cereals

Obianuju Anumnu

12/4/2020

```
Cereal<- read.csv("D:/Machine/cereal assignment/Cereals.csv")
```

```
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
library(caret)
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.2
```

```
## Loading required package: carData
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## Warning: package 'readr' was built under R version 4.0.2
```

```
## Warning: package 'forcats' was built under R version 4.0.2
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()
```

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

```
## Warning: package 'factoextra' was built under R version 4.0.3
```

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

```
library(cluster)
library(hrbrthemes)
```

```
## Warning: package 'hrbrthemes' was built under R version 4.0.3
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
```

```
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
```

```
##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.0.3
```

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

```
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 4.0.3
```

```
## Loading required package: viridisLite
```

```
library(fpc)
```

```
## Warning: package 'fpc' was built under R version 4.0.3
```

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.0.3
```

#Data Preprocessing

```r
#remove cereals with missing data
Cereals<- na.omit(Cereal)
#confirm that each record is unique
record<- as.data.frame(table(Cereals[1]))
#add row names
row.names(Cereals)<- Cereals[,1]
#remove the name column
Cereals<-Cereals[,-1]
```

```r
#review data structure
str(Cereals)
```

```
## 'data.frame':    74 obs. of  15 variables:
##  $ mfr     : chr  "N" "Q" "K" "K" ...
##  $ type    : chr  "C" "C" "C" "C" ...
##  $ calories: int  70 120 70 50 110 110 130 90 90 120 ...
##  $ protein : int  4 3 4 4 2 2 3 2 3 1 ...
##  $ fat     : int  1 5 1 0 2 0 2 1 0 2 ...
##  $ sodium  : int  130 15 260 140 180 125 210 200 210 220 ...
##  $ fiber   : num  10 2 9 14 1.5 1 2 4 5 0 ...
##  $ carbo   : num  5 8 7 8 10.5 11 18 15 13 12 ...
##  $ sugars  : int  6 8 5 0 10 14 8 6 5 12 ...
##  $ potass  : int  280 135 320 330 70 30 100 125 190 35 ...
##  $ vitamins: int  25 0 25 25 25 25 25 25 25 25 ...
##  $ shelf   : int  3 3 3 3 1 2 3 1 3 2 ...
##  $ weight  : num  1 1 1 1 1 1 1.33 1 1 1 ...
##  $ cups    : num  0.33 1 0.33 0.5 0.75 1 0.75 0.67 0.67 0.75 ...
##  $ rating  : num  68.4 34 59.4 93.7 29.5 ...
```

```r
#remove variables that are not required and convert all required variables to numeric.
#"mfr","type" are not required
cereals<- Cereals[,c(-1,-2)]
#convert shelf variable to numeric
cereals$shelf <- as.factor(cereals$shelf)
#create dummy variables to accomodate three shelfs in the dataset
cerealsdummy<- dummyVars(~shelf, data = cereals)
head(predict(cerealsdummy,cereals))
```

```
##                           shelf.1 shelf.2 shelf.3
## 100%_Bran                       0       0       1
## 100%_Natural_Bran               0       0       1
## All-Bran                        0       0       1
## All-Bran_with_Extra_Fiber       0       0       1
## Apple_Cinnamon_Cheerios         1       0       0
## Apple_Jacks                     0       1       0
```

```r
cereals1<-dummy.data.frame(cereals,names = c("shelf"), sep ="-")
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):
## non-list contrasts argument ignored
```

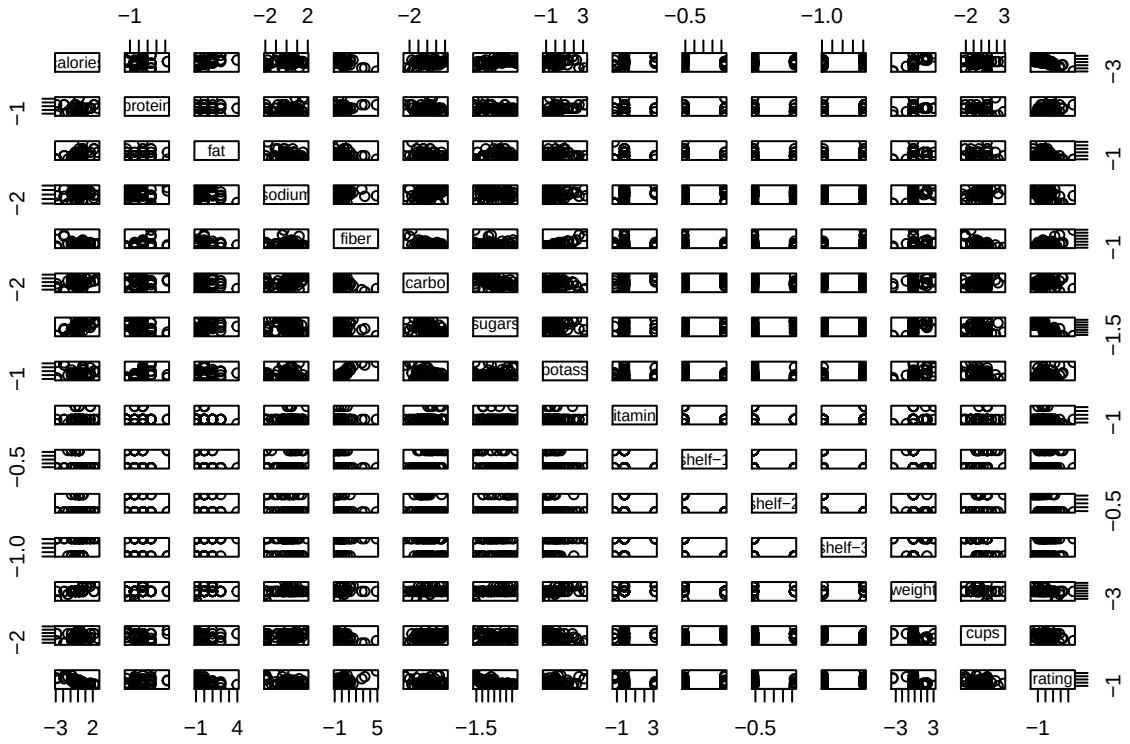
```
summary(cereals1)
```

