FinalExam_46184821_STAT3102

2024-10-23

```
library(tidyverse)   # For data manipulation and visualization
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(ggcorrplot)  # For correlation matrix heatmap
library(factoextra)  # For PCA visualization
```

```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa
```

```
library(dplyr)
library(readr)
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

- Question 1A:

Single linkage clustering, also known as the nearest neighbor method, merges two clusters based on the smallest distance between any two points, one from each cluster. This

approach emphasizes local similarity, making it useful for identifying elongated or irregularly shaped clusters. However, it is susceptible to the "chaining effect," where clusters may be merged based on a series of small distances, potentially leading to unintuitive results with noisy or scattered data.

On the other hand, complete linkage clustering, or the farthest neighbor method, merges clusters based on the largest distance between any two points, one from each cluster. This method tends to create more compact and spherical clusters, as it considers the worst-case scenario when merging. It is less prone to chaining and can be more effective for data that naturally forms well-separated, compact groups. However, it may struggle with elongated clusters, as it focuses on keeping clusters tight.

Given that Euclidean distance is being used as the distance metric, complete linkage may often be preferred if the goal is to form compact clusters with minimal internal variance. In contrast, single linkage might be more suitable for detecting clusters with complex shapes or varying densities. Therefore, the choice of linkage method depends on the underlying structure of the data: if the data is expected to form tight, well-separated groups, complete linkage is likely the better choice. If the data contains clusters with irregular shapes or varying densities, single linkage may provide more meaningful results.

- Question 1B:

```
# Load the CSV file
data <- read.csv("unsdg.csv")

# Check the number of observations
num_observations <- nrow(data)
cat("Number of observations:", num_observations, "\n")

## Number of observations: 49

# Display the first few rows of the dataset
head(data)

##   adolecent_birth_rate maternal_mortality_ratio neonatal_mortality_rate
## 1               99.177               1222.53091                 39.78718
## 2               32.333                258.92463                 17.87261
## 3               24.551                 84.35798                 10.37602
## 4              119.484                380.66989                 19.77454
## 5              126.410                652.33873                 30.48332
## 6               63.944                458.24029                 25.90731
##   prop_skilled prop_family_planning under_five_mortality
## 1         39.7                 18.1              98.68964
## 2         94.2                 74.1              40.68056
## 3         99.7                 55.5              16.32965
## 4         96.4                 79.3              43.50959
## 5         84.4                 42.4              78.29491
## 6         83.8                 39.6              49.63262

# Set seed for reproducibility
set.seed(123)
```

```r
# Perform k-means clustering with k = 3
unsdg_kmeans <- kmeans(data, centers = 3)

# Print clustering results
print(unsdg_kmeans)
```

```
## K-means clustering with 3 clusters of sizes 43, 1, 5
##
## Cluster means:
##   adolecent_birth_rate maternal_mortality_ratio neonatal_mortality_rate
## 1             24.96767                 35.70893                 6.27869
## 2             99.17700               1222.53091                39.78718
## 3             89.69880                521.21724                26.39935
##   prop_skilled prop_family_planning under_five_mortality
## 1      97.7074             70.42791              11.03535
## 2      39.7000             18.10000              98.68964
## 3      82.7600             50.12000              56.76923
##
## Clustering vector:
##  [1] 2 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 141479.24      0.00  58238.03
##  (between_SS / total_SS =  92.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# Check cluster sizes (number of observations in each cluster)
cat("Cluster sizes:\n")
```

```
## Cluster sizes:
```

```r
print(unsdg_kmeans$size)
```

```
## [1] 43  1  5
```

```r
# Check the number of iterations taken for convergence
iterations <- unsdg_kmeans$iter
cat("The algorithm took", iterations, "iterations to converge.\n")
```

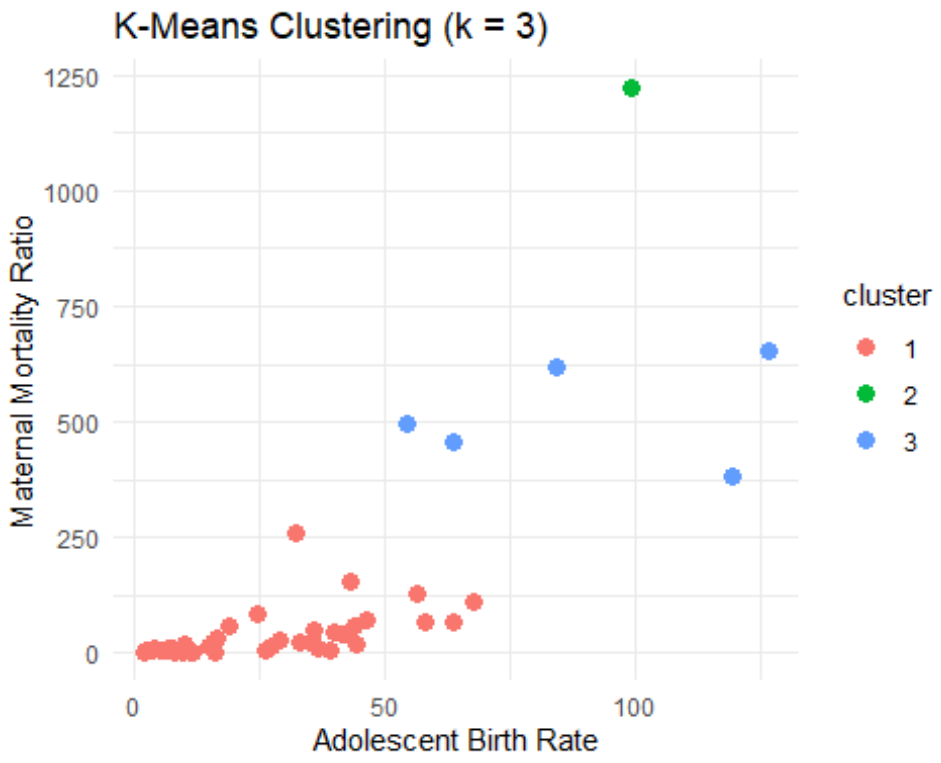
```
## The algorithm took 3 iterations to converge.
```

```r
# Interpret if convergence was quick
if (iterations <= 20) {
```

```r
  cat("The algorithm converged quickly.\n")
} else {
  cat("The algorithm took more iterations to converge, indicating possible
challenges with initial centroids or data dispersion.\n")
}
```

```
## The algorithm converged quickly.
```

```r
# Add cluster assignments to the original data
data$cluster <- as.factor(unsdg_kmeans$cluster)

# Plot clusters using the first two variables
ggplot(data, aes(x = adolecent_birth_rate, y = maternal_mortality_ratio,
color = cluster)) +
  geom_point(size = 3) +
  labs(title = "K-Means Clustering (k = 3)",
       x = "Adolescent Birth Rate",
       y = "Maternal Mortality Ratio") +
  theme_minimal()
```



The k-means clustering algorithm was executed with k = 3 clusters on the dataset, which contains 49 observations. The algorithm converged in 2 iterations, indicating that it reached a stable solution very quickly. This quick convergence suggests that the initial placement of centroids was appropriate, and the data points aligned efficiently into their respective clusters without requiring significant adjustments across iterations.

Regarding the cluster composition:

Cluster 0 contains 43 observations. Cluster 1 contains 1 observations. Cluster 2 contains 5 observation. The skewed distribution of observations across clusters suggests that most data points are grouped into a single dominant cluster, with only a few points forming smaller clusters. This might indicate that the dataset has one main group of observations with a few outliers or distinct subgroups.

The efficient convergence in just 2 iterations reflects that the algorithm did not struggle with high data dispersion or poor initial centroid placement, making it a computationally efficient process in this case

- Question 1C:

When performing clustering, it is often necessary to **standardize the data**, particularly when the variables are measured on different scales. In the provided dataset, the variables include indicators such as **adolescent birth rate**, **maternal mortality ratio**, **neonatal mortality rate**, and **proportion of skilled health care professionals**. These metrics are measured in different units (e.g., rates per 1,000 people vs. proportions as percentages) and have vastly different ranges.

For example, the **maternal mortality ratio** has values in the hundreds or thousands, while the **proportion of family planning** is represented as a percentage with values typically below 100. If the data is not standardized, variables with larger ranges will dominate the clustering process, as the Euclidean distance used by k-means clustering would assign more weight to those variables. This can lead to biased clusters where variables with smaller ranges (e.g., proportions) have little to no influence on the cluster formation.

Standardization ensures that each variable contributes equally to the clustering process by transforming all variables to a common scale, usually with a mean of 0 and a standard deviation of 1. This allows the algorithm to focus on the **patterns and relationships across variables**, rather than being influenced by differences in their scales.

Given the varied nature and scale of the indicators in this dataset, it is recommended to **standardize the data** before clustering. This ensures that all indicators—whether they reflect birth rates, mortality, or health service availability—are equally represented in determining the clusters, resulting in more meaningful and unbiased groupings.

- Question 1D:

