Assignment 2

Sharmin Akhter

2023-03-15

Contents


```r
library(MASS)
library(htmltools)
library(klaR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## x dplyr::select() masks MASS::select()
```

```
library (ISLR)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(mvtnorm)
library(qcc)
```

```
## Package 'qcc' version 2.7
## Type 'citation("qcc")' for citing this R package in publications.
```

Question 1

Observations on two response variables are collected for two treatments. The observation vectors [x1; x2] are Treatment 1:

$$(3, 3)(1, 6), (2, 3)$$

Treatment 2:

$$(2, 3), (5, 1), (3, 1), (2, 3)$$

a) Calculate the Spooled b) Test

$$H_0 : \mu_1 = \mu_2$$

employing a two sample approach with

$$\alpha = 0.01$$

Question 1(a)

##Calculate the Spooled #Ans:

We know,

$$S_{pooled} = \frac{n_1 - 1}{n_1 + n_2 - 2} s_1 + \frac{n_2 - 1}{n_1 + n_2 - 2} s_2$$

#Create matix

```
treatment1 <- matrix(c(3, 1, 2, 3, 6, 3), nrow = 3)
treatment1
```

```
##      [,1] [,2]
## [1,]    3    3
## [2,]    1    6
## [3,]    2    3
```

```
treatment2 <- matrix(c(2, 5, 3, 2, 3, 1, 1, 3), ncol = 2)
treatment2
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    5    1
## [3,]    3    1
## [4,]    2    3
```

Here n1, n2 are sample size(length) of treatment1 and treatment2 which are

```
n1 <- nrow(treatment1)
n2 <- nrow(treatment2)
n1
```

```
## [1] 3
```

```r
n2
```

```
## [1] 4
```

#Mean and Variance of Treatment1 and Treatment 2

```r
mean_treat1 <- colMeans(treatment1)
mean_treat1
```

```
## [1] 2 4
```

```r
mean_treat2 <- colMeans(treatment2)
mean_treat2
```

```
## [1] 3 2
```

```r
s1 <- cov(treatment1)
s1
```

```
##      [,1] [,2]
## [1,]  1.0 -1.5
## [2,] -1.5  3.0
```

```r
s2 <- cov(treatment2)
s2
```

```
##           [,1]      [,2]
## [1,]  2.000000 -1.333333
## [2,] -1.333333  1.333333
```

#Now s_pooles(pooled standard deviation) by formula

```r
sp <- ((n1-1)/(n1+n2-2))*s1+((n2-1)/(n1+n2-2))*s2
sp
```

```
##      [,1] [,2]
## [1,]  1.6 -1.4
## [2,] -1.4  2.0
```

Question 1(b)

Test

$$H_0 : \mu_1 = \mu_2$$

employing a two sample approach with

$$\alpha = 0.01$$

```r
# Calculate the two sample t-test statistic
T2 <- t((mean_treat1 - mean_treat2)) %*% solve((sp*(1/n1 + 1/n2))) %*% (mean_treat1 - mean_treat2)
T2
```

```
##          [,1]
## [1,] 3.870968
```

```r
alpha <- 0.01
F <- qf(1-alpha, 2, n1+n2-2-1)
T <- (((n1+n2-2)*2) / (n1+n2-2-1))*F
T
```

```
## [1] 45
```

```
#t.test(treatment1, treatment2)
```

#Implement

the calculated t-value (t) falls within the acceptance region, and we cannot conclude that there is a significant difference between the means of the two groups. That means we can't reject null hypothesis.

Question 2

Generate a data set with two explanatory variables x1 and x2 from multinomial Normal distribution with covariance matrix

$$\sigma = c(1, .2, .2, 4)$$

in two classes with **mean for Class 0 is (3,7)** and and **Class 1 is (6,10)**. For the Class 0, generate 50 observations and for Class 1, 50 observations. While generating this data, use the set.seed("99"). **Find the linear discriminant function weights.** Plot the data with **two colors and draw the discriminant function for classification.** Also plot the 4 test data (3.68; 5.65); (3.28; 5.20); (3.57; 8.82); (4.64; 7.98) and predict the test data. Use the R program also to predict the test data.

Part1

Generate a data set Given variables

$$x_1, x_2$$

from multinomial normal distribution with covariance matrix

$$\sigma = c(1, .2, .2, 4)$$

. Alse here class 0 mean is (3,7) and class 1 mean is (6,10).

Now we have to generate 50 observations for class 0 and 50 observations for class 1 with set.seed(99)

#For class 0

```
set.seed(99)
mu <- c(3, 7)
sigma <- matrix(c(1, .2, .2, 4), nrow = 2)
class0 <- mvrnorm(50, mu, sigma)
class0
```

```
##           [,1]     [,2]
##  [1,] 4.591003 7.323978
##  [2,] 2.445073 7.999853
##  [3,] 2.684351 7.197286
##  [4,] 3.453491 7.861046
##  [5,] 4.023698 6.203576
##  [6,] 3.092518 7.240155
##  [7,] 3.406281 5.238671
##  [8,] 2.677130 8.004460
##  [9,] 3.626129 6.227396
## [10,] 3.570366 4.363663
## [11,] 3.062533 5.498551
## [12,] 3.235471 8.834590
## [13,] 3.540241 8.470044
## [14,] 2.412492 2.002511
## [15,] 3.686958 0.849049
## [16,] 4.324795 6.912603
## [17,] 2.744637 6.225868
```

```
## [18,] 2.733505  3.514155
## [19,] 3.501188  7.967849
## [20,] 2.639082  7.567956
## [21,] 2.565800  9.235150
## [22,] 2.621973  8.535930
## [23,] 3.789836  6.828284
## [24,] 2.653887  6.331174
## [25,] 3.848414  7.390745
## [26,] 2.933029  8.112280
## [27,] 2.356889  8.415251
## [28,] 1.554774  5.999949
## [29,] 3.446553  4.224927
## [30,] 2.949271  9.814284
## [31,] 3.145881  9.747030
## [32,] 2.773762  7.919007
## [33,] 4.076785  6.634813
## [34,] 2.132644  7.314753
## [35,] 1.524525  2.490766
## [36,] 2.470628  4.291443
## [37,] 2.911350  6.609400
## [38,] 1.627097  7.227821
## [39,] 3.024405  7.180086
## [40,] 2.370740  7.689779
## [41,] 3.073293  7.262120
## [42,] 2.642413  3.652235
## [43,] 1.114960  6.566006
## [44,] 1.655482  3.972427
## [45,] 3.748626  4.180257
## [46,] 2.881293  4.283284
## [47,] 3.250555  5.133981
## [48,] 2.677161  5.281097
## [49,] 2.520572 10.357905
## [50,] 2.333101  6.732907
```

#For class 1

```
mu1 <- c(6, 10)
sigma <- matrix(c(1, 0.2, 0.2, 4), nrow = 2)
class1 <- mvrnorm(50, mu1, sigma)
class1
```

```
##           [,1]      [,2]
##  [1,] 7.469773  6.840529
##  [2,] 6.484840  8.961967
##  [3,] 4.491158  7.664414
##  [4,] 4.696785  8.821077
##  [5,] 5.258125  7.143089
##  [6,] 6.712782  9.617585
##  [7,] 6.137532 13.180997
##  [8,] 5.466893  9.572890
##  [9,] 3.850817  8.991476
## [10,] 5.105644 11.189173
## [11,] 6.080613  9.502926
## [12,] 7.059296 14.021736
## [13,] 4.727180  9.580137
```

```
## [14,] 5.869102  3.988182
## [15,] 7.472393  7.503576
## [16,] 7.140383 11.864606
## [17,] 5.245036  7.667080
## [18,] 5.459430  8.330029
## [19,] 5.043570  9.242343
## [20,] 6.917700  6.269495
## [21,] 4.748965 11.276266
## [22,] 6.768502 10.710625
## [23,] 5.286645  8.152155
## [24,] 5.152890  6.717313
## [25,] 6.822309 10.003599
## [26,] 5.388397 11.634677
## [27,] 7.031621 10.057030
## [28,] 4.394722 12.939472
## [29,] 5.574675 13.042667
## [30,] 5.217067  7.553373
## [31,] 6.808589 10.766968
## [32,] 7.893824  9.439355
## [33,] 6.546334 11.621058
## [34,] 6.925997 11.791458
## [35,] 7.148120  9.052979
## [36,] 7.622595 10.033361
## [37,] 6.612334 11.858975
## [38,] 4.202222 10.946762
## [39,] 5.973736  6.299143
## [40,] 4.944677 10.292652
## [41,] 5.539114 11.104863
## [42,] 5.050976 13.318595
## [43,] 7.728366 10.494864
## [44,] 7.171268 11.447790
## [45,] 6.838125  8.861307
## [46,] 6.985123  8.553408
## [47,] 5.349125  9.530215
## [48,] 8.063451 10.465073
## [49,] 5.719048  8.727354
## [50,] 6.771416 12.933805
```

#Combine into a single data frame with a class variable

```
data <- data.frame(cbind(rbind(class0, class1), rep(0:1, each = 50)))
colnames(data) <- c("x1", "x2", "class")
colnames(data)
```

```
## [1] "x1"    "x2"    "class"
```

```
data
```

```
##          x1       x2 class
## 1  4.591003 7.323978     0
## 2  2.445073 7.999853     0
## 3  2.684351 7.197286     0
## 4  3.453491 7.861046     0
## 5  4.023698 6.203576     0
## 6  3.092518 7.240155     0
## 7  3.406281 5.238671     0
```

```
## 8   2.677130  8.004460      0
## 9   3.626129  6.227396      0
## 10  3.570366  4.363663      0
## 11  3.062533  5.498551      0
## 12  3.235471  8.834590      0
## 13  3.540241  8.470044      0
## 14  2.412492  2.002511      0
## 15  3.686958  0.849049      0
## 16  4.324795  6.912603      0
## 17  2.744637  6.225868      0
## 18  2.733505  3.514155      0
## 19  3.501188  7.967849      0
## 20  2.639082  7.567956      0
## 21  2.565800  9.235150      0
## 22  2.621973  8.535930      0
## 23  3.789836  6.828284      0
## 24  2.653887  6.331174      0
## 25  3.848414  7.390745      0
## 26  2.933029  8.112280      0
## 27  2.356889  8.415251      0
## 28  1.554774  5.999949      0
## 29  3.446553  4.224927      0
## 30  2.949271  9.814284      0
## 31  3.145881  9.747030      0
## 32  2.773762  7.919007      0
## 33  4.076785  6.634813      0
## 34  2.132644  7.314753      0
## 35  1.524525  2.490766      0
## 36  2.470628  4.291443      0
## 37  2.911350  6.609400      0
## 38  1.627097  7.227821      0
## 39  3.024405  7.180086      0
## 40  2.370740  7.689779      0
## 41  3.073293  7.262120      0
## 42  2.642413  3.652235      0
## 43  1.114960  6.566006      0
## 44  1.655482  3.972427      0
## 45  3.748626  4.180257      0
## 46  2.881293  4.283284      0
## 47  3.250555  5.133981      0
## 48  2.677161  5.281097      0
## 49  2.520572 10.357905      0
## 50  2.333101  6.732907      0
## 51  7.469773  6.840529      1
## 52  6.484840  8.961967      1
## 53  4.491158  7.664414      1
## 54  4.696785  8.821077      1
## 55  5.258125  7.143089      1
## 56  6.712782  9.617585      1
## 57  6.137532 13.180997      1
## 58  5.466893  9.572890      1
## 59  3.850817  8.991476      1
## 60  5.105644 11.189173      1
## 61  6.080613  9.502926      1
```

```
## 62  7.059296 14.021736     1
## 63  4.727180  9.580137     1
## 64  5.869102  3.988182     1
## 65  7.472393  7.503576     1
## 66  7.140383 11.864606     1
## 67  5.245036  7.667080     1
## 68  5.459430  8.330029     1
## 69  5.043570  9.242343     1
## 70  6.917700  6.269495     1
## 71  4.748965 11.276266     1
## 72  6.768502 10.710625     1
## 73  5.286645  8.152155     1
## 74  5.152890  6.717313     1
## 75  6.822309 10.003599     1
## 76  5.388397 11.634677     1
## 77  7.031621 10.057030     1
## 78  4.394722 12.939472     1
## 79  5.574675 13.042667     1
## 80  5.217067  7.553373     1
## 81  6.808589 10.766968     1
## 82  7.893824  9.439355     1
## 83  6.546334 11.621058     1
## 84  6.925997 11.791458     1
## 85  7.148120  9.052979     1
## 86  7.622595 10.033361     1
## 87  6.612334 11.858975     1
## 88  4.202222 10.946762     1
## 89  5.973736  6.299143     1
## 90  4.944677 10.292652     1
## 91  5.539114 11.104863     1
## 92  5.050976 13.318595     1
## 93  7.728366 10.494864     1
## 94  7.171268 11.447790     1
## 95  6.838125  8.861307     1
## 96  6.985123  8.553408     1
## 97  5.349125  9.530215     1
## 98  8.063451 10.465073     1
## 99  5.719048  8.727354     1
## 100 6.771416 12.933805     1
```

Part2

Find the linear discriminant function weights. # check the LDA coefficients/scalings

```
lda.model <- lda(class ~ x1 + x2, data = data)
lda.model
```

```
## Call:
## lda(class ~ x1 + x2, data = data)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
```

```
##           x1       x2
## 0 2.922533 6.498367
## 1 6.059386 9.791609
##
## Coefficients of linear discriminants:
##           LD1
## x1 0.9774977
## x2 0.1835658
```

```
lda.model$scaling
```

```
##           LD1
## x1 0.9774977
## x2 0.1835658
```

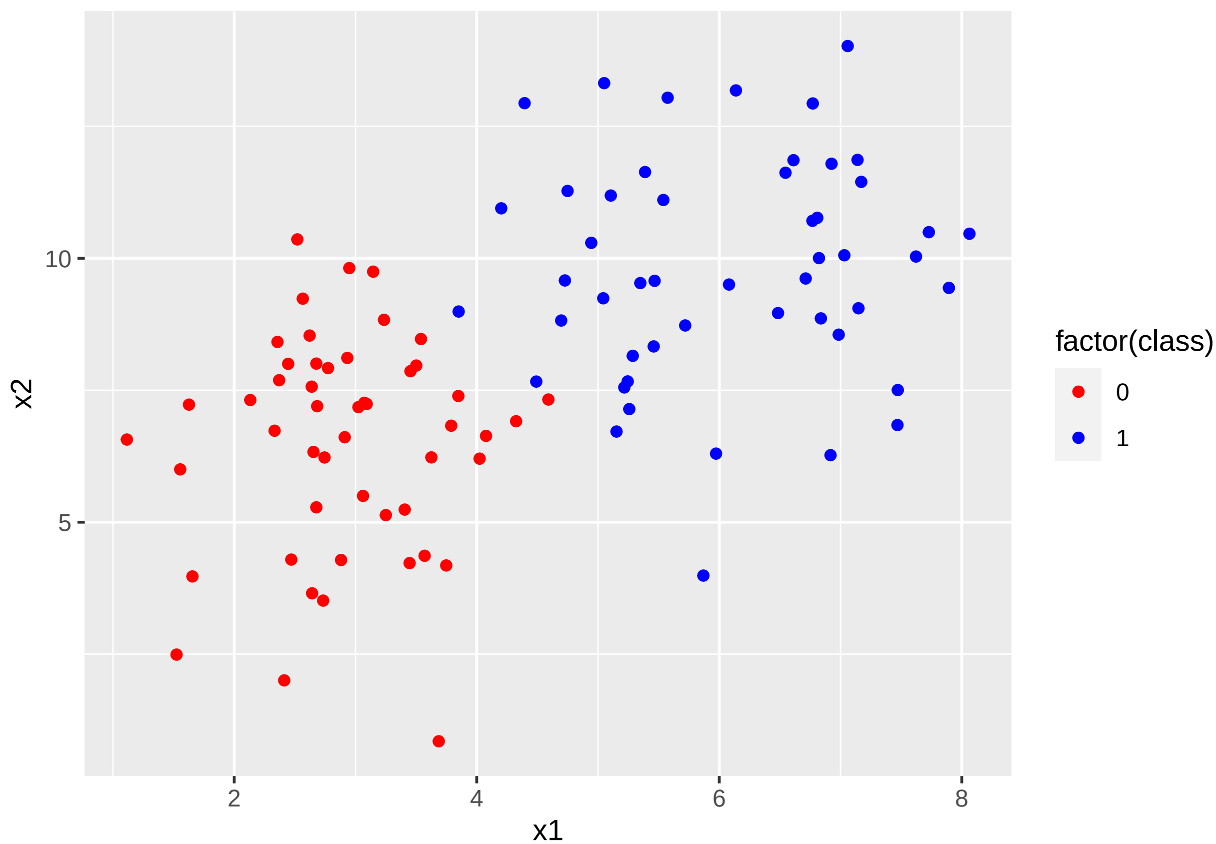
```
diag(lda.model$scaling)
```

```
## [1] 0.9774977
```

Part3

Plot the data with **two colors and draw the discriminant function for classification.**

```
# Plot the data with colors based on class
ggplot(data, aes(x = x1, y = x2, color = factor(class))) +
  geom_point() +
  scale_color_manual(values = c("red", "blue"))
```



Add the linear discriminant function as a line

```
x1_means <- lda.model$means[,1]
x1_means
```

```
##        0        1
## 2.922533 6.059386
```

```
x2_means <- lda.model$means[, 2]
x2_means
```

```
##        0        1
## 6.498367 9.791609
```

```
w <- lda.model$scaling
w1 <- w[1,]
w1
```

```
## [1] 0.9774977
```

```
w2 <- w[2,]
w2
```

```
## [1] 0.1835658
```

```
a <- t(w) %*% ((x1_means + x2_means)/2)
a
```

```
##         [,1]
## LD1 6.059304
```

```
seq_P1 <- seq(min(data$x1), max(data$x2), 0.1)
seq_P1
```

```
##   [1]  1.11496  1.21496  1.31496  1.41496  1.51496  1.61496  1.71496  1.81496
##   [9]  1.91496  2.01496  2.11496  2.21496  2.31496  2.41496  2.51496  2.61496
##  [17]  2.71496  2.81496  2.91496  3.01496  3.11496  3.21496  3.31496  3.41496
##  [25]  3.51496  3.61496  3.71496  3.81496  3.91496  4.01496  4.11496  4.21496
##  [33]  4.31496  4.41496  4.51496  4.61496  4.71496  4.81496  4.91496  5.01496
##  [41]  5.11496  5.21496  5.31496  5.41496  5.51496  5.61496  5.71496  5.81496
##  [49]  5.91496  6.01496  6.11496  6.21496  6.31496  6.41496  6.51496  6.61496
##  [57]  6.71496  6.81496  6.91496  7.01496  7.11496  7.21496  7.31496  7.41496
##  [65]  7.51496  7.61496  7.71496  7.81496  7.91496  8.01496  8.11496  8.21496
##  [73]  8.31496  8.41496  8.51496  8.61496  8.71496  8.81496  8.91496  9.01496
##  [81]  9.11496  9.21496  9.31496  9.41496  9.51496  9.61496  9.71496  9.81496
##  [89]  9.91496 10.01496 10.11496 10.21496 10.31496 10.41496 10.51496 10.61496
##  [97] 10.71496 10.81496 10.91496 11.01496 11.11496 11.21496 11.31496 11.41496
## [105] 11.51496 11.61496 11.71496 11.81496 11.91496 12.01496 12.11496 12.21496
## [113] 12.31496 12.41496 12.51496 12.61496 12.71496 12.81496 12.91496 13.01496
## [121] 13.11496 13.21496 13.31496 13.41496 13.51496 13.61496 13.71496 13.81496
## [129] 13.91496 14.01496
```