```
##     calories      protein           fat         sodium          fiber
##  Min.   : 50   Min.   :1.000   Min.   :0   Min.   :  0.0   Min.   : 0.000
##  1st Qu.:100   1st Qu.:2.000   1st Qu.:0   1st Qu.:135.0   1st Qu.: 0.250
##  Median :110   Median :2.500   Median :1   Median :180.0   Median : 2.000
##  Mean   :107   Mean   :2.514   Mean   :1   Mean   :162.4   Mean   : 2.176
##  3rd Qu.:110   3rd Qu.:3.000   3rd Qu.:1   3rd Qu.:217.5   3rd Qu.: 3.000
##  Max.   :160   Max.   :6.000   Max.   :5   Max.   :320.0   Max.   :14.000
##      carbo           sugars           potass          vitamins
##  Min.   : 5.00   Min.   : 0.000   Min.   : 15.00   Min.   :  0.00
##  1st Qu.:12.00   1st Qu.: 3.000   1st Qu.: 41.25   1st Qu.: 25.00
##  Median :14.50   Median : 7.000   Median : 90.00   Median : 25.00
##  Mean   :14.73   Mean   : 7.108   Mean   : 98.51   Mean   : 29.05
##  3rd Qu.:17.00   3rd Qu.:11.000   3rd Qu.:120.00   3rd Qu.: 25.00
##  Max.   :23.00   Max.   :15.000   Max.   :330.00   Max.   :100.00
##     shelf-1          shelf-2          shelf-3          weight
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.500
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:1.000
##  Median :0.0000   Median :0.0000   Median :0.000   Median :1.000
##  Mean   :0.2568   Mean   :0.2703   Mean   :0.473   Mean   :1.031
##  3rd Qu.:0.7500   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   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
```

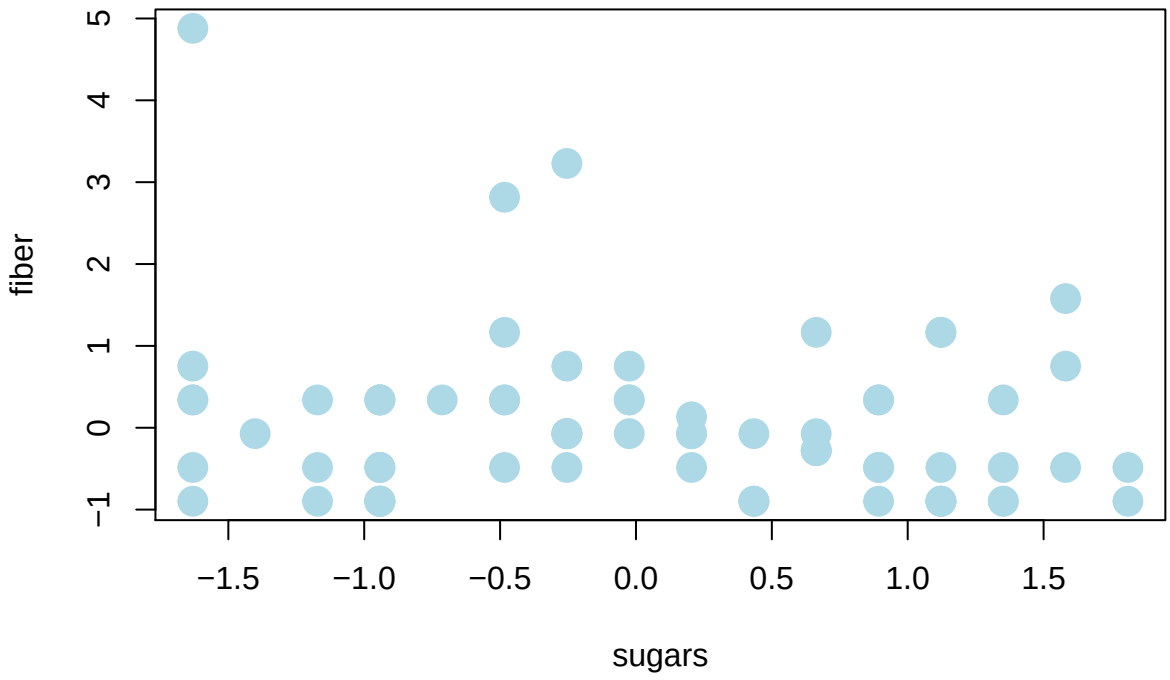
```
#the range of variables are too wide so we normalize to bring all the variables to the same range.
cereals2<- scale(cereals1)
```

#Data Exploration

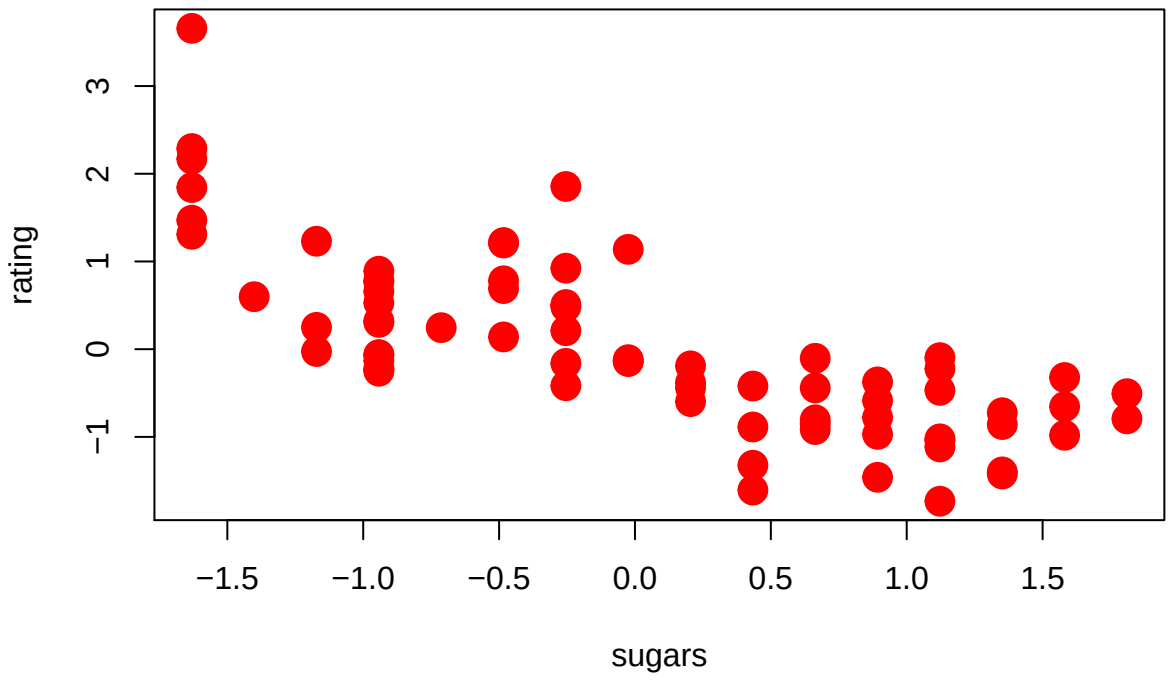
```
pairs(cereals2)
```



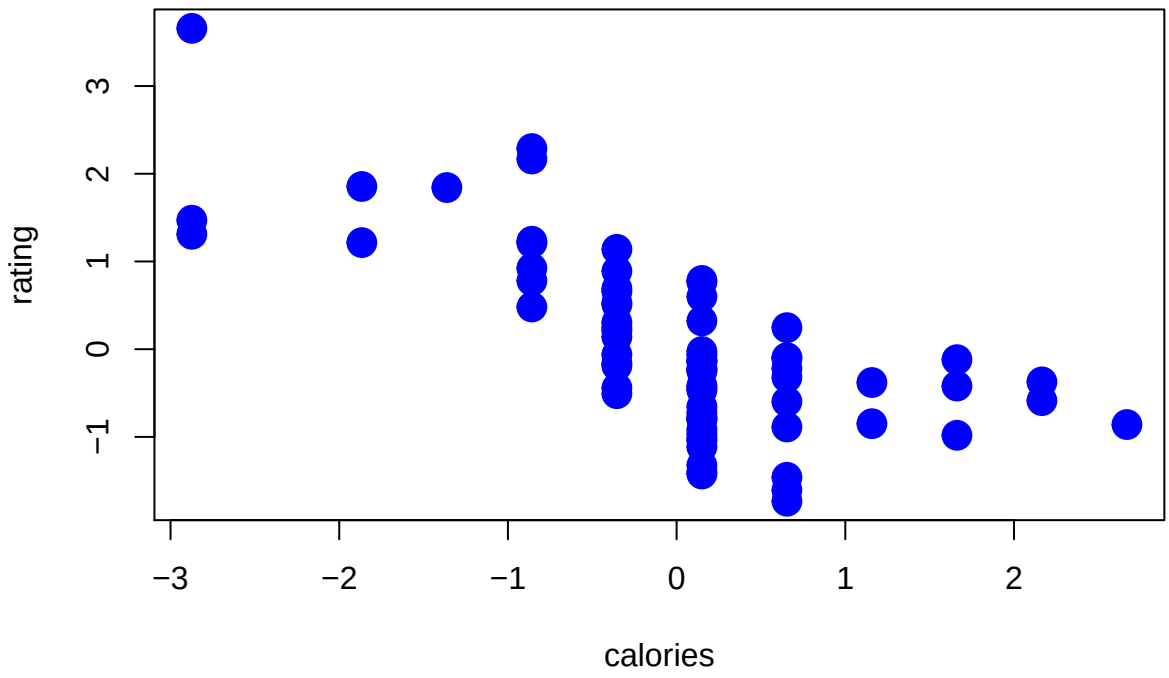

