ASSIGNMENT

ANS 1)

#ANS part-a

library(readxl)

library(dplyr)

file\_path <- "D:/New folder/vehicles.xlsx"

data<- read\_excel(file\_path)

# Extract the input variables

X <- data[, 1:18]

# Detect outliers using the Z-score method

outliers <- as.vector(which(apply(X, 2, function(x) any(abs(x) > 3))))

# Remove outliers

cleaned\_X <- X[-outliers, ]

# Perform scaling using scale()

scaled\_X <- scale(X)

1. Scaling refers to the process of transforming the variables in your dataset to a specific range.

Removing outliers is the process of removing outliers which are data points that deviate significantly from the majority of the data. Now the importance of the order.

Scaling before removing outliers: Outliers can have an impact on the scaling process if you scale the variables first and then get rid of them. By definition, outliers have extreme values that can make the scale of the variables stretch. Outliers can have a greater impact on the scaling process if they are removed before the variables are scaled. A biased scaling transformation that may not accurately represent the majority of the data can result from this.

Prior to scaling, remove outliers: However, if outliers are removed prior to scaling, extreme values that could disproportionately affect scaling can be eliminated. This can ensure that the scaling transformation is based on the majority of the data by first removing outliers, which makes it possible to scale the variables in a more representative manner.

#ANS part-b

data\_subset<-scaled\_X

library(NbClust)

library(cluster)

library(factoextra)

#THE FOUR AUTOMATED TOOLS

# 1. NBclust method

nb\_results <- NbClust(data\_subset, diss = NULL, distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans")

nb\_optimal\_clusters <- nb\_results$Best.nc

cat("Number of clusters suggested by NBclust:", nb\_optimal\_clusters, "\n")

# 2. Elbow method

fviz\_nbclust(data\_subset, kmeans, method = "wss") + ggtitle("Elbow Method")

# 3. Gap statistics method

gap\_stat <- clusGap(data\_subset, FUNcluster = kmeans, K.max = 10, B = 50)

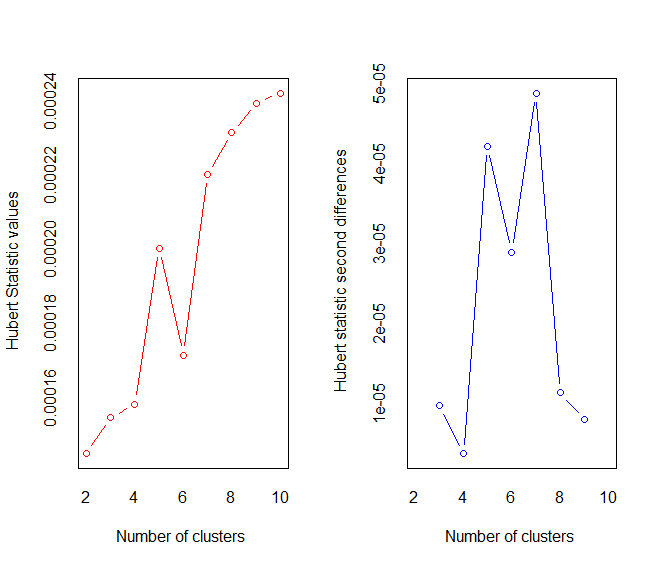
print(gap\_stat$Tab)

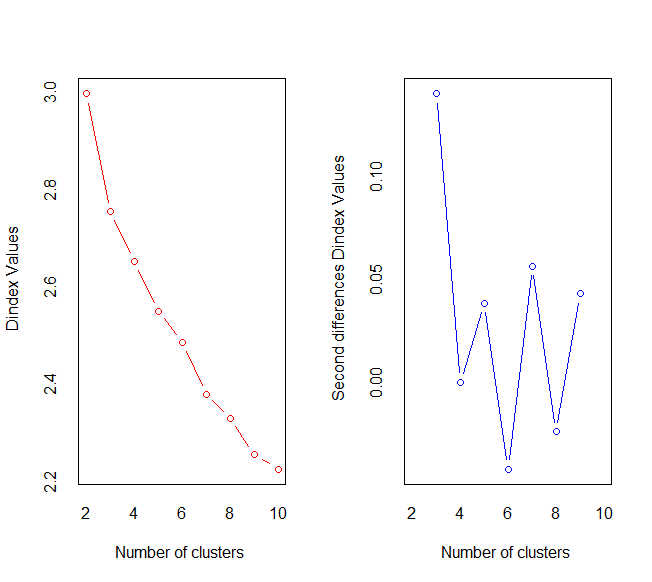
fviz\_gap\_stat(gap\_stat)

