Assignment 01

Mauricio Vazquez & Mariana Luna

2024-09-27

Repository link: https://github.com/MauricioVazquezM/Multivariate_Statistical_Course_Assignments_Fall2024

Ex. 8.2: Modern Multivariate Statistical Techniques (Izenman)

Consider the wine data. Compute a LDA, draw a 2D-scatterplot of the first two LDF coordinates, and color-code the points by wine type. What do you notice?

```
# Loading dataset
load("C:/Users/mauva/OneDrive/Documents/ITAM/9no Semestre/METODOS MULTIVARIADOS/REPOSITORIO/Multivariat

# Parsing to dataframe
wine_df <- as.data.frame(wine)

# Checking head dataset
head(wine_df)
```

```
##   Alcohol MalicAcid  Ash AlcAsh  Mg Phenols Flav NonFlavPhenols Proa Color  Hue
## 1   14.23      1.71 2.43   15.6 127    2.80 3.06           0.28 2.29  5.64 1.04
## 2   13.16      2.36 2.67   18.6 101    2.80 3.24           0.30 2.81  5.68 1.03
## 3   13.86      1.35 2.27   16.0  98    2.98 3.15           0.22 1.85  7.22 1.01
## 4   14.10      2.16 2.30   18.0 105    2.95 3.32           0.22 2.38  5.75 1.25
## 5   14.12      1.48 2.32   16.8  95    2.20 2.43           0.26 1.57  5.00 1.17
## 6   13.75      1.73 2.41   16.0  89    2.60 2.76           0.29 1.81  5.60 1.15
##     OD Proline classdigit  class
## 1 3.92    1065          1 Barolo
## 2 3.17    1185          1 Barolo
## 3 3.55    1045          1 Barolo
## 4 3.17    1510          1 Barolo
## 5 2.82    1280          1 Barolo
## 6 2.90    1320          1 Barolo
```

```
# Parsing dataset column
wine_df$classdigit <- as.factor(wine_df$classdigit)

# Performing LDA
lda_model <- lda(classdigit ~ Alcohol + MalicAcid + Ash + AlcAsh + Mg + Phenols + Flav + NonFlavPhenols
                 data= wine_df)

# Projecting the data onto the LDA axes
```

```r
lda_values <- predict(lda_model)

# Creating a data frame with the LDA components
lda_df <- data.frame(LD1 = lda_values$x[, 1], LD2 = lda_values$x[, 2], WineType = wine_df$class)

# Plotting
ggplot(lda_df, aes(x = LD1, y = LD2, color = WineType)) +
  geom_point(size = 3) +
  labs(title = "Scatterplot of the First Two LDA Components",
       x = "First Discriminant Component (LD1)",
       y = "Second Discriminant Component (LD2)") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

Ex. 8.3: Modern Multivariate Statistical Techniques (Izenman)

Suppose $X_1 \sim N_r(\mu_1, \Sigma_{XX})$ and $X_2 \sim N_r(\mu_2, \Sigma_{XX})$ are independently distributed. Consider the statistic:

$$\frac{(E(a^\tau X_1) - E(a^\tau X_2))^2}{\mathrm{Var}(a^\tau X_1 - a^\tau X_2)}$$

as a function of a. Show that $a \propto \Sigma_{XX}^{-1}(\mu_1 - \mu_2)$ maximizes the statistic using a Lagrange multiplier approach.

Answer on PDF

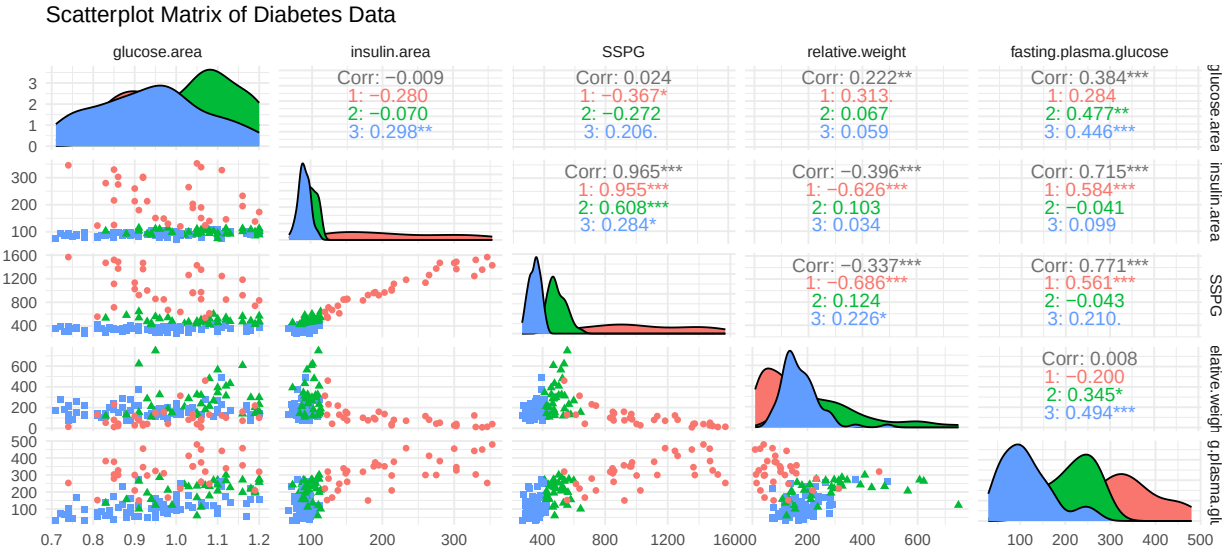
Ex. 8.5: Modern Multivariate Statistical Techniques (Izenman)

Consider the diabetes data. Draw a scatterplot matrix of all five variables with different colors or symbols representing the three classes of diabetes. Do these pairwise plots suggest multivariate Gaussian distributions for each class with equal covariance matrices? Carry out an LDA and draw the 2D-scatterplot of the first two discriminating functions. Using the leave-one-out CV procedure, find the confusion table and identify those observations that are incorrectly classified based upon the LDA classification rule. Do the same for the QDA procedure.