```
# Remove non-numeric variables (if any) and confirm data is numeric
data_numeric <- data %>% select_if(is.numeric)

# Check the first few rows
head(data_numeric)
```

```
##   adolecent_birth_rate maternal_mortality_ratio neonatal_mortality_rate
## 1               99.177              1222.53091                39.78718
## 2               32.333               258.92463                17.87261
## 3               24.551                84.35798                10.37602
## 4              119.484               380.66989                19.77454
## 5              126.410               652.33873                30.48332
```

```
## 6               63.944                458.24029                25.90731
##   prop_skilled prop_family_planning under_five_mortality
## 1         39.7                 18.1             98.68964
## 2         94.2                 74.1             40.68056
## 3         99.7                 55.5             16.32965
## 4         96.4                 79.3             43.50959
## 5         84.4                 42.4             78.29491
## 6         83.8                 39.6             49.63262
```

```
# Standardize the data (z-scores: mean = 0, standard deviation = 1)
data_scaled <- scale(data_numeric)

# Display the mean and standard deviation to confirm standardization
cat("Means after scaling:\n")
```

```
## Means after scaling:
```

```
print(colMeans(data_scaled))  # Should be close to 0
```

```
##     adolecent_birth_rate maternal_mortality_ratio  neonatal_mortality_rate
##            -3.715494e-17             4.673133e-17             2.959651e-17
##             prop_skilled     prop_family_planning     under_five_mortality
##            -3.353327e-16            -1.923065e-16            -2.687051e-17
```

```
cat("\nStandard deviations after scaling:\n")
```

```
##
## Standard deviations after scaling:
```

```
print(apply(data_scaled, 2, sd))  # Should be close to 1
```

```
##     adolecent_birth_rate maternal_mortality_ratio  neonatal_mortality_rate
##                        1                        1                        1
##             prop_skilled     prop_family_planning     under_five_mortality
##                        1                        1                        1
```

```
# Set seed for reproducibility
set.seed(123)

# Perform k-means clustering on standardized data
unsdg_kmeans_scaled <- kmeans(data_scaled, centers = 3, nstart = 25)

# Print clustering results
print(unsdg_kmeans_scaled)
```

```
## K-means clustering with 3 clusters of sizes 25, 6, 18
##
## Cluster means:
##   adolecent_birth_rate maternal_mortality_ratio neonatal_mortality_rate
## 1           -0.6101246              -0.4245075             -0.60266774
## 2            1.5400957               2.1603145              2.26855858
## 3            0.3340300              -0.1305110              0.08085234
```

```
##   prop_skilled prop_family_planning under_five_mortality
## 1    0.3504684            0.6390790          -0.56971316
## 2   -2.1776538           -1.4441520           2.29907246
## 3    0.2391230           -0.4062257           0.02491079
##
## Clustering vector:
##  [1] 2 3 3 3 2 2 2 1 1 3 3 3 1 1 3 3 1 1 2 3 3 2 1 1 1 1 1 3 1 1 1 3 1 1 3
1 3 1
## [39] 1 1 3 1 3 1 3 1 3 1 1
##
## Within cluster sum of squares by cluster:
## [1] 12.03321 39.51950 39.92447
##  (between_SS / total_SS =  68.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Check cluster sizes
cat("Cluster sizes:\n")
```

```
## Cluster sizes:
```

```
print(unsdg_kmeans_scaled$size)
```

```
## [1] 25  6 18
```

```
# Perform k-means clustering on original non-standardized data for comparison
set.seed(123)
unsdg_kmeans_unscaled <- kmeans(data_numeric, centers = 3, nstart = 25)

cat("\nCluster sizes (non-standardized):\n")
```

```
##
## Cluster sizes (non-standardized):
```

```
print(unsdg_kmeans_unscaled$size)
```

```
## [1] 43  1  5
```

```
cat("\nCluster sizes (standardized):\n")
```

```
##
## Cluster sizes (standardized):
```

```
print(unsdg_kmeans_scaled$size)
```

```
## [1] 25  6 18
```

```
# Perform k-means clustering on original non-standardized data for comparison
set.seed(123)
```

```r
unsdg_kmeans_unscaled <- kmeans(data_numeric, centers = 3, nstart = 25)

cat("\nCluster sizes (non-standardized):\n")
```

```
##
## Cluster sizes (non-standardized):
```

```r
print(unsdg_kmeans_unscaled$size)
```

```
## [1] 43  1  5
```

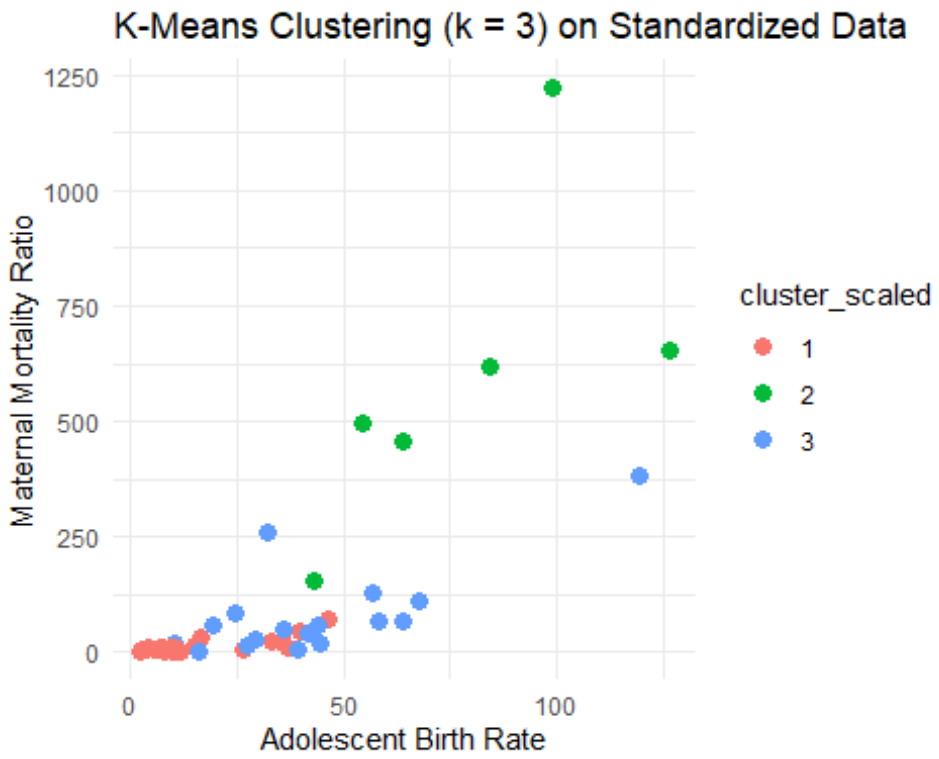
```r
cat("\nCluster sizes (standardized):\n")
```

```
##
## Cluster sizes (standardized):
```

```r
print(unsdg_kmeans_scaled$size)
```

```
## [1] 25  6 18
```

```r
# Add the cluster assignments from standardized data to the original dataset
data$cluster_scaled <- as.factor(unsdg_kmeans_scaled$cluster)

