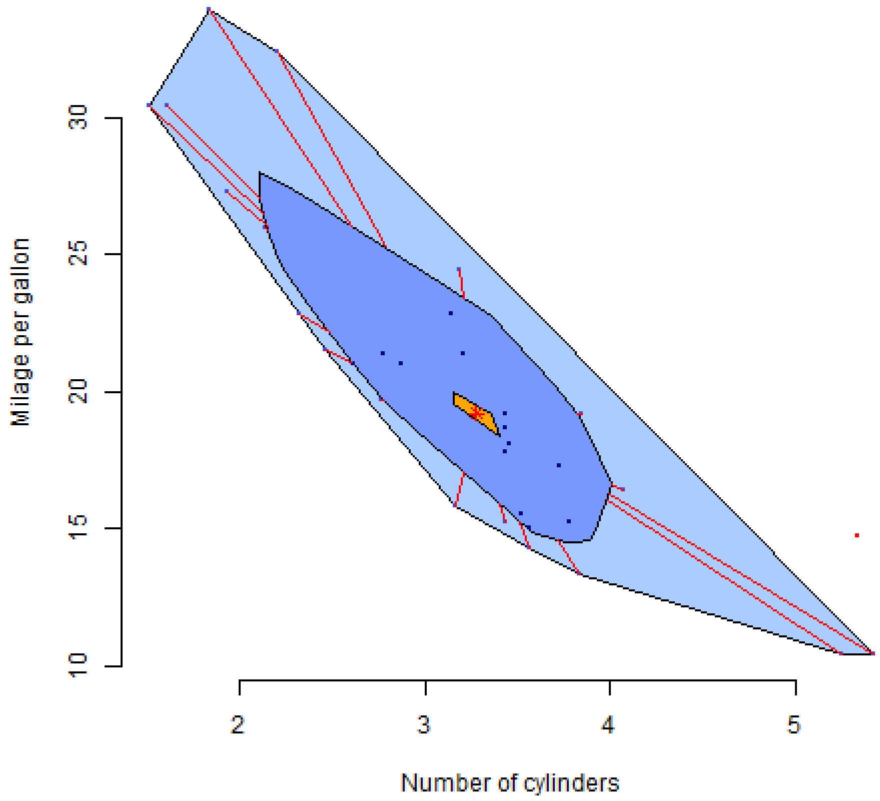




Program:

```
library(aplpack)

png("2DboxPlot1.jpeg")
bagplot(mtcars$wt,mtcars$mpg,
        xlab = "Number of cylinders",
        ylab = "Milage per gallon"

)
dev.off()
```

Output:



Program:

```
airquality
#setwd("D:\\Work")
png("histogram.png")
hist(airquality$Temp,
    main = "Temp observed at ch. Shivaji Maharaj Int Ariport",
    xlim = c(50,100),
    col = "darkmagenta",
    breaks = c(10,40,100,70)
)

dev.off()
```

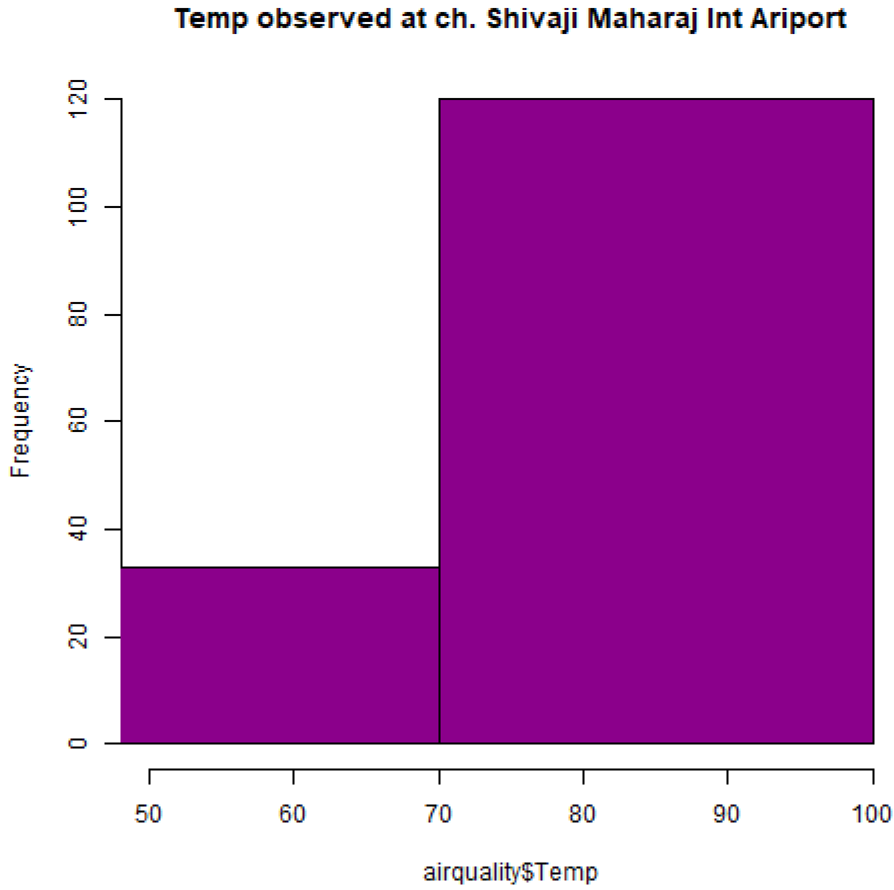
Output:

WebScrapping

Program:

```
airquality
#setwd("D:\\Work")
png("histogram.png")
hist(airquality$Temp,
    main = "Temp observed at ch. Shivaji Maharaj Int Ariport",
    xlim = c(50,100),
    col = "darkmagenta",
    breaks = c(10,40,100,70)
)

dev.off()
```

Output:

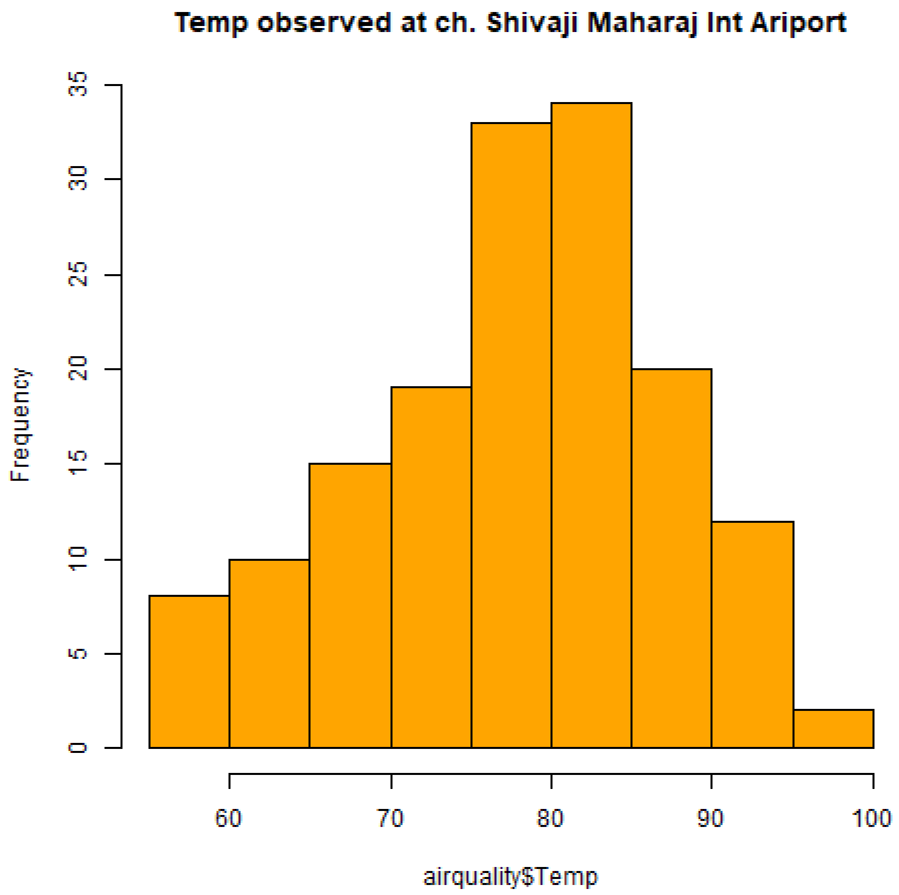


Program:

```
airquality
#setwd("D:\\")
```

```
png("histogram1.png")
hist(airquality$Temp,
    main = "Temp observed at ch. Shivaji Maharaj Int Ariport",

    col = "orange",    # breaks = c(10,40,100,70)
)

dev.off()
```

Output:



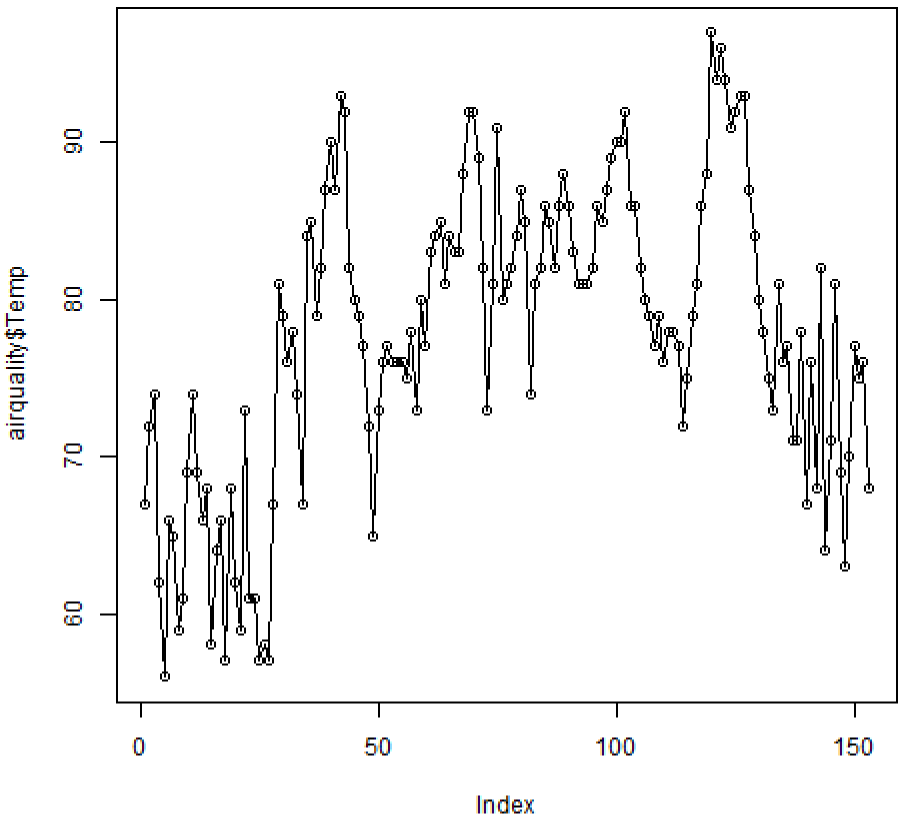
Line Graph

Program:

```
#Line graph

png("lineGraph.png")



plot(airquality$Temp, type = "o")
```

```
dev.off()
```

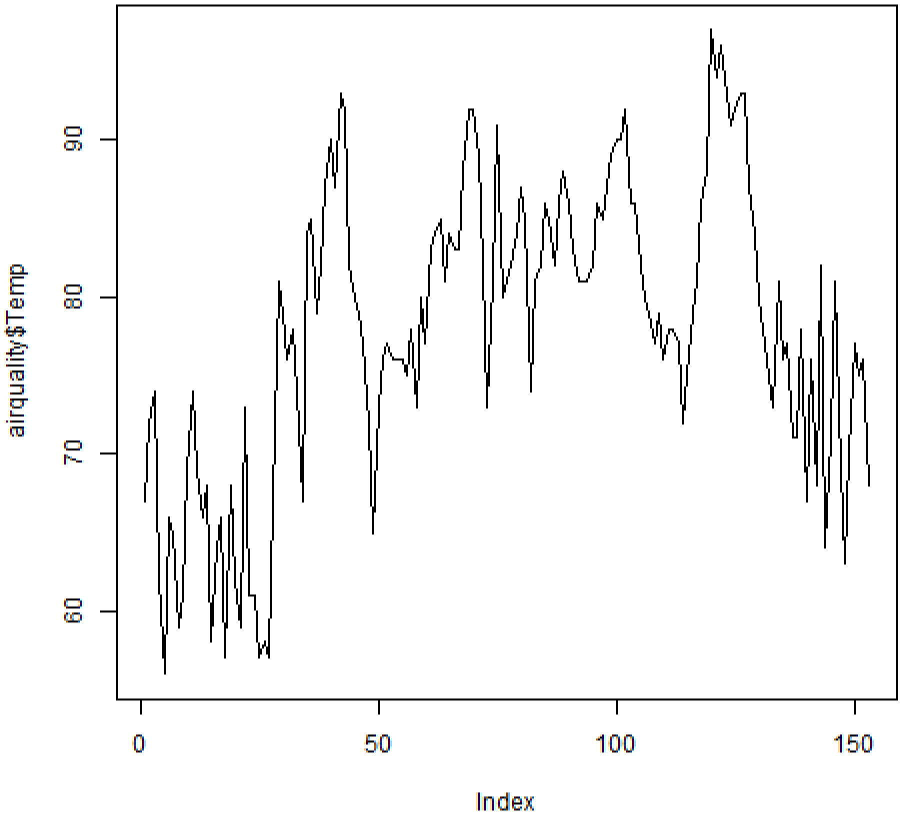
Output:



Program:

```
#Line graph

png("lineGraph1.png")


plot(airquality$Temp, type = "l")

dev.off()
```

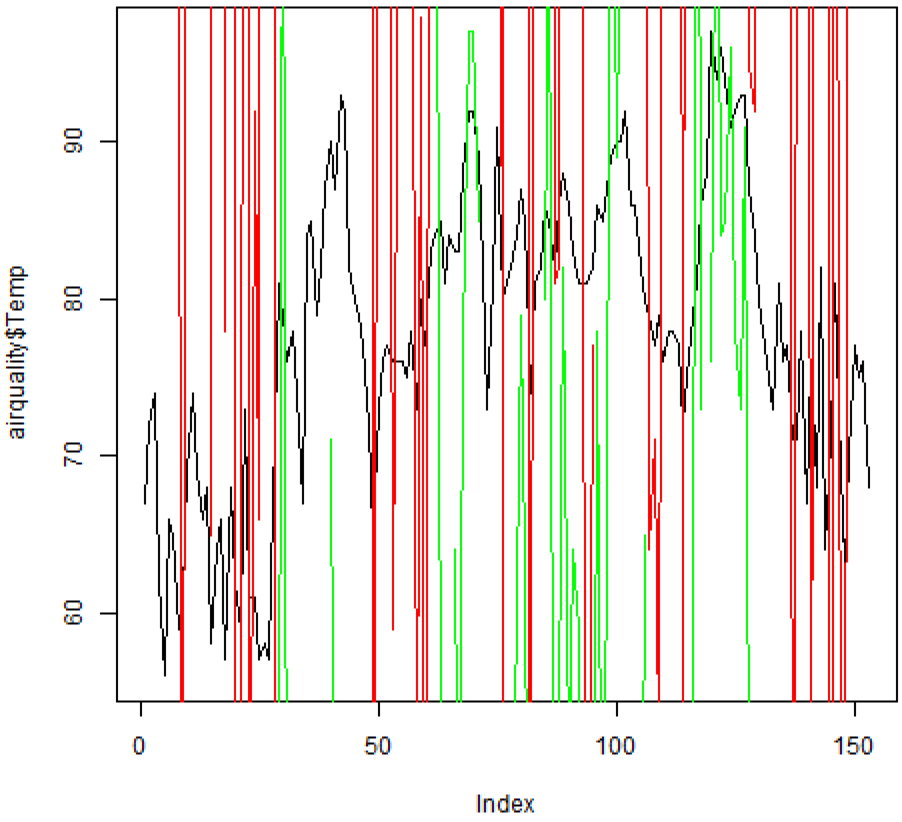
Output:



Program:

```
#Linegraph

png("lineGraph2.png")


plot(airquality$Temp, type = "l")
lines(airquality$Ozone, type="l", col ="green")

lines(airquality$Solar.R, type="l", col ="red")

dev.off()
```

