Assignment\_5

Abhinav

2024-04-07

## R Markdown

#Loading the required libraries as they are pre-installed  
library(stats)  
library(cluster)  
library(factoextra)

## Warning: package 'factoextra' was built under R version 4.3.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.3

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

library(dendextend)

## Warning: package 'dendextend' was built under R version 4.3.3

##   
## ---------------------  
## Welcome to dendextend version 1.17.1  
## 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(caret)

## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: lattice

df <- read.csv("Cereals.csv")  
df\_2 <- na.omit(df) #Remove NA(missing) values  
df<- data.frame(df[,4:16])  
head(df)

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 1 70 4 1 130 10.0 5.0 6 280 25 3 1  
## 2 120 3 5 15 2.0 8.0 8 135 0 3 1  
## 3 70 4 1 260 9.0 7.0 5 320 25 3 1  
## 4 50 4 0 140 14.0 8.0 0 330 25 3 1  
## 5 110 2 2 200 1.0 14.0 8 NA 25 3 1  
## 6 110 2 2 180 1.5 10.5 10 70 25 1 1  
## cups rating  
## 1 0.33 68.40297  
## 2 1.00 33.98368  
## 3 0.33 59.42551  
## 4 0.50 93.70491  
## 5 0.75 34.38484  
## 6 0.75 29.50954

#Scaling the data to normalize

df\_normalized <- scale(df)  
head(df)

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 1 70 4 1 130 10.0 5.0 6 280 25 3 1  
## 2 120 3 5 15 2.0 8.0 8 135 0 3 1  
## 3 70 4 1 260 9.0 7.0 5 320 25 3 1  
## 4 50 4 0 140 14.0 8.0 0 330 25 3 1  
## 5 110 2 2 200 1.0 14.0 8 NA 25 3 1  
## 6 110 2 2 180 1.5 10.5 10 70 25 1 1  
## cups rating  
## 1 0.33 68.40297  
## 2 1.00 33.98368  
## 3 0.33 59.42551  
## 4 0.50 93.70491  
## 5 0.75 34.38484  
## 6 0.75 29.50954

#Dissimilarity matrix

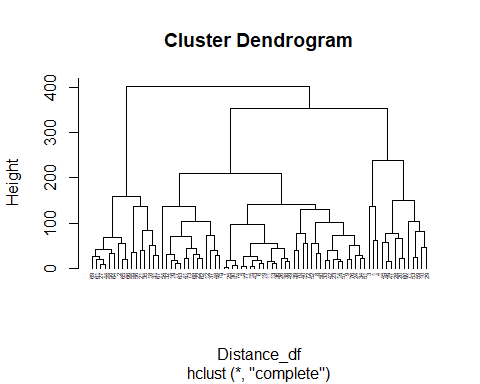
Distance\_df <- dist(df, method = "euclidean")

#Hierarchical clustering using complete linkage

hc1 <- hclust(Distance\_df, method = "complete")

#Plotting the obtained dendrogram

plot(hc1, cex = 0.4, hang = -3)



#Using agnes() function is pretty similar to using hclust() but with the agnes() function we will also be able to get the agglomerative coefficient.