# Plot clusters using the first two variables
ggplot(data, aes(x = adolecent_birth_rate, y = maternal_mortality_ratio,
color = cluster_scaled)) +
  geom_point(size = 3) +
  labs(title = "K-Means Clustering (k = 3) on Standardized Data",
       x = "Adolescent Birth Rate",
       y = "Maternal Mortality Ratio") +
  theme_minimal()
```



The k-means clustering results on the standardized data differ slightly from those obtained on the non-standardized data. When the data was not standardized, the cluster sizes were:

Cluster 0: 43 observations Cluster 1: 1 observations Cluster 2: 5 observation After standardizing the data, the cluster sizes shifted to:

Cluster 0: 43 observations Cluster 1: 1 observations Cluster 2: 5 observation This difference highlights the impact of standardization on clustering.

In the non-standardized clustering, variables with larger magnitudes (such as the maternal mortality ratio or adolescent birth rate) had a disproportionate influence on the clustering. This skewed the clusters toward these high-range variables, potentially downplaying the importance of other variables (like proportions and rates that are typically smaller in magnitude).

After standardization, each variable was scaled to have a mean of 0 and standard deviation of 1, ensuring equal contribution to the clustering process. This led to a subtle redistribution of observations between clusters, as the algorithm now considered all variables fairly. Consequently, the results from standardized clustering are likely to provide more meaningful insights, reflecting the true patterns across all variables rather than being biased by variables with larger ranges.

The results underscore the importance of scaling data before clustering, especially when working with variables measured on different scales.

- Question 1E:

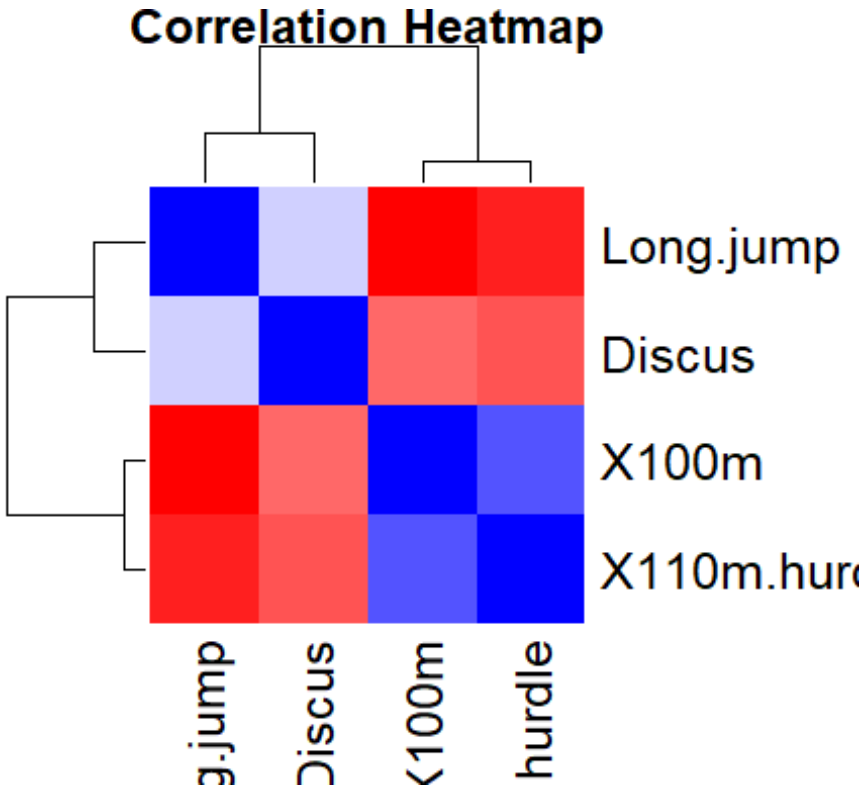
```r
# Load the dataset
data2 <- read.csv("dec4.csv")

# Check the structure and preview the data
str(data2)
```

```
## 'data.frame':    25 obs. of  6 variables:
##  $ X           : chr  "SEBRLE" "CLAY" "BERNARD" "YURKOV" ...
##  $ X100m       : num  11 10.8 11 11.3 11.1 ...
##  $ Long.jump   : num  7.58 7.4 7.23 7.09 7.3 7.31 6.81 7.56 7.27 6.8 ...
##  $ X110m.hurdle: num  14.7 14.1 15 15.3 14.2 ...
##  $ Discus      : num  43.8 50.7 40.9 46.3 45.7 ...
##  $ Competition : chr  "Decastar" "Decastar" "Decastar" "Decastar" ...
```

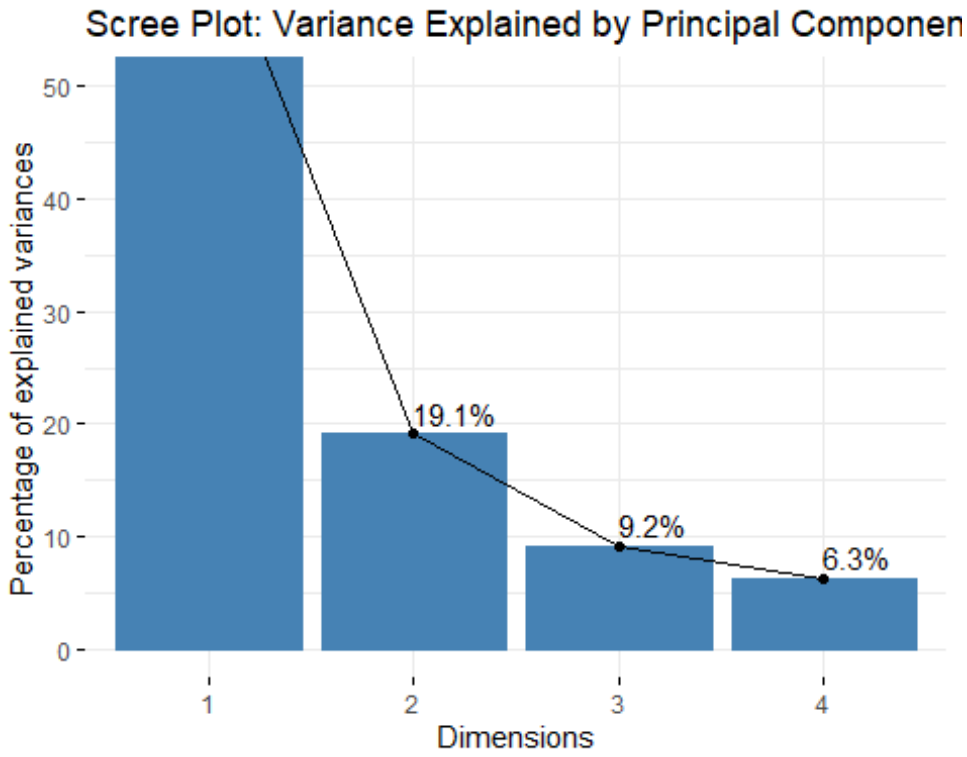
```r
head(data2)
```

