Assignment - 5

Vamshee Deepak Goud Katta

11/29/2021

set.seed(123)  
  
library(cluster)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(dendextend)

##   
## ---------------------  
## Welcome to dendextend version 1.15.2  
## Type citation('dendextend') for how to cite the package.  
##   
## Type browseVignettes(package = 'dendextend') for the package vignette.  
## The github page is: https://github.com/talgalili/dendextend/  
##   
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues  
## You may ask questions at stackoverflow, use the r and dendextend tags:   
## https://stackoverflow.com/questions/tagged/dendextend  
##   
## To suppress this message use: suppressPackageStartupMessages(library(dendextend))  
## ---------------------

##   
## Attaching package: 'dendextend'

## The following object is masked from 'package:stats':  
##   
## cutree

library(knitr)  
library(factoextra)

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

### Importing dataset(Cereals)

library(readr)

## Warning: package 'readr' was built under R version 4.1.2

Cereals <- read\_csv("Cereals.csv", col\_types = cols(calories = col\_number(),   
 protein = col\_number(), fat = col\_number(),   
 sodium = col\_number(), fiber = col\_number(),   
 carbo = col\_number(), sugars = col\_number(),   
 potass = col\_number(), vitamins = col\_number(),   
 shelf = col\_number(), weight = col\_number(),   
 cups = col\_number(), rating = col\_number()))  
  
Cereals1 <- data.frame(Cereals[,4:16])

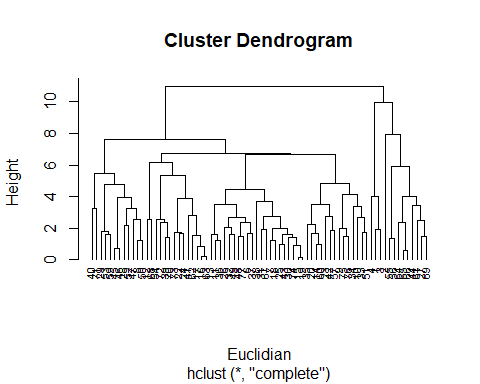
## Pre-processing data and normalizing the data

Cereals1 <- na.omit(Cereals1)  
Cereals2 <- scale(Cereals1)

## Applying hierarchical clustering to the data using Euclidean distance to the normalized measurements.

### Calculating Dissimilarity Matrix and performing Hierarchial Clustering.

Euclidian <- dist(Cereals2, method = "euclidean")  
  
## Clustering the Cereals dataset.  
  
Complete <- hclust(Euclidian, method = "complete")  
  
## Plotting the dendogram.  
  
plot(Complete, cex = 0.7, hang = -1)



### Using Agnes to compare the clustering.

# Single Linkage Method  
Single <- agnes(Cereals2, method = "single")  
# Complete Linkage Method  
Complete1 <- agnes(Cereals2, method = "complete")  
# Average Linkage Method  
Average <- agnes(Cereals2, method = "average")  
# Ward Method  
Ward <- agnes(Cereals2, method = "ward")

### Comparing the agglomerative coefficients.

### Single Linkage vs Complete Linkage vs Average Linkage vs Ward.

print(Single$ac)

## [1] 0.6067859

print(Complete1$ac)

## [1] 0.8353712

print(Average$ac)

## [1] 0.7766075

print(Ward$ac)

## [1] 0.9046042

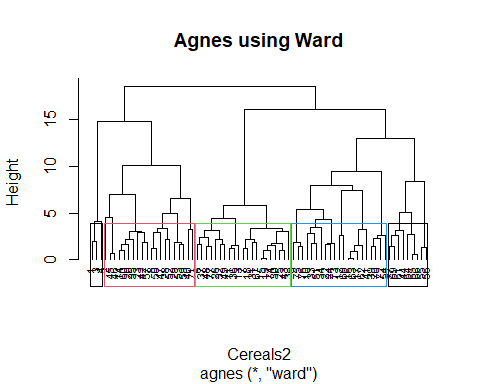
### Here we can see that Ward method is best with highest value of 0.9046042.

