ML Assignment-5

Abhinay Chiranjeeth Marneni

2022-12-01

loading library functions packages

```
library(ISLR)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
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(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8     v purrr   0.3.4
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(cluster)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.2.2
```

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

```
library(proxy)
```

```
##
## Attaching package: 'proxy'
##
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
##
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(Rfast)
```

```
## Warning: package 'Rfast' was built under R version 4.2.2
```

```
## Loading required package: Rcpp
## Loading required package: RcppZiggurat
```

```
## Warning: package 'RcppZiggurat' was built under R version 4.2.2
```

```
##
## Attaching package: 'Rfast'
##
## The following objects are masked from 'package:purrr':
##
##     is_integer, transpose
##
## The following object is masked from 'package:dplyr':
##
##     nth
```

```
library(NbClust)
library(ggplot2)
```

```
data<-read.csv("C:/Users/abhin/OneDrive/Documents/Assigments Buss 1sem/ML/Cereals.csv") # import the da
row.names(data) <- data[,1]
data <- data[-1]
data.norm<-scale(data[c(-1,-2,-12)])
view(data) # using view function to display the whole table str(data)  # to examine the data set struct
head(data) # to get first 6rows of the data set
```

```
##                          mfr type calories protein fat sodium fiber carbo
## 100%_Bran                  N    C       70       4   1    130  10.0   5.0
## 100%_Natural_Bran          Q    C      120       3   5     15   2.0   8.0
```

```
## All-Bran                    K    C       70       4   1    260   9.0   7.0
## All-Bran_with_Extra_Fiber   K    C       50       4   0    140  14.0   8.0
## Almond_Delight              R    C      110       2   2    200   1.0  14.0
## Apple_Cinnamon_Cheerios     G    C      110       2   2    180   1.5  10.5
##                          sugars potass vitamins shelf weight cups   rating
## 100%_Bran                     6    280       25     3      1 0.33 68.40297
## 100%_Natural_Bran             8    135        0     3      1 1.00 33.98368
## All-Bran                      5    320       25     3      1 0.33 59.42551
## All-Bran_with_Extra_Fiber     0    330       25     3      1 0.50 93.70491
## Almond_Delight                8     NA       25     3      1 0.75 34.38484
## Apple_Cinnamon_Cheerios      10     70       25     1      1 0.75 29.50954
```

```
summary(data) # to examine the data set summary
```

```
##      mfr                type              calories        protein
##  Length:77          Length:77          Min.   : 50.0   Min.   :1.000
##  Class :character   Class :character   1st Qu.:100.0   1st Qu.:2.000
##  Mode  :character   Mode  :character   Median :110.0   Median :3.000
##                                        Mean   :106.9   Mean   :2.545
##                                        3rd Qu.:110.0   3rd Qu.:3.000
##                                        Max.   :160.0   Max.   :6.000
##
##       fat            sodium          fiber            carbo
##  Min.   :0.000   Min.   :  0.0   Min.   : 0.000   Min.   : 5.0
##  1st Qu.:0.000   1st Qu.:130.0   1st Qu.: 1.000   1st Qu.:12.0
##  Median :1.000   Median :180.0   Median : 2.000   Median :14.5
##  Mean   :1.013   Mean   :159.7   Mean   : 2.152   Mean   :14.8
##  3rd Qu.:2.000   3rd Qu.:210.0   3rd Qu.: 3.000   3rd Qu.:17.0
##  Max.   :5.000   Max.   :320.0   Max.   :14.000   Max.   :23.0
##                                                   NA's   :1
##      sugars           potass          vitamins          shelf
##  Min.   : 0.000   Min.   : 15.00   Min.   :  0.00   Min.   :1.000
##  1st Qu.: 3.000   1st Qu.: 42.50   1st Qu.: 25.00   1st Qu.:1.000
##  Median : 7.000   Median : 90.00   Median : 25.00   Median :2.000
##  Mean   : 7.026   Mean   : 98.67   Mean   : 28.25   Mean   :2.208
##  3rd Qu.:11.000   3rd Qu.:120.00   3rd Qu.: 25.00   3rd Qu.:3.000
##  Max.   :15.000   Max.   :330.00   Max.   :100.00   Max.   :3.000
##  NA's   :1        NA's   :2
##      weight          cups           rating
##  Min.   :0.50   Min.   :0.250   Min.   :18.04
##  1st Qu.:1.00   1st Qu.:0.670   1st Qu.:33.17
##  Median :1.00   Median :0.750   Median :40.40
##  Mean   :1.03   Mean   :0.821   Mean   :42.67
##  3rd Qu.:1.00   3rd Qu.:1.000   3rd Qu.:50.83
##  Max.   :1.50   Max.   :1.500   Max.   :93.70
##
```