```r
# Loading dataset
load("C:/Users/mauva/OneDrive/Documents/ITAM/9no Semestre/METODOS MULTIVARIADOS/REPOSITORIO/Multivariat

# Parsing to dataframe
diabetes_df <- as.data.frame(diabetes)

# Checking head dataset
head(diabetes_df)
```

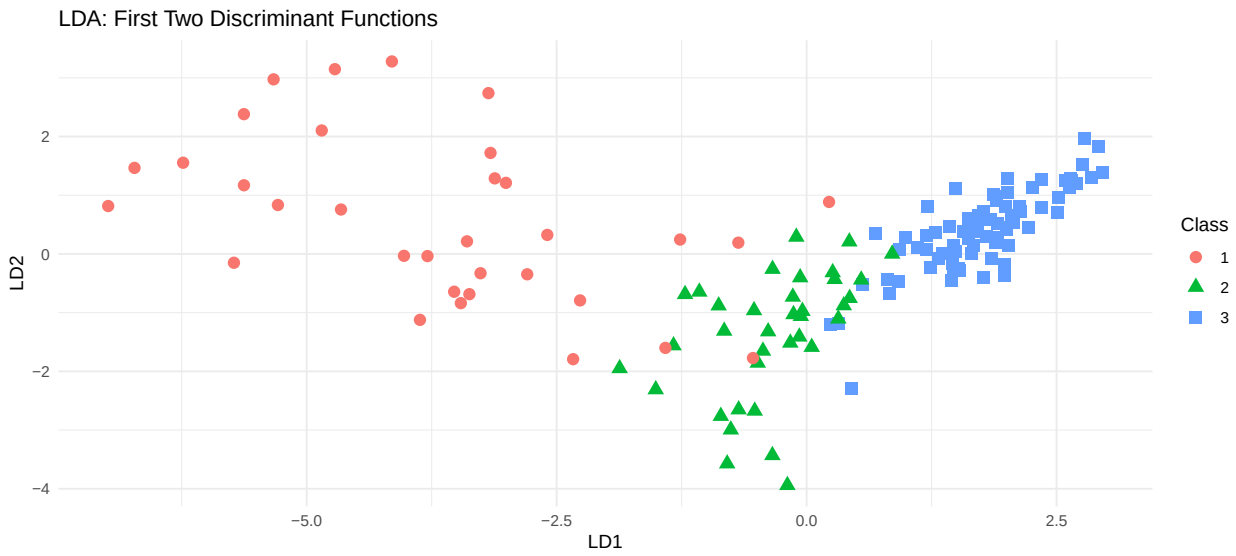
```
##   glucose.area insulin.area SSPG relative.weight fasting.plasma.glucose class
## 1         0.81           80  356             124                      55     3
## 2         0.95           97  289             117                      76     3
## 3         0.94          105  319             143                     105     3
## 4         1.04           90  356             199                     108     3
## 5         1.00           90  323             240                     143     3
## 6         0.76           86  381             157                     165     3
```

