ProgrammingAssignments

July 3, 2024

0.1 Mahdi Anvari 610700002 Homework 3 of Machine Learning

0.2 Question 5

Part II: Programming assignments

5.

Compute the value of a Gaussian pdf, $\mathcal{N}(m,S)$, at $x_1 = [0.2, 1.3]^T$ and $x_2 = [2.2, -1.3]^T$, where

$$m = [0,\ 1]^T,\ S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

[19]:
```
library(mvtnorm)
M <- matrix(c(0,1), nrow=1)
S <- matrix(c(1,0,0,1), nrow=2)
pdf <- function(x) {
  dmvnorm(x, mean = M, sigma = S)
}
print(M)
print(S)
```

```
     [,1] [,2]
[1,]    0    1
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

[20]:
```
x1 <- c(0.2,1.3)
print(pdf(x1))
```

```
[1] 0.1491389
```

[21]:
```
x2 <- c(2.2,-1.3)
print(pdf(x2))
```

```
[1] 0.00100489
```

0.3 Question 6

6.

Generate $N = 500$ 2-dimensional data points that are distributed according to the Gaussian distribution $\mathcal{N}(m, S)$, with mean $m = [0, 0]^T$ and covariance matrix $S = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$, for the following cases:

$\sigma_1^2 = \sigma_2^2 = 1, \sigma_{12} = 0$

$\sigma_1^2 = \sigma_2^2 = 0.2, \sigma_{12} = 0$

$\sigma_1^2 = \sigma_2^2 = 2, \sigma_{12} = 0$

$\sigma_1^2 = 0.2, \sigma_2^2 = 2, \sigma_{12} = 0$

$\sigma_1^2 = 2, \sigma_2^2 = 0.2, \sigma_{12} = 0$

$\sigma_1^2 = \sigma_2^2 = 1, \sigma_{12} = 0.5$

$\sigma_1^2 = 0.3, \sigma_2^2 = 2, \sigma_{12} = 0.5$

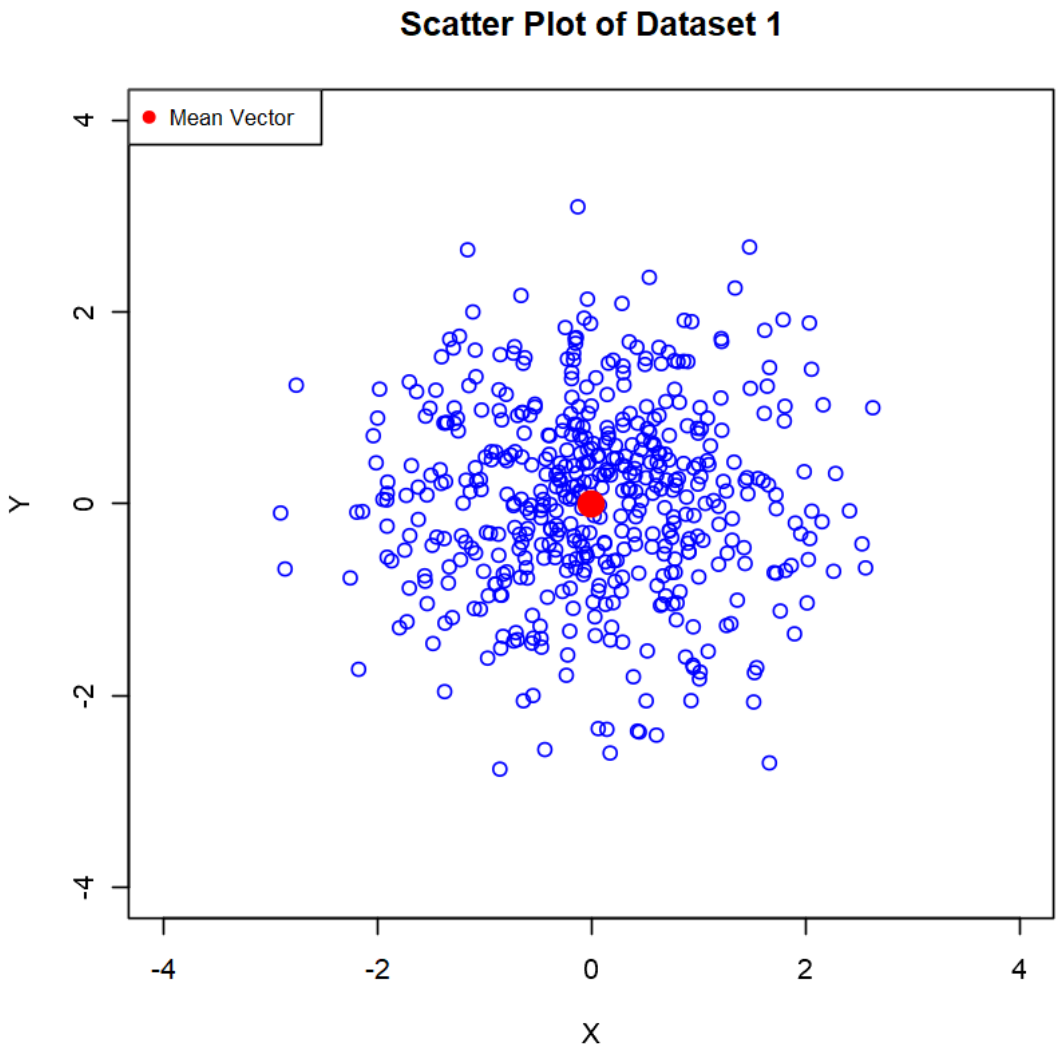
$\sigma_1^2 = 0.3, \sigma_2^2 = 2, \sigma_{12} = -0.5$

Plot each data set and comment on the shape of the clusters formed by the data points.

[82]:
```
library(MASS)
M1 <- matrix(c(0,0), nrow=1)
S1 <- matrix(c(1,0,0,1), nrow=2)
Points1 <- mvrnorm(500, mu = M1, Sigma = S1)
print(M1)
print(S1)
print(head(Points1))
```

```
     [,1] [,2]
[1,]    0    0
     [,1] [,2]
[1,]    1    0
[2,]    0    1
           [,1]       [,2]
[1,] -0.8226540 -1.3810122
[2,]  1.2721996 -0.5131328
[3,] -0.1917056 -0.2072877
[4,]  0.7830568 -0.5740718
[5,]  1.0184191  1.0048632
[6,]  0.9028550 -0.4094177
```

[83]:
```
plot(Points1, main = "Scatter Plot of Dataset 1", xlim = c(-4, 4), ylim = c(-4,␣
↪4) ,
    xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```

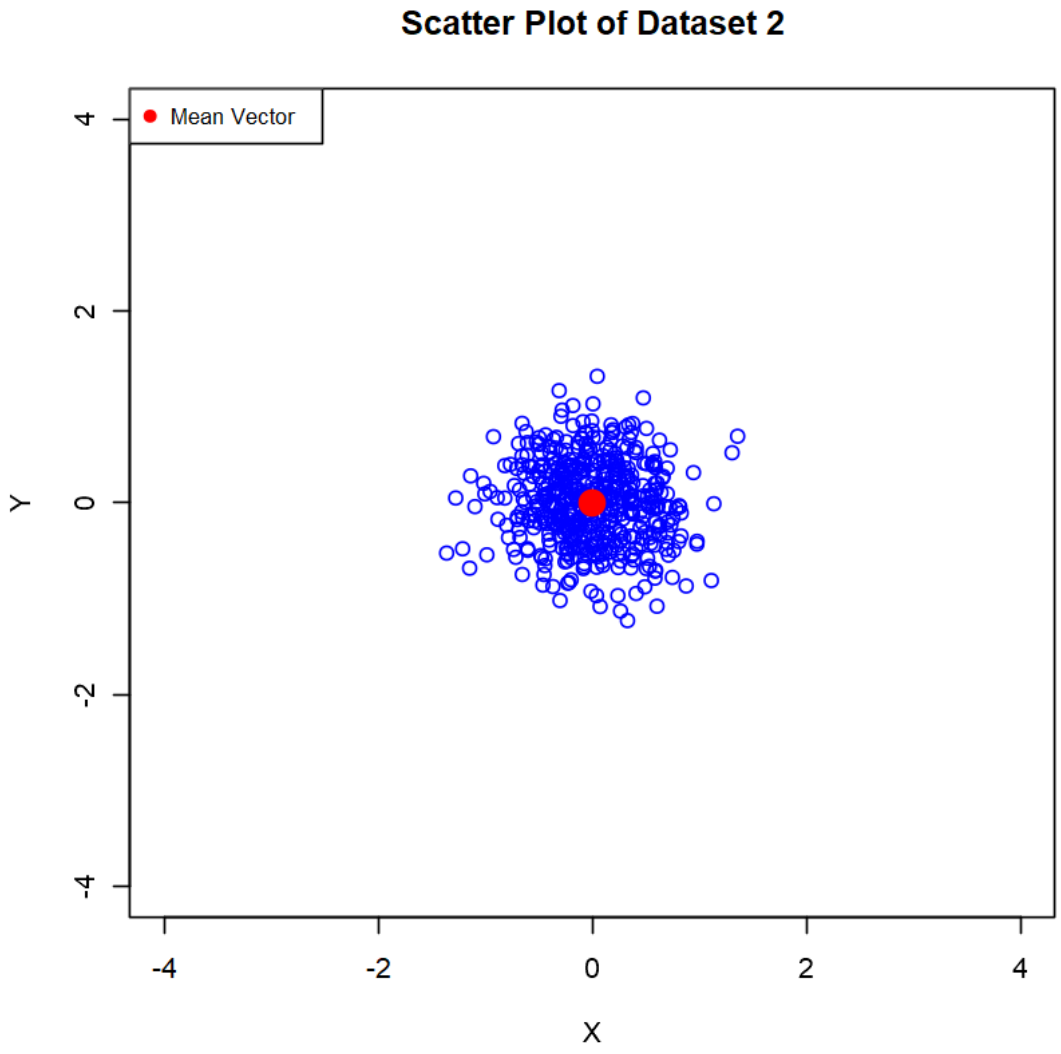


[84]:
```
S2 <- matrix(c(0.2,0,0,0.2), nrow=2)
Points2 <- mvrnorm(500, mu = M1, Sigma = S2)
print(S2)
print(head(Points2))
```

```
     [,1] [,2]
[1,]  0.2  0.0
[2,]  0.0  0.2
           [,1]       [,2]
[1,]  0.1016754 -0.1271067
[2,] -0.3936517  0.5286970
[3,] -0.7629091  0.4022048
[4,] -0.2917314 -0.1280088
```

```
[5,] -0.1552292  0.1439541
[6,] -0.4875342 -0.2104379
```

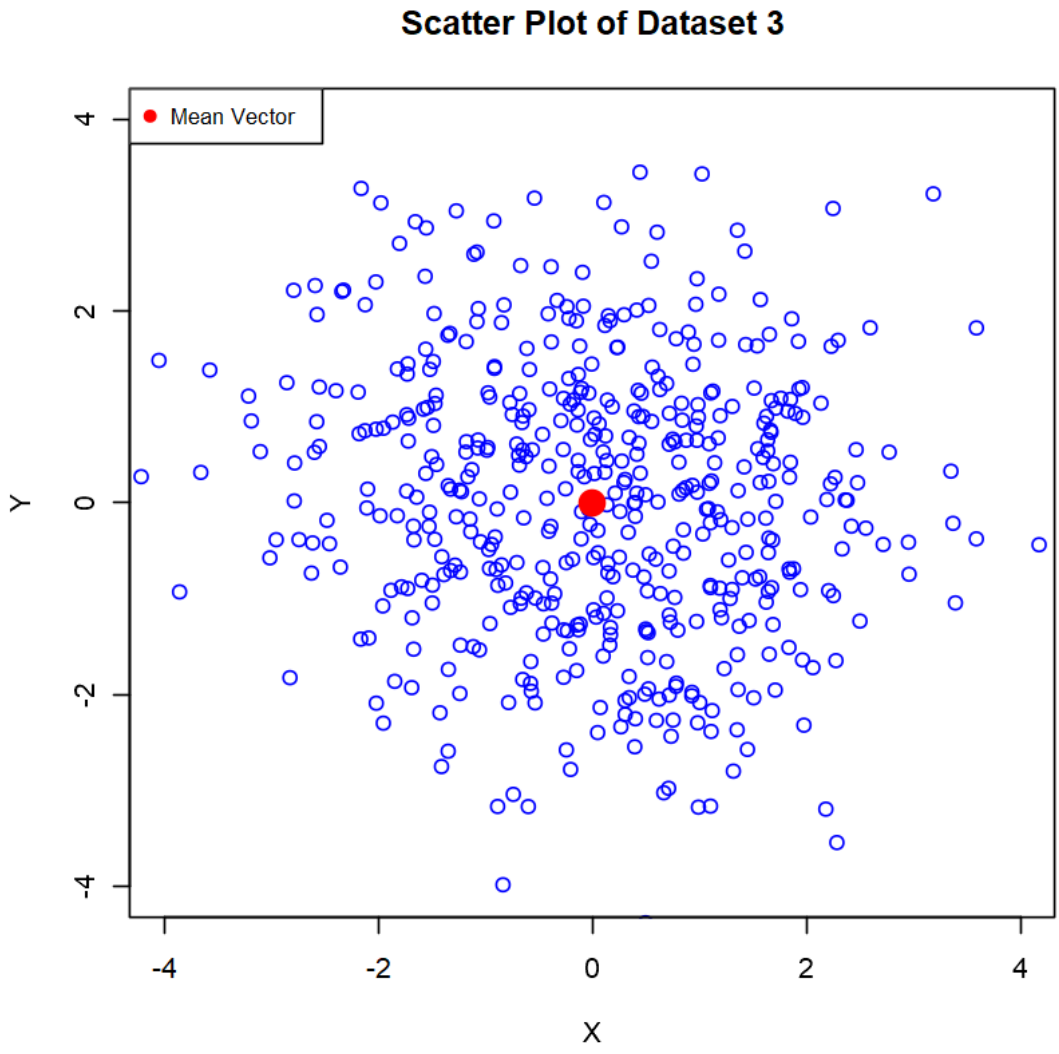
[85]:
```
plot(Points2, main = "Scatter Plot of Dataset 2", xlim = c(-4, 4), ylim = c(-4,␣
↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```



[86]:
```
S3 <- matrix(c(2,0,0,2), nrow=2)
Points3 <- mvrnorm(500, mu = M1, Sigma = S3)
print(S3)
print(head(Points3))
```

