Assignment-5

Harika

2023-12-03

Pre-processing the Data below:

Cereals <- read.csv("C:/Users/Harika/Downloads/Cereals.csv")  
Cereals\_Data <- read.csv("C:/Users/Harika/Downloads/Cereals.csv")  
str(Cereals)

## 'data.frame': 77 obs. of 16 variables:  
## $ name : chr "100%\_Bran" "100%\_Natural\_Bran" "All-Bran" "All-Bran\_with\_Extra\_Fiber" ...  
## $ mfr : chr "N" "Q" "K" "K" ...  
## $ type : chr "C" "C" "C" "C" ...  
## $ calories: int 70 120 70 50 110 110 110 130 90 90 ...  
## $ protein : int 4 3 4 4 2 2 2 3 2 3 ...  
## $ fat : int 1 5 1 0 2 2 0 2 1 0 ...  
## $ sodium : int 130 15 260 140 200 180 125 210 200 210 ...  
## $ fiber : num 10 2 9 14 1 1.5 1 2 4 5 ...  
## $ carbo : num 5 8 7 8 14 10.5 11 18 15 13 ...  
## $ sugars : int 6 8 5 0 8 10 14 8 6 5 ...  
## $ potass : int 280 135 320 330 NA 70 30 100 125 190 ...  
## $ vitamins: int 25 0 25 25 25 25 25 25 25 25 ...  
## $ shelf : int 3 3 3 3 3 1 2 3 1 3 ...  
## $ weight : num 1 1 1 1 1 1 1 1.33 1 1 ...  
## $ cups : num 0.33 1 0.33 0.5 0.75 0.75 1 0.75 0.67 0.67 ...  
## $ rating : num 68.4 34 59.4 93.7 34.4 ...

### structure of the dataset.

head(Cereals)

## name mfr type calories protein fat sodium fiber carbo  
## 1 100%\_Bran N C 70 4 1 130 10.0 5.0  
## 2 100%\_Natural\_Bran Q C 120 3 5 15 2.0 8.0  
## 3 All-Bran K C 70 4 1 260 9.0 7.0  
## 4 All-Bran\_with\_Extra\_Fiber K C 50 4 0 140 14.0 8.0  
## 5 Almond\_Delight R C 110 2 2 200 1.0 14.0  
## 6 Apple\_Cinnamon\_Cheerios G C 110 2 2 180 1.5 10.5  
## sugars potass vitamins shelf weight cups rating  
## 1 6 280 25 3 1 0.33 68.40297  
## 2 8 135 0 3 1 1.00 33.98368  
## 3 5 320 25 3 1 0.33 59.42551  
## 4 0 330 25 3 1 0.50 93.70491  
## 5 8 NA 25 3 1 0.75 34.38484  
## 6 10 70 25 1 1 0.75 29.50954

### to display the first few rows and columns.

sum(is.na(Cereals))

## [1] 4

### to count the total no.of missing values.

Cereals <- na.omit(Cereals)  
Cereals\_Data <- na.omit(Cereals\_Data)  
### to omit the missing values from the given dataset.  
sum(is.na(Cereals))

## [1] 0

rownames(Cereals) <- Cereals$name  
rownames(Cereals\_Data) <- Cereals\_Data$name  
### To convert breakfast cereal names to row names for visualizing the clusters later.

Cereals$name = NULL  
Cereals\_Data$name = NULL  
### to remove the "name" column as it does not contain any information useful.

Before calculating any distance measure, it’s essential to adjust the data. This is because variables with larger ranges can have a considerable impact on the distance calculation.

Cereals <- scale(Cereals[,3:15])

To do hierarchical clustering we used euclidean distance on the given dataset.

distance <- dist(Cereals, method = "euclidean")  
#Dissimilarity matrix  
hier\_clust <- hclust(distance, method = "complete")  
#applied hierarchical clustering using complete linkage.  
plot(hier\_clust, cex = 0.6, hang = -1)

#to plot the dendrogram.