```r
# Drawing scatterplot matrix for the five predictor variables with color distinction for 'class'
ggpairs(diabetes_df, columns = 1:5, aes(color = as.factor(class), shape = as.factor(class))) +
  theme_minimal() +
  labs(title = "Scatterplot Matrix of Diabetes Data",
       color = "Class", shape = "Class")
```




```r
# Performing LDA
lda_model <- lda(class ~ glucose.area + insulin.area + SSPG + relative.weight + fasting.plasma.glucose,
                 data = diabetes_df)

# Predicting the discriminant functions
lda_values <- predict(lda_model)

# Creating 2D scatter plot for the first two discriminant functions
lda_df <- data.frame(lda_values$x, class = diabetes_df$class)

ggplot(lda_df, aes(x = LD1, y = LD2, color = as.factor(class), shape = as.factor(class))) +
```

```
  geom_point(size = 3) +
  theme_minimal() +
  labs(title = "LDA: First Two Discriminant Functions",
       color = "Class", shape = "Class") +
  xlab("LD1") + ylab("LD2")
```




```
# Performing LDA with Leave-One-Out CV
lda_cv <- lda(class ~ glucose.area + insulin.area + SSPG + relative.weight + fasting.plasma.glucose,
              data = diabetes_df, CV = TRUE)

# Confusion table
lda_pred <- lda_cv$class
table(Predicted = lda_pred, Actual = diabetes_df$class)
```

```
##           Actual
## Predicted  1  2  3
##         1 26  0  0
##         2  6 30  3
##         3  1  6 73
```

```
# Identifying misclassified observations
misclassified_lda <- which(lda_pred != diabetes_df$class)
misclassified_lda
```

```
##  [1]  26  59  66  69  82  96 105 110 112 115 124 131 134 135 136 137
```

```
# Performing QDA
qda_model <- qda(class ~ glucose.area + insulin.area + SSPG
                 + relative.weight + fasting.plasma.glucose, data = diabetes_df)

# Performing QDA with Leave-One-Out CV
qda_cv <- qda(class ~ glucose.area + insulin.area + SSPG + relative.weight + fasting.plasma.glucose,
              data = diabetes_df, CV = TRUE)
```

```r
# Confusion table
qda_pred <- qda_cv$class
table(Predicted = qda_pred, Actual = diabetes_df$class)
```

```
##          Actual
## Predicted  1  2  3
##         1 30  3  0
##         2  3 29  4
##         3  0  4 72
```

```r
# Identifying misclassified observations
misclassified_qda <- which(qda_pred != diabetes_df$class)
misclassified_qda
```

