K-means and Hierarchical Clustering

Objective: to identify clusters in wholesale customers data

Data set: Wholesale customers dataset

https://archive.ics.uci.edu/ml/datasets/Wholesale+customers

First, I prepared the environment, set the working directory and loaded the libraries using the following:

```
> ### K-means clustering and HCA
>
> rm(list=ls()) #Clear the environment
> setwd("YOUR_PATH") #Set working directory for the assignment
> getwd() #Check working directory
[1] "YOUR_PATH"
>
> #Apply K-means clustering and HCA to the customer's dataset
> #https://archive.ics.uci.edu/ml/datasets/Wholesale+customers
>
> #Load packages
> library(stats)
> library(fpc)
> library(cluster)
> library(NbClust)
> library(factoextra)
```

Next, I loaded the wholesale customers dataset from the previously downloaded data file (https://archive.ics.uci.edu/ml/machine-learning-databases/00292/) using the following code:

```
> ###Load data
> customers <- read.csv2("WholesaleCustomersData.csv", header = TRUE, sep = ",")
```

To make sure that the data was loaded correctly, I looked at the internal structure of the data frame:

```
> customers <- read.csv2("WholesaleCustomersData.csv", header = TRUE, sep = ",")
> str(customers) #Check structure of the df
'data.frame':      440 obs. of  8 variables:
 $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
 $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
```

```
$ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
$ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
$ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
$ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
$ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
```

The data was loaded correctly.

The dataset contains 440 observations of 8 variables that were all loaded as an int data type, even though, based on the dataset metadata, Channel and Region are categorical variables. The Channel attribute has two possible values – hotel/restaurant/café and retail store. The region attribute includes three possible levels – Lisbon, Oporto, and all other geographical areas.

Next, to familiarize myself with the data, I looked at the first few records in the data frame and at the summary statistics for all variables in the dataset:

```
> head(customers) #First few rows
  Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
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
```

```
> summary(customers) #summary stats for all variables
    Channel          Region          Fresh             Milk          Grocery          Frozen
 Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55   Min.   :    3   Min.   :   25.0
 1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533   1st Qu.: 2153   1st Qu.:  742.2
 Median :1.000   Median :3.000   Median :  8504   Median : 3627   Median : 4756   Median : 1526.0
 Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796   Mean   : 7951   Mean   : 3071.9
 3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190   3rd Qu.:10656   3rd Qu.: 3554.2
 Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498   Max.   :92780   Max.   :60869.0
 Detergents_Paper    Delicassen
 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
>
```

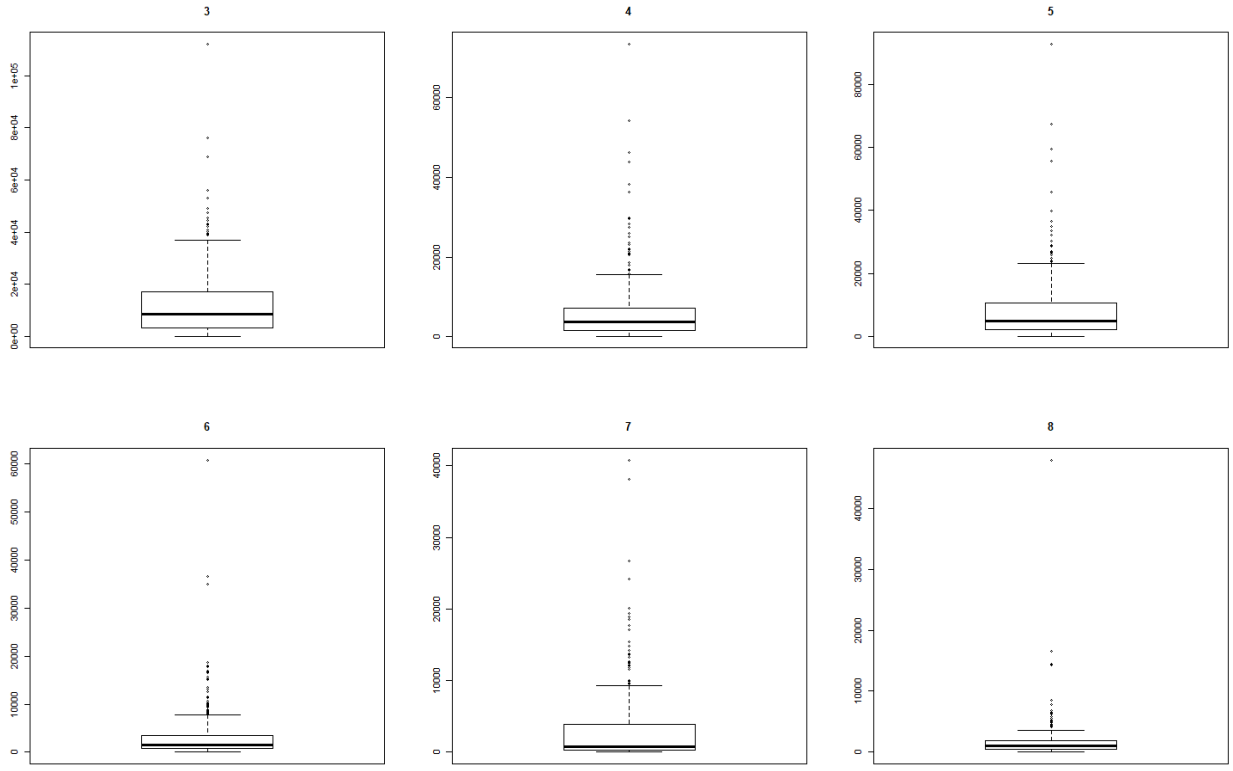
I have not noticed any missing data, but to doublecheck I used the following code, which returned zero missing values:

```
> sum(is.na(customers)) #Missing values?
[1] 0
```

Next, I looked at boxplots for all numerical variables in the dataset:

```
> #Boxplots for all numerical variables in the daataset
> par(mfrow=c(2,3))
> for (i in 3:8){
+   boxplot(customers[i], main=i)
+ }
```

The output is below:



The box plots show that none of the variables are normally distributed. They are all significantly right skewed and contain a large number of outliers – customers with higher ordering amounts in all six product groups. The plots also suggest that the majority of the customers might have very similar purchasing amounts, which might present a problem when clustering.

Next step – data preprocessing. Since clustering algorithms use distance measurements, they cannot work directly with categorical variables. So, I introduced dummy variables to encode the two categorical variables – Channel and Region.

```
> table(customers$Channel)
```

```
  1   2
298 142
```

The dataset contains records for 142 retail customers and 298 customers representing hotels, restaurants, and cafés. So, I added a new variable retail equal to 1 for all retail customers and 0 for all other customers:

```
> #Retail variable
> customers$Retail <- ifelse(customers$Channel == 2, 1, 0)
> #Check results
> table(customers$Retail)

  0   1
298 142
> str(customers)
'data.frame':      440 obs. of  9 variables:
 $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
 $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
 $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
 $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
 $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
 $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
 $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
 $ Retail          : num  1 1 1 0 1 1 1 1 0 1 ...
```

In order to encode Region, I introduced two new variables Lisbon (equal to 1 for businesses from Lisbon and 0 for all others) and Oporto (equal to 1 for customers from Oporto and 0 for customers from all other regions):

```
> #Region
> table(customers$Region)

  1   2   3
 77  47 316
>
> customers$Lisbon <- ifelse(customers$Region == 1, 1, 0)
> customers$Oporto <- ifelse(customers$Region ==2, 1, 0)
>
> #Check results
> table(customers$Lisbon)

  0   1
363  77
> table(customers$Oporto)

  0   1
```

```
393  47
> str(customers)
'data.frame':      440 obs. of  11 variables:
 $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
 $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
 $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
 $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
 $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
 $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
 $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
 $ Retail          : num  1 1 1 0 1 1 1 1 0 1 ...
 $ Lisbon          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Oporto          : num  0 0 0 0 0 0 0 0 0 0 ...
```

The next step – is data standardization, since the data frame contains highly varied purchasing amounts that might make clustering more difficult. So, I copied all numerical fields in a new data frame:

```
> cust_data <- customers[3:8]
>
> head(cust_data)
  Fresh Milk Grocery Frozen Detergents_Paper Delicassen
1 12669 9656    7561    214             2674       1338
2  7057 9810    9568   1762             3293       1776
3  6353 8808    7684   2405             3516       7844
4 13265 1196    4221   6404              507       1788
5 22615 5410    7198   3915             1777       5185
6  9413 8259    5126    666             1795       1451
> str(cust_data)
'data.frame':      440 obs. of  6 variables:
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
 $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
 $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
 $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
 $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
 $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
```

And used z-score standardization using built-in scale() function which yielded the following results:

```
> #Z-score standartization for columns with sales data
> cust_dataz <- as.data.frame(lapply(cust_data[, 1:6], scale))
> head(cust_dataz)
        Fresh        Milk     Grocery     Frozen Detergents_Paper  Delicassen
1  0.05287300  0.52297247 -0.04106815 -0.5886970      -0.04351919 -0.06626363
2 -0.39085706  0.54383861  0.17012470 -0.2698290       0.08630859  0.08904969
3 -0.44652098  0.40807319 -0.02812509 -0.1373793       0.13308016  2.24074190
4  0.09999758 -0.62331041 -0.39253008  0.6863630      -0.49802132  0.09330484
```

```
5  0.83928412 -0.05233688 -0.07926595  0.1736612      -0.23165413  1.29786952
6 -0.20457266  0.33368675 -0.29729863 -0.4955909      -0.22787885 -0.02619421
> str(cust_dataz)
'data.frame':      440 obs. of  6 variables:
 $ Fresh           : num  0.0529 -0.3909 -0.4465 0.1 0.8393 ...
 $ Milk            : num  0.523 0.5438 0.4081 -0.6233 -0.0523 ...
 $ Grocery         : num  -0.0411 0.1701 -0.0281 -0.3925 -0.0793 ...
 $ Frozen          : num  -0.589 -0.27 -0.137 0.686 0.174 ...
 $ Detergents_Paper: num  -0.0435 0.0863 0.1331 -0.498 -0.2317 ...
 $ Delicassen      : num  -0.0663 0.089 2.2407 0.0933 1.2979 ...
```

Last step in preparing the data is to add back the columns with categorical data:

```
> #attach dummy-coded columns to the dataframe witht the sales data
> cust_fullz <- cbind(cust_dataz, customers$Retail, customers$Lisbon,customers$Oporto)
> head(cust_fullz)
       Fresh       Milk     Grocery     Frozen Detergents_Paper  Delicassen customers$Retail customers$Lisbon
1  0.05287300  0.52297247 -0.04106815 -0.5886970      -0.04351919 -0.06626363               1               0
2 -0.39085706  0.54383861  0.17012470 -0.2698290       0.08630859  0.08904969               1               0
3 -0.44652098  0.40807319 -0.02812509 -0.1373793       0.13308016  2.24074190               1               0
4  0.09999758 -0.62331041 -0.39253008  0.6863630      -0.49802132  0.09330484               0               0
5  0.83928412 -0.05233688 -0.07926595  0.1736612      -0.23165413  1.29786952               1               0
6 -0.20457266  0.33368675 -0.29729863 -0.4955909      -0.22787885 -0.02619421               1               0
  customers$Oporto
1                0
2                0
3                0
4                0
5                0
6                0
> str(cust_fullz) #full df for clustering
'data.frame':   440 obs. of  9 variables:
 $ Fresh           : num  0.0529 -0.3909 -0.4465 0.1 0.8393 ...
 $ Milk            : num  0.523 0.5438 0.4081 -0.6233 -0.0523 ...
 $ Grocery         : num  -0.0411 0.1701 -0.0281 -0.3925 -0.0793 ...
 $ Frozen          : num  -0.589 -0.27 -0.137 0.686 0.174 ...
 $ Detergents_Paper: num  -0.0435 0.0863 0.1331 -0.498 -0.2317 ...
 $ Delicassen      : num  -0.0663 0.089 2.2407 0.0933 1.2979 ...
 $ customers$Retail: num  1 1 1 0 1 1 1 1 0 1 ...
 $ customers$Lisbon: num  0 0 0 0 0 0 0 0 0 0 ...
 $ customers$Oporto: num  0 0 0 0 0 0 0 0 0 0 ...
```

