Group Assignment 1

2024-10-06

```
### Setup Libraries
library(stats)
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)
library(ggplot2)
library(scales)
library(cluster)
library(factoextra)
```

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

```
options(warn = -1)
```

Reading Data

```
# import dataset
customer = read.csv("Wholesale customers data.csv")

# view data
head(customer)
```

```
##   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
```

```r
# summary of data for the six categories
summary(customer[,3:8])
```

```
##      Fresh             Milk          Grocery          Frozen
##  Min.   :     3   Min.   :   55   Min.   :    3   Min.   :   25.0
##  1st Qu.:  3128   1st Qu.: 1533   1st Qu.: 2153   1st Qu.:  742.2
##  Median :  8504   Median : 3627   Median : 4756   Median : 1526.0
##  Mean   : 12000   Mean   : 5796   Mean   : 7951   Mean   : 3071.9
##  3rd Qu.: 16934   3rd Qu.: 7190   3rd Qu.:10656   3rd Qu.: 3554.2
##  Max.   :112151   Max.   :73498   Max.   :92780   Max.   :60869.0
##  Detergents_Paper   Delicatessen
##  Min.   :    3.0   Min.   :    3.0
##  1st Qu.:  256.8   1st Qu.:  408.2
##  Median :  816.5   Median :  965.5
##  Mean   : 2881.5   Mean   : 1524.9
##  3rd Qu.: 3922.0   3rd Qu.: 1820.2
##  Max.   :40827.0   Max.   :47943.0
```

Data Exploration and Processing

Normalization

We want to normalzie the data to first check for outliers as it will help in

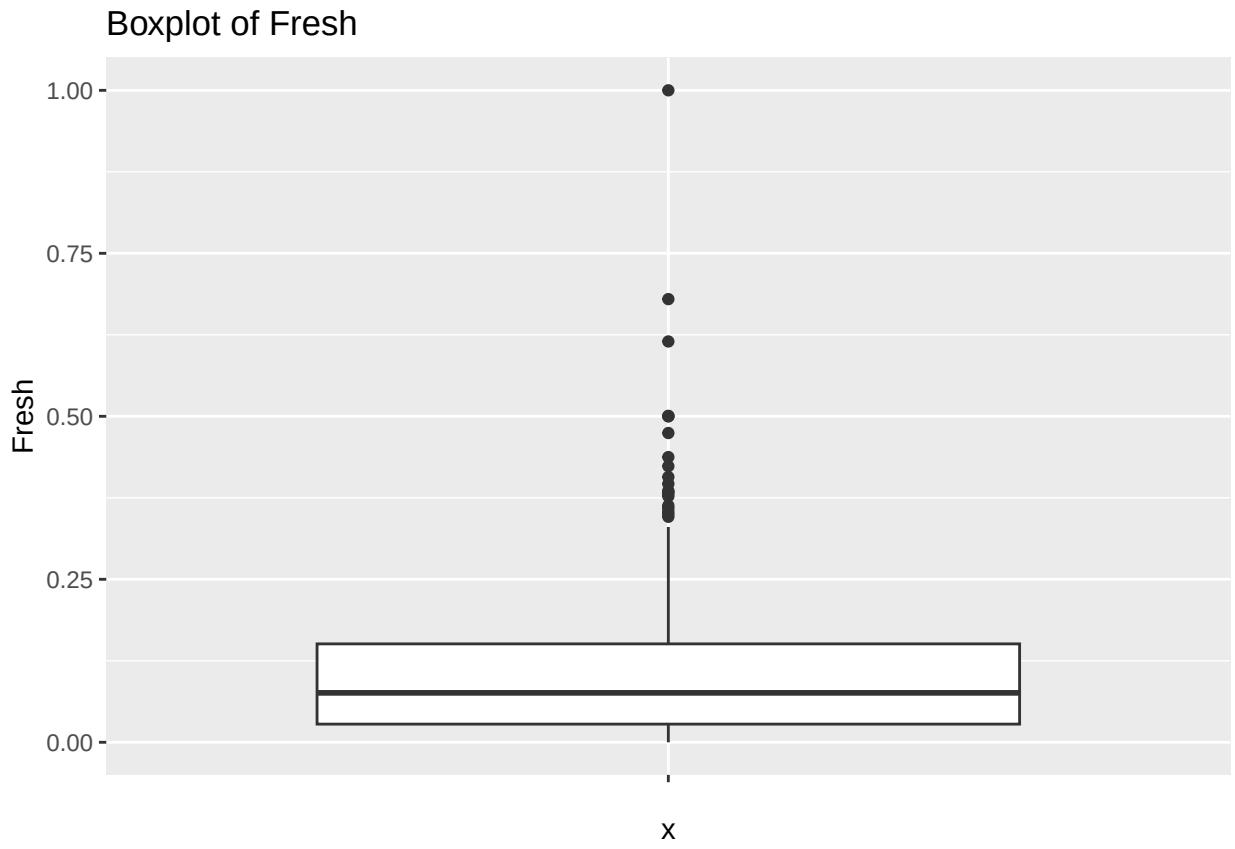
getting better visuals in box plots.

```r
# declare normalize function using min-max method
normalize = function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# normalize iris dataset from columns 1 to 4
customer_normalized = customer %>% mutate_at(c(3:8), normalize)
summary(customer_normalized[,3:8])
```