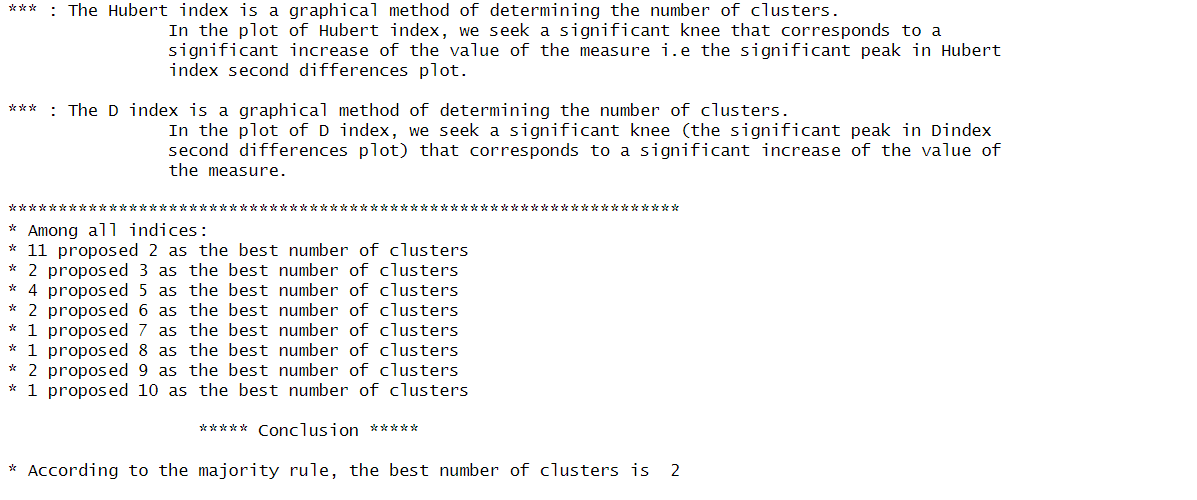
# 4. Silhouette method

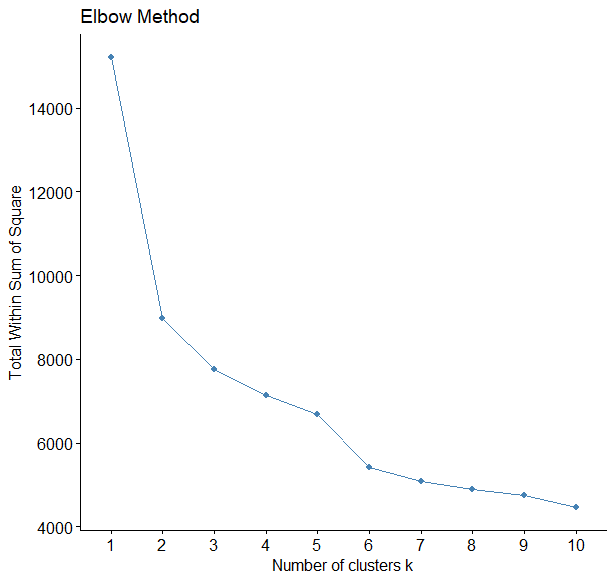
fviz\_nbclust(data\_subset, kmeans, method = "silhouette")