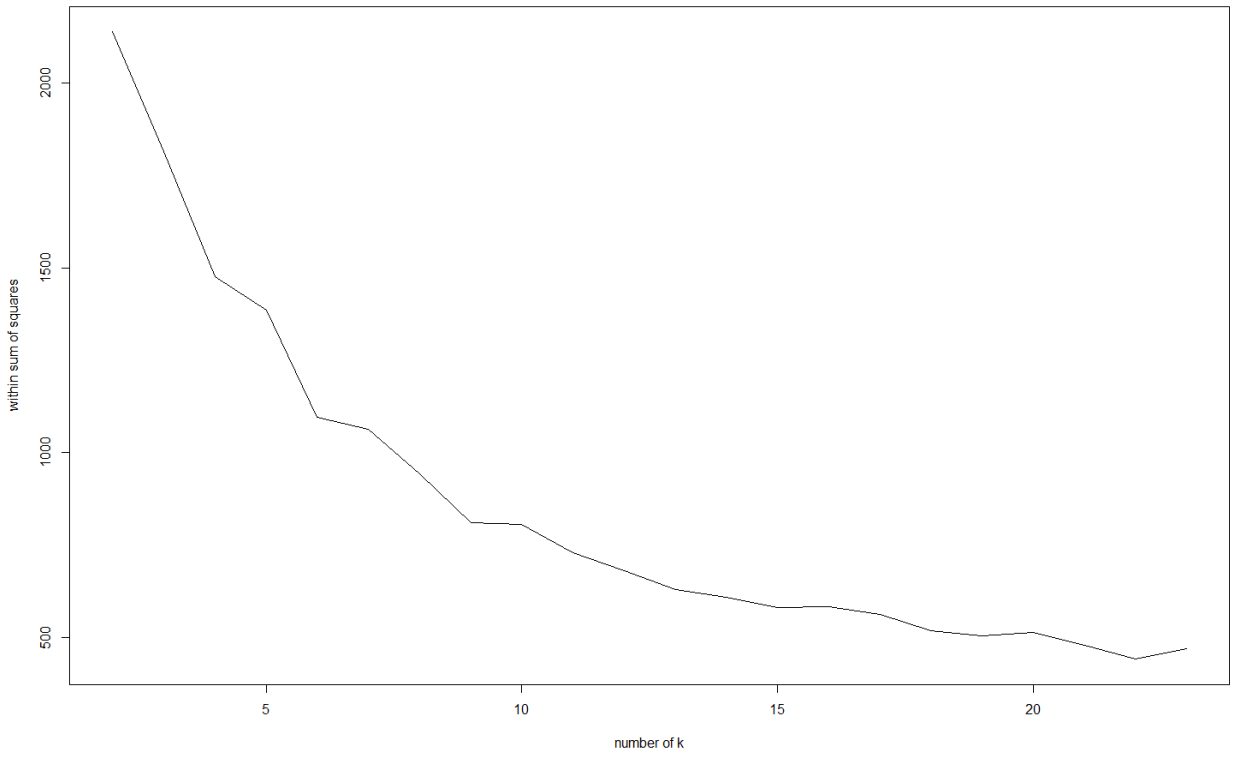
After running the above code, I received the data frame cust_fiullz containing full standardized customer data ready for clustering analysis.

K-means Cluster Analysis

In order to apply the K-means clustering algorithm, I first tried to establish an optimum number of clusters for this data set. I used three different approaches to stashing the best number of clusters and compared their results.

First, I calculated the within sum of squares for a relatively wide range of possible k (to make sure it includes both minimum k=2 and k equal to the square root of the number of observations in the dataset) and plotted it in order to use the "elbow" method using the following code:

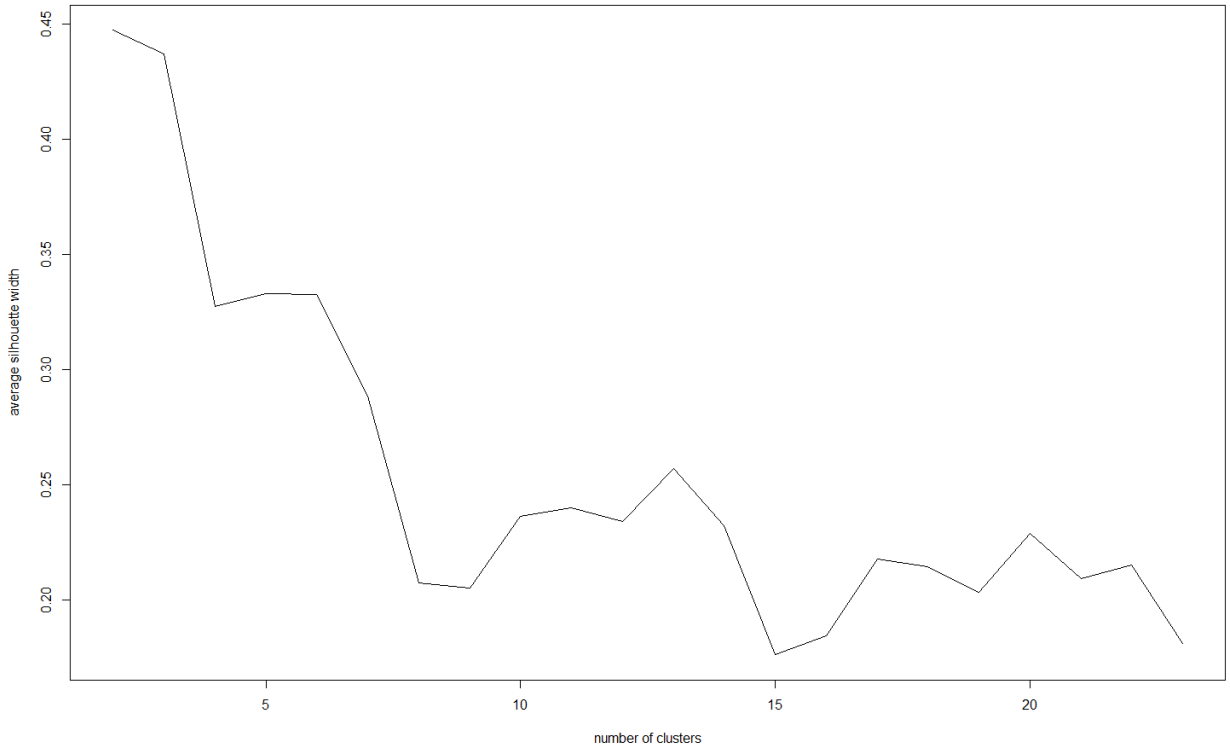
```
> #Calculate within sum of squares
> nk <- 2:23
> set.seed(123)
> wss <- sapply(nk, function(k){
+   kmeans(cust_fullz, centers=k)$tot.withinss
+ })
>
> wss
 [1] 2138.9776 1815.7767 1476.8666 1387.2019 1096.3943 1063.3584  943.6209  811.7298  807.0138  7
30.8010
[11]  682.0656  630.2696  608.0468  580.7228  584.6251  561.7988  517.6911  503.6315  512.8502  4
79.2605
[21]  441.2850  470.9761
> par(mfrow=c(1,1))
> #Plot within sum of squares
> plot(nk, wss, type="l", xlab="number of k", ylab="within sum of squares")
```



As the output above shows, the resulting plot did not have a very clear “elbow” showing the most advantageous number of clusters. Numbers between approximately 3 and 8 all could be considered.

Next, I calculated the average silhouette width of a different number of clusters using the following code:

```
> #average silhouette width
> sw <- sapply(nk, function(k) {
+   cluster.stats(dist(cust_fullz), kmeans(cust_fullz, centers=k)$cluster)$avg.silwidth
+ })
> sw
 [1] 0.4471816 0.4370155 0.3274417 0.3328542 0.3324640 0.2885847 0.2072220 0.2053011 0.2364058 0.
2400225
[11] 0.2340737 0.2569634 0.2321996 0.1761736 0.1844157 0.2179133 0.2143344 0.2034600 0.2290579 0.
2093421
[21] 0.2150139 0.1809548
>
> #plot average silhouette width
> plot(nk, sw, type="l", xlab="number of clusters", ylab = "average silhouette width")
```




```
> #max number of clusters
```

```
> nk[which.max(sw)]
[1] 2
```

As seen above, this method suggests using 2 clusters for optimum results.

Finally, I used clusGap() function to calculate gap statistics:

```
> #gap statistics
> set.seed(123)
> gap_stat <- clusGap(cust_fullz, FUN=kmeans, nstart = 25, K.max = 23, B = 50)
Clustering k = 1,2,..., K.max (= 23): .. done
Bootstrapping, b = 1,2,..., B (= 50)  [one "." per sample]:
.................................................. 50
Warning message:
did not converge in 10 iterations
> gap_stat
Clustering Gap statistic ["clusGap"] from call:
clusGap(x = cust_fullz, FUNcluster = kmeans, K.max = 23, B = 50,     nstart = 25)
B=50 simulated reference sets, k = 1..23; spaceH0="scaledPCA"
 --> Number of clusters (method 'firstSEmax', SE.factor=1): 3
          logW   E.logW      gap      SE.sim
 [1,] 5.726202 7.192076 1.465874 0.010507840
 [2,] 5.588315 7.055454 1.467139 0.010623052
 [3,] 5.473173 6.984309 1.511136 0.011245243
 [4,] 5.438585 6.927122 1.488537 0.010831233
 [5,] 5.345622 6.880371 1.534749 0.010575307
 [6,] 5.307964 6.839318 1.531354 0.009843679
 [7,] 5.254666 6.802701 1.548035 0.010295147
 [8,] 5.210760 6.769982 1.559222 0.010613332
 [9,] 5.165479 6.742187 1.576708 0.010507139
[10,] 5.130666 6.716285 1.585618 0.010066104
[11,] 5.103503 6.692533 1.589030 0.009666854
[12,] 5.076769 6.670072 1.593303 0.009316670
[13,] 5.037499 6.649265 1.611766 0.009707596
[14,] 5.034639 6.629924 1.595285 0.009419537
[15,] 4.990232 6.611778 1.621546 0.009289816
[16,] 4.972893 6.594363 1.621470 0.009378761
[17,] 4.956966 6.578035 1.621069 0.009755392
[18,] 4.930183 6.562579 1.632397 0.009024681
[19,] 4.909903 6.547110 1.637207 0.009247878
[20,] 4.901608 6.533218 1.631610 0.009874003
[21,] 4.875899 6.519420 1.643521 0.009733614
[22,] 4.857407 6.505782 1.648376 0.009324732
[23,] 4.835380 6.493540 1.658160 0.009058457
```

This method did not provide any conclusive results either since the gap kept increasing for larger k values, which suggest that it was just overfitting the data.