```
     [,1] [,2]
[1,]    2    0
[2,]    0    2
           [,1]       [,2]
[1,] -0.4635182  0.7154060
[2,]  0.5008808 -1.3148860
[3,]  0.7570534 -2.2640077
[4,] -4.2123980  0.2738468
[5,] -0.3780862 -1.0448444
[6,] -3.6555127  0.3181904
```

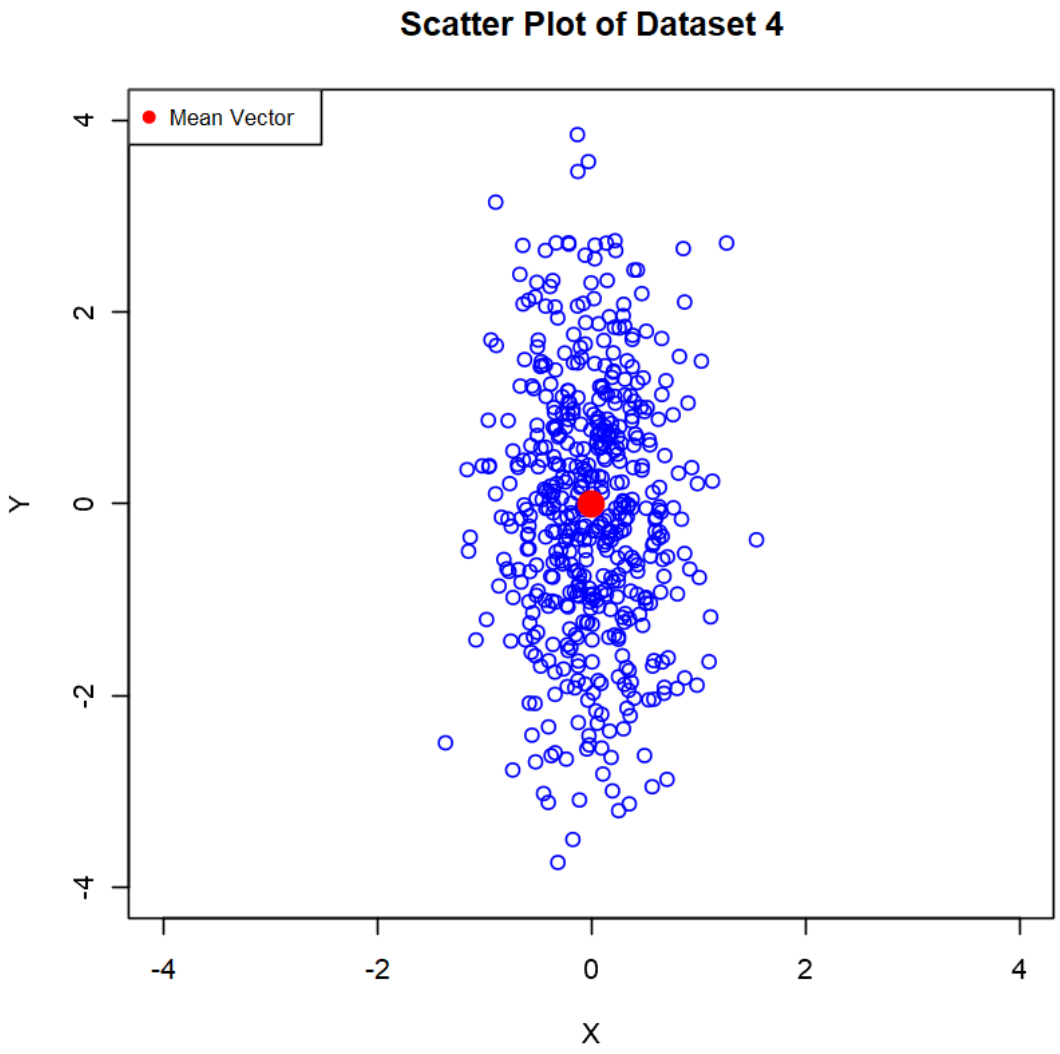
[87]:
```
plot(Points3, main = "Scatter Plot of Dataset 3", xlim = c(-4, 4), ylim = c(-4,␣
↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```



[88]:
```
S4 <- matrix(c(0.2,0,0,2), nrow=2)
Points4 <- mvrnorm(500, mu = M1, Sigma = S4)
print(S4)
print(head(Points4))
```

```
     [,1] [,2]
[1,]  0.2    0
[2,]  0.0    2
            [,1]         [,2]
[1,]  0.06291262  0.857275466
[2,] -0.62546682 -0.009902832
[3,]  0.36463711  0.998762676
[4,] -1.12973953 -0.346787424
[5,] -0.26518682  0.279199009
[6,] -0.89119613  3.146518210
```

[89]:
```
plot(Points4, main = "Scatter Plot of Dataset 4", xlim = c(-4, 4), ylim = c(-4,␣
↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```

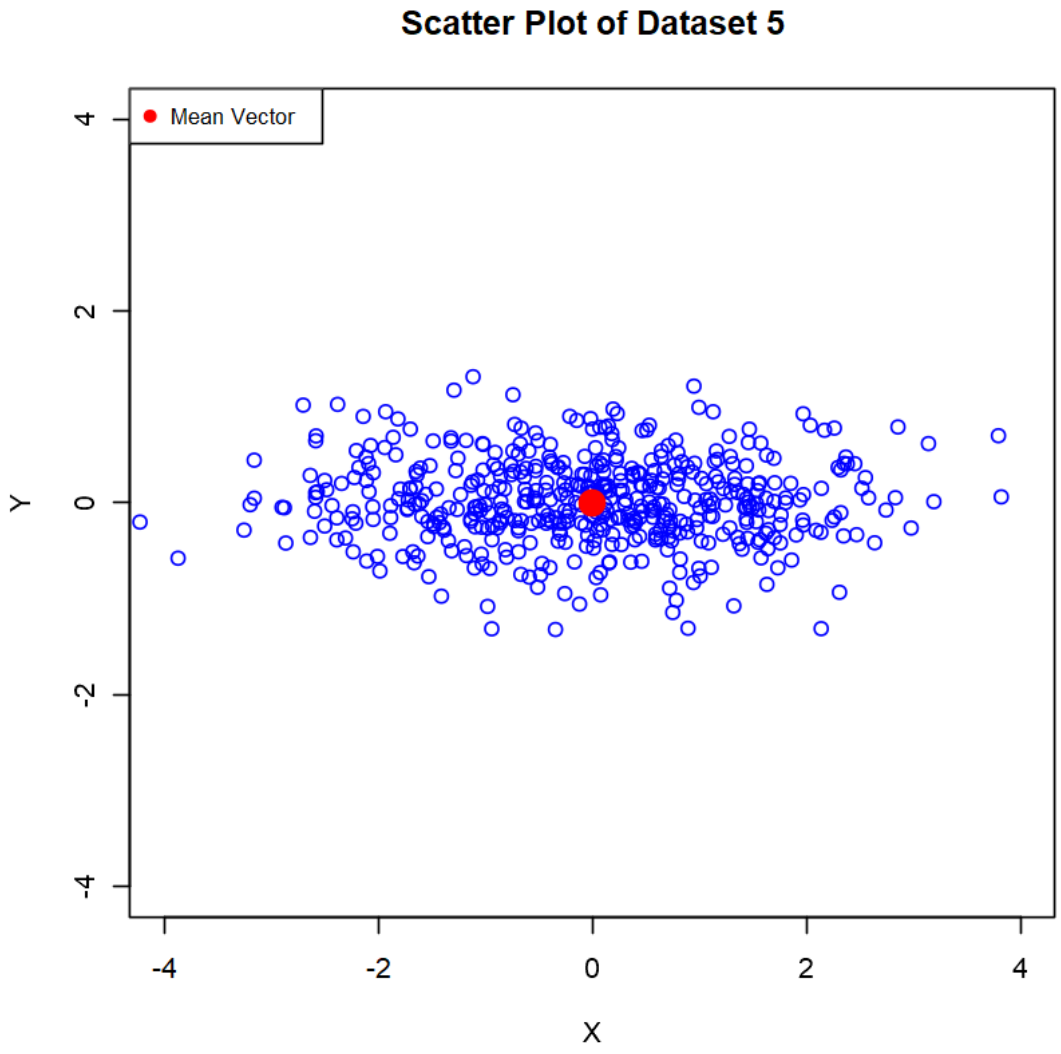


[90]:
```
S5 <- matrix(c(2,0,0,0.2), nrow=2)
Points5 <- mvrnorm(500, mu = M1, Sigma = S5)
print(S5)
print(head(Points5))
```

```
     [,1] [,2]
[1,]    2  0.0
[2,]    0  0.2
           [,1]        [,2]
[1,] -2.3039501 -0.36578991
[2,] -2.1055435 -0.60622449
[3,]  1.4397051  0.04300411
[4,]  0.6447086 -0.37385538
```

```
[5,]  0.1180446 -0.02264317
[6,] -2.0837985  0.11697629
```

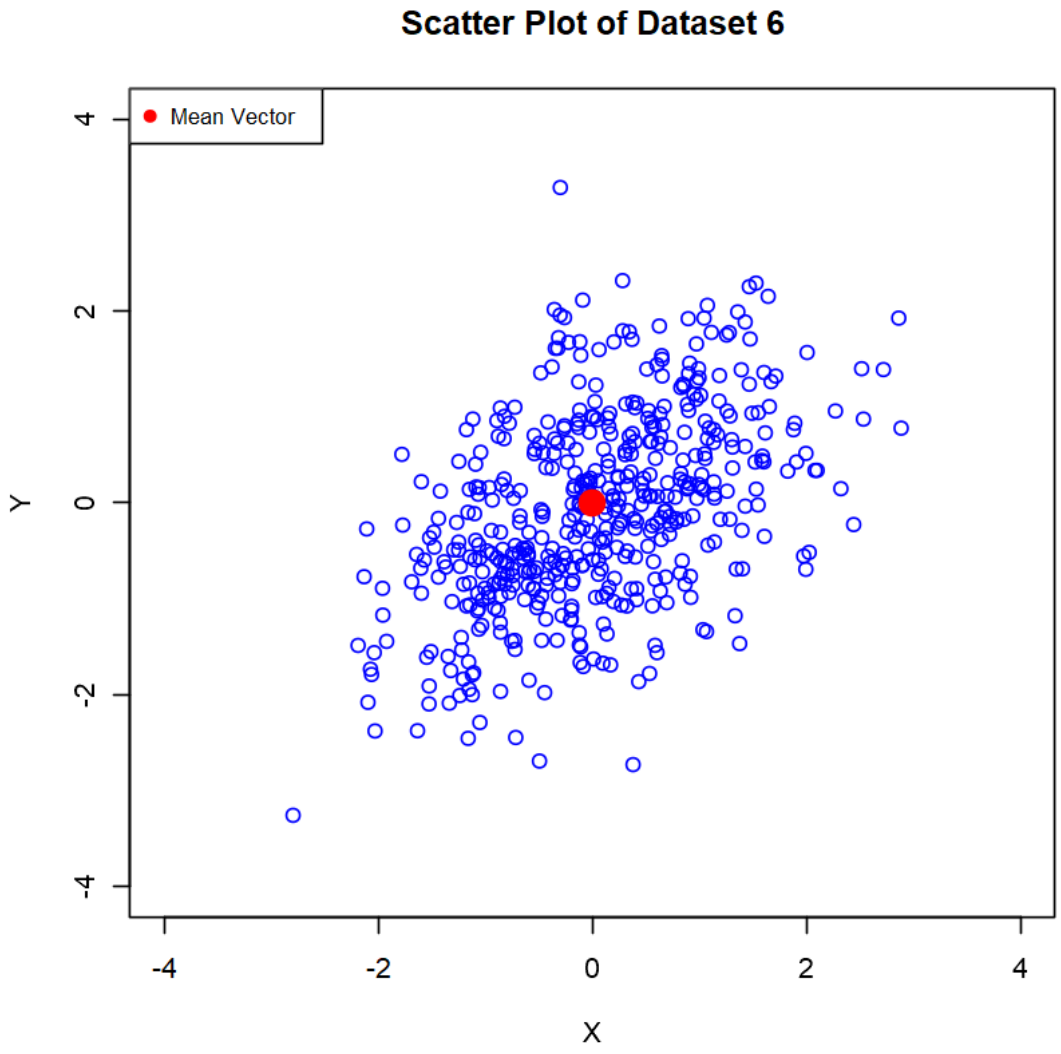
[91]:
```
plot(Points5, main = "Scatter Plot of Dataset 5", xlim = c(-4, 4), ylim = c(-4,␣
↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```



[92]:
```
S6 <- matrix(c(1,0.5,0.5,1), nrow=2)
Points6 <- mvrnorm(500, mu = M1, Sigma = S6)
print(S6)
print(head(Points6))
```

```
     [,1] [,2]
[1,]  1.0  0.5
[2,]  0.5  1.0
           [,1]       [,2]
[1,]  0.5115676  1.3962193
[2,] -0.9407322 -0.2867765
[3,] -0.7411472 -0.7571250
[4,] -0.8566726  0.9899235
[5,] -0.1191626 -1.4823683
[6,]  0.3079634 -0.5580352
```

