Technical Document

2024-10-04

Loading Packages & Data

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(cluster)  # For silhouette calculation
library(factoextra)  # For clustering viz
```

```
## Loading required package: ggplot2

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

```
# Load Data
cust_df <- read.csv("C:\\Users\\auros\\OneDrive\\Documents\\Wholesale_customers.csv")
head(cust_df)
```

```
##   Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicatessen
## 1       2      3 12669 9656    7561    214             2674         1338
## 2       2      3  7057 9810    9568   1762             3293         1776
## 3       2      3  6353 8808    7684   2405             3516         7844
## 4       1      3 13265 1196    4221   6404              507         1788
## 5       2      3 22615 5410    7198   3915             1777         5185
## 6       2      3  9413 8259    5126    666             1795         1451
```

Data Preprocessing

Null Treatment

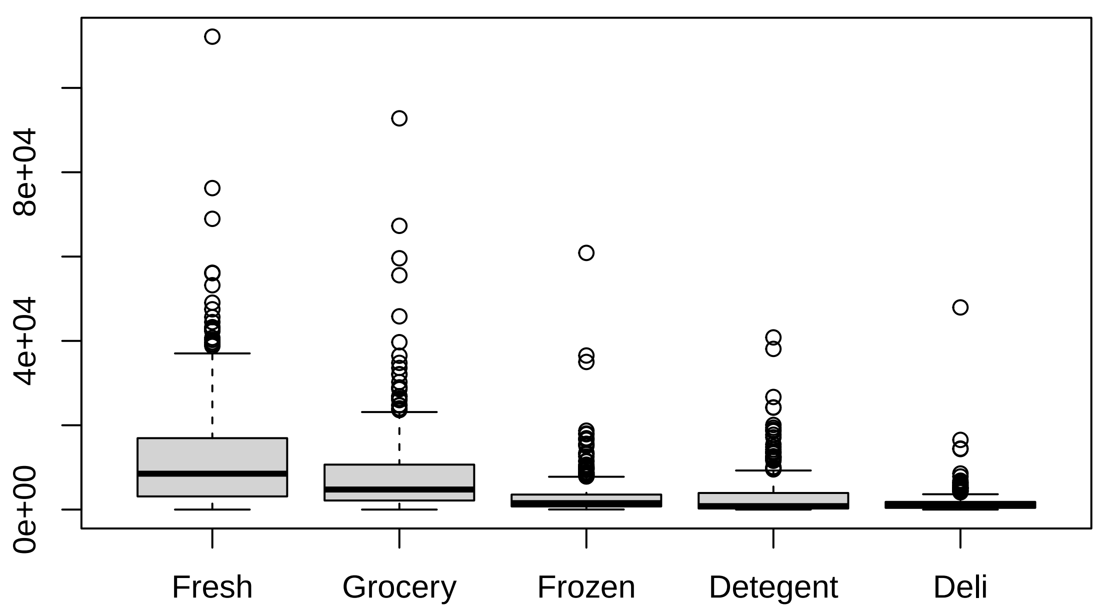
```r
#Checking total no. of nulls
sum(is.na(cust_df))
```

```
## [1] 0
```

```r
#Null treatment not required
```

Identifying & Understanding Outliers

```r
boxplot(cust_df$Fresh,cust_df$Grocery,cust_df$Frozen,cust_df$Detergents_Paper,cust_df$Delicatessen,names
```




```r
# Function to identify outliers based on IQR
identify_outliers <- function(column) {
  Q1 <- quantile(column, 0.25)
  Q3 <- quantile(column, 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 2 * IQR
  upper_bound <- Q3 + 2 * IQR
  return(column < lower_bound | column > upper_bound)
}

# Create a logical matrix indicating outliers
```

```r
outliers_matrix <- sapply(cust_df, identify_outliers)

# Count the number of outliers in each row
outlier_counts <- rowSums(outliers_matrix)
print(('Removing rows where outliers exist in atleast :'))
```

```
## [1] "Removing rows where outliers exist in atleast :"
```

```r
for (i in 1:7){
  cleaned_df <- cust_df[(outlier_counts<i), ]
  print(paste0(i," or more columns"))
  print(nrow(cleaned_df))}
```

```
## [1] "1 or more columns"
## [1] 364
## [1] "2 or more columns"
## [1] 411
## [1] "3 or more columns"
## [1] 429
## [1] "4 or more columns"
## [1] 436
## [1] "5 or more columns"
## [1] 440
## [1] "6 or more columns"
## [1] 440
## [1] "7 or more columns"
## [1] 440
```