```
##  [1]  26  63  68  69  77  82  95  96 107 110 111 131 134 136
```

Ex. 12.2: Modern Multivariate Statistical Techniques (Izenman)

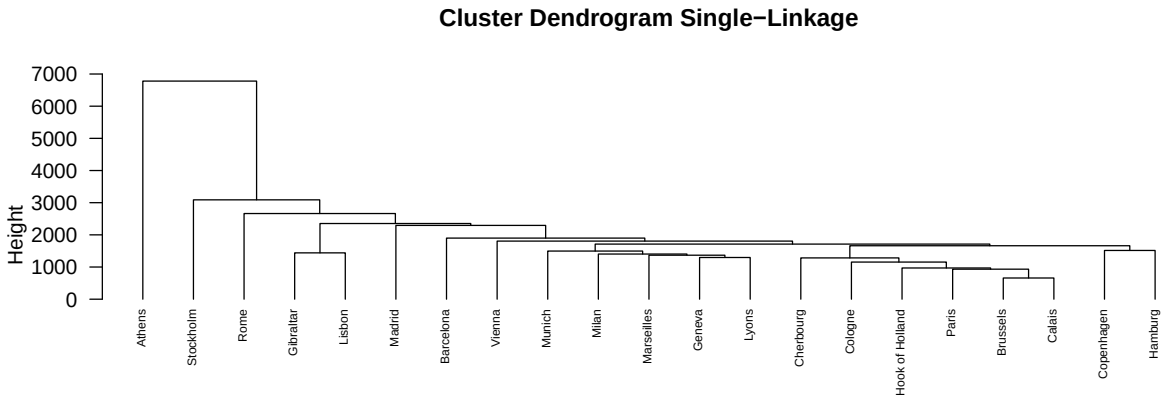
Write a computer program to implement single-linkage, averagelinkage, and complete-linkage agglomerative hierarchical clustering. Try it out on a data set of your choice.

```r
# Loading the eurodist dataset
d7 <- eurodist

# Removing any missing values
d7 <- na.omit(d7)

# Displaying the first 7 rows and 7 columns of the distance matrix
kable(head(as.matrix(d7), c(7L, 7L)), row.names = T)
```

	Athens	Barcelona	Brussels	Calais	Cherbourg	Cologne	Copenhagen
Athens	0	3313	2963	3175	3339	2762	3276
Barcelona	3313	0	1318	1326	1294	1498	2218
Brussels	2963	1318	0	204	583	206	966
Calais	3175	1326	204	0	460	409	1136
Cherbourg	3339	1294	583	460	0	785	1545
Cologne	2762	1498	206	409	785	0	760
Copenhagen	3276	2218	966	1136	1545	760	0

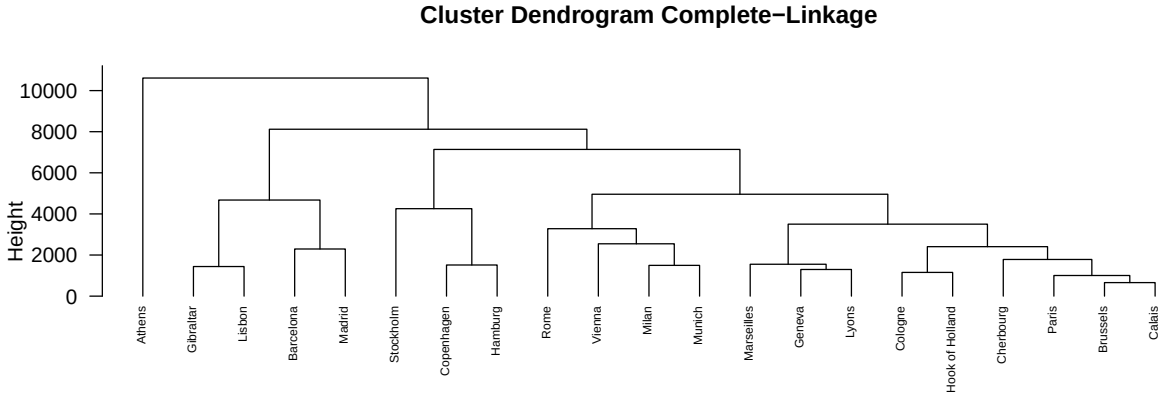
```r
# Computing the Euclidean distance matrix from the dataset 'd7'
diss_matrix7 <- dist(d7, method="euclidean")

# Performing hierarchical clustering using the single-linkage method
hc1_7 <- hclust(diss_matrix7, method="single")

# Performing hierarchical clustering using the complete-linkage method
hc2_7 <- hclust(diss_matrix7, method="complete")

# Performing hierarchical clustering using the average-linkage method
```

```r
hc3_7 <- hclust(diss_matrix7, method="average")

