MSDS 680 - Machine Learning
Alla Topp

K-means and Hierarchical Clustering

Objective: What product categories are more popular to buyers?

The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units on diverse product categories. This data set is taken from UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/Wholesale+customers

Attribute Information:

1. CHANNEL: customers Channel - Horeca (Hotel/Restaurant/Cafe) or Retail channel (Nominal)
2. REGION: customers Region Lisnon, Oporto or Other (Nominal)
3. FRESH: annual spending (m.u.) on fresh products (Continuous);
4. MILK: annual spending (m.u.) on milk products (Continuous);
5. GROCERY: annual spending (m.u.)on grocery products (Continuous);
6. FROZEN: annual spending (m.u.)on frozen products (Continuous)
7. DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)
8. DELICATESSEN: annual spending (m.u.)on and delicatessen products (Continuous);
9.

The first step is to load the data into R and do some data exploration before we clean it, normalize and apply k-means method.

```
#Alla Topp
#MSDS 680 Machine Learning
#K-Means and HCA
#_______________________________________________________

library(corrplot)
library(cluster)
library(factoextra)
library(NbClust)

wholesale_df <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesal
                          na.strings="?", sep = ",", header = TRUE )
str(wholesale_df)
summary(wholesale_df) #produce summary of the data
sum(is.na(wholesale_df)) #checking for missing values
```

```
> str(wholesale_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 ...
> summary(wholesale_df) #produce summary of the data
    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  
> sum(is.na(wholesale_df)) #checking for missing values
[1] 0
```

All the attributes are of same scale except “channel” and “region” and those two are categorical. We can ignore those attributes for clustering.

```
#removing channel and region variables
new_wholesale <- wholesale_df #creating df without channel and region
new_wholesale$Channel <- NULL
new_wholesale$Region <- NULL
summary(new_wholesale) #checking if two columns are gone
```

```
> summary(new_wholesale) #checking if two columns are gone
     Fresh             Milk          Grocery          Frozen        Detergents_Paper    Delicassen     
 Min.   :     3   Min.   :   55   Min.   :    3   Min.   :   25.0   Min.   :    3.0   Min.   :    3.0  
 1st Qu.:  3128   1st Qu.: 1533   1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2  
 Median :  8504   Median : 3627   Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5  
 Mean   : 12000   Mean   : 5796   Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9  
 3rd Qu.: 16934   3rd Qu.: 7190   3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2  
 Max.   :112151   Max.   :73498   Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0  
```

Like in the first two assignments in this class we are going to normalize the data and will see how it affect finding optimal number of clusters:

```
#normalizing the data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
new_wholesale[1:6] <- as.data.frame(lapply(new_wholesale[1:6], normalize))
summary(new_wholesale)
```

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

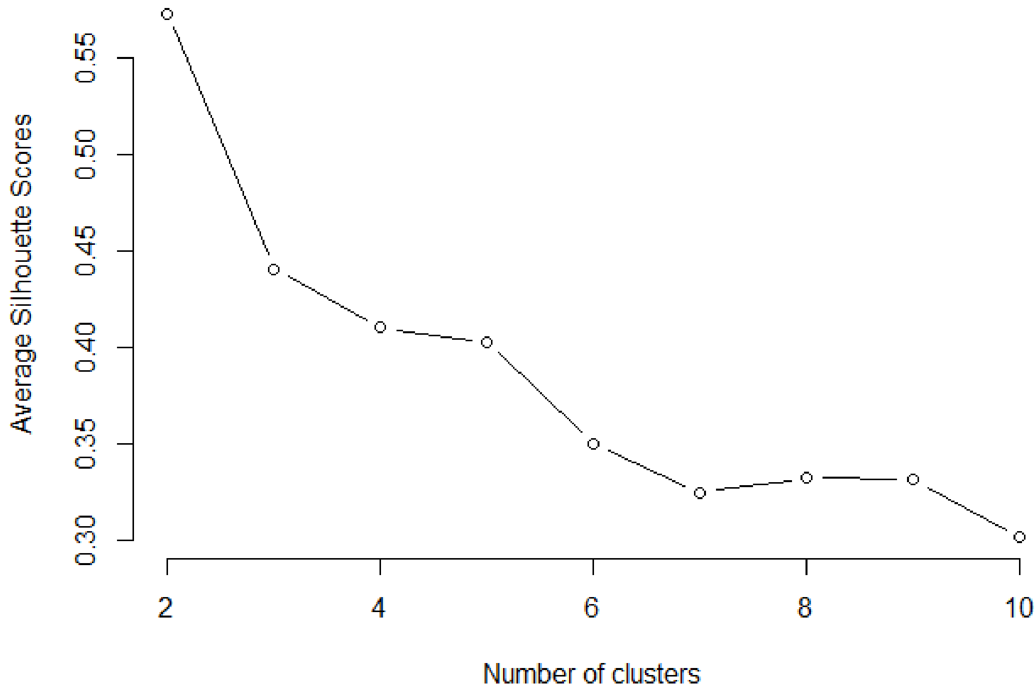
As seen above, all of the values are now between 0 and 1. We can now trust that each variable will be weighted equally in our analysis.