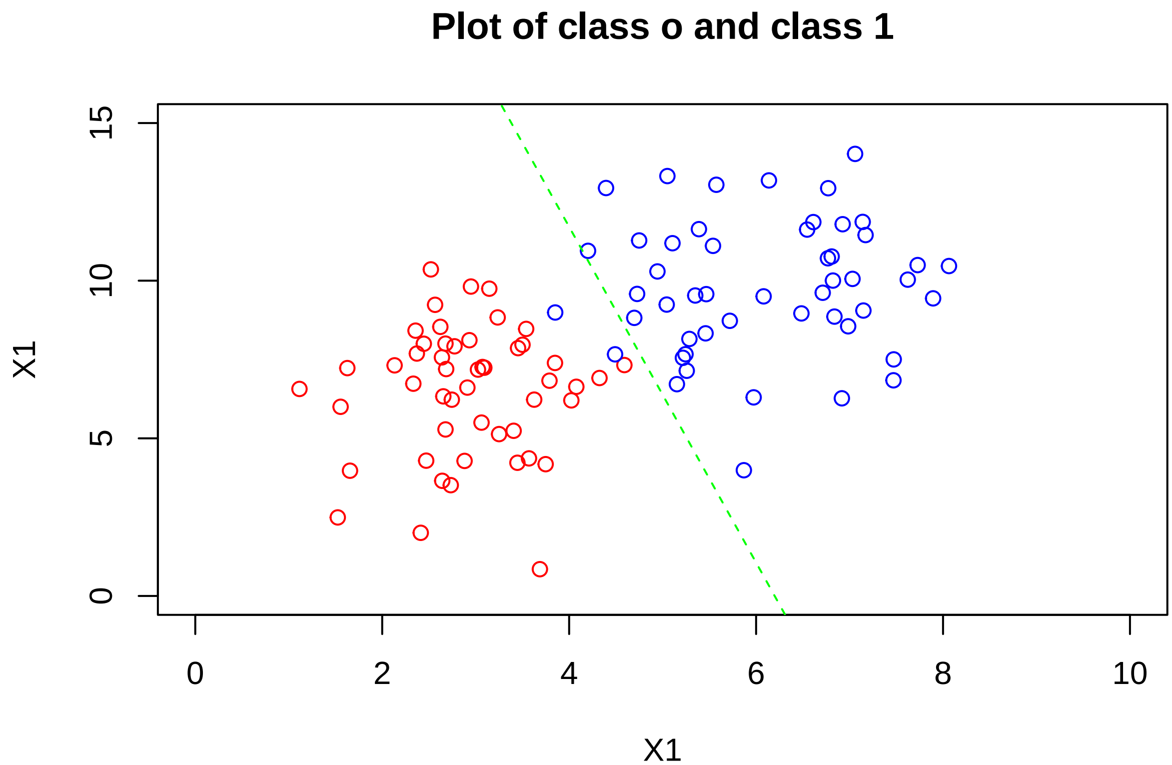
```
seq_P2 <- c(a/w[2,]) - ((w[1,]/w[2,])*seq_P1)
seq_P2
```

```
##   [1]  27.07167094  26.53916561  26.00666029  25.47415496  24.94164964
##   [6]  24.40914431  23.87663899  23.34413366  22.81162834  22.27912301
##  [11]  21.74661769  21.21411236  20.68160703  20.14910171  19.61659638
##  [16]  19.08409106  18.55158573  18.01908041  17.48657508  16.95406976
##  [21]  16.42156443  15.88905911  15.35655378  14.82404846  14.29154313
##  [26]  13.75903781  13.22653248  12.69402716  12.16152183  11.62901651
```

```
##  [31]  11.09651118  10.56400586  10.03150053   9.49899521   8.96648988
##  [36]   8.43398456   7.90147923   7.36897391   6.83646858   6.30396326
##  [41]   5.77145793   5.23895261   4.70644728   4.17394196   3.64143663
##  [46]   3.10893131   2.57642598   2.04392066   1.51141533   0.97891001
##  [51]   0.44640468  -0.08610064  -0.61860597  -1.15111129  -1.68361662
##  [56]  -2.21612195  -2.74862727  -3.28113260  -3.81363792  -4.34614325
##  [61]  -4.87864857  -5.41115390  -5.94365922  -6.47616455  -7.00866987
##  [66]  -7.54117520  -8.07368052  -8.60618585  -9.13869117  -9.67119650
##  [71] -10.20370182 -10.73620715 -11.26871247 -11.80121780 -12.33372312
##  [76] -12.86622845 -13.39873377 -13.93123910 -14.46374442 -14.99624975
##  [81] -15.52875507 -16.06126040 -16.59376572 -17.12627105 -17.65877637
##  [86] -18.19128170 -18.72378702 -19.25629235 -19.78879767 -20.32130300
##  [91] -20.85380832 -21.38631365 -21.91881897 -22.45132430 -22.98382962
##  [96] -23.51633495 -24.04884027 -24.58134560 -25.11385093 -25.64635625
## [101] -26.17886158 -26.71136690 -27.24387223 -27.77637755 -28.30888288
## [106] -28.84138820 -29.37389353 -29.90639885 -30.43890418 -30.97140950
## [111] -31.50391483 -32.03642015 -32.56892548 -33.10143080 -33.63393613
## [116] -34.16644145 -34.69894678 -35.23145210 -35.76395743 -36.29646275
## [121] -36.82896808 -37.36147340 -37.89397873 -38.42648405 -38.95898938
## [126] -39.49149470 -40.02400003 -40.55650535 -41.08901068 -41.62151600
```

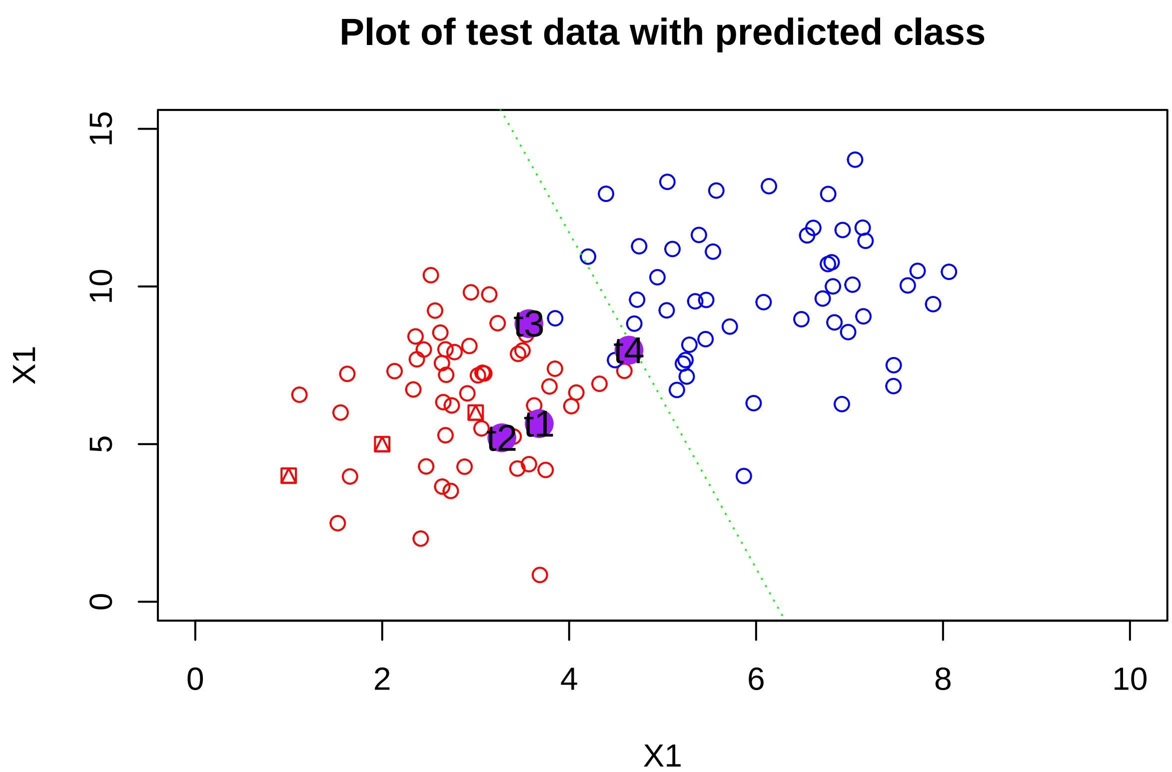
#LDA line

```
plot(class0, xlim = c(0,10), ylim = c(0,15), xlab = "X1", ylab = "X1", col = "red", main = "Plot of clas
points(class1, col = "blue")
lines(seq_P1, seq_P2, col = "green", lty = 2)  # change color to green and line type to dashed
```