# Plotting the dendrogram for the single-linkage clustering result
plot(hc1_7, cex=0.6, hang=-1,
main = "Cluster Dendrogram Single-Linkage",
sub = "Matriz de distancia Euclidiana sobre dataset eurodist", las=2)
```



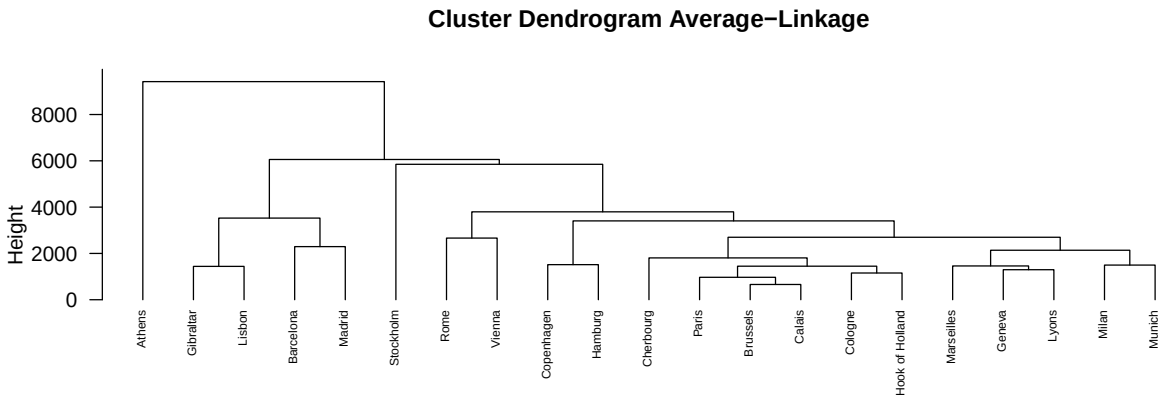
diss_matrix7
Matriz de distancia Euclidiana sobre dataset eurodist

```r
# Plotting the dendrogram for the complete-linkage clustering result
plot(hc2_7, cex=0.6, hang=-1,
main = "Cluster Dendrogram Complete-Linkage",
sub = "Matriz de distancia Euclidiana sobre dataset eurodist", las =2)
```



diss_matrix7
Matriz de distancia Euclidiana sobre dataset eurodist

```r
# Plotting the dendrogram for the average-linkage clustering result
plot(hc3_7, cex=0.6, hang=-1,
main = "Cluster Dendrogram Average-Linkage",
sub = "Matriz de distancia Euclidiana sobre dataset eurodist", las =2)
```



diss_matrix7
Matriz de distancia Euclidiana sobre dataset eurodist

Ex. 12.3: Modern Multivariate Statistical Techniques (Izenman)

Cluster the primate.scapulae data using single-linkage, averagelinkage, and complete-linkage agglomerative hierarchical clustering methods. Find the five-cluster solutions for all three methods, which allows comparison with the true primate classifications. Find the misclassification rate for all three methods. Show that the lowest rate occurs for the completelinkage method and the highest for the single-linkage method.

```
load("C:/Users/mauva/OneDrive/Documents/ITAM/9no Semestre/METODOS MULTIVARIADOS/REPOSITORIO/Multivariat

df <- as.data.frame(primate.scapulae)

# Computing distance matrix
dist_matrix <- dist(df[, -which(names(df) == "class")])

# Clustering with different linkage methods
single_linkage <- hclust(dist_matrix, method = "single")
average_linkage <- hclust(dist_matrix, method = "average")
complete_linkage <- hclust(dist_matrix, method = "complete")

# Cutting the dendrogram to form 5 clusters
single_clusters <- cutree(single_linkage, k = 5)
average_clusters <- cutree(average_linkage, k = 5)
complete_clusters <- cutree(complete_linkage, k = 5)

# Calculating Adjusted Rand Index (ARI) for each method
ari_single <- adjustedRandIndex(primate.scapulae$class, single_clusters)
ari_average <- adjustedRandIndex(primate.scapulae$class, average_clusters)
ari_complete <- adjustedRandIndex(primate.scapulae$class, complete_clusters)