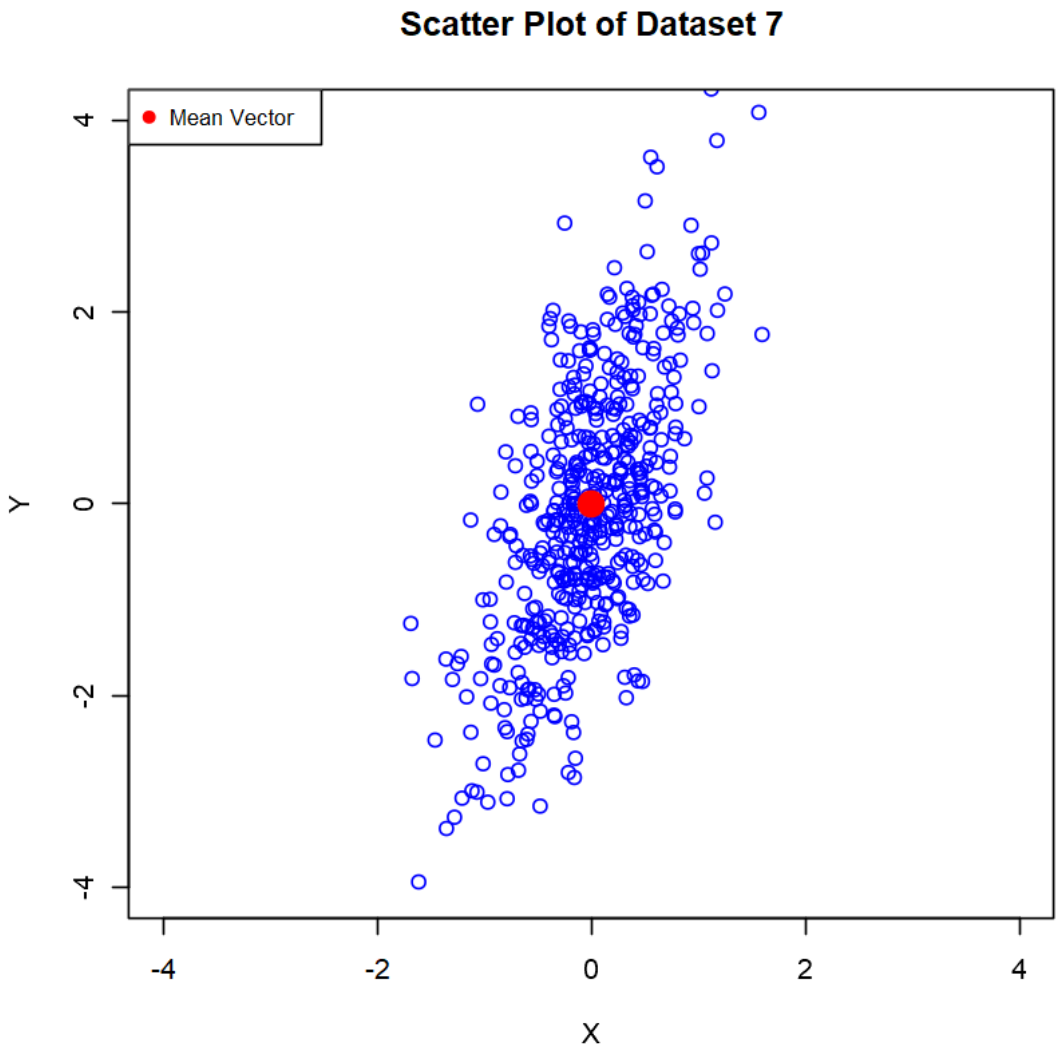
[93]:
```
plot(Points6, main = "Scatter Plot of Dataset 6", xlim = c(-4, 4), ylim = c(-4,␣
↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```



[94]:
```
S7 <- matrix(c(0.3,0.5,0.5,2), nrow=2)
Points7 <- mvrnorm(500, mu = M1, Sigma = S7)
print(S7)
print(head(Points7))
```

```
     [,1] [,2]
[1,]  0.3  0.5
[2,]  0.5  2.0
            [,1]       [,2]
[1,]  0.14112862  0.1425425
[2,] -0.33042437 -0.4140013
[3,]  0.02968122  0.1047217
[4,]  1.25009013  2.1894597
[5,]  0.07234505 -1.2233377
[6,]  0.57947157  1.5659738
```

[95]:
```
plot(Points7, main = "Scatter Plot of Dataset 7", xlim = c(-4, 4), ylim = c(-4,␣
 ↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```

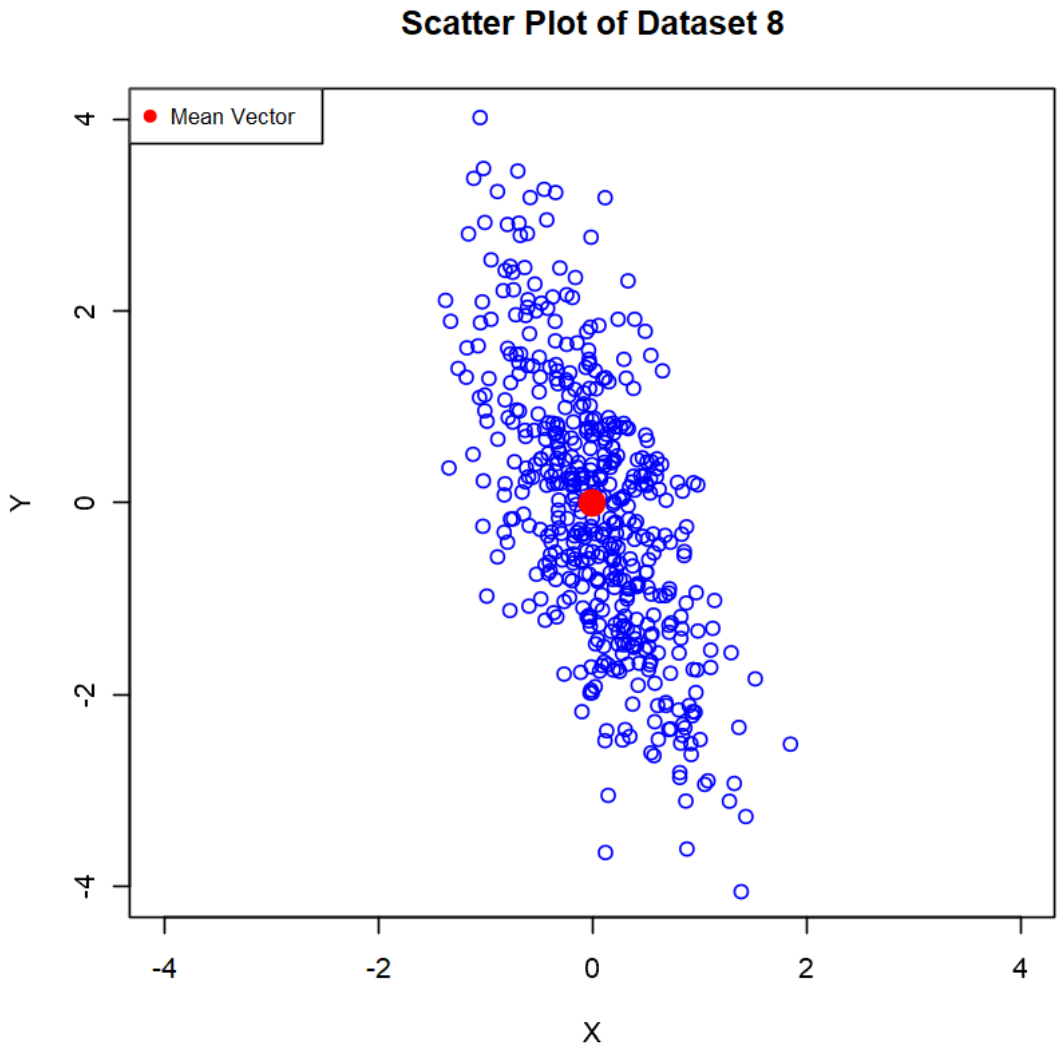


[96]:
```
S8 <- matrix(c(0.3,-0.5,-0.5,2), nrow=2)
Points8 <- mvrnorm(500, mu = M1, Sigma = S8)
print(S8)
print(head(Points8))
```

```
     [,1] [,2]
[1,]  0.3 -0.5
[2,] -0.5  2.0
             [,1]       [,2]
[1,]  0.0892250984 -0.9595301
[2,]  1.5240831936 -1.8345238
[3,]  0.2380326660  0.4920064
[4,] -0.3395291741 -0.7988097
```

```
[5,]  0.0009874955 -1.9802794
[6,] -0.7359841276 -0.1663711
```

[97]:
```
plot(Points8, main = "Scatter Plot of Dataset 8", xlim = c(-4, 4), ylim = c(-4,
↪4) ,
     xlab = "X", ylab = "Y", col = "blue")
points(M1[1], M1[2], col = "red", pch = 19, cex = 2)
legend("topleft", legend = "Mean Vector", col = "red", pch = 19, cex = 0.8)
```



0.4 Question 7

7.

Consider a 2-class classification task in the 2-dimensional space, where the data in both classes, ω_1, ω_2, are distributed according to the Gaussian distributions $\mathcal{N}(m_1, S_1)$ and $\mathcal{N}(m_2, S_2)$, respectively. Let

$$m_1 = [1,\ 1]^T, \quad m_2 = [3,\ 3]^T, \quad S_1 = S_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Assuming that $P(\omega_1) = P(\omega_2) = 1/2$, classify $x = [1.8,\ 1.8]^T$ into ω_1 or ω_2.

[29]:
```
library(mvtnorm)
w1_M <- matrix(c(1,1), nrow=1)
w1_S <- matrix(c(1,0,0,1), nrow=2)
w1_pdf <- function(x) {
  dmvnorm(x, mean = w1_M, sigma = w1_S)
}

x <- c(1.8,1.8)
print(w1_pdf(x))
```

```
[1] 0.0839212
```

[30]:
```
w2_M <- matrix(c(3,3), nrow=1)
w2_S <- matrix(c(1,0,0,1), nrow=2)
w2_pdf <- function(x) {
  dmvnorm(x, mean = w2_M, sigma = w2_S)
}

print(w2_pdf(x))
```

```
[1] 0.03770822
```

[31]:
```
w1_Posterior <- (w1_pdf(x)*0.5)/((w1_pdf(x)*0.5)+(w2_pdf(x)*0.5))
print(w1_Posterior)
```

```
[1] 0.6899745
```

[32]:
```
w2_Posterior <- (w2_pdf(x)*0.5)/((w1_pdf(x)*0.5)+(w2_pdf(x)*0.5))
print(w2_Posterior)
```

```
[1] 0.3100255
```

so x belongs to class w1 with the probability of 0.69

0.5 Question 8

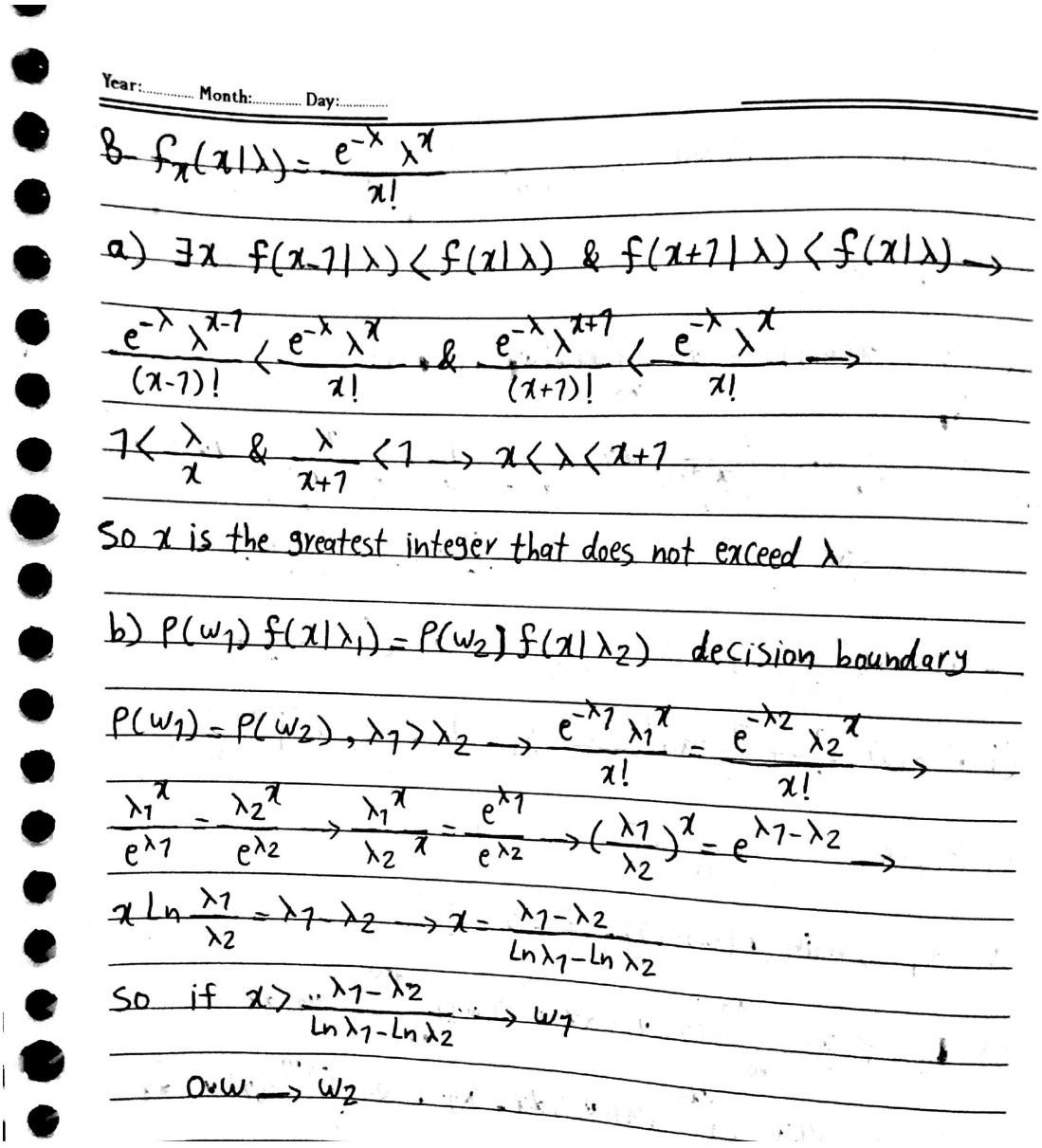
8.

The Poisson distribution for the discrete variable $x = 0, 1, 2, \ldots$ and real parameter λ is

$$f_x(x|\lambda) = \frac{e^{-\lambda}\lambda^x}{x!}$$

a) The *mode* of a distribution is the value of x that has the maximum probability. Prove that the mode of a Poisson distribution is the greatest integer that does not exceed λ, i.e. the mode is $[\lambda]$ (if λ is an integer, then both λ and $\lambda - 1$ are modes).

b) Consider two equally probable categories having Poisson distributions but with differing parameters; assume for definiteness $\lambda_1 > \lambda_2$. What is the Bayes classification decision?