#Also plot the 4 test data (3.68; 5.65); (3.28; 5.20); (3.57; 8.82); (4.64; 7.98) and predict the test data. Use the R program also to predict the test data.

```
#Test data set
t1 <- c(3.68,5.65)
t2 <- c(3.28, 5.20)
t3 <- c(3.57,8.82)
t4 <- c(4.64,7.98)

test_data <- rbind(t1,t2,t3,t4)
test_frame <- data.frame(test_data)

# Plot the above samples and color by class labels
plot(class0, xlim = c(0,10), ylim = c(0,15), xlab = "X1", ylab = "X1", col = "red", main = "Plot of tes
points(class1, col = "blue")
lines(seq_P1, seq_P2, col = "green", lty = 3)  # change color to green and line type to dashed


# Add points to the plot
points(x = c(1, 2, 3), y = c(4, 5, 6), col = "red", pch = 14)

# Add first point of the test dataset
points(t1[1],t1[2],col="purple", pch=16, cex=2)
text(t1[1],t1[2],labels ="t1",cex = 1.2)

# Add second point of the test dataset
points(t2[1],t2[2],col="purple", pch=16, cex=2)
text(t2[1],t2[2],labels ="t2",cex = 1.2)

# Add third point of the test dataset
points(t3[1],t3[2],col="purple", pch=16, cex=2)
text(t3[1],t3[2],labels ="t3",cex = 1.2)

# Add fourth point of the test dataset
points(t4[1],t4[2],col="purple", pch=16, cex=2)
text(t4[1],t4[2],labels ="t4",cex = 1.2)
```




```r
# predict the class of the test data using the lda model
newtest_data <- data[c(3:68, 5:65, 3:28, 5:20, 3:57, 8:82, 4:64, 7:98), ]

pred <- predict(lda.model, newtest_data)
pred
```

```
## $class
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
## [223] 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [297] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
## [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
## [408] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [445] 1 1 1 1 1 1 1 1
## Levels: 0 1
##
## $posterior
##                  0            1
## 3     9.991927e-01 8.072668e-04
## 4     9.804294e-01 1.957057e-02
```

```
## 5    9.518831e-01 4.811686e-02
## 6    9.964156e-01 3.584367e-03
## 7    9.971295e-01 2.870472e-03
## 8    9.986456e-01 1.354374e-03
## 9    9.878153e-01 1.218473e-02
## 10   9.971319e-01 2.868067e-03
## 11   9.990020e-01 9.980381e-04
## 12   9.827112e-01 1.728880e-02
## 13   9.605444e-01 3.945555e-02
## 14   9.999908e-01 9.194121e-06
## 15   9.995909e-01 4.090975e-04
## 16   8.063786e-01 1.936214e-01
## 17   9.994790e-01 5.209654e-04
## 18   9.999194e-01 8.055505e-05
## 19   9.751771e-01 2.482291e-02
## 20   9.991191e-01 8.808756e-04
## 21   9.979199e-01 2.080106e-03
## 22   9.984107e-01 1.589328e-03
## 23   9.677983e-01 3.220174e-02
## 24   9.995961e-01 4.038854e-04
## 25   9.434174e-01 5.658257e-02
## 26   9.963602e-01 3.639755e-03
## 27   9.994334e-01 5.665716e-04
## 28   9.999937e-01 6.262065e-06
## 29   9.983229e-01 1.677129e-03
## 30   9.879557e-01 1.204429e-02
## 31   9.769528e-01 2.304725e-02
## 32   9.981923e-01 1.807670e-03
## 33   9.244037e-01 7.559627e-02
## 34   9.998792e-01 1.207671e-04
## 35   9.999995e-01 5.280199e-07
## 36   9.999470e-01 5.295695e-05
## 37   9.987739e-01 1.226124e-03
## 38   9.999814e-01 1.856705e-05
## 39   9.973018e-01 2.698220e-03
## 40   9.996347e-01 3.652771e-04
## 41   9.966040e-01 3.395971e-03
## 42   9.999362e-01 6.376142e-05
## 43   9.999981e-01 1.892327e-06
## 44   9.999977e-01 2.292551e-06
## 45   9.952042e-01 4.795844e-03
## 46   9.997702e-01 2.298320e-04
## 47   9.984681e-01 1.531885e-03
## 48   9.997836e-01 2.164272e-04
## 49   9.962378e-01 3.762155e-03
## 50   9.998325e-01 1.675075e-04
## 51   5.492056e-05 9.999451e-01
## 52   4.503960e-04 9.995496e-01
## 53   5.800830e-01 4.199170e-01
## 54   2.325244e-01 7.674756e-01
## 55   1.112905e-01 8.887095e-01
## 56   1.278394e-04 9.998722e-01
## 57   9.126567e-05 9.999087e-01
## 58   1.138506e-02 9.886149e-01
```

```
## 59   8.489721e-01 1.510279e-01
## 60   1.396862e-02 9.860314e-01
## 61   1.332980e-03 9.986670e-01
## 62   1.896257e-06 9.999981e-01
## 63   1.400763e-01 8.599237e-01
## 64   1.048886e-01 8.951114e-01
## 65   3.480363e-05 9.999652e-01
## 66   6.064572e-06 9.999939e-01
## 67   8.442101e-02 9.155790e-01
## 68   2.660356e-02 9.733964e-01
## 5.1  9.518831e-01 4.811686e-02
## 6.1  9.964156e-01 3.584367e-03
## 7.1  9.971295e-01 2.870472e-03
## 8.1  9.986456e-01 1.354374e-03
## 9.1  9.878153e-01 1.218473e-02
## 10.1 9.971319e-01 2.868067e-03
## 11.1 9.990020e-01 9.980381e-04
## 12.1 9.827112e-01 1.728880e-02
## 13.1 9.605444e-01 3.945555e-02
## 14.1 9.999908e-01 9.194121e-06
## 15.1 9.995909e-01 4.090975e-04
## 16.1 8.063786e-01 1.936214e-01
## 17.1 9.994790e-01 5.209654e-04
## 18.1 9.999194e-01 8.055505e-05
## 19.1 9.751771e-01 2.482291e-02
## 20.1 9.991191e-01 8.808756e-04
## 21.1 9.979199e-01 2.080106e-03
## 22.1 9.984107e-01 1.589328e-03
## 23.1 9.677983e-01 3.220174e-02
## 24.1 9.995961e-01 4.038854e-04
## 25.1 9.434174e-01 5.658257e-02
## 26.1 9.963602e-01 3.639755e-03
## 27.1 9.994334e-01 5.665716e-04
## 28.1 9.999937e-01 6.262065e-06
## 29.1 9.983229e-01 1.677129e-03
## 30.1 9.879557e-01 1.204429e-02
## 31.1 9.769528e-01 2.304725e-02
## 32.1 9.981923e-01 1.807670e-03
## 33.1 9.244037e-01 7.559627e-02
## 34.1 9.998792e-01 1.207671e-04
## 35.1 9.999995e-01 5.280199e-07
## 36.1 9.999470e-01 5.295695e-05
## 37.1 9.987739e-01 1.226124e-03
## 38.1 9.999814e-01 1.856705e-05
## 39.1 9.973018e-01 2.698220e-03
## 40.1 9.996347e-01 3.652771e-04
## 41.1 9.966040e-01 3.395971e-03
## 42.1 9.999362e-01 6.376142e-05
## 43.1 9.999981e-01 1.892327e-06
## 44.1 9.999977e-01 2.292551e-06
## 45.1 9.952042e-01 4.795844e-03
## 46.1 9.997702e-01 2.298320e-04
## 47.1 9.984681e-01 1.531885e-03
## 48.1 9.997836e-01 2.164272e-04
```

```
## 49.1 9.962378e-01 3.762155e-03
## 50.1 9.998325e-01 1.675075e-04
## 51.1 5.492056e-05 9.999451e-01
## 52.1 4.503960e-04 9.995496e-01
## 53.1 5.800830e-01 4.199170e-01
## 54.1 2.325244e-01 7.674756e-01
## 55.1 1.112905e-01 8.887095e-01
## 56.1 1.278394e-04 9.998722e-01
## 57.1 9.126567e-05 9.999087e-01
## 58.1 1.138506e-02 9.886149e-01
## 59.1 8.489721e-01 1.510279e-01
## 60.1 1.396862e-02 9.860314e-01
## 61.1 1.332980e-03 9.986670e-01
## 62.1 1.896257e-06 9.999981e-01
## 63.1 1.400763e-01 8.599237e-01
## 64.1 1.048886e-01 8.951114e-01
## 65.1 3.480363e-05 9.999652e-01
## 3.1  9.991927e-01 8.072668e-04
## 4.1  9.804294e-01 1.957057e-02
## 5.2  9.518831e-01 4.811686e-02
## 6.2  9.964156e-01 3.584367e-03
## 7.2  9.971295e-01 2.870472e-03
## 8.2  9.986456e-01 1.354374e-03
## 9.2  9.878153e-01 1.218473e-02
## 10.2 9.971319e-01 2.868067e-03
## 11.2 9.990020e-01 9.980381e-04
## 12.2 9.827112e-01 1.728880e-02
## 13.2 9.605444e-01 3.945555e-02
## 14.2 9.999908e-01 9.194121e-06
## 15.2 9.995909e-01 4.090975e-04
## 16.2 8.063786e-01 1.936214e-01
## 17.2 9.994790e-01 5.209654e-04
## 18.2 9.999194e-01 8.055505e-05
## 19.2 9.751771e-01 2.482291e-02
## 20.2 9.991191e-01 8.808756e-04
## 21.2 9.979199e-01 2.080106e-03
## 22.2 9.984107e-01 1.589328e-03
## 23.2 9.677983e-01 3.220174e-02
## 24.2 9.995961e-01 4.038854e-04
## 25.2 9.434174e-01 5.658257e-02
## 26.2 9.963602e-01 3.639755e-03
## 27.2 9.994334e-01 5.665716e-04
## 28.2 9.999937e-01 6.262065e-06
## 5.3  9.518831e-01 4.811686e-02
## 6.3  9.964156e-01 3.584367e-03
## 7.3  9.971295e-01 2.870472e-03
## 8.3  9.986456e-01 1.354374e-03
## 9.3  9.878153e-01 1.218473e-02
## 10.3 9.971319e-01 2.868067e-03
## 11.3 9.990020e-01 9.980381e-04
## 12.3 9.827112e-01 1.728880e-02
## 13.3 9.605444e-01 3.945555e-02
## 14.3 9.999908e-01 9.194121e-06
## 15.3 9.995909e-01 4.090975e-04
```

```
## 16.3 8.063786e-01 1.936214e-01
## 17.3 9.994790e-01 5.209654e-04
## 18.3 9.999194e-01 8.055505e-05
## 19.3 9.751771e-01 2.482291e-02
## 20.3 9.991191e-01 8.808756e-04
## 3.2  9.991927e-01 8.072668e-04
## 4.2  9.804294e-01 1.957057e-02
## 5.4  9.518831e-01 4.811686e-02
## 6.4  9.964156e-01 3.584367e-03
## 7.4  9.971295e-01 2.870472e-03
## 8.4  9.986456e-01 1.354374e-03
## 9.4  9.878153e-01 1.218473e-02
## 10.4 9.971319e-01 2.868067e-03
## 11.4 9.990020e-01 9.980381e-04
## 12.4 9.827112e-01 1.728880e-02
## 13.4 9.605444e-01 3.945555e-02
## 14.4 9.999908e-01 9.194121e-06
## 15.4 9.995909e-01 4.090975e-04
## 16.4 8.063786e-01 1.936214e-01
## 17.4 9.994790e-01 5.209654e-04
## 18.4 9.999194e-01 8.055505e-05
## 19.4 9.751771e-01 2.482291e-02
## 20.4 9.991191e-01 8.808756e-04
## 21.3 9.979199e-01 2.080106e-03
## 22.3 9.984107e-01 1.589328e-03
## 23.3 9.677983e-01 3.220174e-02
## 24.3 9.995961e-01 4.038854e-04
## 25.3 9.434174e-01 5.658257e-02
## 26.3 9.963602e-01 3.639755e-03
## 27.3 9.994334e-01 5.665716e-04
## 28.3 9.999937e-01 6.262065e-06
## 29.2 9.983229e-01 1.677129e-03
## 30.2 9.879557e-01 1.204429e-02
## 31.2 9.769528e-01 2.304725e-02
## 32.2 9.981923e-01 1.807670e-03
## 33.2 9.244037e-01 7.559627e-02
## 34.2 9.998792e-01 1.207671e-04
## 35.2 9.999995e-01 5.280199e-07
## 36.2 9.999470e-01 5.295695e-05
## 37.2 9.987739e-01 1.226124e-03
## 38.2 9.999814e-01 1.856705e-05
## 39.2 9.973018e-01 2.698220e-03
## 40.2 9.996347e-01 3.652771e-04
## 41.2 9.966040e-01 3.395971e-03
## 42.2 9.999362e-01 6.376142e-05
## 43.2 9.999981e-01 1.892327e-06
## 44.2 9.999977e-01 2.292551e-06
## 45.2 9.952042e-01 4.795844e-03
## 46.2 9.997702e-01 2.298320e-04
## 47.2 9.984681e-01 1.531885e-03
## 48.2 9.997836e-01 2.164272e-04
## 49.2 9.962378e-01 3.762155e-03
## 50.2 9.998325e-01 1.675075e-04
## 51.2 5.492056e-05 9.999451e-01
```

```
## 52.2 4.503960e-04 9.995496e-01
## 53.2 5.800830e-01 4.199170e-01
## 54.2 2.325244e-01 7.674756e-01
## 55.2 1.112905e-01 8.887095e-01
## 56.2 1.278394e-04 9.998722e-01
## 57.2 9.126567e-05 9.999087e-01
## 8.5  9.986456e-01 1.354374e-03
## 9.5  9.878153e-01 1.218473e-02
## 10.5 9.971319e-01 2.868067e-03
## 11.5 9.990020e-01 9.980381e-04
## 12.5 9.827112e-01 1.728880e-02
## 13.5 9.605444e-01 3.945555e-02
## 14.5 9.999908e-01 9.194121e-06
## 15.5 9.995909e-01 4.090975e-04
## 16.5 8.063786e-01 1.936214e-01
## 17.5 9.994790e-01 5.209654e-04
## 18.5 9.999194e-01 8.055505e-05
## 19.5 9.751771e-01 2.482291e-02
## 20.5 9.991191e-01 8.808756e-04
## 21.4 9.979199e-01 2.080106e-03
## 22.4 9.984107e-01 1.589328e-03
## 23.4 9.677983e-01 3.220174e-02
## 24.4 9.995961e-01 4.038854e-04
## 25.4 9.434174e-01 5.658257e-02
## 26.4 9.963602e-01 3.639755e-03
## 27.4 9.994334e-01 5.665716e-04
## 28.4 9.999937e-01 6.262065e-06
## 29.3 9.983229e-01 1.677129e-03
## 30.3 9.879557e-01 1.204429e-02
## 31.3 9.769528e-01 2.304725e-02
## 32.3 9.981923e-01 1.807670e-03
## 33.3 9.244037e-01 7.559627e-02
## 34.3 9.998792e-01 1.207671e-04
## 35.3 9.999995e-01 5.280199e-07
## 36.3 9.999470e-01 5.295695e-05
## 37.3 9.987739e-01 1.226124e-03
## 38.3 9.999814e-01 1.856705e-05
## 39.3 9.973018e-01 2.698220e-03
## 40.3 9.996347e-01 3.652771e-04
## 41.3 9.966040e-01 3.395971e-03
## 42.3 9.999362e-01 6.376142e-05
## 43.3 9.999981e-01 1.892327e-06
## 44.3 9.999977e-01 2.292551e-06
## 45.3 9.952042e-01 4.795844e-03
## 46.3 9.997702e-01 2.298320e-04
## 47.3 9.984681e-01 1.531885e-03
## 48.3 9.997836e-01 2.164272e-04
## 49.3 9.962378e-01 3.762155e-03
## 50.3 9.998325e-01 1.675075e-04
## 51.3 5.492056e-05 9.999451e-01
## 52.3 4.503960e-04 9.995496e-01
## 53.3 5.800830e-01 4.199170e-01
## 54.3 2.325244e-01 7.674756e-01
## 55.3 1.112905e-01 8.887095e-01
```

```
## 56.3 1.278394e-04 9.998722e-01
## 57.3 9.126567e-05 9.999087e-01
## 58.2 1.138506e-02 9.886149e-01
## 59.2 8.489721e-01 1.510279e-01
## 60.2 1.396862e-02 9.860314e-01
## 61.2 1.332980e-03 9.986670e-01
## 62.2 1.896257e-06 9.999981e-01
## 63.2 1.400763e-01 8.599237e-01
## 64.2 1.048886e-01 8.951114e-01
## 65.2 3.480363e-05 9.999652e-01
## 66.1 6.064572e-06 9.999939e-01
## 67.1 8.442101e-02 9.155790e-01
## 68.1 2.660356e-02 9.733964e-01
## 69   6.166978e-02 9.383302e-01
## 70   5.846903e-04 9.994153e-01
## 71   4.583761e-02 9.541624e-01
## 72   5.011863e-05 9.999499e-01
## 73   5.417165e-02 9.458283e-01
## 74   1.957420e-01 8.042580e-01
## 75   6.653467e-05 9.999335e-01
## 76   3.789804e-03 9.962102e-01
## 77   3.028651e-05 9.999697e-01
## 78   5.288207e-02 9.471179e-01
## 79   7.544289e-04 9.992456e-01
## 80   9.914455e-02 9.008554e-01
## 81   4.178732e-05 9.999582e-01
## 82   2.081709e-06 9.999979e-01
## 4.3  9.804294e-01 1.957057e-02
## 5.5  9.518831e-01 4.811686e-02
## 6.5  9.964156e-01 3.584367e-03
## 7.5  9.971295e-01 2.870472e-03
## 8.6  9.986456e-01 1.354374e-03
## 9.6  9.878153e-01 1.218473e-02
## 10.6 9.971319e-01 2.868067e-03
## 11.6 9.990020e-01 9.980381e-04
## 12.6 9.827112e-01 1.728880e-02
## 13.6 9.605444e-01 3.945555e-02
## 14.6 9.999908e-01 9.194121e-06
## 15.6 9.995909e-01 4.090975e-04
## 16.6 8.063786e-01 1.936214e-01
## 17.6 9.994790e-01 5.209654e-04
## 18.6 9.999194e-01 8.055505e-05
## 19.6 9.751771e-01 2.482291e-02
## 20.6 9.991191e-01 8.808756e-04
## 21.5 9.979199e-01 2.080106e-03
## 22.5 9.984107e-01 1.589328e-03
## 23.5 9.677983e-01 3.220174e-02
## 24.5 9.995961e-01 4.038854e-04
## 25.5 9.434174e-01 5.658257e-02
## 26.5 9.963602e-01 3.639755e-03
## 27.5 9.994334e-01 5.665716e-04
## 28.5 9.999937e-01 6.262065e-06
## 29.4 9.983229e-01 1.677129e-03
## 30.4 9.879557e-01 1.204429e-02
```

```
## 31.4 9.769528e-01 2.304725e-02
## 32.4 9.981923e-01 1.807670e-03
## 33.4 9.244037e-01 7.559627e-02
## 34.4 9.998792e-01 1.207671e-04
## 35.4 9.999995e-01 5.280199e-07
## 36.4 9.999470e-01 5.295695e-05
## 37.4 9.987739e-01 1.226124e-03
## 38.4 9.999814e-01 1.856705e-05
## 39.4 9.973018e-01 2.698220e-03
## 40.4 9.996347e-01 3.652771e-04
## 41.4 9.966040e-01 3.395971e-03
## 42.4 9.999362e-01 6.376142e-05
## 43.4 9.999981e-01 1.892327e-06
## 44.4 9.999977e-01 2.292551e-06
## 45.4 9.952042e-01 4.795844e-03
## 46.4 9.997702e-01 2.298320e-04
## 47.4 9.984681e-01 1.531885e-03
## 48.4 9.997836e-01 2.164272e-04
## 49.4 9.962378e-01 3.762155e-03
## 50.4 9.998325e-01 1.675075e-04
## 51.4 5.492056e-05 9.999451e-01
## 52.4 4.503960e-04 9.995496e-01
## 53.4 5.800830e-01 4.199170e-01
## 54.4 2.325244e-01 7.674756e-01
## 55.4 1.112905e-01 8.887095e-01
## 56.4 1.278394e-04 9.998722e-01
## 57.4 9.126567e-05 9.999087e-01
## 58.3 1.138506e-02 9.886149e-01
## 59.3 8.489721e-01 1.510279e-01
## 60.3 1.396862e-02 9.860314e-01
## 61.3 1.332980e-03 9.986670e-01
## 62.3 1.896257e-06 9.999981e-01
## 63.3 1.400763e-01 8.599237e-01
## 64.3 1.048886e-01 8.951114e-01
## 7.6  9.971295e-01 2.870472e-03
## 8.7  9.986456e-01 1.354374e-03
## 9.7  9.878153e-01 1.218473e-02
## 10.7 9.971319e-01 2.868067e-03
## 11.7 9.990020e-01 9.980381e-04
## 12.7 9.827112e-01 1.728880e-02
## 13.7 9.605444e-01 3.945555e-02
## 14.7 9.999908e-01 9.194121e-06
## 15.7 9.995909e-01 4.090975e-04
## 16.7 8.063786e-01 1.936214e-01
## 17.7 9.994790e-01 5.209654e-04
## 18.7 9.999194e-01 8.055505e-05
## 19.7 9.751771e-01 2.482291e-02
## 20.7 9.991191e-01 8.808756e-04
## 21.6 9.979199e-01 2.080106e-03
## 22.6 9.984107e-01 1.589328e-03
## 23.6 9.677983e-01 3.220174e-02
## 24.6 9.995961e-01 4.038854e-04
## 25.6 9.434174e-01 5.658257e-02
## 26.6 9.963602e-01 3.639755e-03
```

```
## 27.6 9.994334e-01 5.665716e-04
## 28.6 9.999937e-01 6.262065e-06
## 29.5 9.983229e-01 1.677129e-03
## 30.5 9.879557e-01 1.204429e-02
## 31.5 9.769528e-01 2.304725e-02
## 32.5 9.981923e-01 1.807670e-03
## 33.5 9.244037e-01 7.559627e-02
## 34.5 9.998792e-01 1.207671e-04
## 35.5 9.999995e-01 5.280199e-07
## 36.5 9.999470e-01 5.295695e-05
## 37.5 9.987739e-01 1.226124e-03
## 38.5 9.999814e-01 1.856705e-05
## 39.5 9.973018e-01 2.698220e-03
## 40.5 9.996347e-01 3.652771e-04
## 41.5 9.966040e-01 3.395971e-03
## 42.5 9.999362e-01 6.376142e-05
## 43.5 9.999981e-01 1.892327e-06
## 44.5 9.999977e-01 2.292551e-06
## 45.5 9.952042e-01 4.795844e-03
## 46.5 9.997702e-01 2.298320e-04
## 47.5 9.984681e-01 1.531885e-03
## 48.5 9.997836e-01 2.164272e-04
## 49.5 9.962378e-01 3.762155e-03
## 50.5 9.998325e-01 1.675075e-04
## 51.5 5.492056e-05 9.999451e-01
## 52.5 4.503960e-04 9.995496e-01
## 53.5 5.800830e-01 4.199170e-01
## 54.5 2.325244e-01 7.674756e-01
## 55.5 1.112905e-01 8.887095e-01
## 56.5 1.278394e-04 9.998722e-01
## 57.5 9.126567e-05 9.999087e-01
## 58.4 1.138506e-02 9.886149e-01
## 59.4 8.489721e-01 1.510279e-01
## 60.4 1.396862e-02 9.860314e-01
## 61.4 1.332980e-03 9.986670e-01
## 62.4 1.896257e-06 9.999981e-01
## 63.4 1.400763e-01 8.599237e-01
## 64.4 1.048886e-01 8.951114e-01
## 65.3 3.480363e-05 9.999652e-01
## 66.2 6.064572e-06 9.999939e-01
## 67.2 8.442101e-02 9.155790e-01
## 68.2 2.660356e-02 9.733964e-01
## 69.1 6.166978e-02 9.383302e-01
## 70.1 5.846903e-04 9.994153e-01
## 71.1 4.583761e-02 9.541624e-01
## 72.1 5.011863e-05 9.999499e-01
## 73.1 5.417165e-02 9.458283e-01
## 74.1 1.957420e-01 8.042580e-01
## 75.1 6.653467e-05 9.999335e-01
## 76.1 3.789804e-03 9.962102e-01
## 77.1 3.028651e-05 9.999697e-01
## 78.1 5.288207e-02 9.471179e-01
## 79.1 7.544289e-04 9.992456e-01
## 80.1 9.914455e-02 9.008554e-01
```

```
## 81.1 4.178732e-05 9.999582e-01
## 82.1 2.081709e-06 9.999979e-01
## 83   6.022489e-05 9.999398e-01
## 84   1.374952e-05 9.999863e-01
## 85   3.922186e-05 9.999608e-01
## 86   3.691934e-06 9.999963e-01
## 87   4.048648e-05 9.999595e-01
## 88   2.990388e-01 7.009612e-01
## 89   1.668043e-02 9.833196e-01
## 90   4.414227e-02 9.558577e-01
## 91   3.155524e-03 9.968445e-01
## 92   4.088046e-03 9.959120e-01
## 93   1.850859e-06 9.999981e-01
## 94   7.188667e-06 9.999928e-01
## 95   1.357230e-04 9.998643e-01
## 96   9.856021e-05 9.999014e-01
## 97   1.776370e-02 9.822363e-01
## 98   5.674374e-07 9.999994e-01
##
## $x
##               LD1
## 3    -1.93992096
## 4    -1.06624473
## 5    -0.81312389
## 6    -1.53306988
## 7    -1.59377108
## 8    -1.79881019
## 9    -1.19737404
## 10   -1.59400006
## 11   -1.88207836
## 12   -1.10064934
## 13   -0.86965547
## 14   -3.15924554
## 15   -2.12519405
## 16   -0.38864854
## 17   -2.05931125
## 18   -2.56797064
## 19   -1.00001599
## 20   -1.91612888
## 21   -1.68172215
## 22   -1.75516659
## 23   -0.92704844
## 24   -2.12868854
## 25   -0.76653945
## 26   -1.52887734
## 27   -2.03643733
## 28   -3.26387066
## 29   -1.74049388
## 30   -1.20057087
## 31   -1.02073089
## 32   -1.72003887
## 33   -0.68207111
## 34   -2.45765027
## 35   -3.93760431
```

```
## 36   -2.68224815
## 37   -1.82594616
## 38   -2.96777918
## 39   -1.61067668
## 40   -2.15607053
## 41   -1.54782997
## 42   -2.63166511
## 43   -3.58987845
## 44   -3.53761226
## 45   -1.45341831
## 46   -2.28232202
## 47   -1.76521062
## 48   -2.29869653
## 49   -1.51983341
## 50   -2.36851127
## 51    2.67232915
## 52    2.09898309
## 53   -0.08802295
## 54    0.32530072
## 55    0.56598838
## 56    2.44214469
## 57    2.53396036
## 58    1.21608668
## 59   -0.47035112
## 60    1.15966067
## 61    1.80315361
## 62    3.58931326
## 63    0.49434987
## 64    0.58408346
## 65    2.79660377
## 66    3.27260070
## 67    0.64938018
## 68    0.98064532
## 5.1  -0.81312389
## 6.1  -1.53306988
## 7.1  -1.59377108
## 8.1  -1.79881019
## 9.1  -1.19737404
## 10.1 -1.59400006
## 11.1 -1.88207836
## 12.1 -1.10064934
## 13.1 -0.86965547
## 14.1 -3.15924554
## 15.1 -2.12519405
## 16.1 -0.38864854
## 17.1 -2.05931125
## 18.1 -2.56797064
## 19.1 -1.00001599
## 20.1 -1.91612888
## 21.1 -1.68172215
## 22.1 -1.75516659
## 23.1 -0.92704844
## 24.1 -2.12868854
## 25.1 -0.76653945
```

```
## 26.1 -1.52887734
## 27.1 -2.03643733
## 28.1 -3.26387066
## 29.1 -1.74049388
## 30.1 -1.20057087
## 31.1 -1.02073089
## 32.1 -1.72003887
## 33.1 -0.68207111
## 34.1 -2.45765027
## 35.1 -3.93760431
## 36.1 -2.68224815
## 37.1 -1.82594616
## 38.1 -2.96777918
## 39.1 -1.61067668
## 40.1 -2.15607053
## 41.1 -1.54782997
## 42.1 -2.63166511
## 43.1 -3.58987845
## 44.1 -3.53761226
## 45.1 -1.45341831
## 46.1 -2.28232202
## 47.1 -1.76521062
## 48.1 -2.29869653
## 49.1 -1.51983341
## 50.1 -2.36851127
## 51.1  2.67232915
## 52.1  2.09898309
## 53.1 -0.08802295
## 54.1  0.32530072
## 55.1  0.56598838
## 56.1  2.44214469
## 57.1  2.53396036
## 58.1  1.21608668
## 59.1 -0.47035112
## 60.1  1.15966067
## 61.1  1.80315361
## 62.1  3.58931326
## 63.1  0.49434987
## 64.1  0.58408346
## 65.1  2.79660377
## 3.1  -1.93992096
## 4.1  -1.06624473
## 5.2  -0.81312389
## 6.2  -1.53306988
## 7.2  -1.59377108
## 8.2  -1.79881019
## 9.2  -1.19737404
## 10.2 -1.59400006
## 11.2 -1.88207836
## 12.2 -1.10064934
## 13.2 -0.86965547
## 14.2 -3.15924554
## 15.2 -2.12519405
## 16.2 -0.38864854
```

```
## 17.2 -2.05931125
## 18.2 -2.56797064
## 19.2 -1.00001599
## 20.2 -1.91612888
## 21.2 -1.68172215
## 22.2 -1.75516659
## 23.2 -0.92704844
## 24.2 -2.12868854
## 25.2 -0.76653945
## 26.2 -1.52887734
## 27.2 -2.03643733
## 28.2 -3.26387066
## 5.3  -0.81312389
## 6.3  -1.53306988
## 7.3  -1.59377108
## 8.3  -1.79881019
## 9.3  -1.19737404
## 10.3 -1.59400006
## 11.3 -1.88207836
## 12.3 -1.10064934
## 13.3 -0.86965547
## 14.3 -3.15924554
## 15.3 -2.12519405
## 16.3 -0.38864854
## 17.3 -2.05931125
## 18.3 -2.56797064
## 19.3 -1.00001599
## 20.3 -1.91612888
## 3.2  -1.93992096
## 4.2  -1.06624473
## 5.4  -0.81312389
## 6.4  -1.53306988
## 7.4  -1.59377108
## 8.4  -1.79881019
## 9.4  -1.19737404
## 10.4 -1.59400006
## 11.4 -1.88207836
## 12.4 -1.10064934
## 13.4 -0.86965547
## 14.4 -3.15924554
## 15.4 -2.12519405
## 16.4 -0.38864854
## 17.4 -2.05931125
## 18.4 -2.56797064
## 19.4 -1.00001599
## 20.4 -1.91612888
## 21.3 -1.68172215
## 22.3 -1.75516659
## 23.3 -0.92704844
## 24.3 -2.12868854
## 25.3 -0.76653945
## 26.3 -1.52887734
## 27.3 -2.03643733
## 28.3 -3.26387066
```

```
## 29.2 -1.74049388
## 30.2 -1.20057087
## 31.2 -1.02073089
## 32.2 -1.72003887
## 33.2 -0.68207111
## 34.2 -2.45765027
## 35.2 -3.93760431
## 36.2 -2.68224815
## 37.2 -1.82594616
## 38.2 -2.96777918
## 39.2 -1.61067668
## 40.2 -2.15607053
## 41.2 -1.54782997
## 42.2 -2.63166511
## 43.2 -3.58987845
## 44.2 -3.53761226
## 45.2 -1.45341831
## 46.2 -2.28232202
## 47.2 -1.76521062
## 48.2 -2.29869653
## 49.2 -1.51983341
## 50.2 -2.36851127
## 51.2  2.67232915
## 52.2  2.09898309
## 53.2 -0.08802295
## 54.2  0.32530072
## 55.2  0.56598838
## 56.2  2.44214469
## 57.2  2.53396036
## 8.5  -1.79881019
## 9.5  -1.19737404
## 10.5 -1.59400006
## 11.5 -1.88207836
## 12.5 -1.10064934
## 13.5 -0.86965547
## 14.5 -3.15924554
## 15.5 -2.12519405
## 16.5 -0.38864854
## 17.5 -2.05931125
## 18.5 -2.56797064
## 19.5 -1.00001599
## 20.5 -1.91612888
## 21.4 -1.68172215
## 22.4 -1.75516659
## 23.4 -0.92704844
## 24.4 -2.12868854
## 25.4 -0.76653945
## 26.4 -1.52887734
## 27.4 -2.03643733
## 28.4 -3.26387066
## 29.3 -1.74049388
## 30.3 -1.20057087
## 31.3 -1.02073089
## 32.3 -1.72003887
```

```
## 33.3 -0.68207111
## 34.3 -2.45765027
## 35.3 -3.93760431
## 36.3 -2.68224815
## 37.3 -1.82594616
## 38.3 -2.96777918
## 39.3 -1.61067668
## 40.3 -2.15607053
## 41.3 -1.54782997
## 42.3 -2.63166511
## 43.3 -3.58987845
## 44.3 -3.53761226
## 45.3 -1.45341831
## 46.3 -2.28232202
## 47.3 -1.76521062
## 48.3 -2.29869653
## 49.3 -1.51983341
## 50.3 -2.36851127
## 51.3  2.67232915
## 52.3  2.09898309
## 53.3 -0.08802295
## 54.3  0.32530072
## 55.3  0.56598838
## 56.3  2.44214469
## 57.3  2.53396036
## 58.2  1.21608668
## 59.2 -0.47035112
## 60.2  1.15966067
## 61.2  1.80315361
## 62.2  3.58931326
## 63.2  0.49434987
## 64.2  0.58408346
## 65.2  2.79660377
## 66.1  3.27260070
## 67.1  0.64938018
## 68.1  0.98064532
## 69    0.74161297
## 70    2.02785689
## 71    0.82699535
## 72    2.69725562
## 73    0.77909682
## 74    0.38496382
## 75    2.62006609
## 76    1.51783109
## 77    2.83447667
## 78    0.78603156
## 79    1.95837638
## 80    0.60116867
## 81    2.74678392
## 82    3.56389434
## 4.3  -1.06624473
## 5.5  -0.81312389
## 6.5  -1.53306988
## 7.5  -1.59377108
```

```
## 8.6  -1.79881019
## 9.6  -1.19737404
## 10.6 -1.59400006
## 11.6 -1.88207836
## 12.6 -1.10064934
## 13.6 -0.86965547
## 14.6 -3.15924554
## 15.6 -2.12519405
## 16.6 -0.38864854
## 17.6 -2.05931125
## 18.6 -2.56797064
## 19.6 -1.00001599
## 20.6 -1.91612888
## 21.5 -1.68172215
## 22.5 -1.75516659
## 23.5 -0.92704844
## 24.5 -2.12868854
## 25.5 -0.76653945
## 26.5 -1.52887734
## 27.5 -2.03643733
## 28.5 -3.26387066
## 29.4 -1.74049388
## 30.4 -1.20057087
## 31.4 -1.02073089
## 32.4 -1.72003887
## 33.4 -0.68207111
## 34.4 -2.45765027
## 35.4 -3.93760431
## 36.4 -2.68224815
## 37.4 -1.82594616
## 38.4 -2.96777918
## 39.4 -1.61067668
## 40.4 -2.15607053
## 41.4 -1.54782997
## 42.4 -2.63166511
## 43.4 -3.58987845
## 44.4 -3.53761226
## 45.4 -1.45341831
## 46.4 -2.28232202
## 47.4 -1.76521062
## 48.4 -2.29869653
## 49.4 -1.51983341
## 50.4 -2.36851127
## 51.4  2.67232915
## 52.4  2.09898309
## 53.4 -0.08802295
## 54.4  0.32530072
## 55.4  0.56598838
## 56.4  2.44214469
## 57.4  2.53396036
## 58.3  1.21608668
## 59.3 -0.47035112
## 60.3  1.15966067
## 61.3  1.80315361
```

```
## 62.3  3.58931326
## 63.3  0.49434987
## 64.3  0.58408346
## 7.6  -1.59377108
## 8.7  -1.79881019
## 9.7  -1.19737404
## 10.7 -1.59400006
## 11.7 -1.88207836
## 12.7 -1.10064934
## 13.7 -0.86965547
## 14.7 -3.15924554
## 15.7 -2.12519405
## 16.7 -0.38864854
## 17.7 -2.05931125
## 18.7 -2.56797064
## 19.7 -1.00001599
## 20.7 -1.91612888
## 21.6 -1.68172215
## 22.6 -1.75516659
## 23.6 -0.92704844
## 24.6 -2.12868854
## 25.6 -0.76653945
## 26.6 -1.52887734
## 27.6 -2.03643733
## 28.6 -3.26387066
## 29.5 -1.74049388
## 30.5 -1.20057087
## 31.5 -1.02073089
## 32.5 -1.72003887
## 33.5 -0.68207111
## 34.5 -2.45765027
## 35.5 -3.93760431
## 36.5 -2.68224815
## 37.5 -1.82594616
## 38.5 -2.96777918
## 39.5 -1.61067668
## 40.5 -2.15607053
## 41.5 -1.54782997
## 42.5 -2.63166511
## 43.5 -3.58987845
## 44.5 -3.53761226
## 45.5 -1.45341831
## 46.5 -2.28232202
## 47.5 -1.76521062
## 48.5 -2.29869653
## 49.5 -1.51983341
## 50.5 -2.36851127
## 51.5  2.67232915
## 52.5  2.09898309
## 53.5 -0.08802295
## 54.5  0.32530072
## 55.5  0.56598838
## 56.5  2.44214469
## 57.5  2.53396036
```

```
## 58.4  1.21608668
## 59.4 -0.47035112
## 60.4  1.15966067
## 61.4  1.80315361
## 62.4  3.58931326
## 63.4  0.49434987
## 64.4  0.58408346
## 65.3  2.79660377
## 66.2  3.27260070
## 67.2  0.64938018
## 68.2  0.98064532
## 69.1  0.74161297
## 70.1  2.02785689
## 71.1  0.82699535
## 72.1  2.69725562
## 73.1  0.77909682
## 74.1  0.38496382
## 75.1  2.62006609
## 76.1  1.51783109
## 77.1  2.83447667
## 78.1  0.78603156
## 79.1  1.95837638
## 80.1  0.60116867
## 81.1  2.74678392
## 82.1  3.56389434
## 83    2.64721109
## 84    3.04961134
## 85    2.76404485
## 86    3.40780743
## 87    2.75539954
## 88    0.23206946
## 89    1.11057692
## 90    0.83774571
## 91    1.56790093
## 92    1.49711284
## 93    3.59591451
## 94    3.22627748
## 95    2.42584048
## 96    2.51301117
## 97    1.09313569
## 98    3.91799100
```