Outlier Treatment:

After undergoing multiple iterations of clustering and analyzing the results, we decided on only excluding the rows having atleast 2 or more outliers

The outlier data is separated to be treated as a cluster of it's own named as "Elite Spenders"

```r
# Creating dataframe without outliers
cleaned_df <- cust_df[outlier_counts < 2, ]
#Creating a separate dataframe for the outliers
cust_df_outlier <- setdiff(cust_df, cleaned_df)
```

```r
cleaned_df %>% group_by(Channel,Region) %>% summarise(count_x=n(),fresh_x=mean(Fresh),
                                                       Milk_x=mean(Milk),
                                                       Frozen_x=mean(Frozen),
                                                       grocery_x=mean(Grocery),
                                                       Detergents_Paper_x=mean(Detergents_Paper),
                                                       Delicatessen_x=mean(Delicatessen)
) %>% arrange(Channel)
```

```
## `summarise()` has grouped output by 'Channel'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 6 x 9
## # Groups:   Channel [2]
##   Channel Region count_x fresh_x Milk_x Frozen_x grocery_x Detergents_Paper_x
##     <int>  <int>   <int>   <dbl>  <dbl>    <dbl>     <dbl>              <dbl>
## 1       1      1      58  13023.  3531.    3006.     3859.               953.
## 2       1      2      27  10870.  1768.    3703.     4054.               453.
## 3       1      3     204  12530.  3081.    3238.     3651.               769.
## 4       2      1      15   4854.  8404.    2324.    14849.              6450.
## 5       2      2      17   7070.  8504.    1648.    13250.              6429.
## 6       2      3      90   8999.  7773.    1423.    12428.              5218.
## # i 1 more variable: Delicatessen_x <dbl>
```

Data Normalization

Using scaled normalization instead of min-max normalization as scaling is a better method when we also want to deal with outliers in the data. It is also considered to be a prefered method of normalization for K-means clustering