data <- df  
data

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 1 70 4 1 130 10.0 5.0 6 280 25 3 1.00  
## 2 120 3 5 15 2.0 8.0 8 135 0 3 1.00  
## 3 70 4 1 260 9.0 7.0 5 320 25 3 1.00  
## 4 50 4 0 140 14.0 8.0 0 330 25 3 1.00  
## 5 110 2 2 200 1.0 14.0 8 NA 25 3 1.00  
## 6 110 2 2 180 1.5 10.5 10 70 25 1 1.00  
## 7 110 2 0 125 1.0 11.0 14 30 25 2 1.00  
## 8 130 3 2 210 2.0 18.0 8 100 25 3 1.33  
## 9 90 2 1 200 4.0 15.0 6 125 25 1 1.00  
## 10 90 3 0 210 5.0 13.0 5 190 25 3 1.00  
## 11 120 1 2 220 0.0 12.0 12 35 25 2 1.00  
## 12 110 6 2 290 2.0 17.0 1 105 25 1 1.00  
## 13 120 1 3 210 0.0 13.0 9 45 25 2 1.00  
## 14 110 3 2 140 2.0 13.0 7 105 25 3 1.00  
## 15 110 1 1 180 0.0 12.0 13 55 25 2 1.00  
## 16 110 2 0 280 0.0 22.0 3 25 25 1 1.00  
## 17 100 2 0 290 1.0 21.0 2 35 25 1 1.00  
## 18 110 1 0 90 1.0 13.0 12 20 25 2 1.00  
## 19 110 1 1 180 0.0 12.0 13 65 25 2 1.00  
## 20 110 3 3 140 4.0 10.0 7 160 25 3 1.00  
## 21 100 3 0 80 1.0 21.0 0 NA 0 2 1.00  
## 22 110 2 0 220 1.0 21.0 3 30 25 3 1.00  
## 23 100 2 1 140 2.0 11.0 10 120 25 3 1.00  
## 24 100 2 0 190 1.0 18.0 5 80 25 3 1.00  
## 25 110 2 1 125 1.0 11.0 13 30 25 2 1.00  
## 26 110 1 0 200 1.0 14.0 11 25 25 1 1.00  
## 27 100 3 0 0 3.0 14.0 7 100 25 2 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  
## 30 110 1 1 135 0.0 13.0 12 25 25 2 1.00  
## 31 100 2 0 45 0.0 11.0 15 40 25 1 1.00  
## 32 110 1 1 280 0.0 15.0 9 45 25 2 1.00  
## 33 100 3 1 140 3.0 15.0 5 85 25 3 1.00  
## 34 110 3 0 170 3.0 17.0 3 90 25 3 1.00  
## 35 120 3 3 75 3.0 13.0 4 100 25 3 1.00  
## 36 120 1 2 220 1.0 12.0 11 45 25 2 1.00  
## 37 110 3 1 250 1.5 11.5 10 90 25 1 1.00  
## 38 110 1 0 180 0.0 14.0 11 35 25 1 1.00  
## 39 110 2 1 170 1.0 17.0 6 60 100 3 1.00  
## 40 140 3 1 170 2.0 20.0 9 95 100 3 1.30  
## 41 110 2 1 260 0.0 21.0 3 40 25 2 1.00  
## 42 100 4 2 150 2.0 12.0 6 95 25 2 1.00  
## 43 110 2 1 180 0.0 12.0 12 55 25 2 1.00  
## 44 100 4 1 0 0.0 16.0 3 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  
## 48 100 2 1 220 2.0 15.0 6 90 25 1 1.00  
## 49 120 2 1 190 0.0 15.0 9 40 25 2 1.00  
## 50 140 3 2 220 3.0 21.0 7 130 25 3 1.33  
## 51 90 3 0 170 3.0 18.0 2 90 25 3 1.00  
## 52 130 3 2 170 1.5 13.5 10 120 25 3 1.25  
## 53 120 3 1 200 6.0 11.0 14 260 25 3 1.33  
## 54 100 3 0 320 1.0 20.0 3 45 100 3 1.00  
## 55 50 1 0 0 0.0 13.0 0 15 0 3 0.50  
## 56 50 2 0 0 1.0 10.0 0 50 0 3 0.50  
## 57 100 4 1 135 2.0 14.0 6 110 25 3 1.00  
## 58 100 5 2 0 2.7 NA NA 110 0 1 1.00  
## 59 120 3 1 210 5.0 