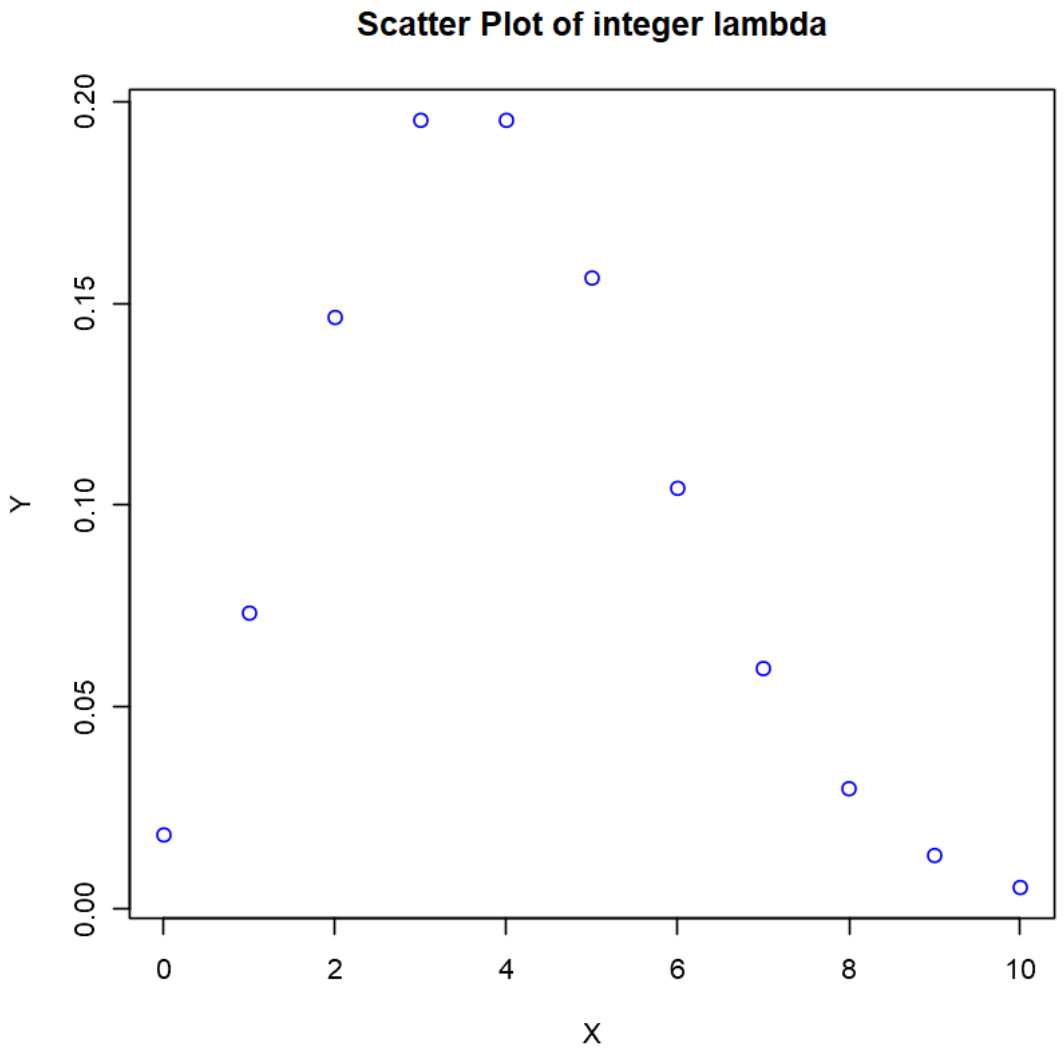


0.6 a)

[25]:
```
lambda <- 4
k <- 0:10
pmf <- dpois(k, lambda)
df <- data.frame(K = k, PMF = pmf)
print(df)
```

```
    K         PMF
1   0 0.018315639
2   1 0.073262556
3   2 0.146525111
4   3 0.195366815
5   4 0.195366815
6   5 0.156293452
7   6 0.104195635
8   7 0.059540363
9   8 0.029770181
10  9 0.013231192
11 10 0.005292477
```

[26]:
```
plot(df, main = "Scatter Plot of integer lambda",
     xlab = "X", ylab = "Y", col = "blue")
```

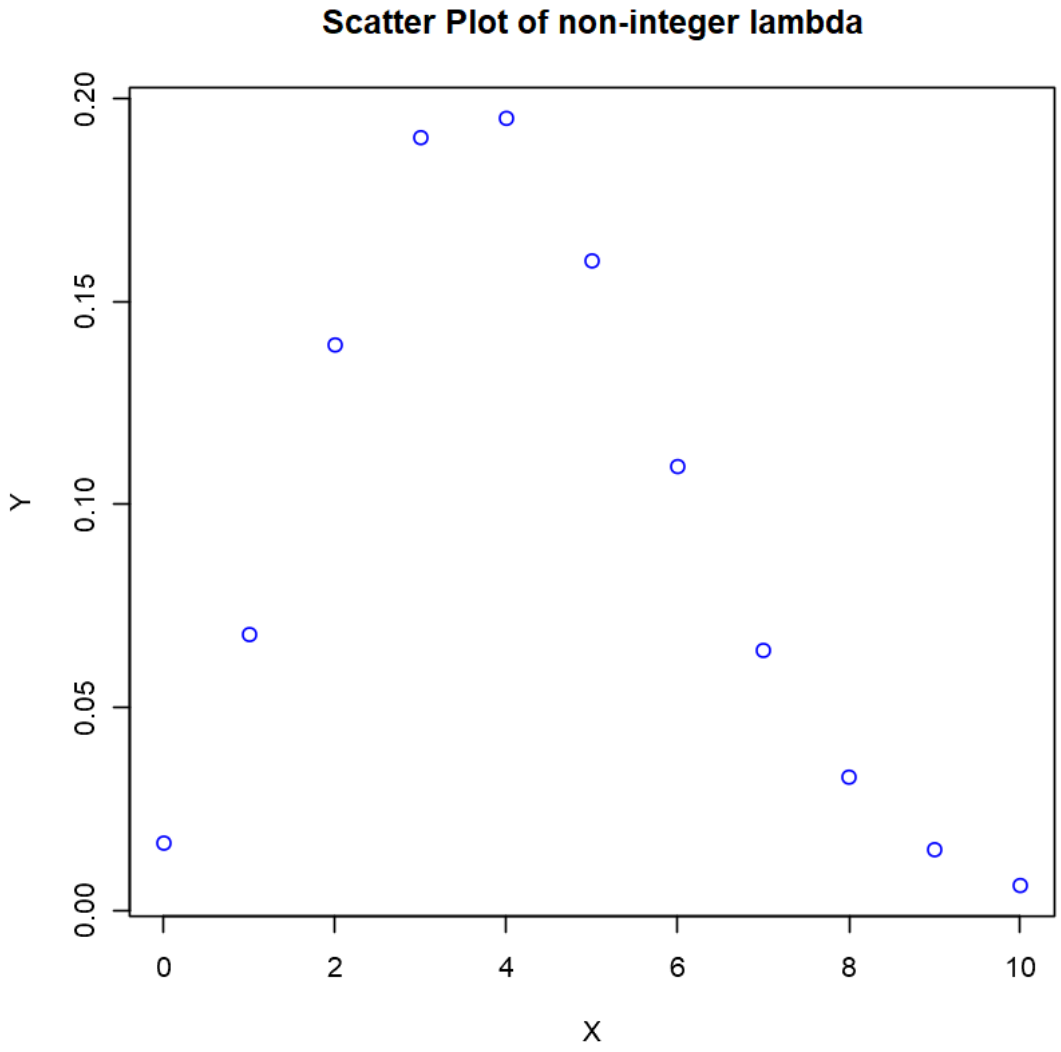


[27]:
```
lambda <- 4.1
k <- 0:10
pmf <- dpois(k, lambda)
df <- data.frame(K = k, PMF = pmf)
print(df)
```

```
    K         PMF
1   0 0.016572675
2   1 0.067947969
3   2 0.139293337
4   3 0.190367560
5   4 0.195126749
6   5 0.160003934
```

```
7    6 0.109336022
8    7 0.064039670
9    8 0.032820331
10   9 0.014951484
11  10 0.006130108
```

[28]:
```
plot(df, main = "Scatter Plot of non-integer lambda",
     xlab = "X", ylab = "Y", col = "blue")
```

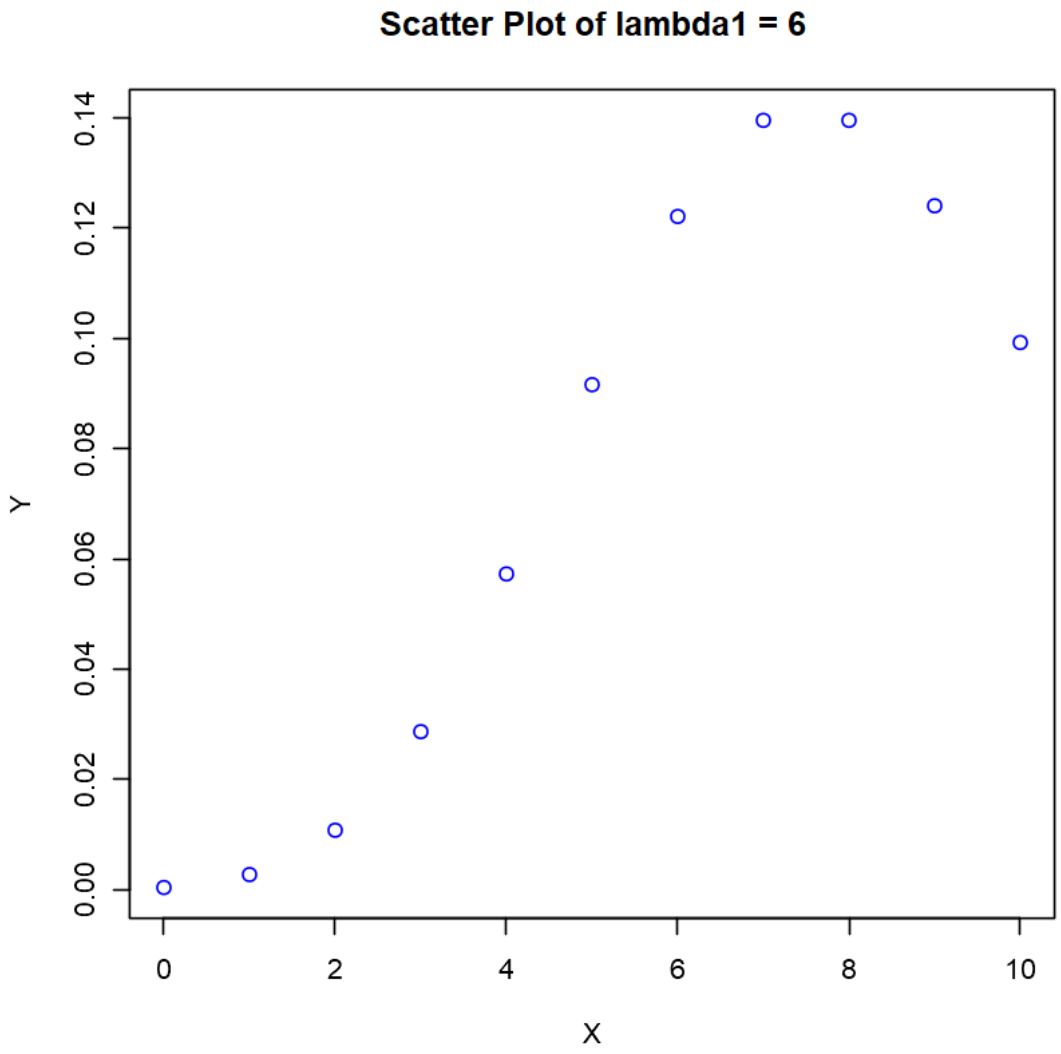


0.7 b)

[6]:
```
lambda <- 8
k <- 0:10
pmf <- dpois(k, lambda)
df <- data.frame(K = k, PMF = pmf)
print(df)
```

```
    K          PMF
1   0 0.0003354626
2   1 0.0026837010
3   2 0.0107348041
4   3 0.0286261442
5   4 0.0572522885
6   5 0.0916036616
7   6 0.1221382155
8   7 0.1395865320
9   8 0.1395865320
10  9 0.1240769173
11 10 0.0992615338
```

[7]:
```
plot(df, main = "Scatter Plot of lambda1 = 6",
     xlab = "X", ylab = "Y", col = "blue")
```

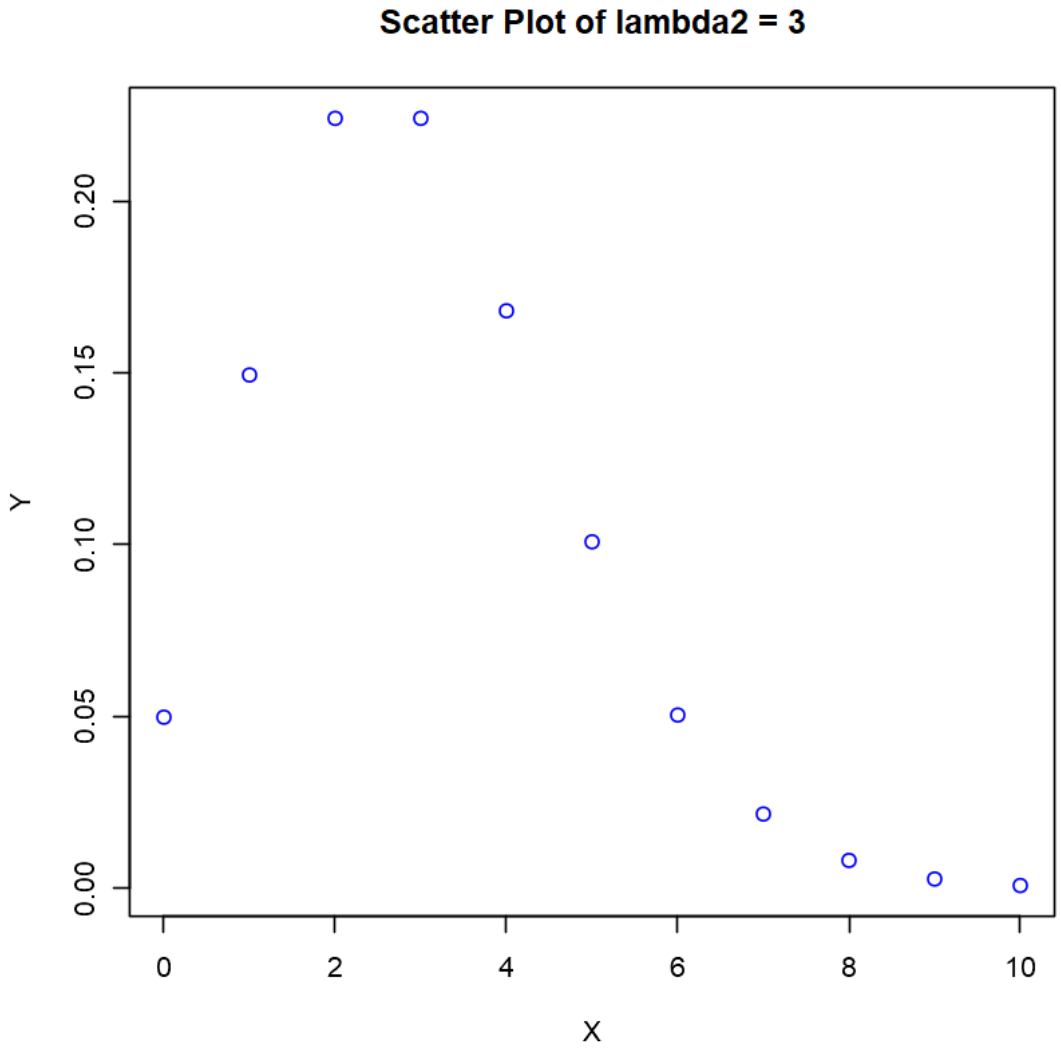


[4]:
```
lambda <- 3
k <- 0:10
pmf <- dpois(k, lambda)
df <- data.frame(K = k, PMF = pmf)
print(df)
```

```
    K          PMF
1   0 0.0497870684
2   1 0.1493612051
3   2 0.2240418077
4   3 0.2240418077
5   4 0.1680313557
6   5 0.1008188134
```

```
7   6 0.0504094067
8   7 0.0216040315
9   8 0.0081015118
10  9 0.0027005039
11 10 0.0008101512
```

[5]:
```
plot(df, main = "Scatter Plot of lambda2 = 3",
     xlab = "X", ylab = "Y", col = "blue")
```



0.8 Question 9

9.