In view of this inconclusive results, I decided to proceed with K-means clustering and use k=2,3 and 4 since those values were amount suggested by the above statistics. In addition, from

the business prospective smaller number of clusters can be more beneficial in customer segmentation, then, for example, establishing 23 very small categories of customers.

So, I computed aa K-means cluster with k=2:

```
> #Compute k-means clustering with k=2
> set.seed(123)
> customers_2clusters <- kmeans(cust_fullz, 2)
> customers_2clusters
K-means clustering with 2 clusters of sizes 364, 76

Cluster means:
        Fresh       Milk    Grocery     Frozen Detergents_Paper Delicassen customers$Retail customers$Lisbon customers$Oporto
1  0.05047511 -0.2815149 -0.3380182  0.0271353       -0.3275067 -0.1042951        0.1978022        0.1730769        0.0989011
2 -0.24174922  1.3483083  1.6189291 -0.1299638        1.5685846  0.4995186        0.9210526        0.1842105        0.1447368

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 2 2 1 2 2 2 1 2 1 1 1 1 1 1 2 2 1 1 1
 [62] 2 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 2 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1
[123] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 1 2 1 2 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 2 1
[184] 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 2 1 1 1 2 2 1 2 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[245] 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 2
[306] 1 2 1 1 2 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 2 1 2 1 2 1 1 1 2 1 1 1 1 1 1
[367] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1
[428] 1 1 1 1 1 1 1 1 1 1 2 1 1

Within cluster sum of squares by cluster:
[1] 1079.632 1059.345
 (between_SS / total_SS =  24.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
```

It classified customers into 2 uneven groups of 364 and 76 observations.

Using k=3: resulted in three groups od 49.72. and 319 observations.

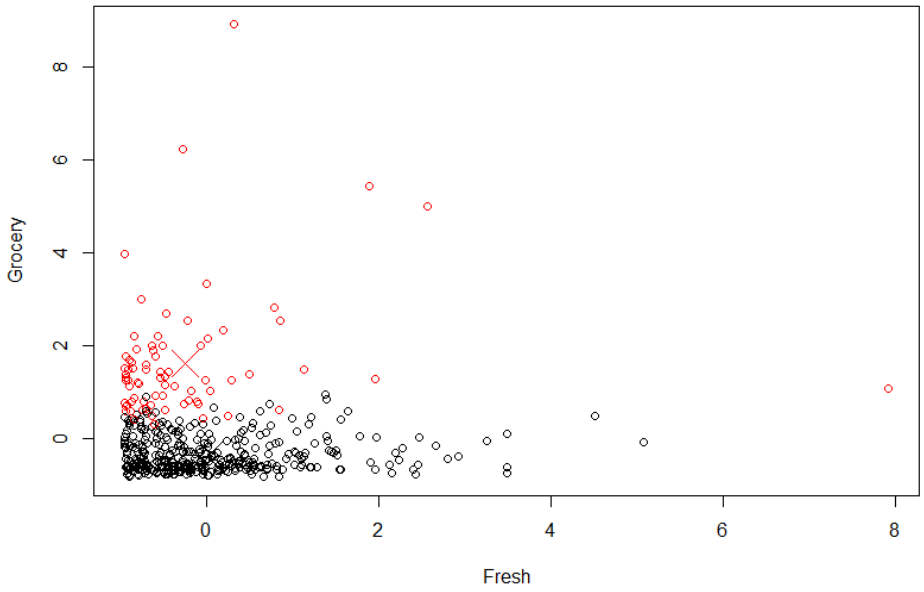
```
> #Compute k-means clustering with k=3
> set.seed(123)
> customers_3clusters <- kmeans(cust_fullz, 3)
> customers_3clusters
K-means clustering with 3 clusters of sizes 49, 72, 319

Cluster means:
       Fresh        Milk    Grocery     Frozen Detergents_Paper Delicassen customers$Retail customers$Lisbon customers$Oporto
1  1.8847077  0.03859924 -0.1644728  1.4031896       -0.3896210  0.8356637        0.1224490        0.1224490       0.04081633
2 -0.4110480  1.30760378  1.6484410 -0.2781687        1.6489816  0.1793536        0.9583333        0.1944444       0.15277778
3 -0.1967248 -0.30106218 -0.3467981 -0.1527528       -0.3123362 -0.1688432        0.2100313        0.1786834       0.10658307

Clustering vector:
  [1] 3 3 3 3 1 3 3 3 2 3 3 1 3 3 3 3 3 3 3 3 3 3 1 2 1 3 3 3 2 1 3 3 3 1 3 3 1 2 2 1 1 3 2 2 3 2 2 3 2 3 3 3 1 3 3 3 2 2 3 3 3
 [62] 2 3 2 3 2 3 3 3 3 3 1 3 3 3 3 3 2 3 3 3 2 3 3 3 2 2 1 3 3 3 3 2 1 3 3 3 3 3 3 2 2 3 1 3 3 3 2 3 2 3 2 1 3 3 3 3 3 3 3 3 3
[123] 3 3 1 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 2 3 3 3 3 3 3 3 3 3 3 2 2 3 3 2 3 3 3 2 3 2 3 3 3 3 2 2 3 2 3 3 1 3 3 3 3 1 3
[184] 1 3 3 3 3 3 3 3 3 2 3 3 1 3 3 3 2 2 1 3 3 2 3 3 3 2 3 2 3 3 3 2 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3
[245] 3 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 1 1 3 3 3 3 3 2 2 2 3 2 3 3 3 3 1 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 3 3 3 1 3 3 3 2 3 3 3 3 3 3 3 3 2 3 2 2
[306] 3 2 3 3 2 3 3 2 3 3 3 2 3 3 3 3 3 1 3 3 3 3 3 2 3 2 1 3 3 3 3 3 3 3 3 3 2 3 3 2 3 3 2 3 2 3 2 3 3 3 3 2 3 3 3 3 3 3 3 3 3
[367] 3 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 1 3 2 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 1 2 3 3 3 3 3 3 1 3 3 2 3 2 3 3 3 3 3 3 1 3
[428] 1 3 3 3 3 3 3 3 3 1 1 2 3 3

Within cluster sum of squares by cluster:
[1] 718.4584 603.6630 502.4615
 (between_SS / total_SS =  35.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
```

K=4 resulted in four clusters of size 15, 10, 111, and 304 customers.

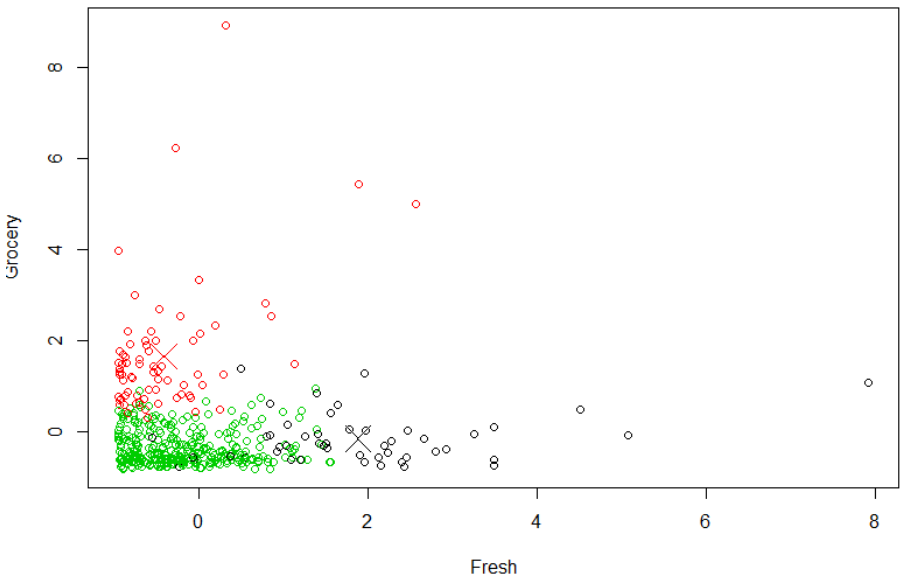
```
> #Compute k-means clustering with k=4
> set.seed(123)
> customers_4clusters <- kmeans(cust_fullz, 4)
> customers_4clusters
K-means clustering with 4 clusters of sizes 15, 10, 111, 304

Cluster means:
        Fresh       Milk    Grocery       Frozen Detergents_Paper  Delicassen customers$Retail customers$Lisbon
1  2.66619953  0.9184175  0.3080437  3.110776046       -0.2973860  2.49576778       0.06666667        0.1333333
2  0.31347349  3.9174467  4.2707490 -0.003570131        4.6129149  0.50279301       1.00000000        0.2000000
3 -0.44418741  0.5642446  0.7949437 -0.344875139        0.8080263  0.03306495       0.93693694        0.1441441
4  0.02031932 -0.3802035 -0.4459437 -0.027449996       -0.4321029 -0.15175874       0.08881579        0.1875000
  customers$Oporto
1       0.06666667
2       0.10000000
3       0.12612613
4       0.10197368

Clustering vector:
  [1] 3 3 3 4 4 4 4 3 4 3 3 4 3 3 3 4 3 4 4 4 4 4 4 1 3 4 4 4 3 4 4 4 4 4 4 4 3 4 3 3 1 4 4 3 3 3 3 3 2 3 3 4 4 4 3 4 4 2 3 4 4 4
 [62] 2 4 3 4 2 4 3 4 4 4 1 4 4 3 4 4 3 4 4 4 3 3 4 4 2 2 1 4 4 4 4 2 1 3 4 3 4 4 4 4 3 3 3 1 4 4 3 3 3 3 4 3 4 4 4 4 4 4 4 4 4 4
[123] 4 3 4 1 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 3 3 4 3 3 3 4 4 3 3 3 3 4 4 4 3 3 4 3 4 3 4 4 4 4 4 1 3
[184] 1 4 4 4 4 3 3 4 4 4 3 4 4 1 3 4 4 3 3 4 4 4 3 4 3 4 3 4 2 4 4 3 3 3 4 3 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[245] 4 3 4 4 4 4 4 4 2 4 4 4 4 4 4 1 4 4 4 4 4 4 3 3 3 4 3 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 1 4 4 4 4 4 4 4 4 3 4 4 4 4 4 3 4 4 3 3 3
[306] 3 3 4 4 3 4 4 3 4 4 3 4 4 4 4 3 4 4 4 4 4 4 1 4 4 4 4 4 4 3 4 2 4 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 3 4 3 4 3 4 4 4 4 3 4 4 4 4 4 4 4 3
[367] 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3
[428] 1 4 4 4 4 4 4 4 4 1 4 3 4 4

Within cluster sum of squares by cluster:
[1] 516.1382 151.9481 254.1753 554.6050
 (between_SS / total_SS =  47.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

It looked like an algorithm was not capable to classify a large group of customers with relatively small purchasing amounts and was starting to overfit the customers data for those who had larger orders in particular categories.

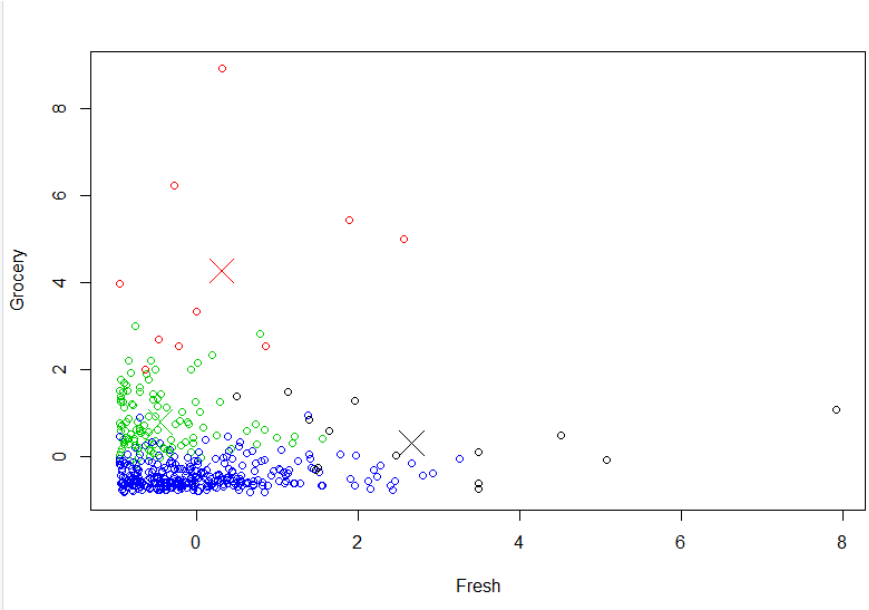
```
> #Plot 2 clusters
> plot(cust_fullz[c("Fresh", "Grocery")], col=customers_2clusters$cluster)
> points(customers_2clusters$centers[, c("Fresh", "Grocery")], col=1:2, pch=4, cex=3)
```