Before we do the actual clustering, we need to identity the Optimal number of clusters (k) for this data set of wholesale customers. One of the methods to determine the number of clusters is Silhouette method, which we will use in out analysis.

```
#finding optimal clusters for the given data set with silhouette method
silhouette_score <- function(k){
  km <- kmeans(new_wholesale, centers = k, nstart=25)
  ss <- silhouette(km$cluster, dist(new_wholesale))
  mean(ss[, 3])}

k <- 2:10
avg_sil <- sapply(k, silhouette_score)
plot(k, type='b', avg_sil, xlab='Number of clusters', ylab='Average Silhouette Scores', frame=FALSE)
k[which.max(avg_sil)]
```

```
> k[which.max(avg_sil)]
[1] 2
```

The above method of calculating silhouette score using silhouette() and plotting the results states that optimal number of clusters is 2. Also, looking at the graph below we could see it (the highest point on the left).



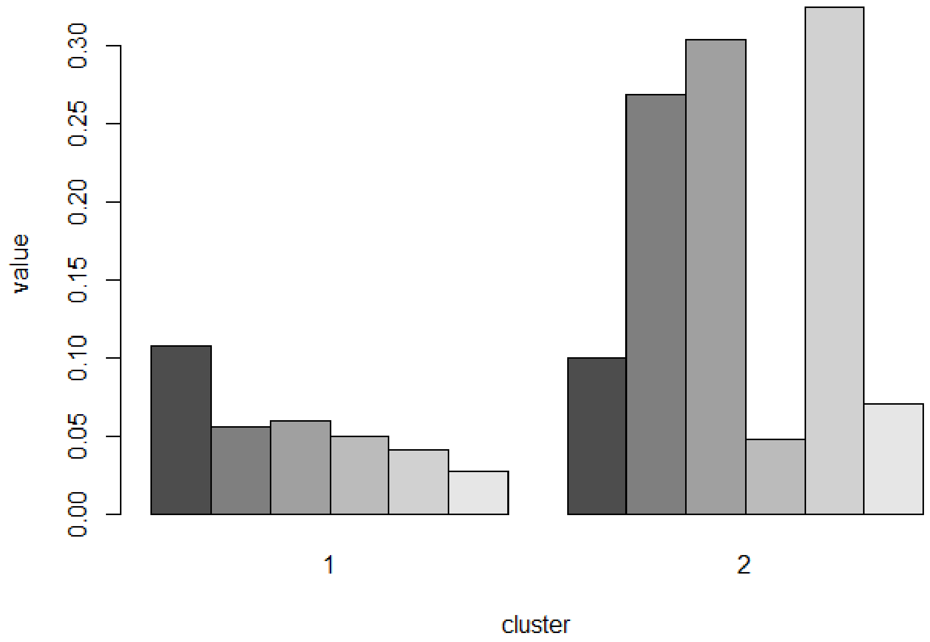
Based on the above information we can create two clusters.

```
#creating two clusters
set.seed(22)
fit = kmeans(new_wholesale, 2)
fit
```

```
> fit
K-means clustering with 2 clusters of sizes 394, 46

Cluster means:
       Fresh       Milk    Grocery     Frozen Detergents_Paper Delicassen
1 0.10783586 0.05593035 0.06023356 0.05026197       0.04094287 0.02718838
2 0.09962418 0.26868693 0.30354600 0.04850001       0.32375723 0.07077642

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 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 2 2 2 1 2 1 1 1 1 1 1
 [57] 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
[113] 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 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 2 1 1
[169] 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 2 1 1 1 1 2 1 1 1 1 1 1 1
[225] 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1
[337] 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1
[393] 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 2 1 1

Within cluster sum of squares by cluster:
[1] 10.170431  7.074975
 (between_SS / total_SS =  30.8 %)

Available components:

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

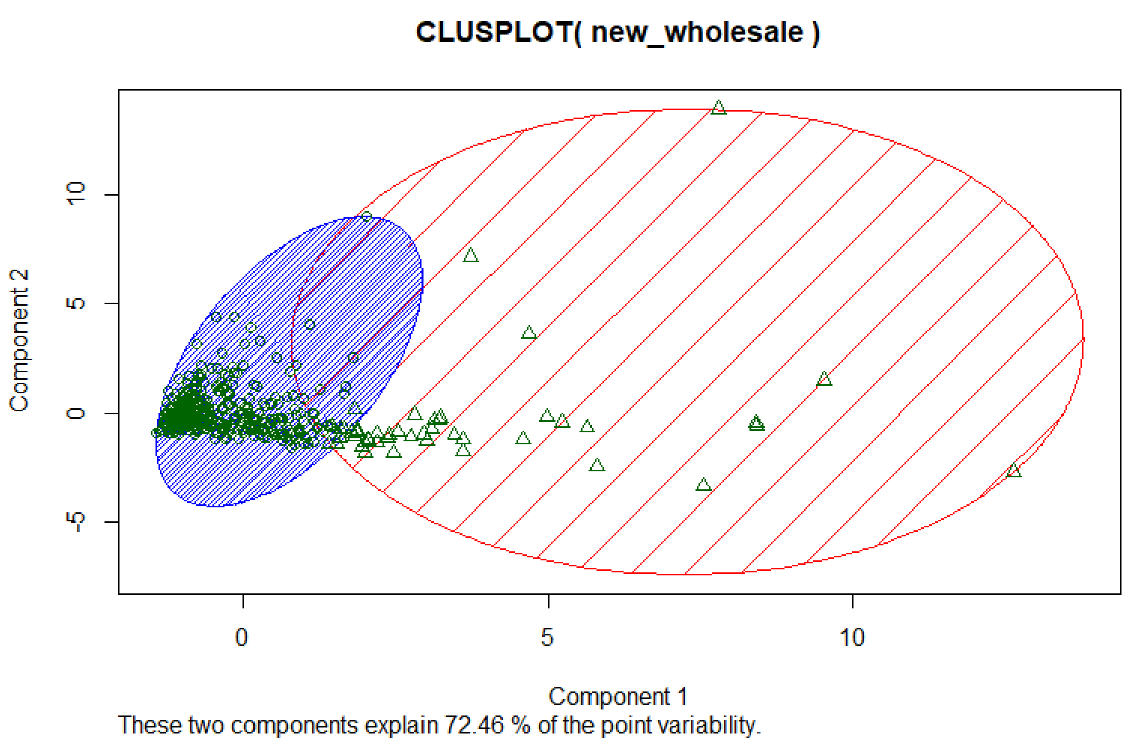
We can see that we have 2 clusters which are of the sizes 394 (1st cluster) and 46 (2nd cluster).