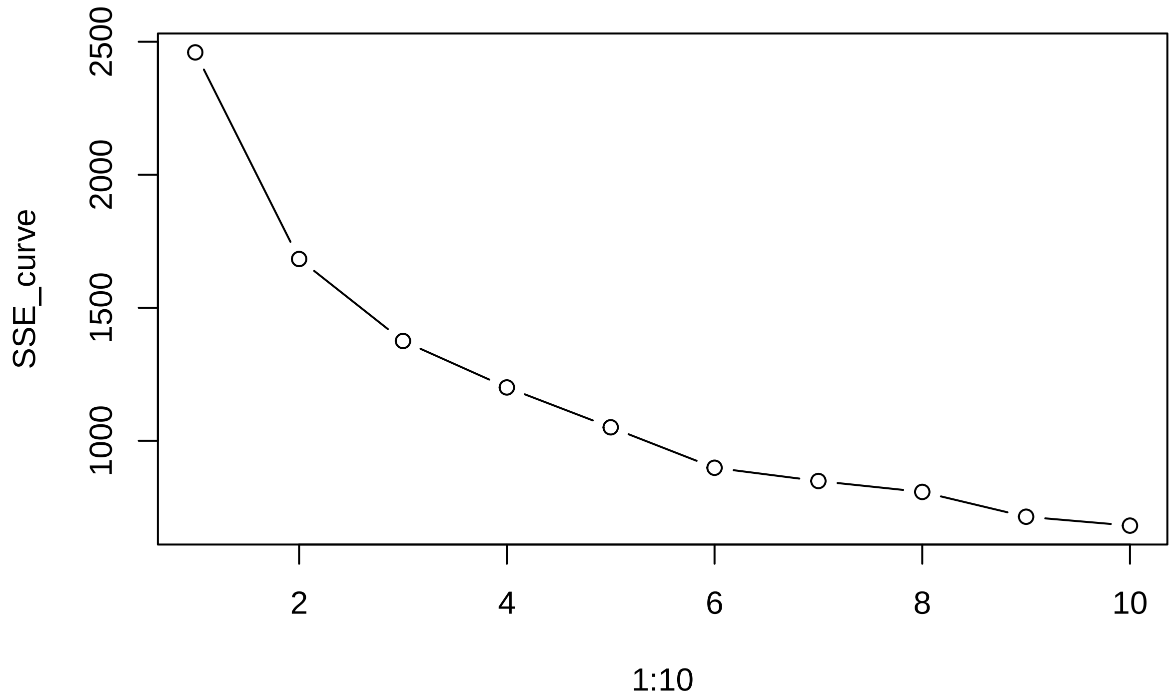
```r
# Removed columns stored in separate variables
Channel <- cleaned_df$Channel  # Save the first removed column
Region <- cleaned_df$Region  # Save the second removed column

cleaned_df <- cleaned_df[3:8]

# Normalize the data
cleaned_df_scaled <- scale(cleaned_df)
head(cleaned_df_scaled)
```

```
##        Fresh       Milk    Grocery      Frozen Detergents_Paper Delicatessen
## 1  0.1347994  1.3407180  0.1917428 -0.72368198        0.1640892   0.05900312
## 2 -0.3841024  1.3807948  0.5333221 -0.28010154        0.3739440   0.36084809
## 3 -0.4491963  1.1200352  0.2126766 -0.09584946        0.4495459   4.54257255
## 4  0.1899073 -0.8609053 -0.3767051  1.05006668       -0.5705719   0.36911782
## 5  1.0544355  0.2357425  0.1299624  0.33684206       -0.1400137   2.71013924
## 6 -0.1662598  0.9771639 -0.2226796 -0.59416108       -0.1339113   0.13687636
```

Performing Clustering:

Elbow plot to identify optimum k-value for K-means Cluster

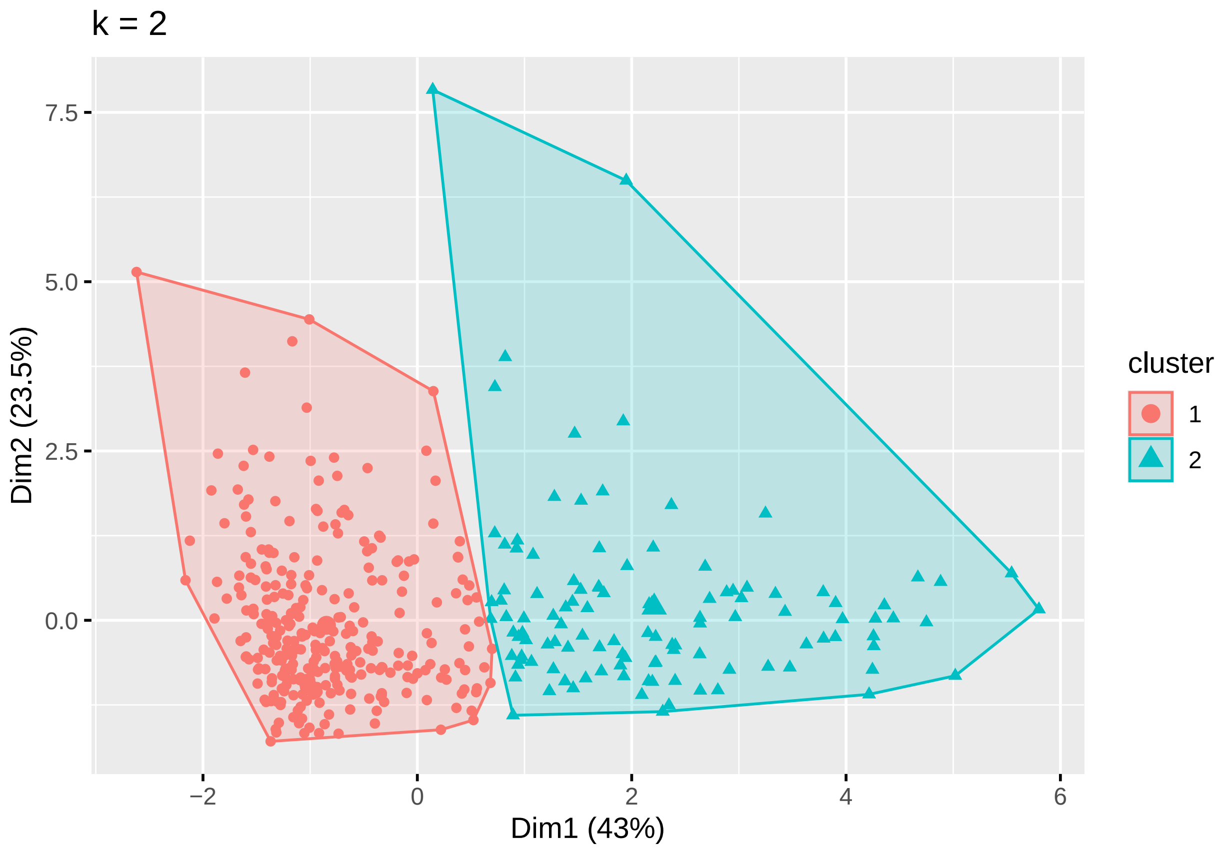
```r
SSE_curve <- c()
for (n in 1:10) {
  kcluster = kmeans(cleaned_df_scaled, n)
  sse = kcluster$tot.withinss
  SSE_curve[n] = sse}
# plot SSE against number of clusters
plot(1:10, SSE_curve, type = "b")
```



Tried multiple iterations for elbow plot for all cases of outlier treatment i.e. For our final selection of data (i.e. when outliers existing in atleast 2 or more columns are excluded), k=2 and k=3 was shortlisted for further analysis

Building K-MEANS Cluster : FOR K=2 (challenger model)

```
kmeans_result <- kmeans(cleaned_df_scaled, centers = 2)
# Add cluster assignment to the cleaned dataframe
cleaned_df_k2 = cleaned_df
cleaned_df_k2$Cluster <- kmeans_result$cluster
cleaned_df_k2 <- cbind(cleaned_df_k2, Channel = Channel, Region = Region)
## Visualizing clustering results using fviz_cluster
fviz_cluster(kmeans_result, geom = "point", data = cleaned_df_scaled[,]) + ggtitle("k = 2")
```



Understanding silhouette coefficient for K=2:

It evaluates how well data points are clustered by comparing the intra-cluster cohesion with inter similarity

Reason behind finalizing K=2: It not only gives us good average silhouette coefficient but also showcases decent scores across each cluster.

```r
# Calculate silhouette coefficient
distance_matrix = dist(cleaned_df_scaled, method = "euclidean")
sc = silhouette(kmeans_result$cluster, dist = distance_matrix)
summary(sc)
```