Question 3

This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

(e) Repeat (d) using LDA and QDA. Interpret the results.

(f) Which of these methods appears to provide the best results on this data?

Question 3(a)

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
names(Weekly)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

###Summary of weekly data

```
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
##
##
##
##
```

###Drop last column

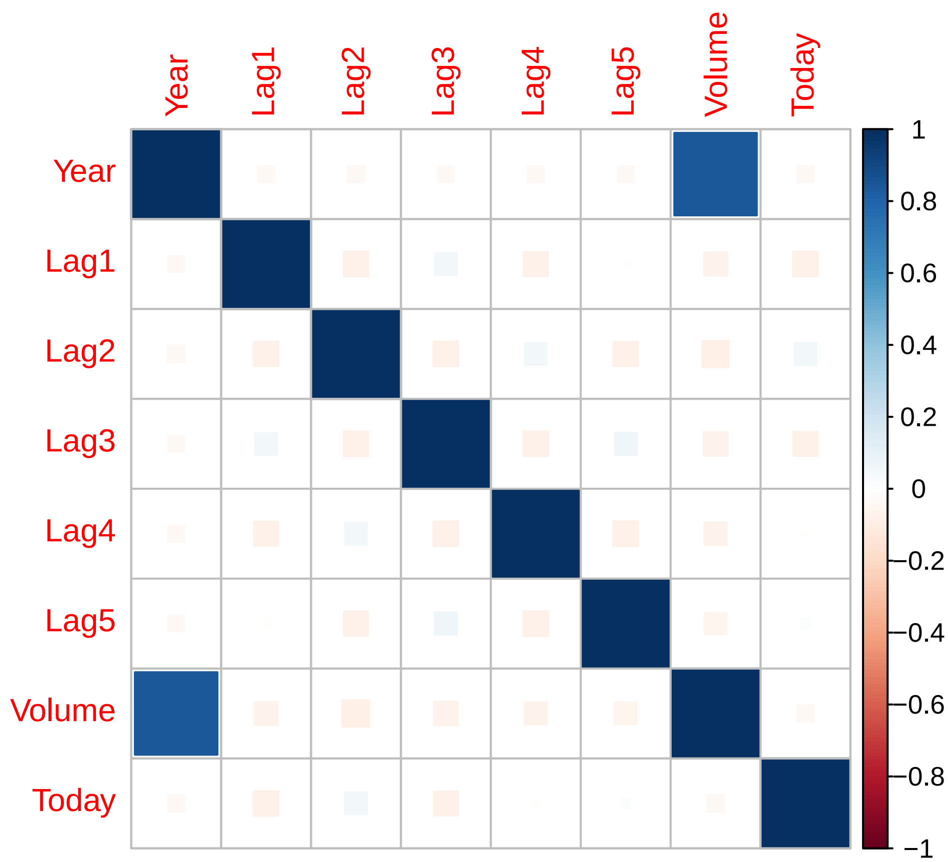
```
cor(Weekly[,-9])
```

```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
```

```
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##               Lag5      Volume         Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

###Find correlation matrix

```
corrplot(cor(Weekly[,-9]), method = "square")
```



The correlational plot doesn't illustrate that any other variables are linearly related except volume and Year

Question 3(b)

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

###Logistic Regreesion with full Datasets

```
attach(Weekly)
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data = Weekly, family = binomial )
glm.fit
```

```
##
```

```
## Call:  glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Coefficients:
## (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
##     0.26686     -0.04127      0.05844     -0.01606     -0.02779     -0.01447
##      Volume
##    -0.02274
##
## Degrees of Freedom: 1088 Total (i.e. Null);  1082 Residual
## Null Deviance:       1496
## Residual Deviance: 1486  AIC: 1500
```

#Summary

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

#Do any of the predictors appear to be statistically significant? If so, which ones?

The only variable that is statistically significant at the level of significance

$$\alpha = 0.05$$

is Lag2. Otherwise the other variables fail to reject the null hypothesis

$$\beta = 0$$

Question 3(c)

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

###Compute the confusion matrix and overall fraction of correct predictions.

```
coef(glm.fit)
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##  0.26686414 -0.04126894  0.05844168 -0.01606114 -0.02779021 -0.01447206
##      Volume
## -0.02274153
```

#Predict function to see the value goes up or down

```
glm_prob <- predict(glm.fit, type = "response")
glm_prob[1:10]
```

```
##         1         2         3         4         5         6         7         8
## 0.6086249 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190 0.5786097 0.5151972
##         9        10
## 0.5715200 0.5554287
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

#The following two commands create a vector of class predictions based on whether the predicted probability of a market increase is greater than or less than 0.5.

```
glm_pred <- rep("Down", length(glm_prob))
glm_pred[glm_prob > 0.5] = "Up"
```

#Given these predictions, the table() function table() can be used to produce a confusion matrix in order to determine how many observations were correctly or incorrectly classified.

```
table(glm_pred, Direction)
```

```
##         Direction
## glm_pred Down  Up
##     Down   54  48
##     Up    430 557
```

The diagonal elements of confusion matrix indicates that the correct predictions and off diagonal elements are incorrect predictions. Hence our model correctly predicted that the market would go up on 557 days and that it would go down on 54 days, for a total of 557 + 54 = 611 correct predictions.