```
#inspect the center of each cluster using barplot
barplot(t(fit$centers), beside = TRUE, xlab="cluster", ylab="value")
plot(new_wholesale$Fresh,new_wholesale$Frozen, col = fit$cluster) #scatter plot of the data
```



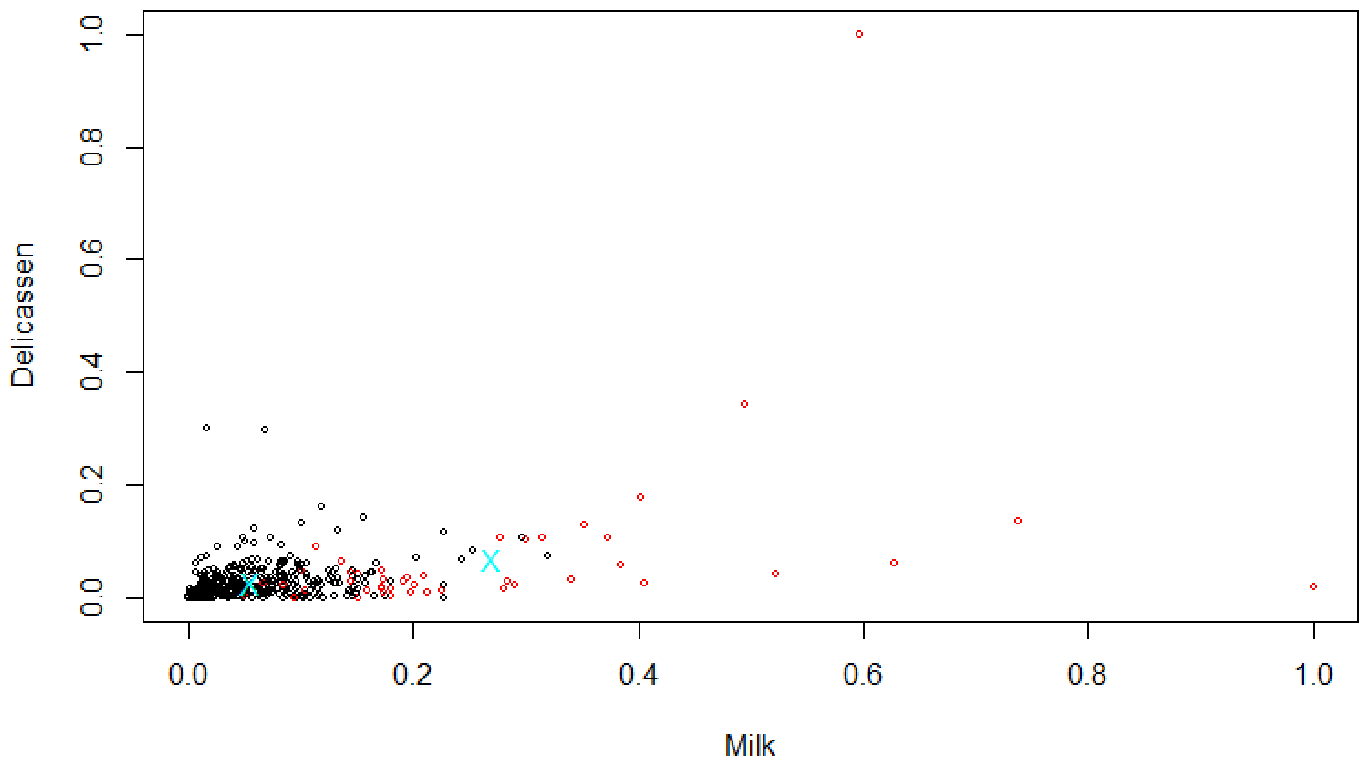
This indicates that the products “milk”, “grocery” and “detergents_paper” are being purchased at a much higher rate from the customers in cluster 2.

We can use a bivariate cluster plot to first reduce variables into two components, and then use components, such as axis and circle, as clusters to show how data is clustered.



Looking at the above plot we can see there is something off. The clusters above contain one of size 394 and the other with the remaining 46 observations which we got after running kmeans() function above. These two clusters might not be optimal for drawing valuable insights, as it seemed to gather most of the data together into one cluster and then the remaining, scattered data into another. Although, we are going to consider two of the variables and loo at them at the graphs below. We decided to choose variables milk and delicatessen because going to the store it would make sense to buy those products together.