**NBclust:**

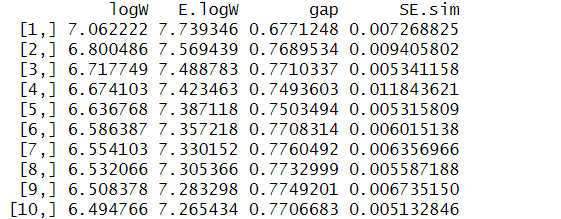
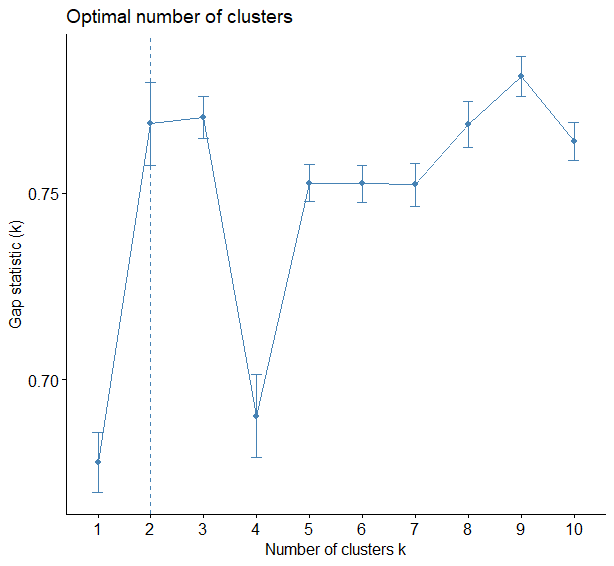
****

****

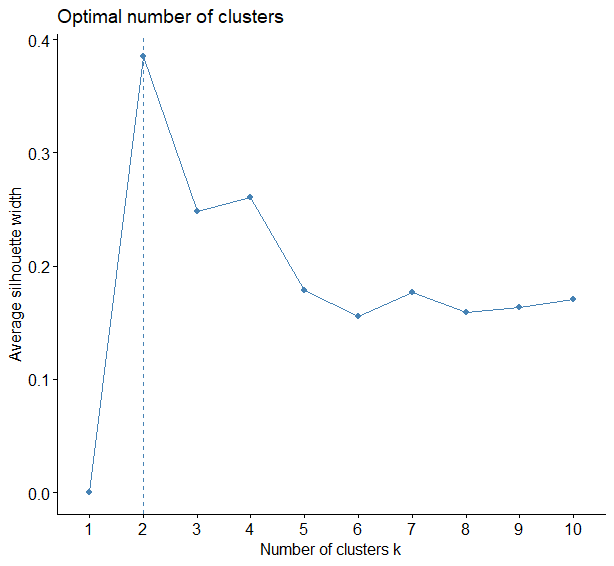
 **Elbow Method:**

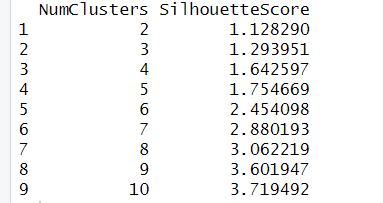
****

**Gap Statistics:**

****

**Silhouette Scores:**

****

****

These are the results of the automated tools:   
(I) NBclust: According to NBclust, there are different opinions among the indices. However, the majority rule suggests that the best number of clusters is 2.

(II) Elbow: Elbow method shows a graph where we can see that as the number of clusters increase the within sum of squares decreases. However the biggest drop is from k=2 as it looks like the hinge part of an elbow. Hence this suggests that the best number of clusters is 2.

(III) Gap Statistics: The Gap Statistics calculates the gap values for clustering k=1 to 10. The table shows the logW, E.logW, gap, and SE.sim values. The table indicates that k=2 might be a good number of clusters. The graph also confirms this.

(IV) Silhouette: The silhouette method shows that the best value of k is 2 as it shows the highest average silhouette width 0.38.

Based on the results from the four automated tools, there is a common indication that the best number of clusters is around 2 or 3. Hence we choose k=2 as majority of the automated tools choose k=2.

#ANS part c

library(cluster)

k<-2

kmeans\_model <- kmeans(data\_subset, centers = k, nstart = 10)

cluster\_centers <- kmeans\_model$centers

cat("Cluster Centers:\n")

print(cluster\_centers)

cluster\_assignments <- kmeans\_model$cluster

# Display the clustered results

cat("Cluster Assignments:\n")

print(cluster\_assignments)

# Calculate the within-cluster sum of squares (WSS)

wss <- kmeans\_model$tot.withinss

# Calculate the between-cluster sum of squares (BSS)

tss <- sum(kmeans\_model$tot.withinss) + sum(kmeans\_model$betweenss)

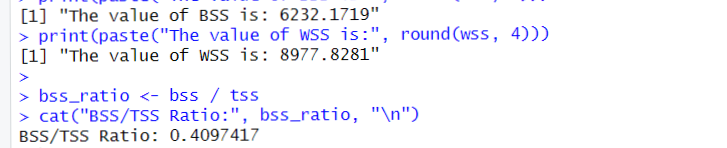
bss <- tss – wss

print(paste("The value of BSS is:", round(bss, 4)))

print(paste("The value of WSS is:", round(wss, 4)))

bss\_ratio <- bss / tss

cat("BSS/TSS Ratio:", bss\_ratio, "\n")



The BSS (Between Sum of Squares) value of 6232.17 indicates the total sum of squared distances between the cluster centroids and the overall mean. It represents the amount of variance explained by the clustering.