To remove NA values from the data set, scale the data now.

```
data1 <- na.omit(data)
data1<- as.data.frame(na.omit(data1))#removing the Cereals dataset missing values
```

```
datascale <- scale(data1[,-c(1:3)])   # Scaling the dataset
df <- as.data.frame(data1)
summary(data1)
```

```
##      mfr                type              calories      protein
##  Length:74          Length:74          Min.   : 50   Min.   :1.000
##  Class :character   Class :character   1st Qu.:100   1st Qu.:2.000
##  Mode  :character   Mode  :character   Median :110   Median :2.500
##                                        Mean   :107   Mean   :2.514
##                                        3rd Qu.:110   3rd Qu.:3.000
##                                        Max.   :160   Max.   :6.000
##       fat        sodium          fiber            carbo           sugars
##  Min.   :0   Min.   :  0.0   Min.   : 0.000   Min.   : 5.00   Min.   : 0.000
##  1st Qu.:0   1st Qu.:135.0   1st Qu.: 0.250   1st Qu.:12.00   1st Qu.: 3.000
##  Median :1   Median :180.0   Median : 2.000   Median :14.50   Median : 7.000
##  Mean   :1   Mean   :162.4   Mean   : 2.176   Mean   :14.73   Mean   : 7.108
##  3rd Qu.:1   3rd Qu.:217.5   3rd Qu.: 3.000   3rd Qu.:17.00   3rd Qu.:11.000
##  Max.   :5   Max.   :320.0   Max.   :14.000   Max.   :23.00   Max.   :15.000
##      potass          vitamins          shelf           weight
##  Min.   : 15.00   Min.   :  0.00   Min.   :1.000   Min.   :0.500
##  1st Qu.: 41.25   1st Qu.: 25.00   1st Qu.:1.250   1st Qu.:1.000
##  Median : 90.00   Median : 25.00   Median :2.000   Median :1.000
##  Mean   : 98.51   Mean   : 29.05   Mean   :2.216   Mean   :1.031
##  3rd Qu.:120.00   3rd Qu.: 25.00   3rd Qu.:3.000   3rd Qu.:1.000
##  Max.   :330.00   Max.   :100.00   Max.   :3.000   Max.   :1.500
##       cups            rating
##  Min.   :0.2500   Min.   :18.04
##  1st Qu.:0.6700   1st Qu.:32.45
##  Median :0.7500   Median :40.25
##  Mean   :0.8216   Mean   :42.37
##  3rd Qu.:1.0000   3rd Qu.:50.52
##  Max.   :1.5000   Max.   :93.70
```

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

```
data_Euclidean <- dist(datascale, method = "euclidean") # Creating the dissimilarity matrix for data se
DataWard.D <- hclust(data_Euclidean, method = "ward.D")
```

#Use Agnes to compare the clustering from single linkage, complete linkage, average linkage, and Ward

```
Single <- agnes(data1, method = "single")
Complete <- agnes(data1, method = "complete")
Average <- agnes(data1, method = "average")
Ward <- agnes(data1, method = "ward")
print(Single$ac)
```

```
## [1] 0.7297141
```

```
print(Complete$ac)
```

```
## [1] 0.9225732
```

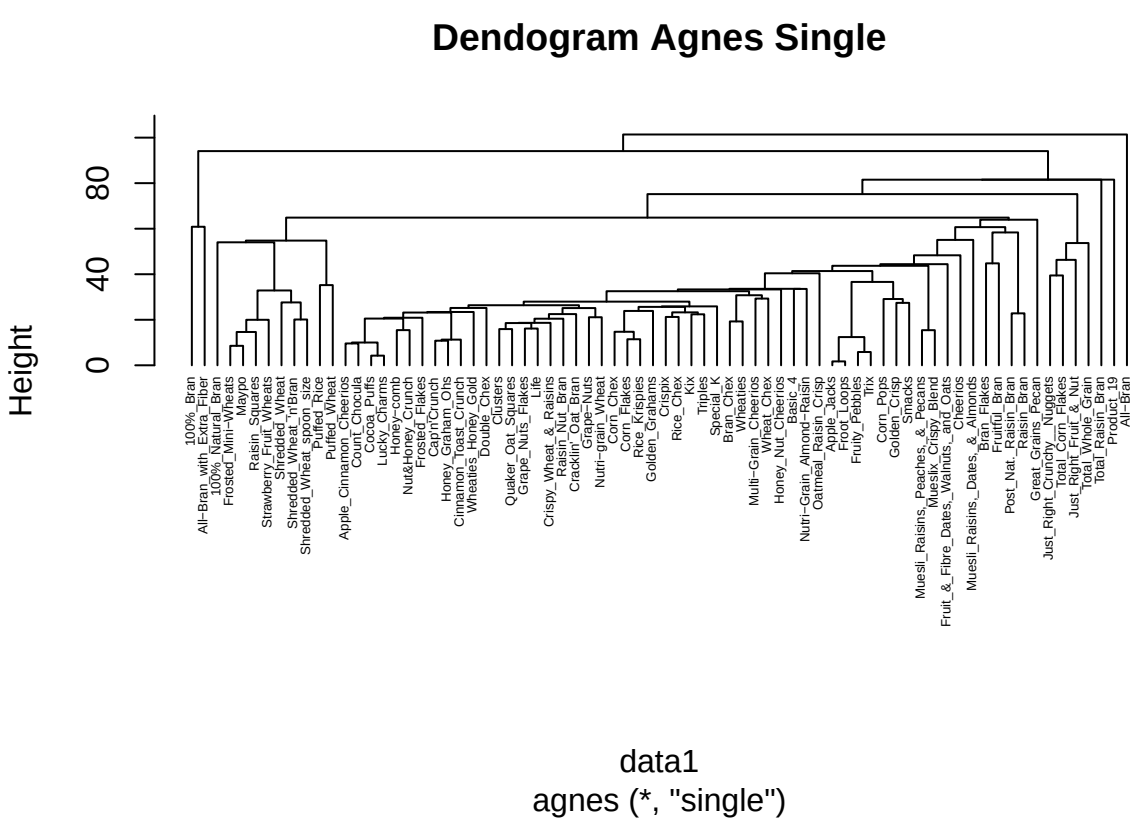
```
print(Average$ac)
```

```
## [1] 0.8786692
```

