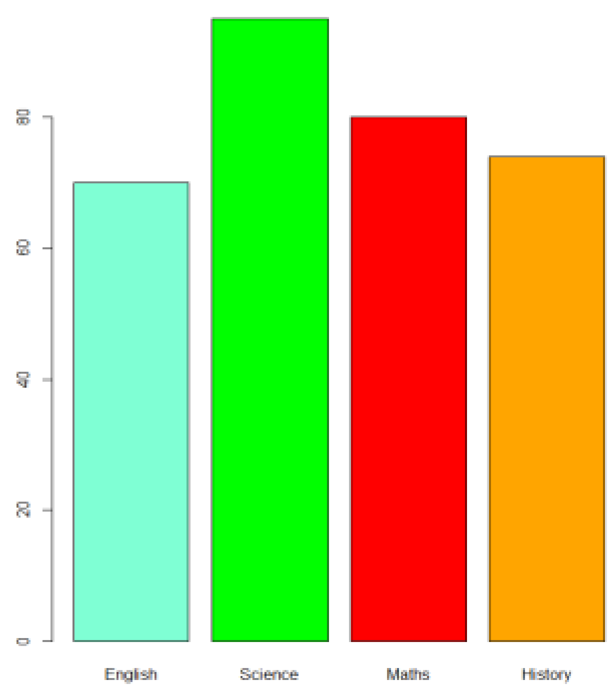
R Exam

> #Q1. Write an R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and the sum of numbers from 51 to 91.

```
> seq(20, 50)
 [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
50
> mean(20:60)
[1] 40
> mean(seq(20, 60))
[1] 40
> sum(51:91)
[1] 2911
> sum(seq(51, 91))
[1] 2911
>
```

> #Q2.A student scored 70 marks in English, 95 marks in Science, 80 marks in Maths and 74 marks in History. Write an R program to plot a simple bar chart displaying the scores of the given subjects.

```
> subjects <- c("English"=70, "Science"=95, "Maths"=80, "History"=74)
> barplot(subjects, col=c("aquamarine", "green", "red", "orange"))
```



>

> #Q3. Write a R program to create a data frame to store the following details of 5 employees.

```
> name <- c("Anastasia S", "Dima R", "Katherine S", "JAMES A", "LAURA MARTIN")
> gender <- c("M", "M", "F", "F", "M")
> age <- c(23, 22, 25, 26, 32)
> desig <- c("Clerk", "Manager", "Executive", "CEO", "ASSISTANT")
> ssn <- c("123-34-2346", "23-44-779", "556-24-433", "123-98-987", "679-77-576")
> employees <- data.frame(name, gender, age, desig, ssn)
> colnames(employees) <- c("Name", "Gender", "Age", "Designation", "SSN")
> employees
          Name Gender Age Designation         SSN
1  Anastasia S      M  23       Clerk 123-34-2346
2       Dima R      M  22     Manager   23-44-779
3  Katherine S      F  25   Executive  556-24-433
4      JAMES A      F  26         CEO  123-98-987
5 LAURA MARTIN      M  32   ASSISTANT  679-77-576
>
```

> #Q4. Write an R program to create a list of heterogeneous data, which includes character, numeric and logical vectors. Print the list.

```
> l <- list(c("Male", "Female"), c(24, 25), TRUE)
> l
[[1]]
[1] "Male"   "Female"

[[2]]
[1] 24 25

[[3]]
[1] TRUE


>
```

> #Q.5 Write an R program to convert a given matrix to a 1-dimensional array.

```
> mat <- matrix(1:12, ncol=4)
> mat
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> array(as.vector(mat))
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
>
```