The WSS (Within Sum of Squares) value of 8977.83 represents the total sum of squared distances within each cluster. It measures the compactness of the clusters and represents the unexplained variance within each cluster.

The BSS/TSS (Total Sum of Squares) Ratio of 0.4097 indicates the proportion of variance in the data that is explained by the clustering. It suggests that the clustering algorithm has captured about 40.97% of the total variability in the dataset.

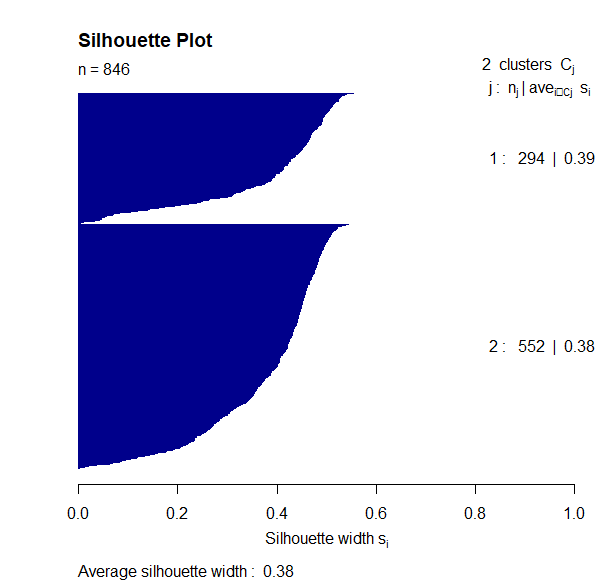
1. #ANS part d

silhouette\_scores <- silhouette(kmeans\_model$cluster, dist(data\_subset))

plot(silhouette\_scores, main = "Silhouette Plot",border=NA,col="darkblue")

avg\_sil\_width <- mean(silhouette\_scores[, "sil\_width"])

print(paste("The average silhouette width is:", round(avg\_sil\_width, 4)))





The average silhouette width is a measure of the quality or coherence of the obtained clusters. It quantifies how well each data point in a cluster is separated from data points in other clusters.

A value close to 1 indicates that the data points are well-clustered and properly separated from other clusters, indicating high quality clusters. Conversely, a value close to -1 suggests that data points might have been assigned to the wrong clusters, indicating poor quality clusters.

The overall average silhouette width for both clusters is calculated to be 0.3849. Based on these results, the clusters seem to have moderate coherence, with average silhouette widths around 0.38-0.39. It implies that there is room for improvement in the clustering results to achieve higher-quality clusters with better separation and cohesion.

ANS 2

1. # ANS 2 part e

pca <- prcomp(data\_subset[, -19], scale = TRUE)

eigenvalues <- pca$sdev^2

eigenvectors <- pca$rotation

print(paste0(" These are the eigenvalues: ", eigenvalues))

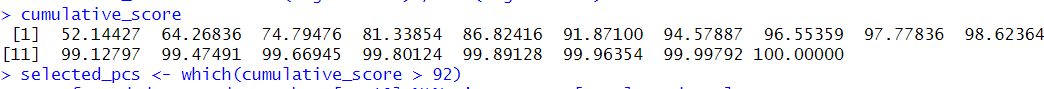
print(paste0(" These are the eigenvectors: ", eigenvectors))

cumulative\_score <- cumsum(eigenvalues) / sum(eigenvalues) \* 100

cumulative\_score

selected\_pcs <- which(cumulative\_score > 92)

transformed\_data <- data\_subset[, -19] %\*% eigenvectors[, selected\_pcs]



The cumulative score represents the amount of variance explained by each PC and provides insights into how much information is retained as we include more PCs. By setting a cumulative score threshold of 92%, we aim to select a sufficient number of PCs that capture a significant portion of the dataset's overall variability.