```
> #Plot 3 clusters
> plot(cust_fullz[c("Fresh", "Grocery")], col=customers_3clusters$cluster)
> points(customers_3clusters$centers[, c("Fresh", "Grocery")], col=1:3, pch=4, cex=3)
```



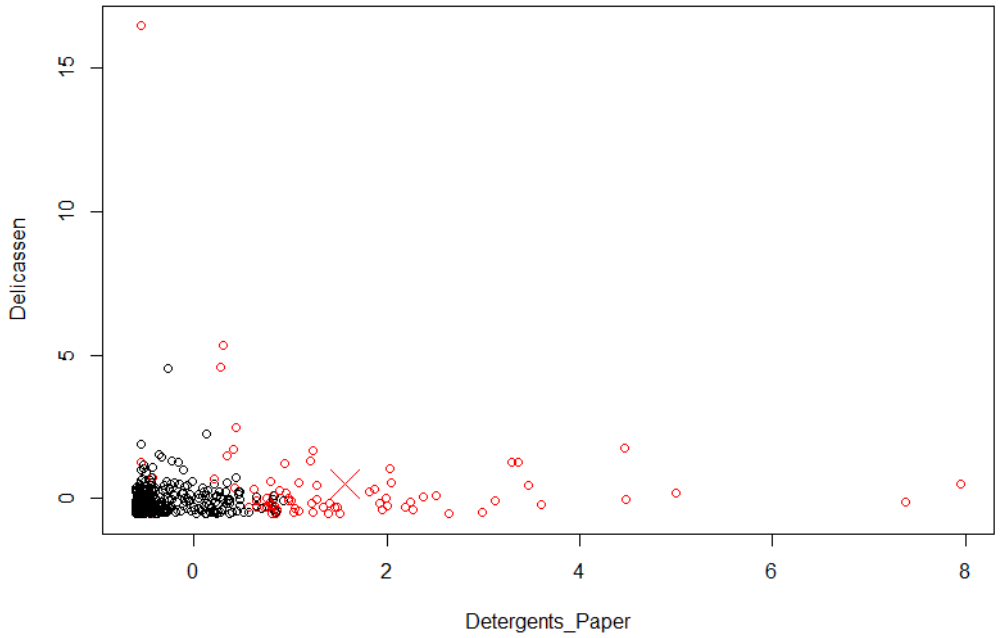

```
> #Plot 4 clusters
> plot(cust_fullz[c("Fresh", "Grocery")], col=customers_4clusters$cluster)
> points(customers_4clusters$centers[, c("Fresh", "Grocery")], col=1:4, pch=4, cex=3)
```

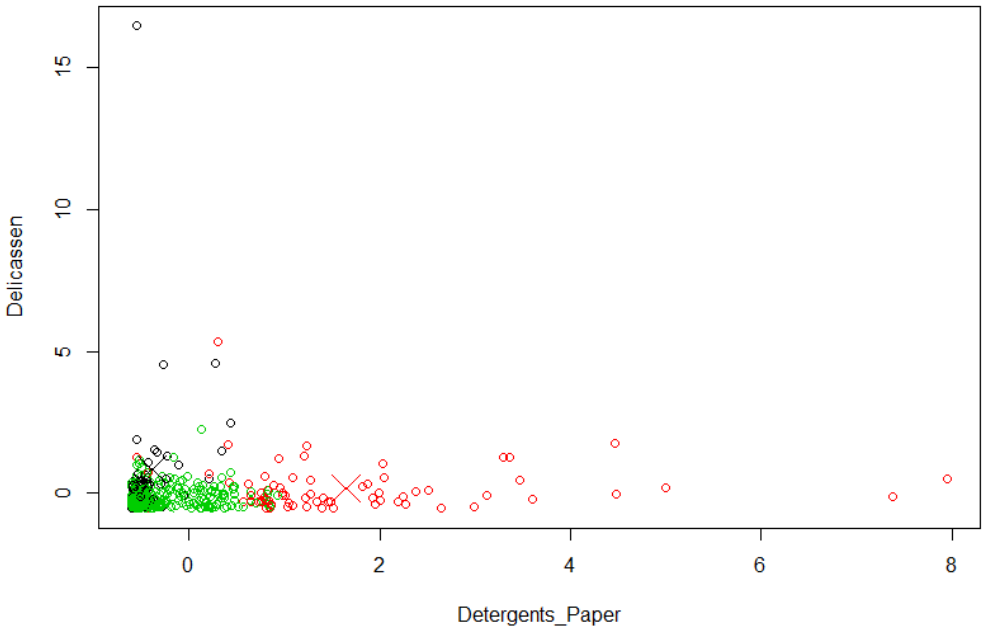


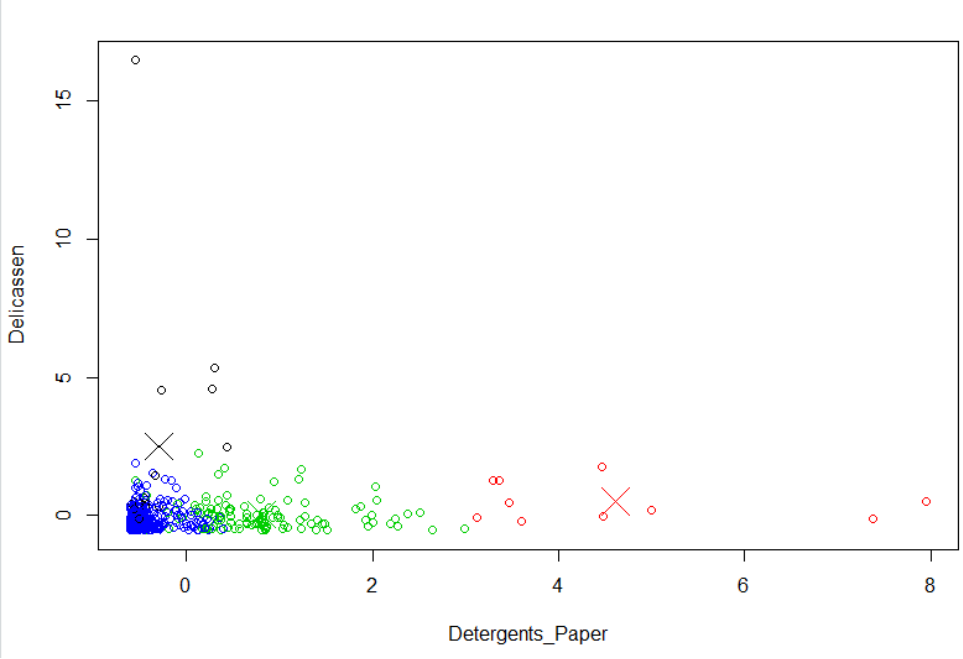
The above cluster plots using the fresh produce and groceries variables seem to confirm the conclusion.

Plotting clusters using a different pair of variables (Detergents_Paper and Delicassen) also showed that a large group customers with relatively smaller order sizes remained difficult to classify:

```
> #Plot 2 clusters
> plot(cust_fullz[c("Detergents_Paper", "Delicassen")], col=customers_2clusters$cluster)
> points(customers_2clusters$centers[, c("Detergents_Paper", "Delicassen")], col=1:2, pch=4, cex=
3)
> #Plot 3 clusters
> plot(cust_fullz[c("Detergents_Paper", "Delicassen")], col=customers_3clusters$cluster)
> points(customers_3clusters$centers[, c("Detergents_Paper", "Delicassen")], col=1:3, pch=4, cex=
3)
> #Plot 4 clusters
> plot(cust_fullz[c("Detergents_Paper", "Delicassen")], col=customers_4clusters$cluster)
> points(customers_4clusters$centers[, c("Detergents_Paper", "Delicassen")], col=1:4, pch=4, cex=
3)
```


Delicassen
0
5
10
15
0
2
4
6
8
Detergents_Paper

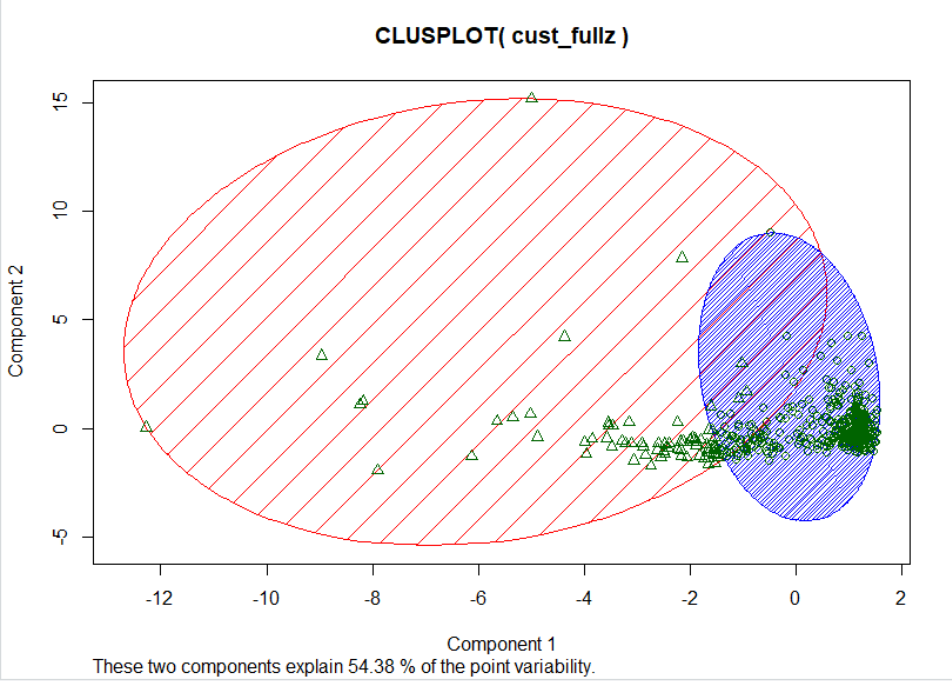

Delicassen
0
5
10
15
0
2
4
6
8
Detergents_Paper



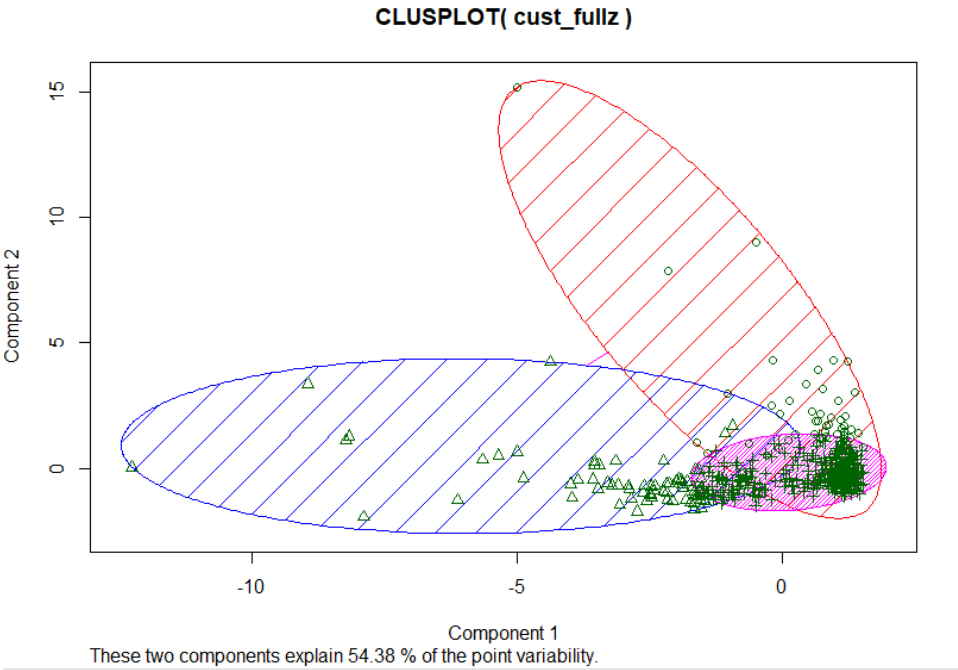