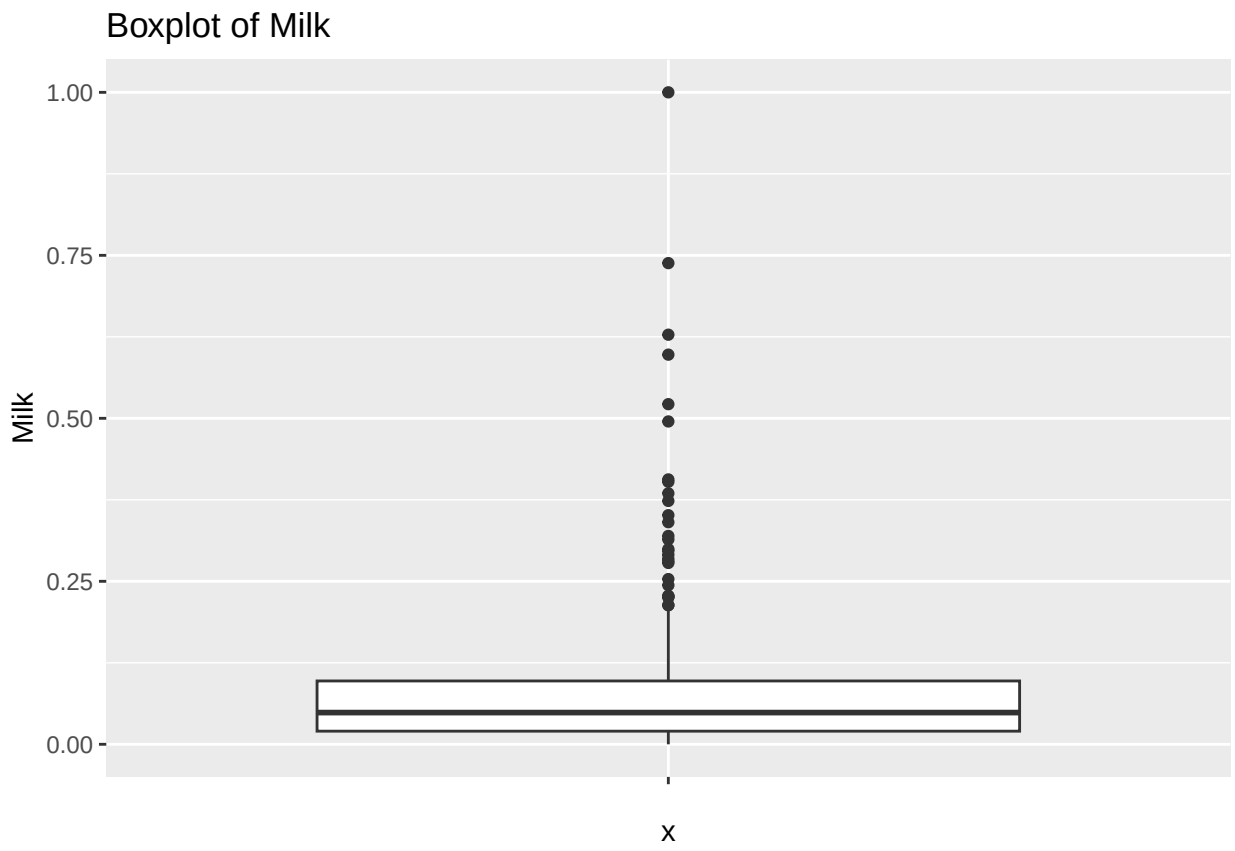
```
##      Fresh              Milk            Grocery           Frozen
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.02786   1st Qu.:0.02012   1st Qu.:0.02317   1st Qu.:0.01179
##  Median :0.07580   Median :0.04864   Median :0.05122   Median :0.02467
##  Mean   :0.10698   Mean   :0.07817   Mean   :0.08567   Mean   :0.05008
##  3rd Qu.:0.15097   3rd Qu.:0.09715   3rd Qu.:0.11482   3rd Qu.:0.05800
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##  Detergents_Paper    Delicatessen
##  Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.006216   1st Qu.:0.008453
##  Median :0.019927   Median :0.020077
##  Mean   :0.070510   Mean   :0.031745
##  3rd Qu.:0.095997   3rd Qu.:0.037907
##  Max.   :1.000000   Max.   :1.000000
```

Boxplots

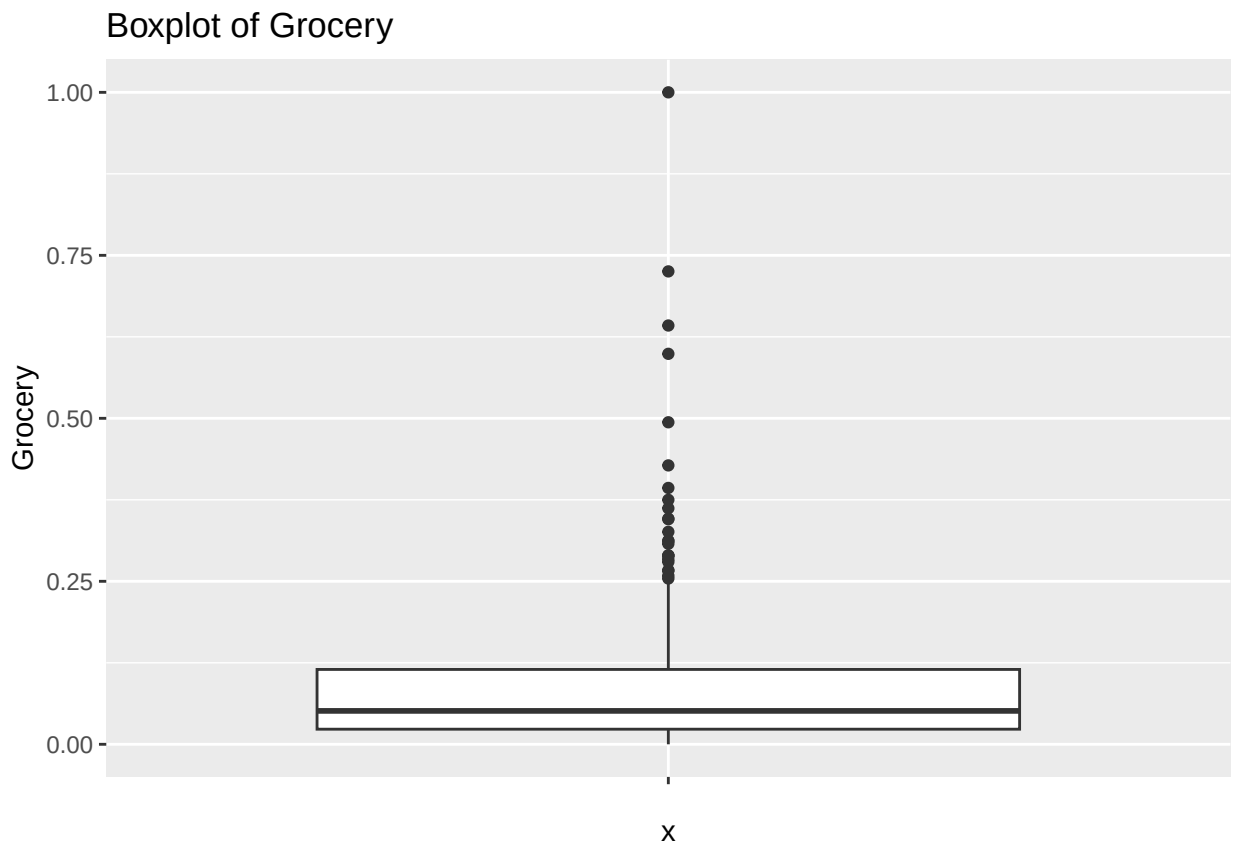
```
# boxplots for each feature
ggplot(data = customer_normalized, aes(x = "", y = Fresh)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 1)) +
  ggtitle("Boxplot of Fresh")
```