# Misclassification rates
misclassification_single <- 1 - ari_single
misclassification_average <- 1 - ari_average
misclassification_complete <- 1 - ari_complete
```

```r
# Displaying misclassification rates
cat("Misclassification rate (Single-linkage): ", misclassification_single, "\n")
```

```
## Misclassification rate (Single-linkage):  0.3821588
```

```r
cat("Misclassification rate (Average-linkage): ", misclassification_average, "\n")
```

```
## Misclassification rate (Average-linkage):  0.3821588
```

```r
cat("Misclassification rate (Complete-linkage): ", misclassification_complete, "\n")
```

```
## Misclassification rate (Complete-linkage):  0.4203452
```

Ex. 11.1: Applied Multivariate Statistical Analysis (Johnson & Wichern)

Consider the two data sets

$$X_1 = \begin{bmatrix} 3 & 7 \\ 2 & 4 \\ 4 & 7 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 6 & 9 \\ 5 & 7 \\ 4 & 8 \end{bmatrix}$$

for which

$$\bar{x}_1 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}, \quad \bar{x}_2 = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$

and

$$S_{\text{pooled}} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

- (a) Calculate the linear discriminant function in (11-19).
- (b) Classify the observation $\mathbf{x_0} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}$ as population π_1 or population π_2, using Rule (11-18) with equal priors and equal costs.

Answer on PDF

Ex. 11.3: Applied Multivariate Statistical Analysis (Johnson & Wichern)

Prove Result 11.1.

Answer on PDF

Ex. 11.5: Applied Multivariate Statistical Analysis (Johnson & Wichern)

Show that

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)+\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)$$

is equivalent to:

$$(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)'\Sigma^{-1}\mathbf{x}-\frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)'\Sigma^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)$$

Answer on PDF

Ex. 12.1: Applied Multivariate Statistical Analysis (Johnson & Wichern)

Certain characteristics associated with a few recent U.S. presidents are listed in Table 12.11:

President	Birthplace	Elected first term?	Party	Prior U.S. congressional experience?	Served as vice president?
R. Reagan	Midwest	Yes	Republican	No	No
J. Carter	South	Yes	Democrat	No	No
G. Ford	Midwest	No	Republican	Yes	Yes
R. Nixon	West	Yes	Republican	Yes	Yes
L. Johnson	South	No	Democrat	Yes	Yes
J. Kennedy	East	Yes	Democrat	Yes	No

1. **(a)** Introducing appropriate binary variables, calculate similarity coefficient 1 in Table 12.1 for pairs of presidents.
2. **(b)** Proceeding as in Part a, calculate similarity coefficients 2 and 3 in Table 12.1 Verify the mono tonicity relation of coefficients 1,2, and 3 by displaying the order of the 15 similarities for each coefficient.

```r
# Creating the table of presidents with their characteristics
presidents <- data.frame(
  Name = c("R. Reagan", "J. Carter", "G. Ford", "R. Nixon", "L. Johnson", "J. Kennedy"),
  Birthplace = c("No South", "South", "No South", "No South", "South", "No South"),
  ElectedFirstTerm = c(1, 1, 0, 0, 0, 1),
  Party = c("Republican", "Democrat", "Republican", "Republican", "Democrat", "Democrat"),
  CongressionalExperience = c(0, 0, 1, 1, 1, 1),
  ServedVP = c(0, 0, 1, 0, 1, 0)
)

# Converting the categorical variables to binary form
presidents$BirthplaceBinary <- ifelse(presidents$Birthplace == "South", 1, 0)
presidents$PartyBinary <- ifelse(presidents$Party == "Republican", 1, 0)

# Final binary matrix of variables
binary_data <- presidents[, c("BirthplaceBinary", "ElectedFirstTerm", "PartyBinary", "CongressionalExpe
binary_data
```

```
##   BirthplaceBinary ElectedFirstTerm PartyBinary CongressionalExperience
## 1                0                1           1                       0
## 2                1                1           0                       0
## 3                0                0           1                       1
## 4                0                0           1                       1
## 5                1                0           0                       1
## 6                0                1           0                       1
##   ServedVP
## 1        0
## 2        0
## 3        1
## 4        0
## 5        1
## 6        0
```