14.0 12 240 25 2 1.33  
## 60 100 3 2 140 2.5 10.5 8 140 25 3 1.00  
## 61 90 2 0 0 2.0 15.0 6 110 25 3 1.00  
## 62 110 1 0 240 0.0 23.0 2 30 25 1 1.00  
## 63 110 2 0 290 0.0 22.0 3 35 25 1 1.00  
## 64 80 2 0 0 3.0 16.0 0 95 0 1 0.83  
## 65 90 3 0 0 4.0 19.0 0 140 0 1 1.00  
## 66 90 3 0 0 3.0 20.0 0 120 0 1 1.00  
## 67 110 2 1 70 1.0 9.0 15 40 25 2 1.00  
## 68 110 6 0 230 1.0 16.0 3 55 25 1 1.00  
## 69 90 2 0 15 3.0 15.0 5 90 25 2 1.00  
## 70 110 2 1 200 0.0 21.0 3 35 100 3 1.00  
## 71 140 3 1 190 4.0 15.0 14 230 100 3 1.50  
## 72 100 3 1 200 3.0 16.0 3 110 100 3 1.00  
## 73 110 2 1 250 0.0 21.0 3 60 25 3 1.00  
## 74 110 1 1 140 0.0 13.0 12 25 25 2 1.00  
## 75 100 3 1 230 3.0 17.0 3 115 25 1 1.00  
## 76 100 3 1 200 3.0 17.0 3 110 25 1 1.00  
## 77 110 2 1 200 1.0 16.0 8 60 25 1 1.00  
## cups rating  
## 1 0.33 68.40297  
## 2 1.00 33.98368  
## 3 0.33 59.42551  
## 4 0.50 93.70491  
## 5 0.75 34.38484  
## 6 0.75 29.50954  
## 7 1.00 33.17409  
## 8 0.75 37.03856  
## 9 0.67 49.12025  
## 10 0.67 53.31381  
## 11 0.75 18.04285  
## 12 1.25 50.76500  
## 13 0.75 19.82357  
## 14 0.50 40.40021  
## 15 1.00 22.73645  
## 16 1.00 41.44502  
## 17 1.00 45.86332  
## 18 1.00 35.78279  
## 19 1.00 22.39651  
## 20 0.50 40.44877  
## 21 1.00 64.53382  
## 22 1.00 46.89564  
## 23 0.75 36.17620  
## 24 0.75 44.33086  
## 25 1.00 32.20758  
## 26 0.75 31.43597  
## 27 0.80 58.34514  
## 28 0.67 40.91705  
## 29 0.67 41.01549  
## 30 0.75 28.02576  
## 31 0.88 35.25244  
## 32 0.75 23.80404  
## 33 0.88 52.07690  
## 34 0.25 53.37101  
## 35 0.33 45.81172  
## 36 1.00 21.87129  
## 37 0.75 31.07222  
## 38 1.33 28.74241  
## 39 1.00 36.52368  
## 40 0.75 36.47151  
## 41 1.50 39.24111  
## 42 0.67 45.32807  
## 43 1.00 26.73451  
## 44 1.00 54.85092  
## 45 1.00 37.13686  
## 46 1.00 34.13976  
## 47 0.67 30.31335  
## 48 1.00 40.10596  
## 49 0.67 29.92429  
## 50 0.67 40.69232  
## 51 1.00 59.64284  
## 52 0.50 30.45084  
## 53 0.67 37.84059  
## 54 1.00 41.50354  
## 55 1.00 60.75611  
## 56 1.00 63.00565  
## 57 0.50 49.51187  
## 58 0.67 50.82839  
## 59 0.75 39.25920  
## 60 0.50 39.70340  
## 61 0.50 55.33314  
## 62 1.13 41.99893  
## 63 1.00 40.56016  
## 64 1.00 68.23588  
## 65 0.67 74.47295  
## 66 0.67 72.80179  
## 67 0.75 31.23005  
## 68 1.00 53.13132  
## 69 1.00 59.36399  
## 70 1.00 38.83975  
## 71 1.00 28.59278  
## 72 1.00 46.65884  
## 73 0.75 39.10617  
## 74 1.00 27.75330  
## 75 0.67 49.78744  
## 76 1.00 51.59219  
## 77 0.75 36.18756