### Choosing the clusters:

### Plotting the agnes using ward method and cutting the Dendogram.

### We will take 5 clusters (k = 5) based on the distance.

pltree(Ward, cex = 0.7, hang = -1, main = "Agnes using Ward")  
rect.hclust(Ward, k = 5, border = 1:4)



Cluster <- cutree(Ward, k=5)  
Cluster1 <- as.data.frame(cbind(Cereals2, Cluster))

### Structure of the clusters and their stability.

### Creating Partitions.

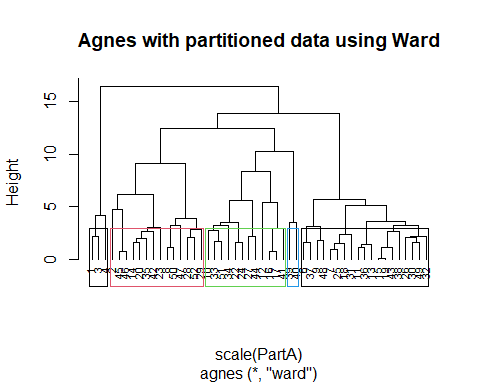
PartA <- Cereals1[1:50,]  
PartB <- Cereals1[51:74,]

### Performing Hierarchial Clustering, plotting and cutting the dendogram with k=5.

# Single Linkage Method of Part A  
Single1 <- agnes(scale(PartA), method = "single")  
# Complete Linkage Method of Part A  
Complete2 <- agnes(scale(PartA), method = "complete")  
# Average Linkage Method of Part A  
Average1 <- agnes(scale(PartA), method = "average")  
# Ward Method of Part A  
Ward1 <- agnes(scale(PartA), method = "ward")  
  
cbind(Single= Single1$ac , Complete=Complete2$ac , Average= Average1$ac , Ward= Ward1$ac)

## Single Complete Average Ward  
## [1,] 0.6393338 0.8138238 0.7408904 0.8764323

pltree(Ward1, cex = 0.7, hang = -1, main = "Agnes with partitioned data using Ward")  
# Clustering Part A of the data set  
rect.hclust(Ward1, k = 5, border = 1:4)



# Cutting the dendrogram  
Cluster2 <- cutree(Ward1, k = 5)

### Calculating the centeroids.

Centroids <- as.data.frame(cbind(PartA, Cluster2))  
Centroids[Centroids$Cluster2==1,]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 1 70 4 1 130 10 5 6 280 25 3 1  
## 3 70 4 1 260 9 7 5 320 25 3 1  
## 4 50 4 0 140 14 8 0 330 25 3 1  
## cups rating Cluster2  
## 1 0.33 68.40297 1  
## 3 0.33 59.42551 1  
## 4 0.50 93.70491 1

# Centroid 1  
centroid1 <- colMeans(Centroids[Centroids$Cluster2==1,])  
Centroids[Centroids$Cluster2==2,]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 2 120 3 5 15 2.0 8.0 8 135 0 3 1.00  
## 8 130 3 2 210 2.0 18.0 8 100 25 3 1.33  
## 14 110 3 2 140 2.0 13.0 7 105 25 3 1.00  
## 20 110 3 3 140 4.0 10.0 7 160 25 3 1.00  
## 23 100 2 1 140 2.0 11.0 10 120 25 3 1.00  
## 28 120 3 2 160 5.0 12.0 10 200 25 3 1.25  
## 29 120 3 0 240 5.0 14.0 12 190 25 3 1.33  
## 35 120 3 3 75 3.0 13.0 4 100 25 3 1.00  
## 42 100 4 2 150 2.0 12.0 6 95 25 2 1.00  
## 45 150 4 3 95 3.0 16.0 11 170 25 3 1.00  
## 46 150 4 3 150 3.0 16.0 11 170 25 3 1.00  
## 47 160 3 2 150 3.0 17.0 13 160 25 3 1.50  
## 50 140 3 2 220 3.0 21.0 7 130 25 3 1.33  
## 52 130 3 2 170 1.5 13.5 10 120 25 3 1.25  
## cups rating Cluster2  
## 2 1.00 33.98368 2  
## 8 0.75 37.03856 2  
## 14 0.50 40.40021 2  
## 20 0.50 40.44877 2  
## 23 0.75 36.17620 2  
## 28 0.67 40.91705 2  
## 29 0.67 41.01549 2  
## 35 0.33 45.81172 2  
## 42 0.67 45.32807 2  
## 45 1.00 37.13686 2  
## 46 1.00 34.13977 2  
## 47 0.67 30.31335 2  
## 50 0.67 40.69232 2  
## 52 0.50 30.45084 2