```
#scatter plot for different pairs of variables
plot(fiber~sugars, col="lightblue", pch=19, cex=2,data=cereals2)
```



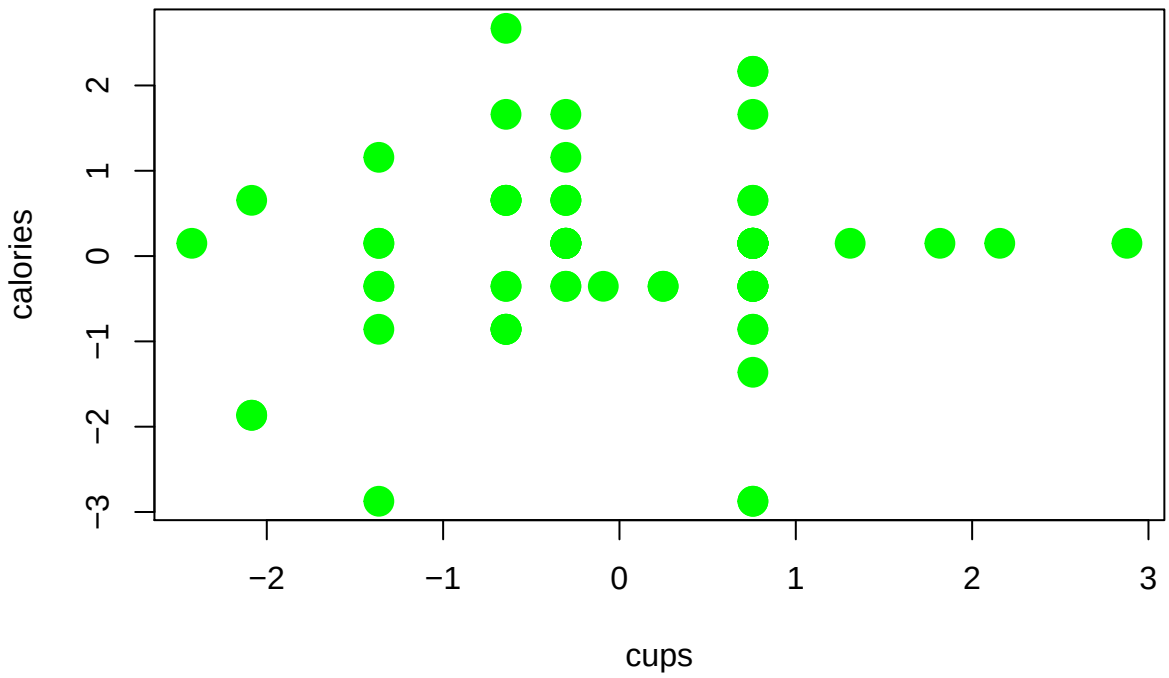

```
plot(rating ~sugars, col="red", pch=19, cex=2,data=cereals2)
```




```
plot(rating~calories, col="blue", pch=19, cex=2, data=cereals2)
```



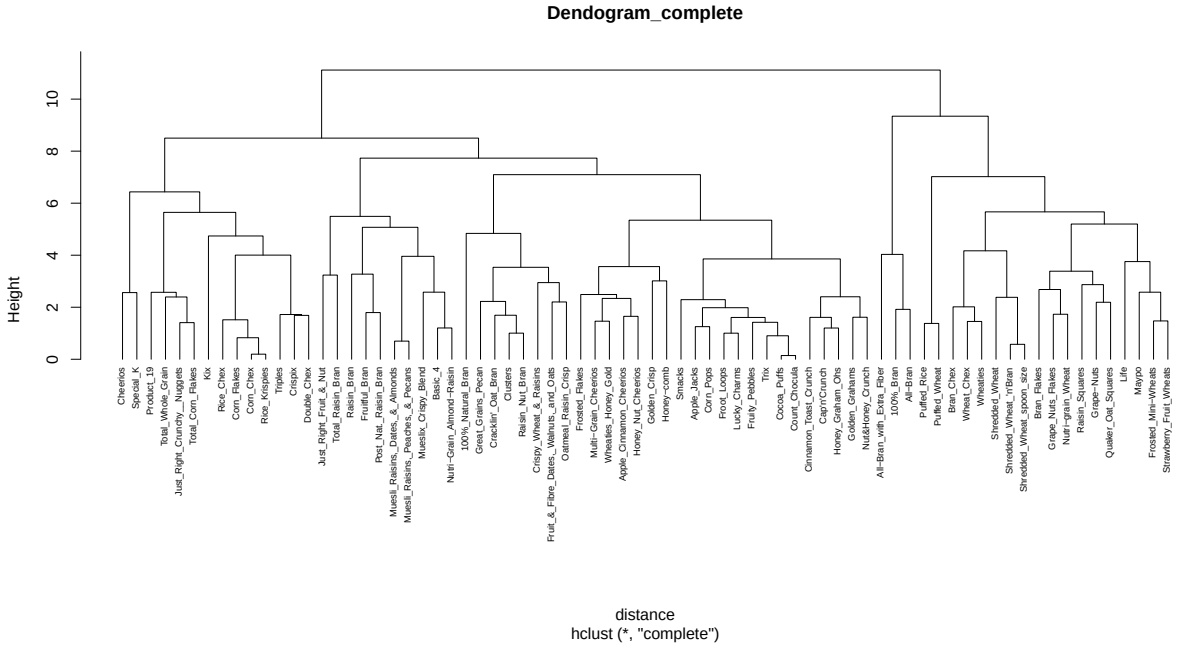

```
plot(calories ~cups, col="green", pch=19, cex=2,data=cereals2)
```



we can see that pairs of variables are showing commonalities and patterns of groups/clusters

#Hierarchical clustering