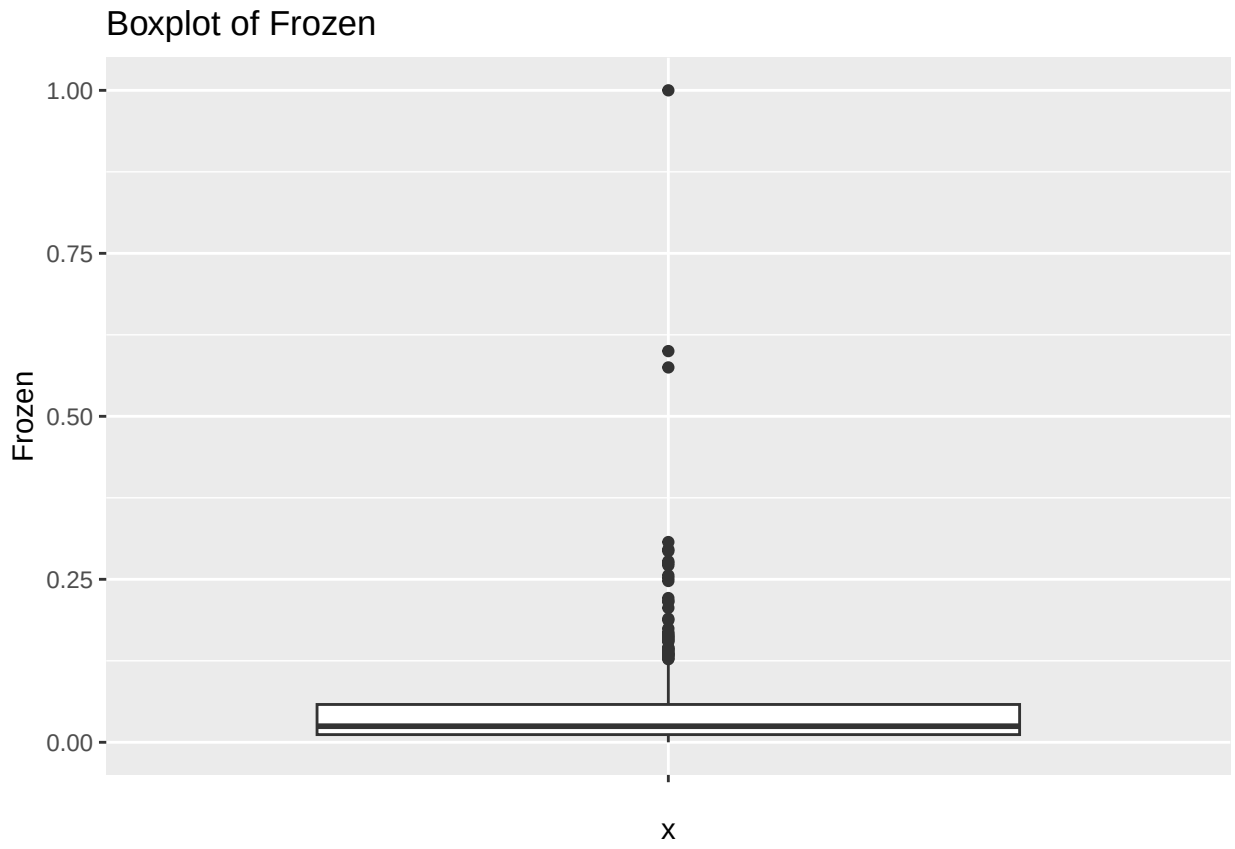

```
ggplot(data = customer_normalized, aes(x = "", y = Milk)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 1)) +
  ggtitle("Boxplot of Milk")
```



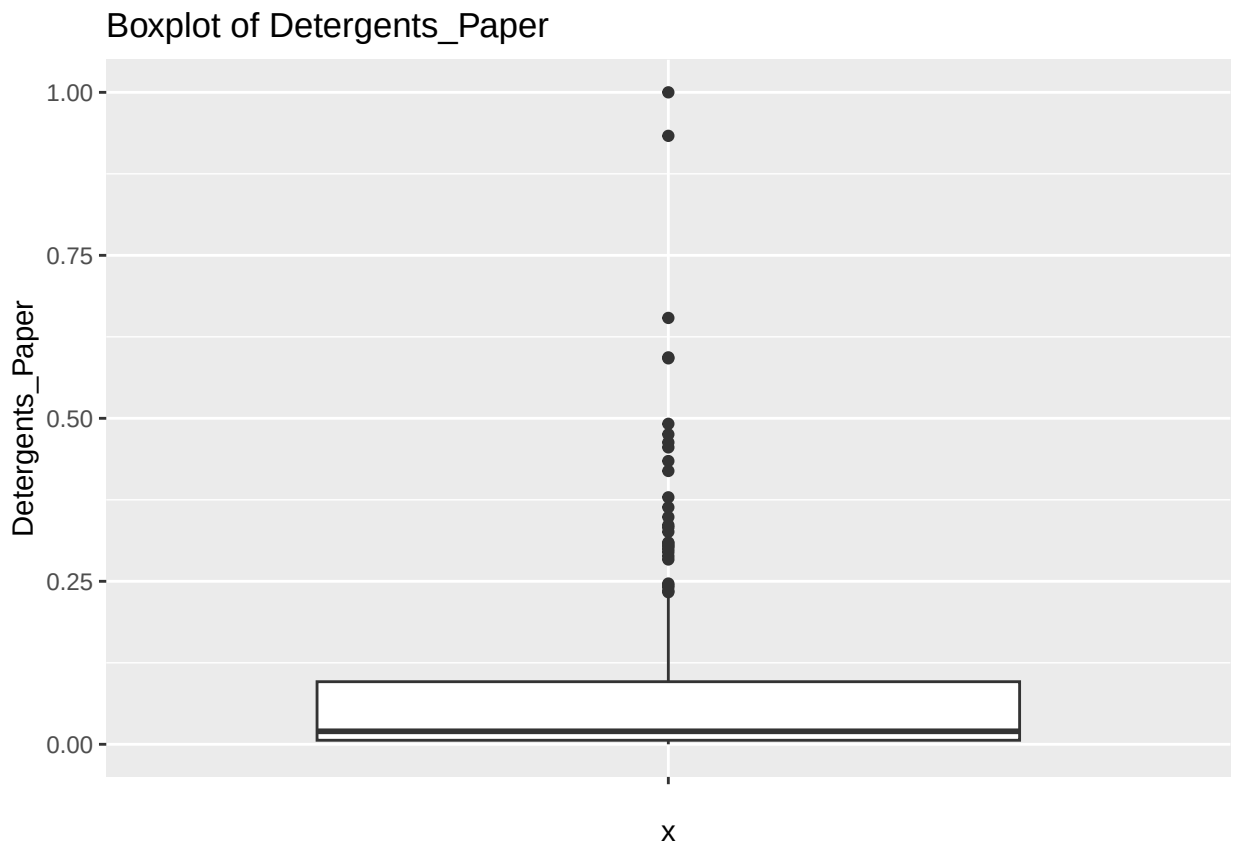

```
ggplot(data = customer_normalized, aes(x = "", y = Grocery)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 1)) +
  ggtitle("Boxplot of Grocery")
```