```
#plotting claster with two variables "milk" and "Delicassen"
plot(new_wholesale[c("Milk", "Delicassen")], col = fit$cluster, cex = .5)
points(fit$centers[,c("Milk", "Delicassen")], col=5, pch="X")
```



Looking at the graph above where we see milk and delicatessen points on the graph based on the two clusters we created, some of the points slightly overlap, but mostly correlated. Also, we can see that center of the clusters are distant from each other which indicates we chose good variables to look at and compare.

HCA clustering

```
#HCA
hca_wholesale <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20custo
                  header = TRUE, sep = ",")
hca_wholesale$Channel <- NULL #removing column channel
hca_wholesale$Region <-NULL #removing culumn region

#normalizing data for hca
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
hca_wholesale[1:6] <- as.data.frame(lapply(hca_wholesale[1:6], normalize))
```

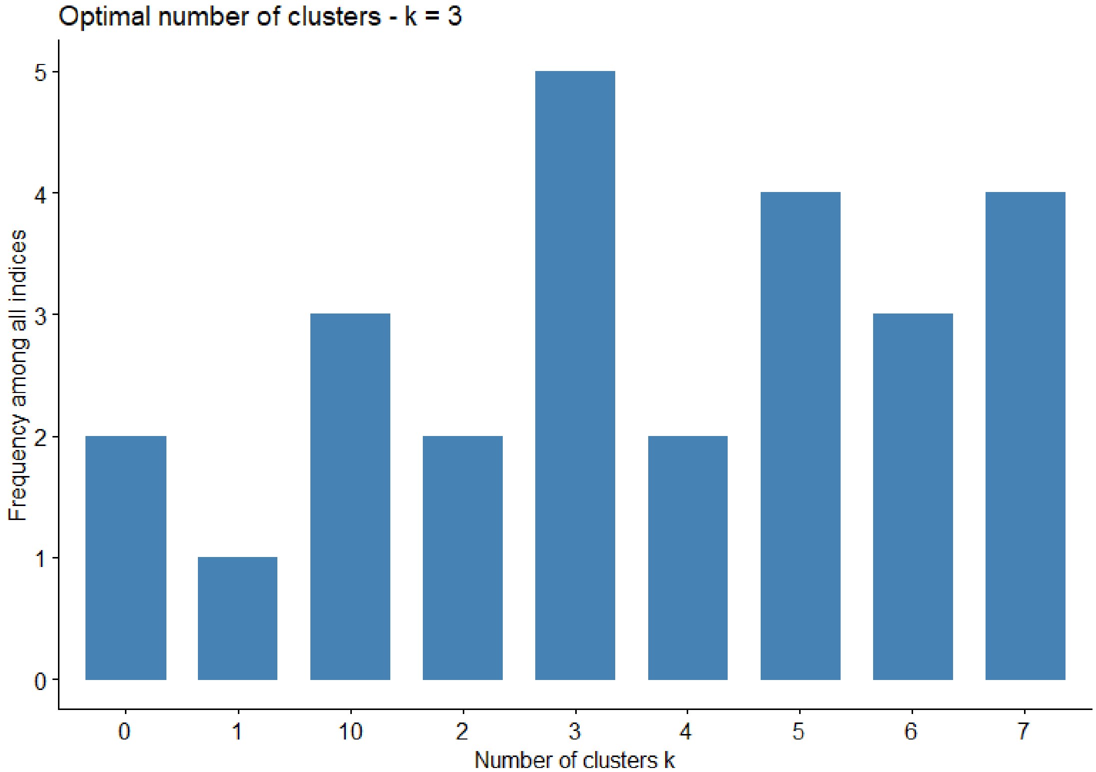
MSDS 680 - Machine Learning
Alla Topp

We'll use packages "NbClust" and "factoextra" to choose the optimal number of clusters for our HCA analysis. Because this package shows the optimal number of clusters in the beginning and makes it easier to move on in the analysis.

```
nb_wholesale <- NbClust(hca_wholesale, distance = "euclidean", min.nc = 2,
                 max.nc = 10, method = "ward.D2")
fviz_nbclust(nb_wholesale)

hc_clust = hclust(dist(hca_wholesale, method="euclidean"), method="ward.D2")
hc_clust
```

```
> nb_wholesale <- NbClust(hca_wholesale, distance = "euclidean", min.nc = 2,
+                max.nc = 10, method = "ward.D2")
*** : 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 Hubert
                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 Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

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

                   ***** Conclusion *****

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


*******************************************************************
Warning message:
In pf(beale, pp, df2) : NaNs produced
> fviz_nbclust(nb_wholesale)
Among all indices:
===================
* 2 proposed  0 as the best number of clusters
* 1 proposed  1 as the best number of clusters
* 2 proposed  2 as the best number of clusters
* 5 proposed  3 as the best number of clusters
* 2 proposed  4 as the best number of clusters
* 4 proposed  5 as the best number of clusters
* 3 proposed  6 as the best number of clusters
* 4 proposed  7 as the best number of clusters
* 3 proposed  10 as the best number of clusters