# Centroid 2  
centroid2 <- colMeans(Centroids[Centroids$Cluster2==2,])  
Centroids[Centroids$Cluster2==3,]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 6 110 2 2 180 1.5 10.5 10 70 25 1 1  
## 7 110 2 0 125 1.0 11.0 14 30 25 2 1  
## 9 90 2 1 200 4.0 15.0 6 125 25 1 1  
## 11 120 1 2 220 0.0 12.0 12 35 25 2 1  
## 13 120 1 3 210 0.0 13.0 9 45 25 2 1  
## 15 110 1 1 180 0.0 12.0 13 55 25 2 1  
## 18 110 1 0 90 1.0 13.0 12 20 25 2 1  
## 19 110 1 1 180 0.0 12.0 13 65 25 2 1  
## 25 110 2 1 125 1.0 11.0 13 30 25 2 1  
## 26 110 1 0 200 1.0 14.0 11 25 25 1 1  
## 30 110 1 1 135 0.0 13.0 12 25 25 2 1  
## 31 100 2 0 45 0.0 11.0 15 40 25 1 1  
## 32 110 1 1 280 0.0 15.0 9 45 25 2 1  
## 36 120 1 2 220 1.0 12.0 11 45 25 2 1  
## 37 110 3 1 250 1.5 11.5 10 90 25 1 1  
## 38 110 1 0 180 0.0 14.0 11 35 25 1 1  
## 43 110 2 1 180 0.0 12.0 12 55 25 2 1  
## 48 100 2 1 220 2.0 15.0 6 90 25 1 1  
## 49 120 2 1 190 0.0 15.0 9 40 25 2 1  
## cups rating Cluster2  
## 6 0.75 29.50954 3  
## 7 1.00 33.17409 3  
## 9 0.67 49.12025 3  
## 11 0.75 18.04285 3  
## 13 0.75 19.82357 3  
## 15 1.00 22.73645 3  
## 18 1.00 35.78279 3  
## 19 1.00 22.39651 3  
## 25 1.00 32.20758 3  
## 26 0.75 31.43597 3  
## 30 0.75 28.02576 3  
## 31 0.88 35.25244 3  
## 32 0.75 23.80404 3  
## 36 1.00 21.87129 3  
## 37 0.75 31.07222 3  
## 38 1.33 28.74241 3  
## 43 1.00 26.73452 3  
## 48 1.00 40.10597 3  
## 49 0.67 29.92429 3