```
#Apply hierarchical clustering to the data using Euclidean distance to the normalized measurements.
distance <- dist(cereals2, method = "euclidean")
# Hierarchical clustering using Complete Linkage
hc1 <- hclust(distance, method = "complete")
# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1, main = "Dendogram_complete")
```




```
#Use Agnes to compare the clustering from single linkage, complete linkage, average linkage, and Ward.
hc2 <- agnes(cereals2, method = "complete")
hc2$ac
```

```
## [1] 0.8341098
```

```
# vector of methods to compare
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")
# function to compute coefficient
ac <- function(x) {
  agnes(cereals2, method = x)$ac
  }
map_dbl(m, ac)
```

```
##   average    single  complete      ward
## 0.7734558 0.6234190 0.8341098 0.9045198
```

#ward linkage has the strongest clustering structure,

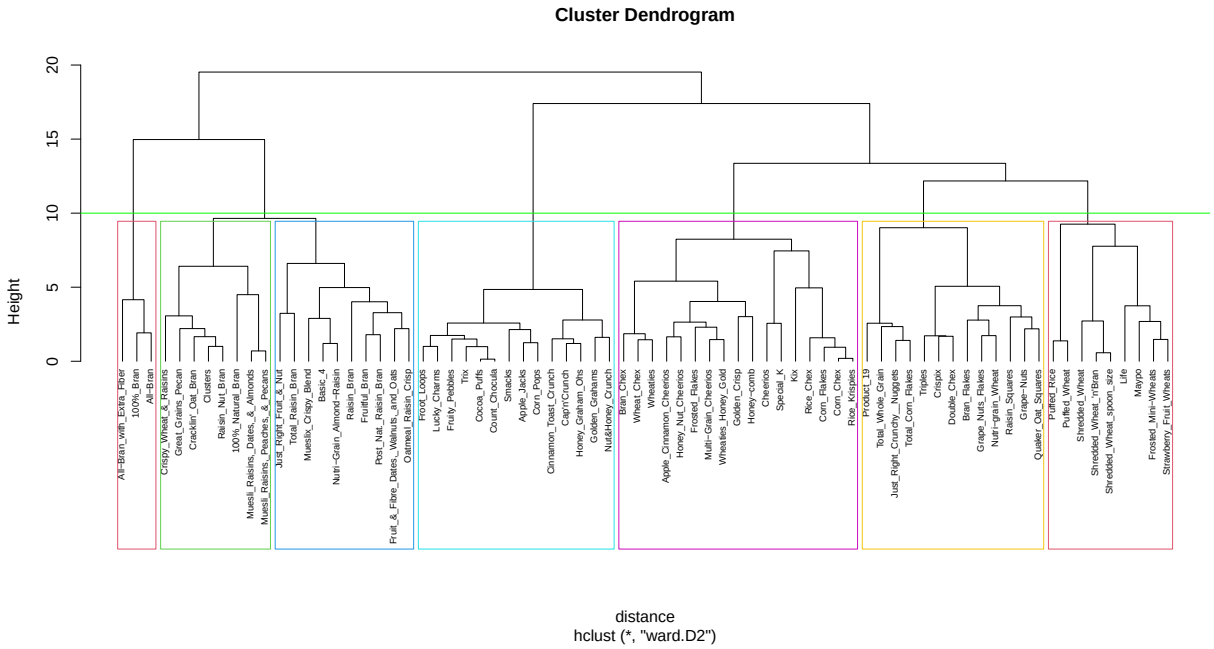
```
#let us apply hierarichical clustring using ward
hc3<-hclust(distance,method = "ward.D2")
plot(hc3, cex = 0.6, hang = -1, main = "Dendogram_Agnes")
```

#Comment on differences between hierarchical Clustering and K-means #K- means clustering divides the set of data objects into non- overlapping subsets(clusters) such that each data object is in exactly one subset. While hierarchical clustering is a set of nested clusters that are arranged as a tree, K Means clustering requires a knowledge of K i.e. no. of clusters one want to divide your data, this is not required in hierarchical clustering(HC), for HC one can stop at any number of clusters determined appropriate by interpreting the dendrogram. for K Means clustering results produced by running the algorithm many times may differ since the algorithm starts by random choice of k(clusters), In Hierarchical Clustering, results remain the same in K Means Methods used are normally less computationally intensive and are suited with very large datasets; in Hierarchical clustering, divisive methods work in the opposite direction, beginning with one cluster that includes all the records. Hierarchical methods are especially useful when the target is to arrange the clusters into a natural hierarchy.

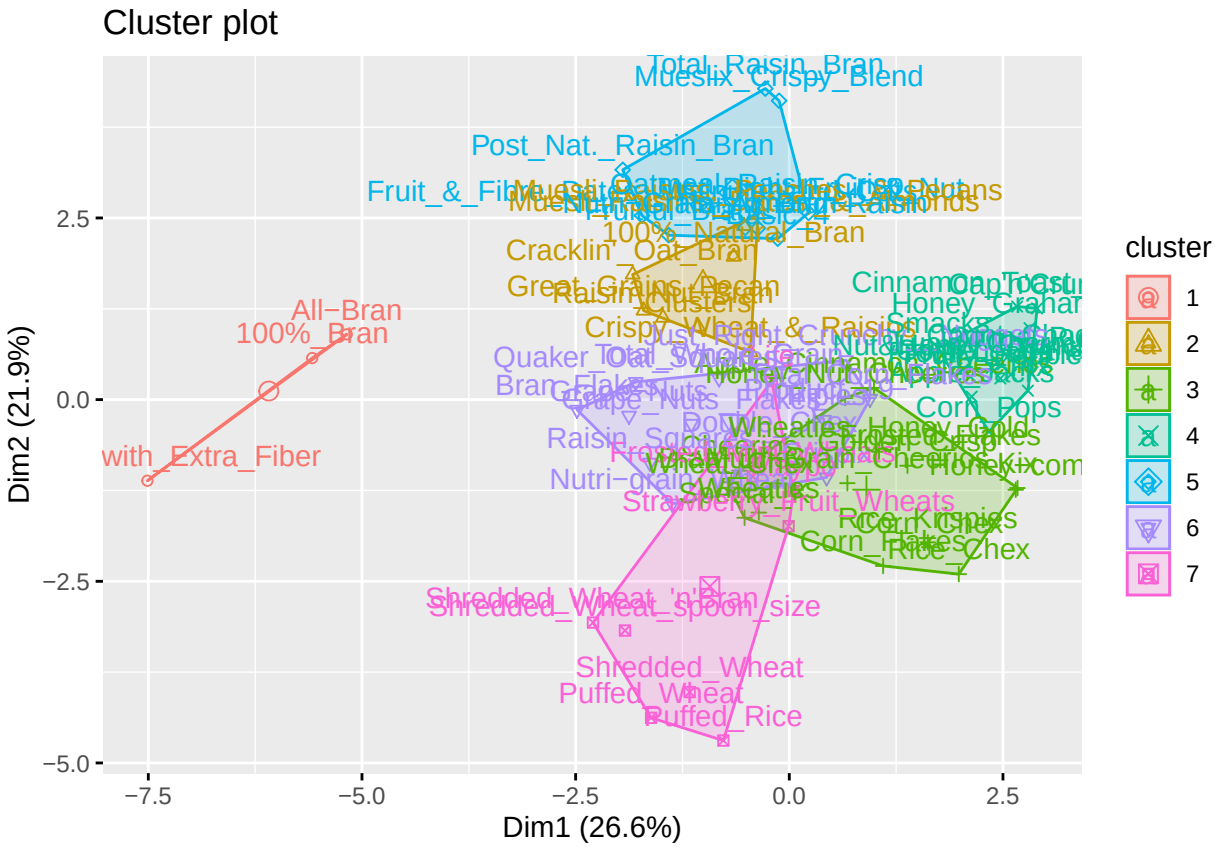
```
#How many clusters would you choose?
#based on data exploration, 7 clusters appear to be common among the paired variables
#Cut the tree into 7 clusters
clusters <- cutree(hc3, k = 7)
#number of cereals in each cluster
table(clusters)
```