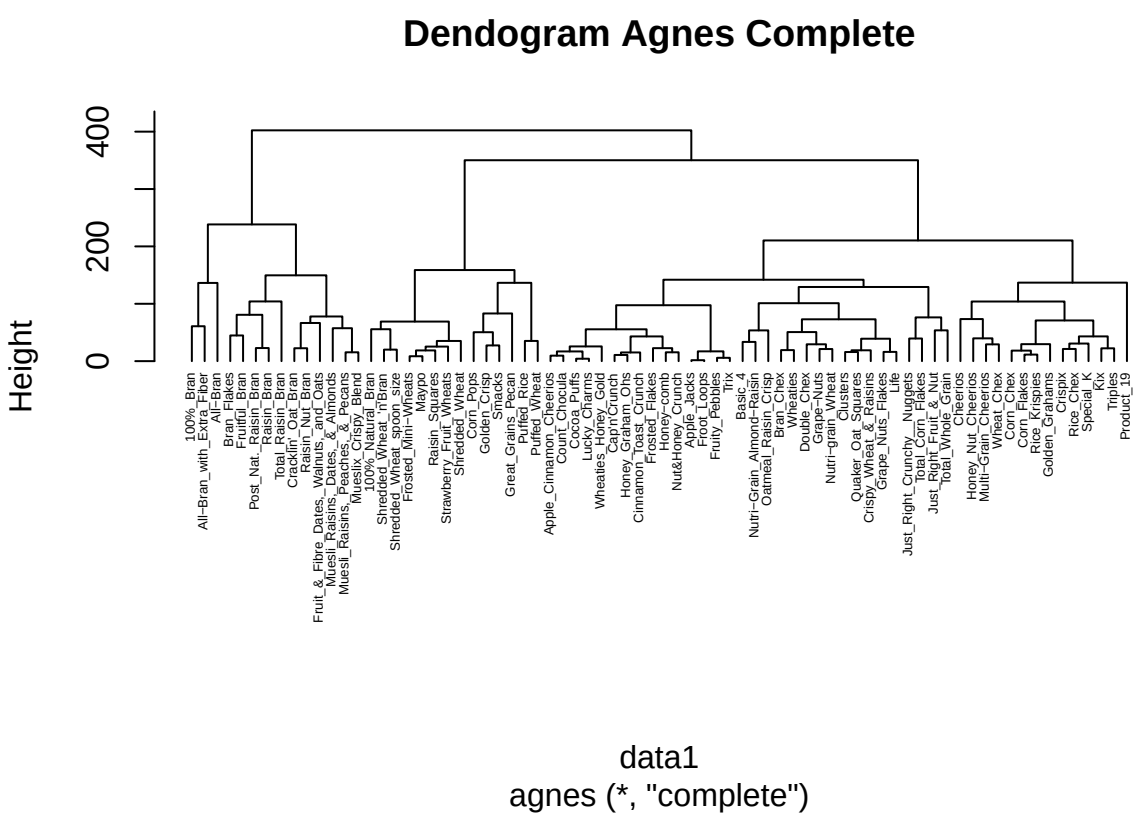
```
print(Ward$ac)
```

```
## [1] 0.959504
```

```
pltree(Single, cex=0.4, hang=-1, main = "Dendogram Agnes Single")
```