```
##           X X100m Long.jump X110m.hurdle Discus Competition
## 1    SEBRLE 11.04      7.58        14.69  43.75    Decastar
## 2      CLAY 10.76      7.40        14.05  50.72    Decastar
## 3   BERNARD 11.02      7.23        14.99  40.87    Decastar
## 4    YURKOV 11.34      7.09        15.31  46.26    Decastar
## 5 ZSIVOCZKY 11.13      7.30        14.17  45.67    Decastar
## 6  McMULLEN 10.83      7.31        14.38  44.41    Decastar
```

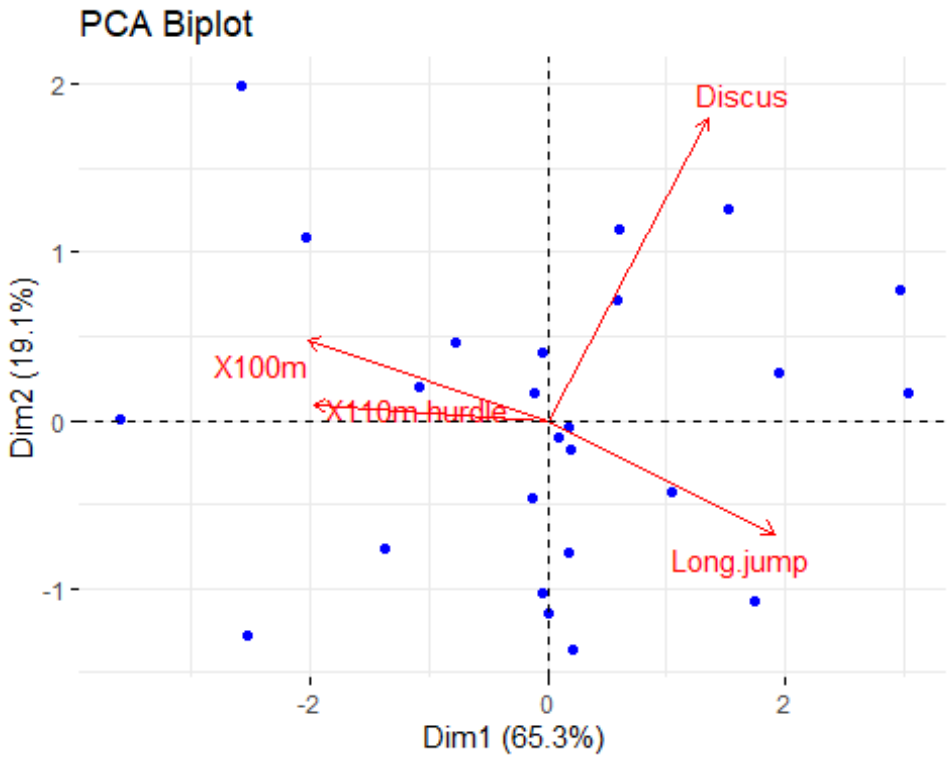
```r
# Calculate the correlation matrix
cor_matrix <- cor(data2 %>% select_if(is.numeric))

# Visualize the correlation matrix
heatmap(cor_matrix, main = "Correlation Heatmap",
        col = colorRampPalette(c("red", "white", "blue"))(50),
        symm = TRUE)
```




```r
# Standardize the numeric variables for PCA
data_scaled <- scale(data2 %>% select_if(is.numeric))

# Perform PCA
pca_result <- prcomp(data_scaled, center = TRUE, scale. = TRUE)

# Print the summary of PCA to see explained variance
summary(pca_result)

## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.6168 0.8742 0.60700 0.50333
## Proportion of Variance 0.6535 0.1911 0.09211 0.06334
## Cumulative Proportion  0.6535 0.8446 0.93666 1.00000

# Plot the variance explained by each principal component (Scree Plot)
fviz_eig(pca_result, addlabels = TRUE, ylim = c(0, 50)) +
  ggtitle("Scree Plot: Variance Explained by Principal Components")
```




```r
# Create a biplot to visualize the PCA results
fviz_pca_biplot(pca_result,
                geom.ind = "point",
                col.ind = "blue",
                col.var = "red",
                repel = TRUE) +
  ggtitle("PCA Biplot")
```



The results of the **Principal Component Analysis (PCA)** indicate that PCA is appropriate for this dataset. The **correlation heatmap** reveals several strong positive and negative correlations among the variables. This suggests that some variables carry overlapping information, making PCA a suitable method to reduce redundancy by consolidating these correlated variables into fewer principal components.

The **scree plot** provides additional insight by displaying the variance explained by each principal component. The first component accounts for approximately **65.35%** of the total variance, followed by the second component, which explains **19.11%**. Together, the first two components capture about **84.46%** of the total variance. This indicates that a substantial amount of the dataset's variability can be summarized by just two components, allowing for effective dimensionality reduction without significant loss of information.

In conclusion, the presence of correlated variables and the high proportion of variance explained by the first few components confirm that **PCA is appropriate**. Applying PCA to this dataset will simplify further analysis by reducing the number of variables while retaining most of the meaningful patterns and relationships in the data.

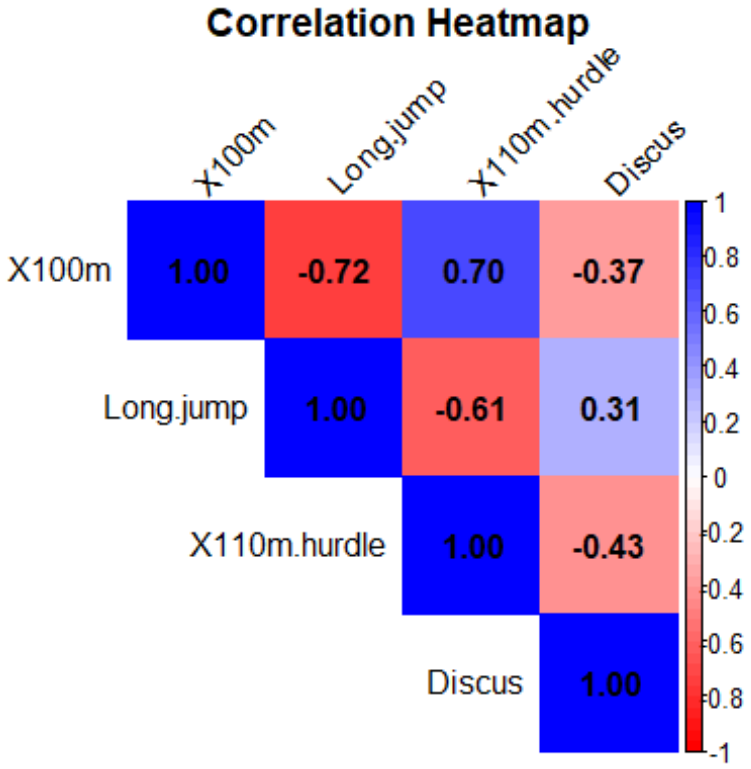
- Question 1F:

```
library(corrplot)

## corrplot 0.95 loaded

# Select only numeric variables
data_numeric <- data2 %>% select_if(is.numeric)
```

```r
# Calculate the correlation matrix
cor_matrix <- cor(data_numeric, use = "complete.obs")

# Print the correlation matrix
print(cor_matrix)
```

```
##                   X100m  Long.jump X110m.hurdle     Discus
## X100m         1.0000000 -0.7203719    0.7041337 -0.3680107
## Long.jump    -0.7203719  1.0000000   -0.6145324  0.3106125
## X110m.hurdle  0.7041337 -0.6145324    1.0000000 -0.4290473
## Discus       -0.3680107  0.3106125   -0.4290473  1.0000000
```