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

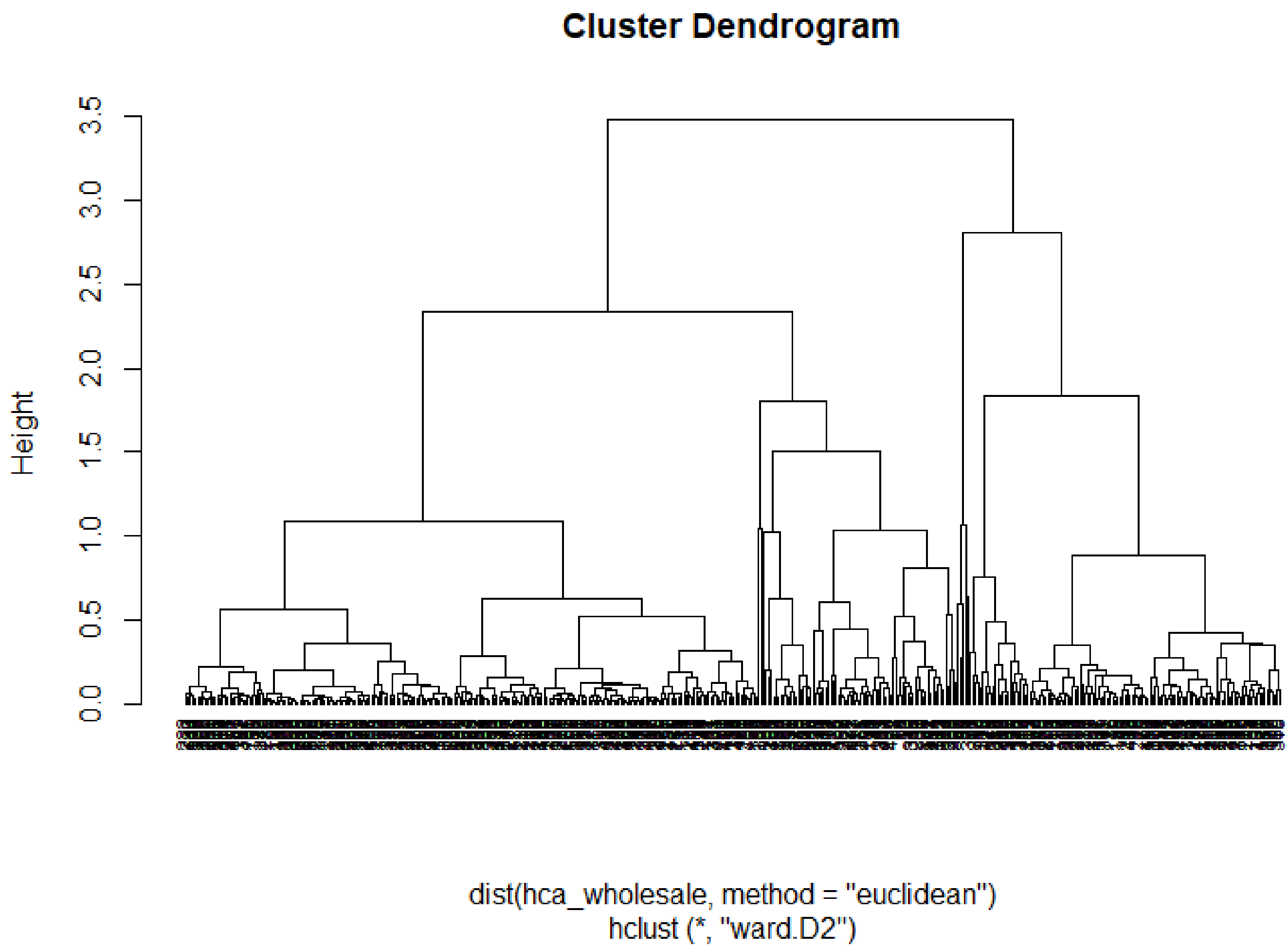
MSDS 680 - Machine Learning
Alla Topp



As shown above, the optimal number of clusters is 3.

```
> hc_clust = hclust(dist(hca_wholesale, method="euclidean"), method="ward.D2")
> hc_clust

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

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

```
#plotting a dendogram
plot(hc_clust, hang = -0.01, cex = 0.7)
```

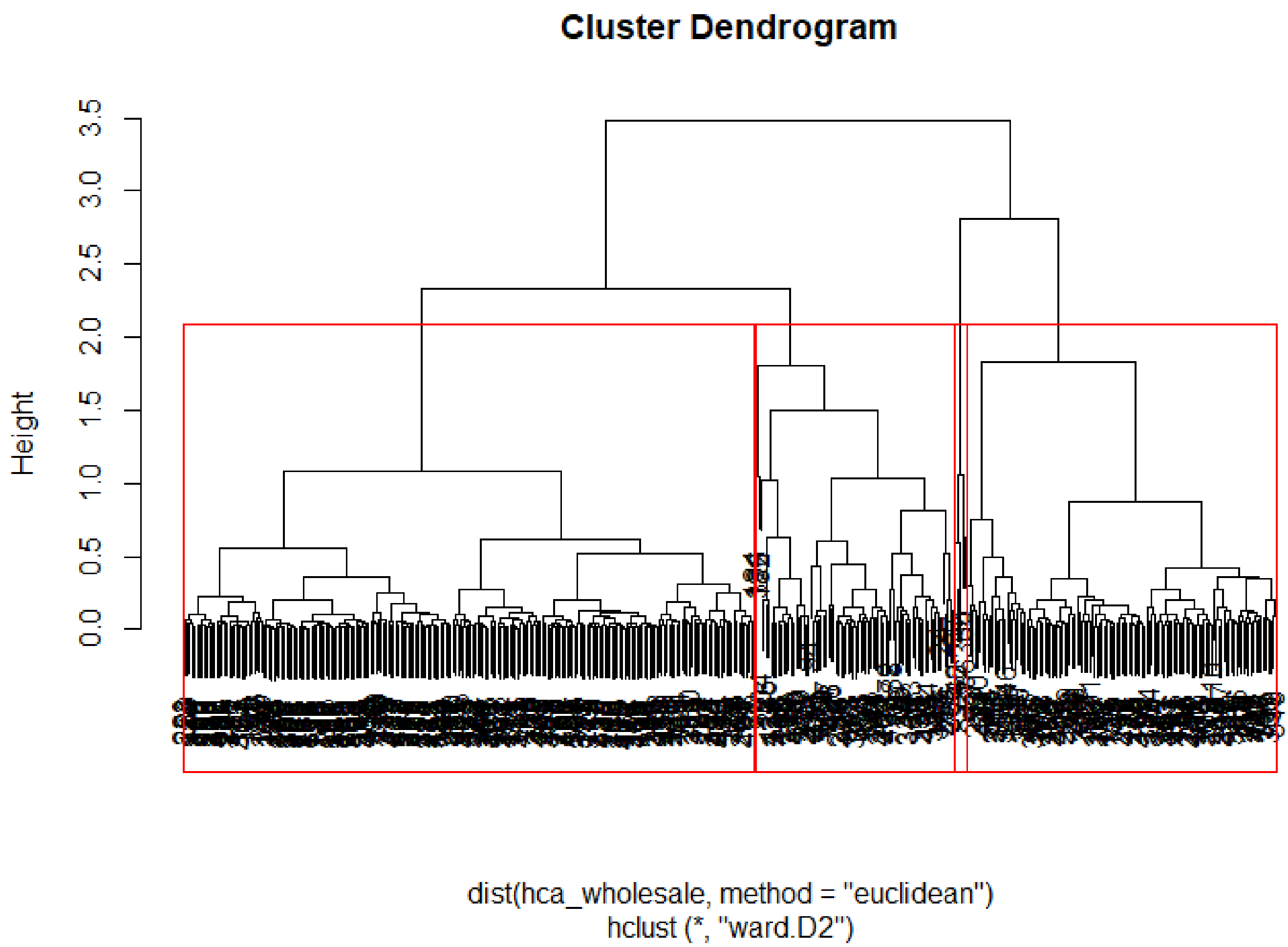


We'll now cut the dentrogram into three clusters.