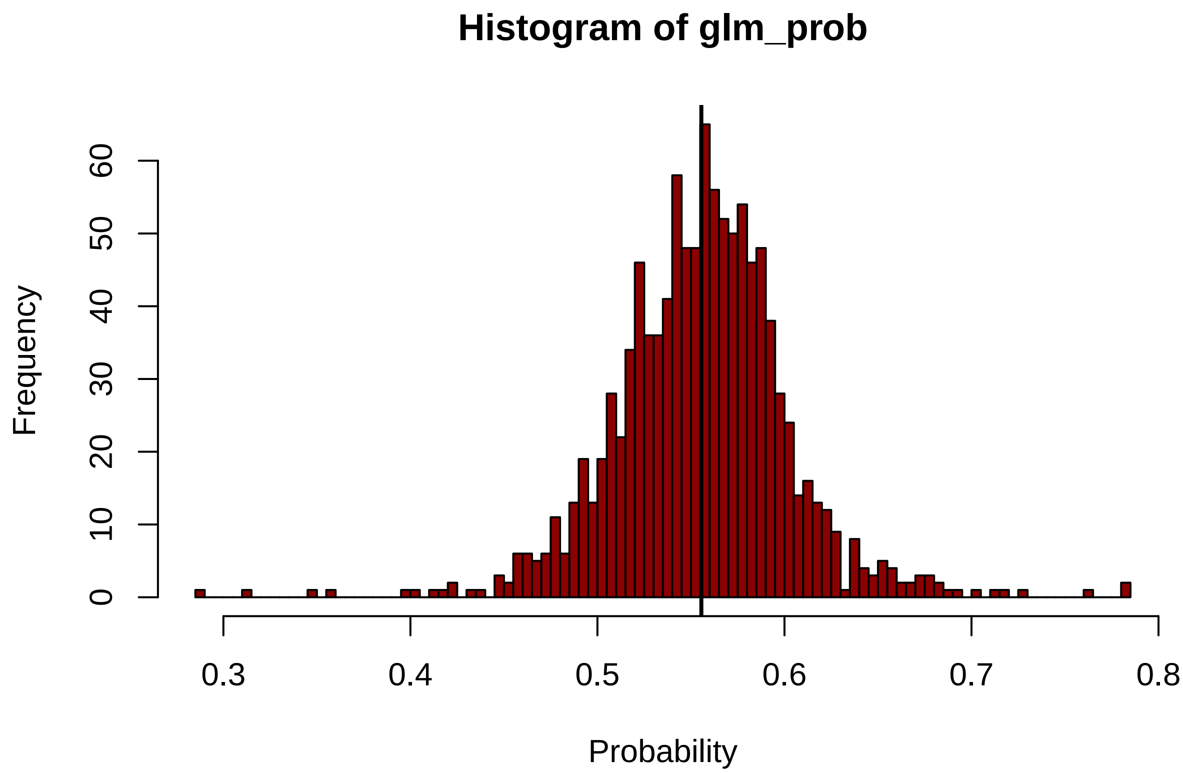
#mean to compute fraction of days which are corrected(percentage of correct predictions)

```
mean(glm_pred == Direction)
```

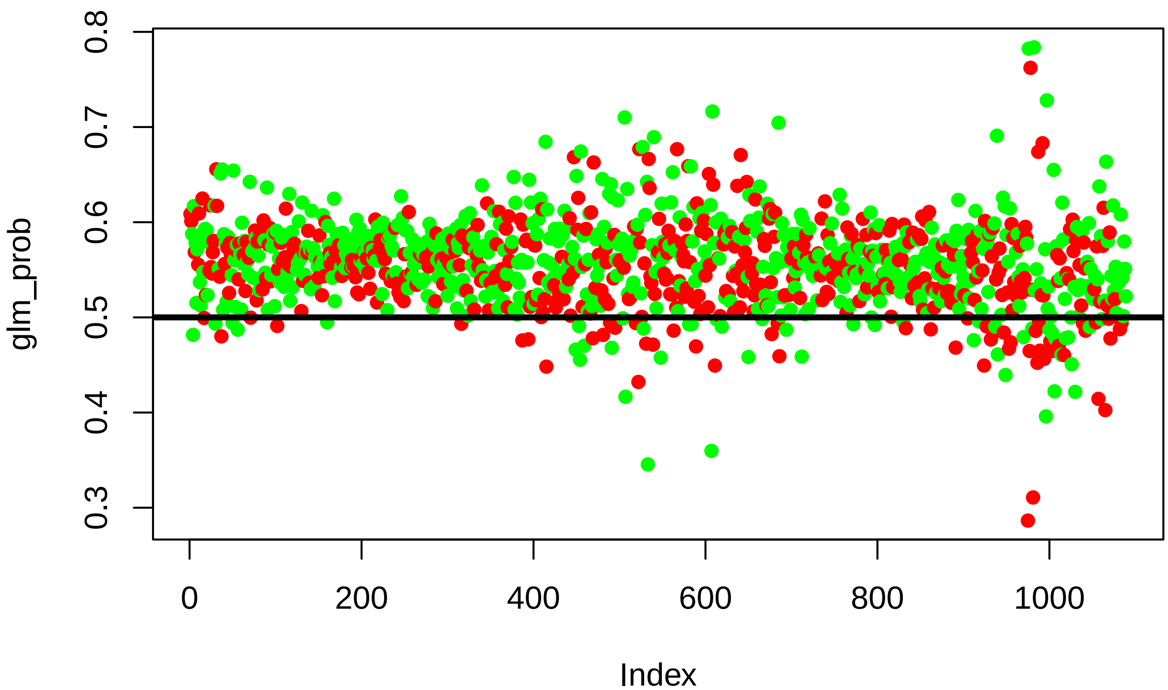
```
## [1] 0.5610652
```

#Plot Meanline

```
hist(glm_prob, breaks = 100, col = "darkred", xlab = "Probability")
abline(v = mean(glm_prob), lwd = 2)
```




```r
plot(glm_prob, col = ifelse(Weekly$Direction == "Down","red","green"), pch = 16)
abline(h = 0.5, lwd= 3)
```



The model's accuracy in predicting the weekly market trend was 56.11%. However, when looking at the two directions of the trend, the model was much better at predicting Up trends with an accuracy of 557/48+557 = 0.9207; 92.07%, compared to only 54/430+54 = 0.1115 ; 11.15% accuracy in predicting Down trends.

#Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
length(glm_prob)
```

```
## [1] 1089
```

The initial assessment of the logistic regression model based on 1089 observations may be misleading as the model was trained and tested on the same set of data. Although the model showed an accuracy of 56.11%, this represents the training error rate, which tends to underestimate the model's error rate on new data. To obtain a more realistic estimate of the model's performance on new, unseen data, it is better to train the model on a portion of the data and test it on a separate set of data that has not been used for training. This approach will provide a better evaluation of the model's ability to predict future market trends, which is of more practical interest.

Question 3(d)

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

#Fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor.

#Train

```r
train <- (Year<2009)
weekly.train <-  Weekly[!train,]
weekly.train
```

```
##      Year   Lag1   Lag2   Lag3   Lag4   Lag5   Volume  Today Direction
## 986  2009  6.760 -1.698  0.926  0.418 -2.251 3.793110 -4.448      Down
## 987  2009 -4.448  6.760 -1.698  0.926  0.418 5.043904 -4.518      Down
## 988  2009 -4.518 -4.448  6.760 -1.698  0.926 5.948758 -2.137      Down
## 989  2009 -2.137 -4.518 -4.448  6.760 -1.698 6.129763 -0.730      Down
## 990  2009 -0.730 -2.137 -4.518 -4.448  6.760 5.602004  5.173        Up
## 991  2009  5.173 -0.730 -2.137 -4.518 -4.448 6.217632 -4.808      Down
## 992  2009 -4.808  5.173 -0.730 -2.137 -4.518 6.008822 -6.868      Down
## 993  2009 -6.868 -4.808  5.173 -0.730 -2.137 6.401515 -4.540      Down
## 994  2009 -4.540 -6.868 -4.808  5.173 -0.730 7.550776 -7.035      Down
## 995  2009 -7.035 -4.540 -6.868 -4.808  5.173 7.592844 10.707        Up
## 996  2009 10.707 -7.035 -4.540 -6.868 -4.808 7.459436  1.585        Up
## 997  2009  1.585 10.707 -7.035 -4.540 -6.868 7.963276  6.168        Up
## 998  2009  6.168  1.585 10.707 -7.035 -4.540 6.952820  3.255        Up
## 999  2009  3.255  6.168  1.585 10.707 -7.035 6.286870  1.669        Up
## 1000 2009  1.669  3.255  6.168  1.585 10.707 6.226188  1.522        Up
## 1001 2009  1.522  1.669  3.255  6.168  1.585 6.839302 -0.388      Down
## 1002 2009 -0.388  1.522  1.669  3.255  6.168 7.083170  1.303        Up
## 1003 2009  1.303 -0.388  1.522  1.669  3.255 6.043558  5.893        Up
## 1004 2009  5.893  1.303 -0.388  1.522  1.669 7.952024 -4.988      Down
## 1005 2009 -4.988  5.893  1.303 -0.388  1.522 6.337752  0.467        Up
## 1006 2009  0.467 -4.988  5.893  1.303 -0.388 6.339728  3.623        Up
## 1007 2009  3.623  0.467 -4.988  5.893  1.303 5.788812  2.279        Up
## 1008 2009  2.279  3.623  0.467 -4.988  5.893 5.662470  0.651        Up
## 1009 2009  0.651  2.279  3.623  0.467 -4.988 4.866352 -2.640      Down
## 1010 2009 -2.640  0.651  2.279  3.623  0.467 5.114026 -0.253      Down
## 1011 2009 -0.253 -2.640  0.651  2.279  3.623 5.119916 -2.446      Down
## 1012 2009 -2.446 -0.253 -2.640  0.651  2.279 4.172433 -1.929      Down
## 1013 2009 -1.929 -2.446 -0.253 -2.640  0.651 4.673382  6.967        Up
## 1014 2009  6.967 -1.929 -2.446 -0.253 -2.640 4.785464  4.134        Up
## 1015 2009  4.134  6.967 -1.929 -2.446 -0.253 5.003300  0.839        Up
## 1016 2009  0.839  4.134  6.967 -1.929 -2.446 5.294932  2.329        Up
## 1017 2009  2.329  0.839  4.134  6.967 -1.929 6.427946 -0.632      Down
## 1018 2009 -0.632  2.329  0.839  4.134  6.967 5.373764  2.195        Up
## 1019 2009  2.195 -0.632  2.329  0.839  4.134 4.664650  0.273        Up
## 1020 2009  0.273  2.195 -0.632  2.329  0.839 5.744582 -1.218      Down
## 1021 2009 -1.218  0.273  2.195 -0.632  2.329 5.286260  2.591        Up
## 1022 2009  2.591 -1.218  0.273  2.195 -0.632 5.137923  2.452        Up
## 1023 2009  2.452  2.591 -1.218  0.273  2.195 6.046968 -2.239      Down
## 1024 2009 -2.239  2.452  2.591 -1.218  0.273 5.081302 -1.836      Down
## 1025 2009 -1.836 -2.239  2.452  2.591 -1.218 5.210080  4.514        Up
## 1026 2009  4.514 -1.836 -2.239  2.452  2.591 4.466710  1.511        Up
## 1027 2009  1.511  4.514 -1.836 -2.239  2.452 4.740370 -0.743      Down
## 1028 2009 -0.743  1.511  4.514 -1.836 -2.239 5.118466 -4.021      Down
## 1029 2009 -4.021 -0.743  1.511  4.514 -1.836 6.081714  3.195        Up
## 1030 2009  3.195 -4.021 -0.743  1.511  4.514 5.290226  2.261        Up
## 1031 2009  2.261  3.195 -4.021 -0.743  1.511 4.218872 -0.192      Down
## 1032 2009 -0.192  2.261  3.195 -4.021 -0.743 4.122504  0.010        Up
## 1033 2009  0.010 -0.192  2.261  3.195 -4.021 3.232000  1.328        Up
## 1034 2009  1.328  0.010 -0.192  2.261  3.195 4.535468  0.039        Up
```

```
## 1035 2009  0.039  1.328  0.010 -0.192  2.261 4.150876 -0.356      Down
## 1036 2009 -0.356  0.039  1.328  0.010 -0.192 5.672874  2.178        Up
## 1037 2009  2.178 -0.356  0.039  1.328  0.010 3.013263 -1.010      Down
## 1038 2010 -1.010  2.178 -0.356  0.039  1.328 2.390427  2.680        Up
## 1039 2010  2.680 -1.010  2.178 -0.356  0.039 4.223070 -0.782      Down
## 1040 2010 -0.782  2.680 -1.010  2.178 -0.356 4.363246 -3.897      Down
## 1041 2010 -3.897 -0.782  2.680 -1.010  2.178 5.654582 -1.639      Down
## 1042 2010 -1.639 -3.897 -0.782  2.680 -1.010 5.079534 -0.715      Down
## 1043 2010 -0.715 -1.639 -3.897 -0.782  2.680 5.082238  0.874        Up
## 1044 2010  0.874 -0.715 -1.639 -3.897 -0.782 4.403416  3.130        Up
## 1045 2010  3.130  0.874 -0.715 -1.639 -3.897 4.040725 -0.422      Down
## 1046 2010 -0.422  3.130  0.874 -0.715 -1.639 4.194034  3.097        Up
## 1047 2010  3.097 -0.422  3.130  0.874 -0.715 4.002330  0.991        Up
## 1048 2010  0.991  3.097 -0.422  3.130  0.874 4.805318  0.862        Up
## 1049 2010  0.862  0.991  3.097 -0.422  3.130 4.588800  0.577        Up
## 1050 2010  0.577  0.862  0.991  3.097 -0.422 4.751278  0.987        Up
## 1051 2010  0.987  0.577  0.862  0.991  3.097 4.237947  1.381        Up
## 1052 2010  1.381  0.987  0.577  0.862  0.991 4.461554 -0.188      Down
## 1053 2010 -0.188  1.381  0.987  0.577  0.862 5.974902  2.110        Up
## 1054 2010  2.110 -0.188  1.381  0.987  0.577 5.800096 -2.513      Down
## 1055 2010 -2.513  2.110 -0.188  1.381  0.987 6.310456 -6.388      Down
## 1056 2010 -6.388 -2.513  2.110 -0.188  1.381 7.683886  2.232        Up
## 1057 2010  2.232 -6.388 -2.513  2.110 -0.188 5.791750 -4.226      Down
## 1058 2010 -4.226  2.232 -6.388 -2.513  2.110 6.528052  0.158        Up
## 1059 2010  0.158 -4.226  2.232 -6.388 -2.513 5.528868 -2.252      Down
## 1060 2010 -2.252  0.158 -4.226  2.232 -6.388 5.368597  2.509        Up
## 1061 2010  2.509 -2.252  0.158 -4.226  2.232 5.369514  2.374        Up
## 1062 2010  2.374  2.509 -2.252  0.158 -4.226 4.637208 -3.646      Down
## 1063 2010 -3.646  2.374  2.509 -2.252  0.158 4.699712 -5.032      Down
## 1064 2010 -5.032 -3.646  2.374  2.509 -2.252 5.100892  5.416        Up
## 1065 2010  5.416 -5.032 -3.646  2.374  2.509 4.419372 -1.213      Down
## 1066 2010 -1.213  5.416 -5.032 -3.646  2.374 4.487664  3.548        Up
## 1067 2010  3.548 -1.213  5.416 -5.032 -3.646 4.580286 -0.096      Down
## 1068 2010 -0.096  3.548 -1.213  5.416 -5.032 4.271320  1.819        Up
## 1069 2010  1.819 -0.096  3.548 -1.213  5.416 3.963460 -3.779      Down
## 1070 2010 -3.779  1.819 -0.096  3.548 -1.213 3.906558 -0.700      Down
## 1071 2010 -0.700 -3.779  1.819 -0.096  3.548 3.777406 -0.663      Down
## 1072 2010 -0.663 -0.700 -3.779  1.819 -0.096 3.951328  3.750        Up
## 1073 2010  3.750 -0.663 -0.700 -3.779  1.819 3.718470  0.456        Up
## 1074 2010  0.456  3.750 -0.663 -0.700 -3.779 3.195238  1.446        Up
## 1075 2010  1.446  0.456  3.750 -0.663 -0.700 3.972432  2.050        Up
## 1076 2010  2.050  1.446  0.456  3.750 -0.663 3.884522 -0.212      Down
## 1077 2010 -0.212  2.050  1.446  0.456  3.750 4.037410  1.650        Up
## 1078 2010  1.650 -0.212  2.050  1.446  0.456 3.905616  0.948        Up
## 1079 2010  0.948  1.650 -0.212  2.050  1.446 4.449160  0.586        Up
## 1080 2010  0.586  0.948  1.650 -0.212  2.050 4.576282  0.015        Up
## 1081 2010  0.015  0.586  0.948  1.650 -0.212 4.116414  3.599        Up
## 1082 2010  3.599  0.015  0.586  0.948  1.650 4.798758 -2.173      Down
## 1083 2010 -2.173  3.599  0.015  0.586  0.948 4.298262  0.043        Up
## 1084 2010  0.043 -2.173  3.599  0.015  0.586 4.177436 -0.861      Down
## 1085 2010 -0.861  0.043 -2.173  3.599  0.015 3.205160  2.969        Up
## 1086 2010  2.969 -0.861  0.043 -2.173  3.599 4.242568  1.281        Up
## 1087 2010  1.281  2.969 -0.861  0.043 -2.173 4.835082  0.283        Up
## 1088 2010  0.283  1.281  2.969 -0.861  0.043 4.454044  1.034        Up
```

```
## 1089 2010  1.034  0.283  1.281  2.969 -0.861 2.707105  0.069        Up
```

#Fit Train Data with lag2

```r
train.fit <- glm(Direction~Lag2, data = Weekly,family = binomial, subset = train)
train.fit
```

```
##
## Call:  glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = train)
##
## Coefficients:
## (Intercept)         Lag2
##      0.2033       0.0581
##
## Degrees of Freedom: 984 Total (i.e. Null);  983 Residual
## Null Deviance:       1355
## Residual Deviance: 1351  AIC: 1355
```

#Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```r
train_prob <- predict(train.fit, weekly.train, type = "response")
train_pred <- rep("Down", length(train_prob))
train_pred[train_prob > 0.5] = "Up"
direction_train <- Direction[!train]
direction_train
```

```
##   [1] Down Down Down Down Up   Down Down Down Down Up   Up   Up   Up   Up   Up
##  [16] Down Up   Up   Down Up   Up   Up   Up   Down Down Down Down Up   Up   Up
##  [31] Up   Down Up   Up   Down Up   Up   Down Down Up   Up   Down Down Up   Up
##  [46] Down Up   Up   Up   Down Up   Down Up   Down Down Down Down Up   Up   Down
##  [61] Up   Up   Up   Up   Up   Up   Down Up   Down Down Up   Down Up   Down Up
##  [76] Up   Down Down Up   Down Up   Down Up   Down Down Down Up   Up   Up   Up
##  [91] Down Up   Up   Up   Up   Up   Down Up   Down Up   Up   Up   Up   Up
## Levels: Down Up
```

```r
table(train_pred, direction_train)
```

```
##           direction_train
## train_pred Down Up
##       Down    9  5
##       Up     34 56
```

#Mean

```r
mean(train_pred == direction_train)
```

```
## [1] 0.625
```

After splitting the Weekly dataset into a training and test dataset, the logistic regression model was able to predict weekly market trends with an accuracy rate of 62.5%, which is a moderate improvement compared to the model that used the entire dataset. However, similar to the previous model, it performed better at predicting upward trends with an accuracy rate of 56/56+5 = 0.9180; 91.80%, compared to downward trends with an accuracy rate of only 9/9+34 = 0.2093; 20.93%.

One notable difference is that this model was able to significantly improve its ability to correctly predict downward trends, which is an improvement from the previous model. Overall, the results suggest that splitting the dataset into training and test sets is a better approach for assessing the performance of the logistic regression model compared to using the entire dataset.

Question 3(e)

(e) Repeat (d) using LDA and QDA. Interpret the results.

#Using LDA

```
lda.fit <- lda(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
lda.pred <- predict(lda.fit, weekly.train)
table(lda.pred$class, direction_train)
```

```
##       direction_train
##        Down Up
##   Down    9  5
##   Up     34 56
```

#Mean

```
mean(lda.pred$class == direction_train)
```

```
## [1] 0.625
```

#The application of Linear Discriminant Analysis (LDA) to develop a classifying model produced outcomes that were comparable to those obtained with the logistic regression model built in section 3(d). Both models were able to predict the weekly trends in the market with similar levels of accuracy, implying that LDA can be a viable alternative to logistic regression in certain scenarios.