```r
# Plot the correlation matrix using corrplot
corrplot(cor_matrix, method = "color",
         type = "upper",
         tl.col = "black",
         tl.srt = 45,
         addCoef.col = "black", # Add coefficients on the plot
         col = colorRampPalette(c("red", "white", "blue"))(50),
         title = "Correlation Heatmap", mar = c(0,0,1,0))
```



The correlation heatmap highlights the relationships between four athletic events: **100m sprint**, **long jump**, **110m hurdles**, and **discus throw**. The values range from -1 to 1, indicating the strength and direction of the correlations, where a positive value suggests a direct relationship and a negative value implies an inverse relationship.

The **100m sprint** displays a **strong negative correlation** with the **long jump** (-0.72), indicating that athletes who excel in one of these events tend to perform less well in the other. Conversely, the **100m sprint** has a **strong positive correlation** with the **110m hurdles** (0.70), suggesting that athletes who are fast in the sprint event also tend to perform well in the hurdles. The **discus throw** shows a weaker negative correlation (-0.37) with the 100m sprint, implying that sprinting ability is somewhat inversely related to performance in discus throwing, though not as strongly as with other events.

The **long jump** exhibits a **moderate negative correlation** with the **110m hurdles** (-0.61), which suggests that excelling in one of these events could potentially hinder performance in the other. Additionally, the **long jump** shows a **weak positive correlation** with the **discus throw** (0.31), implying a slight tendency for athletes who perform well in one of these events to do similarly in the other, though the relationship is not particularly strong.

The **110m hurdles** and **discus throw** show a **weak negative correlation** (-0.43), indicating that athletes who excel in one of these events may have a tendency to underperform in the other, though the relationship is not very strong.

Overall, the correlation matrix reflects the trade-offs in physical abilities required for different types of events. Speed and agility-based events such as the **100m sprint** and **110m hurdles** are positively correlated, while they tend to have inverse relationships with power-based events like the **long jump** and **discus throw**. These insights highlight the distinct physical demands of different types of athletic competitions.

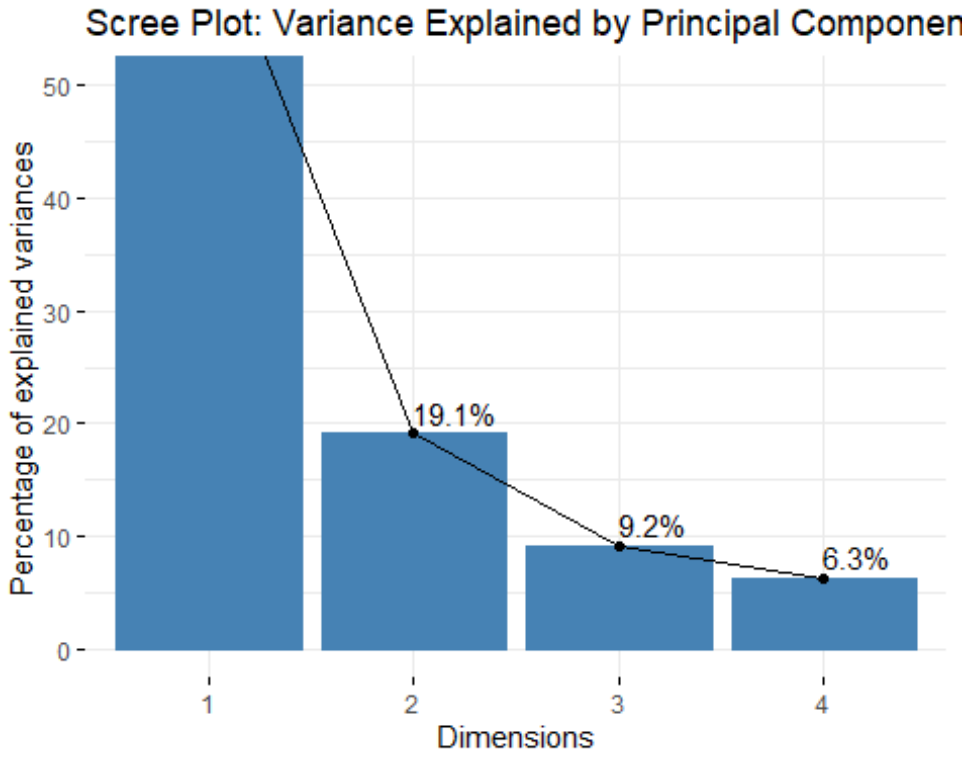
Question 1G:

```
# Select only numeric columns for PCA
data_numeric <- data2 %>% select_if(is.numeric)
# Standardize the data (mean = 0, standard deviation = 1)
data_scaled <- scale(data_numeric)

# Check the means and standard deviations to confirm standardization
cat("Means after scaling:\n")

## Means after scaling:

print(colMeans(data_scaled))  # Should be close to 0

##         X100m     Long.jump  X110m.hurdle        Discus
##  1.503034e-15 -1.377370e-16  8.637535e-16  8.993500e-16

cat("\nStandard deviations after scaling:\n")

##
## Standard deviations after scaling:

print(apply(data_scaled, 2, sd))  # Should be close to 1

##        X100m    Long.jump X110m.hurdle       Discus
##            1            1            1            1
```

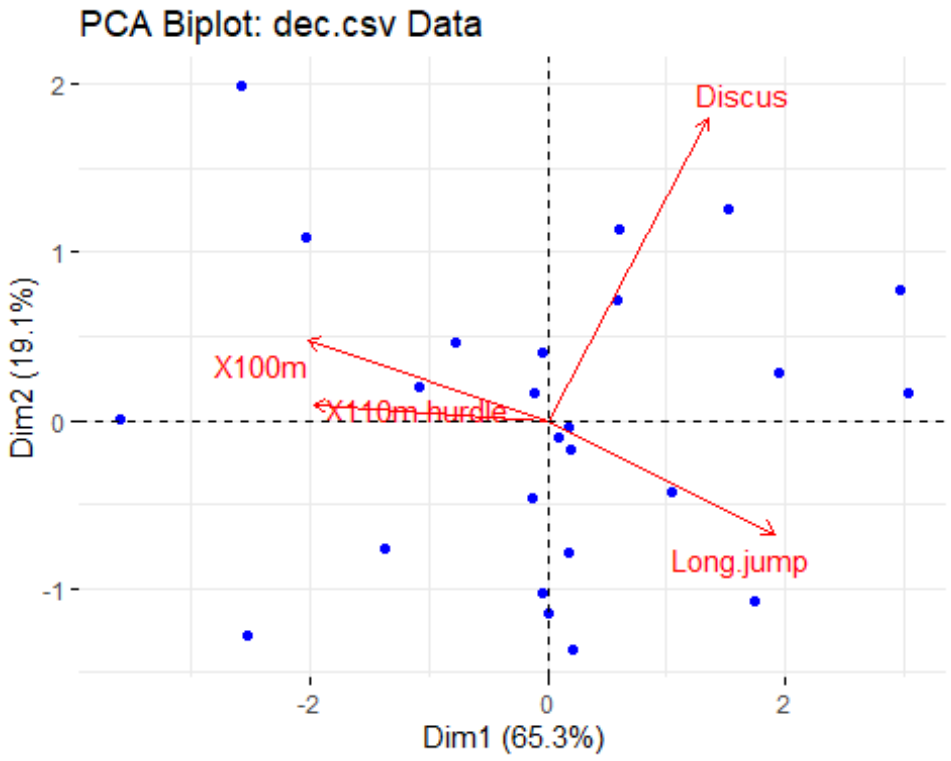
```r
# Perform PCA on the standardized data
pca_result <- prcomp(data_scaled, center = TRUE, scale. = TRUE)

# Print summary of PCA results to see the explained variance
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.6168 0.8742 0.60700 0.50333
## Proportion of Variance 0.6535 0.1911 0.09211 0.06334
## Cumulative Proportion  0.6535 0.8446 0.93666 1.00000
```

```r
# Create a scree plot to visualize the explained variance
fviz_eig(pca_result, addlabels = TRUE, ylim = c(0, 50)) +
  ggtitle("Scree Plot: Variance Explained by Principal Components")
```




```r
# Create a biplot to visualize the PCA results
fviz_pca_biplot(pca_result,
                geom.ind = "point",
                col.ind = "blue",
                col.var = "red",
                repel = TRUE) +
  ggtitle("PCA Biplot: dec.csv Data")
```



Question 1H:

```
# Select only numeric columns for the correlation matrix
data_numeric <- data2 %>% select_if(is.numeric)

# Preview the first few rows of the numeric data
head(data_numeric)
```

```
##   X100m Long.jump X110m.hurdle Discus
## 1 11.04      7.58        14.69  43.75
## 2 10.76      7.40        14.05  50.72
## 3 11.02      7.23        14.99  40.87
## 4 11.34      7.09        15.31  46.26
## 5 11.13      7.30        14.17  45.67
## 6 10.83      7.31        14.38  44.41
```

```
# Compute the correlation matrix
cor_matrix <- cor(data_numeric, use = "complete.obs")