```
#cut the hierarchy of clusters into a given number of clusters
fit <- cutree(hc_clust, k = 3) # categorize the data into three groups
table(fit) #Count the number of data within each cluster

plot(hc_clust) #visualizing how data is clustered
rect.hclust(hc_clust, k = 4, border = "red") #Cutting at different heights of the dendrogram
```

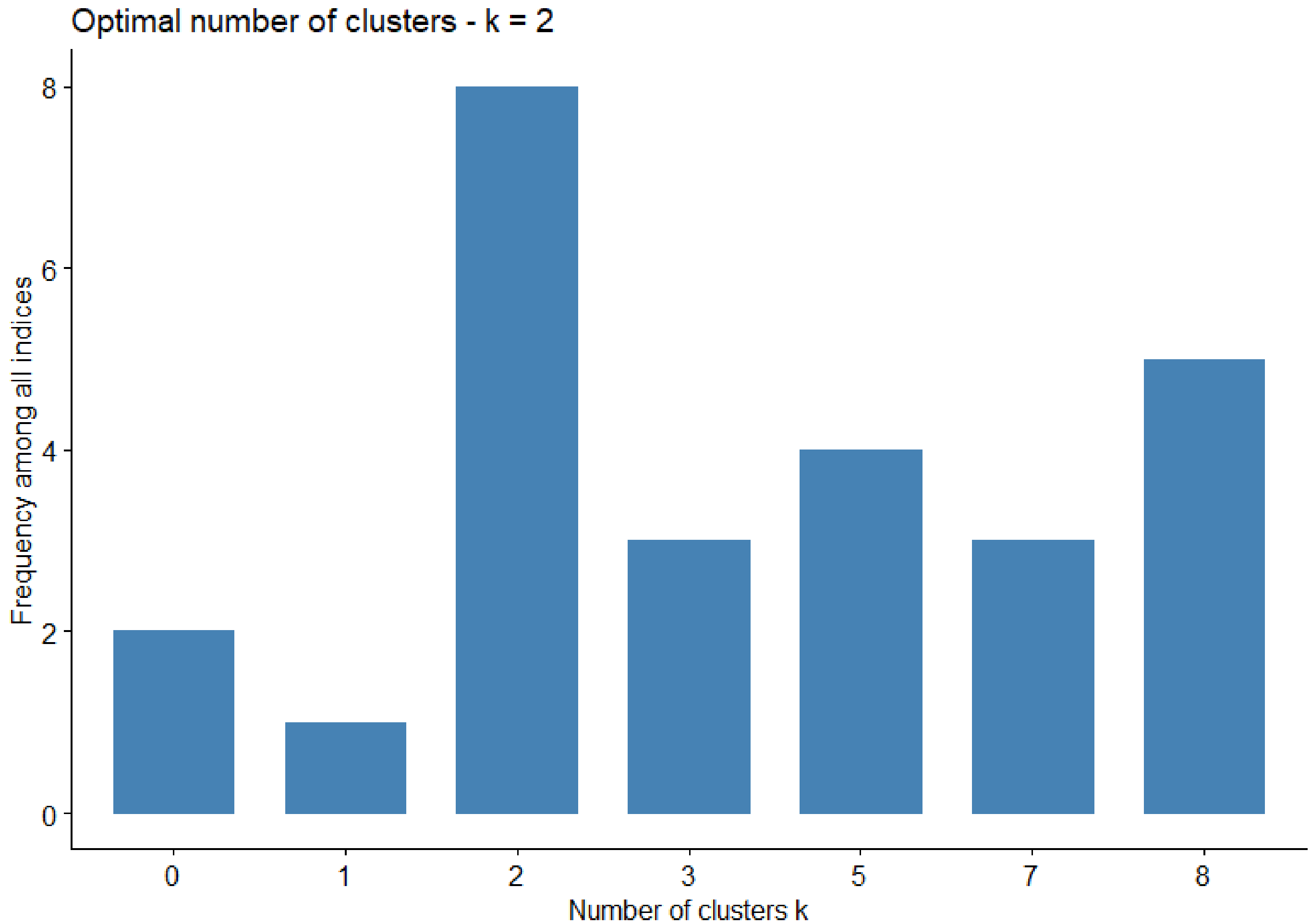
As seen above, 310 of the observations are contained in cluster 1, 125 in cluster 2, and 5 in cluster 3.

MSDS 680 - Machine Learning
Alla Topp



Let’s also create clusters with “single” linkage, hierarchical clustering with the single method to cluster our data. So we perform the same steps, but with different parameters.