> #Q.6 Write a R program to create a list containing a given vector, a matrix, and a list and add an element at the end of the list

```
> li <- list(c("Red", "Green", "Black"), matrix(seq(1, 11, 2), ncol=3), list("Python", "PHP", "Java"))
> li
[[1]]
[1] "Red"   "Green" "Black"

[[2]]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11

[[3]]
[[3]][[1]]
[1] "Python"

[[3]][[2]]
[1] "PHP"

[[3]][[3]]
[1] "Java"


> li <- append(li, 4)
> li
[[1]]
[1] "Red"   "Green" "Black"

[[2]]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11

[[3]]
[[3]][[1]]
[1] "Python"

[[3]][[2]]
[1] "PHP"

[[3]][[3]]
```

```
[1] "Java"


[[4]]
[1] 4

>
> #Q.7 Write an R program to merge two given lists into one list.
> List1= list(1, 2, 3)
> List2 = list("Red", "Green", "Black")
> c(List1, List2)
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] "Red"

[[5]]
[1] "Green"

[[6]]
[1] "Black"

>
> #Q.8 Write an R program to convert a given data frame to a list by rows.
> name <- c("Anastasia", "Dima", "Katherine", "James", "Emily", "Michael", "Matthew", "Laura",
"Kevin")
> score <- c(12.5, 9.0, 16.5, 12.0, 9.0, 20.0, 14.5, 13.5, 8.0)
> attempts <- c(1, 3, 2, 3, 2, 3, 1, 1, 2)
> qualify <- c("yes", "no", "yes", "no", "no", "yes", "yes", "no", "no")
> df <- data.frame(name, score, attempts, qualify)
> colnames(df) <- c("Name", "Score", "attempts", "qualify")
> df
       Name Score attempts qualify
1 Anastasia  12.5        1     yes
```

```
2      Dima   9.0        3      no
3 Katherine  16.5        2     yes
4     James  12.0        3      no
5     Emily   9.0        2      no
6   Michael  20.0        3     yes
7   Matthew  14.5        1     yes
8     Laura  13.5        1      no
9     Kevin   8.0        2      no
> li2 <- list()
> li2 <- apply(df, 1, function(x) append(li2, x))
> li2
[[1]]
[[1]]$Name
[1] "Anastasia"

[[1]]$Score
[1] "12.5"

[[1]]$attempts
[1] "1"

[[1]]$qualify
[1] "yes"


[[2]]
[[2]]$Name
[1] "Dima"

[[2]]$Score
[1] " 9.0"

[[2]]$attempts
[1] "3"

[[2]]$qualify
[1] "no"


[[3]]
[[3]]$Name
```

```
[1] "Katherine"

[[3]]$Score
[1] "16.5"

[[3]]$attempts
[1] "2"

[[3]]$qualify
[1] "yes"


[[4]]
[[4]]$Name
[1] "James"

[[4]]$Score
[1] "12.0"

[[4]]$attempts
[1] "3"

[[4]]$qualify
[1] "no"


[[5]]
[[5]]$Name
[1] "Emily"

[[5]]$Score
[1] " 9.0"

[[5]]$attempts
[1] "2"

[[5]]$qualify
[1] "no"


[[6]]
```

```
[[6]]$Name
[1] "Michael"

[[6]]$Score
[1] "20.0"

[[6]]$attempts
[1] "3"

[[6]]$qualify
[1] "yes"


[[7]]
[[7]]$Name
[1] "Matthew"

[[7]]$Score
[1] "14.5"

[[7]]$attempts
[1] "1"

[[7]]$qualify
[1] "yes"


[[8]]
[[8]]$Name
[1] "Laura"

[[8]]$Score
[1] "13.5"

[[8]]$attempts
[1] "1"

[[8]]$qualify
[1] "no"
```

```
[[9]]
[[9]]$Name
[1] "Kevin"

[[9]]$Score
[1] " 8.0"

[[9]]$attempts
[1] "2"

[[9]]$qualify
[1] "no"


>
```

> #Q.9 Write an R program to create a correlation matrix from a data frame of the same data type.

```
> d <- data.frame(x1=rnorm(5), x2=rnorm(5), x3=rnorm(5))
> d
           x1         x2         x3
1 -0.89691455  0.1324203  0.4176508
2  0.18484918  0.7079547  0.9817528
3  1.58784533 -0.2396980 -0.3926954
4 -1.13037567  1.9844739 -1.0396690
5 -0.08025176 -0.1387870  1.7822290
> cor(d)
           x1         x2         x3
x1  1.0000000 -0.6205198  0.0696402
x2 -0.6205198  1.0000000 -0.5720479
x3  0.0696402 -0.5720479  1.0000000
>
```