```
# Function to calculate Jaccard similarity
jaccard_similarity <- function(x, y) {
  return(sum(x == y) / length(x))
}

# Function to calculate Hamming distance
hamming_distance <- function(x, y) {
  return(sum(x != y))
}

# Function to calculate Dice similarity
dice_similarity <- function(x, y) {
  return(2 * sum(x == y) / (length(x) + sum(x == y)))
}

# Calculating similarities for all pairs of presidents
n <- nrow(binary_data)
similarities_jaccard <- matrix(0, n, n)
similarities_dice <- matrix(0, n, n)
hamming_distances <- matrix(0, n, n)

# Loop through all pairs of presidents
for (i in 1:(n-1)) {
  for (j in (i+1):n) {
    similarities_jaccard[i, j] <- jaccard_similarity(binary_data[i, ], binary_data[j, ])
    similarities_dice[i, j] <- dice_similarity(binary_data[i, ], binary_data[j, ])
    hamming_distances[i, j] <- hamming_distance(binary_data[i, ], binary_data[j, ])
  }
}

# Displaying the similarity matrices
cat("Jaccard Similarity Matrix:\n")
```

```
## Jaccard Similarity Matrix:
```

```
print(similarities_jaccard)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
```

```
## [1,]    0  0.6  0.4  0.6  0.0  0.6
## [2,]    0  0.0  0.0  0.2  0.4  0.6
## [3,]    0  0.0  0.0  0.8  0.6  0.4
## [4,]    0  0.0  0.0  0.0  0.4  0.6
## [5,]    0  0.0  0.0  0.0  0.0  0.4
## [6,]    0  0.0  0.0  0.0  0.0  0.0
```

```
cat("Dice Similarity Matrix:\n")
```

```
## Dice Similarity Matrix:
```

```
print(similarities_dice)
```

```
##      [,1] [,2]      [,3]      [,4]      [,5]      [,6]
## [1,]    0 0.75 0.5714286 0.7500000 0.0000000 0.7500000
## [2,]    0 0.00 0.0000000 0.3333333 0.5714286 0.7500000
## [3,]    0 0.00 0.0000000 0.8888889 0.7500000 0.5714286
## [4,]    0 0.00 0.0000000 0.0000000 0.5714286 0.7500000
## [5,]    0 0.00 0.0000000 0.0000000 0.0000000 0.5714286
## [6,]    0 0.00 0.0000000 0.0000000 0.0000000 0.0000000
```

```
cat("Hamming Distance Matrix:\n")
```

```
## Hamming Distance Matrix:
```

```
print(hamming_distances)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    2    3    2    5    2
## [2,]    0    0    5    4    3    2
## [3,]    0    0    0    1    2    3
## [4,]    0    0    0    0    3    2
## [5,]    0    0    0    0    0    3
## [6,]    0    0    0    0    0    0
```

```
# Converting the matrices into ordered lists to compare the ranks of the coefficients
jaccard_list <- sort(similarities_jaccard[upper.tri(similarities_jaccard)], decreasing = TRUE)
dice_list <- sort(similarities_dice[upper.tri(similarities_dice)], decreasing = TRUE)
hamming_list <- sort(hamming_distances[upper.tri(hamming_distances)])

# Displaying the ordered coefficients
cat("Jaccard Similarities (Ordered):\n")
```

```
## Jaccard Similarities (Ordered):
```

```
print(jaccard_list)
```

```
##  [1] 0.8 0.6 0.6 0.6 0.6 0.6 0.6 0.4 0.4 0.4 0.4 0.4 0.2 0.0 0.0
```

```
cat("Dice Similarities (Ordered):\n")
```

```
## Dice Similarities (Ordered):
```

```
print(dice_list)
```

```
##  [1] 0.8888889 0.7500000 0.7500000 0.7500000 0.7500000 0.7500000 0.7500000
##  [8] 0.5714286 0.5714286 0.5714286 0.5714286 0.5714286 0.3333333 0.0000000
## [15] 0.0000000
```

```
cat("Hamming Distances (Ordered):\n")
```

```
## Hamming Distances (Ordered):
```

```
print(hamming_list)
```

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

Ex. 12.11: Applied Multivariate Statistical Analysis (Johnson & Wichern)

Suppose we measure two variables X_1 and X_2 for four items A, B, C, and D. The data are as follows:

Item	X_1	X_2
A	5	4
B	1	-2
C	-1	1
D	3	1

Use the **K-means clustering** technique to divide the items into $K = 2$ clusters, starting with the initial groups (AB) and (CD).