```
> #performing hierarchical clustering with "single" linkage
> nb_wholesale2 <- NbClust(hca_wholesale, distance = "euclidean", min.nc = 2,
+                max.nc = 10, method = "single")
*** : 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 Hubert
                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 Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 8 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 4 proposed 5 as the best number of clusters
* 3 proposed 7 as the best number of clusters
* 5 proposed 8 as the best number of clusters

                   ***** Conclusion *****

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

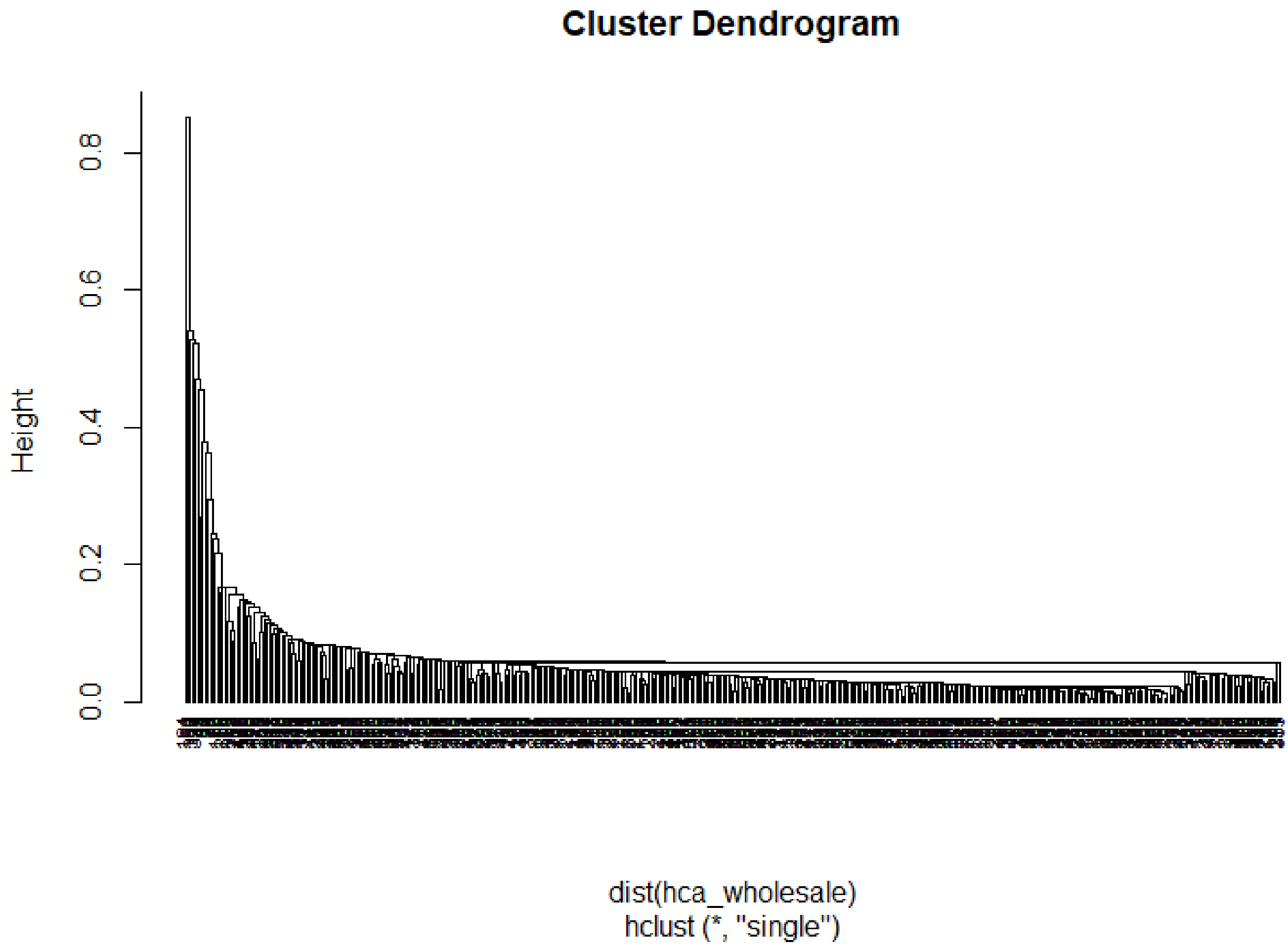


So, this model states that the optimal number of clusters would be 2.