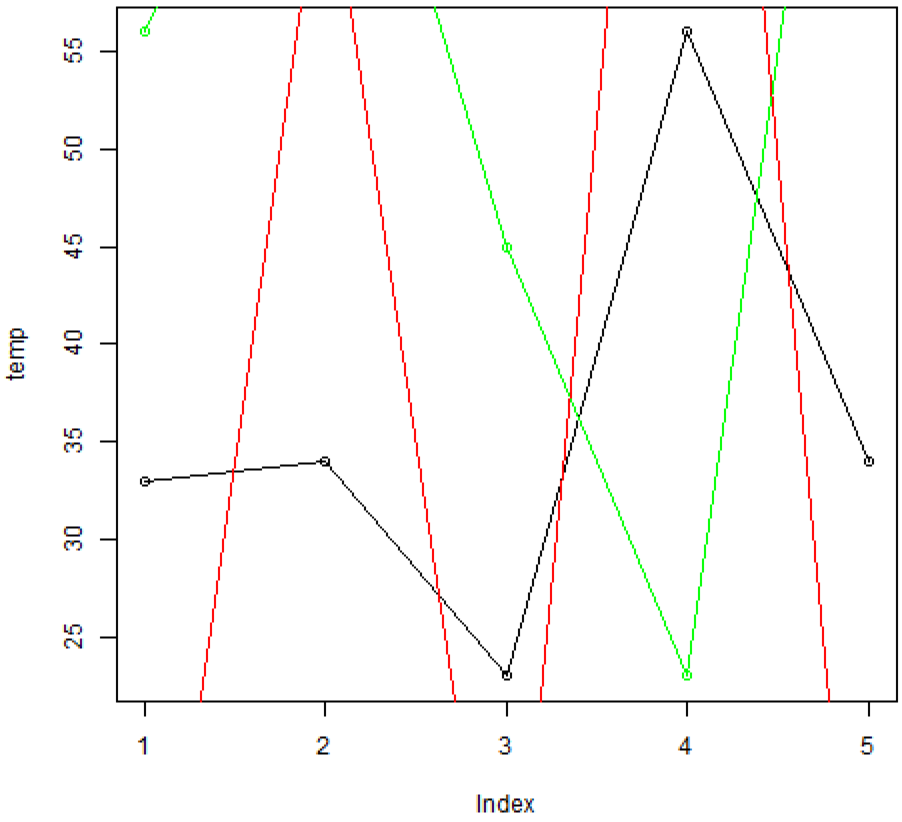
Output:

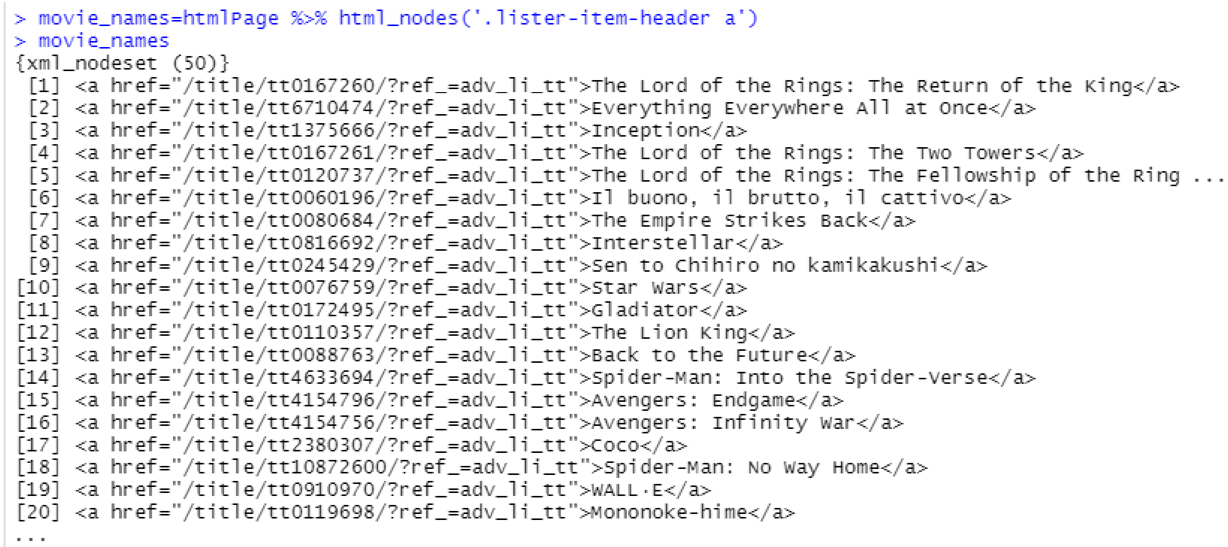


Program:

```
temp = c(33,34,23,56,34)
humidity = c(56,76,45,23,87)
numbers = c(2,66,4,99,0)

png("lineGraph3.png")
plot(temp, type = "o")
lines(humidity, type = "o",col = "green")
lines(numbers,type = "o", col = "red")

dev.off()
```

Output:


temp
25
30
35
40
45
50
55
1
2
3
4
5
Index

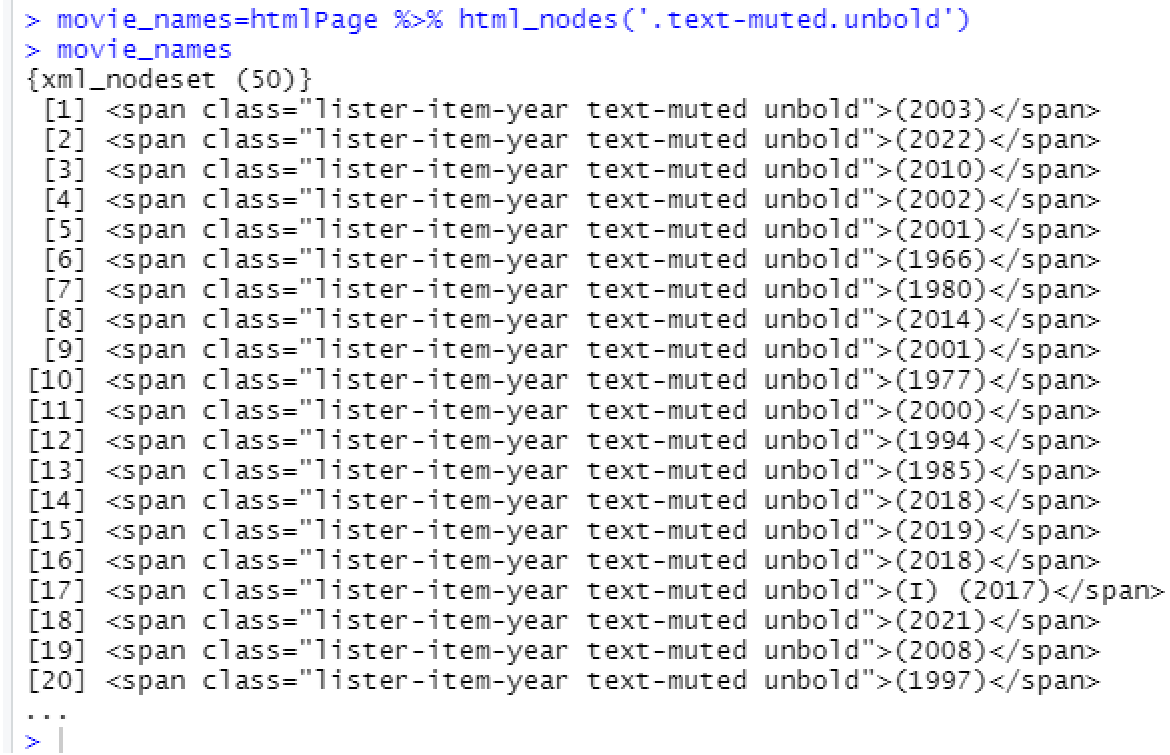


```
> movie_names=htmlPage %>% html_nodes('.lister-item-header a')
> movie_names
{xml_nodeset (50)}
 [1] <a href="/title/tt0167260/?ref_=adv_li_tt">The Lord of the Rings: The Return of the King</a>
 [2] <a href="/title/tt6710474/?ref_=adv_li_tt">Everything Everywhere All at Once</a>
 [3] <a href="/title/tt1375666/?ref_=adv_li_tt">Inception</a>
 [4] <a href="/title/tt0167261/?ref_=adv_li_tt">The Lord of the Rings: The Two Towers</a>
 [5] <a href="/title/tt0120737/?ref_=adv_li_tt">The Lord of the Rings: The Fellowship of the Ring ...
 [6] <a href="/title/tt0060196/?ref_=adv_li_tt">Il buono, il brutto, il cattivo</a>
 [7] <a href="/title/tt0080684/?ref_=adv_li_tt">The Empire Strikes Back</a>
 [8] <a href="/title/tt0816692/?ref_=adv_li_tt">Interstellar</a>
 [9] <a href="/title/tt0245429/?ref_=adv_li_tt">Sen to Chihiro no kamikakushi</a>
[10] <a href="/title/tt0076759/?ref_=adv_li_tt">Star Wars</a>
[11] <a href="/title/tt0172495/?ref_=adv_li_tt">Gladiator</a>
[12] <a href="/title/tt0110357/?ref_=adv_li_tt">The Lion King</a>
[13] <a href="/title/tt0088763/?ref_=adv_li_tt">Back to the Future</a>
[14] <a href="/title/tt4633694/?ref_=adv_li_tt">Spider-Man: Into the Spider-Verse</a>
[15] <a href="/title/tt4154796/?ref_=adv_li_tt">Avengers: Endgame</a>
[16] <a href="/title/tt4154756/?ref_=adv_li_tt">Avengers: Infinity War</a>
[17] <a href="/title/tt2380307/?ref_=adv_li_tt">Coco</a>
[18] <a href="/title/tt10872600/?ref_=adv_li_tt">Spider-Man: No Way Home</a>
[19] <a href="/title/tt0910970/?ref_=adv_li_tt">WALL·E</a>
[20] <a href="/title/tt0119698/?ref_=adv_li_tt">Mononoke-hime</a>
...
```

```
library(rvest)

library(dplyr)

imbd_link="https://www.imdb.com/search/title/?title_type=feature&num_votes=25000,&genres=adventure&sort=user_rating,desc"

#read_html() returns the html code of webpage
htmlPage = read_html(imbd_link)
htmlPage

movie_names=htmlPage %>% html_nodes('.lister-item-header a')
Movie_names
```

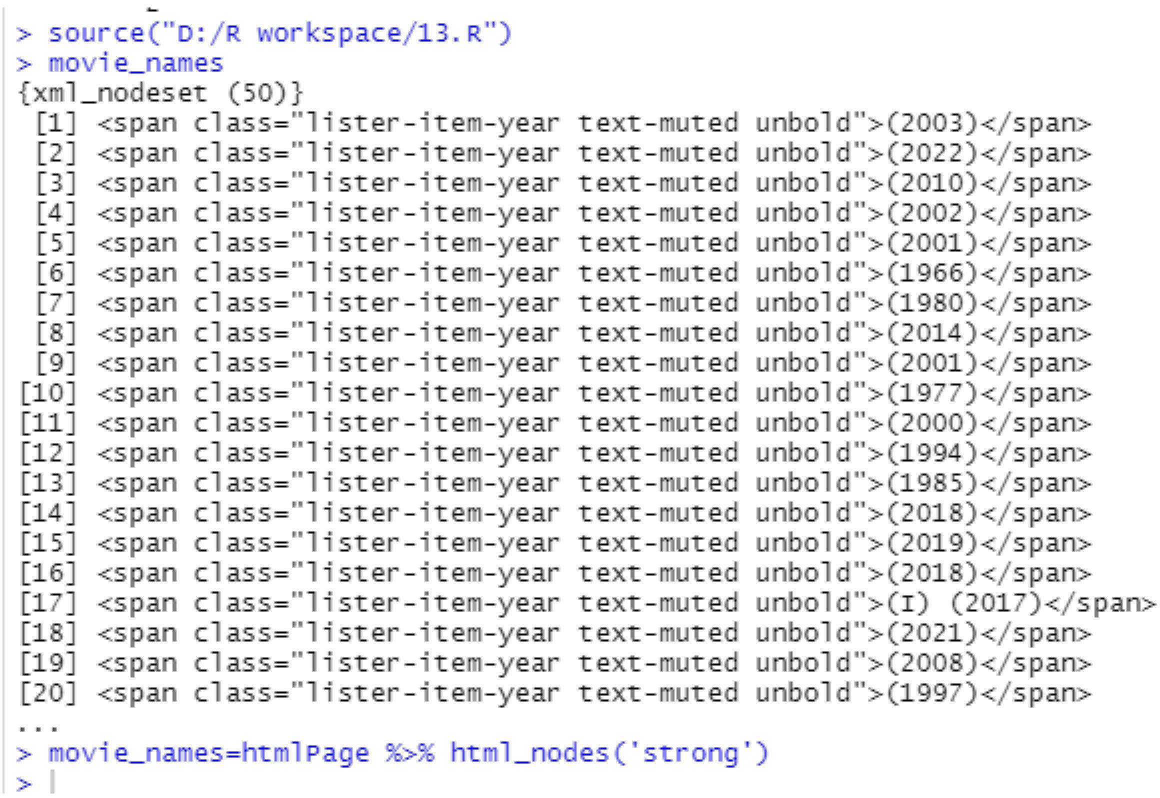
Program:

```
library(rvest)

library(dplyr)

imbd_link="https://www.imdb.com/search/title/?title_type=feature&num_votes=25000,&genres=
adventure&sort=user_rating,desc"

#read_html() returns the html code of webpage
htmlPage = read_html(imbd_link)
htmlPage

movie_names=htmlPage %>% html_nodes('.lister-item-header a')

movie_names
movie_names=htmlPage %>% html_nodes('.text-muted.unbold')
```



```
> movie_names=htmlPage %>% html_nodes('.text-muted.unbold')
> movie_names
{xml_nodeset (50)}
 [1] <span class="lister-item-year text-muted unbold">(2003)</span>
 [2] <span class="lister-item-year text-muted unbold">(2022)</span>
 [3] <span class="lister-item-year text-muted unbold">(2010)</span>
 [4] <span class="lister-item-year text-muted unbold">(2002)</span>
 [5] <span class="lister-item-year text-muted unbold">(2001)</span>
 [6] <span class="lister-item-year text-muted unbold">(1966)</span>
 [7] <span class="lister-item-year text-muted unbold">(1980)</span>
 [8] <span class="lister-item-year text-muted unbold">(2014)</span>
 [9] <span class="lister-item-year text-muted unbold">(2001)</span>
[10] <span class="lister-item-year text-muted unbold">(1977)</span>
[11] <span class="lister-item-year text-muted unbold">(2000)</span>
[12] <span class="lister-item-year text-muted unbold">(1994)</span>
[13] <span class="lister-item-year text-muted unbold">(1985)</span>
[14] <span class="lister-item-year text-muted unbold">(2018)</span>
[15] <span class="lister-item-year text-muted unbold">(2019)</span>
[16] <span class="lister-item-year text-muted unbold">(2018)</span>
[17] <span class="lister-item-year text-muted unbold">(I) (2017)</span>
[18] <span class="lister-item-year text-muted unbold">(2021)</span>
[19] <span class="lister-item-year text-muted unbold">(2008)</span>
[20] <span class="lister-item-year text-muted unbold">(1997)</span>
...
> 
```

```
movie_names
movie_names=htmlPage %>% html_nodes('strong')
```