```
# Creating the dataset with items A, B, C, and D
data <- data.frame(
  Item = c("A", "B", "C", "D"),
  X1 = c(5, 1, -1, 3),
  X2 = c(4, -2, 1, 1)
)

# Displaying the dataset
print(data)
```

```
##   Item X1 X2
## 1    A  5  4
## 2    B  1 -2
## 3    C -1  1
## 4    D  3  1
```

```r
# Defining the initial positions of the cluster centers
initial_centers <- data.frame(
  X1 = c(mean(c(5, 1)), mean(c(-1, 3))),
  X2 = c(mean(c(4, -2)), mean(c(1, 1)))
)

# Displaying the initial centers
print(initial_centers)
```

```
##   X1 X2
## 1  3  1
## 2  1  1
```

```r
# Performing k-means clustering using the initial centers
set.seed(123)
kmeans_result <- kmeans(data[, c("X1", "X2")], centers = initial_centers)

# Displaying the clustering results
cat("Cluster assignments:\n")
```

```
## Cluster assignments:
```

```r
print(kmeans_result$cluster)
```

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

```r
cat("\nCluster centers:\n")
```

```
##
## Cluster centers:
```

```r
print(kmeans_result$centers)
```

```
##   X1   X2
## 1  4  2.5
## 2  0 -0.5
```

Ex. 5.1: Deep Learning: Foundations and Concepts (Bishop & Bishop)

Consider a classification problem with K classes and a target vector $\mathbf{t}$ that uses a 1-of-K binary coding scheme. Show that the conditional expectation $\mathbb{E}[\mathbf{t}|\mathbf{x}]$ is given by the posterior probability $p(C_k|\mathbf{x})$.

Answer on PDF

Ex. 5.6: Deep Learning: Foundations and Concepts (Bishop & Bishop)

Consider two non-negative numbers a and b, and show that if $a \leq b$, then:

$$a \leq \sqrt{ab}$$

Use this result to show that if the decision regions of a two-class classification problem are chosen to minimize the probability of misclassification, then the probability of misclassification will satisfy

$$p(\text{mistake}) \leq \int \sqrt{p(\mathbf{x}|C_1)p(\mathbf{x}|C_2)}\, d\mathbf{x}.$$

Answer on PDF

Ex. 5.12: Deep Learning: Foundations and Concepts (Bishop & Bishop)

Using equations (5.40) and (5.41), derive the result (5.48) for the posterior class probability in the two-class generative model with Gaussian densities, and verify the results (5.49) and (5.50) for the parameters $\mathbf{w}$ and w_0.

Answer on PDF

Ex. 15.2: Deep Learning: Foundations and Concepts (Bishop & Bishop)

In this exercise, we derive the sequential form for the K-means algorithm. At each step we consider a new data point x_n, and only the prototype vector that is closest to x_n is updated. Starting from the expression (15.4) for the prototype vectors in the batch setting, separate out the contribution from the final data point x_n. By rearranging the formula, show that this update takes the form (15.5). Note that, since no approximation is made in this derivation, the resulting prototype vectors will have the property that they each equal the mean of all the data vectors that were assigned to them.

Answer on PDF

Ex. 15.3: Deep Learning: Foundations and Concepts (Bishop & Bishop)

Consider a Gaussian mixture model in which the marginal distribution $p(z)$ for the latent variable is given by (15.9) and the conditional distribution $p(x|z)$ for the observed variable is given by (15.10). Show that the marginal distribution $p(x)$, obtained by summing $p(z)p(x|z)$ over all possible values of z, is a Gaussian mixture of the form (15.6).

Answer on PDF