```
> #Bivariate cluster plot
> #2 clusters
> clusplot(cust_fullz, customers_2clusters$cluster, color=TRUE, shade = TRUE)
```

Difficulty in clustering a considerable number of customers were apparent in bivariate cluster plots as well:

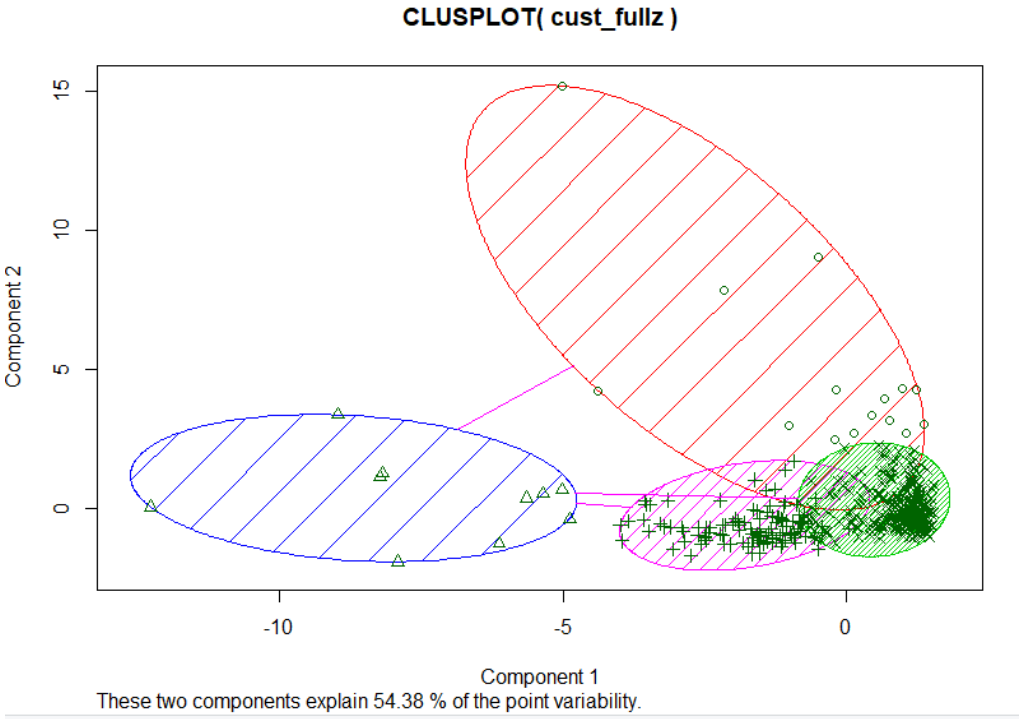

```
> #Bivariate cluster plot
> #2 clusters
> clusplot(cust_fullz, customers_2clusters$cluster, color=TRUE, shade = TRUE)
```




```
> #3 clusters
> clusplot(cust_fullz, customers_3clusters$cluster, color=TRUE, shade = TRUE)
```




```
> #4 clusters
> clusplot(cust_fullz, customers_4clusters$cluster, color=TRUE, shade = TRUE)
```



Hierarchical Clustering

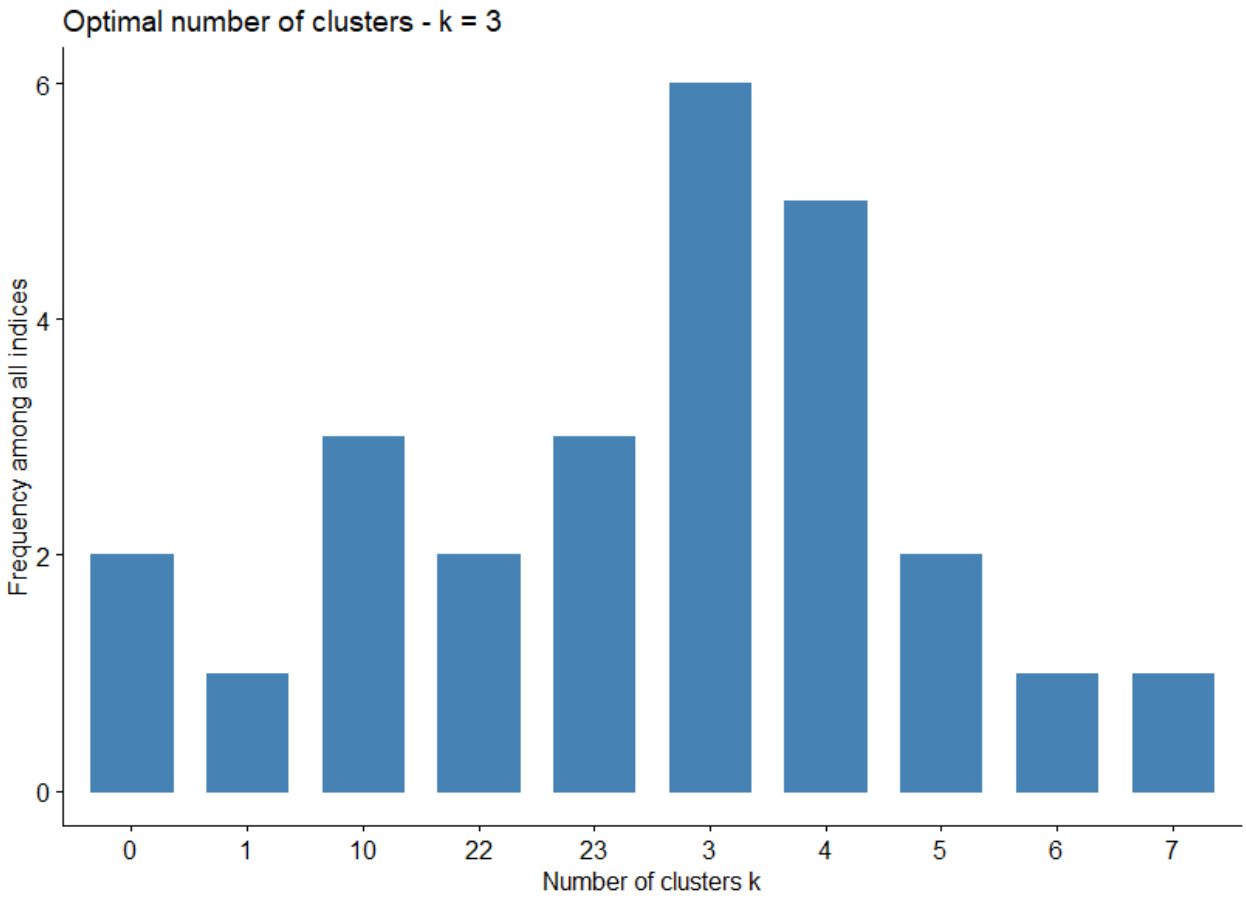
K-means clustering above could not provide conclusive results, so I moved onto hierarchical clustering hoping to get better results. I chose two linkages – the Ward method and the average linkage.

For the Ward method, I first tried to establish the optimal number of clusters using NbClust() function:

```
> #Optimal number of clusters
> set.seed(123)
> num_clust_ward <- NbClust(cust_fullz,  distance = "euclidean", min.nc = 2, max.nc = 23, method=
"ward.D2", index = "all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hube
rt
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dinde
x
```

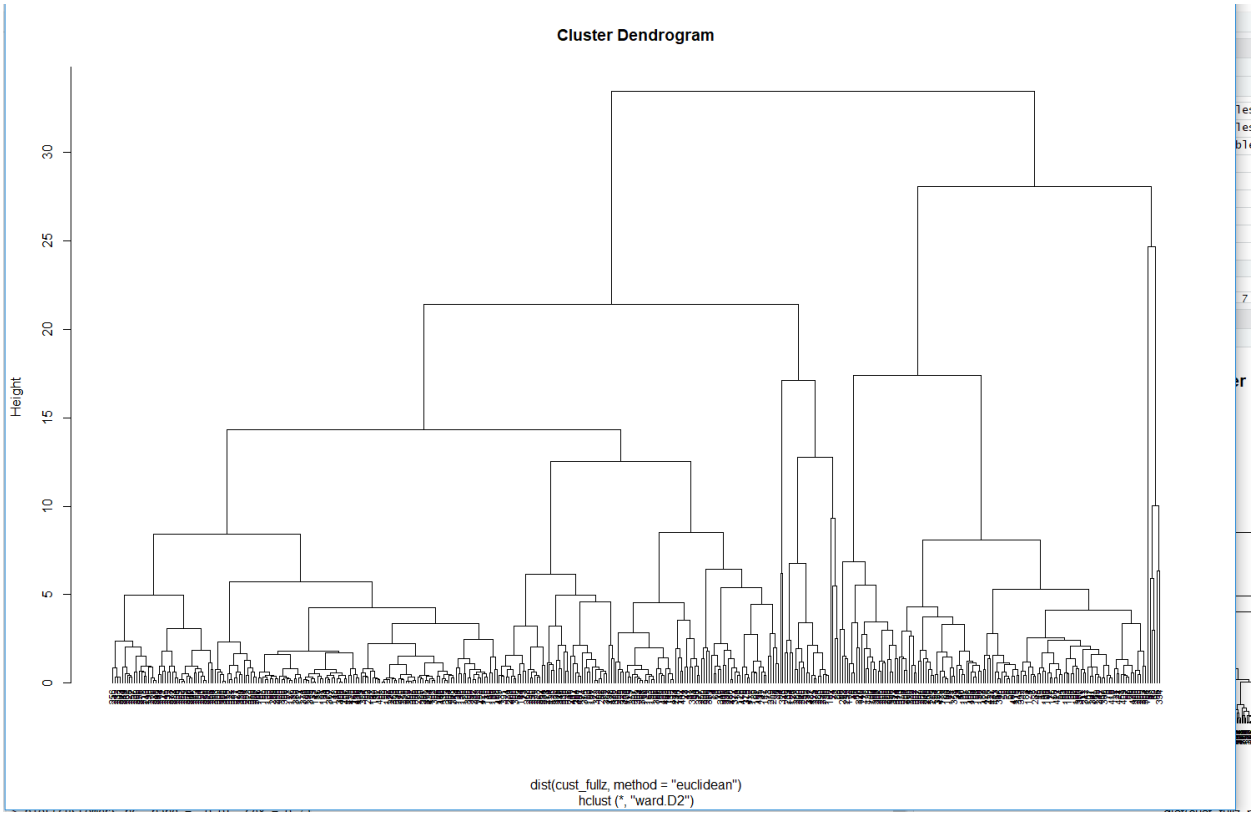
```
                second differences plot) that corresponds to a significant increase of the value
of
                the measure.