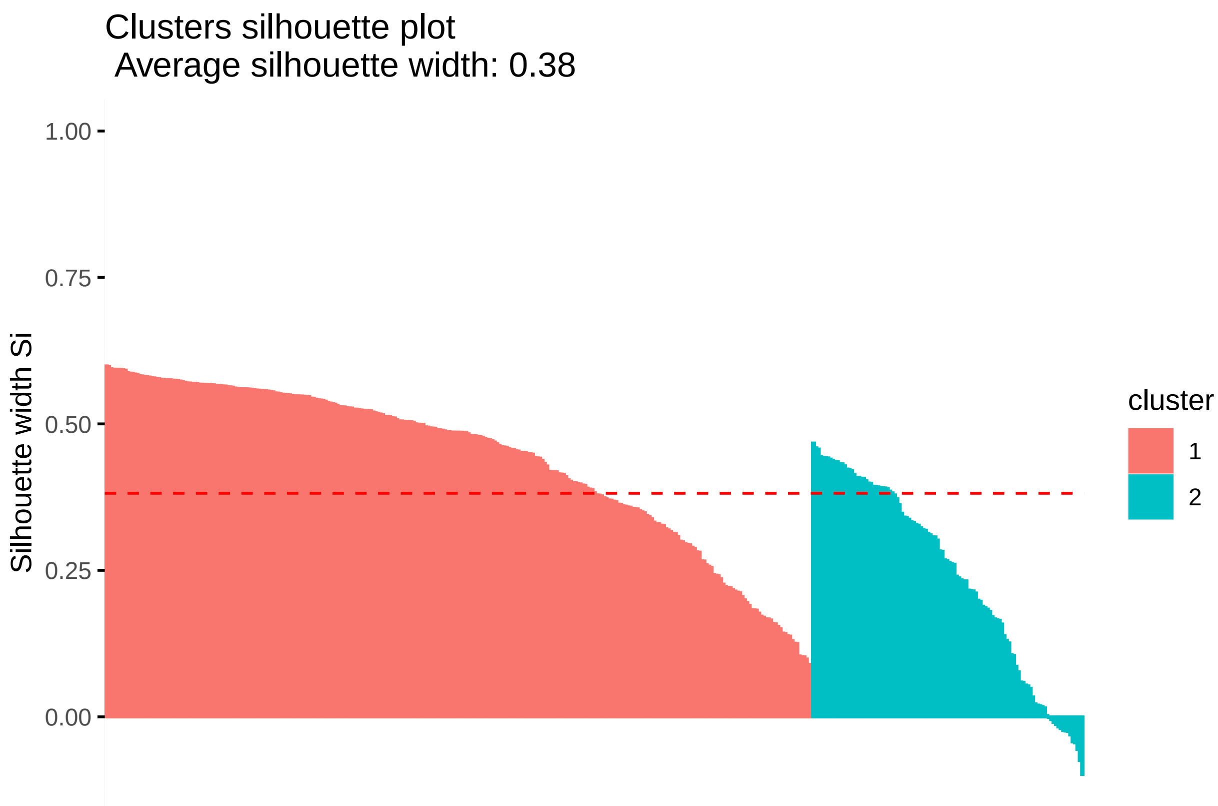
```
## Silhouette of 411 units in 2 clusters from silhouette.default(x = kmeans_result$cluster, dist = dista
##  Cluster sizes and average silhouette widths:
##       297       114
## 0.4368018 0.2377089
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.09863  0.26617  0.41906  0.38158  0.52729  0.59908
```

```r
average_silhouette <- mean(sc[, 3])
# Print the average silhouette coefficient
print(paste("Average Silhouette Coefficient:", average_silhouette))
```

```
## [1] "Average Silhouette Coefficient: 0.381578936565213"
```