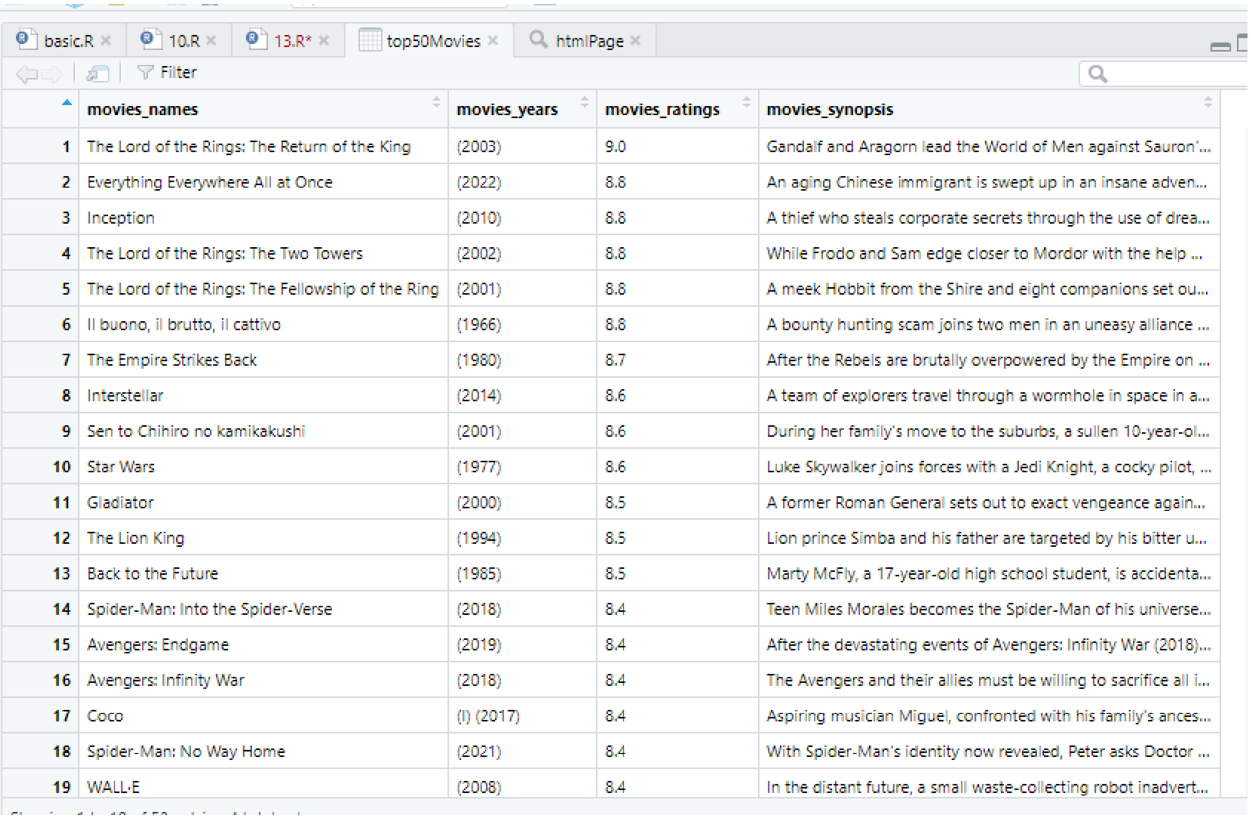


```
> source("D:/R workspace/13.R")
> movie_names
{xml_nodeset (50)}
 [1] <span class="lister-item-year text-muted unbold">(2003)</span>
 [2] <span class="lister-item-year text-muted unbold">(2022)</span>
 [3] <span class="lister-item-year text-muted unbold">(2010)</span>
 [4] <span class="lister-item-year text-muted unbold">(2002)</span>
 [5] <span class="lister-item-year text-muted unbold">(2001)</span>
 [6] <span class="lister-item-year text-muted unbold">(1966)</span>
 [7] <span class="lister-item-year text-muted unbold">(1980)</span>
 [8] <span class="lister-item-year text-muted unbold">(2014)</span>
 [9] <span class="lister-item-year text-muted unbold">(2001)</span>
[10] <span class="lister-item-year text-muted unbold">(1977)</span>
[11] <span class="lister-item-year text-muted unbold">(2000)</span>
[12] <span class="lister-item-year text-muted unbold">(1994)</span>
[13] <span class="lister-item-year text-muted unbold">(1985)</span>
[14] <span class="lister-item-year text-muted unbold">(2018)</span>
[15] <span class="lister-item-year text-muted unbold">(2019)</span>
[16] <span class="lister-item-year text-muted unbold">(2018)</span>
[17] <span class="lister-item-year text-muted unbold">(I) (2017)</span>
[18] <span class="lister-item-year text-muted unbold">(2021)</span>
[19] <span class="lister-item-year text-muted unbold">(2008)</span>
[20] <span class="lister-item-year text-muted unbold">(1997)</span>
...
> movie_names=htmlPage %>% html_nodes('strong')
> 
```

Selector Gadget

```
library(rvest)
library(dplyr)
imdb_link="https://www.imdb.com/search/title/?title_type=feature&num_votes=25000,&genres=adventure&sort=user_rating,desc"
htmlPage=read_html(imdb_link)

movies_names=htmlPage %>% html_nodes('.lister-item-header a')  %>%
  html_text()

movies_years=htmlPage %>% html_nodes('.text-muted.unbold')  %>%
  html_text()

movies_ratings=htmlPage %>% html_nodes('.ratings-imdb-rating strong')  %>%
  html_text()

movies_synopsis=htmlPage %>% html_nodes('.ratings-bar+ .text-muted')  %>%
  html_text()
```

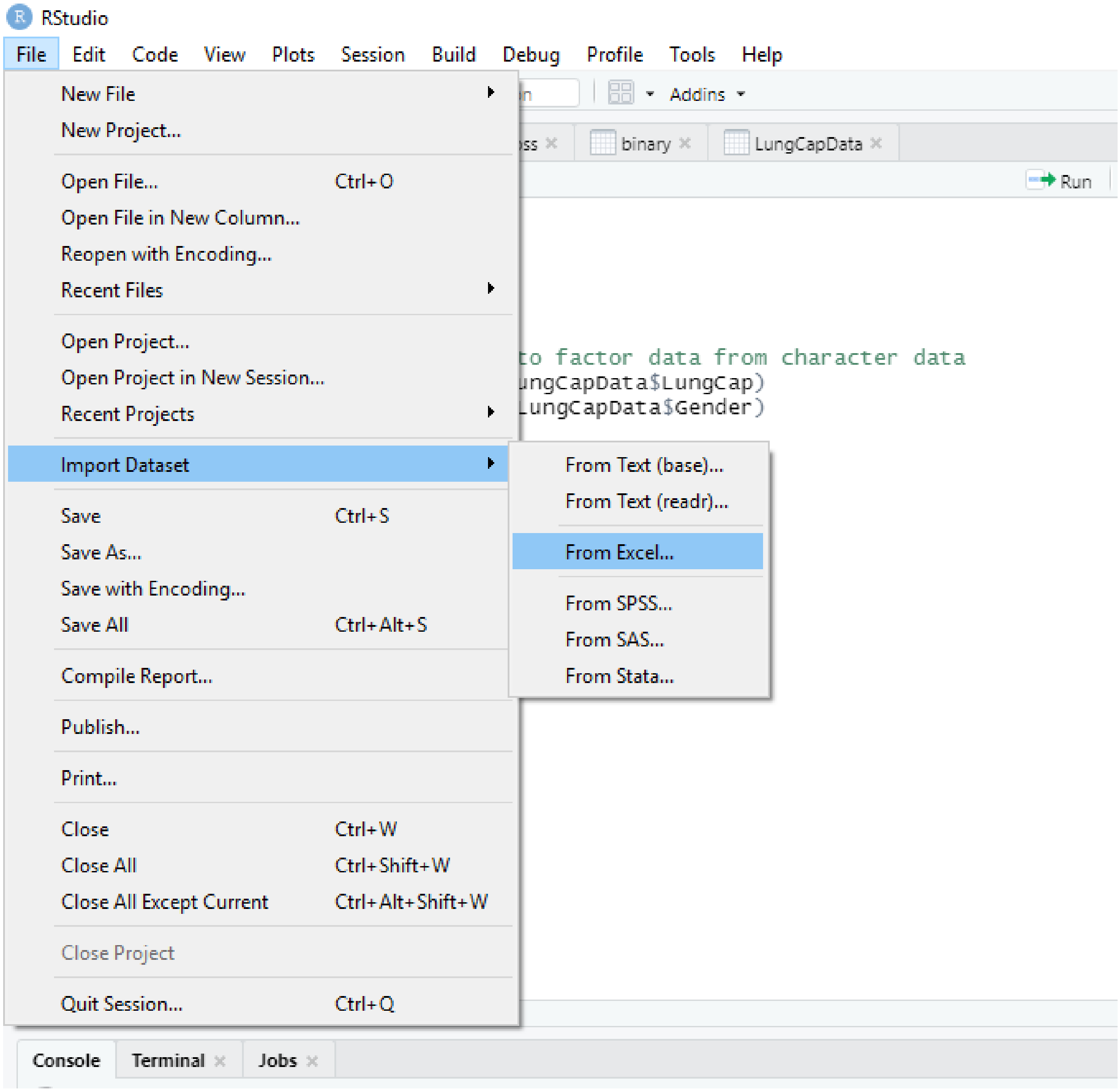
```
top50Movies=data.frame(movies_names,movies_years,movies_ratings,movies_synopsis)
```



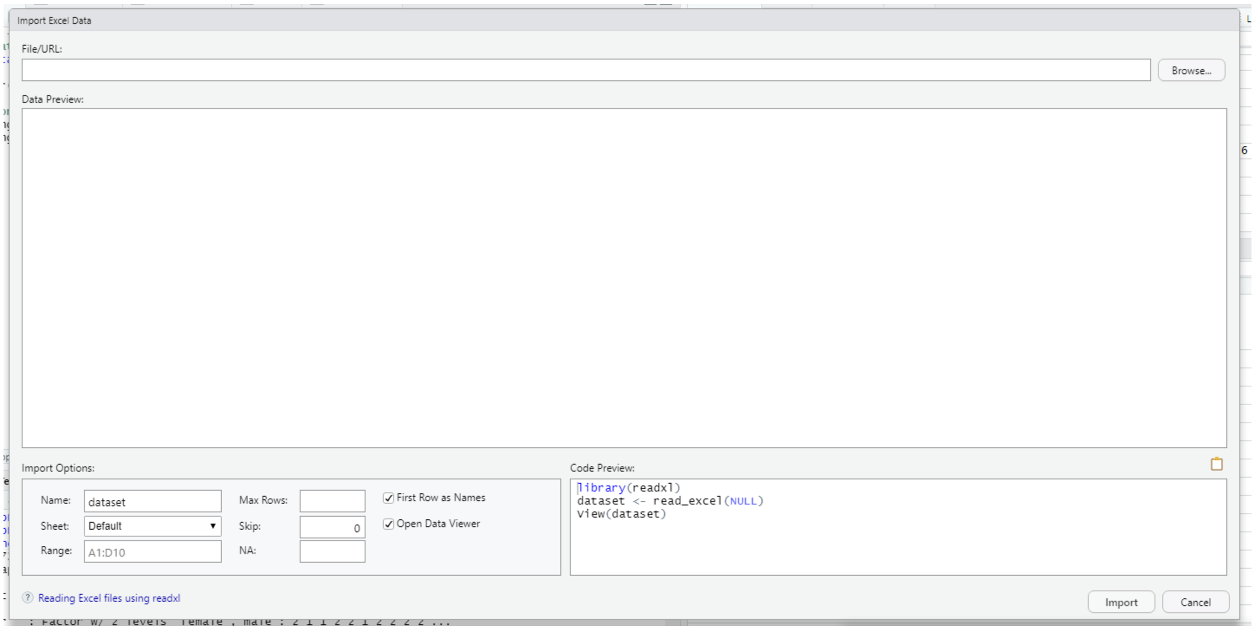
	movies_names	movies_years	movies_ratings	movies_synopsis
1	The Lord of the Rings: The Return of the King	(2003)	9.0	Gandalf and Aragorn lead the World of Men against Sauron'...
2	Everything Everywhere All at Once	(2022)	8.8	An aging Chinese immigrant is swept up in an insane adven...
3	Inception	(2010)	8.8	A thief who steals corporate secrets through the use of drea...
4	The Lord of the Rings: The Two Towers	(2002)	8.8	While Frodo and Sam edge closer to Mordor with the help ...
5	The Lord of the Rings: The Fellowship of the Ring	(2001)	8.8	A meek Hobbit from the Shire and eight companions set ou...
6	Il buono, il brutto, il cattivo	(1966)	8.8	A bounty hunting scam joins two men in an uneasy alliance ...
7	The Empire Strikes Back	(1980)	8.7	After the Rebels are brutally overpowered by the Empire on ...
8	Interstellar	(2014)	8.6	A team of explorers travel through a wormhole in space in a...
9	Sen to Chihiro no kamikakushi	(2001)	8.6	During her family's move to the suburbs, a sullen 10-year-ol...
10	Star Wars	(1977)	8.6	Luke Skywalker joins forces with a Jedi Knight, a cocky pilot, ...
11	Gladiator	(2000)	8.5	A former Roman General sets out to exact vengeance again...
12	The Lion King	(1994)	8.5	Lion prince Simba and his father are targeted by his bitter u...
13	Back to the Future	(1985)	8.5	Marty McFly, a 17-year-old high school student, is accidenta...
14	Spider-Man: Into the Spider-Verse	(2018)	8.4	Teen Miles Morales becomes the Spider-Man of his universe...
15	Avengers: Endgame	(2019)	8.4	After the devastating events of Avengers: Infinity War (2018)...
16	Avengers: Infinity War	(2018)	8.4	The Avengers and their allies must be willing to sacrifice all i...
17	Coco	(I) (2017)	8.4	Aspiring musician Miguel, confronted with his family's ances...
18	Spider-Man: No Way Home	(2021)	8.4	With Spider-Man's identity now revealed, Peter asks Doctor ...
19	WALL·E	(2008)	8.4	In the distant future, a small waste-collecting robot inadvert...

Comparing Mean values in R analytics

How to import Data Sets



```
to factor data from character data
ungCapData$LungCap)
LungCapData$Gender)
```



File/URL:

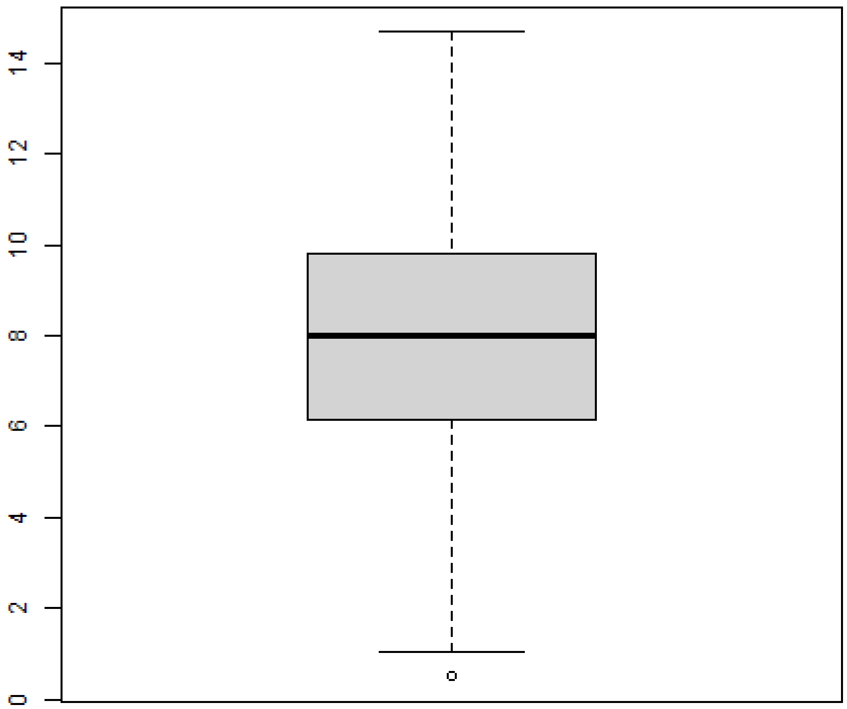
Name: dataset
Sheet: Default ▾
Range: A1:D10
Max Rows:
Skip: 0
NA:
☑ First Row as Names
☑ Open Data Viewer

```
library(readxl)
dataset <- read_excel(NULL)
View(dataset)
```

Program:

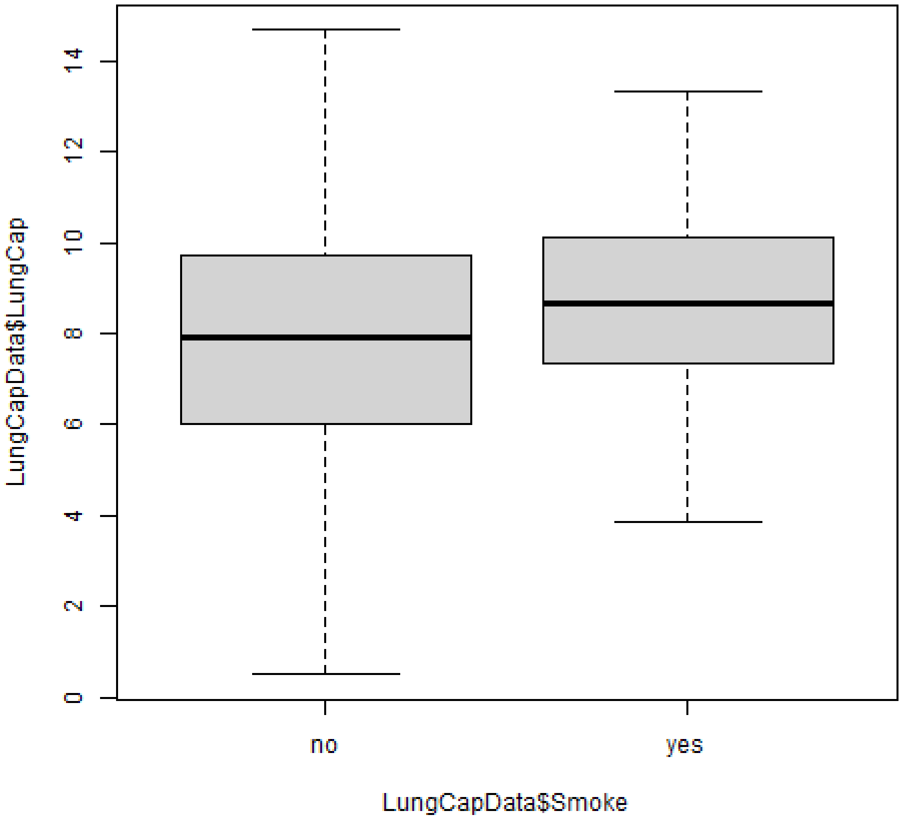
```
#t -test
#data set used
attach(LungCapData)

str(LungCapData)

#converting smoke and gender into factor data from character data
LungCapData$Smoke = as.factor(LungCapData$LungCap)
LungCapData$Gender = as.factor(LungCapData$Gender)

LungCapData = readLines(file.choose()) #for text file
            #read.csv for csv file
            #read_xlx for excel file

data = read_xls(file.choose())
data

setwd("D:\\R workspace\\Graphs")
getwd()


png("LungCapacityBoxPlot.png")
boxplot(data$LungCap)
dev.off()
```

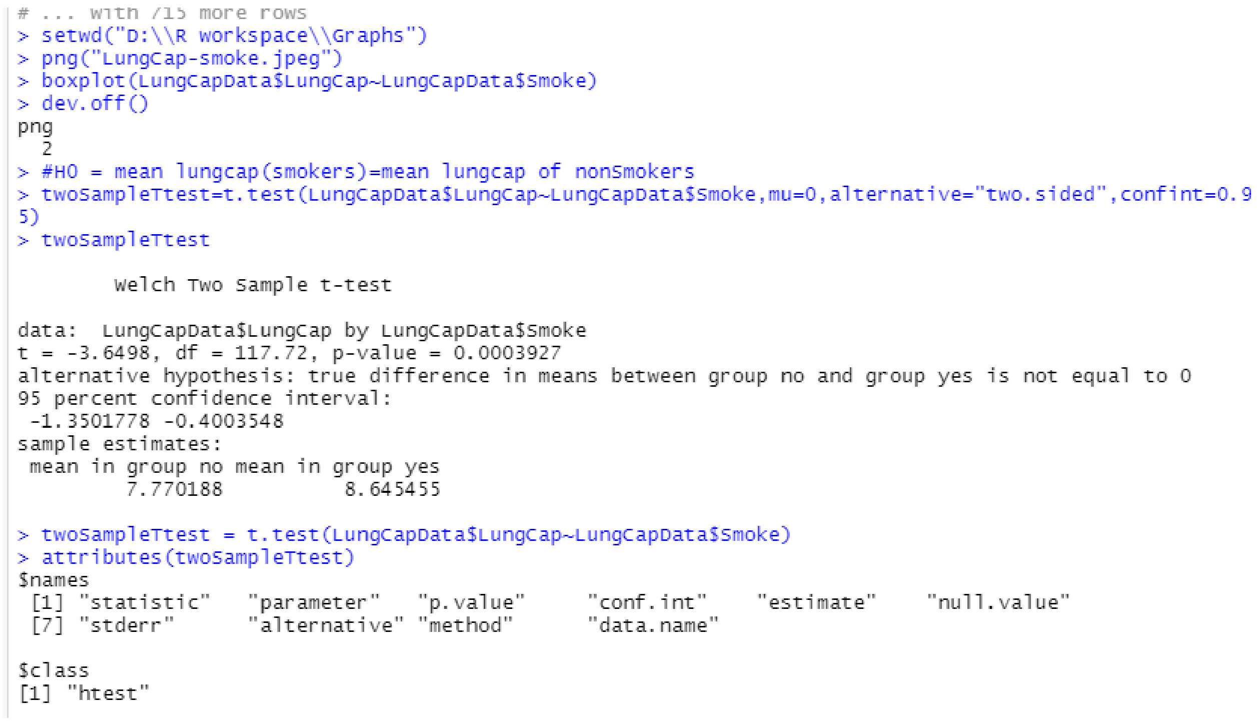
Output:



Two sample t test
Program:

```
data
setwd("D:\\R workspace\\Graphs")
#boxplot for two sample/ two side t test

png("LungCap-smoke.jpeg")
boxplot(LungCapData$LungCap~LungCapData$Smoke)
dev.off()


#H0 = mean lungcap(smokers)=mean lungcap of nonSmokers
twoSampleTtest=t.test(LungCapData$LungCap~LungCapData$Smoke,mu=0,alternative="two.s
ided",confint=0.95)
twoSampleTtest

twoSampleTtest = t.test(LungCapData$LungCap~LungCapData$Smoke)
#Two sample test is default in t.test() even all parameters are
```


0
2
4
6
8
10
12
14
LungCapData$LungCap
no
yes
LungCapData$Smoke



```
# ... with 715 more rows
> setwd("D:\\R workspace\\Graphs")
> png("LungCap-smoke.jpeg")
> boxplot(LungCapData$LungCap~LungCapData$Smoke)
> dev.off()
png
  2
> #H0 = mean lungcap(smokers)=mean lungcap of nonSmokers
> twoSampleTtest=t.test(LungCapData$LungCap~LungCapData$Smoke,mu=0,alternative="two.sided",confint=0.9
5)
> twoSampleTtest

        Welch Two Sample t-test

data:  LungCapData$LungCap by LungCapData$Smoke
t = -3.6498, df = 117.72, p-value = 0.0003927
alternative hypothesis: true difference in means between group no and group yes is not equal to 0
95 percent confidence interval:
 -1.3501778 -0.4003548
sample estimates:
 mean in group no mean in group yes
         7.770188          8.645455

> twoSampleTtest = t.test(LungCapData$LungCap~LungCapData$Smoke)
> attributes(twoSampleTtest)
$names
 [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"    "null.value"
 [7] "stderr"      "alternative" "method"      "data.name"

$class
[1] "htest"
```

Anova test

#anova test

getwd()

attach(DietWeightLoss)

str(DietWeightLoss)

DietWeightLoss$Diet = as.factor(DietWeightLoss$Diet)

levels(DietWeightLoss$Diet)

png("WeightLoss~DietType.png")

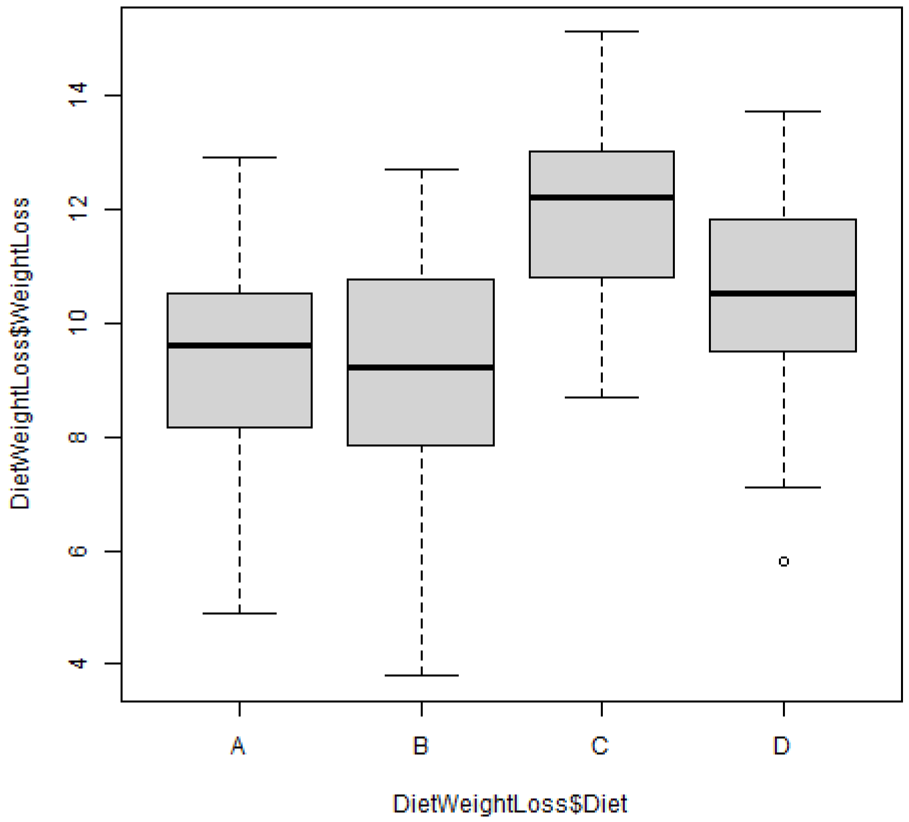
boxplot(DietWeightLoss$WeightLoss~DietWeightLoss$Diet)

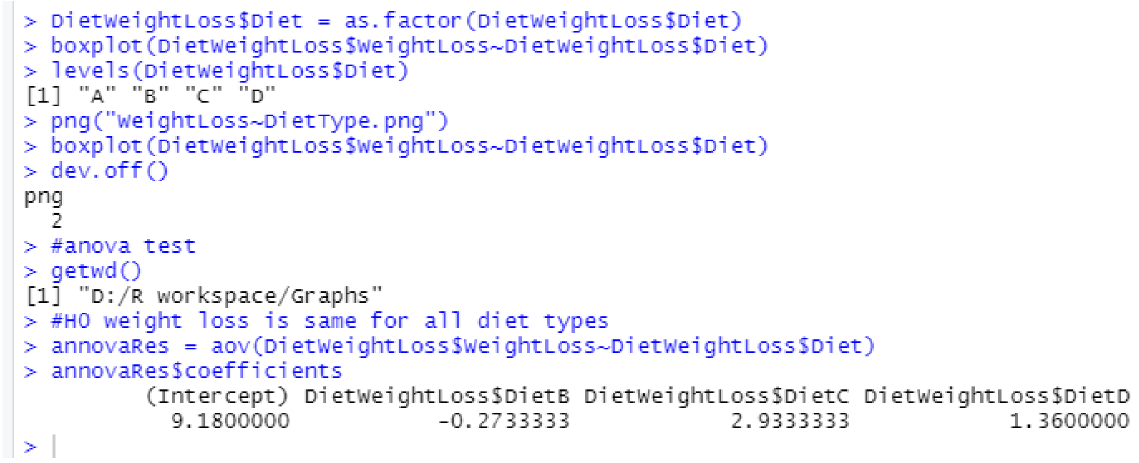
dev.off()

#H0 weight loss is same for all diet types

annovaRes = aov(DietWeightLoss$WeightLoss~DietWeightLoss$Diet)

annovaRes$coefficients





```
> DietWeightLoss$Diet = as.factor(DietWeightLoss$Diet)
> boxplot(DietWeightLoss$WeightLoss~DietWeightLoss$Diet)
> levels(DietWeightLoss$Diet)
[1] "A" "B" "C" "D"
> png("WeightLoss~DietType.png")
> boxplot(DietWeightLoss$WeightLoss~DietWeightLoss$Diet)
> dev.off()
png
  2
> #anova test
> getwd()
[1] "D:/R workspace/Graphs"
> #H0 weight loss is same for all diet types
> annovaRes = aov(DietWeightLoss$WeightLoss~DietWeightLoss$Diet)
> annovaRes$coefficients
        (Intercept) DietWeightLoss$DietB DietWeightLoss$DietC DietWeightLoss$DietD
          9.1800000           -0.2733333            2.9333333            1.3600000
> |
```

Chi square Test

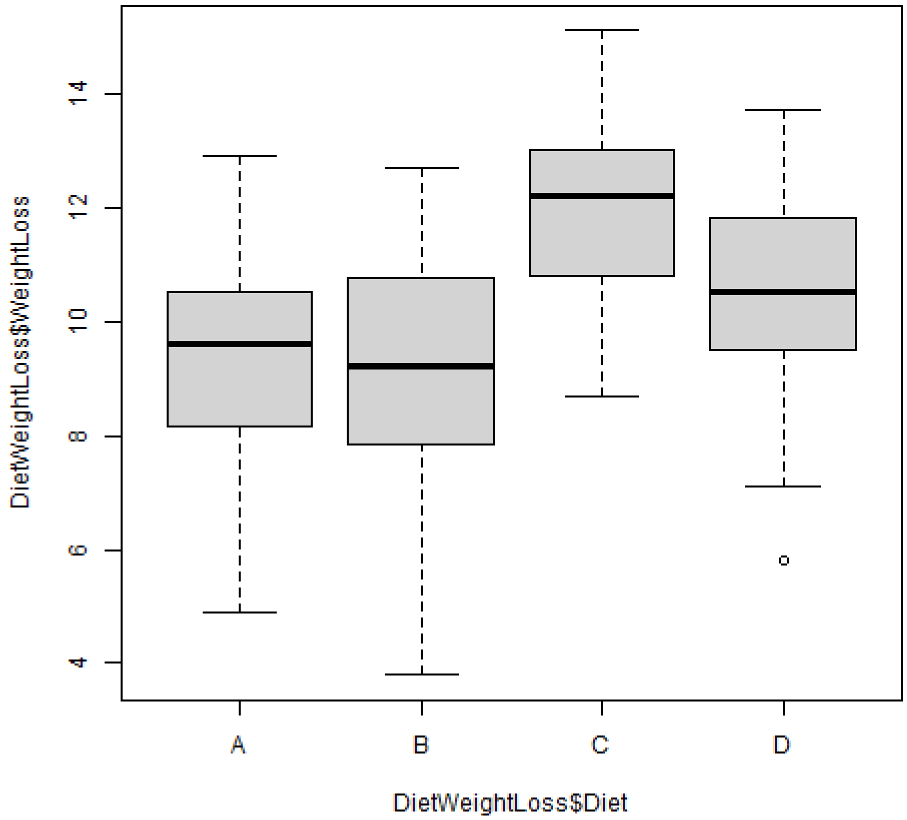
Program:

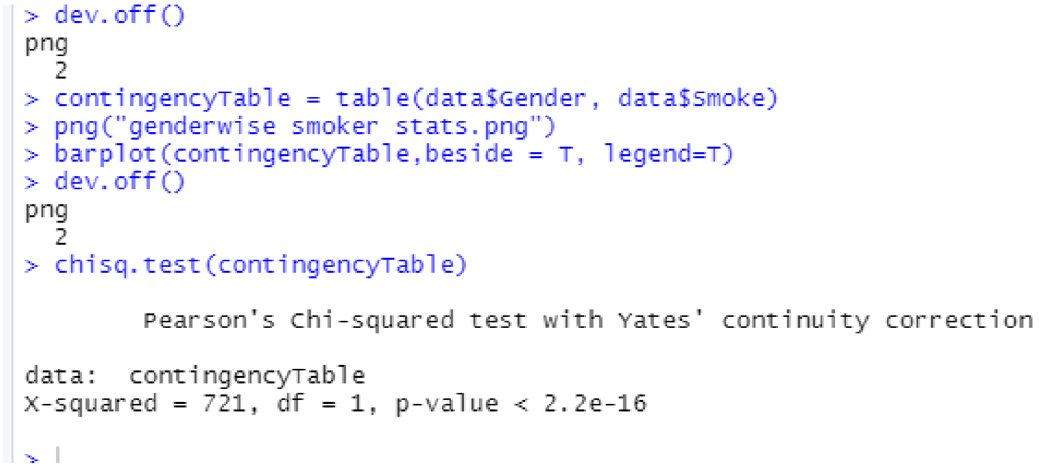
#chi square test

data$Smoke = as.factor(data$Smoke)

data$Smoke = as.factor(data$Gender)

```
str(data)

contingencyTable = table(data$Gender, data$Smoke)
png("genderwise smoker stats.png")
barplot(contingencyTable,beside = T, legend=T)
dev.off()

chisq.test(contingencyTable)
```