Next, we are going to build the plot based on the last model:

```
#performing hierarchical clustering with "single" linkage
hc_clust2 = hclust(dist(hca_wholesale), method="single")
plot(hc_clust2, hang = -.01, cex = .7)
```

```
> hc_clust2

Call:
hclust(d = dist(hca_wholesale), method = "single")

Cluster method   : single
Distance         : euclidean
Number of objects: 440
```

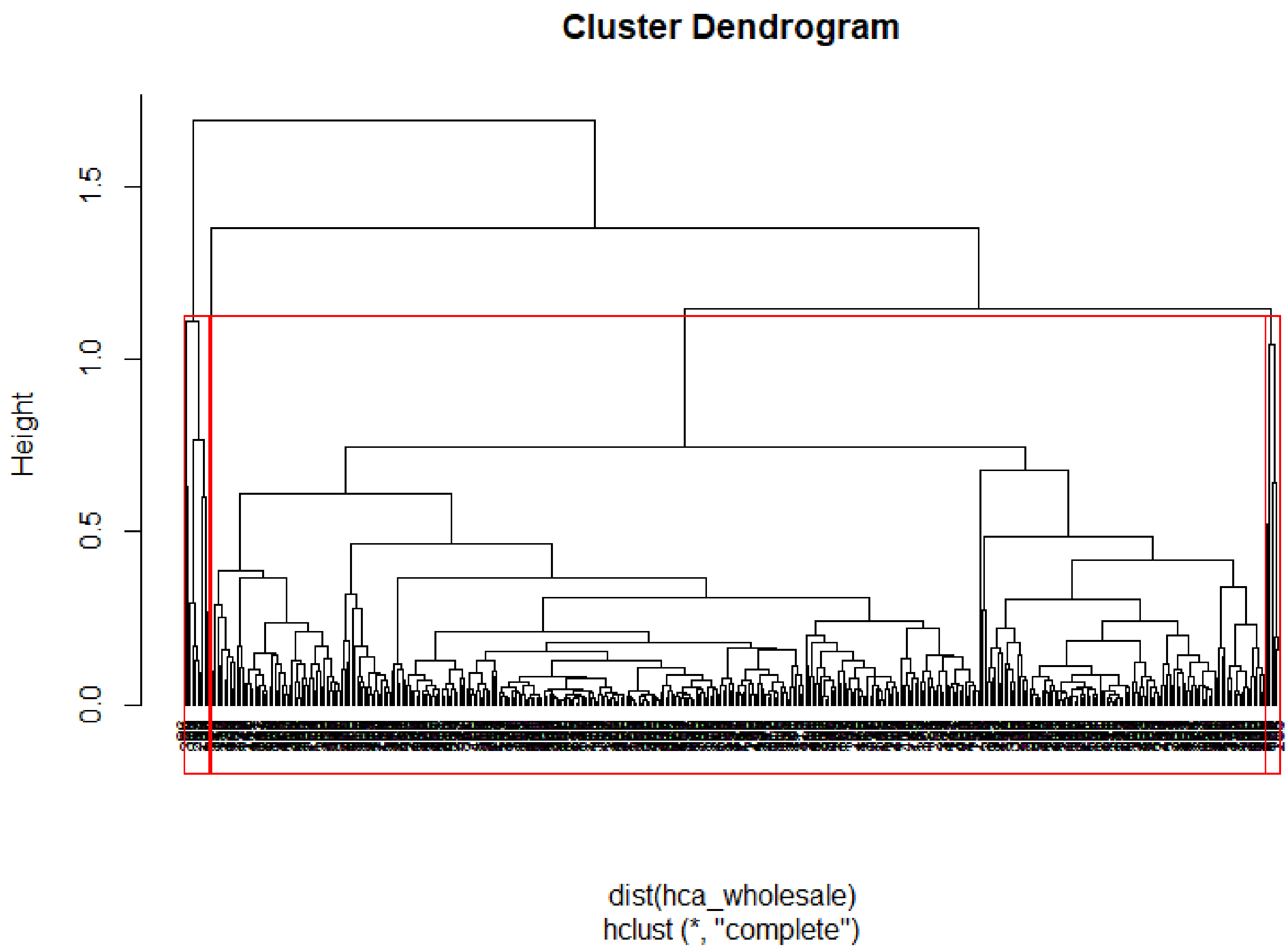


We can see that this demdogram is not possible to read and the single method didn’t do any meaningful clustering that we can work with. So this method is not good for this dataset.

Another linkage we could try would be “complete” and we could see what dendogram could show us.

```
#performing hierarchical clustering with "complete" linkage
hc_clust3 = hclust(dist(hca_wholesale), method="complete")
plot(hc_clust3, hang = -.01, cex = .7)
```

```
> hc_clust3

Call:
hclust(d = dist(hca_wholesale), method = "complete")

Cluster method   : complete
Distance         : euclidean
Number of objects: 440
```



To sum up, both methods k-means and HCA are very interesting and important and based on the analysis of the wholesale data we could state that choosing dataset can affect the results of different clustering methods. For example, our HCA method with single linkage did not work with this dataset probably because of its size. When we ran clustering with “complete” linkage, it doesn’t show us a great result either. I think that k-means clustering makes more sense analyzing data base we have for this assignment. At least we could interpret some results.

References:

Beautiful dendrogram visualizations in R: 5 must known methods - Unsupervised Machine Learning. (n.d.). Retrieved from http://www.sthda.com/english/wiki/beautiful-dendrogram-visualizations-in-r-5-must-known-methods-unsupervised-machine-learning

Hierarchical Cluster Analysis. (n.d.). Retrieved from http://uc-r.github.io/hc_clustering

Sign In. (n.d.). Retrieved from http://rpubs.com/jprakash0205/398634

Yu-Wei, C. (2015). Machine learning with R cookbook explore over 110 recipes to analyze data and build predictive models with the simple and easy-to-use R code. Packt Publishing.

K-means Clustering. (n.d.). Retrieved from http://www.rdatamining.com/examples/kmeans-clustering