Consider a 2-class classification task in the 3-dimensional space, where the two classes, ω_1 and ω_2, are modeled by Gaussian distributions with means $m_1 = [0, 0, 0]^T$ and $m_2 = [0.5, 0.5, 0.5]^T$, respectively. Assume the two classes to be equiprobable. The covariance matrix for both distributions is

$$S = \begin{bmatrix} 0.8 & 0.01 & 0.01 \\ 0.01 & 0.2 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$

Given the point $x = [0.1, 0.5, 0.1]^T$, classify x (1) according to the Euclidean distance classifier and (2) according to the Mahalanobis distance classifier. Comment on the results.

[19]:
```
library(mvtnorm)
w1_M9 <- matrix(c(0,0,0), nrow=1)
w1_S9 <- matrix(c(0.8,0.01,0.01,0.01,0.2,0.01,0.01,0.01,0.2), nrow=3)
print(w1_M9)
print(w1_S9)
```

```
     [,1] [,2] [,3]
[1,]    0    0    0
     [,1] [,2] [,3]
[1,] 0.80 0.01 0.01
[2,] 0.01 0.20 0.01
[3,] 0.01 0.01 0.20
```

[20]:
```
w2_M9 <- matrix(c(0.5,0.5,0.5), nrow=1)
w2_S9 <- matrix(c(0.8,0.01,0.01,0.01,0.2,0.01,0.01,0.01,0.2), nrow=3)
print(w2_M9)
print(w2_S9)
```

```
     [,1] [,2] [,3]
[1,]  0.5  0.5  0.5
     [,1] [,2] [,3]
[1,] 0.80 0.01 0.01
[2,] 0.01 0.20 0.01
[3,] 0.01 0.01 0.20
```

0.9 1)

[21]:
```
x9 <- c(0.1,0.5,0.1)
print(sqrt(sum((x9-w1_M9)^2)))
print(sqrt(sum((x9-w2_M9)^2)))
```

```
[1] 0.5196152
[1] 0.5656854
```

so based on Euclidean distance x9 belongs to class w1

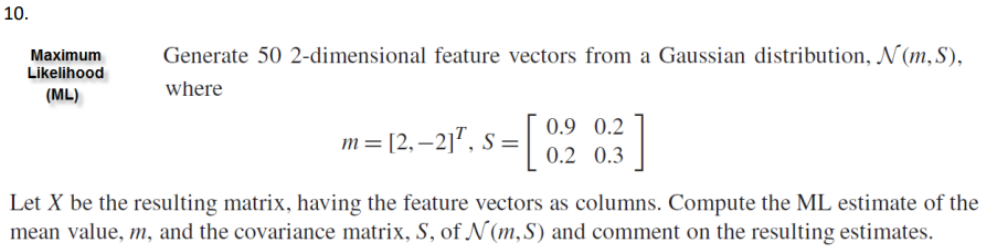
0.10 2)

[22]:
```
library(MASS)
print(sqrt(mahalanobis(x9,w1_M9,w1_S9)))
print(sqrt(mahalanobis(x9,w2_M9,w2_S9)))
#print(dmvnorm(x9, mean = w1_M9, sigma = w1_S9))
#print(dmvnorm(x9, mean = w2_M9, sigma = w2_S9))
```

```
[1] 1.133393
[1] 0.9917798
```

and based on Mahalanobis distance x9 belongs to class w2

0.11 Question 10


10.

Maximum Likelihood (ML)

Generate 50 2-dimensional feature vectors from a Gaussian distribution, $\mathcal{N}(m,S)$, where

$$m=[2,-2]^T,\ S=\begin{bmatrix}0.9 & 0.2\\ 0.2 & 0.3\end{bmatrix}$$

Let X be the resulting matrix, having the feature vectors as columns. Compute the ML estimate of the mean value, m, and the covariance matrix, S, of $\mathcal{N}(m,S)$ and comment on the resulting estimates.

[132]:
```
library(MASS)
M10 <- matrix(c(2,-2), nrow=1)
S10 <- matrix(c(0.9,0.2,0.2,0.3), nrow=2)
Points10 <- mvrnorm(50, mu = M10, Sigma = S10)
print(M10)
print(S10)
print(head(Points10))
```

```
     [,1] [,2]
[1,]    2   -2
     [,1] [,2]
[1,]  0.9  0.2
[2,]  0.2  0.3
          [,1]       [,2]
[1,] 2.7976172 -0.9488456
[2,] 1.8345226 -1.5258908
[3,] 2.2992249 -1.8094637
[4,] 0.7970184 -2.0116495
[5,] 2.0280877 -1.8098402
[6,] 1.2905783 -2.0429834
```

[141]:
```
NormalLikelihood <- function(data){
    n <- nrow(data)
    mu2 <- colMeans(data)
    sigma2 <- (1/n)*(((t(data))-colMeans(data))%*%(t((t(data))-colMeans(data))))
    print(mu2)
    print(sigma2)
}
NormalLikelihood(Points10)
```

```
[1]  2.062615 -1.989254
          [,1]      [,2]
[1,] 0.9469344 0.1983955
[2,] 0.1983955 0.2671389
```

0.12 Question 11

11.

Generate two data sets, X (training set) and X_1 (test set), each consisting of $N = 1000$ 3-dimensional vectors that stem from three *equiprobable* classes, ω_1, ω_2, and ω_3. The classes are modeled by Gaussian distributions with means $m_1 = [0, 0, 0]^T$, $m_2 = [1, 2, 2]^T$, and $m_3 = [3, 3, 4]^T$, respectively; their covariance matrices are

$$S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} = \sigma^2 I$$

1. Using X, compute the maximum likelihood estimates of the mean values and the covariance matrices of the distributions of the three classes. Since the covariance matrices are known to be the same, estimate them for each class and compute their average. Use the latter as the estimate of the (common) covariance matrix.

2. Use the Euclidean distance classifier to classify the points of X_1 based on the ML estimates computed before.
3. Use the Mahalanobis distance classifier to classify the points of X_1 based on the ML estimates computed before.
4. Use the Bayesian classifier to classify the points of X_1 based on the ML estimates computed before.
5. For each case, compute the error probability and compare the results (all classifiers should result in almost the same performance. Why?).

[2]:
```
library(MASS)
w1_M11 <- matrix(c(0,0,0), nrow=1)
w1_S11 <- matrix(c(0.8,0,0,0,0.8,0,0,0,0.8), nrow=3)
w1_Points11_train <- mvrnorm(333, mu = w1_M11, Sigma = w1_S11)
print(w1_M11)
print(w1_S11)
print(head(w1_Points11_train))

w1_Points11_test <- mvrnorm(333, mu = w1_M11, Sigma = w1_S11)
print(head(w1_Points11_test))
```

```
     [,1] [,2] [,3]
```

```
[1,]    0    0    0
     [,1] [,2] [,3]
[1,]  0.8  0.0  0.0
[2,]  0.0  0.8  0.0
[3,]  0.0  0.0  0.8
            [,1]        [,2]       [,3]
[1,]  0.85194640  0.08883733  0.2119193
[2,] -0.19390576  0.60984665  0.6339462
[3,] -0.82725549 -0.73782637 -0.3172135
[4,] -1.52150022  0.41869419  0.8046257
[5,]  1.36376429 -1.01716343 -0.3396704
[6,]  0.08906312 -0.32216654  0.5622013
              [,1]        [,2]        [,3]
[1,] -0.176117164 -1.19812736  0.23737524
[2,]  0.853739725  0.01115708  0.94498901
[3,] -1.033354697  0.45730307 -1.43029817
[4,] -1.784415614  0.10440920 -1.57347048
[5,] -0.001194338  0.21524330 -0.08678483
[6,] -0.275135680  0.31354951  0.62252943
```

[3]:
```
w2_M11 <- matrix(c(1,2,2), nrow=1)
w2_S11 <- matrix(c(0.8,0,0,0,0.8,0,0,0,0.8), nrow=3)
w2_Points11_train <- mvrnorm(333, mu = w2_M11, Sigma = w2_S11)
print(w2_M11)
print(w2_S11)
print(head(w2_Points11_train))

w2_Points11_test <- mvrnorm(333, mu = w2_M11, Sigma = w2_S11)
print(head(w2_Points11_test))
```

```
     [,1] [,2] [,3]
[1,]    1    2    2
     [,1] [,2] [,3]
[1,]  0.8  0.0  0.0
[2,]  0.0  0.8  0.0
[3,]  0.0  0.0  0.8
          [,1]     [,2]     [,3]
[1,]  1.0874808 2.536097 2.263046
[2,]  1.2971836 2.951047 2.340003
[3,]  0.2281794 3.068358 2.097299
[4,]  1.3694715 2.379903 2.359121
[5,]  0.4035289 2.139053 1.018011
[6,] -0.2342734 1.518827 1.764278
          [,1]      [,2]     [,3]
[1,] 0.2709757 3.0627582 2.087566
[2,] 1.5935304 1.3035981 1.669881
[3,] 0.6437138 1.4620350 1.177305
[4,] 1.4218419 0.7546898 2.388350
```

```
[5,] 2.6130674 1.4255920 1.181421
[6,] 0.1966079 0.6945686 4.401973
```

[4]:
```
w3_M11 <- matrix(c(3,3,4), nrow=1)
w3_S11 <- matrix(c(0.8,0,0,0,0.8,0,0,0,0.8), nrow=3)
w3_Points11_train <- mvrnorm(333, mu = w3_M11, Sigma = w3_S11)
print(w3_M11)
print(w3_S11)
print(head(w3_Points11_train))

w3_Points11_test <- mvrnorm(333, mu = w3_M11, Sigma = w3_S11)
print(head(w3_Points11_test))
```

```
     [,1] [,2] [,3]
[1,]    3    3    4
     [,1] [,2] [,3]
[1,]  0.8  0.0  0.0
[2,]  0.0  0.8  0.0
[3,]  0.0  0.0  0.8
         [,1]     [,2]     [,3]
[1,] 3.650695 3.319509 3.094934
[2,] 2.292893 4.242269 3.345690
[3,] 3.473753 4.794814 4.156321
[4,] 2.656222 3.316749 3.534988
[5,] 2.405442 3.822816 4.370387
[6,] 2.844364 3.136329 3.965378
         [,1]     [,2]     [,3]
[1,] 2.592534 1.752488 3.405521
[2,] 3.506219 3.870655 5.458959
[3,] 4.215144 2.692652 4.953835
[4,] 2.114042 3.679023 3.094189
[5,] 3.511140 2.415075 4.740310
[6,] 3.101703 3.619136 2.950230
```

0.13 1)

[5]:
```
NormalLikelihoodMean <- function(data){
    mu2 <- colMeans(data)
    return(mu2)
}

NormalLikelihoodCovariance <- function(data){
    n <- nrow(data)
    sigma2 <- (1/n)*(((t(data))-colMeans(data))%*%(t((t(data))-colMeans(data))))
    return(sigma2)
}
```

[6]:
```
print(NormalLikelihoodMean(w1_Points11_train))
print(NormalLikelihoodCovariance(w1_Points11_train))
```

```
[1] -0.04299272 -0.05321435 -0.06278969
            [,1]       [,2]        [,3]
[1,]  0.85581639 0.03543902 -0.07877294
[2,]  0.03543902 0.78746936  0.02531930
[3,] -0.07877294 0.02531930  0.78052210
```

[7]:
```
print(NormalLikelihoodMean(w2_Points11_train))
print(NormalLikelihoodCovariance(w2_Points11_train))
```

```
[1] 1.004392 1.955568 2.030564
             [,1]        [,2]        [,3]
[1,]  0.846051498 0.001578197 -0.13700258
[2,]  0.001578197 0.685424393  0.03091533
[3,] -0.137002580 0.030915325  0.76004206
```

[8]:
```
print(NormalLikelihoodMean(w3_Points11_train))
print(NormalLikelihoodCovariance(w3_Points11_train))
```

```
[1] 2.998755 3.029420 3.957850
            [,1]       [,2]        [,3]
[1,]  0.84304505 0.01729685 -0.03766544
[2,]  0.01729685 0.69952845  0.01087384
[3,] -0.03766544 0.01087384  0.76035183
```

[9]:
```
w1_mu11 <- NormalLikelihoodMean(w1_Points11_train)
w2_mu11 <- NormalLikelihoodMean(w2_Points11_train)
w3_mu11 <- NormalLikelihoodMean(w3_Points11_train)
ws_s11 <- (NormalLikelihoodCovariance(w1_Points11_train) +␣
 ↪NormalLikelihoodCovariance(w2_Points11_train) +␣
 ↪NormalLikelihoodCovariance(w3_Points11_train))/3
print(ws_s11)
```

```
            [,1]       [,2]        [,3]
[1,]  0.84830431 0.01810469 -0.08448032
[2,]  0.01810469 0.72414073  0.02236949
[3,] -0.08448032 0.02236949  0.76697200
```

0.14 2)