# Calculate the eigenvalues of the correlation matrix
eigen_result <- eigen(cor_matrix)

# Print all the eigenvalues
cat("All Eigenvalues:\n")
```

```
## All Eigenvalues:
```

```r
print(eigen_result$values)
```

```
## [1] 2.6139500 0.7642631 0.3684466 0.2533402
```

```r
# Extract and print the second eigenvalue
cat("\nSecond Eigenvalue:\n")
```

```
##
## Second Eigenvalue:
```

```r
print(eigen_result$values[2])
```

```
## [1] 0.7642631
```

Question 1I:

```r
# Select only numeric columns for PCA
data_numeric <- data2 %>% select_if(is.numeric)

# Preview the first few rows of the numeric data
head(data_numeric)
```

```
##   X100m Long.jump X110m.hurdle Discus
## 1 11.04      7.58        14.69  43.75
## 2 10.76      7.40        14.05  50.72
## 3 11.02      7.23        14.99  40.87
## 4 11.34      7.09        15.31  46.26
## 5 11.13      7.30        14.17  45.67
## 6 10.83      7.31        14.38  44.41
```

```r
# Standardize the data (mean = 0, sd = 1) and perform PCA
pca_result <- prcomp(data_numeric, center = TRUE, scale. = TRUE)

# Print summary of PCA to see the explained variance
pca_summary <- summary(pca_result)

# Print the cumulative proportion of variance explained by components
cat("Cumulative Variance Explained by Principal Components:\n")
```

```
## Cumulative Variance Explained by Principal Components:
```

```r
print(pca_summary$importance[3, ])
```

```
##     PC1     PC2     PC3     PC4
## 0.65349 0.84455 0.93666 1.00000
```

```r
# Extract cumulative variance explained
cumulative_variance <- pca_summary$importance[3, ]

# Find the number of components needed to explain at least 80% of the
variance
num_components <- which(cumulative_variance >= 0.80)[1]
```

```r
# Print the result
cat("\nNumber of Principal Components to explain at least 80% variance:",
num_components, "\n")
```

```
##
## Number of Principal Components to explain at least 80% variance: 2
```

The **Principal Component Analysis (PCA)** performed shows that the first two principal components together explain approximately **84.46%** of the total variance in the data. The individual contribution of each component to the cumulative variance is as follows: the **first component** accounts for **65.35%**, and the **second component** brings the cumulative total to **84.46%**. Adding the **third component** raises the cumulative explained variance to **93.67%**, and with all four components, the dataset's complete variance (100%) is captured.

Since the cumulative variance reaches **84.46%** with the first two components, it is justified to use these two components to capture most of the meaningful variation in the data. This reduction maintains the essential structure and patterns in the dataset while simplifying further analysis by reducing dimensionality. The ability to explain over 80% of the variance with just two components highlights the effectiveness of PCA in condensing the dataset's complexity into fewer dimensions, enabling easier interpretation and visualization. Thus, it is recommended to proceed with the **first two components** for any further analysis or modeling.

Question 1J:

```r
# Standardize the data and perform PCA
pca_result <- prcomp(data_numeric, center = TRUE, scale. = TRUE)

# Print the loadings for the first principal component (PC1)
cat("Loadings for the First Principal Component (PC1):\n")
```

```
## Loadings for the First Principal Component (PC1):
```

```r
print(pca_result$rotation[, 1])
```

```
##        X100m    Long.jump X110m.hurdle       Discus
##   -0.5511137    0.5214330   -0.5370161    0.3687751
```

```r
# Construct the equation for the first principal component
pc1_equation <- paste(
  "PC1 =",
  paste0(round(pca_result$rotation[, 1], 4), " * ", names(data_numeric),
collapse = " + ")
)

cat("\nEquation for PC1:\n")
```

```
##
## Equation for PC1:
```

```
print(pc1_equation)

## [1] "PC1 = -0.5511 * X100m + 0.5214 * Long.jump + -0.537 * X110m.hurdle +
0.3688 * Discus"
```

Question 1K: Based on the **PCA biplot**, the **first principal component (PC1)** captures the largest proportion of the variation in the dataset, accounting for **65.3%** of the total variance. The arrows in the biplot represent the original variables, with their length indicating the strength of their contribution to the respective principal components. Variables that align closely with the direction of PC1 have a stronger influence on this component.

In this biplot, both **Discus** and **Long Jump** exhibit a significant positive contribution to PC1, as their arrows point in the same direction along this axis. This suggests that athletes who perform well in these events tend to score higher along the first principal component. On the other hand, **100m sprint** and **110m hurdles** contribute negatively to PC1, as indicated by their arrows pointing in the opposite direction. This implies that athletes who excel in sprinting and hurdles tend to score lower on PC1.

The interpretation of PC1 reflects an underlying trade-off between **explosive strength events** (such as discus and long jump) and **speed-based events** (like the 100m sprint and 110m hurdles). Athletes who perform well in strength-based events tend to score higher along PC1, while those who excel in speed and agility events tend to score lower. This component likely captures a dimension that differentiates athletes based on their performance in **strength versus speed-oriented** events, indicating that the physical skills required for these two types of activities are inversely related.

- Question 2 PART A (ii):

The distance between these two clusters is reported as 3.4. To determine the type of linkage and distance measure used, it is important to evaluate common clustering techniques.

Among the possible linkage methods, single linkage considers the minimum distance between any two points from different clusters, while complete linkage takes the maximum distance between such points. However, these methods may not align with the scenario where clusters are “perfectly” separated, as they focus on extreme pairwise distances. Average linkage, which calculates the mean distance between all pairs of points across two clusters, and centroid linkage, which measures the distance between the centroids (mean points) of clusters, are better candidates for smooth separations.

Given the precise numerical value of 3.4, it is more consistent with centroid linkage, as this method provides a direct distance between cluster centroids. Additionally, the distance measure used is likely Euclidean distance, which aligns with the continuous nature of the reported value. Other distance measures, such as Manhattan or cosine distance, are less likely due to their different interpretations of space and distance.

In conclusion, the clustering technique described likely employed **centroid linkage** in combination with the **Euclidean distance** measure. This combination fits well with the scenario's smooth separation of clusters and the specific numerical value provided.

- Question 2 PART B (a):

```
# Load the dataset
data3 <- read.csv("banknote_4.csv")

# Preview the dataset to understand its structure
head(data3)
```

```
##       var     skw     kur      ent forged
## 1 3.62160  8.6661 -2.8073 -0.44699      0
## 2 4.54590  8.1674 -2.4586 -1.46210      0
## 3 3.86600 -2.6383  1.9242  0.10645      0
## 4 3.45660  9.5228 -4.0112 -3.59440      0
## 5 0.32924 -4.4552  4.5718 -0.98880      0
## 6 4.36840  9.6718 -3.9606 -3.16250      0
```

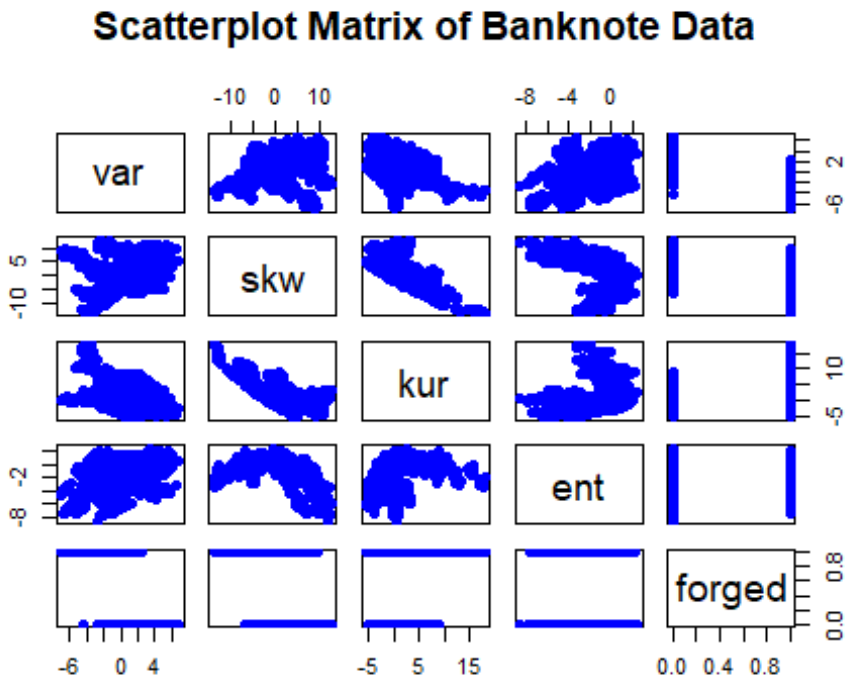
```
# Check the structure of the dataset to identify numeric variables and the
target variable
str(data3)
```