#Performing hcluster using three methods : single, complete and average

hclust\_single <- agnes( data, method = "single")  
hclust\_complete <- agnes(data, method = "complete")  
hclust\_average <- agnes(data, method = "average")  
hclust\_ward <- agnes(data, method = "ward")

#Printing agglomerative coefficients

print( hclust\_single$ac)

## [1] 0.6902217

print( hclust\_average$ac)

## [1] 0.8822055

print( hclust\_complete$ac)

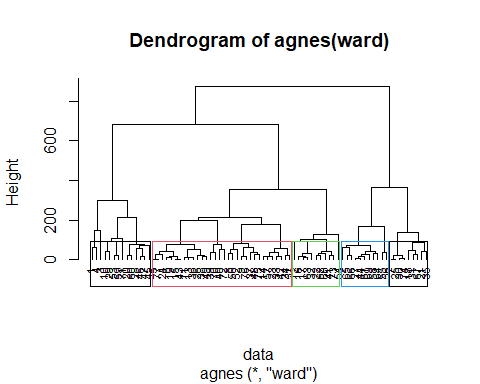
## [1] 0.9256151

print( hclust\_ward)

## Call: agnes(x = data, method = "ward")   
## Agglomerative coefficient: 0.9652554   
## Order of objects:  
## [1] 1 4 3 10 29 53 59 71 20 60 28 46 47 45 5 77 24 6 19 15 43 11 13 36 26  
## [26] 38 49 39 70 40 72 8 50 52 9 76 37 48 75 14 57 23 33 42 34 51 12 16 17 63  
## [51] 32 22 62 68 41 73 54 2 65 66 27 44 61 69 58 64 55 56 7 25 30 74 18 31 67  
## [76] 21 35  
## Height (summary):  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.713 20.503 39.847 84.528 79.624 873.887   
##   
## Available components:  
## [1] "order" "height" "ac" "merge" "diss" "call" "method" "data"

#Single Linkage Method: Agglomerative coefficient = 0.7311616 #Average Linkage Method: Agglomerative coefficient = 0.8792621 #Complete Linkage Method: Agglomerative coefficient = 0.922957 #Ward’s Method: Agglomerative coefficient = 0.9597071 #Agglomerative coefficient measures the cohesion within clusters. Higher values indicate better clustering structures, with clusters being more internally homogeneous. #From the above output the best value we got is 0.904. Plotting the agnes using ward method #and cutting the Dendrogram. we will take k =4 by noticing the distance.

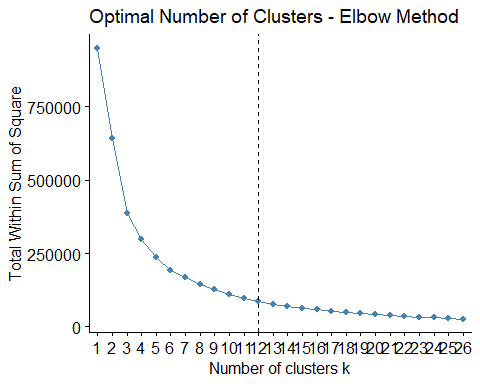
pltree(hclust\_ward,cex = 0.7, hang = -2, main = "Dendrogram of agnes(ward)")  
rect.hclust(hclust\_ward, k = 5,border = 1:4 )



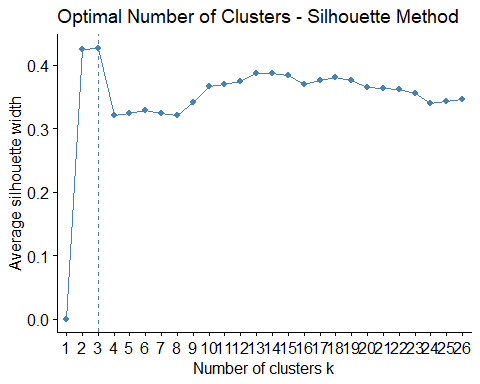
cluster\_a <- cutree(hclust\_ward, k= 5)  
dataframe\_a <- as.data.frame(cbind(df\_normalized, cluster\_a))

#Task-B

fviz\_nbclust(df\_2[, c(4:16)], hcut, method = "wss", k.max = 26) +  
 labs(title = "Optimal Number of Clusters - Elbow Method") +  
 geom\_vline(xintercept = 12, linetype = 2)



fviz\_nbclust(df\_2[, c(4:16)], hcut, method = "silhouette", k.max = 26) +  
 labs(title = "Optimal Number of Clusters - Silhouette Method")

 #Based on the agreement of the silhouette and elbow method, the appropriate number of clusters would be 3 in this case. #Below we will outline the 3 clusters on the hierarchical tree

#Creating pratitions in the dataset

set.seed(123)  
Partition\_a <- df[1:50,]  
Partition\_b <- df[51:78,]

#Scaling the partitioned above data

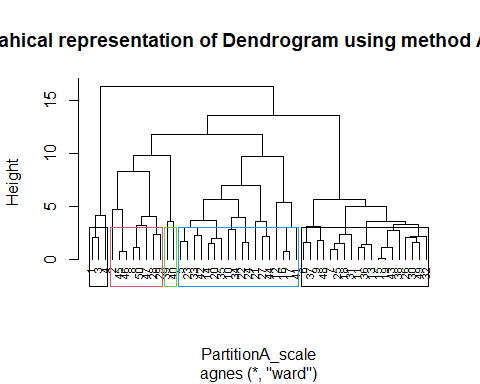
PartitionA\_scale <- scale(Partition\_a)  
PartitionB\_scale <- scale(Partition\_b)