The clustering methods, such as single linkage, complete linkage, and average linkage, each have their own agglomerative coefficients.

library(cluster)

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

hier\_clust\_single <- agnes(Cereals, method = "single")  
pltree(hier\_clust\_single, cex = 0.6, hang = -1, main = " Dendrogram of agnes ")

hier\_clust\_avg <- agnes(Cereals, method = "average")  
pltree(hier\_clust\_avg, cex = 0.6, hang = -1, main = " Dendrogram of agnes ")

To calculate the agnes coefficient for each approach.

#install.packages("purrr")  
library(purrr)

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

m <- c( "average", "single", "complete", "ward")  
# to asses the methods.  
names(m) <- c( "average", "single", "complete", "ward")  
# here function is used to compute coefficient.  
ac <- function(x) {  
 agnes(Cereals, method = x)$ac  
}  
map\_dbl(m, ac)

## average single complete ward   
## 0.7766075 0.6067859 0.8353712 0.9046042

Ward is considered the most effective way to connect things, scoring a high 0.9046042 on the agglomerative coefficient.

Using the wards approach to visualize the dendrogram:

hier\_clust\_Ward <- agnes(Cereals, method = "ward")  
pltree(hier\_clust\_Ward, cex = 0.6, hang = -1, main = " Dendrogram of agnes ")

To Cut the dendrogram with cutree() so that we can find sub-groups (i.e. clusters):

distance <- dist(Cereals, method = "euclidean")  
### to create the distance matrix.  
hier\_clust\_Ward\_clust <- hclust(distance, method = "ward.D2")  
### ward's method for hierarchical clustering.  
plot(hier\_clust\_Ward\_clust, cex=0.6)  
rect.hclust(hier\_clust\_Ward\_clust, k=6, border = 1:6)

To examine how many data records have been categorized and that are allocated to clusters:

# to cut decision tree into 6 groups  
sub\_groups <- cutree(hier\_clust\_Ward\_clust, k=6)  
# total no.of members in each single cluster.  
table(sub\_groups)

## sub\_groups  
## 1 2 3 4 5 6   
## 3 10 21 10 21 9

Correlation Matrix:

#install.packages("GGally")  
library(GGally)

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

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.2

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

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

Cereals\_Data %>%   
 select(calories, protein, fat, sodium, fiber, carbo, sugars, potass,vitamins,rating) %>%   
 ggcorr(palette = "RdBu", label = TRUE, label\_round = 2)

>>The correlation matrix helps us figure out if there’s a strong or weak connection between variables. It also allows us to calculate descriptive statistics for a better understanding.

The pvclust() function in the pvclust package helps assess the strength of clusters in hierarchical clustering using p-values from multiscale bootstrap resampling. Higher p-values indicate stronger support for clusters in the data. It’s important to note that pvclust groups columns, so transpose your data before using the function. Suzuki provides guidance on interpreting the results.

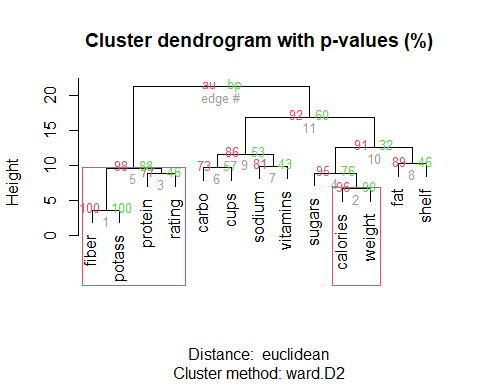
#install.packages("pvclust")  
library(pvclust)

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

fit.pv <- pvclust(Cereals, method.hclust = "ward.D2", method.dist = "euclidean")

## Bootstrap (r = 0.5)... Done.  
## Bootstrap (r = 0.59)... Done.  
## Bootstrap (r = 0.69)... Done.  
## Bootstrap (r = 0.8)... Done.  
## Bootstrap (r = 0.89)... Done.  
## Bootstrap (r = 1.0)... Done.  
## Bootstrap (r = 1.09)... Done.  
## Bootstrap (r = 1.19)... Done.  
## Bootstrap (r = 1.3)... Done.  
## Bootstrap (r = 1.39)... Done.