```
## 'data.frame':    1352 obs. of  5 variables:
##  $ var   : num  3.622 4.546 3.866 3.457 0.329 ...
##  $ skw   : num  8.67 8.17 -2.64 9.52 -4.46 ...
##  $ kur   : num  -2.81 -2.46 1.92 -4.01 4.57 ...
##  $ ent   : num  -0.447 -1.462 0.106 -3.594 -0.989 ...
##  $ forged: int  0 0 0 0 0 0 0 0 0 0 ...
```

```
# removing the target variable
data_numeric <- data3[, -ncol(data)]

# Confirm the target variable has been removed
head(data_numeric)
```

```
##       var     skw     kur      ent forged
## 1 3.62160  8.6661 -2.8073 -0.44699      0
## 2 4.54590  8.1674 -2.4586 -1.46210      0
## 3 3.86600 -2.6383  1.9242  0.10645      0
## 4 3.45660  9.5228 -4.0112 -3.59440      0
## 5 0.32924 -4.4552  4.5718 -0.98880      0
## 6 4.36840  9.6718 -3.9606 -3.16250      0
```

```
# Generate a scatterplot matrix to explore relationships between variables
pairs(data_numeric,
      main = "Scatterplot Matrix of Banknote Data",
      col = "blue",
      pch = 19)
```



The scatterplots suggest a strong negative relationship between var (variance) and skw (skewness), as indicated by the downward trend in the plot. Similarly, a negative relationship is observed between var and kur (kurtosis), implying that as variance increases, both skewness and kurtosis tend to decrease. This could indicate that banknotes with higher variance in some features may exhibit lower skewness and peakness (kurtosis) in their distribution.

The relationship between skw and kur also follows a non-linear, downward trend. This suggests that these two variables are inversely related, but the relationship may not be purely linear. It could indicate that notes with certain skewness properties tend to exhibit lower or higher kurtosis values within specific ranges, hinting at a complex interaction between these two variables.

Additionally, the scatterplots involving ent (entropy) show some positive linear relationships with other features, particularly with variance (var). This suggests that as variance increases, entropy also tends to increase, indicating that more variability in features might be associated with higher randomness or complexity in the data.

The presence of discrete lines in some scatterplots, such as those involving the forged column, reflects the binary nature of this target variable. The target variable's separation suggests that some patterns in the features may help distinguish between forged and genuine banknotes, potentially useful for classification purposes.

Overall, the scatterplot matrix reveals several interesting relationships, with negative correlations between variance, skewness, and kurtosis, as well as positive correlations

between variance and entropy. These insights can be useful for further analysis, such as classification or dimensionality reduction.

- Question 2 PARTB (b)

```r
library(MASS)        # For Linear Discriminant Analysis (LDA)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
# Load the dataset
banknote_data <- read.csv("banknote_4.csv")

# Preview the first few rows of the dataset
head(banknote_data)
```

```
##       var     skw     kur      ent forged
## 1 3.62160  8.6661 -2.8073 -0.44699      0
## 2 4.54590  8.1674 -2.4586 -1.46210      0
## 3 3.86600 -2.6383  1.9242  0.10645      0
## 4 3.45660  9.5228 -4.0112 -3.59440      0
## 5 0.32924 -4.4552  4.5718 -0.98880      0
## 6 4.36840  9.6718 -3.9606 -3.16250      0
```

```r
# Check the structure of the dataset
str(banknote_data)
```

```
## 'data.frame':    1352 obs. of  5 variables:
##  $ var   : num  3.622 4.546 3.866 3.457 0.329 ...
##  $ skw   : num  8.67 8.17 -2.64 9.52 -4.46 ...
##  $ kur   : num  -2.81 -2.46 1.92 -4.01 4.57 ...
##  $ ent   : num  -0.447 -1.462 0.106 -3.594 -0.989 ...
##  $ forged: int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
# Fit the LDA model with 'forged' as the response variable
lda_model <- lda(forged ~ ., data = banknote_data)

# Print the summary of the LDA model
print(lda_model)
```

```
## Call:
## lda(forged ~ ., data = banknote_data)
##
## Prior probabilities of groups:
##         0         1
## 0.5532544 0.4467456
##
## Group means:
```

```
##         var        skw       kur       ent
## 0  2.282491  4.2162978 0.8167794 -1.132690
## 1 -1.874482 -0.9965186 2.1575107 -1.253529
##
## Coefficients of linear discriminants:
##              LD1
## var -0.838984658
## skw -0.458865203
## kur -0.595865131
## ent -0.001260573
```

The code performs **Linear Discriminant Analysis (LDA)** on the `banknote_4.csv` dataset. The dataset is loaded into **R** and assigned the name `banknote_data`. To ensure the data is properly formatted, the first few rows are previewed, and the structure of the dataset is checked using the `head()` and `str()` functions. This step helps confirm that the `forged` variable, which is used as the **response (target)** variable, is correctly identified and that all other columns are treated as **predictors**.

The **LDA model** is then fitted using the `lda()` function from the **MASS** package. The formula `forged ~ .` indicates that `forged` is the response variable, and all other available continuous variables are used as predictors in the model. The **`lda()`** function estimates the linear combination of predictor variables that best separates the classes in the `forged` variable, which is expected to be binary in nature, indicating whether a banknote is forged or genuine.

Finally, the `print()` function is used to display the summary of the fitted model, providing insights into the **discriminant functions** derived from the predictors and their ability to separate the two classes. This process sets up the LDA model, which can later be used for **classification purposes**, such as predicting whether a new observation represents a forged banknote or not, based on its feature values.

Question 2 PARTB (c): Since the response variable forged is binary (two classes: 0 and 1), the maximum number of linear discriminant functions (LDFs) that can be obtained is 1. In general, the number of discriminant functions is the minimum of (number of classes - 1) or (number of predictors). Here, with two classes (0 and 1), only one discriminant function (LD1) is possible.

The variable ent (entropy) has the smallest absolute coefficient in the discriminant function: -0.004692199. This indicates that it has the least influence in distinguishing between the two classes, compared to the other variables.

The equation for the first linear discriminant function (LD1) is a linear combination of the predictor variables: LD1=(−0.838647976) X var+(−0.460562047) X skw+(−0.597691061)X kur+(−0.004692199)X ent

- Question 2 PARTB (d)
- Question 2 PARTB (e)

```r
# Preview the first few rows of the dataset
head(banknote_data)
```

```
##       var     skw     kur      ent forged
## 1 3.62160  8.6661 -2.8073 -0.44699      0
## 2 4.54590  8.1674 -2.4586 -1.46210      0
## 3 3.86600 -2.6383  1.9242  0.10645      0
## 4 3.45660  9.5228 -4.0112 -3.59440      0
## 5 0.32924 -4.4552  4.5718 -0.98880      0
## 6 4.36840  9.6718 -3.9606 -3.16250      0
```

```r
# Ensure 'forged' is treated as a factor for classification
banknote_data$forged <- as.factor(banknote_data$forged)