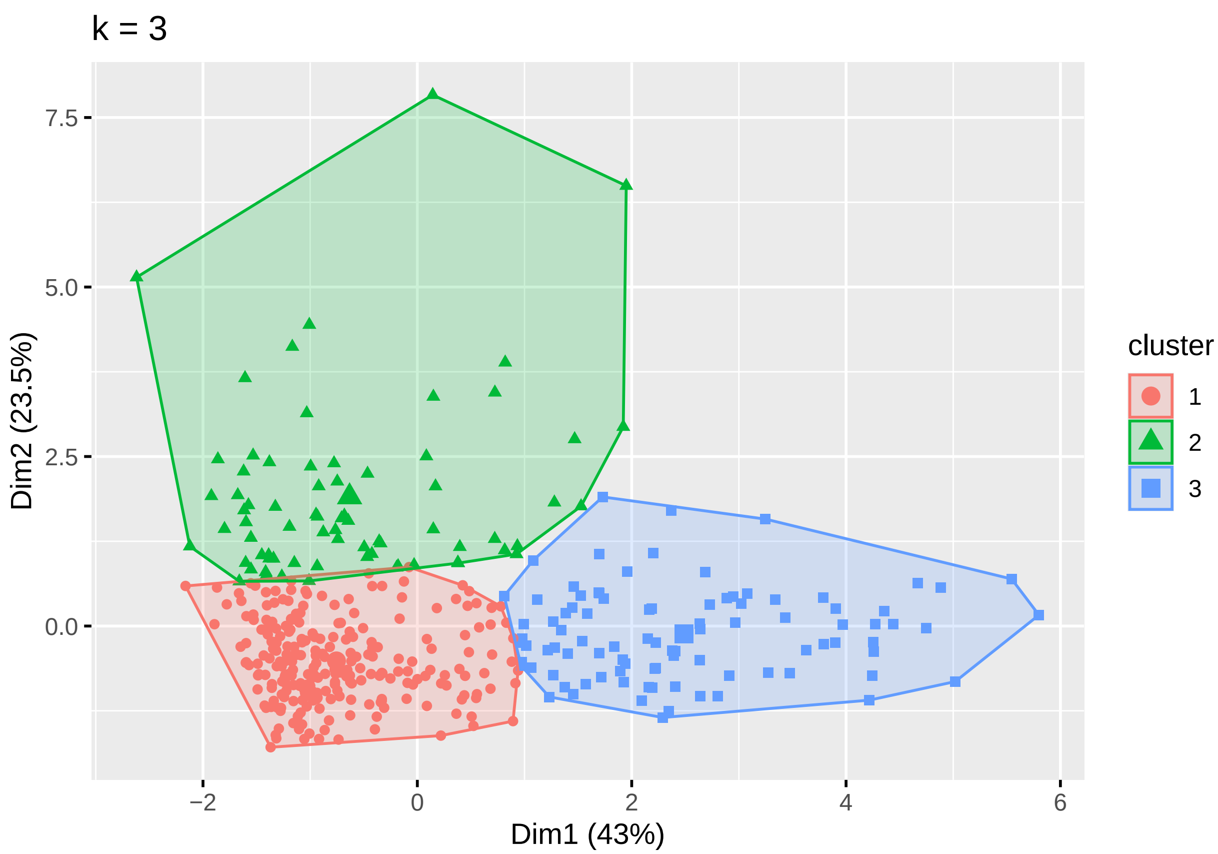
```
fviz_silhouette(sc)
```

```
##   cluster size ave.sil.width
## 1       1  297          0.44
## 2       2  114          0.24
```



Building K-MEANS Cluster : FOR K=3 (Champion model)

```
kmeans_result <- kmeans(cleaned_df_scaled, centers = 3)
# Add cluster assignment to the cleaned dataframe
cleaned_df$Cluster <- kmeans_result$cluster
cleaned_df <- cbind(cleaned_df, Channel = Channel, Region = Region)
## Visualizing clustering results using fviz_cluster
fviz_cluster(kmeans_result, geom = "point", data = cleaned_df_scaled[,]) + ggtitle("k = 3")
```



Understanding silhouette coefficient for K=3.