```
## clusters
##  1  2  3  4  5  6  7
##  3  8 17 14 10 13  9
```

```
plot(hc3, cex= 0.6, hang = -1)
#Plot clusters with borders
rect.hclust(hc3, k = 7, border = 2:7)
abline(h = 10, col = 'green')
```




```
#more cluster visualization using the fviz_cluster function in a scatterplot.
fviz_cluster(list(data = cereals2, cluster = clusters))
```

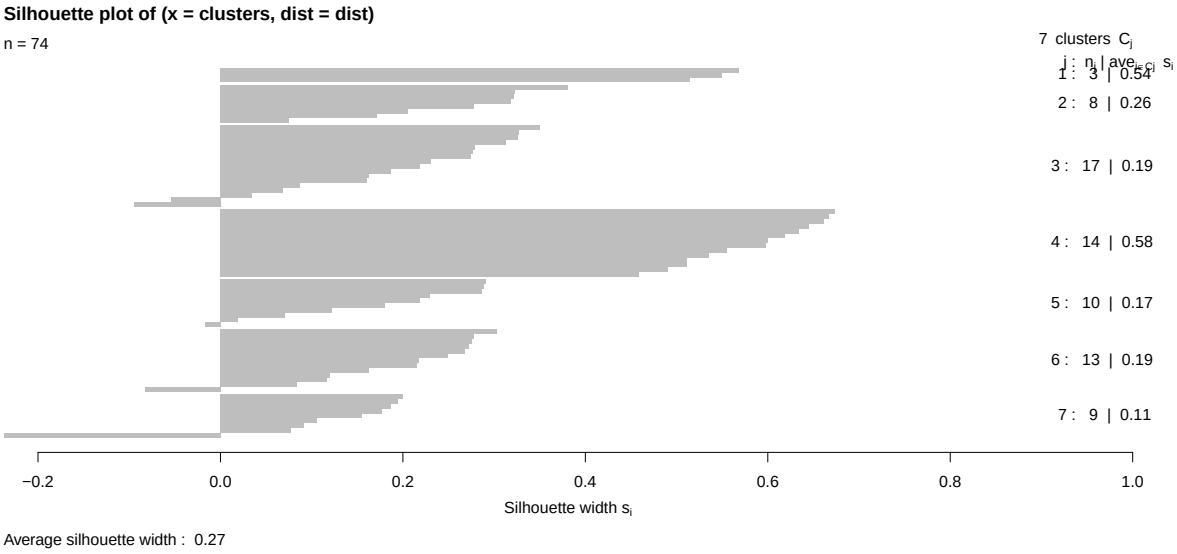


#determining optimal number of clusters using the silhouette method - (intra-cluster cohesion and inter-cluster separation)

```
set.seed(123)
dist = daisy(x = cereals2, metric = "euclidean")
```

```
## Warning in daisy(x = cereals2, metric = "euclidean"): binary variable(s) 10, 11,
## 12 treated as interval scaled
```

```
sil_value = silhouette(clusters, dist = dist)
plot(sil_value)
```

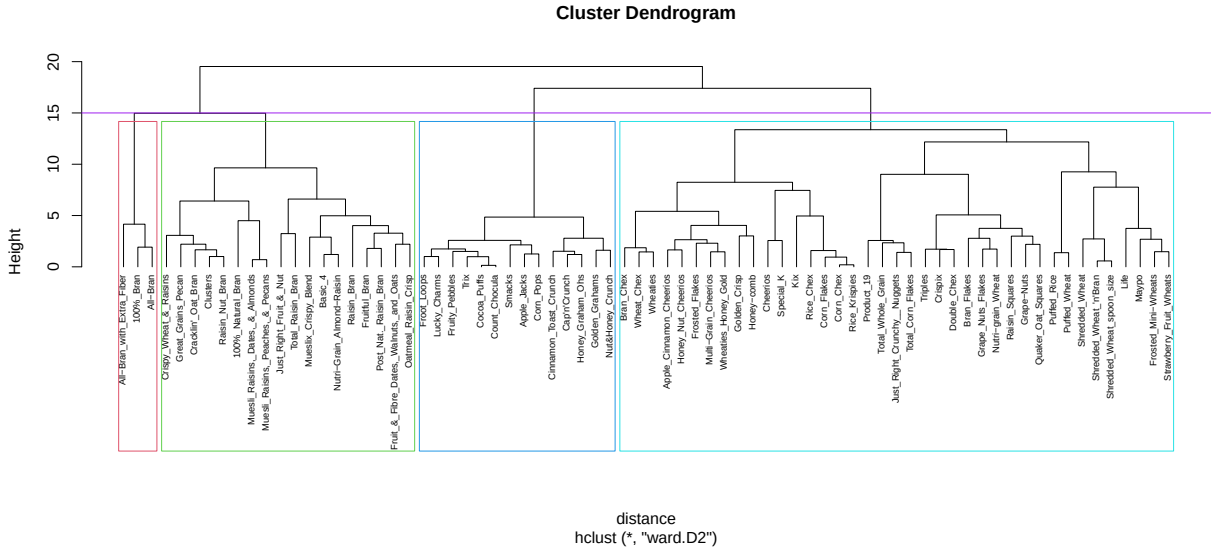


4 clusters has the highest silhouette score of 0.58