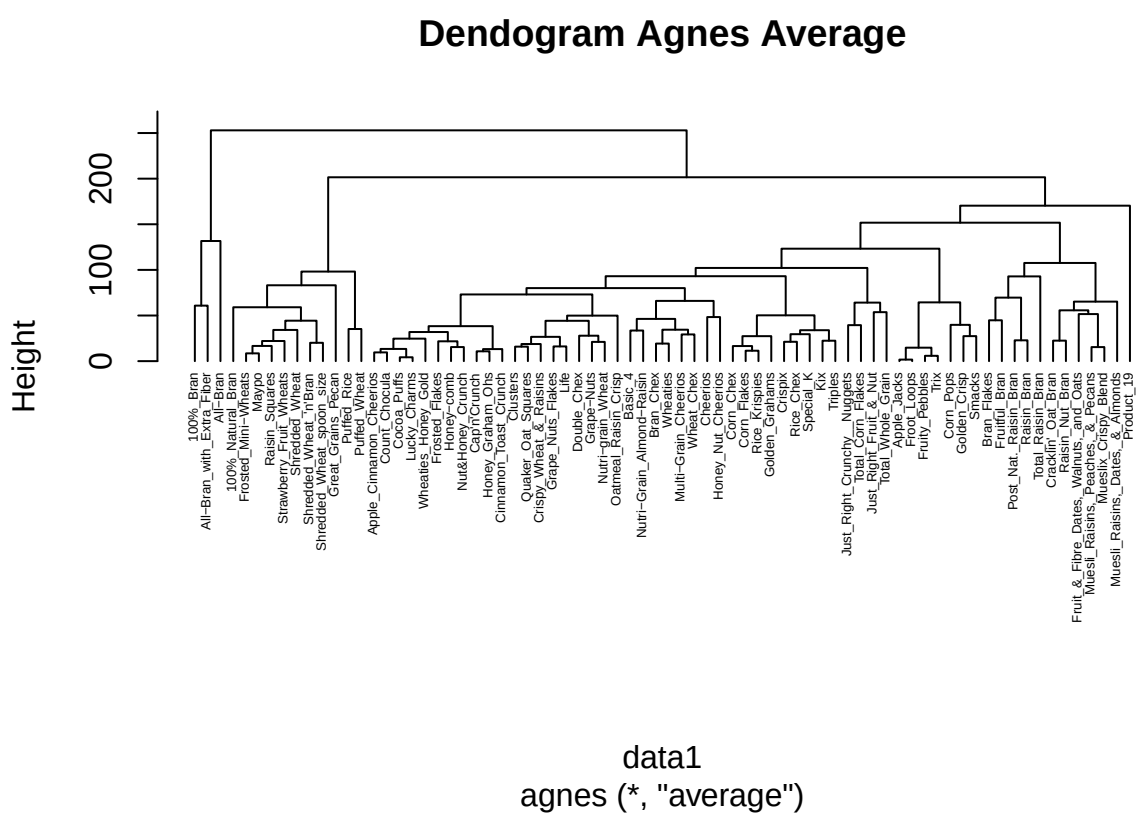



```
pltree(Complete, cex=0.4, hang=-1, main = "Dendogram Agnes Complete")
```

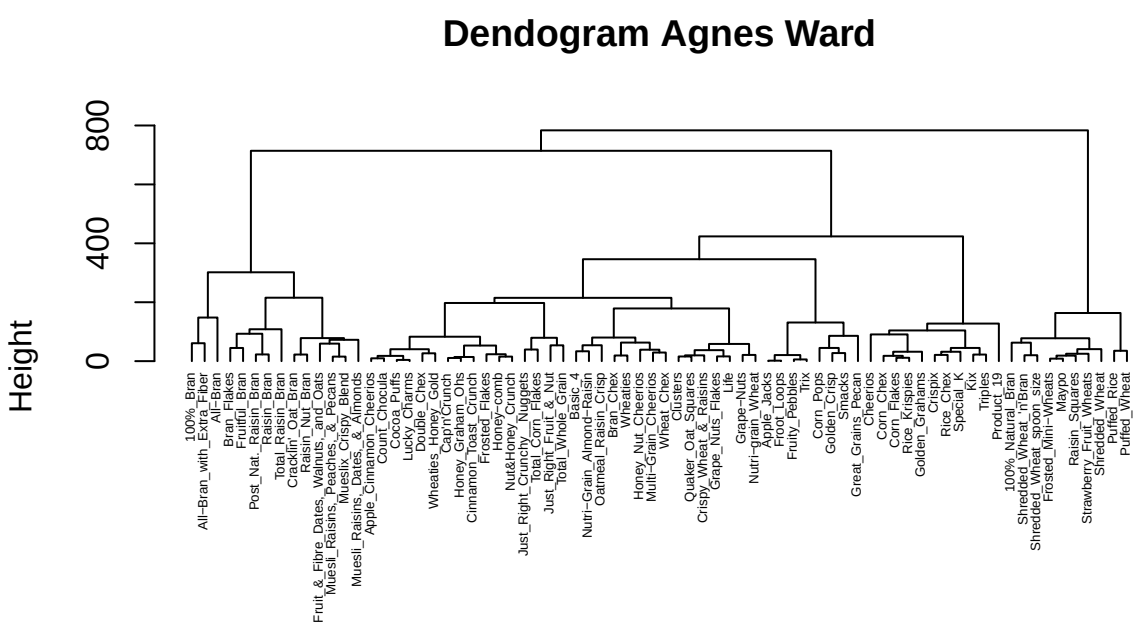


data1
agnes (*, "complete")

```
pltree(Average, cex=0.4, hang=-1, main = "Dendogram Agnes Average")
```




```
pltree(Ward, cex=0.4, hang=-1, main = "Dendogram Agnes Ward")
```



"Ward Method" is the best approach based on the results above. B. # How many clusters would you choose.