#Now, applying hcluster using agnes to the normalized partitioned datasets

agnes\_single <- agnes(PartitionA\_scale, method = "single" )  
agnes\_average <- agnes(PartitionA\_scale, method = "average" )  
agnes\_complete <- agnes(PartitionA\_scale, method = "complete" )  
agnes\_ward <- agnes(PartitionA\_scale, method = "ward" )  
  
cbind( SINGLE = agnes\_single, AVERAGE = agnes\_average, COMPLETE = agnes\_complete, WARD = agnes\_ward)

## SINGLE AVERAGE COMPLETE   
## order integer,50 integer,50 integer,50   
## height numeric,49 numeric,49 numeric,49   
## ac 0.623718 0.7412724 0.8129231   
## merge integer,98 integer,98 integer,98   
## diss dissimilarity,1225 dissimilarity,1225 dissimilarity,1225  
## call expression expression expression   
## method "single" "average" "complete"   
## order.lab character,50 character,50 character,50   
## data numeric,650 numeric,650 numeric,650   
## WARD   
## order integer,50   
## height numeric,49   
## ac 0.8764152   
## merge integer,98   
## diss dissimilarity,1225  
## call expression   
## method "ward"   
## order.lab character,50   
## data numeric,650

pltree(agnes\_ward, cex = 0.7, hang = -1 , main = "Grahical representation of Dendrogram using method AGNES")  
rect.hclust(agnes\_ward, k = 5, border = 1:4)



cut\_2 <- cutree(agnes\_ward, k = 5)

#Calculating the centeroids  
  
result <- as.data.frame(cbind(Partition\_a, cut\_2))  
result[result$cut\_2==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 cut\_2  
## 1 0.33 68.40297 1  
## 3 0.33 59.42551 1  
## 4 0.50 93.70491 1

centroid\_1 <- colMeans(result[result$cut\_2==1,])  
result[result$cut\_2==2,]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 2 120 3 5 15 2 8 8 135 0 3 1.00  
## 8 130 3 2 210 2 18 8 100 25 3 1.33  
## 28 120 3 2 160 5 12 10 200 25 3 1.25  
## 29 120 3 0 240 5 14 12 190 25 3 1.33  
## 45 150 4 3 95 3 16 11 170 25 3 1.00  
## 46 150 4 3 150 3 16 11 170 25 3 1.00  
## 47 160 3 2 150 3 17 13 160 25 3 1.50  
## 50 140 3 2 220 3 21 7 130 25 3 1.33  
## cups rating cut\_2  
## 2 1.00 33.98368 2  
## 8 0.75 37.03856 2  
## 28 0.67 40.91705 2  
## 29 0.67 41.01549 2  
## 45 1.00 37.13686 2  
## 46 1.00 34.13976 2  
## 47 0.67 30.31335 2  
## 50 0.67 40.69232 2

centroid\_2 <- colMeans(result[result$cut\_2==2,])  
result[result$cut\_2==3,]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 5 110 2 2 200 1 14 8 NA 25 3 1  
## 10 90 3 0 210 5 13 5 190 25 3 1  
## 12 110 6 2 290 2 17 1 105 25 1 1  
## 14 110 3 2 140 2 13 7 105 25 3 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  
## 20 110 3 3 140 4 10 7 160 25 3 1  
## 21 100 3 0 80 1 21 0 NA 0 2 1  
## 22 110 2 0 220 1 21 3 30 25 3 1  
## 23 100 2 1 140 2 11 10 120 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  
## 35 120 3 3 75 3 13 4 100 25 3 1  
## 41 110 2 1 260 0 21 3 40 25 2 1  
## 42 100 4 2 150 2 12 6 95 25 2 1  
## 44 100 4 1 0 0 16 3 95 25 2 1  
## cups rating cut\_2  
## 5 0.75 34.38484 3  
## 10 0.67 53.31381 3  
## 12 1.25 50.76500 3  
## 14 0.50 40.40021 3  
## 16 1.00 41.44502 3  
## 17 1.00 45.86332 3  
## 20 0.50 40.44877 3  
## 21 1.00 64.53382 3  
## 22 1.00 46.89564 3  
## 23 0.75 36.17620 3  
## 24 0.75 44.33086 3  
## 27 0.80 58.34514 3  
## 33 0.88 52.07690 3  
## 34 0.25 53.37101 3  
## 35 0.33 45.81172 3  
## 41 1.50 39.24111 3  
## 42 0.67 45.32807 3  
## 44 1.00 54.85092 3

centroid\_3 <- colMeans(result[result$cut\_2==3,])  
result[result$cut\_2==4,]