> #Q.10 Write an R program to rotate a given matrix 90 degrees clockwise.

```
> mt <- matrix(1:9, ncol=3)
> mt
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> clockwise <- function(x){ t(apply(x, 2, rev)) }
> clockwise(clockwise(mt))
     [,1] [,2] [,3]
```

```
[1,]    9    6    3
[2,]    8    5    2
[3,]    7    4    1
>
```

> #Q.11 Check for missing values in the ‘mtcars’ data set.

```
> sum(is.na(mtcars))
[1] 0
>
```

> #Q.12 Check which attributes are important to determine the mpg of a car in the ‘mtcars’ data set.

```
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
> library(caret)
Loading required package: ggplot2
Need help getting started? Try the R Graphics Cookbook: https://r-graphics.org
Loading required package: lattice
> reg_model1 <- lm(mpg ~ ., data=mtcars)
> summary(reg_model1)

Call:
lm(formula = mpg ~ ., data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4506 -1.6044 -0.1196  1.2193  4.6271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.30337   18.71788   0.657   0.5181
cyl         -0.11144    1.04502  -0.107   0.9161
```

```
disp         0.01334    0.01786   0.747   0.4635
hp          -0.02148    0.02177  -0.987   0.3350
drat         0.78711    1.63537   0.481   0.6353
wt          -3.71530    1.89441  -1.961   0.0633 .
qsec         0.82104    0.73084   1.123   0.2739
vs           0.31776    2.10451   0.151   0.8814
am           2.52023    2.05665   1.225   0.2340
gear         0.65541    1.49326   0.439   0.6652
carb        -0.19942    0.82875  -0.241   0.8122
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.65 on 21 degrees of freedom
Multiple R-squared:  0.869,    Adjusted R-squared:  0.8066
F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07

> reg_model2 <- lm(mpg ~ cyl+hp+drat+wt+qsec+vs+am+gear+carb, data=mtcars)
> summary(reg_model2)

Call:
lm(formula = mpg ~ cyl + hp + drat + wt + qsec + vs + am + gear +
    carb, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7863 -1.4055 -0.2635  1.2029  4.4753

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.55052   18.52585   0.677   0.5052
cyl          0.09627    0.99715   0.097   0.9240
hp          -0.01295    0.01834  -0.706   0.4876
drat         0.92864    1.60794   0.578   0.5694
wt          -2.62694    1.19800  -2.193   0.0392 *
qsec         0.66523    0.69335   0.959   0.3478
vs           0.16035    2.07277   0.077   0.9390
am           2.47882    2.03513   1.218   0.2361
gear         0.74300    1.47360   0.504   0.6191
carb        -0.61686    0.60566  -1.018   0.3195
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 2.623 on 22 degrees of freedom
Multiple R-squared:  0.8655,  Adjusted R-squared:  0.8105
F-statistic: 15.73 on 9 and 22 DF,  p-value: 1.183e-07

> reg_model3 <- lm(mpg ~ cyl+wt+qsec+wt+am+gear+carb, data=mtcars)
> summary(reg_model3)

Call:
lm(formula = mpg ~ cyl + wt + qsec + wt + am + gear + carb, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.2148 -1.1992 -0.2412  1.4018  4.4595

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  16.2624    15.9126   1.022   0.3166
cyl          -0.3137     0.7971  -0.393   0.6973
wt           -2.9548     1.0611  -2.785   0.0101 *
qsec          0.7695     0.5853   1.315   0.2005
am            2.6522     1.8807   1.410   0.1708
gear          0.6415     1.3835   0.464   0.6469
carb         -0.6764     0.5481  -1.234   0.2287
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.509 on 25 degrees of freedom
Multiple R-squared:  0.8602,  Adjusted R-squared:  0.8267
F-statistic: 25.64 on 6 and 25 DF,  p-value: 1.542e-09

> reg_model4 <- lm(mpg ~ cyl+qsec+am+wt+carb, data=mtcars)
> summary(reg_model4)

Call:
lm(formula = mpg ~ cyl + qsec + am + wt + carb, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.2795 -1.2098 -0.3826  1.3961  4.4050
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  20.0379    13.4638   1.488  0.14871
cyl          -0.4612     0.7198  -0.641  0.52728
qsec          0.7272     0.5693   1.277  0.21276
am            2.9417     1.7471   1.684  0.10419
wt           -3.0462     1.0268  -2.967  0.00638 **
carb         -0.5222     0.4291  -1.217  0.23456
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.471 on 26 degrees of freedom
Multiple R-squared:  0.859,	Adjusted R-squared:  0.8319
F-statistic: 31.69 on 5 and 26 DF,  p-value: 2.847e-10

> reg_model5 <- lm(mpg ~ qsec+gear+wt+carb, data=mtcars)
> summary(reg_model5)

Call:
lm(formula = mpg ~ qsec + gear + wt + carb, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5079 -1.9616 -0.0451  0.8937  5.2861

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  13.5291     8.1655   1.657 0.109125
qsec          0.7613     0.3462   2.199 0.036626 *
gear          1.9228     1.0973   1.752 0.091081 .
wt           -3.7019     0.8902  -4.159 0.000291 ***
carb         -0.7848     0.5470  -1.435 0.162841
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.543 on 27 degrees of freedom
Multiple R-squared:  0.8449,  Adjusted R-squared:  0.8219
F-statistic: 36.77 on 4 and 27 DF,  p-value: 1.471e-10

> reg_model6 <- lm(mpg ~ qsec+gear+wt, data=mtcars)
> summary(reg_model6)
```

```
Call:
lm(formula = mpg ~ qsec + gear + wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.7178 -1.7915 -0.3533  1.1897  5.6333 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  12.9432     8.3080   1.558  0.13049    
qsec          1.0486     0.2877   3.645  0.00108 ** 
gear          0.8914     0.8446   1.055  0.30025    
wt           -4.6178     0.6320  -7.306 5.91e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.591 on 28 degrees of freedom
Multiple R-squared:  0.8331,  Adjusted R-squared:  0.8152 
F-statistic: 46.57 on 3 and 28 DF,  p-value: 5.2e-11

> #After comparing the Adjusted R squared, Error and R squared value selecting model 5
>
```
> #Q.13 Build a simple linear model to predict the mpg of a car in the 'mtcars' data set.
```
> final_reg_model <- lm(mpg ~ qsec+gear+wt+carb, data=mtcars)
> prediction <- predict(final_reg_model, newdata=data.frame(qsec=16.5, wt=2.62, gear=4, carb=4))
> prediction
       1 
20.94371 
>
```
> #Q.14 Build a logistic regression model using the glm function to know the effect of admission into graduate school. The target variable,