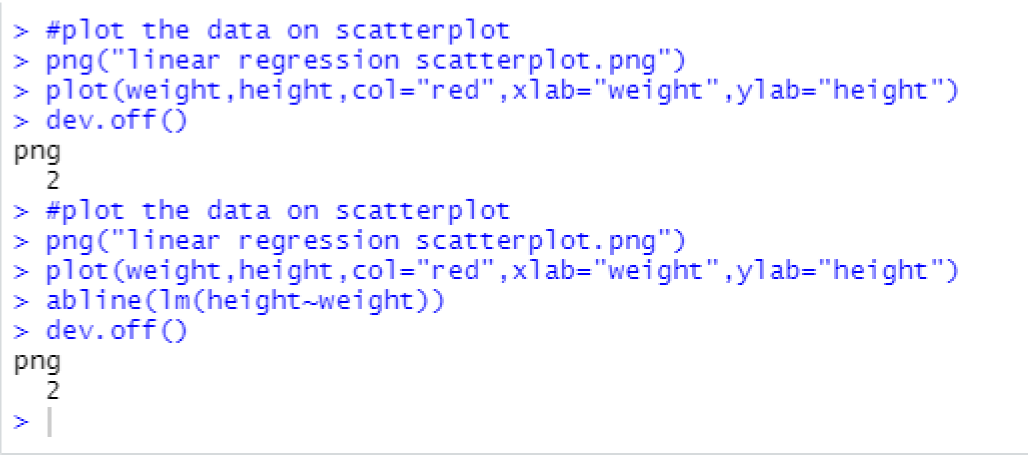




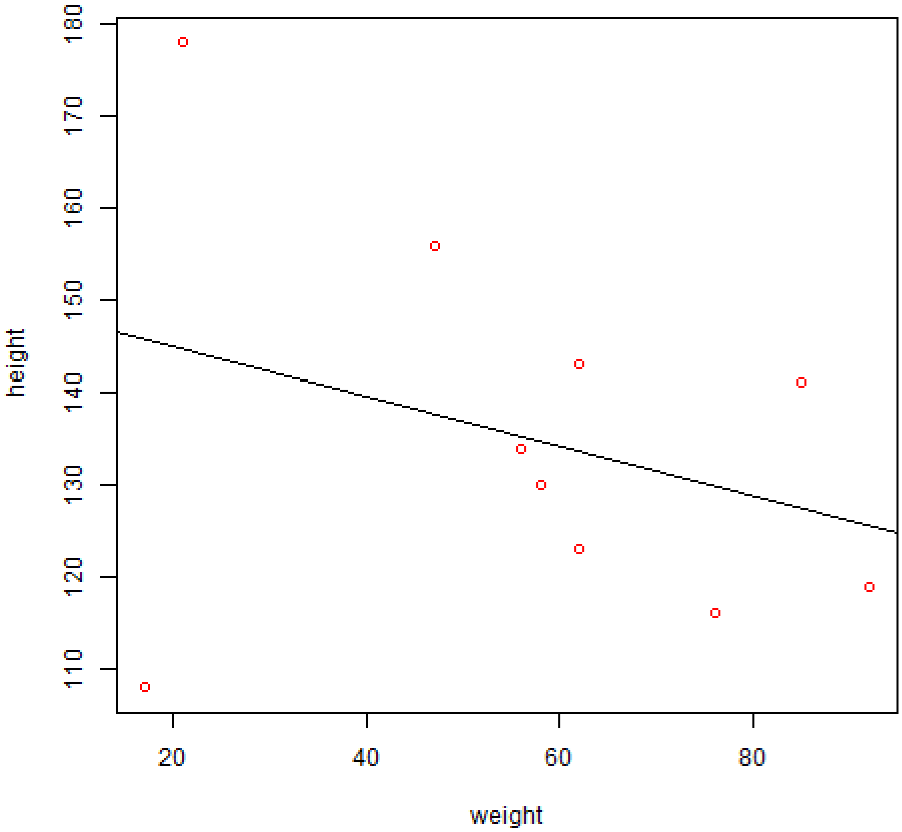
```
> dev.off()
png 
  2 
> contingencyTable = table(data$Gender, data$Smoke)
> png("genderwise smoker stats.png")
> barplot(contingencyTable,beside = T, legend=T)
> dev.off()
png 
  2 
> chisq.test(contingencyTable)

	Pearson's Chi-squared test with Yates' continuity correction

data:  contingencyTable
X-squared = 721, df = 1, p-value < 2.2e-16

> 
```

Regression & Linear Regression

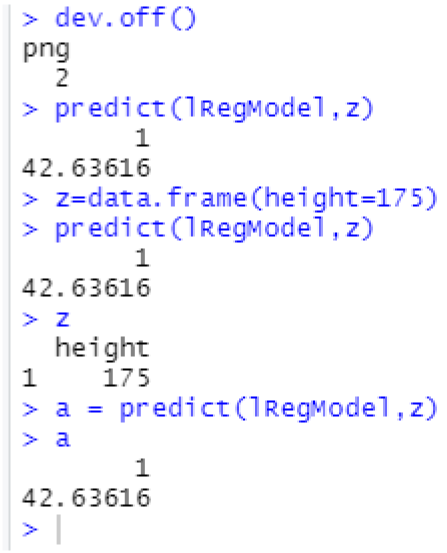
```
#regression models in R
#linear regression

#height in cm
height = c(123,141,134,178,156,108,116,119,143,130) #(x)

#weight in kg
weight = c(62,85,56,21,47,17,76,92,62,58) #(y)

lRegModel = lm(weight~height)
lRegModel

#plot the data on scatterplot
png("linear regression scatterplot.png")
plot(weight,height,col="red",xlab="weight",ylab="height")
abline(lm(height~weight))
dev.off()
```



```
> #plot the data on scatterplot
> png("linear regression scatterplot.png")
> plot(weight,height,col="red",xlab="weight",ylab="height")
> dev.off()
png
  2
> #plot the data on scatterplot
> png("linear regression scatterplot.png")
> plot(weight,height,col="red",xlab="weight",ylab="height")
> abline(lm(height~weight))
> dev.off()
png
  2
> |
```



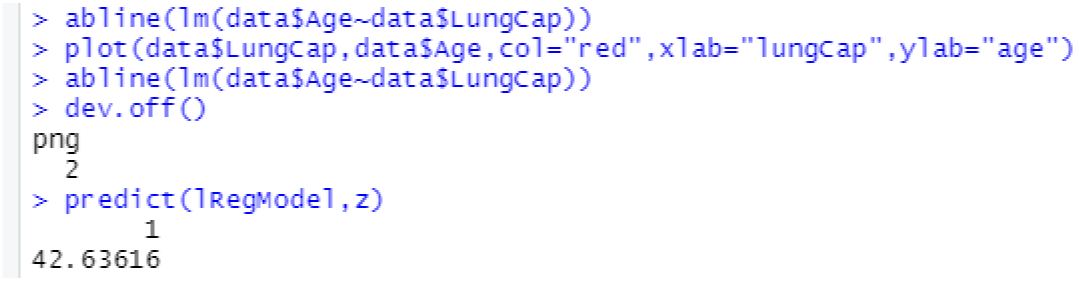
z=data.frame(height=175)
predict(lRegModel,z)



```
> dev.off()
png
  2
> predict(lRegModel,z)
         1
42.63616
> z=data.frame(height=175)
> predict(lRegModel,z)
         1
42.63616
> z
  height
1    175
> a = predict(lRegModel,z)
> a
         1
42.63616
> 
```

Linear regression

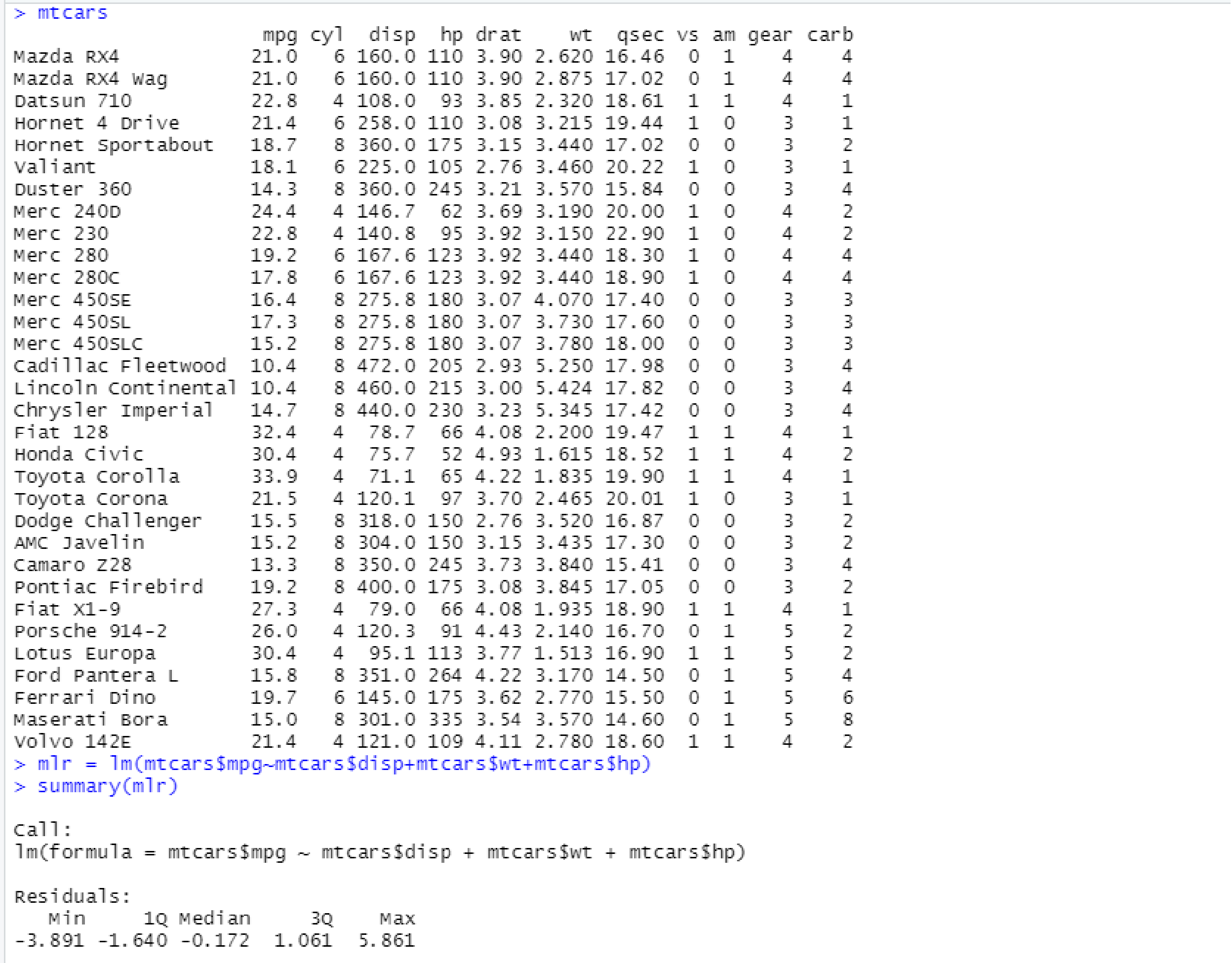
```
data
png("LungCapacityLM.png")
plot(data$LungCap,data$Age,col="red",xlab="lungCap",ylab="age")
abline(lm(data$Age~data$LungCap))
dev.off()
predict(lRegModel,z)
```



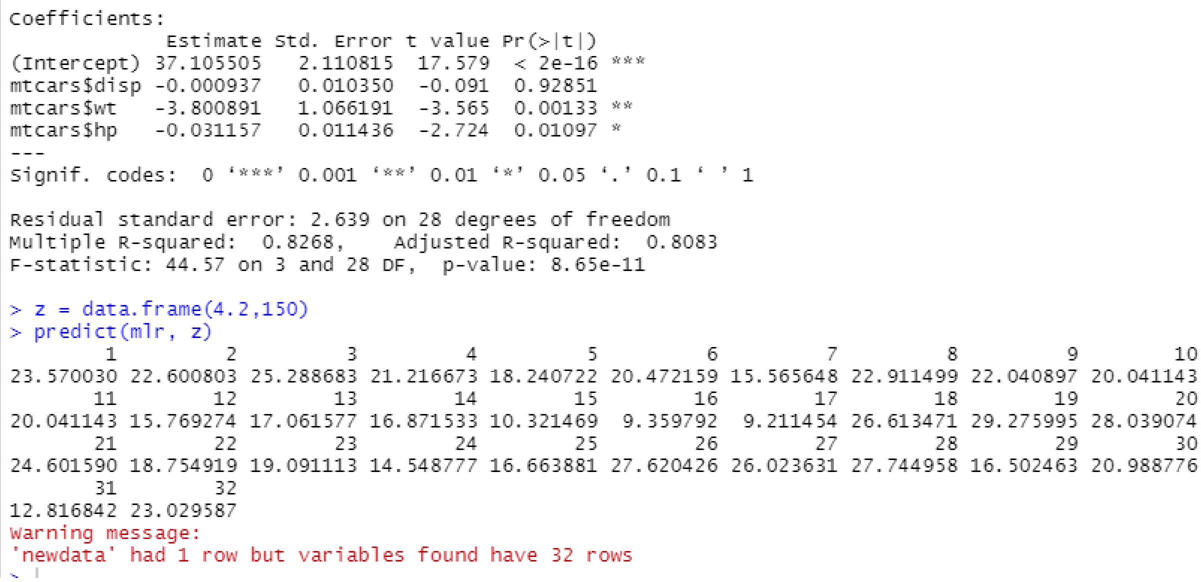
```
> abline(lm(data$Age~data$LungCap))
> plot(data$LungCap,data$Age,col="red",xlab="lungCap",ylab="age")
> abline(lm(data$Age~data$LungCap))
> dev.off()
png
  2
> predict(lRegModel,z)
         1
42.63616
```

Program:

```
#multiple linear regression
attach(mtcars)
str(mtcars)
summary(mtcars)
mlr = lm(mtcars$mpg~mtcars$disp+mtcars$wt+mtcars$hp)
summary(mlr)
z = data.frame(4.2,150)
predict(mlr, z)
```

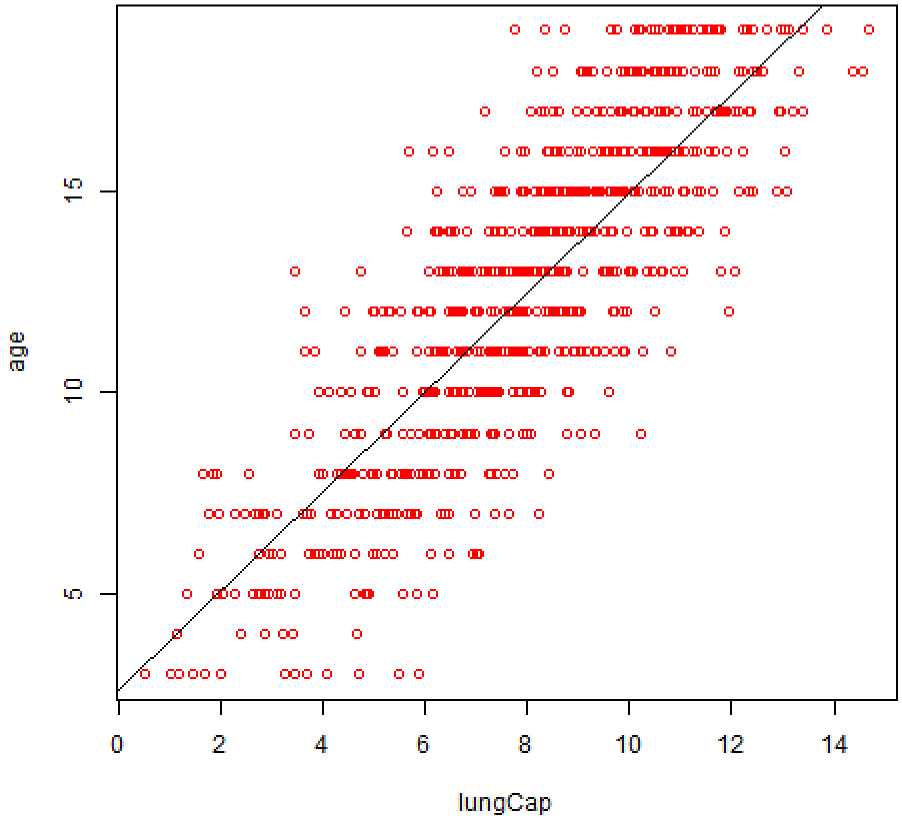


```
> mtcars
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
> mlr = lm(mtcars$mpg~mtcars$disp+mtcars$wt+mtcars$hp)
> summary(mlr)

Call:
lm(formula = mtcars$mpg ~ mtcars$disp + mtcars$wt + mtcars$hp)

Residuals:
   Min     1Q Median     3Q    Max 
-3.891 -1.640 -0.172  1.061  5.861 
```