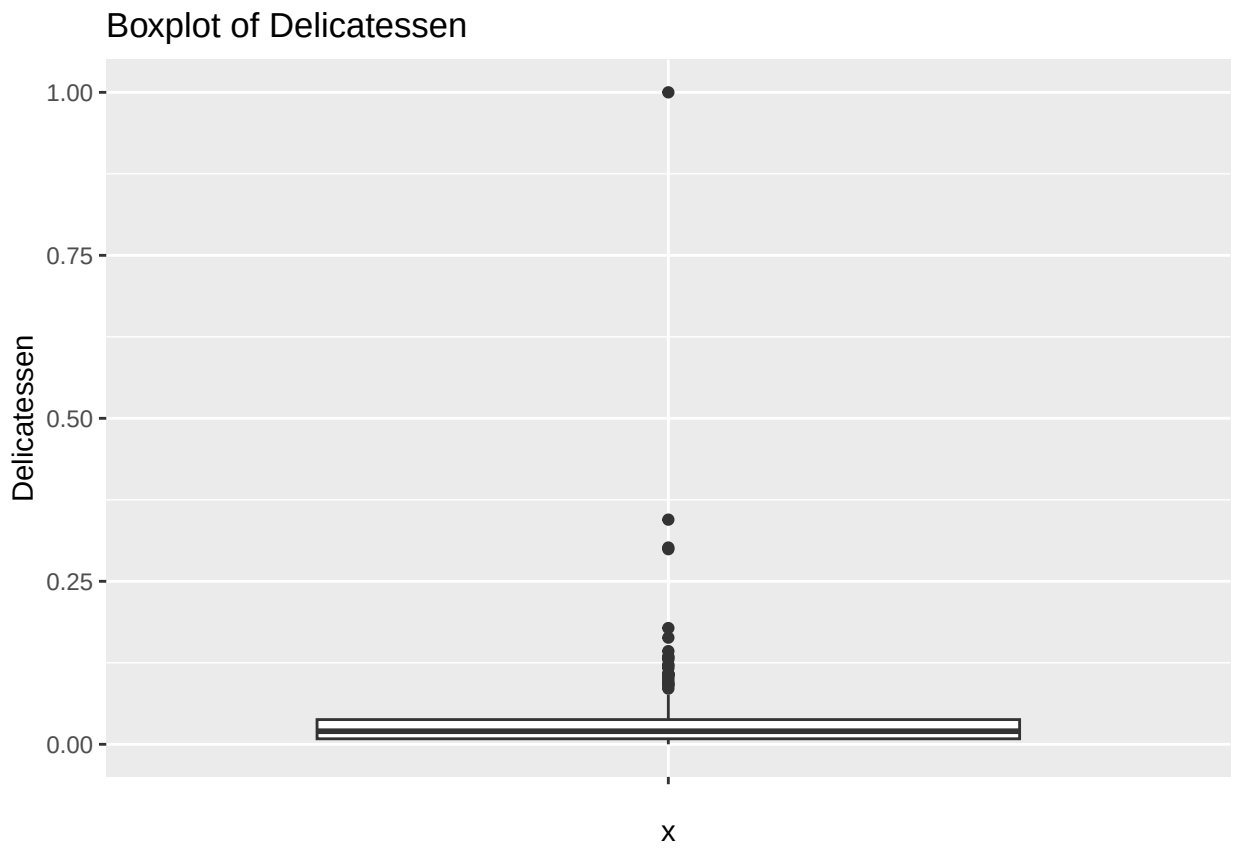

```
ggplot(data = customer_normalized, aes(x = "", y = Frozen)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 1)) +
  ggtitle("Boxplot of Frozen")
```




```
ggplot(data = customer_normalized, aes(x = "", y = Detergents_Paper)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 1)) +
  ggtitle("Boxplot of Detergents_Paper")
```




```
ggplot(data = customer_normalized, aes(x = "", y = Delicatessen)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 1)) +
  ggtitle("Boxplot of Delicatessen")
```



We can certainly see that there are outliers in the data(in each feature column) that need to treated/removed.

Outlier Flagging

We want to flag outliers on a row level, i.e., for each client and check the distribution of outliers for each feature colum among all the clients.

```
# create list of outlier data points for each feature column
fresh_outliers <- boxplot(customer$Fresh, plot=FALSE)$out
milk_outliers <- boxplot(customer$Milk, plot=FALSE)$out
grocery_outliers <- boxplot(customer$Grocery, plot=FALSE)$out
frozen_outliers <- boxplot(customer$Frozen, plot=FALSE)$out
detergent_outliers <- boxplot(customer$Detergents_Paper, plot=FALSE)$out
delicate_outliers <- boxplot(customer$Delicatessen, plot=FALSE)$out

# create an outlier flag based on the lists created above
customer$Fresh_Outlier_Flag <- ifelse(customer$Fresh %in% fresh_outliers, 1, 0)
customer$Milk_Outlier_Flag <- ifelse(customer$Milk %in% milk_outliers, 1, 0)
customer$Grocery_Outlier_Flag <- ifelse(customer$Grocery %in% grocery_outliers
                                        , 1, 0)
customer$Frozen_Outlier_Flag <- ifelse(customer$Frozen %in% frozen_outliers
```

```
                                        , 1, 0)
customer$Detergent_Outlier_Flag <- ifelse(customer$Detergents_Paper %in%
                                            detergent_outliers, 1, 0)
customer$Delicate_Outlier_Flag <- ifelse(customer$Delicatessen %in%
                                           delicate_outliers, 1, 0)

# create a total outlier column containing total outliers per client across
# all feature columns
customer$Total_Outliers <- customer$Fresh_Outlier_Flag +
  customer$Milk_Outlier_Flag + customer$Grocery_Outlier_Flag +
  customer$Frozen_Outlier_Flag + customer$Detergent_Outlier_Flag +
  customer$Delicate_Outlier_Flag

head(customer[,9:15])
```

```
##   Fresh_Outlier_Flag Milk_Outlier_Flag Grocery_Outlier_Flag Frozen_Outlier_Flag
## 1                  0                 0                    0                   0
## 2                  0                 0                    0                   0
## 3                  0                 0                    0                   0
## 4                  0                 0                    0                   0
## 5                  0                 0                    0                   0
## 6                  0                 0                    0                   0
##   Detergent_Outlier_Flag Delicate_Outlier_Flag Total_Outliers
## 1                      0                    0              0
## 2                      0                    0              0
## 3                      0                    1              1
## 4                      0                    0              0
## 5                      0                    1              1
## 6                      0                    0              0
```