```
#cluster data with k=4
clusters1 <- cutree(hc3, k = 4)
#number of cereals in each cluster
table(clusters1)
```

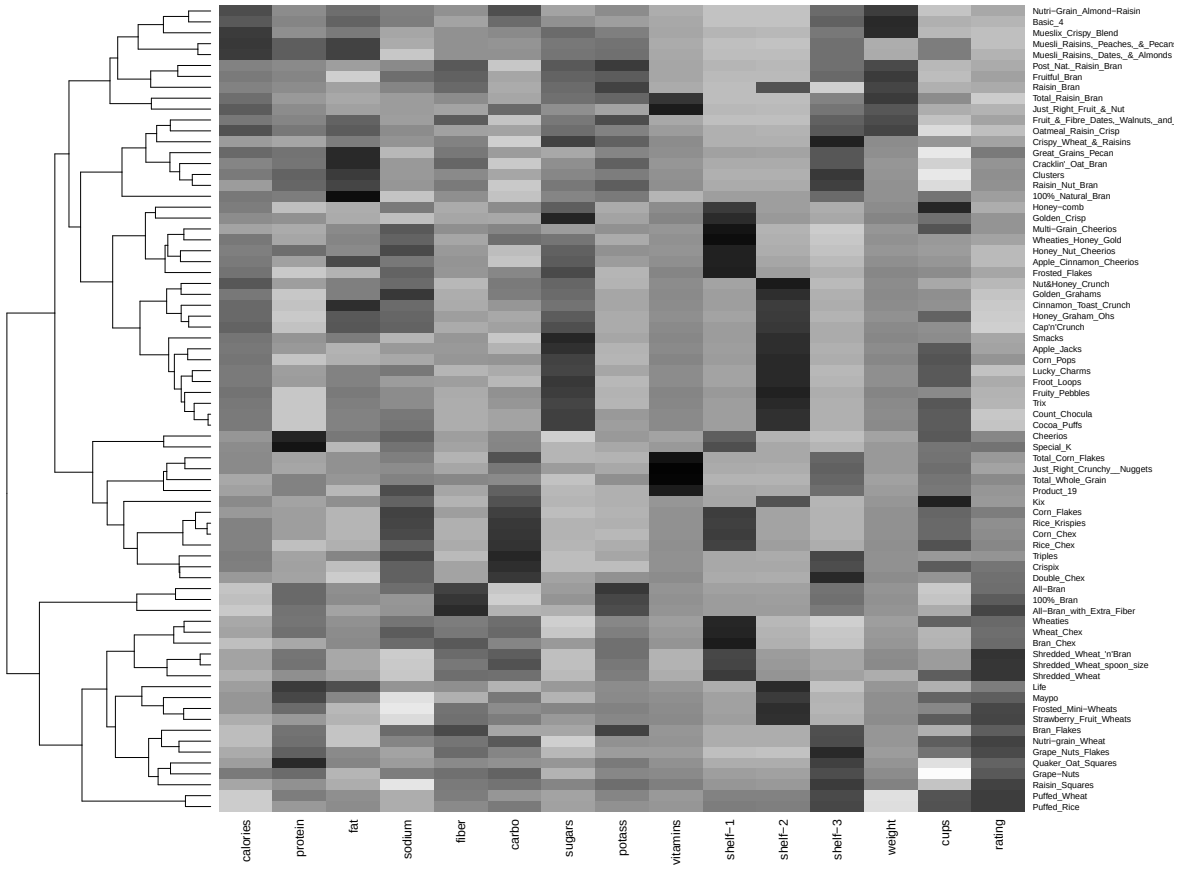
```
## clusters1
##  1  2  3  4
##  3 18 39 14
```

```
#putting the data all together to identify which cluster each cereal belongs to
cereal.cluster <- cbind(clusters1, cereals2)
#plot cluster
plot(hc3, cex= 0.6, hang = -1)
#Plot clusters with borders
rect.hclust(hc3, k = 4, border = 2:7)
abline(h = 15, col = 'purple')
```



#Comment on the structure of the clusters and on their stability #we will use heatmap to calculate cluster centroids to understand the structure of the data

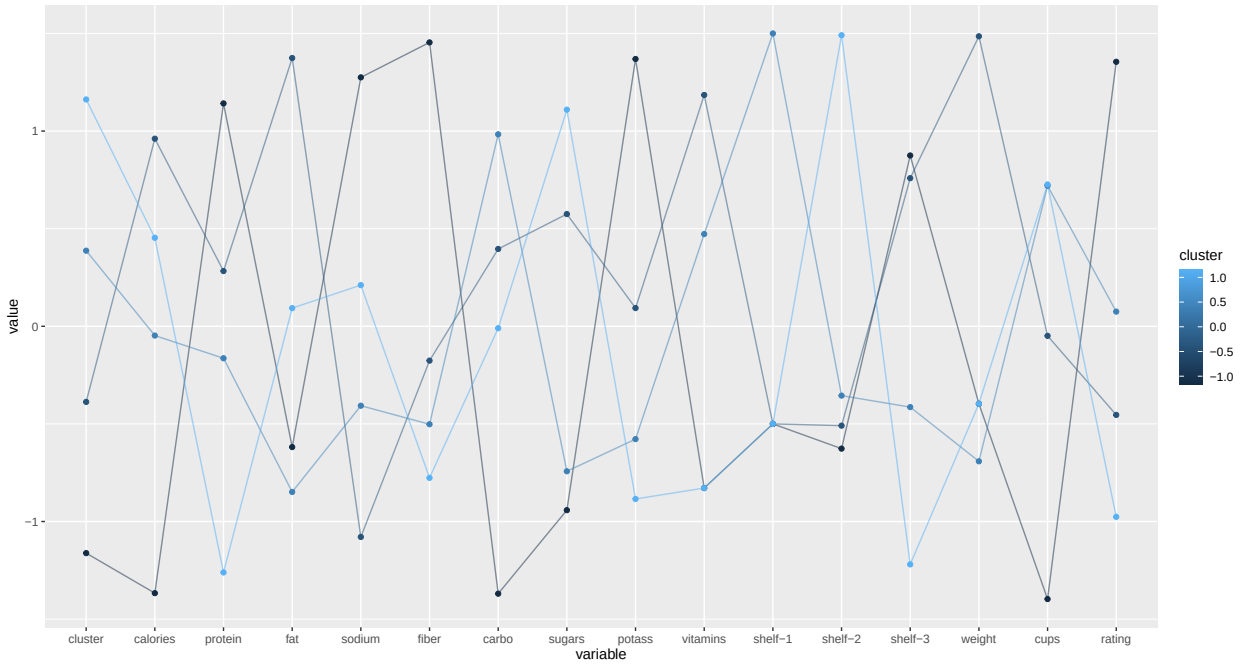
```
#heatmap
cereal.cluster1<- cereal.cluster[,-1]
heatmap(as.matrix(cereal.cluster1), Colv = NA, hclustfun = hclust,
        col=rev(paste("gray",1:99,sep="")))
```



```
#cluster centroids
centroids_cereals <- aggregate(cereals2, by=list(cluster=clusters1), mean)
centroids_cereals
```

```
##   cluster   calories     protein        fat       sodium      fiber      carbo
## 1       1 -2.2018711  1.38174776 -0.3310734  0.172790124  3.6413124 -2.0718749
## 2       2  0.9897070  0.50384944  0.9932203 -0.072199968  0.4318521 -0.1732350
## 3       3 -0.3928795  0.04702452 -0.4838765 -0.002239678 -0.2100400  0.4581808
## 4       4  0.2937990 -1.07489209  0.1418886  0.062041179 -0.7504083 -0.6096570
##       sugars     potass    vitamins    shelf-1    shelf-2    shelf-3     weight
## 1 -0.7894824  2.9837813 -0.18184220 -0.5837690 -0.6044546  1.0484407 -0.2008324
## 2  0.5996897  0.8243795  0.12964675 -0.5837690 -0.4802056  0.9379213  1.0484980
## 3 -0.6071351 -0.3118463  0.01942759  0.5238953 -0.3177261 -0.1757741 -0.3963798
## 4  1.0894502 -0.8305834 -0.18184220 -0.5837690  1.6320274 -0.9409083 -0.2008324
##         cups     rating
## 1 -1.8452553  2.2426479
## 2 -0.4971131 -0.3653894
## 3  0.2715963  0.3975404
## 4  0.2779676 -1.1182150
```