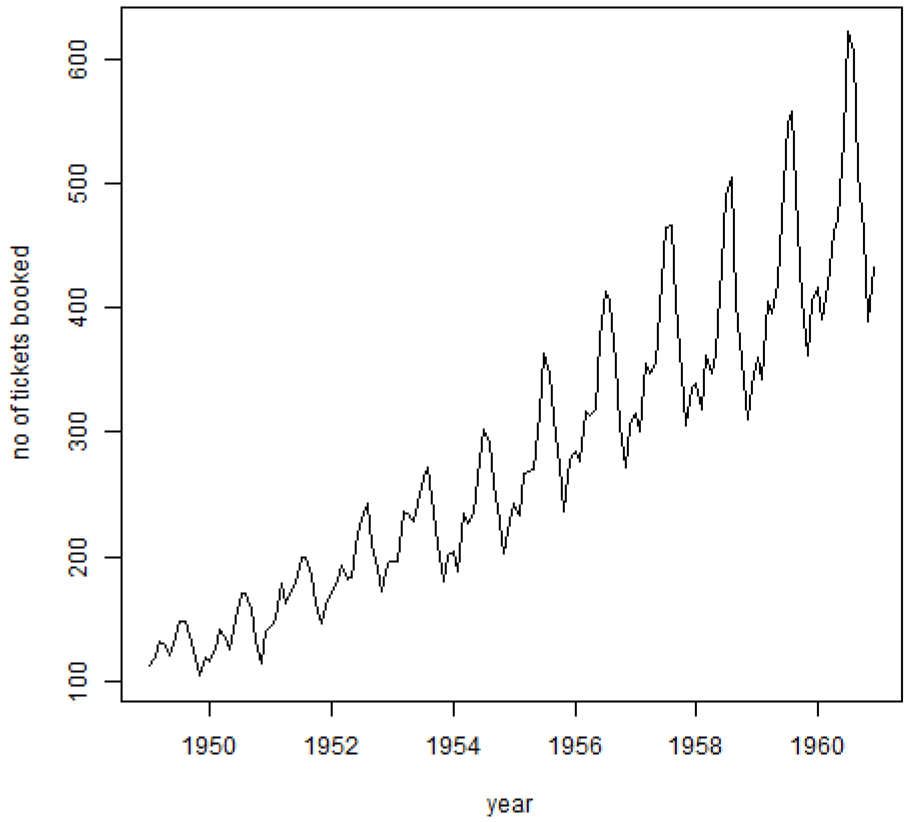
```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.105505   2.110815  17.579  < 2e-16 ***
mtcars$disp -0.000937   0.010350  -0.091  0.92851    
mtcars$wt   -3.800891   1.066191  -3.565  0.00133 ** 
mtcars$hp   -0.031157   0.011436  -2.724  0.01097 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.639 on 28 degrees of freedom
Multiple R-squared:  0.8268,	Adjusted R-squared:  0.8083 
F-statistic: 44.57 on 3 and 28 DF,  p-value: 8.65e-11

> z = data.frame(4.2,150)
> predict(mlr, z)
        1         2         3         4         5         6         7         8         9        10 
23.570030 22.600803 25.288683 21.216673 18.240722 20.472159 15.565648 22.911499 22.040897 20.041143 
       11        12        13        14        15        16        17        18        19        20 
20.041143 15.769274 17.061577 16.871533 10.321469  9.359792  9.211454 26.613471 29.275995 28.039074 
       21        22        23        24        25        26        27        28        29        30 
24.601590 18.754919 19.091113 14.548777 16.663881 27.620426 26.023631 27.744958 16.502463 20.988776 
       31        32 
12.816842 23.029587 
Warning message:
'newdata' had 1 row but variables found have 32 rows 
> 
```

Program:



Time series

```
#time series analysis
str(AirPassengers)

AP = AirPassengers
str(AP)

#freq = 12 because data collected for every month for every year
#starting from 1949 jan

ts(AP,frequency = 12,start=c(1949,1))
png("timeseriesPlot.png")
plot(AP,ylab="no of tickets booked", xlab="year")
dev.off()
```



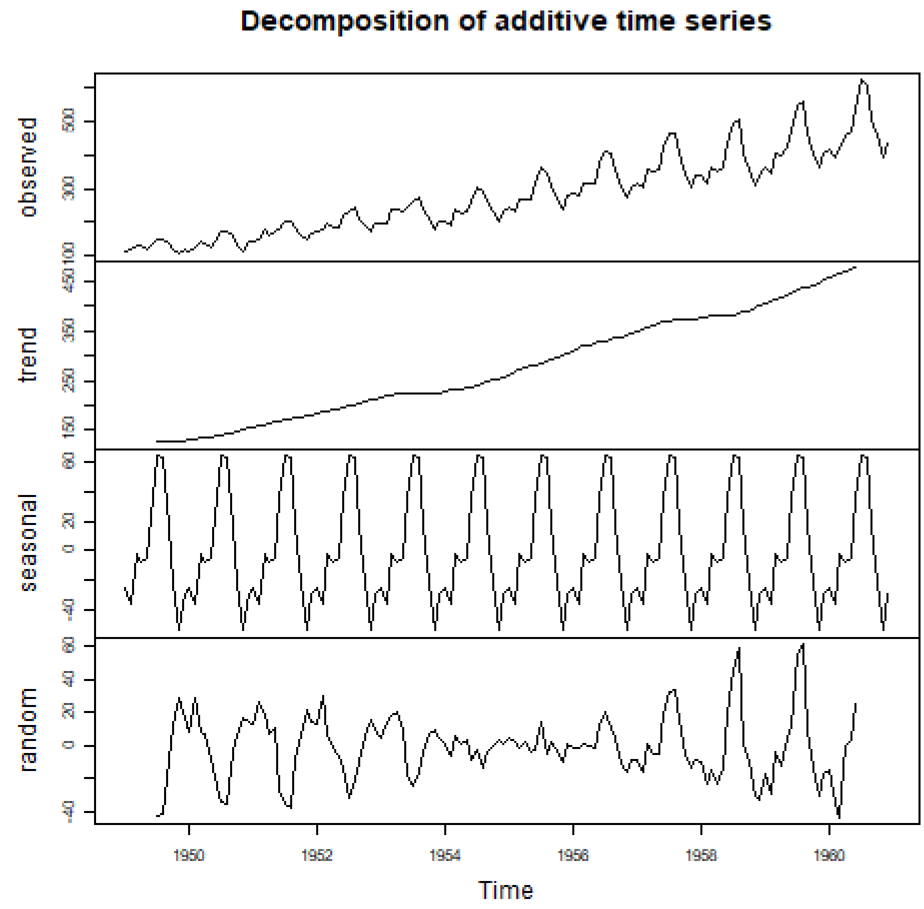
```
#decomposition of data or log transformation
decompAP = log(AP)
png("timeseriesLoggedPvalues.png")
plot(AP,ylab="no of tickets booked", xlab="year")
dev.off()
```

Program:

```
str(AirPassengers)

AP = AirPassengers
str(AP)

decompA = decompose(AP)
attributes(decompA)

png("Season&TrendInAPValues.jpeg")
plot(decompA$figure,ylab="Seasonality",xlab="Months",type='b',col='blue')
plot(decompA)
```

```
dev.off()
```

Output:



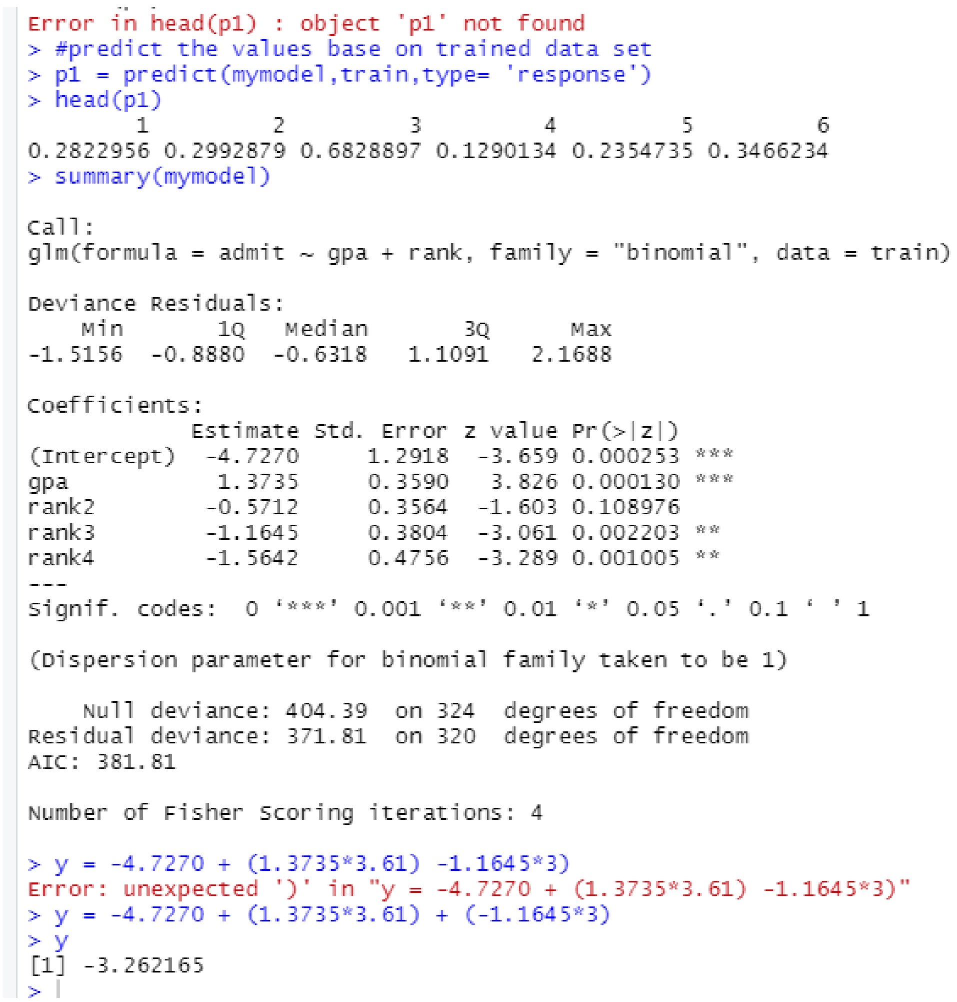
Logistic regression

```
#logistic regression

mydata = binary
mydata$admit = as.factor(mydata$admit)
mydata$rank = as.factor(mydata$rank)

xtabs(~mydata$admit+mydata$rank)

#partition the data into train and test datasets
set.seed(1234)
ind = sample(2,nrow(mydata),replace =T, prob = c(0.8,0.2))
ind
```

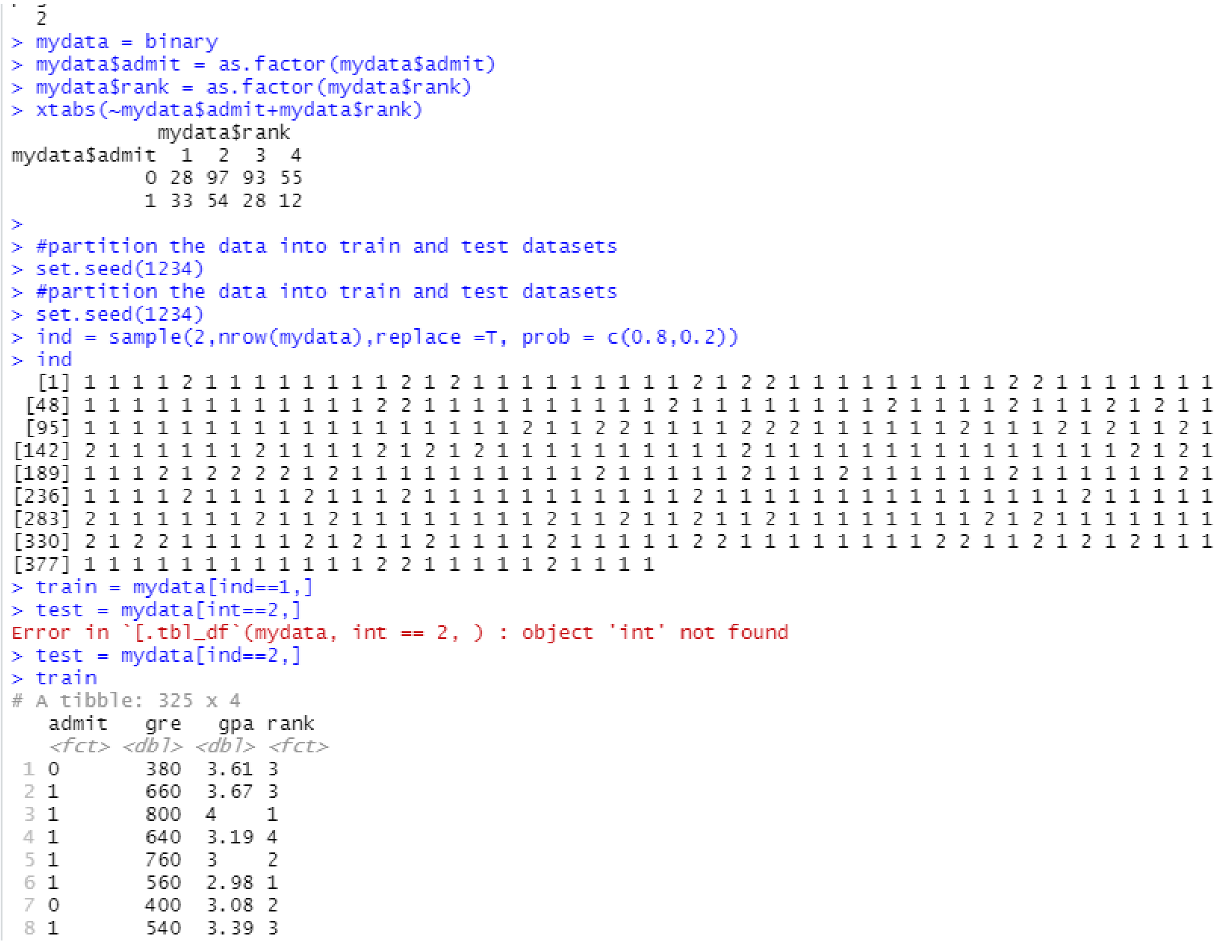
```
#train data model
train = mydata[ind==1,]

#test data model
test = mydata[ind==2,]
train
test

#logistic regression model
mymodel = glm(admit~gpa+gre+rank, data = train, family = "binomial")
summary(mymodel)

mymodel = glm(admit~gpa+rank, data = train, family = "binomial")
summary(mymodel)

#predict the values base on trained data set
p1 = predict(mymodel,train,type= 'response')
head(p1)
```

Testing

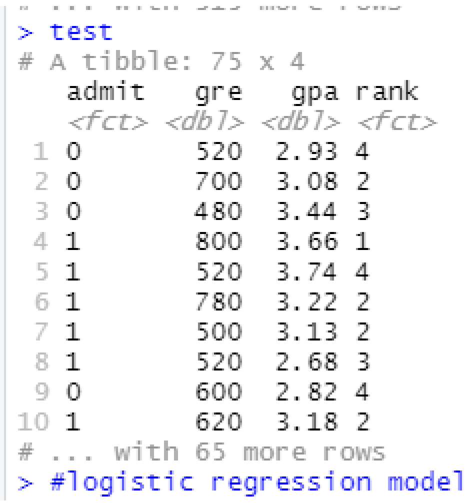
```
mymodel = glm(admit~gpa+rank, data = test, family = "binomial")
summary(mymodel)
#predict the values base on trained data set
p2 = predict(mymodel,test,type= 'response')
head(p2)
```