Outlier Summary

```
# get outlier percentages for each feature column
cat("Total Rows: ",nrow(customer), "\n")
```

```
## Total Rows:  440
```

```
cat("Total Outliers: ",nrow(customer %>% filter(Total_Outliers>=1)), "\n\n")
```

```
## Total Outliers:  108
```

```
cat("Outliers Percentage\n")
```

```
## Outliers Percentage
```

```
cat("--------------------\n")
```

```
## --------------------
```

```r
# Total
cat("Total: ", percent(nrow(customer %>% filter(Total_Outliers>=1))
                       / nrow(customer)), "\n")
```

```
## Total:  25%
```

```r
# Fresh
cat("Fresh: ", percent(nrow(customer %>% filter(Fresh_Outlier_Flag==1))
                       / nrow(customer)), "\n")
```

```
## Fresh:  5%
```

```r
# Milk
cat("Milk: ", percent(nrow(customer %>% filter(Milk_Outlier_Flag==1))
                      / nrow(customer)), "\n")
```

```
## Milk:  6%
```

```r
# Grocery
cat("Grocery: ", percent(nrow(customer %>% filter(Grocery_Outlier_Flag==1))
                         / nrow(customer)), "\n")
```

```
## Grocery:  5%
```

```r
# Frozen
cat("Frozen: ", percent(nrow(customer %>%
                               filter(Frozen_Outlier_Flag==1))
                        / nrow(customer)), "\n")
```

```
## Frozen:  10%
```

```r
# Detergent
cat("Detergent Paper: ", percent(nrow(customer %>%
                                        filter(Detergent_Outlier_Flag==1))
                                 / nrow(customer)), "\n")
```

```
## Detergent Paper:  7%
```

```r
# Delicatessen
cat("Delicatessen: ", percent(nrow(customer %>%
                                     filter(Delicate_Outlier_Flag==1))
                              / nrow(customer)), "\n")
```

```
## Delicatessen:  6%
```

```r
# get how many clients have 0,1,2.. outliers
customer %>% group_by(Total_Outliers) %>% summarise(Count = n())
```

```
## # A tibble: 6 x 2
##   Total_Outliers Count
##            <dbl> <int>
## 1              0   332
## 2              1    67
## 3              2    24
## 4              3    13
## 5              4     3
## 6              5     1
```

We can see that out of 440 clients, 332 don't have the outliers present in the dataset. Our clustering approach will start with taking data with no outliers and then we will start including clients with 1/2/3.. outlier per row to see if results drastically change. We will perform a different exploration exercise on the remaining outlier data to check for meaningful insights.

Clustering

Remove Outliers

```
# outlier dataframe
customer_out = customer %>% filter(Total_Outliers >= 2)

# removing outliers and storin in new dataframe
customer_final = customer %>% filter(Total_Outliers < 2)
head(customer_final)
```

```
##   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
##   Fresh_Outlier_Flag Milk_Outlier_Flag Grocery_Outlier_Flag Frozen_Outlier_Flag
## 1                  0                 0                    0                   0
## 2                  0                 0                    0                   0
## 3                  0                 0                    0                   0
## 4                  0                 0                    0                   0
## 5                  0                 0                    0                   0
## 6                  0                 0                    0                   0
##   Detergent_Outlier_Flag Delicate_Outlier_Flag Total_Outliers
## 1                      0                    0              0
## 2                      0                    0              0
## 3                      0                    1              1
```

```
## 4                    0                    0              0
## 5                    0                    1              1
## 6                    0                    0              0
```

We tried running clustering models using k = 1,2,3,4. We observed that k = 2 is gving us the best results. Hence, the above piece of code takes Total Outliers < 2 to get the final dataset on which clustering will be performed.

Normalization

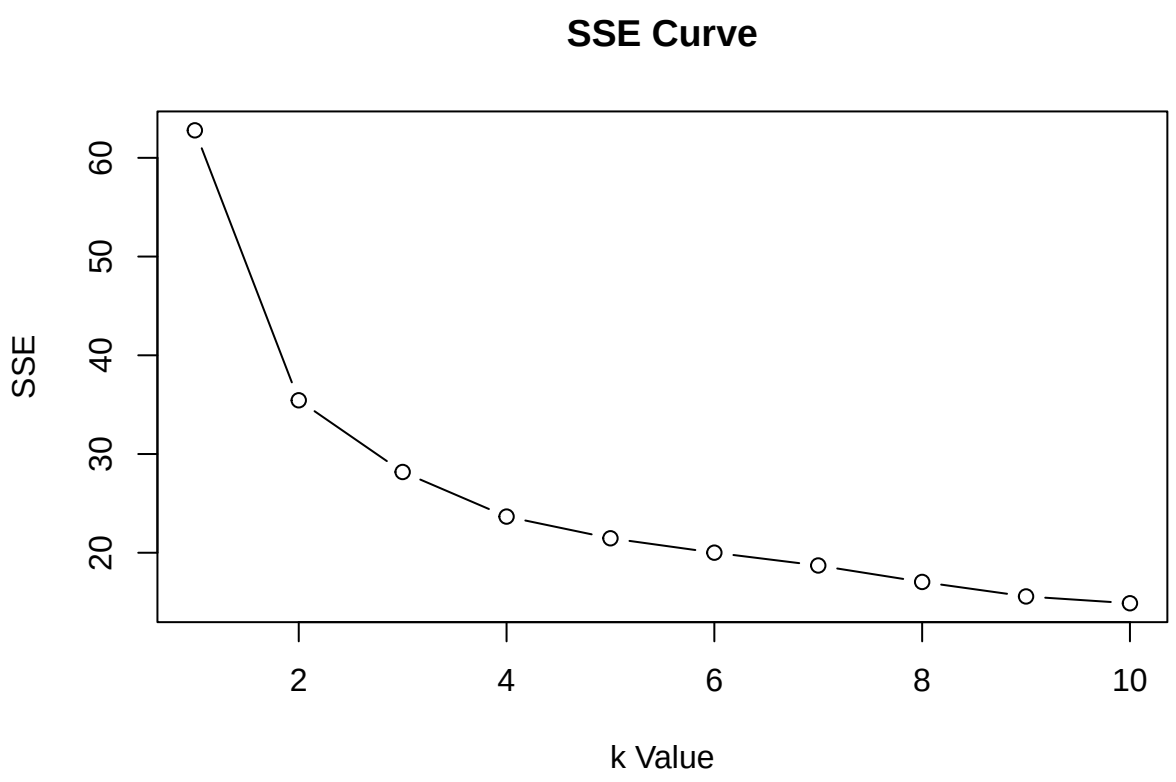
Normalizing data for running clustering models using min-max method

```
# normalize
customer_final_norm = customer_final %>% mutate_at(c(3:8), normalize)

# create distance matrix with euclidean distance
distance_matrix = dist(customer_final_norm[,3:8], method = "euclidean")
```

K-Means Clustering

Check for k value

```
# calculating SSE
SSE_curve <- c()
for (k in 1:10) {
  kcluster = kmeans(customer_final_norm[,3:8], k)
  sse = kcluster$tot.withinss
  SSE_curve[k] = sse}

# plot SSE against number of clusters
plot(1:10, SSE_curve, type = "b", main = "SSE Curve", xlab = "k Value", ylab = "SSE")
```