Choosing a cumulative score threshold of 92% ensures that we retain a large proportion of the original dataset's information while reducing its dimensionality. This reduction in dimensionality helps in simplifying the dataset and potentially removing noise or irrelevant features.

# ANS 2 part f

library(NbClust)

library(cluster)

#NBclust

nb\_results <- NbClust(transformed\_data, diss = NULL, distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans")

#ELBOW

fviz\_nbclust(transformed\_data, kmeans, method = "wss") + ggtitle("Elbow Method")

#Gap Statistics

gap\_stat <- clusGap(transformed\_data, FUNcluster = kmeans, K.max = 10, B = 50)

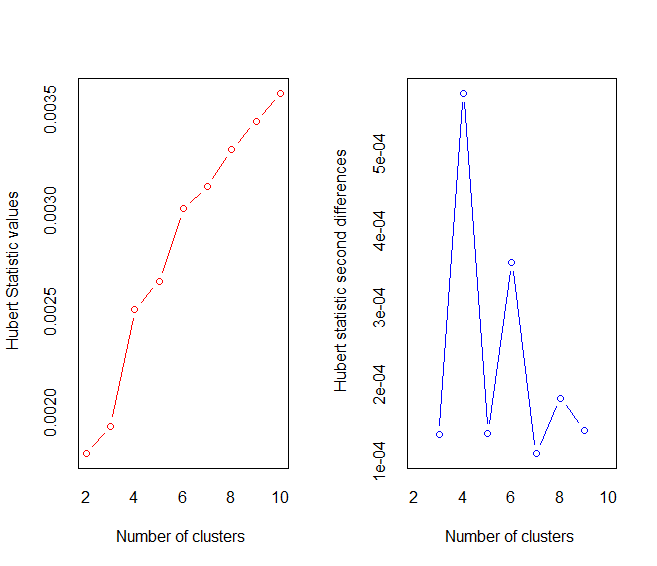
print(gap\_stat$Tab)

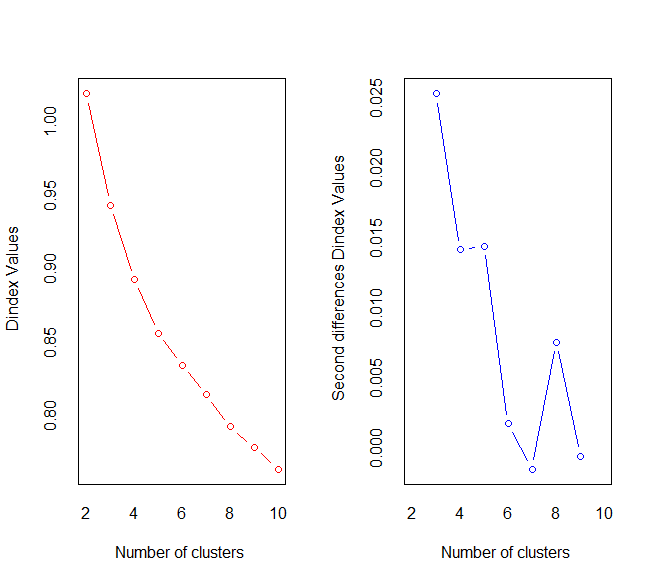
fviz\_gap\_stat(gap\_stat)