[75]:
```
Euclidean_D <- function(x,mu){
    dis <- sqrt(sum((x-mu)^2))
    return(dis)
}
```

[76]:
```
Euclidean_C <- c()
for (j in 1:nrow(w1_Points11_test)) {
```

```
  d1 <- Euclidean_D(w1_Points11_test[j,],w1_mu11)
  d2 <- Euclidean_D(w1_Points11_test[j,],w2_mu11)
  d3 <- Euclidean_D(w1_Points11_test[j,],w3_mu11)
  md <- min(c(d1,d2,d3))

  if (md == d1) {
    Euclidean_C <- append(Euclidean_C,1)
  }

  if (md == d2) {
    Euclidean_C <- append(Euclidean_C,2)
  }

  if (md == d3) {
    Euclidean_C <- append(Euclidean_C,3)
  }

}

print(Euclidean_C)
w1_Points11_test <- cbind((w1_Points11_test), Euclidean_C)
print(head(w1_Points11_test))
```

```
  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 1 1 1
 [75] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1
[112] 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 2 2 2 1 1 1 1 1
[149] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
[186] 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 1 1 1 1 1
[223] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[260] 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
                                          Euclidean_C
[1,] -0.176117164 -1.19812736  0.23737524           1
[2,]  0.853739725  0.01115708  0.94498901           1
[3,] -1.033354697  0.45730307 -1.43029817           1
[4,] -1.784415614  0.10440920 -1.57347048           1
[5,] -0.001194338  0.21524330 -0.08678483           1
[6,] -0.275135680  0.31354951  0.62252943           1
```

[77]:
```
Euclidean_C <- c()
for (j in 1:nrow(w2_Points11_test)) {
  d1 <- Euclidean_D(w2_Points11_test[j,],w1_mu11)
  d2 <- Euclidean_D(w2_Points11_test[j,],w2_mu11)
  d3 <- Euclidean_D(w2_Points11_test[j,],w3_mu11)
  md <- min(c(d1,d2,d3))
```

```
  if (md == d1) {
    Euclidean_C <- append(Euclidean_C,1)
  }

  if (md == d2) {
    Euclidean_C <- append(Euclidean_C,2)
  }

  if (md == d3) {
    Euclidean_C <- append(Euclidean_C,3)
  }

}

print(Euclidean_C)
w2_Points11_test <- cbind((w2_Points11_test), Euclidean_C)
print(head(w2_Points11_test))
```

```
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 3 2 2 2 2 2 2 2 1 2 3 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
[112] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 3 2 2 2 1 2 2 2
[149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 3 2 2 2
[260] 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 3 2 3 1 2 2 3 2 2 2 2 2
[297] 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 2
```

```
                                  Euclidean_C
[1,] 0.2709757 3.0627582 2.087566           2
[2,] 1.5935304 1.3035981 1.669881           2
[3,] 0.6437138 1.4620350 1.177305           2
[4,] 1.4218419 0.7546898 2.388350           2
[5,] 2.6130674 1.4255920 1.181421           2
[6,] 0.1966079 0.6945686 4.401973           2
```

[78]:
```
Euclidean_C <- c()
for (j in 1:nrow(w3_Points11_test)) {
  d1 <- Euclidean_D(w3_Points11_test[j,],w1_mu11)
  d2 <- Euclidean_D(w3_Points11_test[j,],w2_mu11)
  d3 <- Euclidean_D(w3_Points11_test[j,],w3_mu11)
  md <- min(c(d1,d2,d3))

  if (md == d1) {
    Euclidean_C <- append(Euclidean_C,1)
  }

  if (md == d2) {
```

```
    Euclidean_C <- append(Euclidean_C,2)
  }

  if (md == d3) {
    Euclidean_C <- append(Euclidean_C,3)
  }

}

print(Euclidean_C)
w3_Points11_test <- cbind((w3_Points11_test), Euclidean_C)
print(head(w3_Points11_test))
```

```
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[186] 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3 2 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3
[223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
[260] 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[297] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 2 3 3
                                 Euclidean_C
[1,] 2.592534 1.752488 3.405521           3
[2,] 3.506219 3.870655 5.458959           3
[3,] 4.215144 2.692652 4.953835           3
[4,] 2.114042 3.679023 3.094189           3
[5,] 3.511140 2.415075 4.740310           3
[6,] 3.101703 3.619136 2.950230           3
```

0.15 3)

[79]:
```
library(MASS)
Mahalanobis_D <- function(x,mu,sigma){
    dis <- sqrt(mahalanobis(x,mu,sigma))
    return(dis)
}
#w1_Points11_test <- w1_Points11_test[,1:3]
#w2_Points11_test <- w2_Points11_test[,1:3]
#w3_Points11_test <- w3_Points11_test[,1:3]
```

[80]:
```
Mahalanobis_C <- c()
for (j in 1:nrow(w1_Points11_test)) {
  d1 <- Mahalanobis_D(w1_Points11_test[j,1:3],w1_mu11,ws_s11)
  d2 <- Mahalanobis_D(w1_Points11_test[j,1:3],w2_mu11,ws_s11)
  d3 <- Mahalanobis_D(w1_Points11_test[j,1:3],w3_mu11,ws_s11)
  md <- min(c(d1,d2,d3))
```

```
  if (md == d1) {
    Mahalanobis_C <- append(Mahalanobis_C,1)
  }

  if (md == d2) {
    Mahalanobis_C <- append(Mahalanobis_C,2)
  }

  if (md == d3) {
    Mahalanobis_C <- append(Mahalanobis_C,3)
  }

}

print(Mahalanobis_C)
w1_Points11_test <- cbind((w1_Points11_test), Mahalanobis_C)
print(head(w1_Points11_test))
```

```
  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 1 1 1
 [75] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1
[112] 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 2 2 2 1 1 1 1 1
[149] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 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 1 1 1 1 1
[223] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[260] 1 2 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
[297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
                                          Euclidean_C Mahalanobis_C
[1,] -0.176117164 -1.19812736  0.23737524           1              1
[2,]  0.853739725  0.01115708  0.94498901           1              1
[3,] -1.033354697  0.45730307 -1.43029817           1              1
[4,] -1.784415614  0.10440920 -1.57347048           1              1
[5,] -0.001194338  0.21524330 -0.08678483           1              1
[6,] -0.275135680  0.31354951  0.62252943           1              1
```

[81]:
```
Mahalanobis_C <- c()
for (j in 1:nrow(w2_Points11_test)) {
  d1 <- Mahalanobis_D(w2_Points11_test[j,1:3],w1_mu11,ws_s11)
  d2 <- Mahalanobis_D(w2_Points11_test[j,1:3],w2_mu11,ws_s11)
  d3 <- Mahalanobis_D(w2_Points11_test[j,1:3],w3_mu11,ws_s11)
  md <- min(c(d1,d2,d3))

  if (md == d1) {
    Mahalanobis_C <- append(Mahalanobis_C,1)
  }
```

```
  if (md == d2) {
    Mahalanobis_C <- append(Mahalanobis_C,2)
  }

  if (md == d3) {
    Mahalanobis_C <- append(Mahalanobis_C,3)
  }

}

print(Mahalanobis_C)
w2_Points11_test <- cbind((w2_Points11_test), Mahalanobis_C)
print(head(w2_Points11_test))
```

```
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 3 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2
 [38] 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
[112] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 3 2 2 2 1 2 2 2
[149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 3 2 2 2
[260] 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 3 2 3 1 2 2 3 2 2 2 2 2
[297] 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 2
                                     Euclidean_C Mahalanobis_C
[1,] 0.2709757 3.0627582 2.087566           2             2
[2,] 1.5935304 1.3035981 1.669881           2             2
[3,] 0.6437138 1.4620350 1.177305           2             2
[4,] 1.4218419 0.7546898 2.388350           2             2
[5,] 2.6130674 1.4255920 1.181421           2             2
[6,] 0.1966079 0.6945686 4.401973           2             2
```

[82]:
```
Mahalanobis_C <- c()
for (j in 1:nrow(w3_Points11_test)) {
  d1 <- Mahalanobis_D(w3_Points11_test[j,1:3],w1_mu11,ws_s11)
  d2 <- Mahalanobis_D(w3_Points11_test[j,1:3],w2_mu11,ws_s11)
  d3 <- Mahalanobis_D(w3_Points11_test[j,1:3],w3_mu11,ws_s11)
  md <- min(c(d1,d2,d3))

  if (md == d1) {
    Mahalanobis_C <- append(Mahalanobis_C,1)
  }

  if (md == d2) {
    Mahalanobis_C <- append(Mahalanobis_C,2)
  }

  if (md == d3) {
```

```
    Mahalanobis_C <- append(Mahalanobis_C,3)
  }

}

print(Mahalanobis_C)
w3_Points11_test <- cbind((w3_Points11_test), Mahalanobis_C)
print(head(w3_Points11_test))
```

```
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[186] 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3 3 2 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3
[223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
[260] 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[297] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 2 3 3
                                 Euclidean_C Mahalanobis_C
[1,] 2.592534 1.752488 3.405521           3             3
[2,] 3.506219 3.870655 5.458959           3             3
[3,] 4.215144 2.692652 4.953835           3             3
[4,] 2.114042 3.679023 3.094189           3             3
[5,] 3.511140 2.415075 4.740310           3             3
[6,] 3.101703 3.619136 2.950230           3             3
```

0.16 4)

[89]:
```
Class <- c()
for (j in 1:nrow(w1_Points11_train)){
    Class <- append(Class,1)
}
w1_Points11_train <- cbind((w1_Points11_train[,1:3]), Class)
print(head(w1_Points11_train))
```

```
                                      Class
[1,]  0.85194640  0.08883733  0.2119193     1
[2,] -0.19390576  0.60984665  0.6339462     1
[3,] -0.82725549 -0.73782637 -0.3172135     1
[4,] -1.52150022  0.41869419  0.8046257     1
[5,]  1.36376429 -1.01716343 -0.3396704     1
[6,]  0.08906312 -0.32216654  0.5622013     1
```

[91]:
```
Class <- c()
for (j in 1:nrow(w2_Points11_train)){
    Class <- append(Class,2)
}
w2_Points11_train <- cbind((w2_Points11_train[,1:3]), Class)
```

```
print(head(w2_Points11_train))
```

```
                                  Class
[1,]  1.0874808 2.536097 2.263046     2
[2,]  1.2971836 2.951047 2.340003     2
[3,]  0.2281794 3.068358 2.097299     2
[4,]  1.3694715 2.379903 2.359121     2
[5,]  0.4035289 2.139053 1.018011     2
[6,] -0.2342734 1.518827 1.764278     2
```

[93]:
```
Class <- c()
for (j in 1:nrow(w3_Points11_train)){
    Class <- append(Class,3)
}
w3_Points11_train <- cbind((w3_Points11_train[,1:3]), Class)
print(head(w3_Points11_train))
```

```
                                Class
[1,] 3.650695 3.319509 3.094934     3
[2,] 2.292893 4.242269 3.345690     3
[3,] 3.473753 4.794814 4.156321     3
[4,] 2.656222 3.316749 3.534988     3
[5,] 2.405442 3.822816 4.370387     3
[6,] 2.844364 3.136329 3.965378     3
```

[103]:
```
X_train <- rbind(w1_Points11_train,w2_Points11_train,w3_Points11_train)
X_train= X_train[sample(1:nrow(X_train)), ]
print(head(X_train))
```

```
                                   Class
[1,]  3.1935160 4.4110308  5.0243840     3
[2,] -1.7256263 0.1878608 -0.4110372     1
[3,]  0.4052086 1.0296290  1.4955518     2
[4,]  2.1261638 0.8222232  2.1790025     2
[5,]  1.9652315 2.6670102  5.2098708     3
[6,]  3.1364735 3.5276999  3.0377531     3
```

[97]:
```
Class <- c()
for (j in 1:nrow(w1_Points11_test)){
    Class <- append(Class,1)
}
w1_Points11_test <- cbind((w1_Points11_test[,1:5]), Class)
print(head(w1_Points11_test))
```

```
                                        Euclidean_C Mahalanobis_C Class
[1,] -0.176117164 -1.19812736  0.23737524           1             1     1
[2,]  0.853739725  0.01115708  0.94498901           1             1     1
[3,] -1.033354697  0.45730307 -1.43029817           1             1     1
[4,] -1.784415614  0.10440920 -1.57347048           1             1     1
```

```
[5,] -0.001194338  0.21524330 -0.08678483           1             1     1
[6,] -0.275135680  0.31354951  0.62252943           1             1     1
```

[98]:
```
Class <- c()
for (j in 1:nrow(w2_Points11_test)){
    Class <- append(Class,2)
}
w2_Points11_test <- cbind((w2_Points11_test[,1:5]), Class)
print(head(w2_Points11_test))
```

```
                                Euclidean_C Mahalanobis_C Class
[1,] 0.2709757 3.0627582 2.087566           2             2     2
[2,] 1.5935304 1.3035981 1.669881           2             2     2
[3,] 0.6437138 1.4620350 1.177305           2             2     2
[4,] 1.4218419 0.7546898 2.388350           2             2     2
[5,] 2.6130674 1.4255920 1.181421           2             2     2
[6,] 0.1966079 0.6945686 4.401973           2             2     2
```

[99]:
```
Class <- c()
for (j in 1:nrow(w3_Points11_test)){
    Class <- append(Class,3)
}
w3_Points11_test <- cbind((w3_Points11_test[,1:5]), Class)
print(head(w3_Points11_test))
```

```
                              Euclidean_C Mahalanobis_C Class
[1,] 2.592534 1.752488 3.405521           3             3     3
[2,] 3.506219 3.870655 5.458959           3             3     3
[3,] 4.215144 2.692652 4.953835           3             3     3
[4,] 2.114042 3.679023 3.094189           3             3     3
[5,] 3.511140 2.415075 4.740310           3             3     3
[6,] 3.101703 3.619136 2.950230           3             3     3
```

[104]:
```
X_test <- rbind(w1_Points11_test,w2_Points11_test,w3_Points11_test)
X_test= X_test[sample(1:nrow(X_test)), ]
print(head(X_test))
```

```
                                    Euclidean_C Mahalanobis_C Class
[1,]  1.1816532  0.71757131  2.5672068           2             2     2
[2,] -1.4309453  0.08633006 -1.6426835           1             1     1
[3,]  1.5988318 -0.66691118  0.7721132           1             1     1
[4,]  3.0676229  3.63089447  3.0470177           3             3     3
[5,] -0.5386412 -0.54097102  1.1809247           1             1     1
[6,]  1.0318869  2.24537844  1.1670985           2             2     2
```