> #admit/don't admit, is a binary variable Use the given "binary.csv" dataset.
```
> setwd("C:/zubeda/PGA02_Zubu/R Programming/R Exam/Dataset")
> admission <- read.csv("binary.csv")
> head(admission)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
```

```
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
> str(admission)
'data.frame':   400 obs. of  4 variables:
 $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
> admission$rank <- as.factor(admission$rank)
> logit <- glm(admit ~ gre+gpa+rank, data=admission, family="binomial")
> summary(logit)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6268  -0.8662  -0.6388   1.1490   2.0790

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4

>
```

> #Q.15 Use the given variables from the titanic dataset and build the decision tree on train data. Variables from dataset: survived, embarked, sex, sibsp, parch, fare

```
> titanic <- read.csv("Titanic_data.csv")
> head(titanic)
  PassengerId Survived Pclass                                                Name    Sex Age SibSp Parch           Ticket
Fare Cabin
1           1        0      3                             Braund, Mr. Owen Harris   male  22     1     0        A/5 21171
7.2500
2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
PC 17599 71.2833   C85
3           3        1      3                              Heikkinen, Miss. Laina female  26     0     0 STON/O2.
3101282  7.9250
4           4        1      1        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
113803 53.1000  C123
5           5        0      3                            Allen, Mr. William Henry   male  35     0     0           373450
8.0500
6           6        0      3                                    Moran, Mr. James   male  NA     0     0           330877
8.4583
  Embarked
1        S
2        C
3        S
4        S
5        S
6        Q
> titanic_df <- titanic[, c("Survived", "Embarked", "Sex", "SibSp", "Parch", "Fare")]
> head(titanic_df)
  Survived Embarked    Sex SibSp Parch    Fare
1        0        S   male     1     0  7.2500
2        1        C female     1     0 71.2833
3        1        S female     0     0  7.9250
4        1        S female     1     0 53.1000
5        0        S   male     0     0  8.0500
6        0        Q   male     0     0  8.4583
> library(rpart)
> library(caTools)
> set.seed(123)
> #Splitting dataset
> split <- sample.split(titanic_df, SplitRatio=0.8)
> training <- subset(titanic_df, split == TRUE)
> test <- subset(titanic_df, split == FALSE)
```