*******************************************************************
* Among all indices:
* 6 proposed 3 as the best number of clusters
* 5 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 3 proposed 10 as the best number of clusters
* 2 proposed 22 as the best number of clusters
* 3 proposed 23 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3
```

(Part of the output is omitted.)



As the output above shows, the optimum number of clusters for the Ward method is 3. So I proceeded with constructing the clusters:

```
> plot(customers_hc, hang = -0.01, cex = 0.7)
> #Clustering
> customers_hc <- hclust(dist(cust_fullz, method="euclidean"), method= "ward.D2")
> customers_hc

Call:
hclust(d = dist(cust_fullz, method = "euclidean"), method = "ward.D2")

Cluster method   : ward.D2
Distance         : euclidean
Number of objects: 440

> plot(customers_hc, hang = -0.01, cex = 0.7)
```



Cutting this tree into the three suggested clusters would produce uneven clusters of the size 129, 305, and 3.

```
> #Cutting trees into clusters
```

```
> hc1_cut <- cutree(customers_hc, k=3)
> table(hc1_cut)
hc1_cut
  1   2   3
129 305   6
```

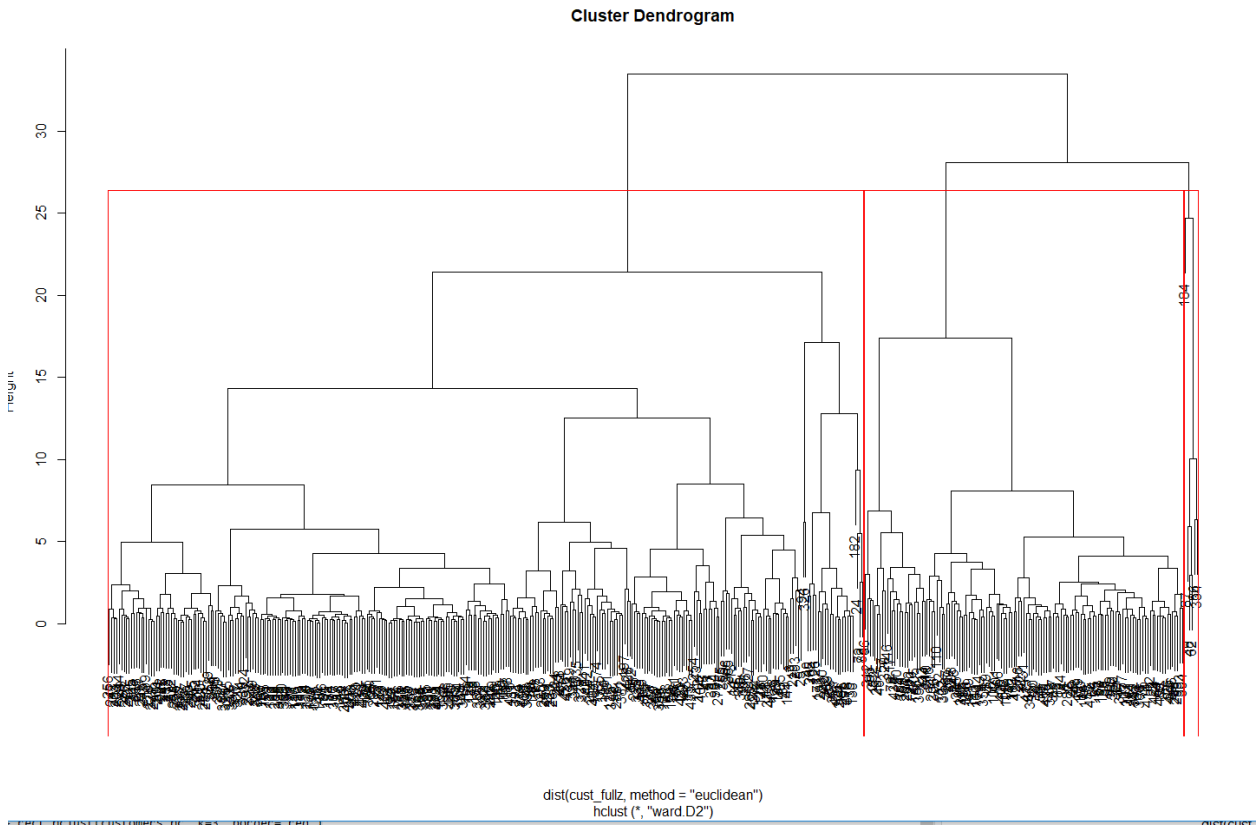
Like in the case of K-means clustering, increasing the number of clusters would produce increasingly smaller clusters and leave a large group of customers in the same cluster:

```
> hc1_cut2 <- cutree(customers_hc, k=2)
> table(hc1_cut2)
hc1_cut2
  1   2
135 305
>
> hc1_cut3 <- cutree(customers_hc, k=4)
> table(hc1_cut3)
hc1_cut3
  1   2   3   4
129 305   5   1
```

To visually inspect this division, I plotted the tree for k=3 using the following code:

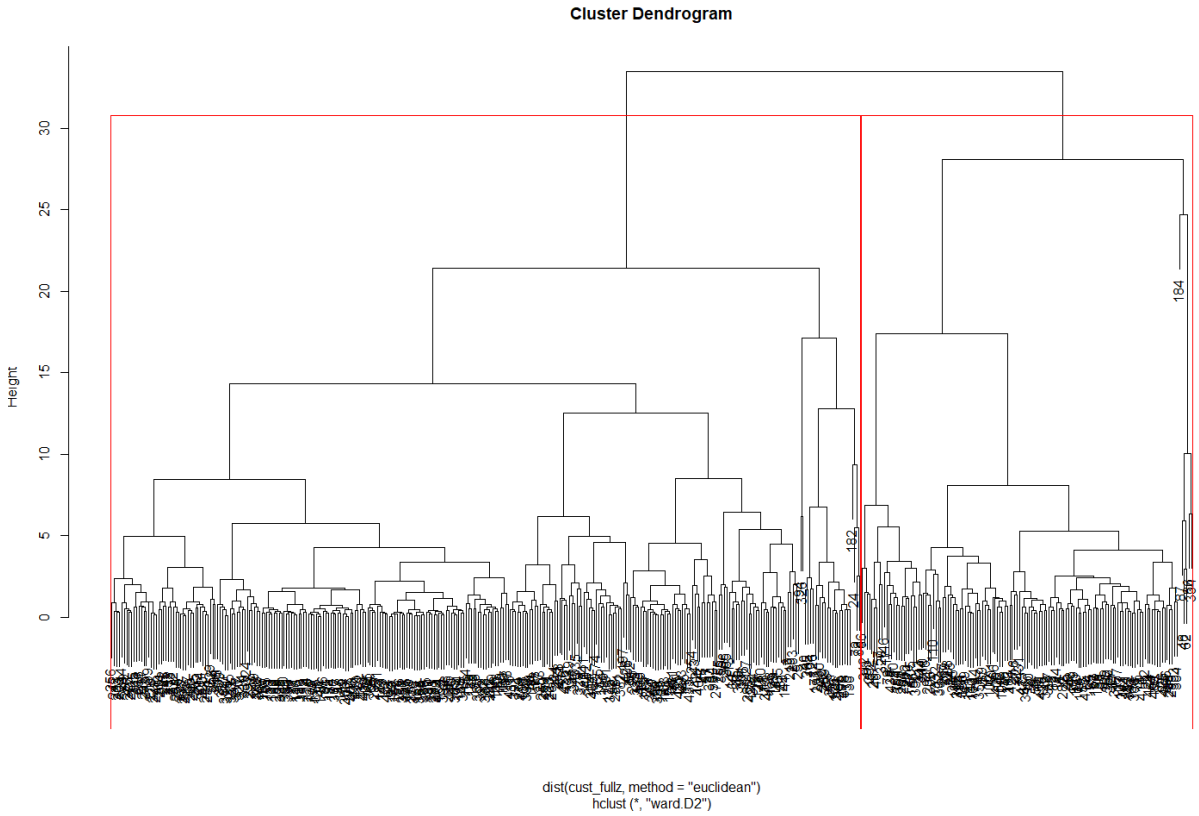
```
> #Plot the tree
> #k=3
> plot(customers_hc)
> rect.hclust(customers_hc, k=3, border="red")
```

With the following result:



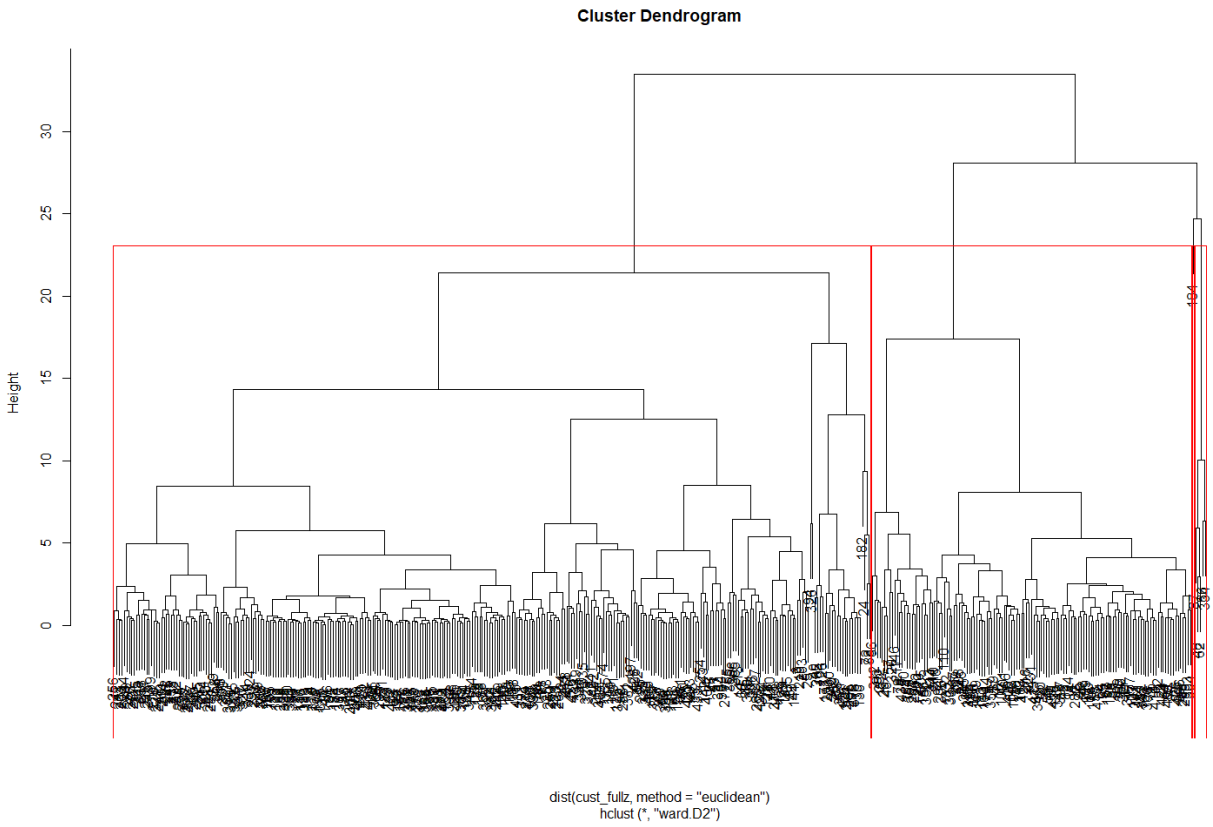