It evaluates how well data points are clustered by comparing the intra-cluster cohesion with inter similarity

Reason behind finalizing K=3: It not only gives us good average silhouette coefficient but also showcases decent scores across 2 clusters and a positive score for the 3rd clusters which reduces any chance of misclassification

```r
# Calculate silhouette coefficient
distance_matrix = dist(cleaned_df_scaled, method = "euclidean")
sc = silhouette(kmeans_result$cluster, dist = distance_matrix)
summary(sc)
```

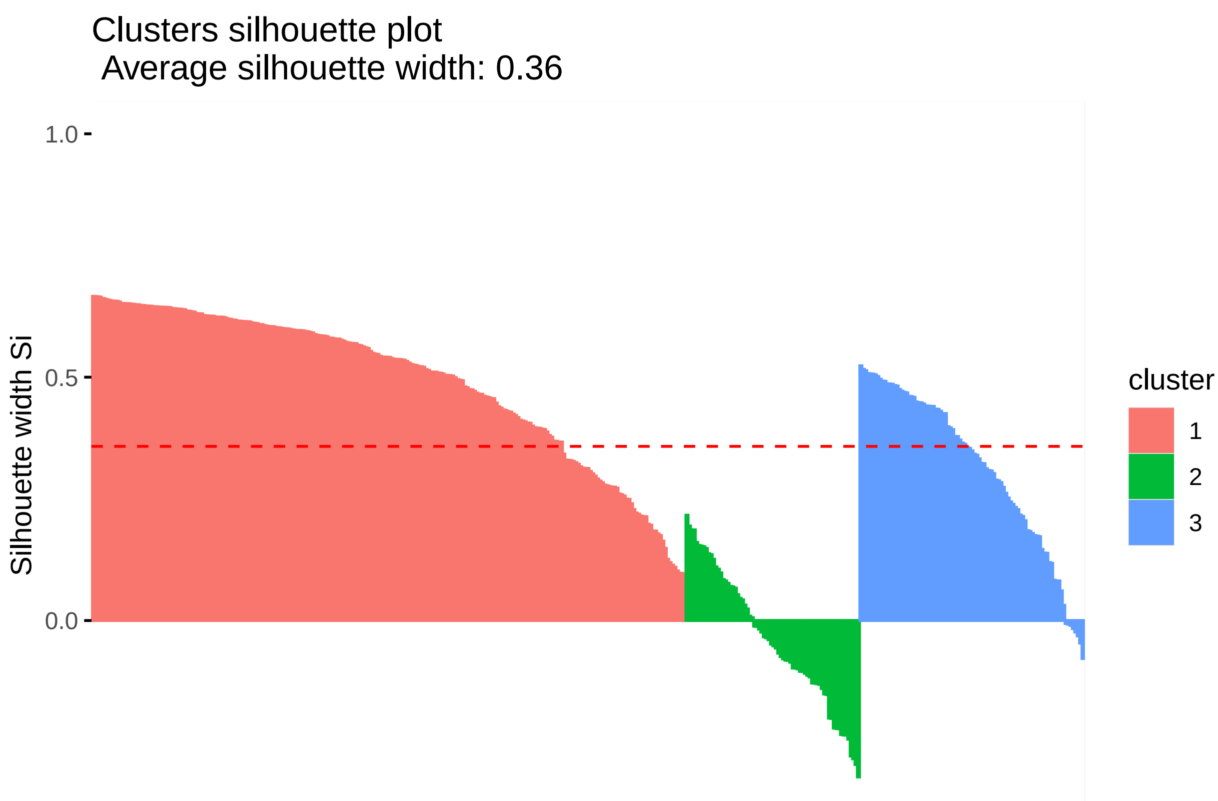
```
## Silhouette of 411 units in 3 clusters from silhouette.default(x = kmeans_result$cluster, dist = dista
##  Cluster sizes and average silhouette widths:
##         246          72          93
##  0.49161808 -0.03989826  0.31198371
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3214  0.1837  0.4302  0.3579  0.5772  0.6661
```

```r
average_silhouette <- mean(sc[, 3])
# Print the average silhouette coefficient
print(paste("Average Silhouette Coefficient:", average_silhouette))
```

```
## [1] "Average Silhouette Coefficient: 0.357858533658904"
```