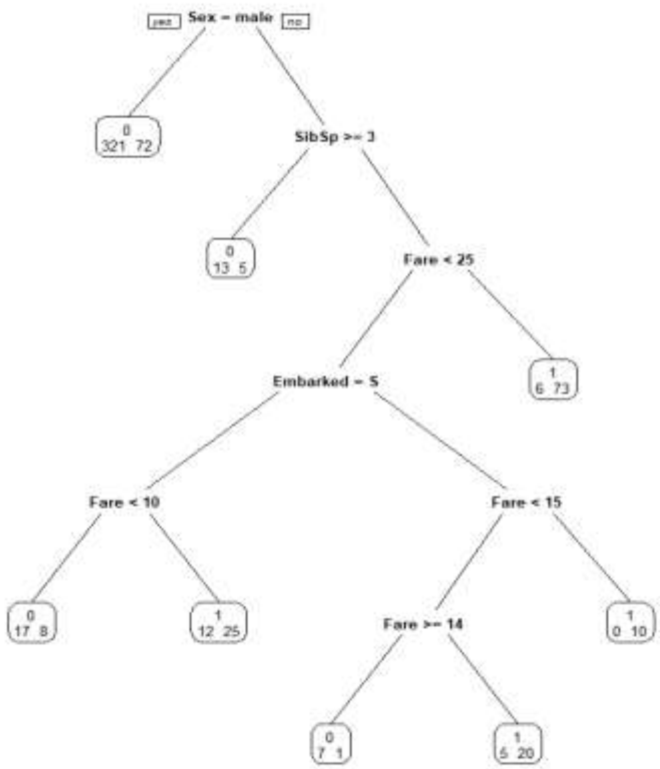
```
> dim(training)
[1] 595   6
> dim(test)
[1] 296   6
> training$Survived <- as.factor(training$Survived)
> test$Survived <- as.factor(test$Survived)
> #Building Decision Tree model
> mtree <- rpart(Survived ~ ., data=training, method="class")
> mtree
n= 595

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 595 214 0 (0.64033613 0.35966387)
    2) Sex=male 393  72 0 (0.81679389 0.18320611) *
    3) Sex=female 202  60 1 (0.29702970 0.70297030)
      6) SibSp>=2.5 18   5 0 (0.72222222 0.27777778) *
      7) SibSp< 2.5 184  47 1 (0.25543478 0.74456522)
       14) Fare< 25.075 105  41 1 (0.39047619 0.60952381)
         28) Embarked=S 62  29 1 (0.46774194 0.53225806)
           56) Fare< 10.48125 25   8 0 (0.68000000 0.32000000) *
           57) Fare>=10.48125 37  12 1 (0.32432432 0.67567568) *
         29) Embarked=C,Q 43  12 1 (0.27906977 0.72093023)
           58) Fare< 15.3729 33  12 1 (0.36363636 0.63636364)
            116) Fare>=14.15625 8   1 0 (0.87500000 0.12500000) *
            117) Fare< 14.15625 25   5 1 (0.20000000 0.80000000) *
           59) Fare>=15.3729 10   0 1 (0.00000000 1.00000000) *
       15) Fare>=25.075 79   6 1 (0.07594937 0.92405063) *
>
```

> #Q.16 Create a plot to display the result of decision tree.

```
> library(rpart.plot)
> prp(mtree, faclen=0, extra=1, cex=0.8)
```