## 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 cut\_2  
## 6 0.75 29.50954 4  
## 7 1.00 33.17409 4  
## 9 0.67 49.12025 4  
## 11 0.75 18.04285 4  
## 13 0.75 19.82357 4  
## 15 1.00 22.73645 4  
## 18 1.00 35.78279 4  
## 19 1.00 22.39651 4  
## 25 1.00 32.20758 4  
## 26 0.75 31.43597 4  
## 30 0.75 28.02576 4  
## 31 0.88 35.25244 4  
## 32 0.75 23.80404 4  
## 36 1.00 21.87129 4  
## 37 0.75 31.07222 4  
## 38 1.33 28.74241 4  
## 43 1.00 26.73451 4  
## 48 1.00 40.10596 4  
## 49 0.67 29.92429 4

centroid\_4 <- colMeans(result[result$cut\_2==4,])

centroids <- rbind(centroid\_1, centroid\_2, centroid\_3, centroid\_4)  
x2 <- as.data.frame(rbind(centroids[,-14], Partition\_b))

# Calculating the Distance  
Distance\_1 <- get\_dist(x2)  
Matrix\_1 <- as.matrix(Distance\_1)  
  
dataframe1 <- data.frame(data = seq(1, nrow(Partition\_b)), Clusters = rep(0, nrow(Partition\_b)))  
  
for (i in 1:nrow(Partition\_b)) {  
 # Check if there are non-empty elements in the subset  
 min\_index <- which.min(Matrix\_1[i + 4, 1:4])  
 if (length(min\_index) > 0) {  
 dataframe1$Clusters[i] <- min\_index  
 } else {  
 # If subset is empty, assign NA or any other appropriate value  
 dataframe1$Clusters[i] <- NA  
 }  
}  
  
dataframe1

## data Clusters  
## 1 1 3  
## 2 2 3  
## 3 3 3  
## 4 4 4  
## 5 5 3  
## 6 6 3  
## 7 7 3  
## 8 8 2  
## 9 9 3  
## 10 10 3  
## 11 11 2  
## 12 12 4  
## 13 13 4  
## 14 14 3  
## 15 15 2  
## 16 16 2  
## 17 17 3  
## 18 18 4  
## 19 19 3  
## 20 20 4  
## 21 21 3  
## 22 22 3  
## 23 23 4  
## 24 24 3  
## 25 25 3  
## 26 26 3  
## 27 27 4  
## 28 28 NA

cbind(dataframe\_a$Cluster1[51:74], dataframe1$Clusters)

## [,1]  
## [1,] 3  
## [2,] 3  
## [3,] 3  
## [4,] 4  
## [5,] 3  
## [6,] 3  
## [7,] 3  
## [8,] 2  
## [9,] 3  
## [10,] 3  
## [11,] 2  
## [12,] 4  
## [13,] 4  
## [14,] 3  
## [15,] 2  
## [16,] 2  
## [17,] 3  
## [18,] 4  
## [19,] 3  
## [20,] 4  
## [21,] 3  
## [22,] 3  
## [23,] 4  
## [24,] 3  
## [25,] 3  
## [26,] 3  
## [27,] 4  
## [28,] NA

table(dataframe\_a$Cluster1[51:74] == dataframe1$Clusters)

## < table of extent 0 >

#We can say that the model is partially stable as we are getting 12 FALSE and 12 TRUE #3) The elementary public schools would like to choose a set of Cereals\_Data to #include in their daily cafeterias. Every day a different cereal is offered, #but all Cereals\_Data should support a healthy diet. For this goal, you are requested to find a cluster of “healthy #Cereals\_Data.”

#Clustering Healthy Cereals\_Data.  
# Ensure both Healthy\_Cereals\_new and cluster\_a have the same number of rows  
Healthy\_Cereals <- df  
Healthy\_Cereals\_new <- na.omit(Healthy\_Cereals)  
  
# Subset cluster\_a to match the number of rows in Healthy\_Cereals\_new  
cluster\_a <- cluster\_a[1:nrow(Healthy\_Cereals\_new)]  
  