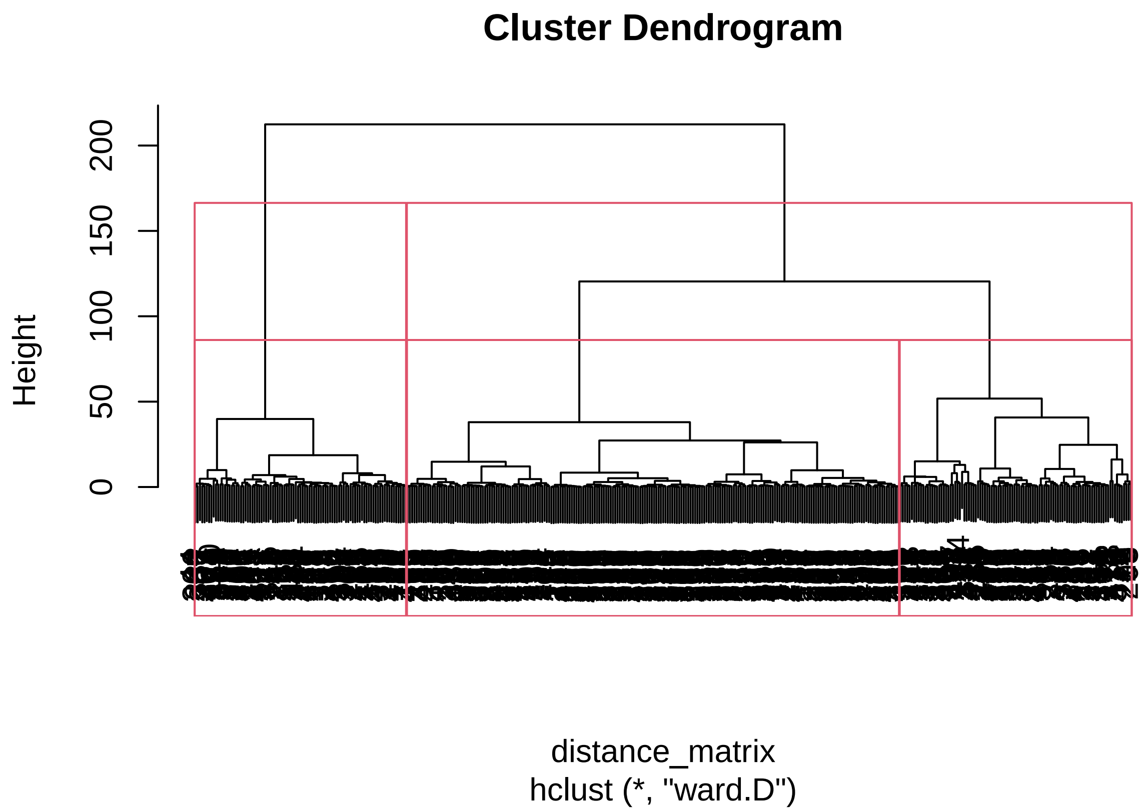
```r
fviz_silhouette(sc)
```

```
##   cluster size ave.sil.width
## 1       1  246          0.49
## 2       2   72         -0.04
## 3       3   93          0.31
```



Building Hierarchical Cluster for K=3 & K=2

```r
# Hierarchical Clustering
hierarchical = hclust(distance_matrix, method = "ward.D")
plot(hierarchical)
rect.hclust(hierarchical, k = 3)
rect.hclust(hierarchical, k = 2)
```



The dendogram also showcases a possible cluster of either K=3 or K=2 which supports our decision of considering k=3/4 from Kmeans clustering

Hence our next step would be to analyze the clusters from both K=3 and K=2 kmeans model and gain insights based on customers spending habits across each clusters and their relevant channel/region subgroups

EXPORTING RESULTS

```
#Printing output for K=3 to excel for further analysis
#write.xlsx(cleaned_df, 'C:\\Users\\auros\\OneDrive\\Documents\\Outliers_3k.xlsx')
#Printing output for K=2 to excel for further analysis
#write.xlsx(cleaned_df_k2, 'C:\\Users\\auros\\OneDrive\\Documents\\Outliers_2k.xlsx')
```