>
> #Q.17 Create the confusion matrix for the above model.
> prediction <- predict(mtree, newdata=test[, -1], type="class")
> prediction
 4 5 10 11 16 17 22 23 28 29 34 35 40 41 46 47 52 53 58 59 64 65 70 71 76 77 82
83 88 89 94 95 100 101 106 107
 1 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0
112 113 118 119 124 125 130 131 136 137 142 143 148 149 154 155 160 161 166 167 172 173
178 179 184 185 190 191 196 197 202 203 208 209 214 215
 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0
1 0 0
220 221 226 227 232 233 238 239 244 245 250 251 256 257 262 263 268 269 274 275 280 281
286 287 292 293 298 299 304 305 310 311 316 317 322 323
 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 1 1 0
1 0 1
328 329 334 335 340 341 346 347 352 353 358 359 364 365 370 371 376 377 382 383 388 389
394 395 400 401 406 407 412 413 418 419 424 425 430 431
 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0 1
0 0 0

```
436 437 442 443 448 449 454 455 460 461 466 467 472 473 478 479 484 485 490 491 496 497
502 503 508 509 514 515 520 521 526 527 532 533 538 539
  1   1   0   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   1   1   1   0   0   1   0   0   1   0   1   0
0   1   0
544 545 550 551 556 557 562 563 568 569 574 575 580 581 586 587 592 593 598 599 604 605
610 611 616 617 622 623 628 629 634 635 640 641 646 647
  0   0   0   0   0   1   0   0   1   0   1   0   0   1   1   0   1   0   0   0   0   0   1   1   1   0   0   0   1   0   0   0   0
0   0   0
652 653 658 659 664 665 670 671 676 677 682 683 688 689 694 695 700 701 706 707 712 713
718 719 724 725 730 731 736 737 742 743 748 749 754 755
  1   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   1   0   1   0   0   1   0   0   0   0   1   0   1   0   1   1
0   0   1
760 761 766 767 772 773 778 779 784 785 790 791 796 797 802 803 808 809 814 815 820 821
826 827 832 833 838 839 844 845 850 851 856 857 862 863
  1   0   1   0   0   1   1   0   0   0   0   0   0   1   1   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   1   0   0
1   0   1
868 869 874 875 880 881 886 887
  0   0   0   1   1   1   1   0
Levels: 0 1
> confusionMatrix(data=prediction, reference=test$Survived, positive="1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 154  45
         1  14  83

               Accuracy : 0.8007
                 95% CI : (0.7506, 0.8447)
    No Information Rate : 0.5676
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5819

 Mcnemar's Test P-Value : 9.397e-05

            Sensitivity : 0.6484
            Specificity : 0.9167
         Pos Pred Value : 0.8557
         Neg Pred Value : 0.7739
             Prevalence : 0.4324
```

```
      Detection Rate : 0.2804
   Detection Prevalence : 0.3277
      Balanced Accuracy : 0.7826

       'Positive' Class : 1

>
```

> #Q.18 Perform k-means clustering on USArrest dataset. Scale the data before performing clustering.

```
> arrests <- USArrests
> head(arrests)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
> dim(arrests)
[1] 50  4
> str(arrests)
'data.frame':	50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
> data_standardized <- data.frame(scale(arrests))
> head(data_standardized)
               Murder   Assault   UrbanPop         Rape
Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
Arizona    0.07163341 1.4788032  0.9989801  1.042878388
Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
California 0.27826823 1.2628144  1.7589234  2.067820292
Colorado   0.02571456 0.3988593  0.8608085  1.864967207
> #Build kmeans model
> km <- kmeans(data_standardized, centers=2, nstart=20)
> cluster_df <- data_standardized
> cluster_df$Cluster <- km$cluster
> cluster_df
                    Murder     Assault    UrbanPop         Rape Cluster
```