```
ggparcoord((centroids_cereals),
           columns = 1:16, groupColumn = 1,
           showPoints = TRUE,
           alphaLines = 0.5
)
```



#from the heatmap and cluster centroids, it appears that cluster 1 and cluster 4 are extremes, cluster 1 is highest in Protein, fiber, Potass and Ratings, while cluster 4 is lowest in the same variables. cluster 1 is lowest in sugar, cups and shelf.2; cluster 4 is highest in the same variables. cluster2 and cluster 3 are in the middle and somehow paired, cluster 2 is highest in fat and weight while cluster 3 is lowest in the same variables.

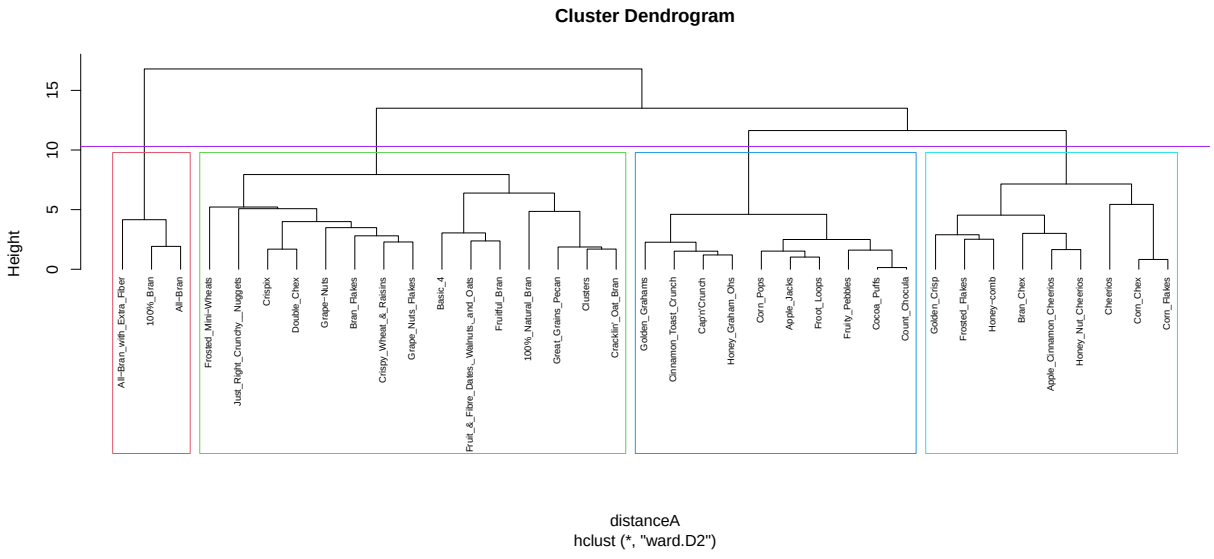
#checking cluster stability - we will do this by partition the data, clustering with one partition and then assigning clusters to the second partition using the closet centroid and then assessing how consistent the cluster assignments are compared to the assignments based on all the data.

```
# partition data into A and B - 50% (data has 74rows)
set.seed(123)
A<-cereals2[1:37,]
B<-cereals2[38:74,]
```

```
#clustering partition A
# Apply hierarchical clustering using Euclidean distance
distanceA <- dist(A, method = "euclidean")
#Hierarchical clustering using Ward (we had determined that ward had the strongest clustering structure
hc_A <- hclust(distanceA, method = "ward.D2")
# Cut tree into 4 groups(we had determined that optimal k =4)
clust_A <- cutree(hc_A, k = 4)
# Number of members in each cluster
table(clust_A)
```

```
## clust_A
##  1  2  3  4
##  3 15  9 10
```

```
#putting the data all together to identify which cluster each cereal belongs to
clust_A1 <- cbind(clust_A, A)
plot(hc_A, cex= 0.6, hang = -1)
#Plot clusters with borders
rect.hclust(hc_A, k = 4, border = 2:7)
abline(h = 10.3, col = 'purple')
```




```
# calculating centroids of partition A
A<-as.data.frame(A)
centroids_A <- aggregate(A, by=list(cluster=clust_A), mean)
```

```r
#assign each record in partition B to the cluster with the closest centroid).
Assign<-data.frame(observations=seq(1,37,1),cluster=rep(0,37))
for(i in 0:37)
{
  x<-as.data.frame(rbind(centroids_A[,-1],B[i,]))
  y<-as.matrix(get_dist(x))
  Assign[i,2]<-which.min(y[4,-4])
}
rownames(Assign) <-rownames(B)
table(Assign)
```

```
##             cluster
## observations 3 4
##            1  1 0
##            2  1 0
##            3  0 1
##            4  0 1
##            5  1 0
##            6  1 0
##            7  1 0
##            8  1 0
##            9  1 0
##            10 0 1
##            11 1 0
##            12 1 0
##            13 1 0
##            14 1 0
##            15 1 0
##            16 1 0
##            17 1 0
##            18 1 0
##            19 1 0
##            20 1 0
##            21 1 0
##            22 1 0
##            23 1 0
##            24 1 0
##            25 1 0
##            26 1 0
##            27 0 1
##            28 1 0
##            29 1 0
##            30 1 0
##            31 1 0
##            32 1 0
##            33 1 0
##            34 0 1
##            35 1 0
##            36 1 0
##            37 0 1
```

#only cluster 3 and 4 have been assigned to partition B

```r
#compare assigned clusters derived from partioned data with assigned clusters derived from complete dat
cbind(cluster_partition=Assign$cluster,cluster_complete=cereal.cluster[38:74,1])
```

```
##                                   cluster_partition cluster_complete
## Just_Right_Fruit_&_Nut                            3                2
## Kix                                               3                3
## Life                                              4                3
## Lucky_Charms                                      4                4
## Maypo                                             3                3
## Muesli_Raisins,_Dates,_&_Almonds                  3                2
## Muesli_Raisins,_Peaches,_&_Pecans                 3                2
## Mueslix_Crispy_Blend                              3                2
## Multi-Grain_Cheerios                              3                3
## Nut&Honey_Crunch                                  4                4
## Nutri-Grain_Almond-Raisin                         3                2
## Nutri-grain_Wheat                                 3                3
## Oatmeal_Raisin_Crisp                              3                2
## Post_Nat._Raisin_Bran                             3                2
## Product_19                                        3                3
## Puffed_Rice                                       3                3
## Puffed_Wheat                                      3                3
## Quaker_Oat_Squares                                3                3
## Raisin_Bran                                       3                2
## Raisin_Nut_Bran                                   3                2
## Raisin_Squares                                    3                3
## Rice_Chex                                         3                3
## Rice_Krispies                                     3                3
## Shredded_Wheat                                    3                3
## Shredded_Wheat_'n'Bran                            3                3
## Shredded_Wheat_spoon_size                         3                3
## Smacks                                            4                4
## Special_K                                         3                3
## Strawberry_Fruit_Wheats                           3                3
## Total_Corn_Flakes                                 3                3
## Total_Raisin_Bran                                 3                2
## Total_Whole_Grain                                 3                3
## Triples                                           3                3
## Trix                                              4                4
## Wheat_Chex                                        3                3
## Wheaties                                          3                3
## Wheaties_Honey_Gold                               4                3
```

```r
table(Assign$cluster==cereal.cluster[38:74,1])
```

```
##
## FALSE  TRUE
##    12    25
```

#59% of data was assigned to the same cluster with both partitioned data and complete data.this represents the stability of the cluster, which is not too high. however, centroids were used for assigning clusters in partitioned dataset whereas in the complete dataset, total figures were used.