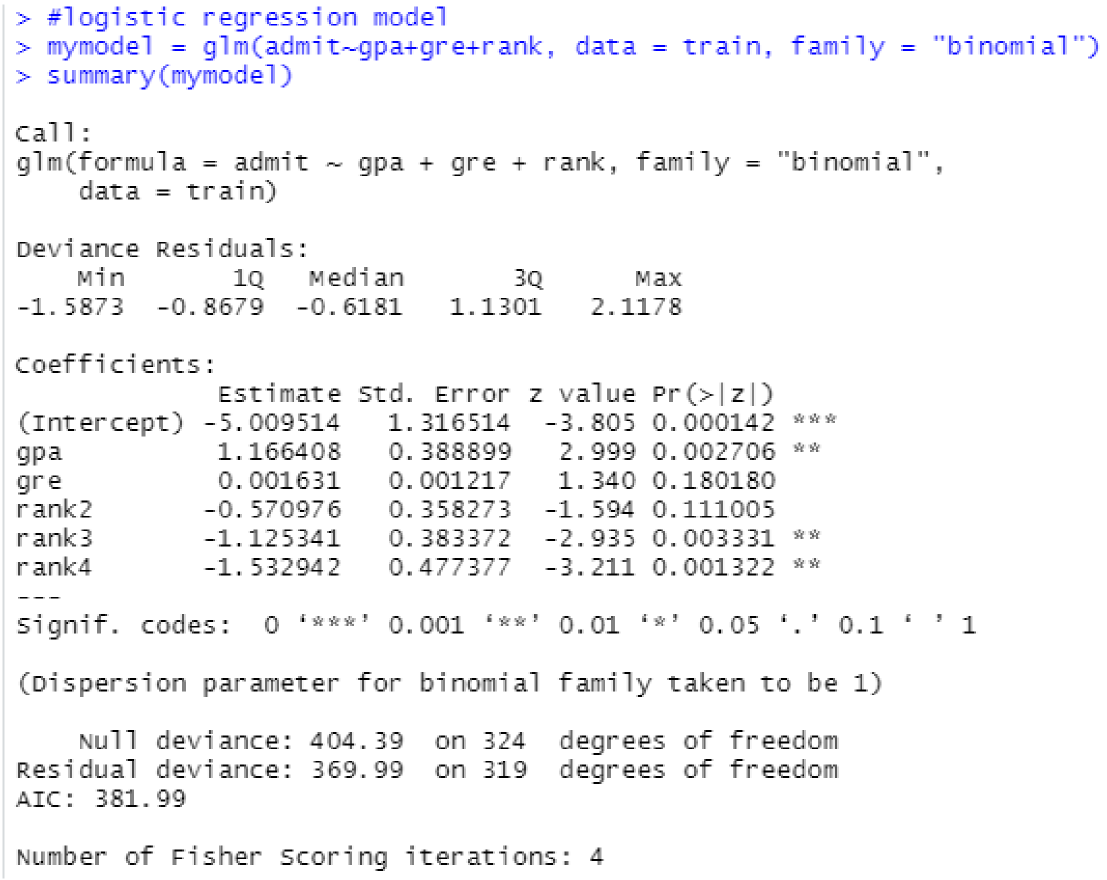
```
Error in head(p1) : object 'p1' not found
> #predict the values base on trained data set
> p1 = predict(mymodel,train,type= 'response')
> head(p1)
        1         2         3         4         5         6 
0.2822956 0.2992879 0.6828897 0.1290134 0.2354735 0.3466234 
> summary(mymodel)

Call:
glm(formula = admit ~ gpa + rank, family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.5156  -0.8880  -0.6318   1.1091   2.1688  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  -4.7270     1.2918  -3.659 0.000253 ***
gpa           1.3735     0.3590   3.826 0.000130 ***
rank2        -0.5712     0.3564  -1.603 0.108976    
rank3        -1.1645     0.3804  -3.061 0.002203 ** 
rank4        -1.5642     0.4756  -3.289 0.001005 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.39  on 324  degrees of freedom
Residual deviance: 371.81  on 320  degrees of freedom
AIC: 381.81

Number of Fisher Scoring iterations: 4

> y = -4.7270 + (1.3735*3.61) -1.1645*3)
Error: unexpected ')' in "y = -4.7270 + (1.3735*3.61) -1.1645*3)"
> y = -4.7270 + (1.3735*3.61) + (-1.1645*3)
> y
[1] -3.262165
> 
```



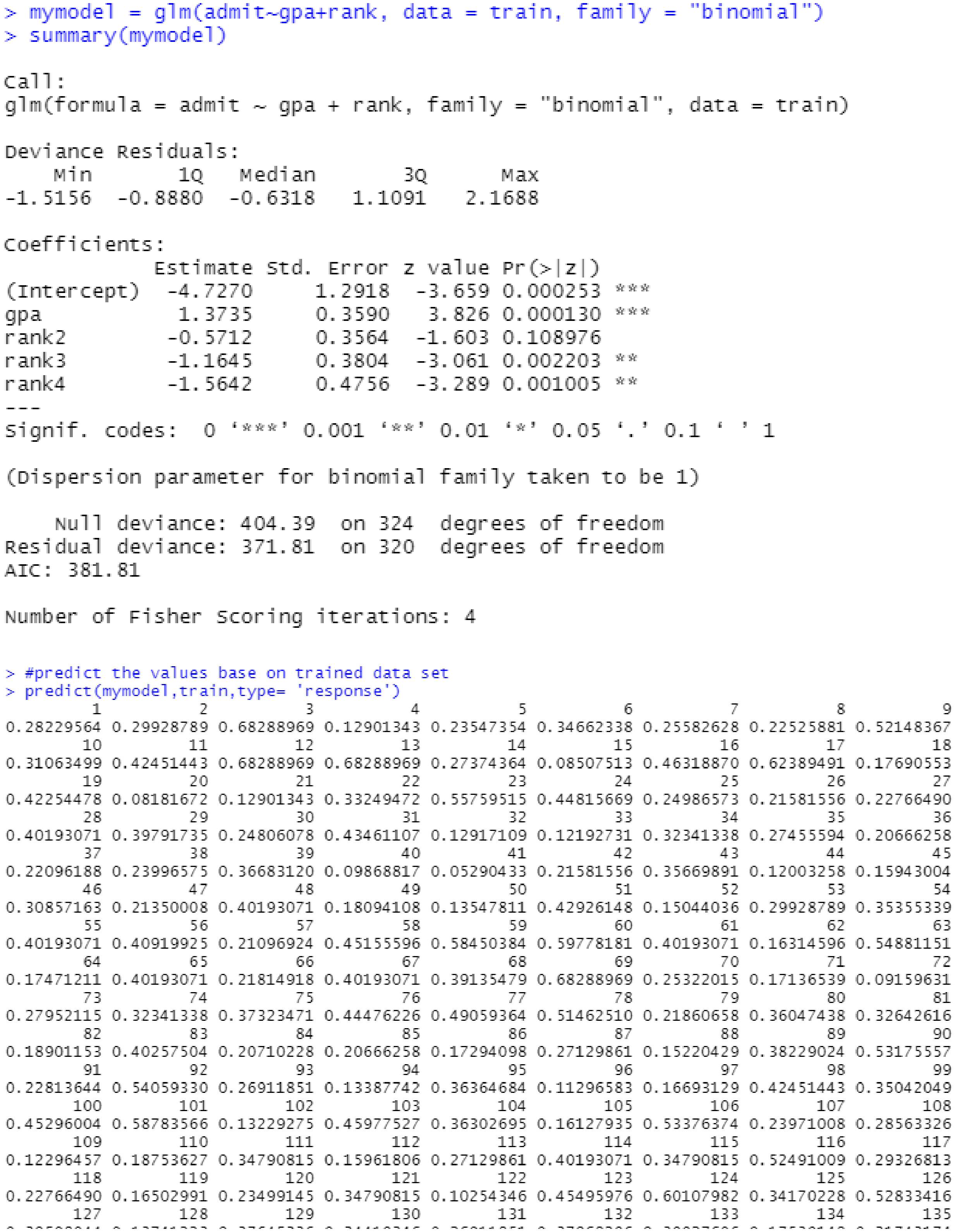
```
  2
> mydata = binary
> mydata$admit = as.factor(mydata$admit)
> mydata$rank = as.factor(mydata$rank)
> xtabs(~mydata$admit+mydata$rank)
            mydata$rank
mydata$admit  1  2  3  4
           0 28 97 93 55
           1 33 54 28 12
>
> #partition the data into train and test datasets
> set.seed(1234)
> #partition the data into train and test datasets
> set.seed(1234)
> ind = sample(2,nrow(mydata),replace =T, prob = c(0.8,0.2))
> ind
  [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1
 [48] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 2 1 1
 [95] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 1 2 1 2 1 1 2 1
[142] 2 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1
[189] 1 1 1 2 1 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1
[236] 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
[283] 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1
[330] 2 1 2 2 1 1 1 1 1 2 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 1 1
[377] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1
> train = mydata[ind==1,]
> test = mydata[int==2,]
Error in `[.tbl_df`(mydata, int == 2, ) : object 'int' not found
> test = mydata[ind==2,]
> train
# A tibble: 325 x 4
   admit   gre   gpa rank
   <fct> <dbl> <dbl> <fct>
 1 0       380  3.61 3
 2 1       660  3.67 3
 3 1       800  4    1
 4 1       640  3.19 4
 5 1       760  3    2
 6 1       560  2.98 1
 7 0       400  3.08 2
 8 1       540  3.39 3
```



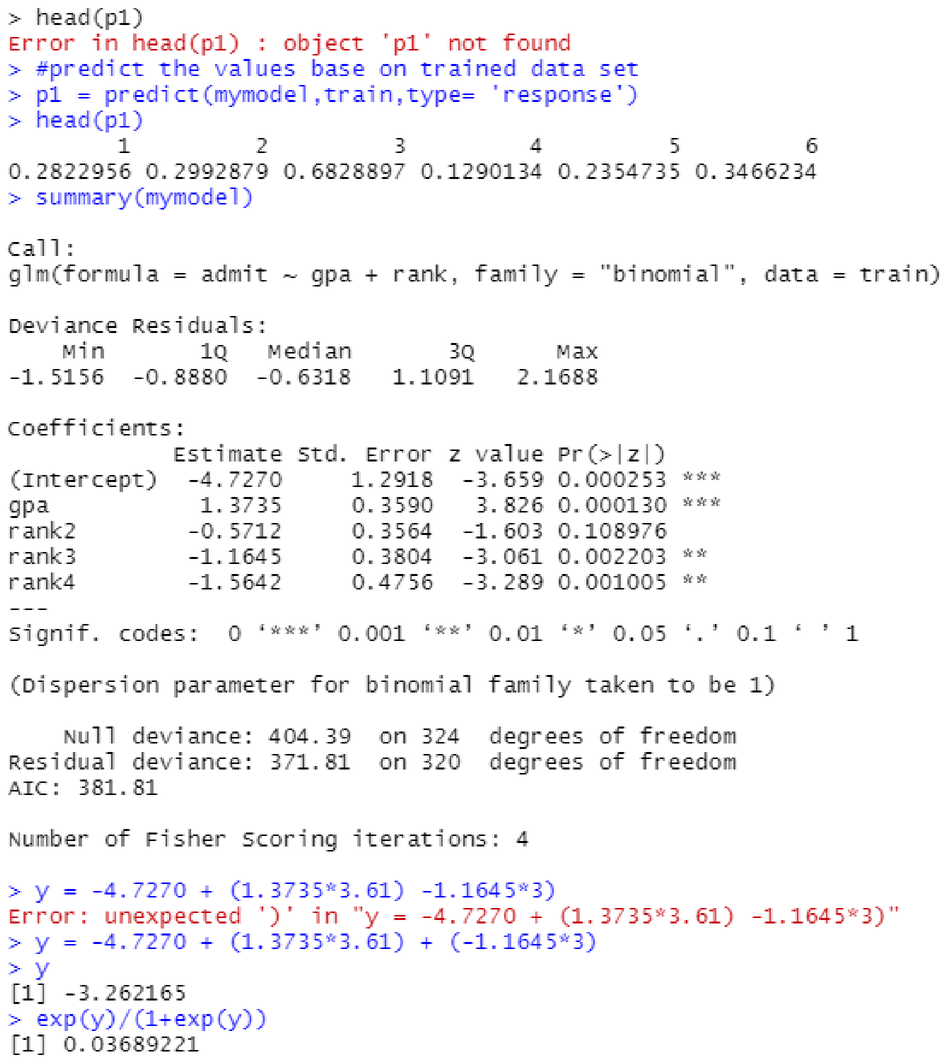
```
# ... with 315 more rows
> test
# A tibble: 75 x 4
   admit   gre   gpa rank
   <fct> <dbl> <dbl> <fct>
 1 0       520  2.93 4
 2 0       700  3.08 2
 3 0       480  3.44 3
 4 1       800  3.66 1
 5 1       520  3.74 4
 6 1       780  3.22 2
 7 1       500  3.13 2
 8 1       520  2.68 3
 9 0       600  2.82 4
10 1       620  3.18 2
# ... with 65 more rows
> #logistic regression model
```



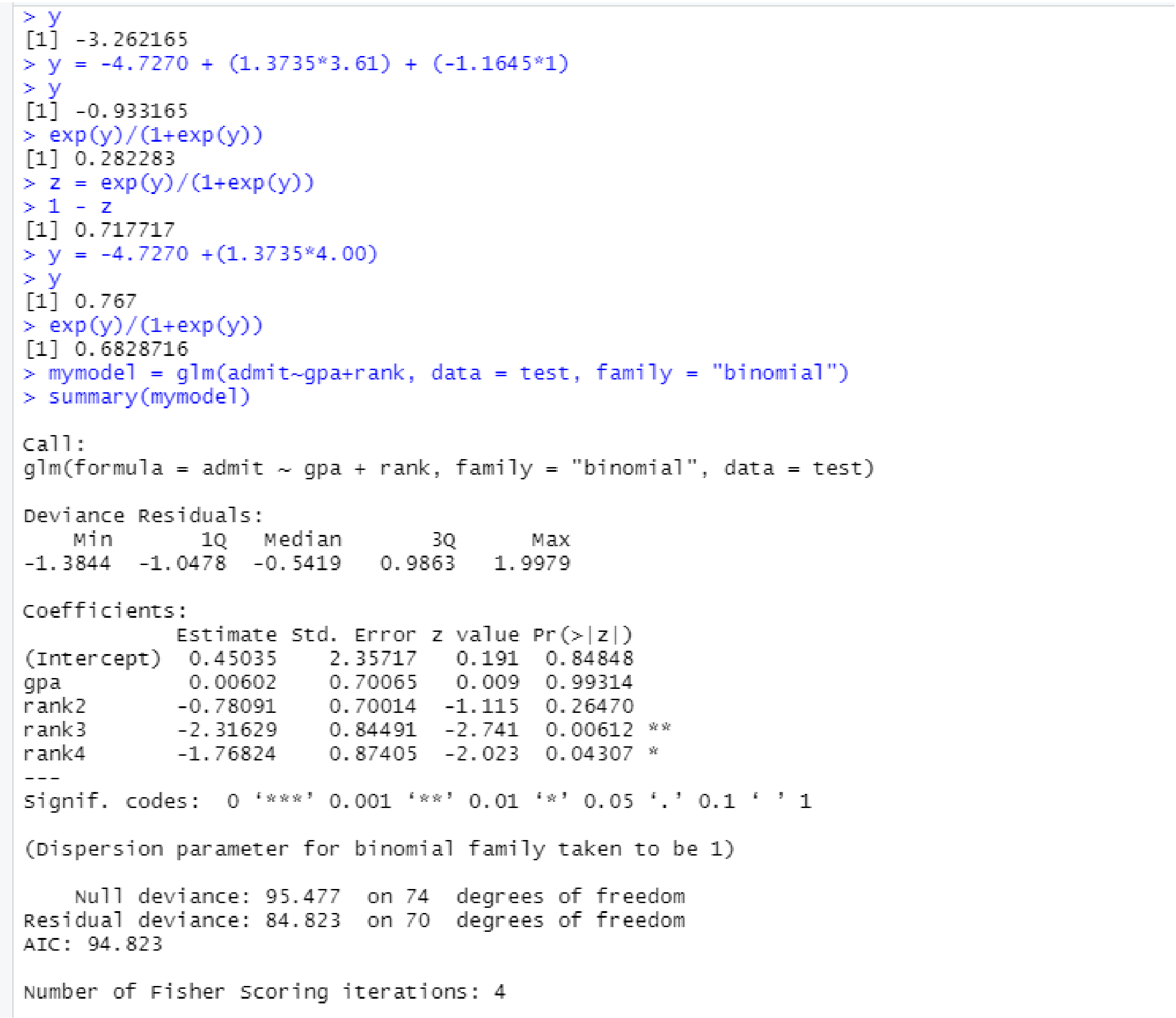
```
> #logistic regression model
> mymodel = glm(admit~gpa+gre+rank, data = train, family = "binomial")
> summary(mymodel)

Call:
glm(formula = admit ~ gpa + gre + rank, family = "binomial",
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5873  -0.8679  -0.6181   1.1301   2.1178

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.009514   1.316514  -3.805 0.000142 ***
gpa          1.166408   0.388899   2.999 0.002706 **
gre          0.001631   0.001217   1.340 0.180180
rank2       -0.570976   0.358273  -1.594 0.111005
rank3       -1.125341   0.383372  -2.935 0.003331 **
rank4       -1.532942   0.477377  -3.211 0.001322 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.39  on 324  degrees of freedom
Residual deviance: 369.99  on 319  degrees of freedom
AIC: 381.99

Number of Fisher Scoring iterations: 4
```