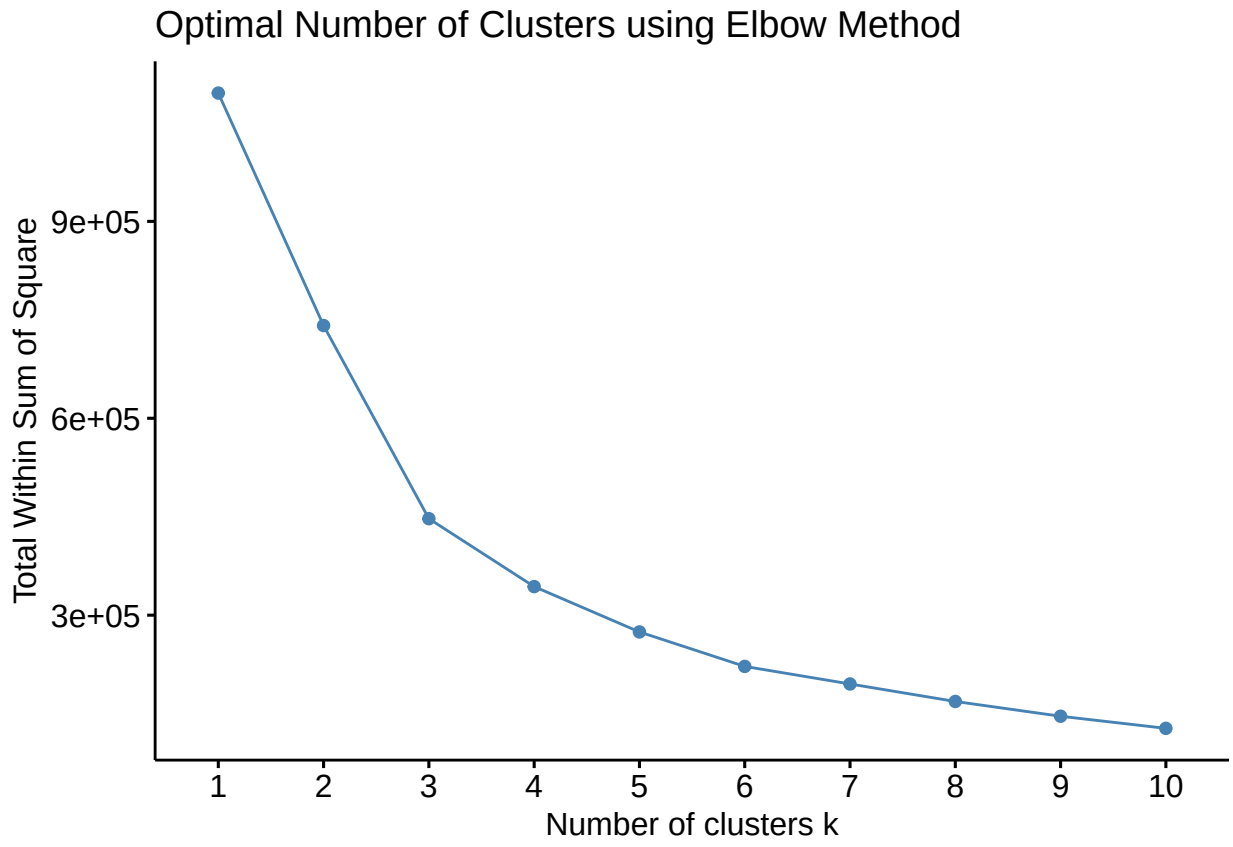
```
Ward_Euclidean<- hclust(data_Euclidean , method = "ward.D" ) #Ward method is the optimum. As  the data
```

##Here, the elbow and silhouette approaches are used to determine the optimal number of clusters. ## Elbow Method:

```
fviz_nbclust(data1, hcut, method = "wss") +labs(title = "Optimal Number of Clusters using Elbow Method")
```

```
## Warning in stats::dist(x): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```