#Using QDA

```
qda.fit <- qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
```

```
## Up     0.26036581
```

```
qda.pred <- predict(qda.fit, weekly.train)$class
table(qda.pred, direction_train)
```

```
##          direction_train
## qda.pred Down Up
##     Down    0  0
##     Up     43 61
```

#Mean

```
mean(qda.pred == direction_train)
```

```
## [1] 0.5865385
```

#Comment

The implementation of Quadratic Linear Analysis (QLA) resulted in a model that had a lower accuracy rate of 58.65% compared to the previous methods discussed. It is worth noting that this model only focused on predicting the correctness of weekly upward trends and did not take into account the downward weekly trends.

Question 3(f)

(f) Which of these methods appears to provide the best results on this data?

The methods that have the highest accuracy rates are the Logistic Regression and Linear Discriminant Analysis; both having rates of 62.5%.

Question 4

Construct the Hotelling T2 charts for future observations using the a simulated data Simulation Set-up

a) Use the set.seed("6559)
b) Generate 100 observations from bivariate normal distribution with

$$\mu = (2, 5)$$

   and covariance matrix-
$$var(x1) = 1; var(x2) = 0.5, cov(x1, x2) = 0.3$$

   .
c) Estimate the classical estimators of mean and covariances.
d) Generate 25 future observations, using bivariate normal distribution with

$$\mu = (2, 5)$$

   and covariance matrix -

$$var(x1) = 1; var(x2) = 0.5, cov(x1, x2) = 0.3.$$

e) Draw three T2 control chart for future observation using classical estimator and robust estimators of mean and covariance matrix. Draw your conclusions. g) Generate another 25 future observations, using bivariate normal distribution with
$$\mu = (2.4, 6)$$

   and covariance matrix -
$$var(x1) = 1; var(x2) = 0.5, cov(x1, x2) = 0.3$$

   . and repeat (e).

f) Offer your comments. Compare your results with univariate charts for individual observations.

Question 4(a)

a) Use the set.seed(6559)

```
set.seed("6559")
```

Question 4(b)

Generate 100 observations from bivariate normal distribution with

$$\mu = (2, 5)$$

and covariance matrix-

$$var(x1) = 1; var(x2) = 0.5, cov(x1, x2) = 0.3$$

.

```
mu_new <- c(2, 5)
mu_new
```

```
## [1] 2 5
```

```
sigma_new <- matrix(c(1, 0.3, 0.3, 0.5), nrow = 2)
sigma_new
```

```
##      [,1] [,2]
## [1,]  1.0  0.3
## [2,]  0.3  0.5
```

```
data_new <- mvrnorm(100, mu_new, sigma_new)
data_new
```

```
##              [,1]     [,2]
##   [1,]  3.22824656 5.244019
##   [2,]  0.82464533 4.847112
##   [3,]  2.29063607 5.225665
##   [4,]  2.04520349 5.138790
##   [5,]  1.51479957 5.355329
##   [6,]  1.87727336 4.746011
##   [7,]  2.64622117 5.585640
##   [8,]  0.59590487 4.590484
##   [9,]  2.34407443 5.372027
##  [10,]  0.90005698 5.273129
##  [11,] -0.10794711 4.384537
##  [12,]  3.06191057 5.756621
##  [13,]  1.11118514 4.576824
##  [14,]  1.53909049 4.756555
##  [15,]  0.94037206 4.189063
##  [16,]  3.64865710 5.432835
##  [17,]  1.88358425 6.029792
##  [18,]  3.00937154 4.996733
##  [19,]  1.85932053 4.319459
##  [20,]  1.95549394 4.871051
##  [21,]  1.37694618 3.762683
##  [22,] -0.51718312 4.262277
##  [23,]  2.59961697 4.531685
```

```
##   [24,]  2.36024560 5.003060
##   [25,]  1.83949426 5.328928
##   [26,]  3.81436455 6.707470
##   [27,]  2.87022786 6.031748
##   [28,]  2.15015213 4.171342
##   [29,]  2.71664025 6.229400
##   [30,]  1.07394424 4.903813
##   [31,]  3.23675627 5.702506
##   [32,]  1.72128409 4.465321
##   [33,]  0.52375701 5.016169
##   [34,]  1.36558683 3.349871
##   [35,]  4.21887440 5.788419
##   [36,]  1.08344813 4.020579
##   [37,]  1.72237978 5.015685
##   [38,]  2.46159490 5.248217
##   [39,]  1.88642040 5.392653
##   [40,]  2.01027492 6.274578
##   [41,]  1.15250797 4.094235
##   [42,]  1.59478369 5.505659
##   [43,] -0.17587091 5.156064
##   [44,]  1.31173476 5.424628
##   [45,]  2.05388625 5.435978
##   [46,]  1.08925775 4.879550
##   [47,]  0.89769110 3.929704
##   [48,]  2.86727320 4.605870
##   [49,]  0.63164384 4.296384
##   [50,]  2.12313052 5.126278
##   [51,]  2.24767531 4.066196
##   [52,]  2.50275134 4.250906
##   [53,]  1.90759515 5.575957
##   [54,]  1.25776244 5.474296
##   [55,]  2.12303671 4.881284
##   [56,]  2.45603198 4.982059
##   [57,]  1.99997537 4.448861
##   [58,]  3.44847926 5.273657
##   [59,]  2.57809349 4.819681
##   [60,]  3.28246786 5.474550
##   [61,]  1.60648153 4.765050
##   [62,] -0.08931654 4.552810
##   [63,]  1.49822639 5.989117
##   [64,]  2.72328631 4.766697
##   [65,]  4.01028553 5.852390
##   [66,]  1.65863544 5.343390
##   [67,]  2.05597431 3.635656
##   [68,]  2.42367855 5.729630
##   [69,]  2.69840635 5.484062
##   [70,] -0.98412681 3.653791
##   [71,]  4.16616926 6.860416
##   [72,]  0.30386819 4.062290
##   [73,]  2.46711076 5.464220
##   [74,]  3.15667968 5.756438
##   [75,]  3.20675867 6.133512
##   [76,]  1.42863463 4.892832
##   [77,]  1.56015266 5.165574
```

```
##  [78,]  1.95884447 4.629197
##  [79,]  2.36793453 5.173926
##  [80,]  3.09132665 6.094633
##  [81,]  1.95542592 4.771736
##  [82,]  1.67996037 4.432422
##  [83,]  1.15910367 4.250807
##  [84,]  2.09108945 5.121783
##  [85,]  3.43271914 5.021997
##  [86,]  3.08052421 4.558840
##  [87,]  2.46897349 4.271504
##  [88,]  2.68740057 5.073524
##  [89,]  2.86201160 5.134098
##  [90,]  1.58203175 5.115631
##  [91,]  3.90432840 5.028372
##  [92,]  1.33411508 5.727546
##  [93,]  3.09653205 4.893115
##  [94,]  2.41555903 5.513601
##  [95,]  1.99871396 4.536350
##  [96,]  2.38950535 5.702710
##  [97,]  1.46564201 5.165978
##  [98,]  3.08723238 5.268898
##  [99,]  2.38025676 4.542060
## [100,]  1.39841710 5.053427
```

Question 4(c)

Estimate the classical estimators of mean and covariances

```r
classical.mean <- colMeans(data_new)
classical.cov <- cov(data_new)
classical.mean
```

```
## [1] 2.028114 5.027899
```

```r
classical.cov
```

```
##           [,1]      [,2]
## [1,] 1.0293141 0.3524414
## [2,] 0.3524414 0.4439050
```

Question 4(d)

Generate 25 future observations, using bivariate normal distribution with

$$\mu = (2, 5)$$

and covariance matrix -

$$var(x1) = 1; var(x2) = 0.5, cov(x1, x2) = 0.3.$$

```r
future.data <- mvrnorm(25, mu_new, sigma_new)
future.data
```

```
##              [,1]     [,2]
##  [1,]  2.49459807 5.105514
##  [2,]  3.37172120 4.895565
##  [3,]  1.52056474 5.441217
##  [4,]  0.44763081 5.651829
```

```
##  [5,]  1.29044758 4.273317
##  [6,]  2.49444071 4.186979
##  [7,] -0.07751139 5.140148
##  [8,]  0.87364506 5.741826
##  [9,]  1.93671789 4.215154
## [10,] -0.55367721 4.149838
## [11,]  2.95646967 5.074981
## [12,]  1.35781685 5.005896
## [13,]  3.23063347 5.183684
## [14,]  2.17826263 6.421456
## [15,] -0.30044607 5.146776
## [16,]  2.18925186 4.968234
## [17,]  2.10165085 5.815150
## [18,] -0.16423246 4.918217
## [19,]  1.66315792 5.079373
## [20,]  2.22281729 6.550927
## [21,]  1.53838520 5.229286
## [22,]  3.22041104 6.454068
## [23,] -0.95563927 4.136426
## [24,]  1.54430347 4.994265
## [25,]  3.15049309 5.694374
```

```
classical.mean1 <- colMeans(future.data)
classical.cov1 <- cov(future.data)
classical.mean1
```

```
## [1] 1.589277 5.178980
```

```
classical.cov1
```

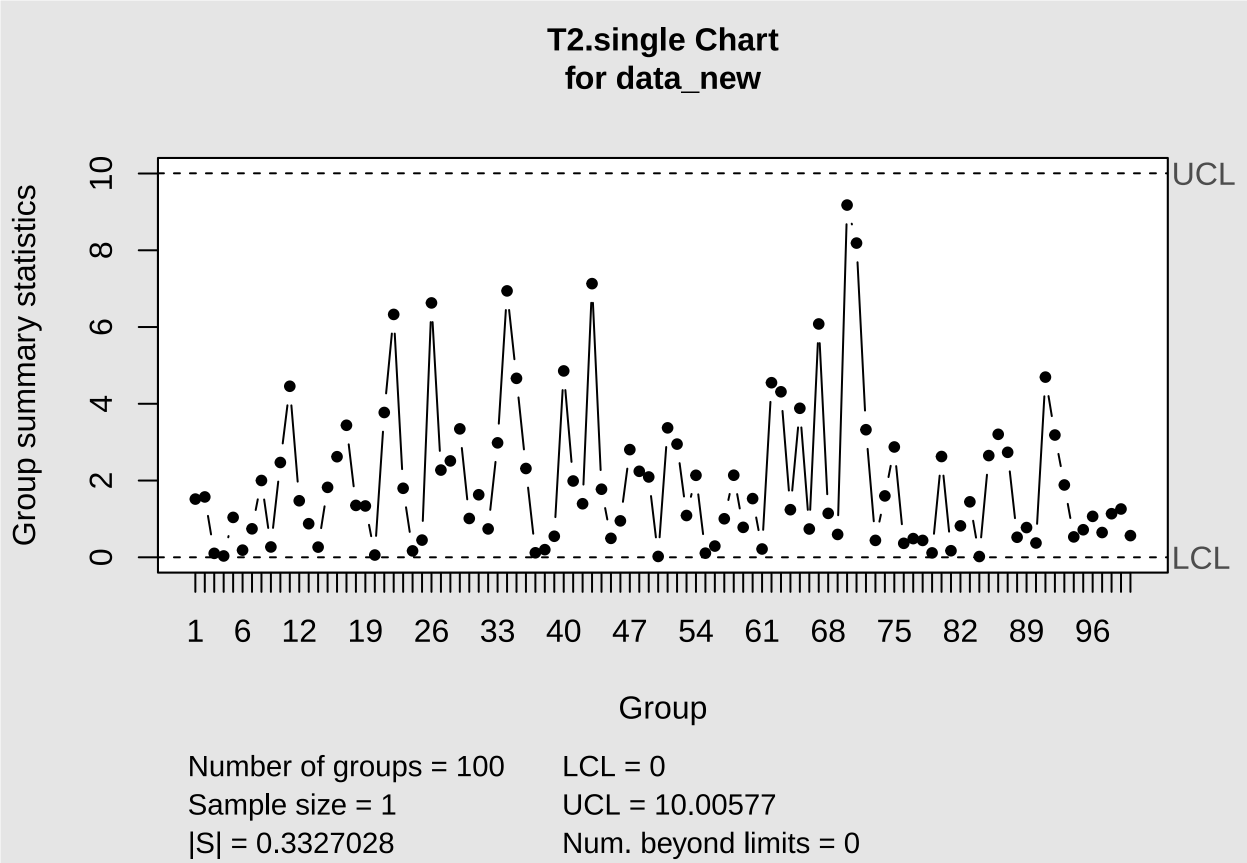
```
##           [,1]      [,2]
## [1,] 1.6014508 0.3139975
## [2,] 0.3139975 0.4791117
```

Question 4(e)

Draw three T2 control chart for future observation using classical estimator and robust estimators of mean and covariance matrix. Draw your conclusions. g) Generate another 25 future observations, using bivariate normal distribution with mu = (2.4; 6) and covariance matrix - var(x1)=1; var(x2)=.5, cov(x1,x2)=0.3. and repeat (e).

#T2 control chart for Datanew

```
qmd <- mqcc(data_new, type = "T2")
```



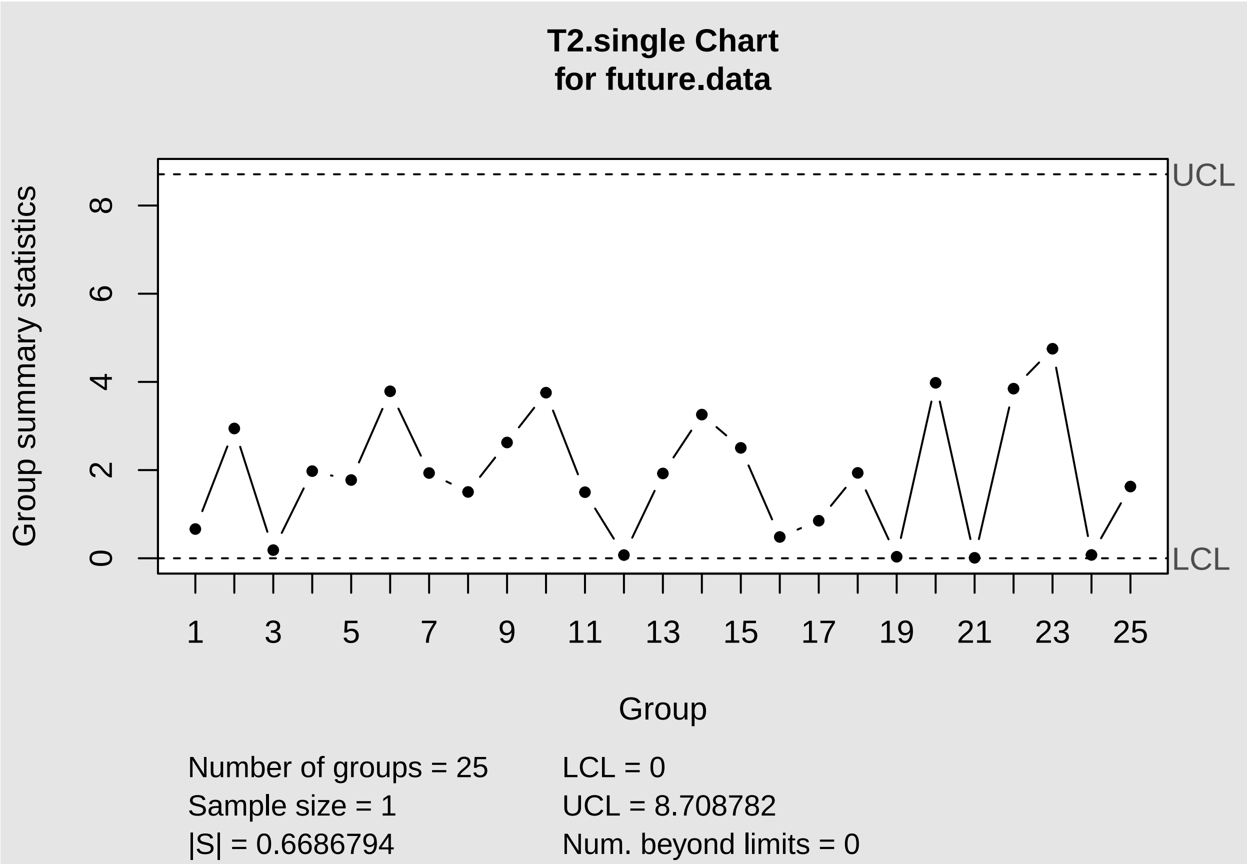

```
summary(qmd)
```

```
##
## Call:
## mqcc(data = data_new, type = "T2")
##
## T2.single chart for data_new
##
## Summary of group statistics:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.020035 0.543787 1.420222 1.980000 2.753418 9.178539
##
## Number of variables:  2
## Number of groups:  100
## Group sample size:  1
##
## Center:
##       V1       V2
## 2.028114 5.027899
##
## Covariance matrix:
##           V1        V2
## V1 1.0293141 0.3524414
## V2 0.3524414 0.4439050
## |S|:  0.3327028
##
```

```
## Control limits:
##   LCL      UCL
##     0 10.00577
```

#For future data(#T2 control chart for new data)

```r
qmf <- mqcc(future.data, type = "T2")
```