plot(fit.pv)  
#dendogram with p-values  
pvrect(fit.pv, alpha =.95)



In the initial clustering, we assess how stable each cluster is by looking at the average Jaccard coefficient across multiple bootstrap iterations. If a cluster’s stability rating is below 0.6, it is considered unstable. Ratings between 0.6 and 0.75 suggest that the cluster identifies a pattern in the data, but there isn’t strong agreement on which points should be grouped together. On the other hand, clusters with ratings above 0.85 are deemed exceptionally stable. 1.It’s most effective to enhance the Jaccard bootstrap on a cluster level. 2.Minimize the occurrence of dissolved clusters, and aim to enhance the recovery of clusters while maintaining proximity to the original number.

To Run Clusterboot()

#install.packages("fpc")  
library(fpc)

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

library(cluster)  
kbest\_p <- 6  
cboot\_hclust <- clusterboot(Cereals, clustermethod = hclustCBI, method = "ward.D2", k=kbest\_p)

## boot 1   
## boot 2   
## boot 3   
## boot 4   
## boot 5   
## boot 6   
## boot 7   
## boot 8   
## boot 9   
## boot 10   
## boot 11   
## boot 12   
## boot 13   
## boot 14   
## boot 15   
## boot 16   
## boot 17   
## boot 18   
## boot 19   
## boot 20   
## boot 21   
## boot 22   
## boot 23   
## boot 24   
## boot 25   
## boot 26   
## boot 27   
## boot 28   
## boot 29   
## boot 30   
## boot 31   
## boot 32   
## boot 33   
## boot 34   
## boot 35   
## boot 36   
## boot 37   
## boot 38   
## boot 39   
## boot 40   
## boot 41   
## boot 42   
## boot 43   
## boot 44   
## boot 45   
## boot 46   
## boot 47   
## boot 48   
## boot 49   
## boot 50   
## boot 51   
## boot 52   
## boot 53   
## boot 54   
## boot 55   
## boot 56   
## boot 57   
## boot 58   
## boot 59   
## boot 60   
## boot 61   
## boot 62   
## boot 63   
## boot 64   
## boot 65   
## boot 66   
## boot 67   
## boot 68   
## boot 69   
## boot 70   
## boot 71   
## boot 72   
## boot 73   
## boot 74   
## boot 75   
## boot 76   
## boot 77   
## boot 78   
## boot 79   
## boot 80   
## boot 81   
## boot 82   
## boot 83   
## boot 84   
## boot 85   
## boot 86   
## boot 87   
## boot 88   
## boot 89   
## boot 90   
## boot 91   
## boot 92   
## boot 93   
## boot 94   
## boot 95   
## boot 96   
## boot 97   
## boot 98   
## boot 99   
## boot 100

summary(cboot\_hclust$result)

## Length Class Mode   
## result 7 hclust list   
## noise 1 -none- logical   
## nc 1 -none- numeric   
## clusterlist 6 -none- list   
## partition 74 -none- numeric   
## clustermethod 1 -none- character  
## nccl 1 -none- numeric

groups <- cboot\_hclust$result$partition  
head(data.frame(groups))

## groups  
## 100%\_Bran 1  
## 100%\_Natural\_Bran 2  
## All-Bran 1  
## All-Bran\_with\_Extra\_Fiber 1  
## Apple\_Cinnamon\_Cheerios 3  
## Apple\_Jacks 3

cboot\_hclust$bootmean

## [1] 0.9001426 0.5208173 0.9074587 0.6352364 0.5966988 0.6776262

### vector for cluster stabilities.

cboot\_hclust$bootbrd

## [1] 11 58 0 38 33 36

### to Count how many times each group was broken apart. The clusterboot() function usually runs 100 bootstrap iterations by default.

The findings suggest that Clusters 1 and 3 are quite reliable. However, Clusters 4 and 5 show some pattern recognition, but there’s disagreement on which points should be grouped together. Clusters 2 and 5 are currently not stable.

