Assignment 5

Rob DiVincenzo 11-30-2023 BA 64060

This is the submission for Assignment 5.

#Read data  
DF=read.csv("./Cereals.csv") # Read the Cereals csv file  
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(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(factoextra)

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

#Load libraries  
#library(ISLR)  
#library(tidyverse) # data manipulation  
#library(factoextra) # clustering algorithms & visualization  
#library(flexclust)  
  
set.seed(123)

# Data Preprocessing  
#Select only the 9 numerical variables  
DF <- select(DF, calories, protein, fat, sodium, fiber, carbo, sugars, potass, vitamins, shelf, weight, cups, rating)  
  
cleaned\_data <- na.omit(DF) #omit any rows with na column values  
  
normalized\_data <- scale(cleaned\_data) #normalize data

# 1) Ward is the best result  
  
# Compute with agnes and with different linkage methods  
hc\_single <- agnes(normalized\_data, method="single")  
hc\_complete <- agnes(normalized\_data, method="complete")  
hc\_average <- agnes(normalized\_data, method="average")  
hc\_ward <- agnes(normalized\_data, method="ward")  
  
print(hc\_single$ac)

## [1] 0.6067859

print(hc\_complete$ac)

## [1] 0.8353712

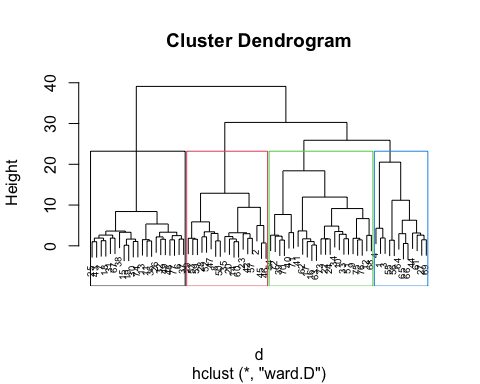
print(hc\_average$ac)

## [1] 0.7766075

print(hc\_ward$ac) # best result

## [1] 0.9046042

# 2) I would choose 4 clusters for our dendrogram since clusters in k=3 seemed too large and clusters in k=5 resulted in too narrow a cluster  
d <- dist(normalized\_data, method = "euclidean")  
  
hc\_ward2 <- hclust(d, method = "ward.D")  
  
#plot the denrogram  
plot(hc\_ward2, cex = 0.6)  
rect.hclust(hc\_ward2, k=4, border = 1:4)



#3) # It appears that most observations remained in their original clusters, indicating a good level of stability  
  
#3a)  
  
#Combine the cluster you obtained in the first question to the data, and then make a partition A and B (you could use 70% of data as partition A).  
  
# Extract cluster assignments  
ward\_clusters <- cutree(hc\_ward, k = 4)  
# Combine cluster assignments with normalized data using data.frame without converting to factors  
data\_with\_clusters <- data.frame(normalized\_data, Ward\_Cluster = ward\_clusters)  
  
# Make partition A (70%) and partition B (30%)  
index\_partition\_A <- createDataPartition(data\_with\_clusters$Ward\_Cluster, p = 0.7, list = FALSE)  
partition\_A <- data\_with\_clusters[index\_partition\_A, ]  
partition\_B <- data\_with\_clusters[-index\_partition\_A, ]  
  
#3b)  
#Compute the mean/average for each numerical variable in each cluster in partition A (so you obtain the centroid for A).  
  
# Assuming your numerical variables are columns 1 to n in 'partition\_A'  
centroid\_A <- partition\_A %>%  
 group\_by(Ward\_Cluster) %>%  
 summarize(across(everything(), mean))  
  
#3c)  
  
# Exclude Ward Cluster number  
numeric\_columns\_B <- partition\_B[, -14]  
  
# Initialize a vector to store new cluster numbers in partition B  
new\_clusters\_B <- integer(nrow(partition\_B))  
  
# Compute Euclidean distance between observations in partition B and centroids in A using a for loop  
for (i in 1:nrow(partition\_B)) {  
 observation <- as.matrix(numeric\_columns\_B[i, , drop = FALSE])  
 distances <- apply(centroid\_A[, -1], 1, function(centroid) sqrt(sum((observation - centroid)^2)))  
 new\_clusters\_B[i] <- which.min(distances)  
}  
  
# Create a new data frame with the original columns and the new cluster assignments  
partition\_B\_new <- data.frame(partition\_B, New\_Cluster = new\_clusters\_B)  
  
#3d)  
#Compare the difference between the old and new cluster numbers in partition B to check its stability.  
  
# Check stability by comparing old and new cluster assignments  
stability\_comparison <- table(partition\_B\_new$Ward\_Cluster, partition\_B\_new$New\_Cluster)  
  
# Add labels and show a clearer stability comparison  
stability\_labels <- c("Cluster 2 (old)", "Cluster 3 (old)", "Cluster 4 (old)")  
rownames(stability\_comparison) <- colnames(stability\_comparison) <- stability\_labels  
  
print(stability\_comparison)

##   
## Cluster 2 (old) Cluster 3 (old) Cluster 4 (old)  
## Cluster 2 (old) 5 1 0  
## Cluster 3 (old) 0 6 0  
## Cluster 4 (old) 0 0 9

# It appears that most observations remained in their original clusters, indicating a good level of stability

#4) Cluster 1 is the healthiest cluster. We absolutely need to use normalized data here because we are going to sum the healthy factors and subtract the sum of the unhealthy factors. And an unnormalized factor like calories (50-140) will vastly outweigh something like grams of fat (1-5)  
  
  
# Define the criteria for "Healthy Cereals" (healthy\_factors and unhealthy factors)  
healthy\_factors <- c("protein", "fiber", "potass", "vitamins") # high is good  
unhealthy\_factors <- c("calories", "fat", "carbo", "sugars") # high is bad  
  
# Obtain the cluster centroids  
centroids <- data\_with\_clusters %>%  
 group\_by(Ward\_Cluster) %>%  
 summarize(across(healthy\_factors, mean), across(unhealthy\_factors, mean))

## Warning: There were 2 warnings in `summarize()`.  
## The first warning was:  
## ℹ In argument: `across(healthy\_factors, mean)`.  
## Caused by warning:  
## ! Using an external vector in selections was deprecated in tidyselect 1.1.0.  
## ℹ Please use `all\_of()` or `any\_of()` instead.  
## # Was:  
## data %>% select(healthy\_factors)  
##   
## # Now:  
## data %>% select(all\_of(healthy\_factors))  
##   
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.  
## ℹ Run `dplyr::last\_dplyr\_warnings()` to see the 1 remaining warning.

# Calculate the "health score" score for each cluster by summing up the healthy columns and subtracting the sum of the unhealthy columns.  
cluster\_scores <- rowSums(centroids[, -1][, 1:4] - centroids[, -1][, 5:8])  
  
# Identify the healthiest Ward Cluster, the one with the highest health score.  
healthiest\_cluster <- centroids$Ward\_Cluster[which.max(cluster\_scores)]  
  
print(healthiest\_cluster)

## [1] 1

#Cluster 1 is the healthiest cluster.