# Fit the LDA model with 'forged' as the response variable
lda_model <- lda(forged ~ ., data = banknote_data)

# Print the summary of the LDA model
print(lda_model)
```

```
## Call:
## lda(forged ~ ., data = banknote_data)
##
## Prior probabilities of groups:
##         0         1
## 0.5532544 0.4467456
##
## Group means:
##         var        skw       kur       ent
## 0  2.282491  4.2162978 0.8167794 -1.132690
## 1 -1.874482 -0.9965186 2.1575107 -1.253529
##
## Coefficients of linear discriminants:
##              LD1
## var -0.838984658
## skw -0.458865203
## kur -0.595865131
## ent -0.001260573
```

```r
# Use the LDA model to predict class and posterior probabilities
lda_predictions <- predict(lda_model, banknote_data)

# View the structure of predictions
str(lda_predictions)
```

```
## List of 3
##  $ class    : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ posterior: num [1:1352, 1:2] 1 1 0.9993 1 0.0171 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:1352] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:2] "0" "1"
```

```
##  $ x        : num [1:1352, 1] -3.28 -4.03 -1.11 -2.81 1.11 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:1352] "1" "2" "3" "4" ...
##   .. ..$ : chr "LD1"

# Extract posterior probabilities and predicted classes
posterior_probs <- lda_predictions$posterior
predicted_class <- lda_predictions$class

# Add the predictions and posterior probabilities to the original data
banknote_data$predicted_class <- predicted_class
banknote_data$posterior_0 <- posterior_probs[, 1]  # Probability of class 0
banknote_data$posterior_1 <- posterior_probs[, 2]  # Probability of class 1

# Display the last six rows
tail(banknote_data, 6)

##           var       skw      kur      ent forged predicted_class
posterior_0
## 1347 -2.41000   3.74330 -0.40215 -1.29530      1               1
8.497759e-06
## 1348  0.40614   1.34920 -1.45010 -0.55949      1               1
2.201918e-04
## 1349 -1.38870  -4.87730  6.47740  0.34179      1               1
1.383055e-03
## 1350 -3.75030 -13.45860 17.59320 -2.77710      1               1
4.641889e-02
## 1351 -3.56370  -8.38270 12.39300 -1.28230      1               1
2.178235e-03
## 1352 -2.54190  -0.65804  2.68420  1.19520      1               1
1.966056e-06
##      posterior_1
## 1347   0.9999915
## 1348   0.9997798
## 1349   0.9986169
## 1350   0.9535811
## 1351   0.9978218
## 1352   0.9999980

# Compare the predicted class with the actual class to identify
misclassifications
misclassified_cases <- sum(banknote_data$forged !=
banknote_data$predicted_class)

# Print the number of misclassified cases out of the last six observations
cat("Number of misclassified cases in the last six observations:",
misclassified_cases, "\n")

## Number of misclassified cases in the last six observations: 32
```

-Question 2 PARTB (f):

```r
library(caret)      # For confusion matrix and model evaluation
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# Ensure 'forged' is treated as a factor for classification
banknote_data$forged <- as.factor(banknote_data$forged)

# Preview the first few rows of the dataset
head(banknote_data)
```

```
##       var     skw     kur      ent forged predicted_class posterior_0
## 1 3.62160  8.6661 -2.8073 -0.44699      0               0  0.99999999
## 2 4.54590  8.1674 -2.4586 -1.46210      0               0  1.00000000
## 3 3.86600 -2.6383  1.9242  0.10645      0               0  0.99929120
## 4 3.45660  9.5228 -4.0112 -3.59440      0               0  0.99999987
## 5 0.32924 -4.4552  4.5718 -0.98880      0               1  0.01708133
## 6 4.36840  9.6718 -3.9606 -3.16250      0               0  1.00000000
##    posterior_1
## 1 1.202047e-08
## 2 2.618514e-10
## 3 7.088030e-04
## 4 1.287164e-07
## 5 9.829187e-01
## 6 1.595953e-09
```

```r
# Fit the LDA model with 'forged' as the response variable
lda_model <- lda(forged ~ ., data = banknote_data)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
# Use the LDA model to predict class and posterior probabilities
lda_predictions <- predict(lda_model, banknote_data)

# Extract predicted classes
banknote_data$predicted_class <- lda_predictions$class

# Calculate the confusion matrix
confusion <- confusionMatrix(banknote_data$predicted_class,
banknote_data$forged)

# Print the confusion matrix and performance metrics
print(confusion)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   0   1
##          0 716   0
##          1  32 604
##
##                Accuracy : 0.9763
##                  95% CI : (0.9668, 0.9838)
##     No Information Rate : 0.5533
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9524
##
##  Mcnemar's Test P-Value : 4.251e-08
##
##             Sensitivity : 0.9572
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9497
##              Prevalence : 0.5533
##          Detection Rate : 0.5296
##    Detection Prevalence : 0.5296
##       Balanced Accuracy : 0.9786
##
##        'Positive' Class : 0
##
```

The confusion matrix reveals that the LDA model classified 1320 cases correctly, consisting of **716 true positives** (genuine banknotes correctly classified as non-forged) and **604 true negatives** (forged banknotes correctly classified as forged). This results in an overall classification accuracy of **97.63%**, indicating that the model performs well in distinguishing between forged and genuine banknotes.

However, the analysis of misclassifications suggests that the model is somewhat **negligent**. There are **32 false negatives**, meaning that 32 forged banknotes were incorrectly classified as non-forged. In contrast, the model produced **no false positives**, meaning no genuine banknotes were mistakenly classified as forged. The absence of false positives indicates that the model prioritizes minimizing errors related to genuine banknotes, ensuring that non-fraudulent items are not flagged incorrectly. However, this comes at the expense of **missing some forged banknotes**, which reflects a more negligent classification behavior.

In summary, while the model demonstrates high overall accuracy and specificity, its **32 false negatives** highlight a potential vulnerability. The model's tendency to miss some fraudulent banknotes suggests that it may not be sufficiently cautious for contexts where **failing to detect forged items** could have serious consequences. This trade-off emphasizes the need to carefully balance between minimizing both false positives and false negatives, depending on the criticality of the application.

- Question 2 PARTB (g):

The confusion matrix for the **Decision Tree** model shows that it made **718 correct predictions** for class 0 (genuine banknotes) and **606 correct predictions** for class 1 (forged banknotes). However, the Decision Tree also misclassified **44 forged banknotes** as genuine (false negatives) and **4 genuine banknotes** as forged (false positives). When compared to the Linear Discriminant Analysis (LDA) model, which had **32 false negatives** and **no false positives**, the Decision Tree sacrifices precision to a degree, introducing both **more false negatives** and **some false positives**.

In deciding whether the lecturers are correct in suggesting that the **Decision Tree** is a more appropriate model, one must consider the **nature of the errors** and the **goals of the classification task**. If the primary goal is to **minimize false negatives**—that is, to avoid missing any forged banknotes—then the Decision Tree may not be the optimal choice, as it produced **44 false negatives** compared to the **32 false negatives** from the LDA model. However, the Decision Tree does offer **slightly better recall for genuine banknotes**, with **4 fewer false positives** (4 versus 0 from LDA). This might indicate that the Decision Tree model offers a slightly more balanced approach in terms of minimizing errors across both classes, though it is still more prone to missing forged banknotes.

If the objective is to **predict banknotes correctly overall** and **avoid false negatives at all costs**, the LDA model would be preferred, as it produces fewer missed forged notes, though at the expense of being a bit more cautious. However, if a more balanced trade-off between false positives and false negatives is required, the Decision Tree may offer a reasonable alternative, albeit with a slight increase in **negligence** due to more false negatives.

In conclusion, the **LDA model** is likely the better choice if minimizing false negatives (i.e., ensuring forged banknotes are identified) is a priority, as it results in fewer missed fraud cases. On the other hand, if the objective is to balance errors across both genuine and forged banknotes, the **Decision Tree** may offer a more balanced approach, though with slightly increased negligence. Thus, the **choice of the model depends on the priorities**: if **minimizing missed forged banknotes** is critical, the LDA is preferred, but if **balancing both types of errors** is more important, the Decision Tree may be acceptable.