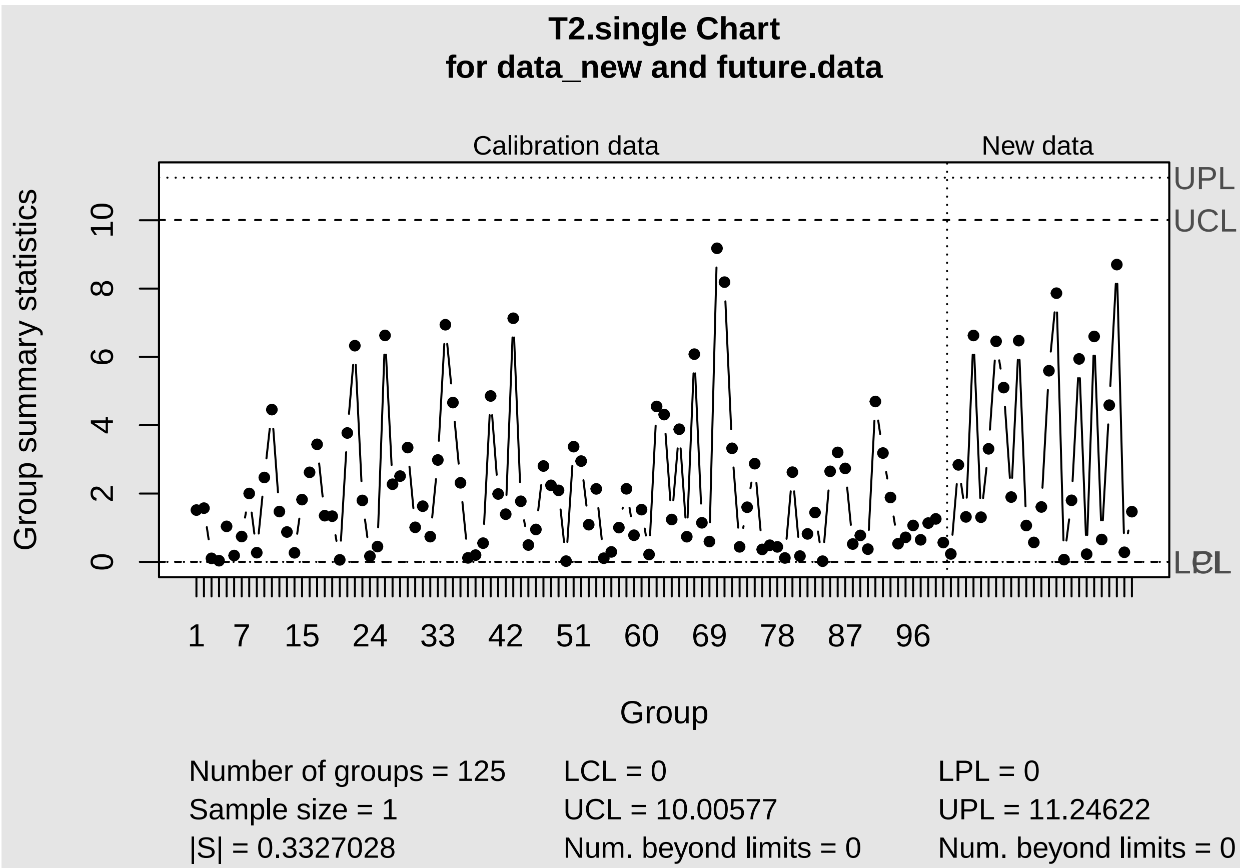

```r
summary(qmf)
```

```
##
## Call:
## mqcc(data = future.data, type = "T2")
##
## T2.single chart for future.data
##
## Summary of group statistics:
##     Min.   1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.010321 0.662642 1.923103 1.920000 2.943220 4.751839
##
## Number of variables:  2
## Number of groups:  25
## Group sample size:  1
##
## Center:
##        V1       V2
## 1.589277 5.178980
##
```

```
## Covariance matrix:
##            V1        V2
## V1 1.6014508 0.3139975
## V2 0.3139975 0.4791117
## |S|:  0.6686794
##
## Control limits:
##  LCL       UCL
##    0 8.708782
```

#For Old data and new Data

```
qq <- mqcc(data_new, type = "T2", newdata = future.data, pred.limits = TRUE)
```



#Comment

Based on the given chart, we can observe that all the data points fall within the control limits and there are no observations that are considered outliers or out of control. This indicates that the process is stable and in control. There is no evidence of any special cause variation, which would be indicated by data points outside of the control limits, indicating that the process is operating as expected and within acceptable levels of variation. Therefore, we can conclude that the process is stable and under statistical control.

#Classical T2 chart

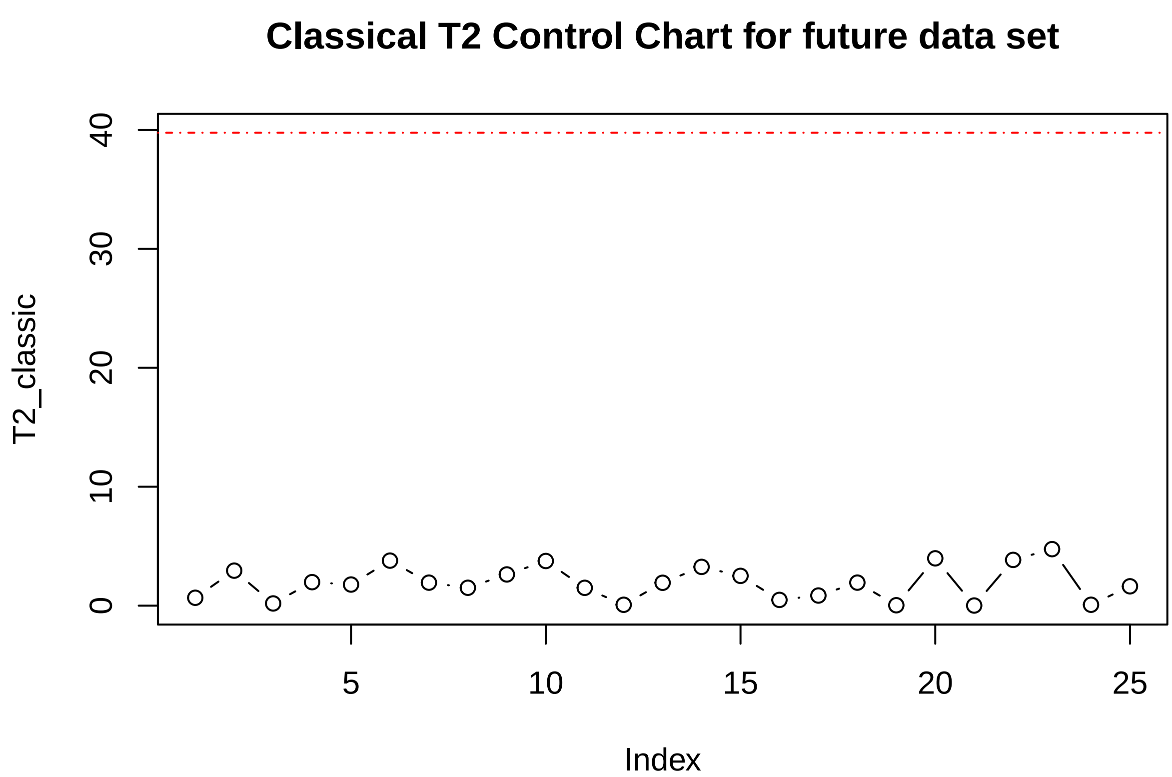
```
T2_classic <- apply(future.data, 1, function(x) t(x - classical.mean1) %*% solve(classical.cov1) %*% (x
print(T2_classic)
```

```
##  [1] 0.66264192 2.94322022 0.18500187 1.97631815 1.77420495 3.78712233
##  [7] 1.93340395 1.50393425 2.62580038 3.75571517 1.49874148 0.07250945
## [13] 1.92310257 3.25847315 2.50400465 0.48303913 0.85124098 1.93652319
```

```
## [19] 0.03458370 3.97914526 0.01032084 3.84685820 4.75183909 0.07536243
## [25] 1.62689268
```

```
n <- nrow(data_new)
m <- nrow(future.data)
p <- ncol(data_new)
alpha <- 0.05
UCL_classic <- ((n-1)*(n+1)*p)/((n**2 -n*p)*qf(alpha, p, n-p))
```

```
plot(T2_classic, ylim = c(0, UCL_classic), type = "b", main = "Classical T2 Control Chart for future da
abline(h = UCL_classic, col = "red", lty = 4)
```



Question 4(f)

Offer your comments. Compare your results with univariate charts for individual observations.

##can draw individual x-bar charts using each variable, considering the bonferroni correction

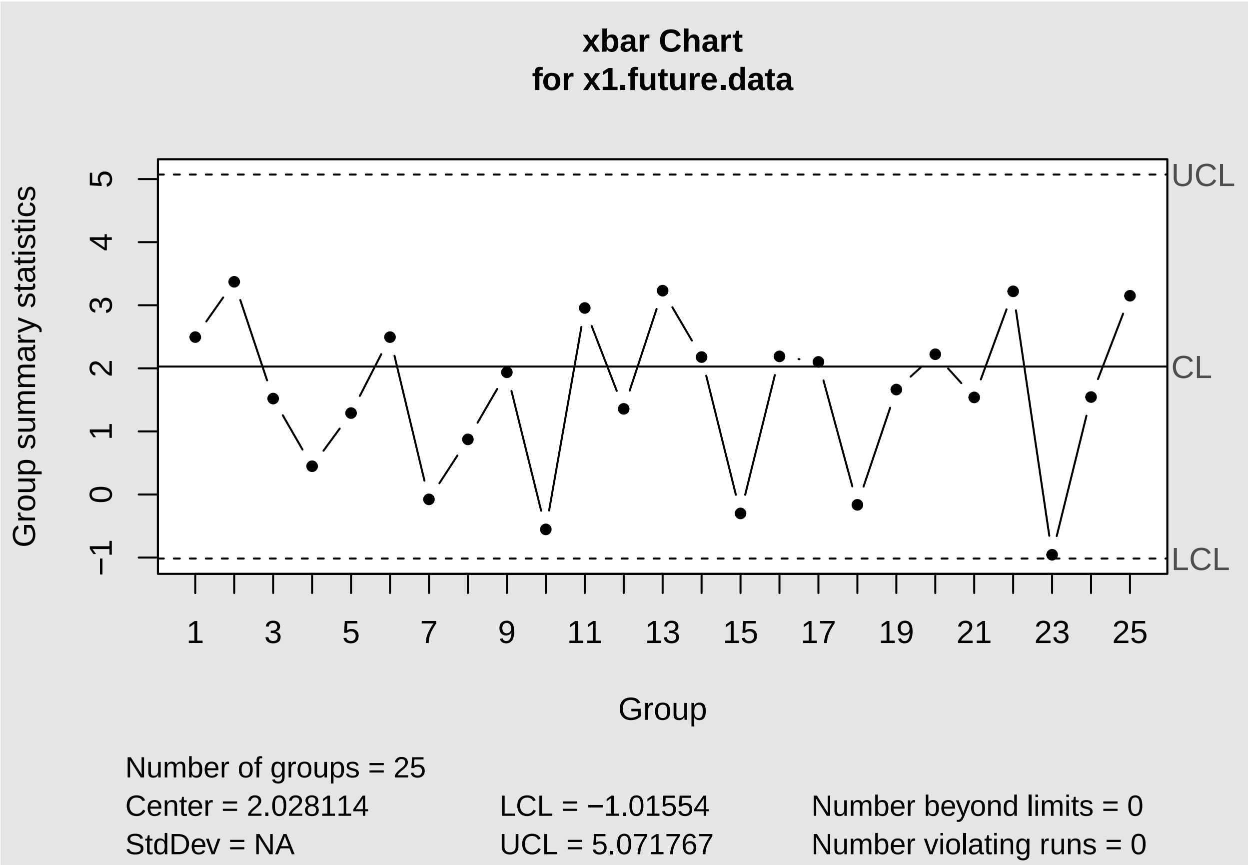
```
x1.data_new <-data_new[,1]
x2.data_new <- data_new[,2]
x1.future.data <- future.data[,1]
x2.future.data <- future.data[,2]
x1.data_new_mean <- mean(x1.data_new)
x2.data_new_mean <- mean(x2.data_new)
x1.data_new_sd <- sd(x1.data_new)
x2.data_new_sd <- sd(x2.data_new)
```

#Upper lower limit specification

```r
uclx1 <- x1.data_new_mean + (3*x1.data_new_sd)
lclx1 <- x1.data_new_mean - (3*x1.data_new_sd)
uclx2 <- x2.data_new_mean + (3*x2.data_new_sd)
lclx2 <- x2.data_new_mean - (3*x2.data_new_sd)
```

#Plot the xbar x1 future

```r
qx1 <- qcc(x1.future.data, type = "xbar", center = x1.data_new_mean, x1.data_new_sd, limits = c(lclx1,uc
```

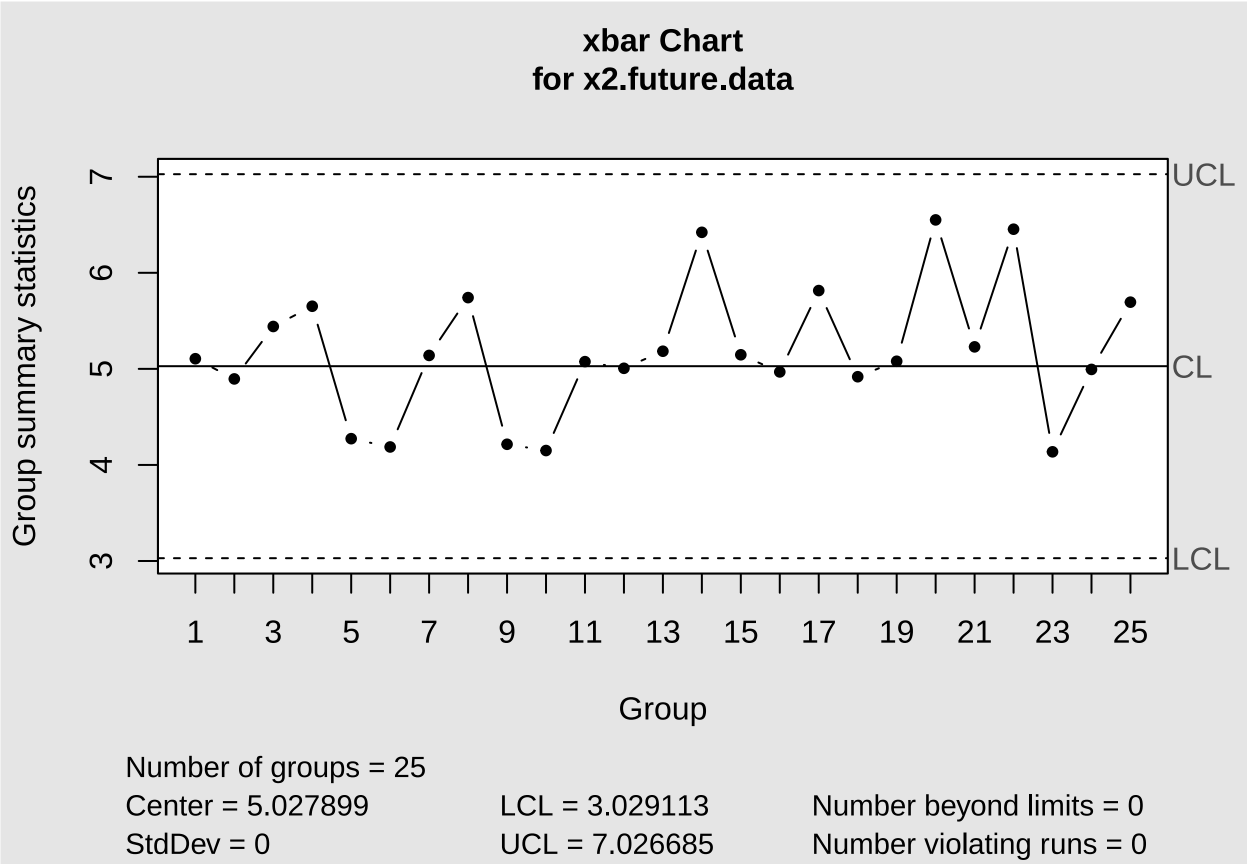



```r
qx1
```

```
## List of 11
##  $ call      : language qcc(data = x1.future.data, type = "xbar", sizes = x1.data_new_sd, center = x1
##  $ type      : chr "xbar"
##  $ data.name : chr "x1.future.data"
##  $ data      : num [1:25, 1] 2.495 3.372 1.521 0.448 1.29 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics: Named num [1:25] 2.495 3.372 1.521 0.448 1.29 ...
##   ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
##  $ sizes     : num [1:25] 1.01 1.01 1.01 1.01 1.01 ...
##  $ center    : num 2.03
##  $ std.dev   : num NA
##  $ nsigmas   : num 3
##  $ limits    : num [1, 1:2] -1.02 5.07
##   ..- attr(*, "dimnames")=List of 2
##  $ violations:List of 2
##  - attr(*, "class")= chr "qcc"
```

#for xbar x2 future

```
qx2 <- qcc(x2.future.data, type = "xbar", center = x2.data_new_mean, x2.data_new_sd, limits = c(lclx2,u
```




```
qx2
```

```
## List of 11
##  $ call      : language qcc(data = x2.future.data, type = "xbar", sizes = x2.data_new_sd, center = x
##  $ type      : chr "xbar"
##  $ data.name : chr "x2.future.data"
##  $ data      : num [1:25, 1] 5.11 4.9 5.44 5.65 4.27 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics: Named num [1:25] 5.11 4.9 5.44 5.65 4.27 ...
##   ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
##  $ sizes     : num [1:25] 0.666 0.666 0.666 0.666 0.666 ...
##  $ center    : num 5.03
##  $ std.dev   : num 0
##  $ nsigmas   : num 3
##  $ limits    : num [1, 1:2] 3.03 7.03
##   ..- attr(*, "dimnames")=List of 2
##  $ violations:List of 2
##  - attr(*, "class")= chr "qcc"
```

#Comment

1. It can be inferred that all the observations in the chart were within control limits, as none of them exceeded the upper or lower control limit.
2. In the first chart, most of the observations were below the control limit, while in the second chart, they

were above the control limit.

3. The lower control limit of the first chart was in the negative range.

Question 4(g)

Generate another 25 future observations, using bivariate normal distribution with mu= (2.4; 6) and covariance matrix - var(x1)=1; var(x2)=.5, cov(x1,x2)=0.3. and repeat (e).