```r
#We can also cxheck for cluster stability using Cluster stability assessment
set.seed(123)
#Input the scaled cereals_data
hclust_stability = clusterboot(cereals2, clustermethod=hclustCBI, method="ward.D2", k=4, count = FALSE)
hclust_stability
```

```
## * Cluster stability assessment *
## Cluster method:  hclust/cutree
## Full clustering results are given as parameter result
## of the clusterboot object, which also provides further statistics
## of the resampling results.
## Number of resampling runs:  100
##
## Number of clusters found in data:  4
##
##  Clusterwise Jaccard bootstrap (omitting multiple points) mean:
## [1] 0.4611412 0.7479811 0.6554305 0.9285665
## dissolved:
## [1] 61 11 28  6
## recovered:
## [1] 39 50 28 87
```

#stability values over Values > 0.85 indicates stable clusters. 0.6 - 0.75 indicates that clusters show some pattern but requires further investigations. Cluster 4 is stable, cluster 2 and 3 show pattern but requires further investigation, cluster1 is not stable.

#The elementary public schools would like to choose a set of cereals to include in their daily cafeterias. Every day a different cereal is offered, but all cereals should support a healthy diet. For this goal, you are requested to find a cluster of "healthy cereals." first of all, let us look at the variables to determine which variables are healthy. We will look for a cluster that is high in the healthy variables and low in the unhealthy variables

```r
colnames(cereals2)
```

```
##  [1] "calories" "protein"  "fat"      "sodium"   "fiber"    "carbo"
##  [7] "sugars"   "potass"   "vitamins" "shelf-1"  "shelf-2"  "shelf-3"
## [13] "weight"   "cups"     "rating"
```

```r
#healthy variables are - Protein; Fiber; Potass; Vitamins.
#Unhealthy variables are Sugars; calories;
centroids_cereals
```

```
##   cluster   calories     protein        fat       sodium      fiber      carbo
## 1       1 -2.2018711  1.38174776 -0.3310734  0.172790124  3.6413124 -2.0718749
## 2       2  0.9897070  0.50384944  0.9932203 -0.072199968  0.4318521 -0.1732350
## 3       3 -0.3928795  0.04702452 -0.4838765 -0.002239678 -0.2100400  0.4581808
## 4       4  0.2937990 -1.07489209  0.1418886  0.062041179 -0.7504083 -0.6096570
##       sugars     potass    vitamins    shelf-1    shelf-2    shelf-3     weight
## 1 -0.7894824  2.9837813 -0.18184220 -0.5837690 -0.6044546  1.0484407 -0.2008324
## 2  0.5996897  0.8243795  0.12964675 -0.5837690 -0.4802056  0.9379213  1.0484980
## 3 -0.6071351 -0.3118463  0.01942759  0.5238953 -0.3177261 -0.1757741 -0.3963798
## 4  1.0894502 -0.8305834 -0.18184220 -0.5837690  1.6320274 -0.9409083 -0.2008324
```

```
##         cups     rating
## 1 -1.8452553  2.2426479
## 2 -0.4971131 -0.3653894
## 3  0.2715963  0.3975404
## 4  0.2779676 -1.1182150
```

#cluster 1 is highest in Protein; fiber; Potass and lowest in Sugar and calories. Cluster 1 has the healthy cereals.

```
#Should the data be normalized? If not, how should they be used in the cluster analysis?
#let us look at the summary of the data to determine if it requires normalization
summary(Cereal)
```

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

Yes, the data should be normalized, the range of variables are too wide, for example “vitamin” ranges from 0 to 100 whereas "fiber ranges from 0 to 14. the variables with larger ranges will outweigh variables with smaller ranges. Using these variables without standardization will give the variables with the larger ranges more weight in the analysis.normalization will change the values of numeric columns in the dataset to a common scale.

#What are the main advantages of hierarchical clustering compared to k-means? Hierarchical clustering does not require domain knowledge to determine the value of k, whereas in K-means you need to take a

guess at k value. Hierarchical clustering gives deep insight of each step of converging different clusters; the dendogram helps you to figure out which clusters combination makes sense and where you want to stop. it basically gives you a much bigger picture than k means

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

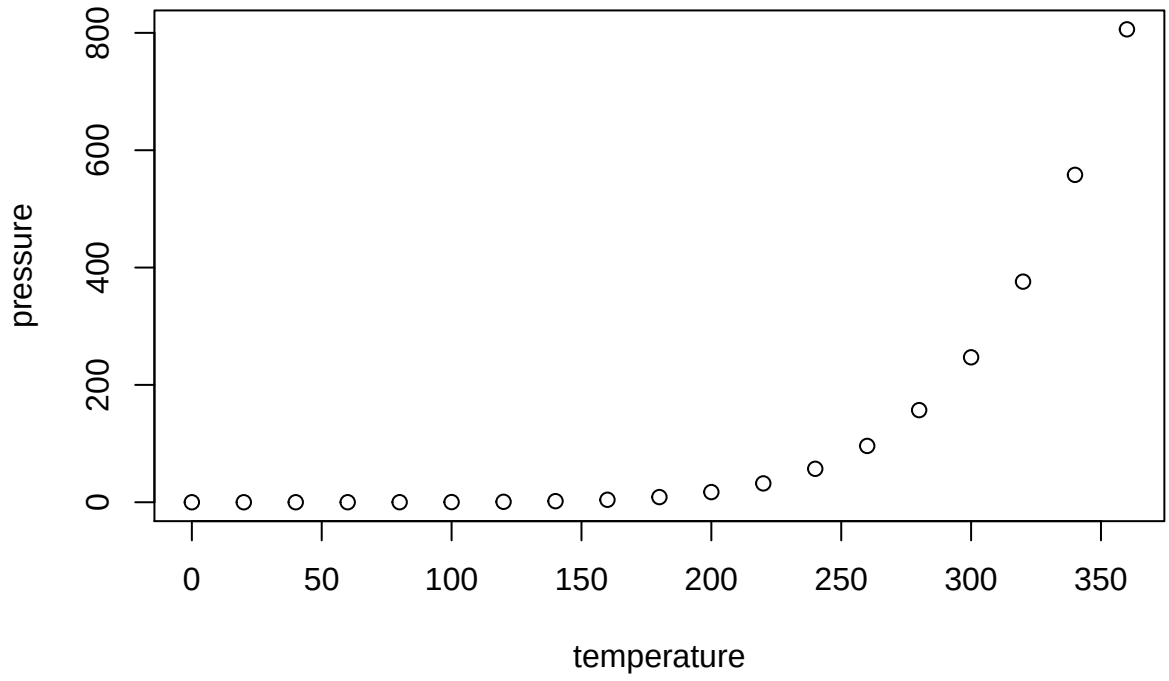
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.