To Extract the clusters found by hclust()

groups <- cutree(hier\_clust\_Ward\_clust, k = 6)  
print\_cluster <- function(labels, k)  
{  
 for (i in 1:k)   
 {  
 print(paste("cluster",i))  
 print(Cereals\_Data[labels==i,c("mfr","calories","protein","fat","sodium","fiber","carbo","sugars","potass","vitamins","rating")])  
 }  
}  
print\_cluster(groups, 6)

## [1] "cluster 1"  
## mfr calories protein fat sodium fiber carbo sugars  
## 100%\_Bran N 70 4 1 130 10 5 6  
## All-Bran K 70 4 1 260 9 7 5  
## All-Bran\_with\_Extra\_Fiber K 50 4 0 140 14 8 0  
## potass vitamins rating  
## 100%\_Bran 280 25 68.40297  
## All-Bran 320 25 59.42551  
## All-Bran\_with\_Extra\_Fiber 330 25 93.70491  
## [1] "cluster 2"  
## mfr calories protein fat sodium fiber carbo  
## 100%\_Natural\_Bran Q 120 3 5 15 2.0 8.0  
## Clusters G 110 3 2 140 2.0 13.0  
## Cracklin'\_Oat\_Bran K 110 3 3 140 4.0 10.0  
## Crispy\_Wheat\_&\_Raisins G 100 2 1 140 2.0 11.0  
## Great\_Grains\_Pecan P 120 3 3 75 3.0 13.0  
## Life Q 100 4 2 150 2.0 12.0  
## Muesli\_Raisins,\_Dates,\_&\_Almonds R 150 4 3 95 3.0 16.0  
## Muesli\_Raisins,\_Peaches,\_&\_Pecans R 150 4 3 150 3.0 16.0  
## Quaker\_Oat\_Squares Q 100 4 1 135 2.0 14.0  
## Raisin\_Nut\_Bran G 100 3 2 140 2.5 10.5  
## sugars potass vitamins rating  
## 100%\_Natural\_Bran 8 135 0 33.98368  
## Clusters 7 105 25 40.40021  
## Cracklin'\_Oat\_Bran 7 160 25 40.44877  
## Crispy\_Wheat\_&\_Raisins 10 120 25 36.17620  
## Great\_Grains\_Pecan 4 100 25 45.81172  
## Life 6 95 25 45.32807  
## Muesli\_Raisins,\_Dates,\_&\_Almonds 11 170 25 37.13686  
## Muesli\_Raisins,\_Peaches,\_&\_Pecans 11 170 25 34.13976  
## Quaker\_Oat\_Squares 6 110 25 49.51187  
## Raisin\_Nut\_Bran 8 140 25 39.70340  
## [1] "cluster 3"  
## mfr calories protein fat sodium fiber carbo sugars  
## Apple\_Cinnamon\_Cheerios G 110 2 2 180 1.5 10.5 10  
## Apple\_Jacks K 110 2 0 125 1.0 11.0 14  
## Cap'n'Crunch Q 120 1 2 220 0.0 12.0 12  
## Cinnamon\_Toast\_Crunch G 120 1 3 210 0.0 13.0 9  
## Cocoa\_Puffs G 110 1 1 180 0.0 12.0 13  
## Corn\_Pops K 110 1 0 90 1.0 13.0 12  
## Count\_Chocula G 110 1 1 180 0.0 12.0 13  
## Froot\_Loops K 110 2 1 125 1.0 11.0 13  
## Frosted\_Flakes K 110 1 0 200 1.0 14.0 11  
## Fruity\_Pebbles P 110 1 1 135 0.0 13.0 12  
## Golden\_Crisp P 100 2 0 45 0.0 11.0 15  
## Golden\_Grahams G 110 1 1 280 0.0 15.0 9  
## Honey\_Graham\_Ohs Q 120 1 2 220 1.0 12.0 11  
## Honey\_Nut\_Cheerios G 110 3 1 250 1.5 11.5 10  
## Honey-comb P 110 1 0 180 0.0 14.0 11  
## Lucky\_Charms G 110 2 1 180 0.0 12.0 12  
## Multi-Grain\_Cheerios G 100 2 1 