# Centroid 3  
centroid3 <- colMeans(Centroids[Centroids$Cluster2==3,])  
Centroids[Centroids$Cluster2==4,]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 10 90 3 0 210 5 13 5 190 25 3 1  
## 12 110 6 2 290 2 17 1 105 25 1 1  
## 16 110 2 0 280 0 22 3 25 25 1 1  
## 17 100 2 0 290 1 21 2 35 25 1 1  
## 22 110 2 0 220 1 21 3 30 25 3 1  
## 24 100 2 0 190 1 18 5 80 25 3 1  
## 27 100 3 0 0 3 14 7 100 25 2 1  
## 33 100 3 1 140 3 15 5 85 25 3 1  
## 34 110 3 0 170 3 17 3 90 25 3 1  
## 41 110 2 1 260 0 21 3 40 25 2 1  
## 44 100 4 1 0 0 16 3 95 25 2 1  
## 51 90 3 0 170 3 18 2 90 25 3 1  
## cups rating Cluster2  
## 10 0.67 53.31381 4  
## 12 1.25 50.76500 4  
## 16 1.00 41.44502 4  
## 17 1.00 45.86332 4  
## 22 1.00 46.89564 4  
## 24 0.75 44.33086 4  
## 27 0.80 58.34514 4  
## 33 0.88 52.07690 4  
## 34 0.25 53.37101 4  
## 41 1.50 39.24111 4  
## 44 1.00 54.85092 4  
## 51 1.00 59.64284 4

# Centroid 4  
centroid4 <- colMeans(Centroids[Centroids$Cluster2==4,])  
Centroids1 <- rbind(centroid1, centroid2, centroid3, centroid4)  
Centroids2 <- as.data.frame(rbind(Centroids[,-14], PartB))

### Calculating the Distance.

Distance <- get\_dist(Centroids2)  
Matrix <- as.matrix(Distance)  
Distance1 <- data.frame(data=seq(1,nrow(PartB),1), Clusters = rep(0,nrow(PartB)))  
for(i in 1:nrow(PartB))   
 {Distance1[i,2] <- which.min(Matrix[i+4, 1:4])}  
Distance1

## data Clusters  
## 1 1 2  
## 2 2 2  
## 3 3 2  
## 4 4 1  
## 5 5 1  
## 6 6 2  
## 7 7 3  
## 8 8 2  
## 9 9 2  
## 10 10 2  
## 11 11 2  
## 12 12 3  
## 13 13 2  
## 14 14 2  
## 15 15 1  
## 16 16 2  
## 17 17 2  
## 18 18 2  
## 19 19 2  
## 20 20 2  
## 21 21 2  
## 22 22 1  
## 23 23 3  
## 24 24 2

cbind(Cluster1$Cluster[51:74], Distance1$Clusters)

## [,1] [,2]  
## [1,] 2 2  
## [2,] 4 2  
## [3,] 5 2  
## [4,] 5 1  
## [5,] 2 1  
## [6,] 2 2  
## [7,] 2 3  
## [8,] 5 2  
## [9,] 4 2  
## [10,] 4 2  
## [11,] 5 2  
## [12,] 5 3  
## [13,] 5 2  
## [14,] 3 2  
## [15,] 4 1  
## [16,] 5 2  
## [17,] 4 2  
## [18,] 2 2  
## [19,] 4 2  
## [20,] 4 2  
## [21,] 3 2  
## [22,] 4 1  
## [23,] 4 3  
## [24,] 3 2

# Tabulating the results  
table(Cluster1$Cluster[51:74] == Distance1$Clusters)

##   
## FALSE TRUE   
## 12 12

### We are getting 12 FALSE and 12 TRUE, so we can conclude that the model is partially stable.

### Finding a cluster of “healthy cereals.”

### Clustering Healthy Cereals

Healthy <- Cereals  
Healthy <- na.omit(Healthy)  
Healthy1 <- cbind(Healthy, Cluster)  
Healthy1[Healthy1$Cluster==1,]  
Healthy1[Healthy1$Cluster==2,]  
Healthy1[Healthy1$Cluster==3,]  
Healthy1[Healthy1$Cluster==4,]

#### Mean ratings to determine the best cluster.

mean(Healthy1[Healthy1$Cluster==1,"rating"])

## [1] 73.84446

mean(Healthy1[Healthy1$Cluster==2,"rating"])

## [1] 38.26161

mean(Healthy1[Healthy1$Cluster==3,"rating"])

## [1] 28.84825

mean(Healthy1[Healthy1$Cluster==4,"rating"])

## [1] 46.46513

### As we can se that the mean rating of the cluster 1 is the highest(i.e. 73.84446), we will choose cluster 1.