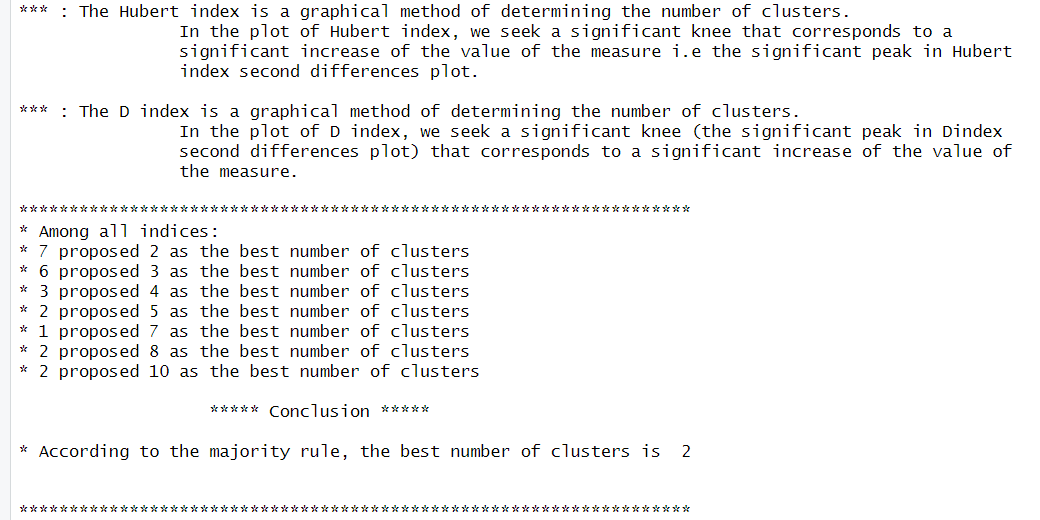
#Silhouette

fviz\_nbclust(transformed\_data, kmeans, method = "silhouette")