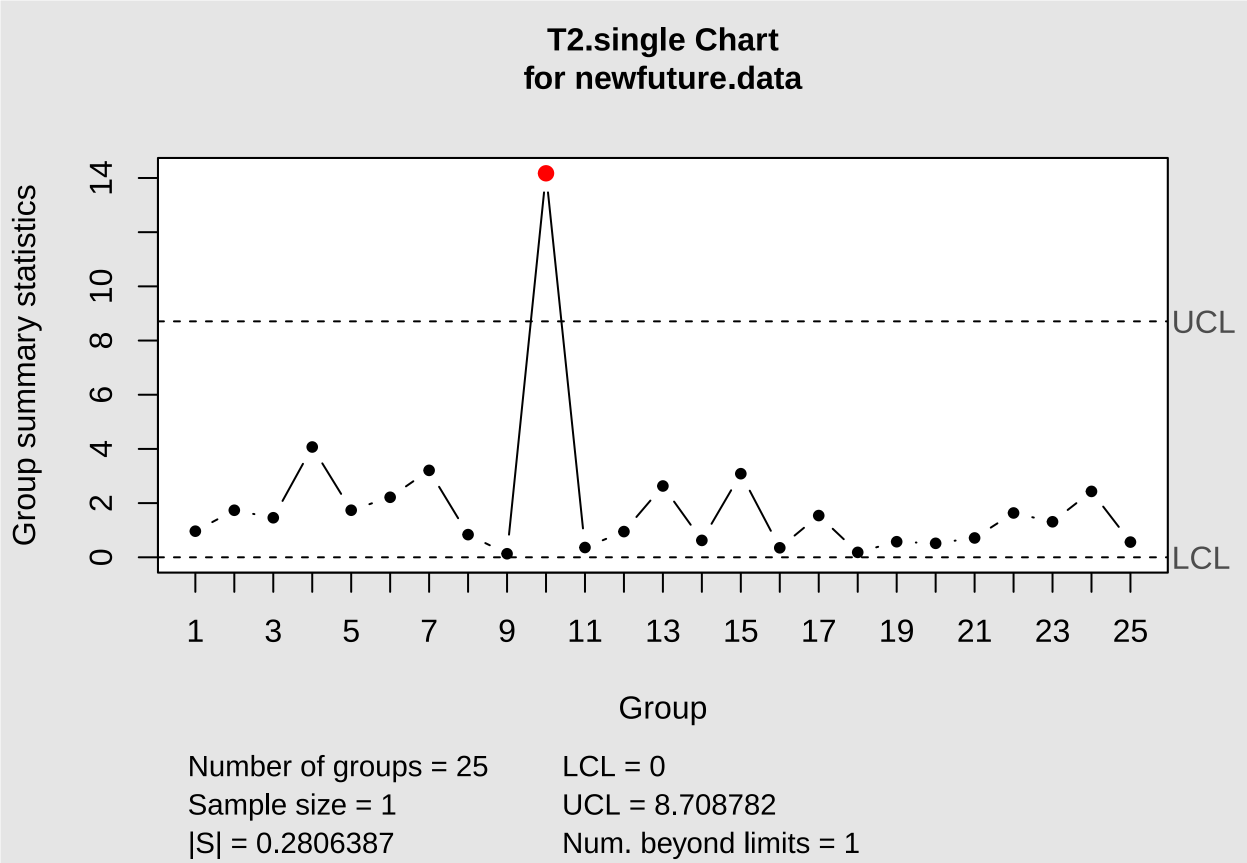
Ans:

```
mu_f <- c(2.4, 6)
sigma_new <- matrix(c(1, 0.3, 0.3, 0.5), nrow = 2)
newfuture.data <- mvrnorm(25, mu_f, sigma_new)
newfuture.data
```

```
##           [,1]     [,2]
##  [1,] 3.307050 6.815931
##  [2,] 1.732756 5.626937
##  [3,] 3.552650 5.821028
##  [4,] 4.932011 6.756723
##  [5,] 4.183412 6.359303
##  [6,] 3.586077 5.670151
##  [7,] 1.770237 5.107920
##  [8,] 2.871253 5.676942
##  [9,] 3.240412 6.423278
## [10,] 2.108495 7.914922
## [11,] 2.431065 6.145469
## [12,] 2.053433 6.024066
## [13,] 1.562588 5.351807
## [14,] 2.209174 5.931489
## [15,] 4.684600 6.767460
## [16,] 2.470257 6.193990
## [17,] 4.123836 6.372095
## [18,] 2.671862 5.949128
## [19,] 2.380706 5.768234
## [20,] 2.356018 6.183119
## [21,] 3.622273 6.701503
## [22,] 4.068034 6.896960
## [23,] 2.240925 5.497583
## [24,] 3.994280 7.168850
## [25,] 2.255869 6.030680
```

#For New future data

```
qmf_new_future <- mqcc(newfuture.data, type = "T2")
```



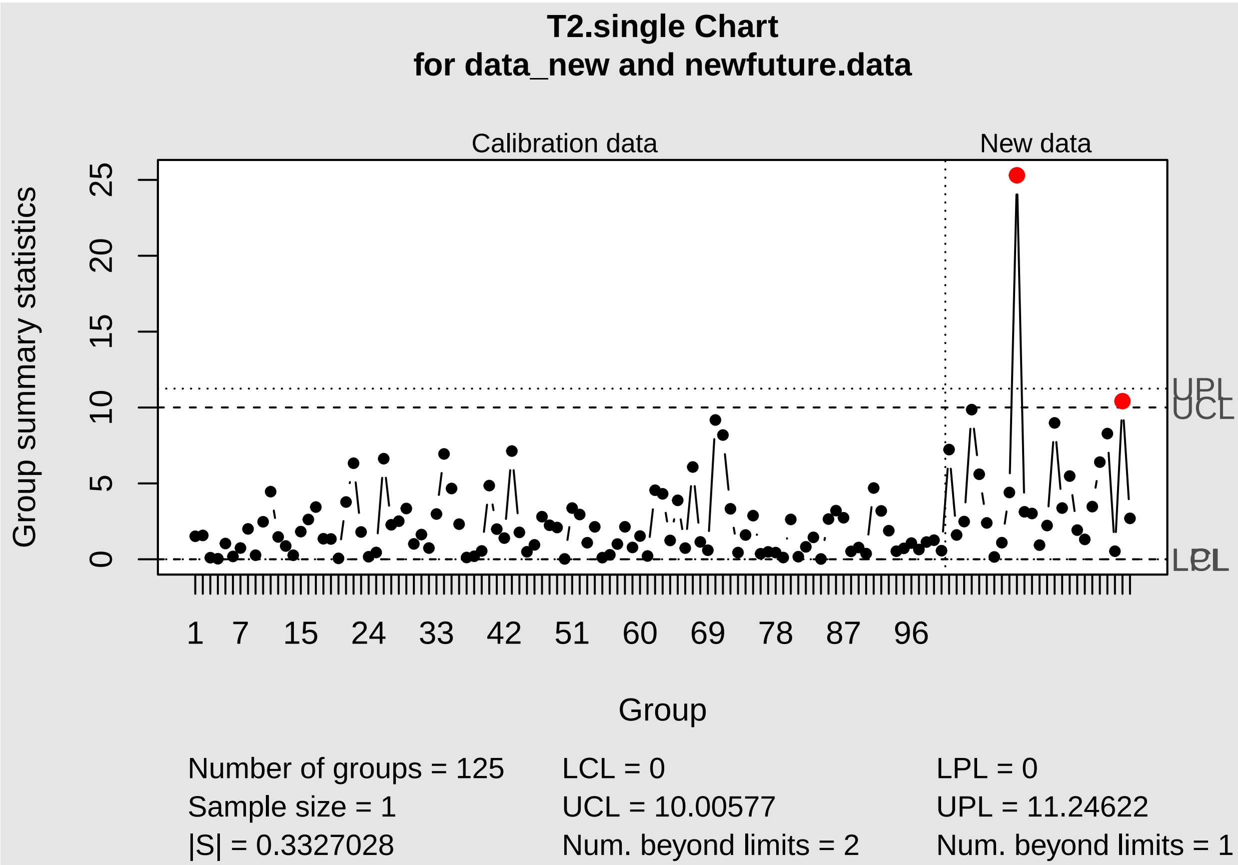

```
summary(qmf_new_future)
```

```
##
## Call:
## mqcc(data = newfuture.data, type = "T2")
##
## T2.single chart for newfuture.data
##
## Summary of group statistics:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.131317  0.574601  1.310639  1.920000  2.215396 14.172752
##
## Number of variables:  2
## Number of groups:  25
## Group sample size:  1
##
## Center:
##       V1       V2
## 2.976371 6.206223
##
## Covariance matrix:
##           V1        V2
## V1 0.9456184 0.3091447
## V2 0.3091447 0.3978446
## |S|:  0.2806387
##
```

```
## Control limits:
##  LCL      UCL
##    0 8.708782
```

#New future data and Old data

```
qq_newfuture <- mqcc(data_new, type = "T2", newdata = newfuture.data, pred.limits = TRUE)
```



#Implement

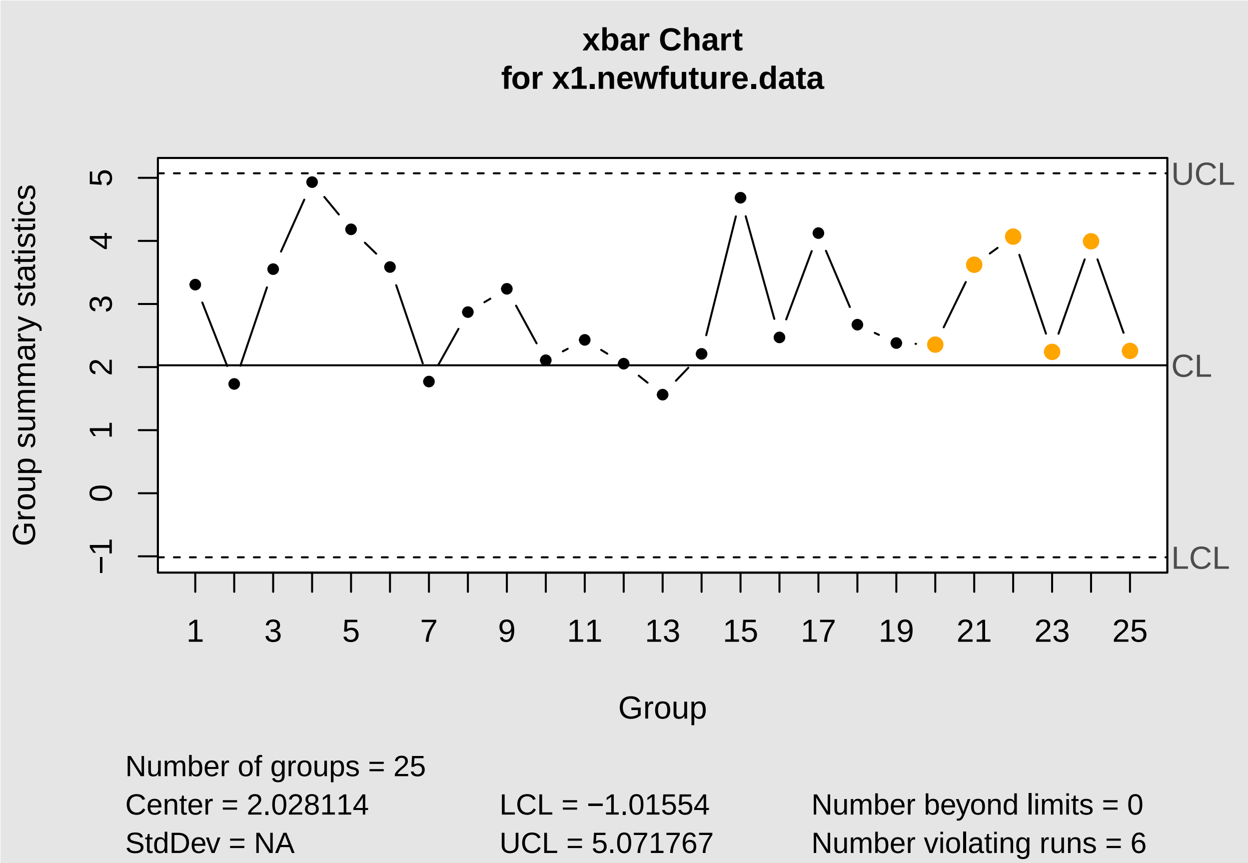
The chart shows two data points above the upper control limit, which could be outliers in future observations. One of these points is also above the upper prediction limit, indicating a deviation from the expected trend. Further investigation is needed to determine the cause of this deviation and take corrective actions as necessary.

#X1, X2 for new future data

```
x1.newfuture.data <- newfuture.data[,1]
x2.newfuture.data <- newfuture.data[,2]
```

#Xbar chart for New future data ##For X1

```
qx1_newfuture <- qcc(x1.newfuture.data, type = "xbar", center = x1.data_new_mean, x1.data_new_sd, limits
```

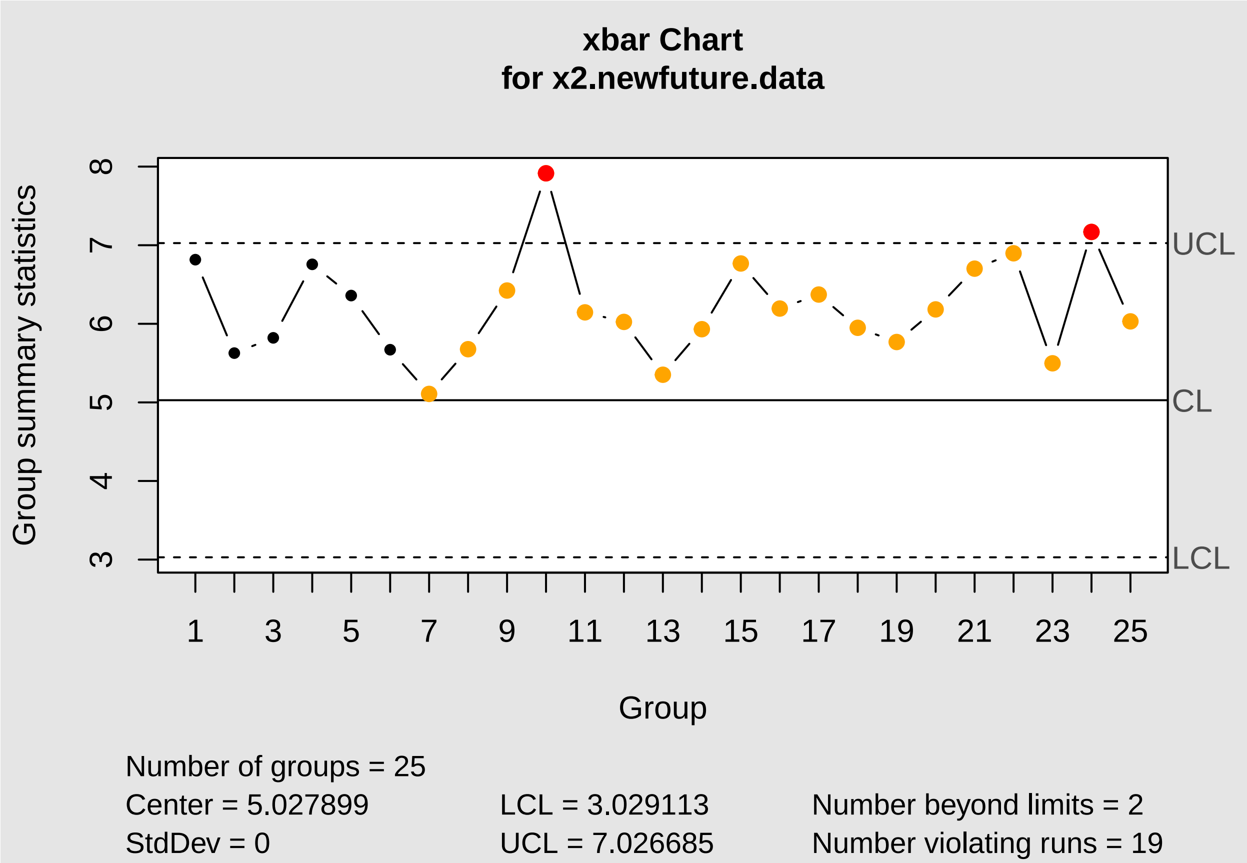



```
qx1_newfuture
```

```
## List of 11
##  $ call      : language qcc(data = x1.newfuture.data, type = "xbar", sizes = x1.data_new_sd, center =
##  $ type      : chr "xbar"
##  $ data.name : chr "x1.newfuture.data"
##  $ data      : num [1:25, 1] 3.31 1.73 3.55 4.93 4.18 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics: Named num [1:25] 3.31 1.73 3.55 4.93 4.18 ...
##   ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
##  $ sizes     : num [1:25] 1.01 1.01 1.01 1.01 1.01 ...
##  $ center    : num 2.03
##  $ std.dev   : num NA
##  $ nsigmas   : num 3
##  $ limits    : num [1, 1:2] -1.02 5.07
##   ..- attr(*, "dimnames")=List of 2
##  $ violations:List of 2
##  - attr(*, "class")= chr "qcc"
```

#For X2

```
qx2_newfuture <- qcc(x2.newfuture.data, type = "xbar", center = x2.data_new_mean, x2.data_new_sd, limits
```




```
qx2_newfuture
```

```
## List of 11
##  $ call      : language qcc(data = x2.newfuture.data, type = "xbar", sizes = x2.data_new_sd, center =
##  $ type      : chr "xbar"
##  $ data.name : chr "x2.newfuture.data"
##  $ data      : num [1:25, 1] 6.82 5.63 5.82 6.76 6.36 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics: Named num [1:25] 6.82 5.63 5.82 6.76 6.36 ...
##   ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
##  $ sizes     : num [1:25] 0.666 0.666 0.666 0.666 0.666 ...
##  $ center    : num 5.03
##  $ std.dev   : num 0
##  $ nsigmas   : num 3
##  $ limits    : num [1, 1:2] 3.03 7.03
##   ..- attr(*, "dimnames")=List of 2
##  $ violations:List of 2
##  - attr(*, "class")= chr "qcc"
```

Question 5

Refer the class note on discriminant analysis and definition notations of \$ w,B,S\$. Show that the w maximizing

$$\frac{w^T B w}{w^T S w}$$

satisfies

$$S^{-1}Bw = \lambda w$$

. Hence, w is eigen vector and λ is eigen value of $S^{-1}B$. Argue that we can maximize $w^T Bw$ subject to $w^T Sw = a$ where a is a constant. Then introduce a Lagrange multiplier for the constraint and differentiate with respect to elements of w.

#Ans:

The goal of discriminant analysis is to find a linear combination of the input features that maximizes the between-class scatter while minimizing the within-class scatter. We can define the within-class scatter matrix as

$$Sw = \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

where C_i is the set of observations in class i, μ_i is the mean vector of the observations in class i, and k is the number of classes.

Similarly, we can define the between-class scatter matrix as

$$Bw = \sum_{i=1}^{k} n_i(\mu_i - \mu)(\mu_i - \mu)^T$$

where n_i is the number of observations in class i, and μ is the mean vector of all the observations.

To find the weight vector w that maximizes the ratio of between-class scatter to within-class scatter, we can maximize the function

$$\frac{w^T Bw}{w^T Sw}$$

subject to the constraint $w^T Sw = 1$.

We can also maximize $w^T Bw$ subject to the constraint $w^T Sw = a$, where a is a constant, by introducing a Lagrange multiplier and writing the Lagrangian function as

$$L(w, \lambda) = w^T Bw - \lambda(w^T Sw - a)$$

.

Taking the derivative of the Lagrangian with respect to w and setting it to zero, we get

$$2Bw - 2\lambda Sw = 0$$

.

Multiplying both sides of the equation by S^{-1}, we get

$$S^{-1}Bw = \lambda w$$

.

This equation shows that w is an eigenvector of $S^{-1}Bw$ with eigenvalue λ. Therefore, to maximize $w^T Bw$ subject to the constraint $w^T Sw = a$, we need to find the eigenvector w that corresponds to the largest eigenvalue of $S^{-1}Bw$.