```
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```

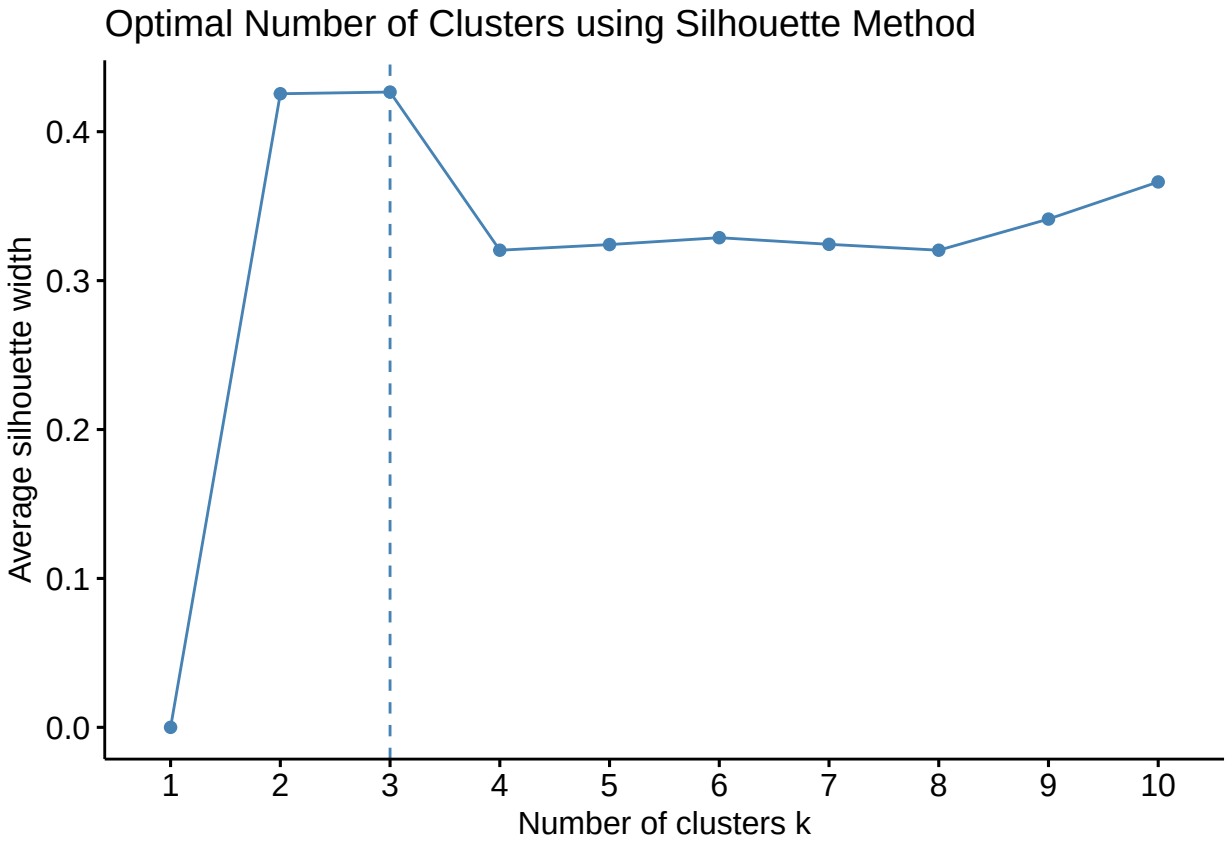


##Silhouette Method:

```
fviz_nbclust(data1, hcut,  method = "silhouette") +labs(title = "Optimal Number of Clusters using Silhou
```

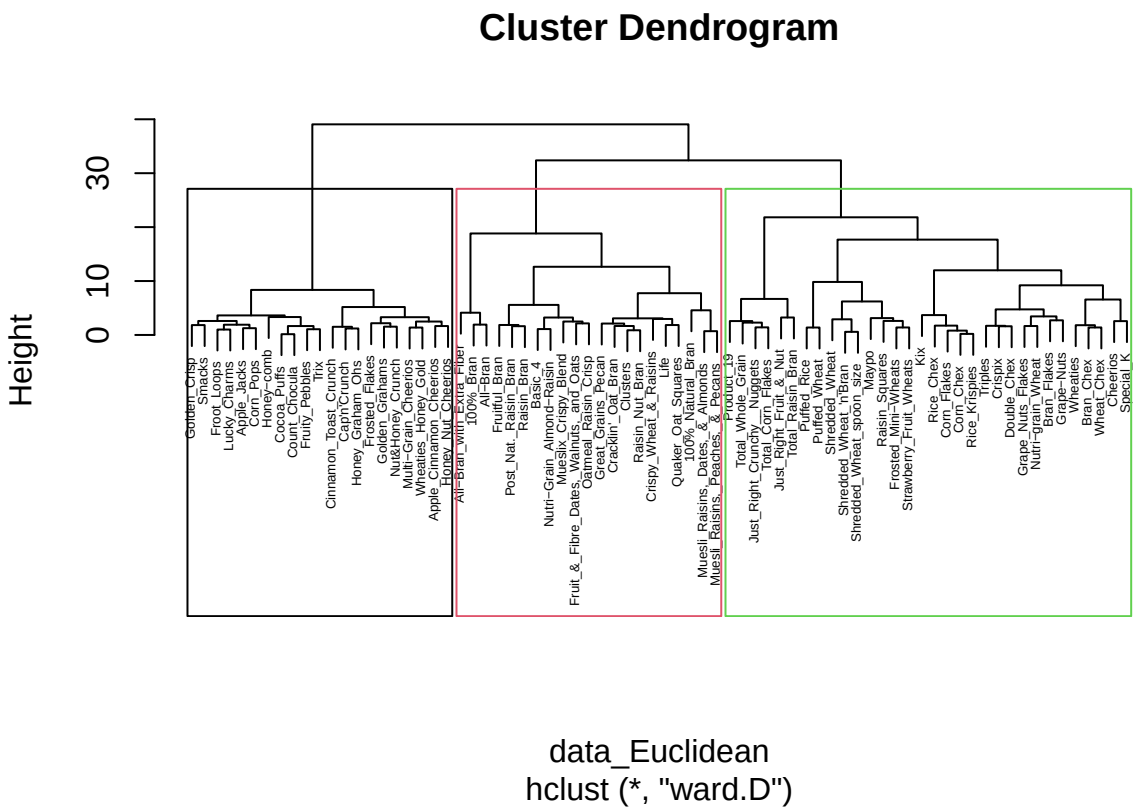
```
## Warning in stats::dist(x): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```