# Combine Healthy\_Cereals\_new and cluster\_a  
HealthyClust <- cbind(Healthy\_Cereals\_new, cluster\_a)  
  
# Check the subsets  
HealthyClust[HealthyClust$cluster\_a == 1, ]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 1 70 4 1 130 10 5 6 280 25 3 1.0  
## 3 70 4 1 260 9 7 5 320 25 3 1.0  
## 4 50 4 0 140 14 8 0 330 25 3 1.0  
## 11 120 1 2 220 0 12 12 35 25 2 1.0  
## 22 110 2 0 220 1 21 3 30 25 3 1.0  
## 30 110 1 1 135 0 13 12 25 25 2 1.0  
## 31 100 2 0 45 0 11 15 40 25 1 1.0  
## 47 160 3 2 150 3 17 13 160 25 3 1.5  
## 48 100 2 1 220 2 15 6 90 25 1 1.0  
## 49 120 2 1 190 0 15 9 40 25 2 1.0  
## 55 50 1 0 0 0 13 0 15 0 3 0.5  
## 62 110 1 0 240 0 23 2 30 25 1 1.0  
## 63 110 2 0 290 0 22 3 35 25 1 1.0  
## 74 110 1 1 140 0 13 12 25 25 2 1.0  
## cups rating cluster\_a  
## 1 0.33 68.40297 1  
## 3 0.33 59.42551 1  
## 4 0.50 93.70491 1  
## 11 0.75 18.04285 1  
## 22 1.00 46.89564 1  
## 30 0.75 28.02576 1  
## 31 0.88 35.25244 1  
## 47 0.67 30.31335 1  
## 48 1.00 40.10596 1  
## 49 0.67 29.92429 1  
## 55 1.00 60.75611 1  
## 62 1.13 41.99893 1  
## 63 1.00 40.56016 1  
## 74 1.00 27.75330 1

HealthyClust[HealthyClust$cluster\_a == 2, ]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 2 120 3 5 15 2 8 8 135 0 3 1.00  
## 29 120 3 0 240 5 14 12 190 25 3 1.33  
## 46 150 4 3 150 3 16 11 170 25 3 1.00  
## 57 100 4 1 135 2 14 6 110 25 3 1.00  
## 59 120 3 1 210 5 14 12 240 25 2 1.33  
## 61 90 2 0 0 2 15 6 110 25 3 1.00  
## 64 80 2 0 0 3 16 0 95 0 1 0.83  
## 67 110 2 1 70 1 9 15 40 25 2 1.00  
## 68 110 6 0 230 1 16 3 55 25 1 1.00  
## 69 90 2 0 15 3 15 5 90 25 2 1.00  
## 72 100 3 1 200 3 16 3 110 100 3 1.00  
## cups rating cluster\_a  
## 2 1.00 33.98368 2  
## 29 0.67 41.01549 2  
## 46 1.00 34.13976 2  
## 57 0.50 49.51187 2  
## 59 0.75 39.25920 2  
## 61 0.50 55.33314 2  
## 64 1.00 68.23588 2  
## 67 0.75 31.23005 2  
## 68 1.00 53.13132 2  
## 69 1.00 59.36399 2  
## 72 1.00 46.65884 2