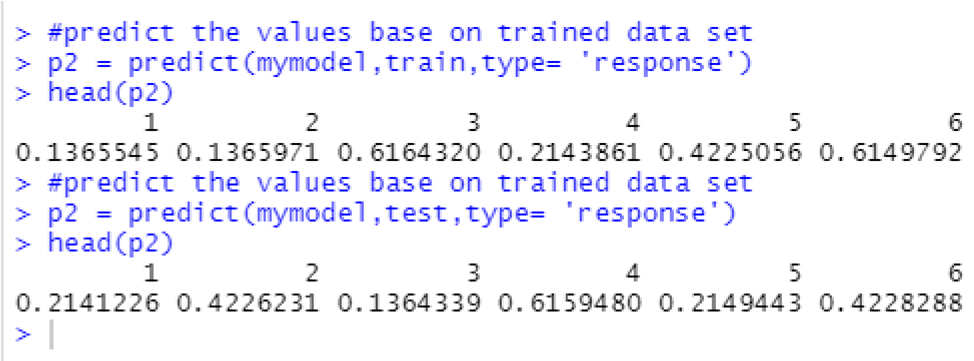
```
> mymodel = glm(admit~gpa+rank, data = train, family = "binomial")
> summary(mymodel)

Call:
glm(formula = admit ~ gpa + rank, family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.5156  -0.8880  -0.6318   1.1091   2.1688  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  -4.7270     1.2918  -3.659 0.000253 ***
gpa           1.3735     0.3590   3.826 0.000130 ***
rank2        -0.5712     0.3564  -1.603 0.108976    
rank3        -1.1645     0.3804  -3.061 0.002203 ** 
rank4        -1.5642     0.4756  -3.289 0.001005 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.39  on 324  degrees of freedom
Residual deviance: 371.81  on 320  degrees of freedom
AIC: 381.81

Number of Fisher Scoring iterations: 4


> #predict the values base on trained data set
> predict(mymodel,train,type= 'response')
         1          2          3          4          5          6          7          8          9 
0.28229564 0.29928789 0.68288969 0.12901343 0.23547354 0.34662338 0.25582628 0.22525881 0.52148367 
        10         11         12         13         14         15         16         17         18 
0.31063499 0.42451443 0.68288969 0.68288969 0.27374364 0.08507513 0.46318870 0.62389491 0.17690553 
        19         20         21         22         23         24         25         26         27 
0.42254478 0.08181672 0.12901343 0.33249472 0.55759515 0.44815669 0.24986573 0.21581556 0.22766490 
        28         29         30         31         32         33         34         35         36 
0.40193071 0.39791735 0.24806078 0.43461107 0.12917109 0.12192731 0.32341338 0.27455594 0.20666258 
        37         38         39         40         41         42         43         44         45 
0.22096188 0.23996575 0.36683120 0.09868817 0.05290433 0.21581556 0.35669891 0.12003258 0.15943004 
        46         47         48         49         50         51         52         53         54 
0.30857163 0.21350008 0.40193071 0.18094108 0.13547811 0.42926148 0.15044036 0.29928789 0.35355339 
        55         56         57         58         59         60         61         62         63 
0.40193071 0.40919925 0.21096924 0.45155596 0.58450384 0.59778181 0.40193071 0.16314596 0.54881151 
        64         65         66         67         68         69         70         71         72 
0.17471211 0.40193071 0.21814918 0.40193071 0.39135479 0.68288969 0.25322015 0.17136539 0.09159631 
        73         74         75         76         77         78         79         80         81 
0.27952115 0.32341338 0.37323471 0.44476226 0.49059364 0.51462510 0.21860658 0.36047438 0.32642616 
        82         83         84         85         86         87         88         89         90 
0.18901153 0.40257504 0.20710228 0.20666258 0.17294098 0.27129861 0.15220429 0.38229024 0.53175557 
        91         92         93         94         95         96         97         98         99 
0.22813644 0.54059330 0.26911851 0.13387742 0.36364684 0.11296583 0.16693129 0.42451443 0.35042049 
       100        101        102        103        104        105        106        107        108 
0.45296004 0.58783566 0.13229275 0.45977527 0.36302695 0.16127935 0.53376374 0.23971008 0.28563326 
       109        110        111        112        113        114        115        116        117 
0.12296457 0.18753627 0.34790815 0.15961806 0.27129861 0.40193071 0.34790815 0.52491009 0.29326813 
       118        119        120        121        122        123        124        125        126 
0.22766490 0.16502991 0.23499145 0.34790815 0.10254346 0.45495976 0.60107982 0.34170228 0.52833416 
       127        128        129        130        131        132        133        134        135 
```



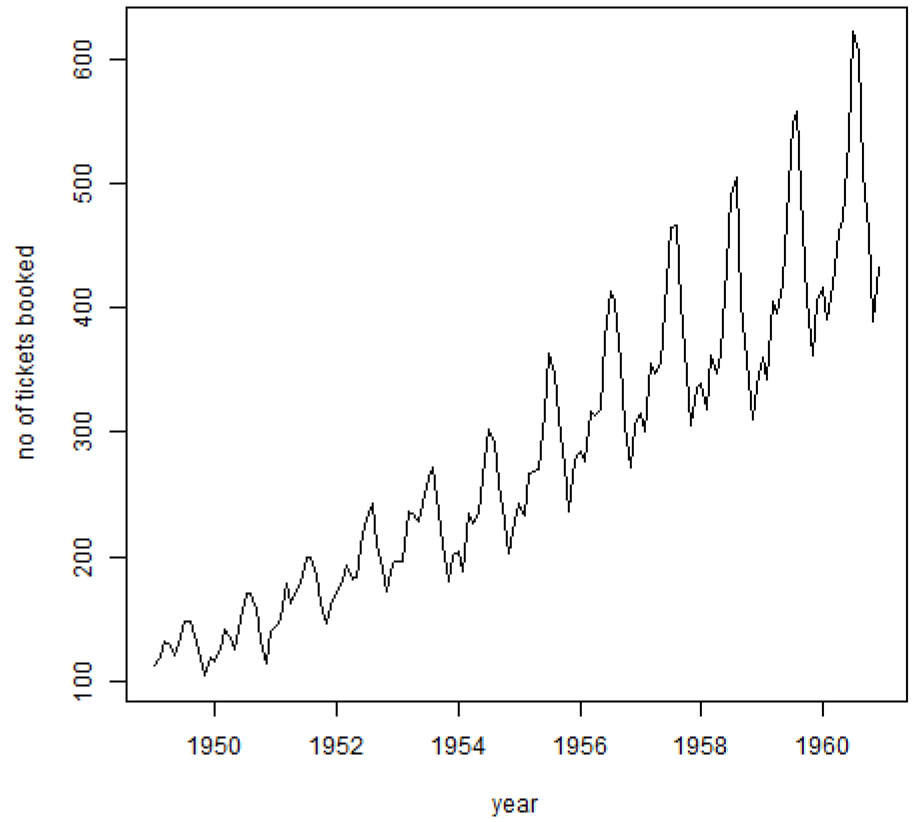
```
> head(p1)
Error in head(p1) : object 'p1' not found
> #predict the values base on trained data set
> p1 = predict(mymodel,train,type= 'response')
> head(p1)
        1         2         3         4         5         6 
0.2822956 0.2992879 0.6828897 0.1290134 0.2354735 0.3466234 
> summary(mymodel)

Call:
glm(formula = admit ~ gpa + rank, family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.5156  -0.8880  -0.6318   1.1091   2.1688  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  -4.7270     1.2918  -3.659 0.000253 ***
gpa           1.3735     0.3590   3.826 0.000130 ***
rank2        -0.5712     0.3564  -1.603 0.108976    
rank3        -1.1645     0.3804  -3.061 0.002203 ** 
rank4        -1.5642     0.4756  -3.289 0.001005 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.39  on 324  degrees of freedom
Residual deviance: 371.81  on 320  degrees of freedom
AIC: 381.81

Number of Fisher Scoring iterations: 4

> y = -4.7270 + (1.3735*3.61) -1.1645*3)
Error: unexpected ')' in "y = -4.7270 + (1.3735*3.61) -1.1645*3)"
> y = -4.7270 + (1.3735*3.61) + (-1.1645*3)
> y
[1] -3.262165
> exp(y)/(1+exp(y))
[1] 0.03689221
```



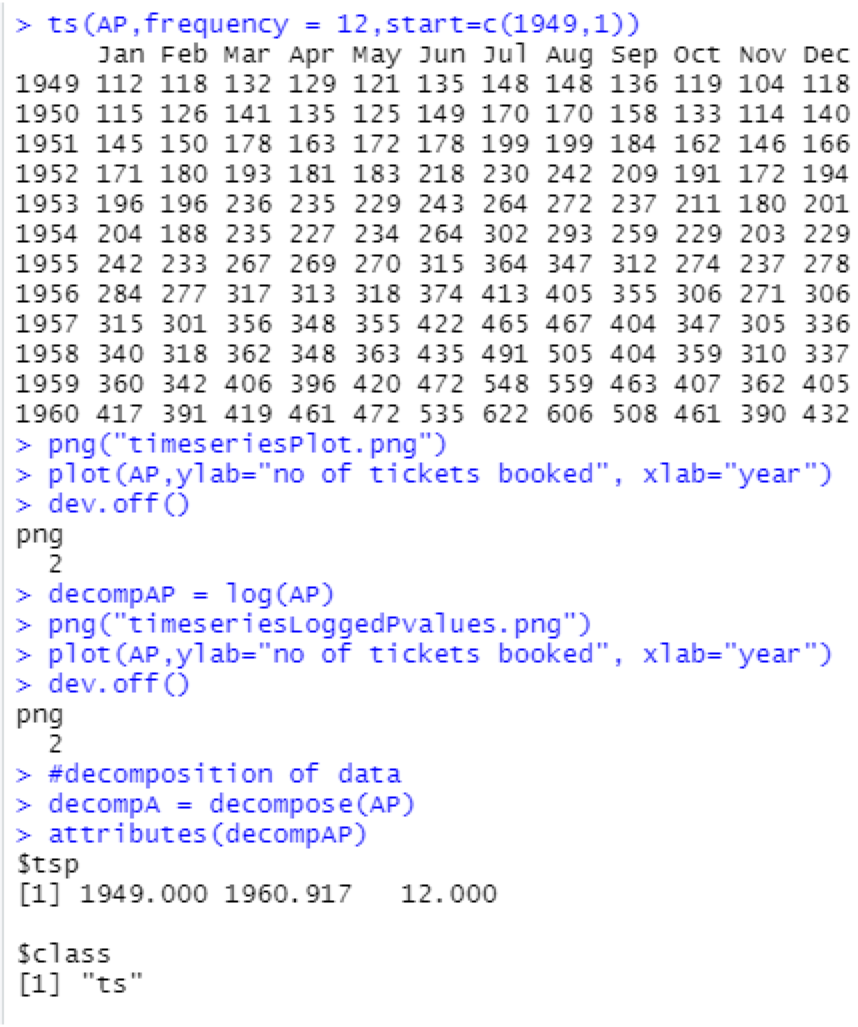
```
> y
[1] -3.262165
> y = -4.7270 + (1.3735*3.61) + (-1.1645*1)
> y
[1] -0.933165
> exp(y)/(1+exp(y))
[1] 0.282283
> z = exp(y)/(1+exp(y))
> 1 - z
[1] 0.717717
> y = -4.7270 +(1.3735*4.00)
> y
[1] 0.767
> exp(y)/(1+exp(y))
[1] 0.6828716
> mymodel = glm(admit~gpa+rank, data = test, family = "binomial")
> summary(mymodel)

Call:
glm(formula = admit ~ gpa + rank, family = "binomial", data = test)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.3844  -1.0478  -0.5419   0.9863   1.9979  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)   
(Intercept)  0.45035    2.35717   0.191  0.84848   
gpa          0.00602    0.70065   0.009  0.99314   
rank2       -0.78091    0.70014  -1.115  0.26470   
rank3       -2.31629    0.84491  -2.741  0.00612 **
rank4       -1.76824    0.87405  -2.023  0.04307 * 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 95.477  on 74  degrees of freedom
Residual deviance: 84.823  on 70  degrees of freedom
AIC: 94.823

Number of Fisher Scoring iterations: 4
```



```
> #predict the values base on trained data set
> p2 = predict(mymodel,train,type= 'response')
> head(p2)
        1         2         3         4         5         6 
0.1365545 0.1365971 0.6164320 0.2143861 0.4225056 0.6149792 
> #predict the values base on trained data set
> p2 = predict(mymodel,test,type= 'response')
> head(p2)
        1         2         3         4         5         6 
0.2141226 0.4226231 0.1364339 0.6159480 0.2149443 0.4228288 
> 
```



```
#decomposition of data
decompA = decompose(AP)
attributes(decompAP)
```

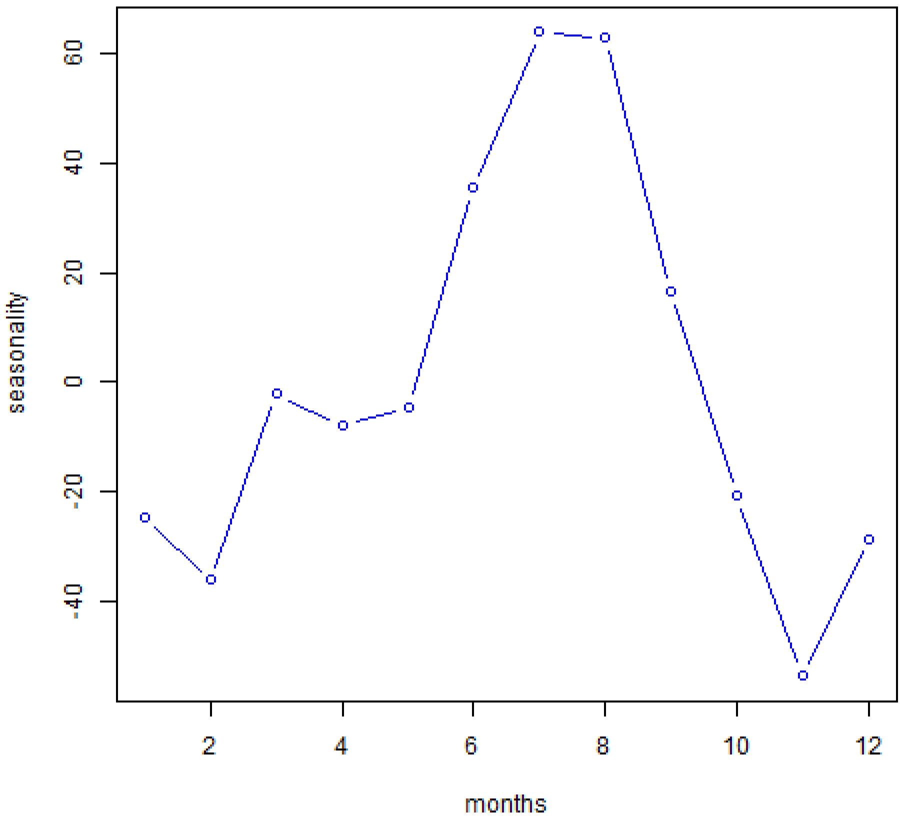


```
> ts(AP,frequency = 12,start=c(1949,1))
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
> png("timeseriesPlot.png")
> plot(AP,ylab="no of tickets booked", xlab="year")
> dev.off()
png 
  2 
> decompAP = log(AP)
> png("timeseriesLoggedPvalues.png")
> plot(AP,ylab="no of tickets booked", xlab="year")
> dev.off()
png 
  2 
> #decomposition of data
> decompA = decompose(AP)
> attributes(decompAP)
$tsp
[1] 1949.000 1960.917   12.000

$class
[1] "ts"
```

Program:

```
#decomposition of data
decompAP = decompose(AP)
attributes(decompAP)

png("Season&TrendInAPValues.png")
plot(decompAP$figure, ylab = "seasonality", xlab="months",type="b",col="blue")
dev.off()
```

Output:


seasonality
60
40
20
0
-20
-40
2
4
6
8
10
12
months