```
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```



#Here, we can see from the outcomes of the elbow and silhouette approaches that 3 clusters would be the optimal number. # so iam choosing Silhouette Method kmean value 3

```
plot(Ward_Euclidean, cex = 0.4)
rect.hclust(Ward_Euclidean, k = 3, border = 1:3)
```




```
Group <- cutree(DataWard.D, k = 3)
table(Group)
```

```
## Group
##  1  2  3
## 21 21 32
```

```
data2 <- cbind.data.frame(data1, Group)
```

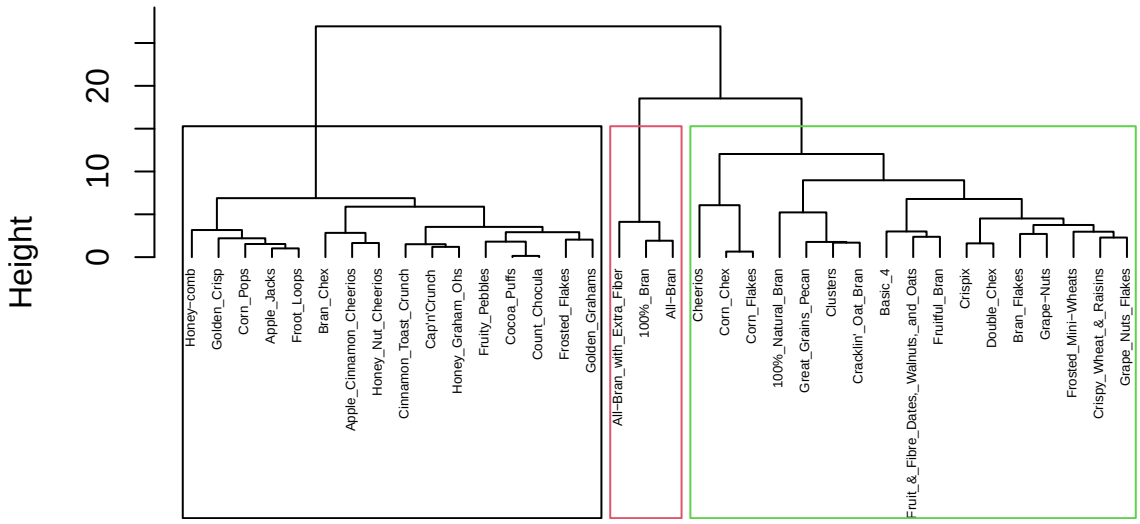
C.

#Partition the data.

```
TrainData <- datascale [1:36,] # Partition 1
TestData <- datascale [37:74,] # partition 2
```

#Use the cluster centroids from A to assign each record in partition B.

```
Euclidean_Train <- dist(TrainData, method = "euclidean")
DataWard.D1 <- hclust(Euclidean_Train, method = "ward.D")
plot(DataWard.D1, cex = 0.4, hang = -1)
rect.hclust(DataWard.D1, k = 3, border = 1:3)
```



Euclidean_Train
hclust (*, "ward.D")

```
ClusterGroup <- cutree(DataWard.D1, k = 3)
table(ClusterGroup)
```

```
## ClusterGroup
##  1  2  3
##  3 17 16
```

```
TrainData1 <- cbind.data.frame(TrainData, ClusterGroup)
```