```
> #K=2
> plot(customers_hc)
> rect.hclust(customers_hc, k=2, border="red")
```

Dendrogtram for k=2:



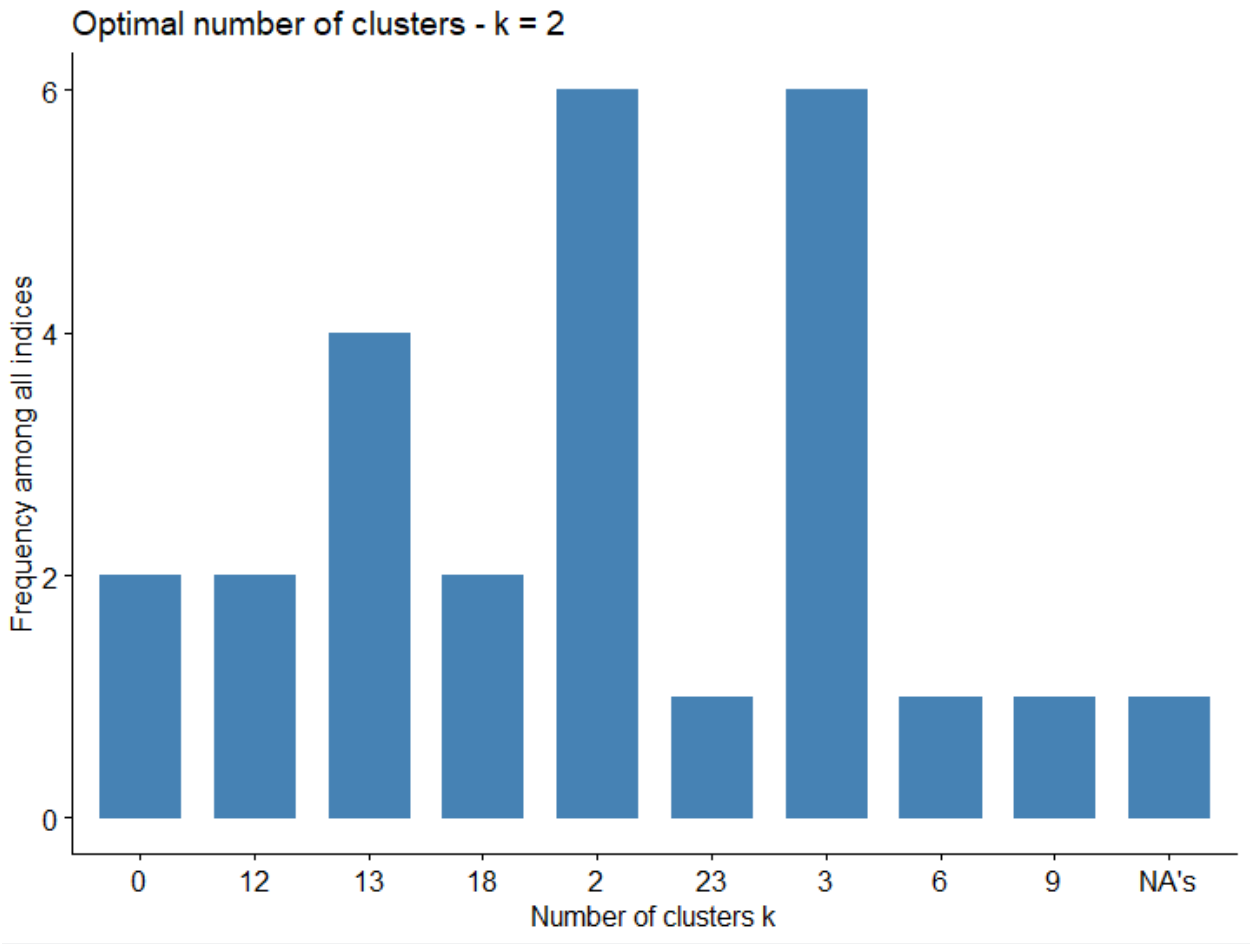
Dendrogram for k=4:

```
> #k=4
> plot(customers_hc)
> rect.hclust(customers_hc, k=4, border="red")
```



Next, I used similar procedures for the average linkage.

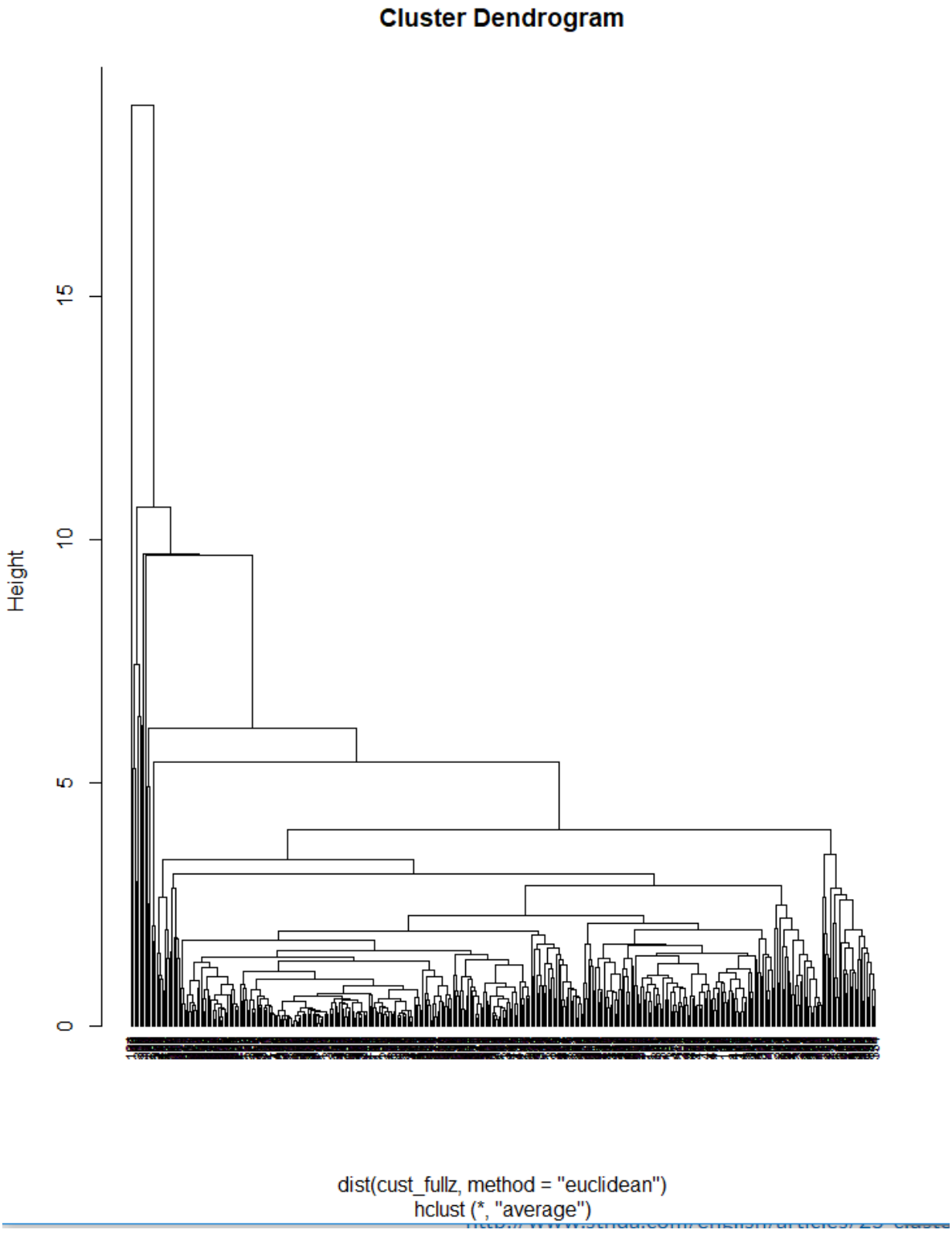
```
> set.seed(123)
> num_clust_av <- NbClust(cust_fullz,  distance = "euclidean", min.nc = 2, max.nc = 23, method= "
average", index = "all")
> fviz_nbclust(num_clust_av, ggtheme = theme_minimal())
Among all indices:
===================
* 2 proposed  0 as the best number of clusters
* 6 proposed  2 as the best number of clusters
* 6 proposed  3 as the best number of clusters
* 1 proposed  6 as the best number of clusters
* 1 proposed  9 as the best number of clusters
* 2 proposed  12 as the best number of clusters
* 4 proposed  13 as the best number of clusters
* 2 proposed  18 as the best number of clusters
* 1 proposed  23 as the best number of clusters
* 1 proposed  NA's as the best number of clusters

Conclusion
=========================
* According to the majority rule, the best number of clusters is  2 .
```



In this case NbClust() function formally returned k=2 as the optimal number of clusters, however both k=2 and k=3 received the same score.

Applying HCA to the dataset produced the following model and cluster dendrogram:

```
> #Clustering
> customers_hc2 <- hclust(dist(cust_fullz, method="euclidean"), method = "average")
> customers_hc2

Call:
hclust(d = dist(cust_fullz, method = "euclidean"), method = "average")

Cluster method   : average
Distance         : euclidean
Number of objects: 440

> plot(customers_hc2, hang = -0.01, cex = 0.7)