Alabama	1.24256408	0.78283935	-0.52090661	-0.003416473	1
Alaska	0.50786248	1.10682252	-1.21176419	2.484202941	1
Arizona	0.07163341	1.47880321	0.99898006	1.042878388	1
Arkansas	0.23234938	0.23086801	-1.07359268	-0.184916602	2
California	0.27826823	1.26281442	1.75892340	2.067820292	1
Colorado	0.02571456	0.39885929	0.86080854	1.864967207	1
Connecticut	-1.03041900	-0.72908214	0.79172279	-1.081740768	2
Delaware	-0.43347395	0.80683810	0.44629400	-0.579946294	2
Florida	1.74767144	1.97077766	0.99898006	1.138966691	1
Georgia	2.20685994	0.48285493	-0.38273510	0.487701523	1
Hawaii	-0.57123050	-1.49704226	1.20623733	-0.110181255	2
Idaho	-1.19113497	-0.60908837	-0.79724965	-0.750769945	2
Illinois	0.59970018	0.93883125	1.20623733	0.295524916	1
Indiana	-0.13500142	-0.69308401	-0.03730631	-0.024769429	2
Iowa	-1.28297267	-1.37704849	-0.58999237	-1.060387812	2
Kansas	-0.41051452	-0.66908525	0.03177945	-0.345063775	2
Kentucky	0.43898421	-0.74108152	-0.93542116	-0.526563903	2
Louisiana	1.74767144	0.93883125	0.03177945	0.103348309	1
Maine	-1.30593210	-1.05306531	-1.00450692	-1.434064548	2
Maryland	0.80633501	1.55079947	0.10086521	0.701231086	1
Massachusetts	-0.77786532	-0.26110644	1.34440885	-0.526563903	2
Michigan	0.99001041	1.01082751	0.58446551	1.480613993	1
Minnesota	-1.16817555	-1.18505846	0.03177945	-0.676034598	2
Mississippi	1.90838741	1.05882502	-1.48810723	-0.441152078	1
Missouri	0.27826823	0.08687549	0.30812248	0.743936999	1
Montana	-0.41051452	-0.74108152	-0.86633540	-0.515887425	2
Nebraska	-0.80082475	-0.82507715	-0.24456358	-0.505210947	2
Nevada	1.01296983	0.97482938	1.06806582	2.644350114	1
New Hampshire	-1.30593210	-1.36504911	-0.65907813	-1.252564419	2
New Jersey	-0.08908257	-0.14111267	1.62075188	-0.259651949	2
New Mexico	0.82929443	1.37080881	0.30812248	1.160319648	1
New York	0.76041616	0.99882813	1.41349461	0.519730957	1
North Carolina	1.19664523	1.99477641	-1.41902147	-0.547916860	1
North Dakota	-1.60440462	-1.50904164	-1.48810723	-1.487446939	2
Ohio	-0.11204199	-0.60908837	0.65355127	0.017936483	2
Oklahoma	-0.27275797	-0.23710769	0.16995096	-0.131534211	2
Oregon	-0.66306820	-0.14111267	0.10086521	0.861378259	2
Pennsylvania	-0.34163624	-0.77707965	0.44629400	-0.676034598	2
Rhode Island	-1.00745957	0.03887798	1.48258036	-1.380682157	2
South Carolina	1.51807718	1.29881255	-1.21176419	0.135377743	1
South Dakota	-0.91562187	-1.01706718	-1.41902147	-0.900240639	2

```
Tennessee      1.24256408  0.20686926 -0.45182086  0.605142783       1
Texas          1.12776696  0.36286116  0.99898006  0.455672088       1
Utah          -1.05337842 -0.60908837  0.99898006  0.178083656       2
Vermont       -1.28297267 -1.47304350 -2.31713632 -1.071064290       2
Virginia       0.16347111 -0.17711080 -0.17547783 -0.056798864       2
Washington    -0.86970302 -0.30910395  0.51537975  0.530407436       2
West Virginia -0.47939280 -1.07706407 -1.83353601 -1.273917376       2
Wisconsin     -1.19113497 -1.41304662  0.03177945 -1.113770203       2
Wyoming       -0.22683912 -0.11711392 -0.38273510 -0.601299251       2
>
```

> #Q.19 Print the cluster number for each observation and cluster size for the above k-means model.

```
> table(cluster_df$Cluster)

 1  2
20 30
>
```

> #Q.20 Plot the result of the k-means cluster.

```
> library(factoextra)
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
> fviz_cluster(km, data=cluster_df[, -5], palette=c("aquamarine", "orange"), geom="point", ellipse.type="convex", ggtheme=theme_bw())
```