The elbow plot indicates we can take 2/3/4 clusters as the

sum of squared errors start to get stagnant after k = 5

Run Model

```
# run k-means clustering
cat("For k = 2: \n")
```

```
## For k = 2:
```

```
kcluster = kmeans(customer_final_norm[,3:8], centers = 2)
sc = silhouette(kcluster$cluster, dist = distance_matrix)
summary(sc)
```

```
## Silhouette of 399 units in 2 clusters from silhouette.default(x = kcluster$cluster, dist = distance_m
##  Cluster sizes and average silhouette widths:
##       105       294
## 0.3236288 0.4942762
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.02347  0.34134  0.49653  0.44937  0.59949  0.65648
```

```r
cat("\n For k = 3: \n")
```

```
##
##  For k = 3:
```

```r
kcluster = kmeans(customer_final_norm[,3:8], centers = 3)
sc = silhouette(kcluster$cluster, dist = distance_matrix)
summary(sc)
```

```
## Silhouette of 399 units in 3 clusters from silhouette.default(x = kcluster$cluster, dist = distance_m
##  Cluster sizes and average silhouette widths:
##       102        91       206
## 0.2949904 0.1438929 0.4183334
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1260  0.1919  0.3531  0.3242  0.4866  0.5877
```

```r
cat("\n For k = 4: \n")
```

```
##
##  For k = 4:
```

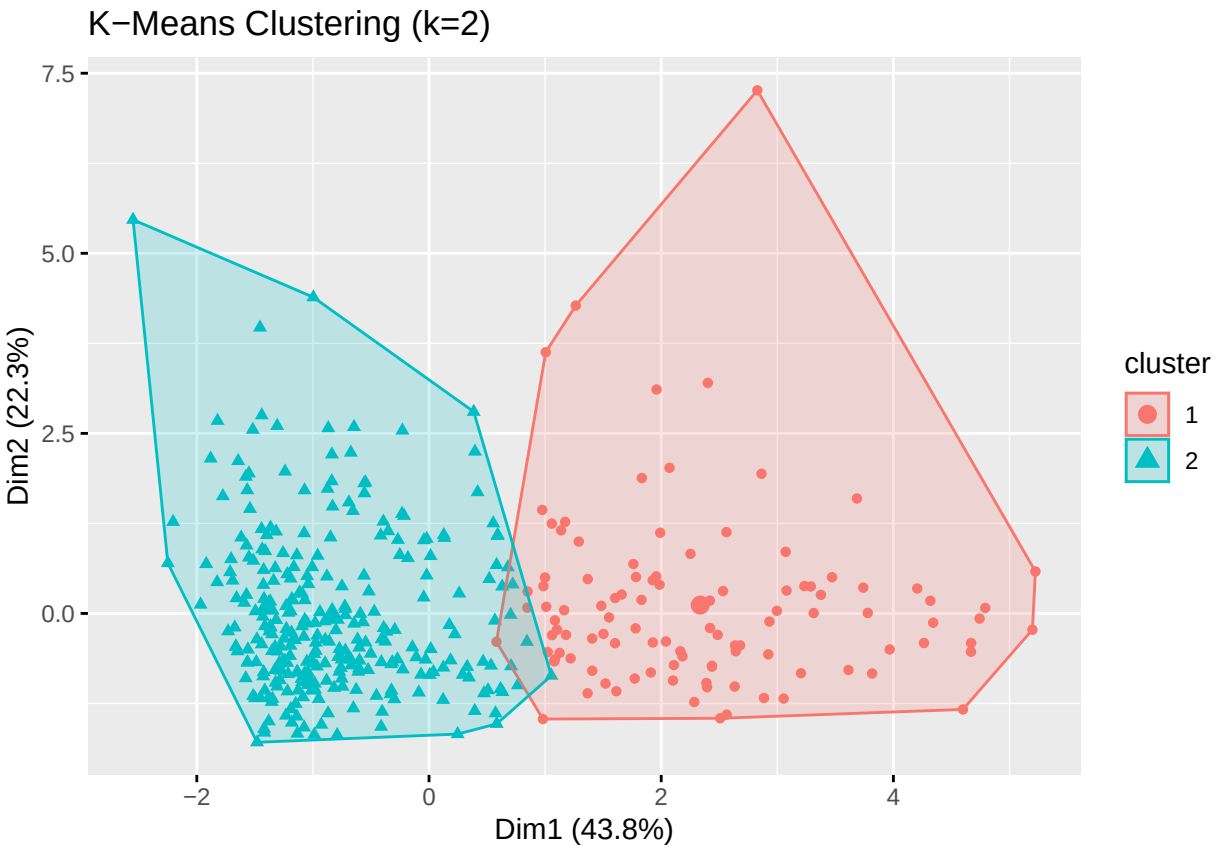
```r
kcluster = kmeans(customer_final_norm[,3:8], centers = 4)
sc = silhouette(kcluster$cluster, dist = distance_matrix)
summary(sc)
```

```
## Silhouette of 399 units in 4 clusters from silhouette.default(x = kcluster$cluster, dist = distance_m
##  Cluster sizes and average silhouette widths:
##         96        181         24         98
## 0.09274386 0.36386961 0.28493515 0.29982111
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1211  0.1425  0.2842  0.2782  0.4419  0.5399
```