220 2.0 15.0 6  
## Nut&Honey\_Crunch K 120 2 1 190 0.0 15.0 9  
## Smacks K 110 2 1 70 1.0 9.0 15  
## Trix G 110 1 1 140 0.0 13.0 12  
## Wheaties\_Honey\_Gold G 110 2 1 200 1.0 16.0 8  
## potass vitamins rating  
## Apple\_Cinnamon\_Cheerios 70 25 29.50954  
## Apple\_Jacks 30 25 33.17409  
## Cap'n'Crunch 35 25 18.04285  
## Cinnamon\_Toast\_Crunch 45 25 19.82357  
## Cocoa\_Puffs 55 25 22.73645  
## Corn\_Pops 20 25 35.78279  
## Count\_Chocula 65 25 22.39651  
## Froot\_Loops 30 25 32.20758  
## Frosted\_Flakes 25 25 31.43597  
## Fruity\_Pebbles 25 25 28.02576  
## Golden\_Crisp 40 25 35.25244  
## Golden\_Grahams 45 25 23.80404  
## Honey\_Graham\_Ohs 45 25 21.87129  
## Honey\_Nut\_Cheerios 90 25 31.07222  
## Honey-comb 35 25 28.74241  
## Lucky\_Charms 55 25 26.73451  
## Multi-Grain\_Cheerios 90 25 40.10596  
## Nut&Honey\_Crunch 40 25 29.92429  
## Smacks 40 25 31.23005  
## Trix 25 25 27.75330  
## Wheaties\_Honey\_Gold 60 25 36.18756  
## [1] "cluster 4"  
## mfr calories protein fat sodium fiber  
## Basic\_4 G 130 3 2 210 2.0  
## Fruit\_&\_Fibre\_Dates,\_Walnuts,\_and\_Oats P 120 3 2 160 5.0  
## Fruitful\_Bran K 120 3 0 240 5.0  
## Just\_Right\_Fruit\_&\_Nut K 140 3 1 170 2.0  
## Mueslix\_Crispy\_Blend K 160 3 2 150 3.0  
## Nutri-Grain\_Almond-Raisin K 140 3 2 220 3.0  
## Oatmeal\_Raisin\_Crisp G 130 3 2 170 1.5  
## Post\_Nat.\_Raisin\_Bran P 120 3 1 200 6.0  
## Raisin\_Bran K 120 3 1 210 5.0  
## Total\_Raisin\_Bran G 140 3 1 190 4.0  
## carbo sugars potass vitamins rating  
## Basic\_4 18.0 8 100 25 37.03856  
## Fruit\_&\_Fibre\_Dates,\_Walnuts,\_and\_Oats 12.0 10 200 25 40.91705  
## Fruitful\_Bran 14.0 12 190 25 41.01549  
## Just\_Right\_Fruit\_&\_Nut 20.0 9 95 100 36.47151  
## Mueslix\_Crispy\_Blend 17.0 13 160 25 30.31335  
## Nutri-Grain\_Almond-Raisin 21.0 7 130 25 40.69232  
## Oatmeal\_Raisin\_Crisp 13.5 10 120 25 30.45084  
## Post\_Nat.\_Raisin\_Bran 11.0 14 260 25 37.84059  
## Raisin\_Bran 14.0 12 240 25 39.25920  
## Total\_Raisin\_Bran 15.0 14 230 100 28.59278  
## [1] "cluster 5"  
## mfr calories protein fat sodium fiber carbo sugars  
## Bran\_Chex R 90 2 1 200 4 15 6  
## Bran\_Flakes P 90 3 0 210 5 13 5  
## Cheerios G 110 6 2 290 2 17 1  
## Corn\_Chex R 110 2 0 280 0 22 3  
## Corn\_Flakes K 100 2 0 290 1 21 2  
## Crispix K 110 2 0 220 1 21 3  
## Double\_Chex R 100 2 0 190 1 18 5  
## Grape\_Nuts\_Flakes P 100 3 1 140 3 15 5  
## Grape-Nuts P 110 3 0 170 3 17 3  
## Just\_Right\_Crunchy\_\_Nuggets K 110 2 1 170 1 17 6  
## Kix G 110 2 1 260 0 21 3  
## Nutri-grain\_Wheat K 90 3 0 170 3 18 2  
## Product\_19 