[139]:
```
library(mvtnorm)
Y_train <- X_train[,4]
PriorProb <- table(Y_train) / length(Y_train)
print(PriorProb)
```

```r
predictions <- rep(NA, nrow(X_test))
X_test2 <- X_test[,1:3]

for (i in 1:nrow(X_test2)) {
  LogLikelihoods <- rep(0, length(unique(Y_train)))

  for (j in 1:length(unique(Y_train))) {
    if (j==1){
    Likelihood <- dmvnorm(X_test2[i,], mean = w1_mu11, sigma = ws_s11)}
    if (j==2){
    Likelihood <- dmvnorm(X_test2[i,], mean = w2_mu11, sigma = ws_s11)}
    if (j==3){
    Likelihood <- dmvnorm(X_test2[i,], mean = w3_mu11, sigma = ws_s11)}

    LogLikelihoods[j] <- log(Likelihood) + log(PriorProb[j])
  }
  predictions[i] <- (which.max(LogLikelihoods))
}

print(predictions)
```

```
Y_train
        1         2         3
0.3333333 0.3333333 0.3333333
  [1] 2 1 1 3 1 2 2 1 3 3 2 2 3 1 3 1 3 3 3 3 2 1 3 3 3 2 2 3 2 1 2 2 3 1 1 3 3
 [38] 1 3 3 2 1 2 1 1 3 2 2 3 3 3 1 2 3 2 1 2 3 2 1 3 1 2 1 3 1 1 1 2 2 2 1 3 3
 [75] 1 1 2 2 2 3 3 2 2 2 1 1 3 1 1 2 3 2 1 1 2 3 3 2 2 2 3 1 1 1 1 1 3 1 1 1 3
[112] 1 1 2 2 2 1 2 1 2 1 3 2 1 2 3 3 3 2 2 3 3 1 3 2 2 1 2 3 2 2 3 2 3 1 3 3 1
[149] 1 2 1 3 2 1 2 2 3 3 2 2 1 1 1 2 2 1 2 1 3 3 3 3 3 1 3 3 3 2 1 1 1 1 1 2 3
[186] 2 3 3 2 1 2 1 1 1 1 3 3 3 2 2 2 1 3 3 2 3 2 2 1 2 3 3 3 2 3 1 2 3 1 1 1 2
[223] 2 1 1 2 3 1 3 3 1 1 3 3 2 2 2 1 3 3 3 2 3 2 2 3 3 1 1 3 3 2 3 1 1 3 2 2 1
[260] 2 2 2 2 1 1 1 1 1 3 3 3 3 1 2 2 3 3 1 2 1 3 3 3 3 1 3 3 1 1 1 2 1 3 3 2 2
[297] 2 1 3 1 2 2 1 3 1 1 2 1 1 1 2 3 3 3 3 3 3 3 3 3 3 1 3 2 3 2 2 2 3 2 2 2 2 1
[334] 2 3 3 2 1 2 3 2 1 3 3 1 1 1 1 2 1 1 2 3 2 2 3 1 3 1 3 3 3 2 1 2 3 2 2 3 3
[371] 2 3 3 1 3 2 1 2 2 2 2 2 1 3 3 1 2 3 2 3 2 1 1 2 2 1 2 1 1 2 2 1 2 1 1 1 2
[408] 2 1 1 3 1 1 1 2 3 2 1 1 1 3 2 1 2 2 3 3 3 3 2 3 1 3 3 2 2 3 1 2 3 1 1 3 3
[445] 2 2 1 3 1 3 2 2 1 2 1 1 2 2 1 3 2 2 2 3 2 3 2 3 1 2 3 3 2 2 3 3 3 3 3 2 3
[482] 3 2 2 3 1 2 1 3 1 2 1 2 2 1 3 3 2 2 1 3 3 1 1 3 2 2 1 2 2 1 1 2 2 3 1 2 2
[519] 3 1 3 3 2 1 3 2 2 3 2 3 3 3 3 1 2 1 2 2 3 1 1 2 1 3 1 2 2 2 2 2 3 2 2 1 1
[556] 1 2 3 1 1 3 3 1 2 2 1 2 3 3 3 1 2 2 2 3 2 1 2 3 1 1 2 2 1 1 1 3 2 1 1 3 1
[593] 3 3 3 1 2 3 1 3 3 2 3 2 3 2 2 3 1 2 3 3 2 1 3 2 3 1 2 3 3 2 2 3 3 1 1 3 3
[630] 1 3 3 3 2 3 1 2 3 2 2 1 3 3 2 2 2 3 2 1 3 1 3 3 2 1 3 2 1 2 1 3 2 1 2 1 2
[667] 3 3 1 3 3 2 2 1 1 1 3 2 2 2 1 3 1 2 3 2 3 3 2 2 1 2 2 1 2 1 3 1 3 1 3 2 3
[704] 2 1 3 2 3 1 3 1 2 1 3 3 2 2 1 1 2 1 3 3 3 2 1 2 3 2 3 1 1 3 2 2 1 2 3 1 3
[741] 3 1 3 1 2 2 1 2 3 1 2 1 2 3 3 2 3 1 3 2 2 2 2 3 2 3 1 1 3 1 1 3 1 2 2 3 3
[778] 1 1 2 2 2 2 2 2 1 2 2 2 2 3 3 1 1 2 1 2 1 1 1 2 1 1 1 2 1 2 2 3 1 2 1 3 1
[815] 3 2 1 3 3 1 1 3 2 3 1 1 1 2 3 2 2 1 2 1 2 2 3 1 1 1 2 3 1 3 2 3 1 1 3 3 3
```

```
[852] 1 1 2 2 3 1 1 3 3 2 3 3 2 3 1 1 3 1 2 2 1 1 3 3 1 3 1 2 1 1 2 3 2 3 3 3 1
[889] 2 1 3 3 1 2 1 2 1 2 2 2 3 1 3 3 1 2 1 1 2 3 1 3 1 1 1 2 1 3 1 1 3 2 1 2 2
[926] 1 1 3 3 1 1 2 3 2 1 3 1 1 3 2 1 1 2 1 1 1 3 3 1 2 3 1 2 3 2 3 2 2 1 3 2 2
[963] 3 2 2 1 3 3 2 3 1 2 3 3 1 2 2 1 2 3 3 3 2 1 3 2 3 2 3 1 1 2 2 2 2 1 3 3 2
```

[140]:
```
Bayesian_C <- predictions
X_test <- cbind((X_test[,1:6]), Bayesian_C)
print(head(X_test))
```

```
                                    Euclidean_C Mahalanobis_C Class
[1,]  1.1816532  0.71757131  2.5672068           2             2     2
[2,] -1.4309453  0.08633006 -1.6426835           1             1     1
[3,]  1.5988318 -0.66691118  0.7721132           1             1     1
[4,]  3.0676229  3.63089447  3.0470177           3             3     3
[5,] -0.5386412 -0.54097102  1.1809247           1             1     1
[6,]  1.0318869  2.24537844  1.1670985           2             2     2
     Bayesian_C
[1,]          2
[2,]          1
[3,]          1
[4,]          3
[5,]          1
[6,]          2
```

0.17 5)

[163]:
```
Euclidean_E <- length(X_test[X_test[,4]==X_test[,6]]) / (ncol(X_test) *␣
 ↪nrow(X_test))
print(Euclidean_E)
Mahalanobis_E <- length(X_test[X_test[,5]==X_test[,6]]) / (ncol(X_test) *␣
 ↪nrow(X_test))
print(Mahalanobis_E)
Bayesian_E <- length(X_test[X_test[,7]==X_test[,6]]) / (ncol(X_test) *␣
 ↪nrow(X_test))
print(Bayesian_E)
```

```
[1] 0.9349349
[1] 0.9339339
[1] 0.9339339
```

As we can observe, the accuracy of each of these three classifiers is nearly identical. This could be attributed to the covariance matrix being diagonal, resulting in all covariances between features being zero. This can cause Euclidean and Mahalanobis distances to be similar. Additionally, the Naive Bayes classifier relies on the independence among features that we have in this question. All of these factors can lead to these algorithms working similarly.

0.18 Question 12

Generate a set X of $N = 500$ 2-dimensional points that stem from the following pdf:

$$p(x) = \sum_{j=1}^{J=3} P_j p(x|j)$$

where the $p(x|j)$'s, $j = 1,2,3$ are (2-dimensional) normal distributions with mean values $m_1 = [1,\ 1]^T$, $m_2 = [3,\ 3]^T$, $m_3 = [2,\ 6]^T$ and covariance matrices $S_1 = 0.1I$, $S_2 = 0.2I$, $S_3 = 0.3I$, respectively (I is the 2×2 identity matrix). In addition, $P_1 = 0.4$, $P_2 = 0.4$, and $P_3 = 0.2$.

The idea is to use the previously generated data and pretend that we do not know how they were generated. We assume that the pdf $p(x)$ underlying X is a weighted sum of J normal distributions with covariance matrices of the form $S_i = \sigma_i^2 I$, and we employ the EM algorithm to estimate the unknown parameters in the adopted model of $p(x)$. The goal is to demonstrate the dependence of the EM algorithm on the initial conditions and the parameter J. To this end, we use the following sets of initial parameter estimates:

- $J = 3$, $m_{1,ini} = [0,\ 2]^T$, $m_{2,ini} = [5,\ 2]^T$, $m_{3,ini} = [5,\ 5]^T$, $S_{1,ini} = 0.15I$, $S_{2,ini} = 0.27I$, $S_{3,ini} = 0.4I$ and $P_{1,ini} = P_{2,ini} = P_{3,ini} = 1/3$
- $J = 3$, $m_{1,ini} = [1.6,\ 1.4]^T$, $m_{2,ini} = [1.4,\ 1.6]^T$, $m_{3,ini} = [1.3,\ 1.5]^T$, $S_{1,ini} = 0.2I$, $S_{2,ini} = 0.4I$, $S_{3,ini} = 0.3I$ and $P_{1,ini} = 0.2$, $P_{2,ini} = 0.4$, $P_{3,ini} = 0.4$
- $J = 2$, $m_{1,ini} = [1.6,\ 1.4]^T$, $m_{2,ini} = [1.4,\ 1.6]^T$, $S_{1,ini} = 0.2I$, $S_{2,ini} = 0.4I$ and $P_{1,ini} = P_{2,ini} = 1/2$

Comment on the results.

[80]:
```
library(MASS)
mu1_12 <- matrix(c(1,1), nrow=1)
sigma1_12 <- matrix(c(0.1,0,0,0.1), nrow=2)
p1_12 <- 0.4
Points1_12 <- mvrnorm(200, mu = mu1_12, Sigma = sigma1_12)
print(mu1_12)
print(sigma1_12)
print(head(Points1_12))
```

```
     [,1] [,2]
[1,]    1    1
     [,1] [,2]
[1,]  0.1  0.0
[2,]  0.0  0.1
          [,1]      [,2]
[1,] 0.9589238 1.2338546
[2,] 0.9085704 0.7735388
[3,] 1.3430984 0.2959268
[4,] 1.8633380 0.8428438
[5,] 1.2476955 1.0125095
[6,] 1.0185360 0.9666111
```

[81]:
```
mu2_12 <- matrix(c(3,3), nrow=1)
sigma2_12 <- matrix(c(0.2,0,0,0.2), nrow=2)
p2_12 <- 0.4
Points2_12 <- mvrnorm(200, mu = mu2_12, Sigma = sigma2_12)
print(mu2_12)
```

```
print(sigma2_12)
print(head(Points2_12))
```

```
     [,1] [,2]
[1,]    3    3
     [,1] [,2]
[1,]  0.2  0.0
[2,]  0.0  0.2
         [,1]     [,2]
[1,] 3.769739 3.004831
[2,] 2.612332 2.612040
[3,] 3.246297 3.249079
[4,] 2.807147 2.975534
[5,] 3.154904 2.234707
[6,] 3.163072 3.443462
```

[82]:
```
mu3_12 <- matrix(c(2,6), nrow=1)
sigma3_12 <- matrix(c(0.3,0,0,0.3), nrow=2)
p3_12 <- 0.2
Points3_12 <- mvrnorm(100, mu = mu3_12, Sigma = sigma3_12)
print(mu3_12)
print(sigma3_12)
print(head(Points3_12))
```

```
     [,1] [,2]
[1,]    2    6
     [,1] [,2]
[1,]  0.3  0.0
[2,]  0.0  0.3
         [,1]     [,2]
[1,] 2.154551 6.073213
[2,] 2.092688 5.512972
[3,] 1.721648 5.144747
[4,] 1.453212 5.113920
[5,] 1.105025 6.927554
[6,] 2.821193 5.750192
```

[83]:
```
pX <- rbind(Points1_12,Points2_12,Points3_12)
pX = pX[sample(1:nrow(pX)), ]
#pX = p1_12*Points1_12 + p2_12*Points2_12 + p3_12*Points3_12
print(head(pX))
```

```
         [,1]      [,2]
[1,] 2.077403 6.6858447
[2,] 1.079652 0.8981972
[3,] 1.377633 1.3929323
[4,] 3.015497 3.2291169
[5,] 2.139651 7.5793479
[6,] 2.842079 3.8188254
```

0.19 a)