#Create the Cluster centroids for above partition data

```
Centroid_1 <- colMeans(TrainData1 [TrainData1 $ClusterGroup == "1",])
Centroid_2 <- colMeans(TrainData1 [TrainData1 $ClusterGroup == "2",])
Centroid_3 <- colMeans(TrainData1 [TrainData1 $ClusterGroup == "3",])
Centroid_4 <- colMeans(TrainData1 [TrainData1 $ClusterGroup == "4",])
Centroid <- rbind(Centroid_1, Centroid_2, Centroid_3, Centroid_4)
CentroidTest <- rowMins(dist(TestData, Centroid[,-13]))
CentroidPartition <- c(TrainData1 $ClusterGroup,CentroidTest)
data3<- cbind(data2 ,CentroidPartition)
table(data3$Group == data3 $CentroidPartition)# compared the total Data clusters with the TrainData and
```

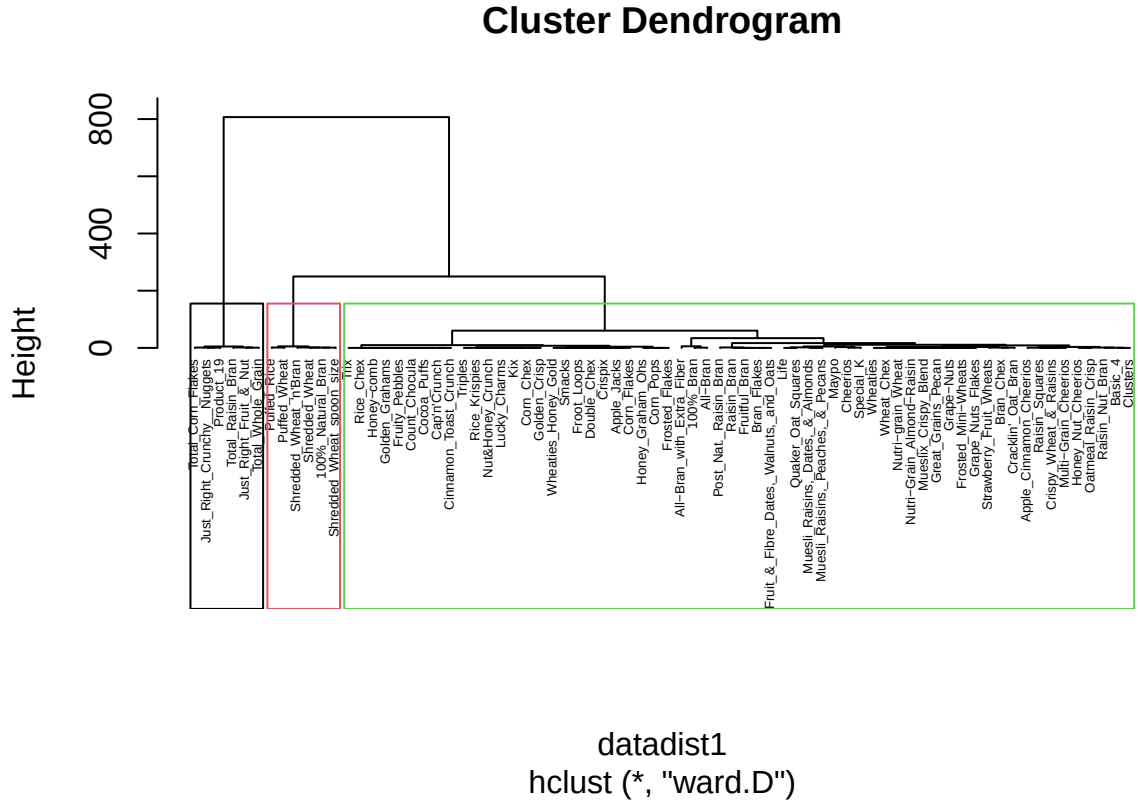
```
##
## FALSE  TRUE
##    67     7
```

```
table(data3$Group[37:74] == data3$CentroidPartition[37:74])
```

```
##
## FALSE  TRUE
##    35     3
```

D.

```
data4 <- data1[, c(4,7,11)]
datadist1 <- dist(data4, method = "euclidean")
DataWard.D2 <- hclust(datadist1, method = "ward.D")
plot(DataWard.D2, cex = 0.4, hang = -1)
rect.hclust(DataWard.D2, k = 3, border = 1:3)
```




```
Group1 <- cutree(DataWard.D2, k = 3)
table(Group1)
```

```
## Group1
##  1  2  3
## 62  6  6
```

```r
aggregate(data4, by = list(cluster = Group1), mean)
```

```
##   cluster  protein    fiber vitamins
## 1       1 2.516129 2.209677       25
## 2       2 2.333333 2.166667        0
## 3       3 2.666667 1.833333      100
```

Determine the healthy cereals.I create a clusters base on data of cereals i choose protein, fiber and vitamins and to know how much amount protein, fiber and vitamins the cereals contain. So won't normalized data intained of that i analysis the cluster base on this three factory.4 clusters were formed after protein, fiber and vitamins clustering. The The elementary public schools has choosen Cluster 3, which has somany cereals and Some clusters were not accept because they lacked a suitable ratio of protein, fiber and vitamins. They either had a lot of protein and fiber, vitamins have less amount compare to data.