K 100 3 0 320 1 20 3  
## Rice\_Chex R 110 1 0 240 0 23 2  
## Rice\_Krispies K 110 2 0 290 0 22 3  
## Special\_K K 110 6 0 230 1 16 3  
## Total\_Corn\_Flakes G 110 2 1 200 0 21 3  
## Total\_Whole\_Grain G 100 3 1 200 3 16 3  
## Triples G 110 2 1 250 0 21 3  
## Wheat\_Chex R 100 3 1 230 3 17 3  
## Wheaties G 100 3 1 200 3 17 3  
## potass vitamins rating  
## Bran\_Chex 125 25 49.12025  
## Bran\_Flakes 190 25 53.31381  
## Cheerios 105 25 50.76500  
## Corn\_Chex 25 25 41.44502  
## Corn\_Flakes 35 25 45.86332  
## Crispix 30 25 46.89564  
## Double\_Chex 80 25 44.33086  
## Grape\_Nuts\_Flakes 85 25 52.07690  
## Grape-Nuts 90 25 53.37101  
## Just\_Right\_Crunchy\_\_Nuggets 60 100 36.52368  
## Kix 40 25 39.24111  
## Nutri-grain\_Wheat 90 25 59.64284  
## Product\_19 45 100 41.50354  
## Rice\_Chex 30 25 41.99893  
## Rice\_Krispies 35 25 40.56016  
## Special\_K 55 25 53.13132  
## Total\_Corn\_Flakes 35 100 38.83975  
## Total\_Whole\_Grain 110 100 46.65884  
## Triples 60 25 39.10617  
## Wheat\_Chex 115 25 49.78744  
## Wheaties 110 25 51.59219  
## [1] "cluster 6"  
## mfr calories protein fat sodium fiber carbo sugars  
## Frosted\_Mini-Wheats K 100 3 0 0 3 14 7  
## Maypo A 100 4 1 0 0 16 3  
## Puffed\_Rice Q 50 1 0 0 0 13 0  
## Puffed\_Wheat Q 50 2 0 0 1 10 0  
## Raisin\_Squares K 90 2 0 0 2 15 6  
## Shredded\_Wheat N 80 2 0 0 3 16 0  
## Shredded\_Wheat\_'n'Bran N 90 3 0 0 4 19 0  
## Shredded\_Wheat\_spoon\_size N 90 3 0 0 3 20 0  
## Strawberry\_Fruit\_Wheats N 90 2 0 15 3 15 5  
## potass vitamins rating  
## Frosted\_Mini-Wheats 100 25 58.34514  
## Maypo 95 25 54.85092  
## Puffed\_Rice 15 0 60.75611  
## Puffed\_Wheat 50 0 63.00565  
## Raisin\_Squares 110 25 55.33314  
## Shredded\_Wheat 95 0 68.23588  
## Shredded\_Wheat\_'n'Bran 140 0 74.47295  
## Shredded\_Wheat\_spoon\_size 120 0 72.80179  
## Strawberry\_Fruit\_Wheats 90 25 59.36399

I chose clusters based on both statistical and nutritional values to create a healthy diet, but it’s subjective as there’s no clear measure for a healthy diet. I decided not to normalize the data to keep the original scale for better analysis. The cereal diet levels in the six clusters vary in richness, adequacy, and nutrient deficiencies. Cluster 1 has balanced guidelines, but limited options. Clusters 2 and 3 are not recommended due to poor ratings and high fat and sugar content. Clusters 4 and 5 have well-balanced nutrition and high consumer satisfaction, making them ideal for primary school cafeterias. So, for schools aiming to provide healthy meals, Clusters 4 and 5 are the best choices.