We are observing that k=2 has the best silhouette score. Therefore, we will

be going ahead with k=2 for K-Means

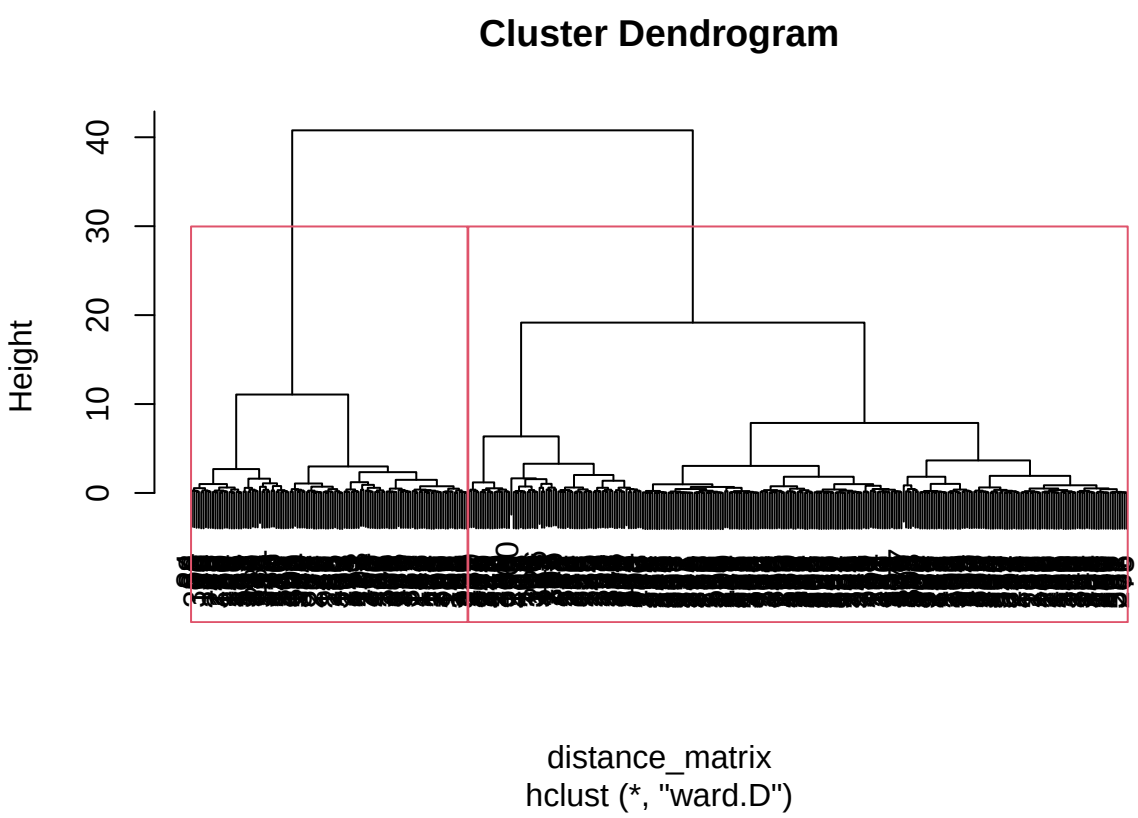
```r
kcluster = kmeans(customer_final_norm[,3:8], centers = 2)

# get cluster labels into original dataframe
customer_final$cluster = kcluster$cluster

fviz_cluster(kcluster, geom = "point", data = customer_final_norm[,3:8]) +
  ggtitle("K-Means Clustering (k=2)")
```



Hierarchical Clustering

Check for k value

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



The dendogram indicates we can take 2/3 clusters.

Run Model

```
cat("For k = 2: \n")
```

```
## For k = 2:
```

```
customer_final_norm$cluster = cutree(hierarchical, k = 2)
sc = silhouette(customer_final_norm$cluster, dist = distance_matrix)
summary(sc)
```

```
## Silhouette of 399 units in 2 clusters from silhouette.default(x = customer_final_norm$cluster, dist =
##  Cluster sizes and average silhouette widths:
##       118       281
## 0.2801186 0.4164518
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3253  0.3178  0.4431  0.3761  0.5230  0.5901
```

```
cat("For k = 3: \n")
```

```
## For k = 3:
```

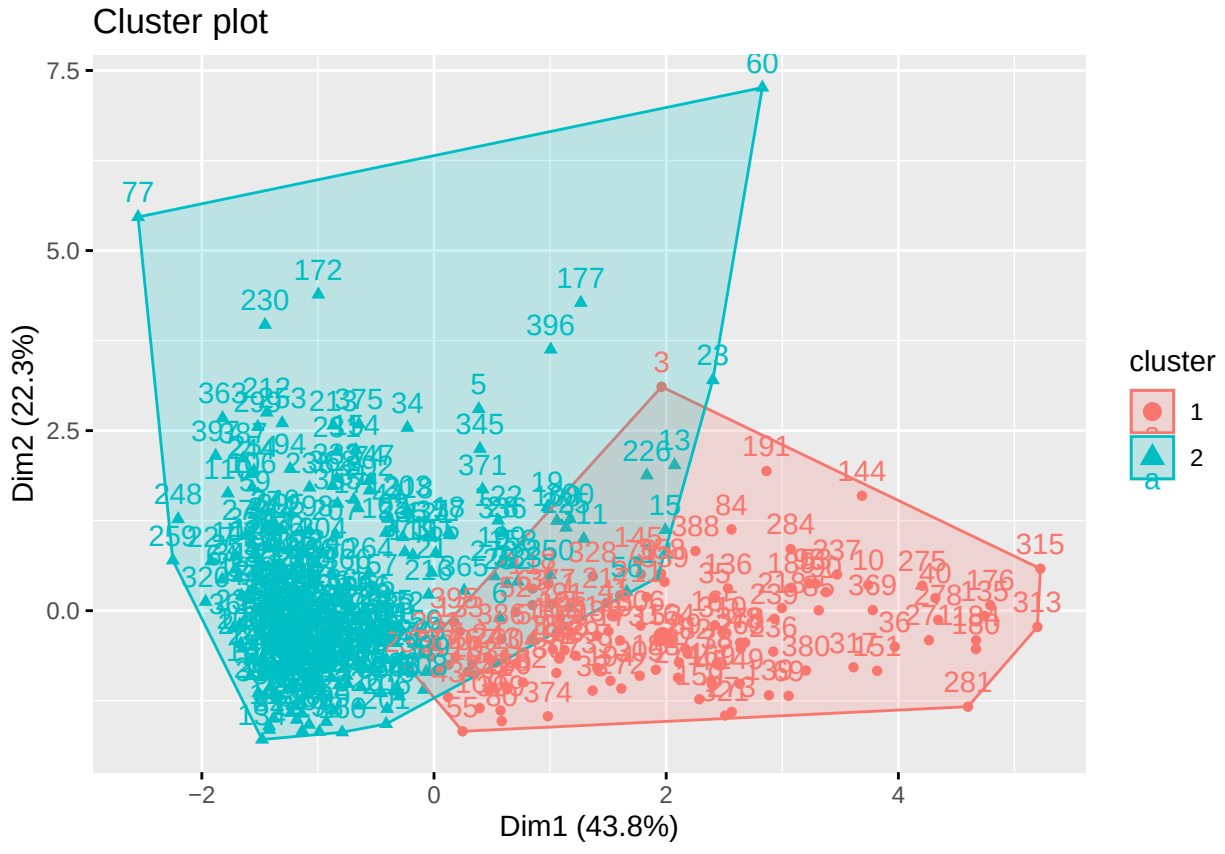
```
customer_final_norm$cluster = cutree(hierarchical, k = 3)
sc = silhouette(customer_final_norm$cluster, dist = distance_matrix)
summary(sc)
```

```
## Silhouette of 399 units in 3 clusters from silhouette.default(x = customer_final_norm$cluster, dist =
##  Cluster sizes and average silhouette widths:
##        118        207         74
## 0.23689748 0.52078491 0.04048801
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.4536  0.1866  0.4059  0.3478  0.5694  0.6649
```