>
```

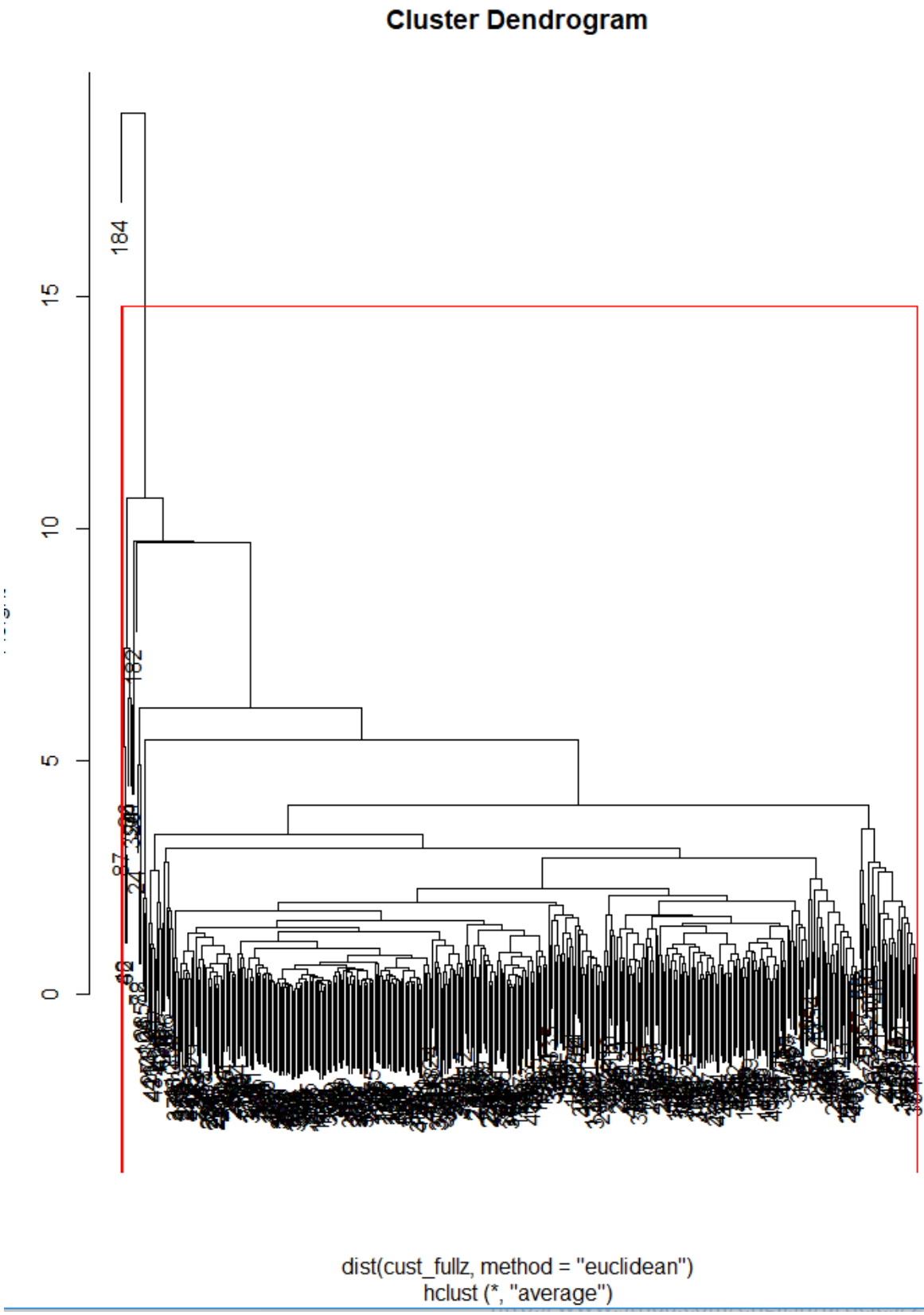

Cluster Dendrogram
Height
0
5
10
15
dist(cust_fullz, method = "euclidean")
hclust (*, "average")

This approach produced even more uneven cluster distribution Cutting this tree at the level k=2 would produce two clusters with sizes 439 and 1; cutting at k=3 would produce clusters with the sizes of 434, 5 and 1; finally cutting at k=4 would result in clusters of sizes 129, 305, 5 and 1).

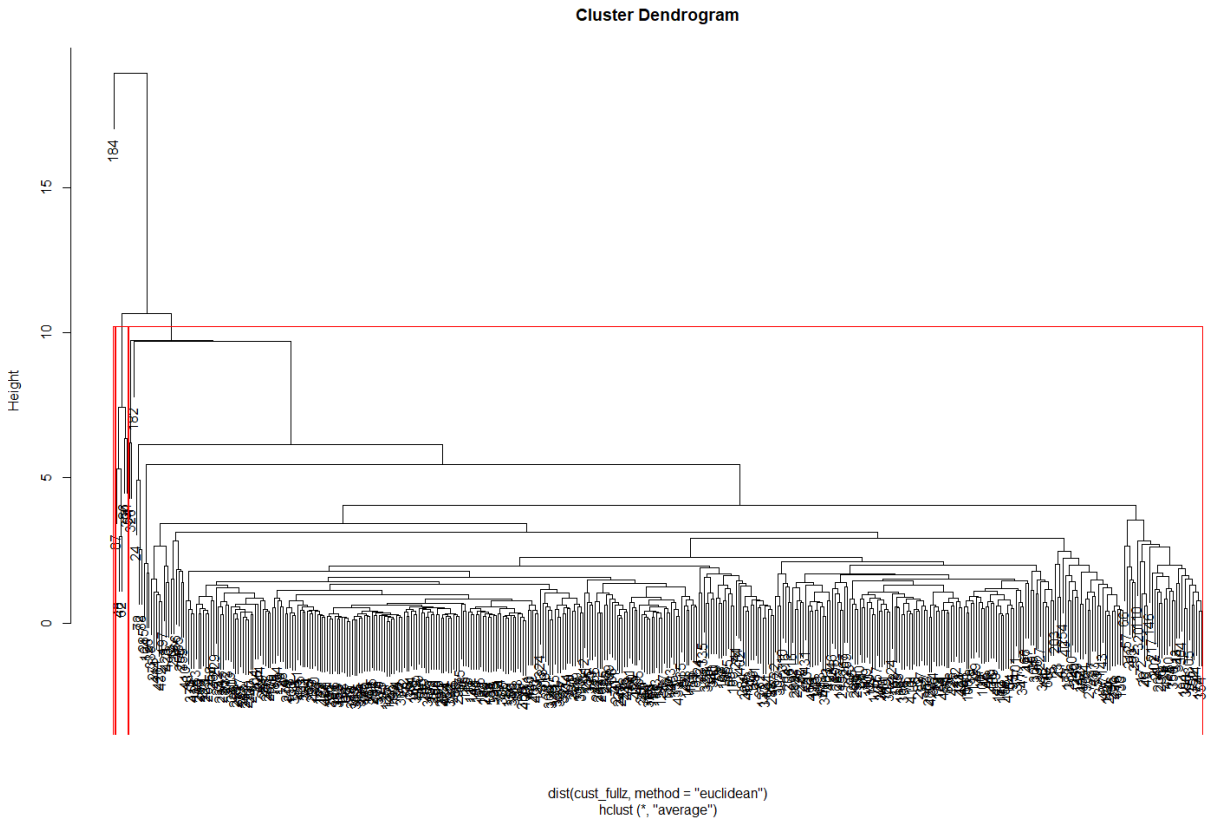
```
> #Cutting trees into clusters
> hc2_cut <- cutree(customers_hc2, k=3)
> table(hc2_cut)
hc2_cut
  1   2   3
434   5   1
>
> hc2_cut2 <- cutree(customers_hc2, k=2)
> table(hc2_cut2)
hc2_cut2
  1   2
439   1
>
> hc2_cut3 <- cutree(customers_hc, k=4)
> table(hc2_cut3)
hc2_cut3
  1   2   3   4
129 305   5   1
```

Cluster dendrograms below illustrate the division:

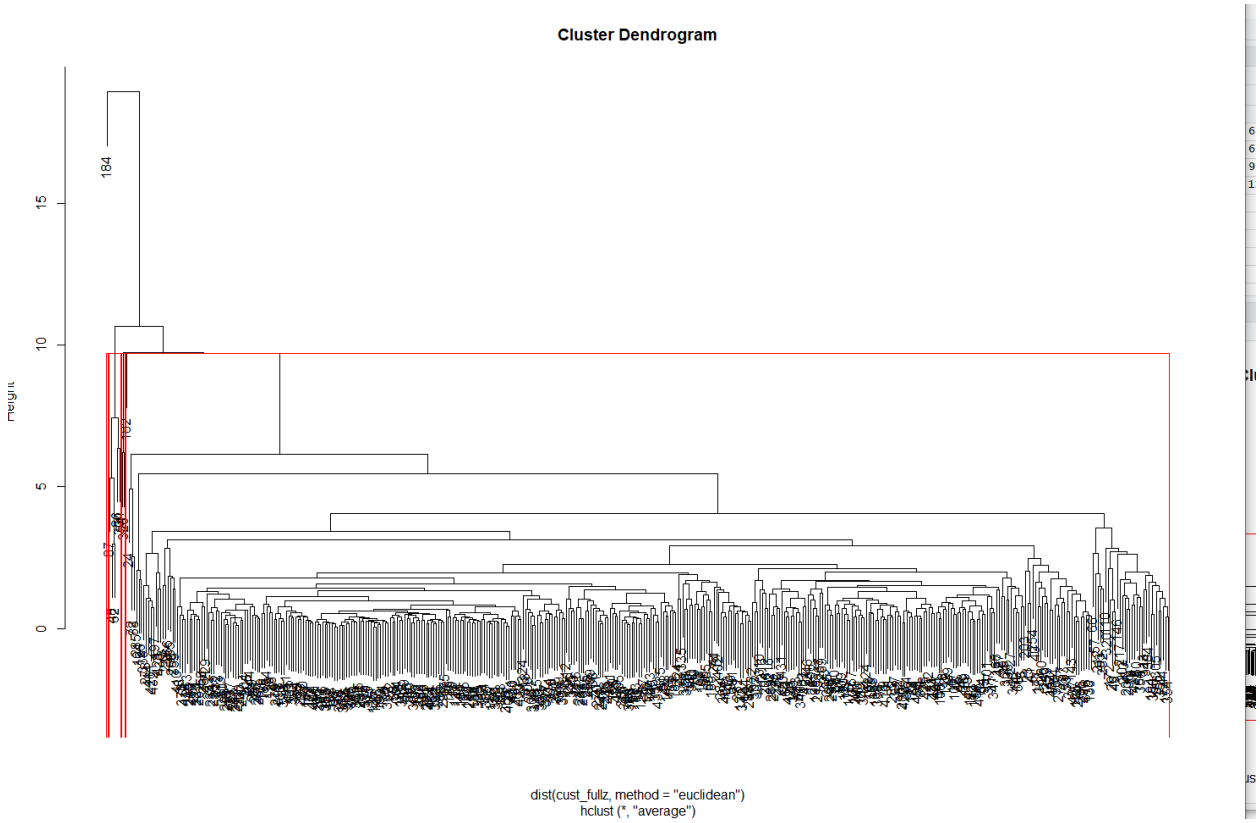
```
> #Plot the tree
> #K=2
> plot(customers_hc2)
> rect.hclust(customers_hc2, k=2, border="red")
```


Cluster Dendrogram
184
182
15
10
5
0
dist(cust_fullz, method = "euclidean")
hclust (*, "average")

```
> #k=3
> plot(customers_hc2)
> rect.hclust(customers_hc2, k=3, border="red")
```




```
> #k=4
> plot(customers_hc2)
> rect.hclust(customers_hc2, k=4, border="red")
```



Conclusions:

Applying both K-means and HCA algorithms to the same wholesale customers dataset demonstrated that these clustering techniques are sensitive to the high number of outliers in this dataset. The resulting clusters included the majority of the customers in the same cluster and were more closely fitting the customers with the unusually high purchase volumes. While it might be useful for detecting unusual purchasing behavior and some particular clients that might benefit from extra personalized attention, this clustering approach would be less valuable, for example, for creating marketing campaigns or choosing customer groups for promotions. As a result, the effectiveness of these clusters depends on the business objectives.

Similarly, possible ways to improve the result and would also depend on the business objective. If the main focus is on the majority of the clients, then excluding outliers from the

dataset could be helpful. If the goal is clustering all existing clients, then including additional variables into the analysis or adding rule-based cluster approach could also be helpful.