HealthyClust[HealthyClust$cluster\_a == 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 1.00  
## 7 110 2 0 125 1.0 11.0 14 30 25 2 1.00  
## 9 90 2 1 200 4.0 15.0 6 125 25 1 1.00  
## 10 90 3 0 210 5.0 13.0 5 190 25 3 1.00  
## 12 110 6 2 290 2.0 17.0 1 105 25 1 1.00  
## 14 110 3 2 140 2.0 13.0 7 105 25 3 1.00  
## 15 110 1 1 180 0.0 12.0 13 55 25 2 1.00  
## 16 110 2 0 280 0.0 22.0 3 25 25 1 1.00  
## 20 110 3 3 140 4.0 10.0 7 160 25 3 1.00  
## 25 110 2 1 125 1.0 11.0 13 30 25 2 1.00  
## 26 110 1 0 200 1.0 14.0 11 25 25 1 1.00  
## 28 120 3 2 160 5.0 12.0 10 200 25 3 1.25  
## 35 120 3 3 75 3.0 13.0 4 100 25 3 1.00  
## 36 120 1 2 220 1.0 12.0 11 45 25 2 1.00  
## 38 110 1 0 180 0.0 14.0 11 35 25 1 1.00  
## 39 110 2 1 170 1.0 17.0 6 60 100 3 1.00  
## 40 140 3 1 170 2.0 20.0 9 95 100 3 1.30  
## 41 110 2 1 260 0.0 21.0 3 40 25 2 1.00  
## 42 100 4 2 150 2.0 12.0 6 95 25 2 1.00  
## 44 100 4 1 0 0.0 16.0 3 95 25 2 1.00  
## 45 150 4 3 95 3.0 16.0 11 170 25 3 1.00  
## 50 140 3 2 220 3.0 21.0 7 130 25 3 1.33  
## 51 90 3 0 170 3.0 18.0 2 90 25 3 1.00  
## 52 130 3 2 170 1.5 13.5 10 120 25 3 1.25  
## 53 120 3 1 200 6.0 11.0 14 260 25 3 1.33  
## 54 100 3 0 320 1.0 20.0 3 45 100 3 1.00  
## 60 100 3 2 140 2.5 10.5 8 140 25 3 1.00  
## 73 110 2 1 250 0.0 21.0 3 60 25 3 1.00  
## 75 100 3 1 230 3.0 17.0 3 115 25 1 1.00  
## cups rating cluster\_a  
## 6 0.75 29.50954 3  
## 7 1.00 33.17409 3  
## 9 0.67 49.12025 3  
## 10 0.67 53.31381 3  
## 12 1.25 50.76500 3  
## 14 0.50 40.40021 3  
## 15 1.00 22.73645 3  
## 16 1.00 41.44502 3  
## 20 0.50 40.44877 3  
## 25 1.00 32.20758 3  
## 26 0.75 31.43597 3  
## 28 0.67 40.91705 3  
## 35 0.33 45.81172 3  
## 36 1.00 21.87129 3  
## 38 1.33 28.74241 3  
## 39 1.00 36.52368 3  
## 40 0.75 36.47151 3  
## 41 1.50 39.24111 3  
## 42 0.67 45.32807 3  
## 44 1.00 54.85092 3  
## 45 1.00 37.13686 3  
## 50 0.67 40.69232 3  
## 51 1.00 59.64284 3  
## 52 0.50 30.45084 3  
## 53 0.67 37.84059 3  
## 54 1.00 41.50354 3  
## 60 0.50 39.70340 3  
## 73 0.75 39.10617 3  
## 75 0.67 49.78744 3

HealthyClust[HealthyClust$cluster\_a == 4, ]

## calories protein fat sodium fiber carbo sugars potass vitamins shelf weight  
## 8 130 3 2 210 2.0 18.0 8 100 25 3 1.33  
## 19 110 1 1 180 0.0 12.0 13 65 25 2 1.00  
## 23 100 2 1 140 2.0 11.0 10 120 25 3 1.00  
## 27 100 3 0 0 3.0 14.0 7 100 25 2 1.00  
## 32 110 1 1 280 0.0 15.0 9 45 25 2 1.00  
## 33 100 3 1 140 3.0 15.0 5 85 25 3 1.00  
## 37 110 3 1 250 1.5 11.5 10 90 25 1 1.00  
## 70 110 2 1 200 0.0 21.0 3 35 100 3 1.00  
## 77 110 2 1 200 1.0 16.0 8 60 25 1 1.00  
## cups rating cluster\_a  
## 8 0.75 37.03856 4  
## 19 1.00 22.39651 4  
## 23 0.75 36.17620 4  
## 27 0.80 58.34514 4  
## 32 0.75 23.80404 4  
## 33 0.88 52.07690 4  
## 37 0.75 31.07222 4  
## 70 1.00 38.83975 4  
## 77 0.75 36.18756 4

#Mean ratings to determine the best cluster.  
mean(HealthyClust[HealthyClust$cluster\_a==1,"rating"])

## [1] 44.36873

mean(HealthyClust[HealthyClust$cluster\_a==2,"rating"])

## [1] 46.53302

mean(HealthyClust[HealthyClust$cluster\_a==3,"rating"])

## [1] 39.66133

mean(HealthyClust[HealthyClust$cluster\_a==4,"rating"])

## [1] 37.32632

#Cluster 2 has the highest mean rating (60.11492), indicating that cereals in this cluster generally have higher ratings for healthiness or nutritional quality compared to cereals in other clusters. #Cluster 1 and Cluster 3 have moderate mean ratings (46.01532 and 37.30956, respectively). #Cluster 4 has the lowest mean rating (33.65472), suggesting that cereals in this cluster may have lower health ratings compared to cereals in other clusters. #Overall, the analysis suggests that cereals can be grouped into clusters based on their nutritional characteristics, with some clusters having higher mean ratings for healthiness than others. This information can be valuable for consumers, nutritionists, and food manufacturers to understand the nutritional profiles of different cereal products.