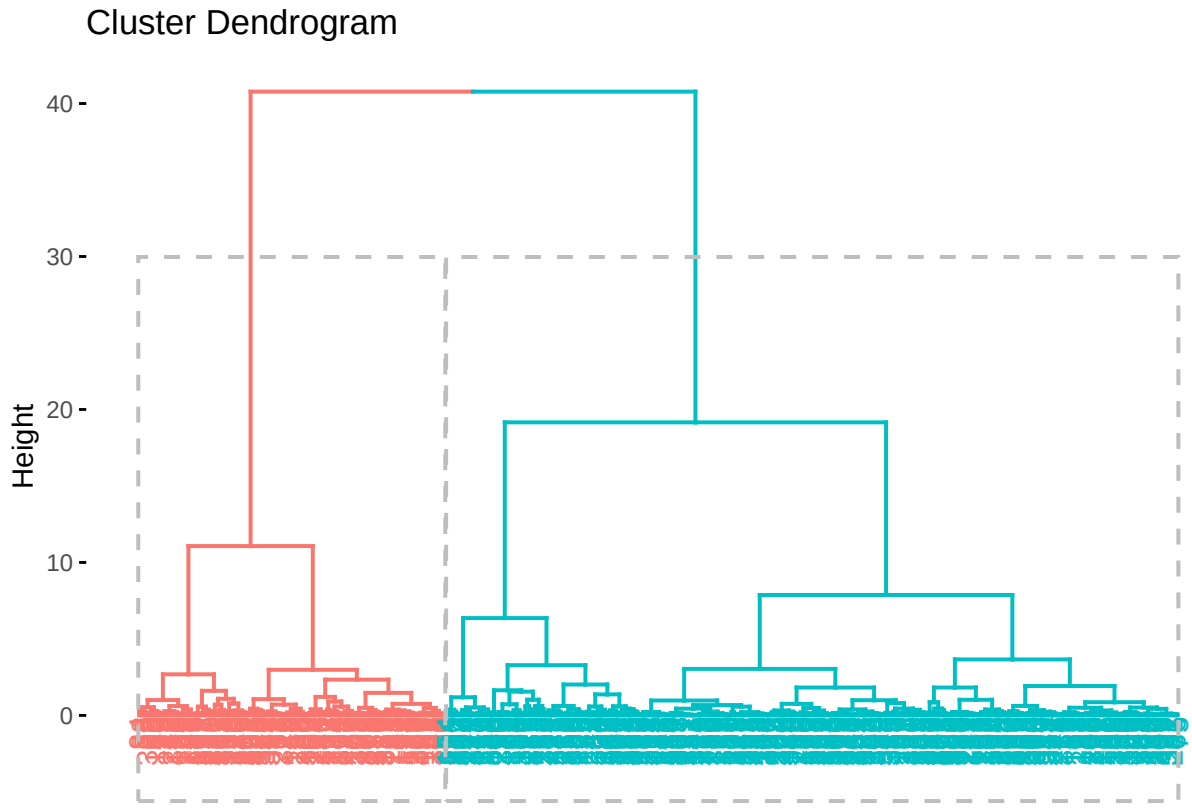
We are observing that k=2 has the best silhouette score. Therefore, we will

be going ahead with k=2 for Hierarchical

```
clusters = cutree(hierarchical, k = 2)

# Visualize clusters
fviz_cluster(list(data = customer_final_norm[,3:8], geom = "point", cluster = clusters))
```



```r
# Visualize the dendrogram with rectangles around clusters
fviz_dend(hierarchical, k = 2, rect = TRUE)
```



Model Evaluation

We compared the silhouette score from best models from K-Means(0.449) and Hierarchical(0.376) clustering techniques and came to a conclusion that we should use K-Means with k=2 as the clusters formed are easily distinguishable.

Result Interpretation

To intrepret the results, we are taking an average of each feature column across each level of data - Cluster, Region and Channel to get the patterns around spendings of each cluster. Using those insights, we will generate recommendations for the XYZ company.

```
customer_cluster_summary <-
  customer_final %>%
  group_by(cluster,Region,Channel) %>%
  summarise(
    Count = n(),
    Avg_Fresh = mean(Fresh),
    Avg_Milk = mean(Milk),
    Avg_Grocery = mean(Grocery),
    Avg_Frozen = mean(Frozen),
    Avg_Detergents_Paper = mean(Detergents_Paper),
    Avg_Delicatessen = mean(Delicatessen)
  )
```

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

```
customer_cluster_summary
```

```
## # A tibble: 11 x 10
## # Groups:   cluster, Region [6]
##    cluster Region Channel Count Avg_Fresh Avg_Milk Avg_Grocery Avg_Frozen
##      <int>  <int>   <int> <int>     <dbl>    <dbl>       <dbl>      <dbl>
##  1       1      1       1     7    11833.   10009.      10724.      2023.
##  2       1      1       2    10     3115.    9248.      17184       1267.
##  3       1      2       2    11     5441.    9997.      13847.      1082.
##  4       1      3       1    10     8160     6638.      13192.      3400.
##  5       1      3       2    67     7563.    8402.      13214.      1388.
##  6       2      1       1    51    13186.    2642.       2917.      3141.
##  7       2      1       2     4     9296.    4794.       6545.      4660.
##  8       2      2       1    27    10870.    1768.       4054.      3703.
```

```
##  9       2      2       2     4    12237.    3067.       5331.      3564.
## 10       2      3       1   189    12165.    2727.       3072.      3074.
## 11       2      3       2    19    15466.    4532        6686       1569.
## # i 2 more variables: Avg_Detergents_Paper <dbl>, Avg_Delicatessen <dbl>
```

```
#write.csv(customer_cluster_summary, "customer_cluster_summary.csv")
```

We also want to check patterns for clients that came out as outliers during

our analysis.

```
customer_outlier_summary <-
  customer_out %>%
  group_by(Region,Channel) %>%
  summarise(
    Count = n(),
    Avg_Fresh = mean(Fresh),
    Avg_Milk = mean(Milk),
    Avg_Grocery = mean(Grocery),
    Avg_Frozen = mean(Frozen),
    Avg_Detergents_Paper = mean(Detergents_Paper),
    Avg_Delicatessen = mean(Delicatessen)
  )
```

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

```
customer_outlier_summary
```

```
## # A tibble: 6 x 9
## # Groups:   Region [3]
##   Region Channel Count Avg_Fresh Avg_Milk Avg_Grocery Avg_Frozen
##    <int>   <int> <int>     <dbl>    <dbl>       <dbl>      <dbl>
## 1      1       1     1     5909    23527       13699      10155
## 2      1       2     4     6317.   20614.      33618.      3802.
## 3      2       1     1    32717    16784       13626      60869
## 4      2       2     4     7427    13098       34139.       778.
## 5      3       1    12    45621.   12834.       8972.     13050
## 6      3       2    19    12195.   26524.      34883.      1898.
## # i 2 more variables: Avg_Detergents_Paper <dbl>, Avg_Delicatessen <dbl>
```

```
#write.csv(customer_outlier_summary, "customer_outliers_summary.csv")
```