[84]:
```
library(mvtnorm)
g1_j <- 3
g1_mu1_12 <- matrix(c(0,2), nrow=1)
g1_sigma1_12 <- matrix(c(0.15,0,0,0.15), nrow=2)
g1_p1_12 <- (1/3)
g1_mu2_12 <- matrix(c(5,2), nrow=1)
g1_sigma2_12 <- matrix(c(0.27,0,0,0.27), nrow=2)
g1_p2_12 <- (1/3)
g1_mu3_12 <- matrix(c(5,5), nrow=1)
g1_sigma3_12 <- matrix(c(0.4,0,0,0.4), nrow=2)
g1_p3_12 <- (1/3)

MaxIter <- 75
for (iter in 1:MaxIter){
  g1_responsibilities <- matrix(0,nrow(pX),g1_j)

  for (jj in 1:g1_j){
    if (jj==1){
    for (kk in 1:nrow(pX)){
    g1_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g1_mu1_12, sigma =␣
 ↪g1_sigma1_12) * g1_p1_12
    }}
    if (jj==2){
    for (kk in 1:nrow(pX)){
    g1_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g1_mu2_12, sigma =␣
 ↪g1_sigma2_12) * g1_p2_12
    }}
    if (jj==3){
    for (kk in 1:nrow(pX)){
    g1_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g1_mu3_12, sigma =␣
 ↪g1_sigma3_12) * g1_p3_12
    }}
  }
  g1_responsibilities <- g1_responsibilities / rowSums(g1_responsibilities)


  for (jj in 1:g1_j){
    if (jj==1){ sum_jj1 <- c(0,0)
    for (kk in 1:nrow(pX)){
    sum_jj1 = sum_jj1 + g1_responsibilities[kk,jj]*pX[kk,]
    }}
    if (jj==2){ sum_jj2 <- c(0,0)
    for (kk in 1:nrow(pX)){
    sum_jj2 = sum_jj2 + g1_responsibilities[kk,jj]*pX[kk,]
    }}
```

```
    if (jj==3){ sum_jj3 <- c(0,0)
    for (kk in 1:nrow(pX)){
    sum_jj3 = sum_jj3 + g1_responsibilities[kk,jj]*pX[kk,]
    }}
  }
  Nk <- colSums(g1_responsibilities)
  g1_p1_12 <- Nk[1] / nrow(pX)
  g1_p2_12 <- Nk[2] / nrow(pX)
  g1_p3_12 <- Nk[3] / nrow(pX)

  g1_mu1_12 <- sum_jj1 / Nk[1]
  g1_mu2_12 <- sum_jj2 / Nk[2]
  g1_mu3_12 <- sum_jj3 / Nk[3]


  for (jj in 1:g1_j){
    if (jj==1){
    cov_jj1 <- matrix(c(0,0,0,0),nrow=2)
    for (kk in 1:nrow(pX)){
    cov_jj1 = cov_jj1 + g1_responsibilities[kk,jj]*␣
 ↪(((pX[kk,]-g1_mu1_12))%*%(t(pX[kk,]-g1_mu1_12)))
    }}
    if (jj==2){
    cov_jj2 <- matrix(c(0,0,0,0),nrow=2)
    for (kk in 1:nrow(pX)){
    cov_jj2 = cov_jj2 + g1_responsibilities[kk,jj]*␣
 ↪(((pX[kk,]-g1_mu2_12))%*%(t(pX[kk,]-g1_mu2_12)))
    }}
    if (jj==3){
    cov_jj3 <- matrix(c(0,0,0,0),nrow=2)
    for (kk in 1:nrow(pX)){
    cov_jj3 = cov_jj3 + g1_responsibilities[kk,jj]*␣
 ↪(((pX[kk,]-g1_mu3_12))%*%(t(pX[kk,]-g1_mu3_12)))
    }}
  }
  g1_sigma1_12 <- cov_jj1 / Nk[1]
  g1_sigma2_12 <- cov_jj2 / Nk[2]
  g1_sigma3_12 <- cov_jj3 / Nk[3]
}

print(g1_mu1_12)
print(g1_mu2_12)
print(g1_mu3_12)

print(g1_sigma1_12)
print(g1_sigma2_12)
print(g1_sigma3_12)
```

```
[1] 1.002013 1.013994
[1] 2.97952 3.03992
[1] 1.987067 5.908202
             [,1]         [,2]
[1,] 0.092081689 0.006636957
[2,] 0.006636957 0.116734012
           [,1]       [,2]
[1,] 0.19536489 0.01041489
[2,] 0.01041489 0.19123991
           [,1]       [,2]
[1,] 0.28057924 0.02073512
[2,] 0.02073512 0.29256049
```

0.20 b)

[87]:
```
g2_j <- 3
g2_mu1_12 <- matrix(c(1.6,1.4), nrow=1)
g2_sigma1_12 <- matrix(c(0.2,0,0,0.2), nrow=2)
g2_p1_12 <- 0.2
g2_mu2_12 <- matrix(c(1.4,1.6), nrow=1)
g2_sigma2_12 <- matrix(c(0.4,0,0,0.4), nrow=2)
g2_p2_12 <- 0.4
g2_mu3_12 <- matrix(c(1.3,1.5), nrow=1)
g2_sigma3_12 <- matrix(c(0.3,0,0,0.3), nrow=2)
g2_p3_12 <- 0.4

MaxIter <- 100
for (iter in 1:MaxIter){
  g2_responsibilities <- matrix(0,nrow(pX),g1_j)

  for (jj in 1:g1_j){
    if (jj==1){
    for (kk in 1:nrow(pX)){
    g2_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g2_mu1_12, sigma =␣
 ↪g2_sigma1_12) * g2_p1_12
    }}
    if (jj==2){
    for (kk in 1:nrow(pX)){
    g2_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g2_mu2_12, sigma =␣
 ↪g2_sigma2_12) * g2_p2_12
    }}
    if (jj==3){
    for (kk in 1:nrow(pX)){
    g2_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g2_mu3_12, sigma =␣
 ↪g2_sigma3_12) * g2_p3_12
    }}
  }
```

```
g2_responsibilities <- g2_responsibilities / rowSums(g2_responsibilities)


for (jj in 1:g1_j){
  if (jj==1){ sum_jj1 <- c(0,0)
  for (kk in 1:nrow(pX)){
  sum_jj1 = sum_jj1 + g2_responsibilities[kk,jj]*pX[kk,]
  }}
  if (jj==2){ sum_jj2 <- c(0,0)
  for (kk in 1:nrow(pX)){
  sum_jj2 = sum_jj2 + g2_responsibilities[kk,jj]*pX[kk,]
  }}
  if (jj==3){ sum_jj3 <- c(0,0)
  for (kk in 1:nrow(pX)){
  sum_jj3 = sum_jj3 + g2_responsibilities[kk,jj]*pX[kk,]
  }}
}
Nk <- colSums(g2_responsibilities)
g2_p1_12 <- Nk[1] / nrow(pX)
g2_p2_12 <- Nk[2] / nrow(pX)
g2_p3_12 <- Nk[3] / nrow(pX)

g2_mu1_12 <- sum_jj1 / Nk[1]
g2_mu2_12 <- sum_jj2 / Nk[2]
g2_mu3_12 <- sum_jj3 / Nk[3]


for (jj in 1:g1_j){
  if (jj==1){
  cov_jj1 <- matrix(c(0,0,0,0),nrow=2)
  for (kk in 1:nrow(pX)){
  cov_jj1 = cov_jj1 + g2_responsibilities[kk,jj]*␣
↪(((pX[kk,]-g2_mu1_12))%*%(t(pX[kk,]-g2_mu1_12)))
  }}
  if (jj==2){
  cov_jj2 <- matrix(c(0,0,0,0),nrow=2)
  for (kk in 1:nrow(pX)){
  cov_jj2 = cov_jj2 + g2_responsibilities[kk,jj]*␣
↪(((pX[kk,]-g2_mu2_12))%*%(t(pX[kk,]-g2_mu2_12)))
  }}
  if (jj==3){
  cov_jj3 <- matrix(c(0,0,0,0),nrow=2)
  for (kk in 1:nrow(pX)){
  cov_jj3 = cov_jj3 + g2_responsibilities[kk,jj]*␣
↪(((pX[kk,]-g2_mu3_12))%*%(t(pX[kk,]-g2_mu3_12)))
  }}
}
```

```
  g2_sigma1_12 <- cov_jj1 / Nk[1]
  g2_sigma2_12 <- cov_jj2 / Nk[2]
  g2_sigma3_12 <- cov_jj3 / Nk[3]
}

print(g2_mu1_12)
print(g2_mu2_12)
print(g2_mu3_12)

print(g2_sigma1_12)
print(g2_sigma2_12)
print(g2_sigma3_12)
```

```
[1] 0.9780199 0.9140752
[1] 2.649109 3.996623
[1] 1.015341 1.069548
            [,1]        [,2]
[1,] 0.119860613 0.006076316
[2,] 0.006076316 0.056728888
           [,1]       [,2]
[1,]  0.4422859 -0.6196101
[2,] -0.6196101  2.0538612
            [,1]        [,2]
[1,] 0.076634760 0.005674274
[2,] 0.005674274 0.142990631
```

0.21 c)

[88]:
```
g3_j <- 2
g3_mu1_12 <- matrix(c(1.6,1.4), nrow=1)
g3_sigma1_12 <- matrix(c(0.2,0,0,0.2), nrow=2)
g3_p1_12 <- 0.5
g3_mu2_12 <- matrix(c(1.4,1.6), nrow=1)
g3_sigma2_12 <- matrix(c(0.4,0,0,0.4), nrow=2)
g3_p2_12 <- 0.5

MaxIter <- 100
for (iter in 1:MaxIter){
  g3_responsibilities <- matrix(0,nrow(pX),g1_j)

  for (jj in 1:g1_j){
    if (jj==1){
    for (kk in 1:nrow(pX)){
    g3_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g3_mu1_12, sigma =␣
 ↪g3_sigma1_12) * g3_p1_12
    }}
    if (jj==2){
```

```
    for (kk in 1:nrow(pX)){
    g3_responsibilities[kk,jj] <- dmvnorm(pX[kk,], mean = g3_mu2_12, sigma =␣
↪g3_sigma2_12) * g3_p2_12
    }}
  }
  g3_responsibilities <- g3_responsibilities / rowSums(g3_responsibilities)


  for (jj in 1:g1_j){
    if (jj==1){ sum_jj1 <- c(0,0)
    for (kk in 1:nrow(pX)){
    sum_jj1 = sum_jj1 + g3_responsibilities[kk,jj]*pX[kk,]
    }}
    if (jj==2){ sum_jj2 <- c(0,0)
    for (kk in 1:nrow(pX)){
    sum_jj2 = sum_jj2 + g3_responsibilities[kk,jj]*pX[kk,]
    }}
  }
  Nk <- colSums(g3_responsibilities)
  g3_p1_12 <- Nk[1] / nrow(pX)
  g3_p2_12 <- Nk[2] / nrow(pX)

  g3_mu1_12 <- sum_jj1 / Nk[1]
  g3_mu2_12 <- sum_jj2 / Nk[2]


  for (jj in 1:g1_j){
    if (jj==1){
    cov_jj1 <- matrix(c(0,0,0,0),nrow=2)
    for (kk in 1:nrow(pX)){
    cov_jj1 = cov_jj1 + g3_responsibilities[kk,jj]*␣
↪(((pX[kk,]-g3_mu1_12))%*%(t(pX[kk,]-g3_mu1_12)))
    }}
    if (jj==2){
    cov_jj2 <- matrix(c(0,0,0,0),nrow=2)
    for (kk in 1:nrow(pX)){
    cov_jj2 = cov_jj2 + g3_responsibilities[kk,jj]*␣
↪(((pX[kk,]-g3_mu2_12))%*%(t(pX[kk,]-g3_mu2_12)))
    }}
  }
  g3_sigma1_12 <- cov_jj1 / Nk[1]
  g3_sigma2_12 <- cov_jj2 / Nk[2]
}

print(g3_mu1_12)
print(g3_mu2_12)
```

```
print(g3_sigma1_12)
print(g3_sigma2_12)
```

```
[1] 1.001918 1.014016
[1] 2.648592 3.995956
             [,1]         [,2]
[1,] 0.092010039 0.006636009
[2,] 0.006636009 0.116833944
           [,1]       [,2]
[1,]  0.4426808 -0.6185596
[2,] -0.6185596  2.0539198
```

0.22 Question 13

13. Two coins are used for a coin-tossing experiment, that is, coin A and coin B. The probability that coin A returns heads is 0.6, and the respective probability for coin B is 0.4. An individual standing behind a curtain decides which coin to toss as follows: the first coin to be tossed is always coin A, the probability that coin A is re-tossed is 0.4, and similarly, the probability that coin B is re-tossed is 0.6. An observer can only have access to the outcome of the experiment, that is, the sequence of heads and tails that is produced. (a) Model the experiment by means of a HMM (i.e., define the vector of the initial state probabilities, the transition matrix and the matrix of the emission probabilities) and (b) use the *Baum Welch* algorithm to compute the HMM score for the sequence of observations $\{H, H, T, H, T, T\}$ where H stands for heads and T stands for tails.

0.23 a)

[23]:
```
StateProbs <- c(1,0)
TransProbs <- matrix(c(0.4,0.6,0.4,0.6), nrow=2, byrow=T)
rownames(TransProbs) <- c("A","B")
colnames(TransProbs) <- c("A","B")
EmissProbs <- matrix(c(0.6,0.4,0.4,0.6), nrow=2, byrow=T)
rownames(EmissProbs) <- c("A","B")
colnames(EmissProbs) <- c("H","T")

print(StateProbs)
print(TransProbs)
print(EmissProbs)
```

```
[1] 1 0
    A   B
A 0.4 0.6
B 0.4 0.6
    H   T
A 0.6 0.4
B 0.4 0.6
```

0.24 b)

[33]:
```
X_HMM <- c("^","H","H","T","H","T","T")
ForwardAlg <- matrix(0,7, nrow=3, byrow=T)
rownames(ForwardAlg) <- c("Start","A","B")
colnames(ForwardAlg) <- X_HMM
ForwardAlg[1,1] <- 1
print(ForwardAlg)
```

```
      ^ H H T H T T
Start 1 0 0 0 0 0 0
A     0 0 0 0 0 0 0
B     0 0 0 0 0 0 0
```

[34]:
```
for (i in 2:ncol(ForwardAlg)){

    if (i == 2){
        for (j in 2:nrow(ForwardAlg)){
            ForwardAlg[j,i] <- StateProbs[j-1] * ForwardAlg[1,i-1] *␣
 ↪EmissProbs[rownames(ForwardAlg)[j],colnames(ForwardAlg)[i]]
        }
    }

    else{
        for (j in 2:nrow(ForwardAlg)){
            for (k in 2:nrow(ForwardAlg)){
                ForwardAlg[j,i] <- ForwardAlg[j,i] + TransProbs[k-1,j-1] *␣
 ↪ForwardAlg[k,i-1]
            }
            ForwardAlg[j,i] <- ForwardAlg[j,i] *␣
 ↪EmissProbs[rownames(ForwardAlg)[j],colnames(ForwardAlg)[i]]
        }
    }
}
print(ForwardAlg)
```

```
      ^   H     H       T         H           T           T
Start 1 0.0 0.000 0.00000 0.0000000 0.00000000 0.000000000
A     0 0.6 0.144 0.04608 0.0359424 0.01150157 0.005980815
B     0 0.0 0.144 0.10368 0.0359424 0.02587853 0.013456835
```

[35]:
```
answer <- 0
for (k in 2:nrow(ForwardAlg)){
    answer <- answer + ForwardAlg[k,ncol(ForwardAlg)]
}
print(answer)
```

```
[1] 0.01943765
```