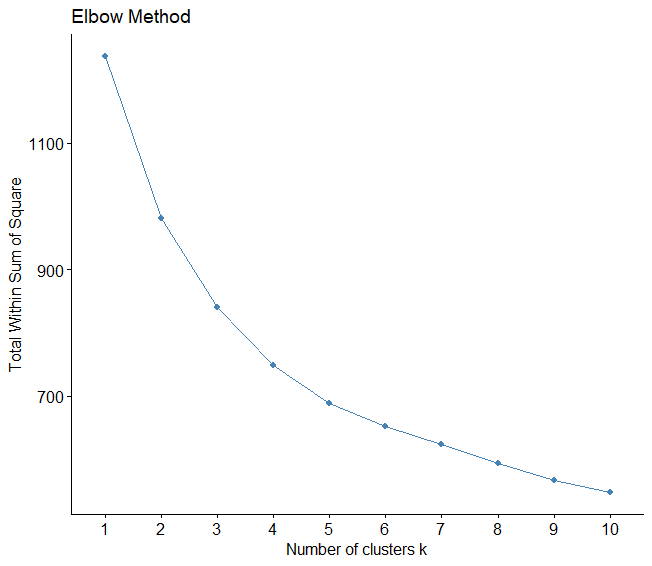
**NBclust:**

****

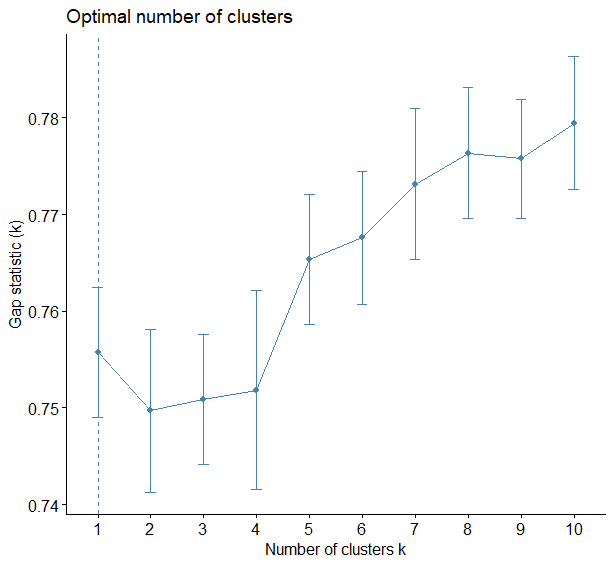


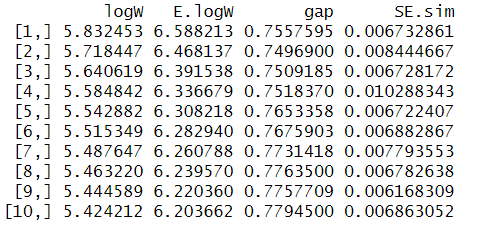


**ELBOW:**

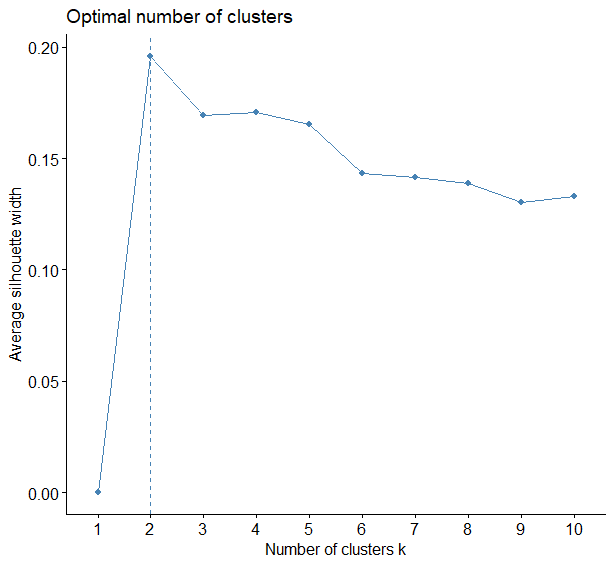


**GAP Statistics:**





**Silhouette:**



These are the results of the automated tools:   
 (I) NBclust: According to NBclust, there are different opinions among the indices. However, the

majority rule suggests that the best number of clusters is 2.

(II) Elbow: Elbow method shows a graph where we can see that as the number of clusters increase the within sum of squares decreases. However we can see that k=4 is most suitable as it looks like the hinge part of an elbow. Hence this suggests that the best number of clusters is 4.

(III) Gap Statistics: The Gap Statistics calculates the gap values for clustering k=1 to 10. The table shows the logW, E.logW, gap, and SE.sim values. Based on the table, it looks like that k=1 might be the best option. The graph also reassures this finding

(IV) Silhouette: The silhouette method shows that the best value of k is 2 as it shows the highest average silhouette width 0.18.

Based on the results from the four automated tools, there is a common indication that the best number of clusters is 2 or 4. Hence we choose k=2 as majority of the automated tools choose k=2.

# ANS 2 part g

k <- 2

kmeans\_model <- kmeans(transformed\_data,centers = k, nstart = 10)

cluster\_centers <- kmeans\_model$centers

cat("Cluster Centers:\n")

print(cluster\_centers)

cluster\_assignments <- kmeans\_model$cluster

# Display the clustered results

cat("Cluster Assignments:\n")

print(cluster\_assignments)

# Calculate the within-cluster sum of squares (WSS)

wss <- kmeans\_model$tot.withinss

# Calculate the between-cluster sum of squares (BSS)

tss <- sum(kmeans\_model$tot.withinss) + sum(kmeans\_model$betweenss)

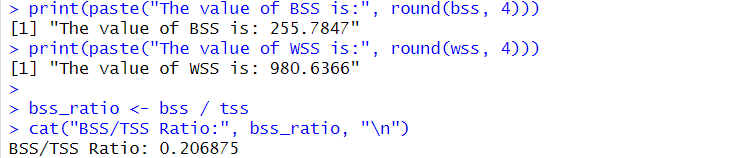
bss <- tss - wss

print(paste("The value of BSS is:", round(bss, 4)))

print(paste("The value of WSS is:", round(wss, 4)))

bss\_ratio <- bss / tss

cat("BSS/TSS Ratio:", bss\_ratio, "\n")



The BSS/TSS ratio represents the proportion of the total sum of squares (TSS) that is explained by the between-cluster sum of squares (BSS). In this case, the BSS/TSS ratio is approximately 0.2069, indicating that around 20.69% of the total variance in the dataset is accounted for by the clustering. A higher BSS/TSS ratio suggests better separation and distinctiveness among the clusters.

The BSS (255.7847) represents the sum of the squared distances between the cluster centers and the overall mean of the transformed dataset. It quantifies the separation between the clusters.

The WSS (980.6366) represents the sum of the squared distances between the data points and their respective cluster centers. It measures the compactness or cohesion within each cluster.

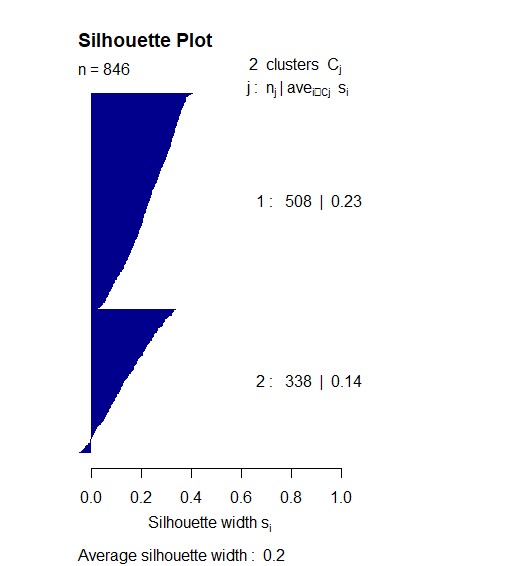
# ANS 2 part h

silhouette\_scores <- silhouette(kmeans\_model$cluster, dist(transformed\_data))

plot(silhouette\_scores, main = "Silhouette Plot",border=NA,col="darkblue")

avg\_sil\_width <- mean(silhouette\_scores[, "sil\_width"])

print(paste("The average silhouette width is:", round(avg\_sil\_width, 4)))





Based on the average silhouette width of 0.1957, it indicates a moderate level of cluster quality. The average silhouette width ranges from -1 to 1, with values closer to 1 indicating well-separated clusters. A value of 0.1957 suggests that, on average, the data points are relatively close to the neighboring clusters and not strongly associated with their own cluster.

# ANS 2 part i

library(fpc)

library(ggplot2)

# Initialize empty vectors to store the index values and k values

index\_values <- c()

k\_values <- c()

# Iterate over different values of k and calculate the Calinski-Harabasz Index

for (k in 2:10) {

kmeans\_result <- kmeans(transformed\_data, centers = k)

diss\_matrix <- dist(transformed\_data)

ch\_index <- cluster.stats(diss\_matrix, kmeans\_result$cluster)$ch

index\_values <- c(index\_values, ch\_index)

k\_values <- c(k\_values, k)

}

# Create a data frame of k values and index values

index\_df <- data.frame(k = k\_values, index = index\_values)

# Create a line plot to illustrate the Calinski-Harabasz Index

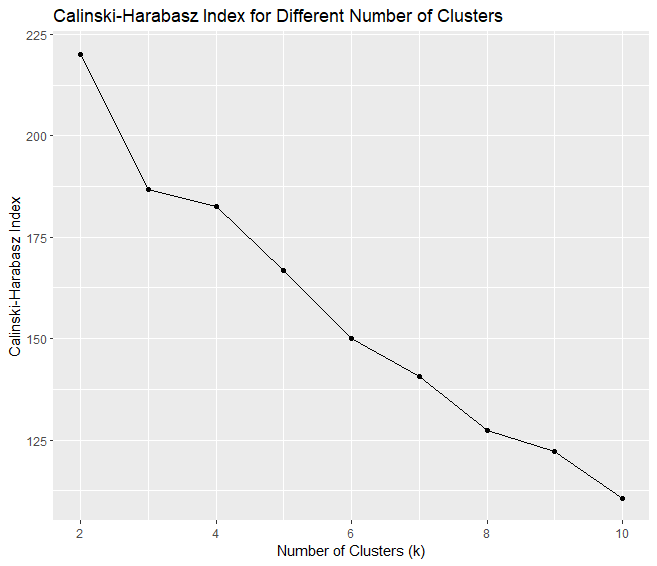
ggplot(index\_df, aes(x = k, y = index)) +

geom\_line() +

geom\_point() +

labs(x = "Number of Clusters (k)", y = "Calinski-Harabasz Index") +

ggtitle("Calinski-Harabasz Index for Different Number of Clusters")



From the plot, we can observe that as the number of clusters (k) increases, the Calinski-Harabasz Index generally decreases. This suggests that a larger number of clusters leads to a higher within-cluster dispersion relative to between-cluster dispersion. The highest Calinski-Harabasz Index value is observed at k = 2, around 223. This indicates that dividing the data into two clusters provides a better separation and compactness compared to other values of k. Additionally, the plot reveals that the lowest Calinski-Harabasz Index value of approximately 112 is observed when k=10. This suggests that dividing the data into 10 clusters results in less distinct and more scattered clusters, leading to a poorer clustering solution. Therefore as confirmed by the plot, it was reasonable to choose k=2 as the optimal number of clusters, as it corresponds to the highest Calinski-Harabasz Index value.