Hierarchical Clustering

Rakhee Moolchandani

12/06/2020

Assignment 5

The purpose of this assignment is to use Hierarchical Clustering.

For this project, we are going to use cereals dataset which includes nutritional information, store display, and consumer ratings for 77 breakfast cereals. For each cereal, there are 16 measurements, which are the following:

- Name: Name of cereal
- mfr: Manufacturer of cereal
  - A = American Home Food Products;
  - G = General Mills
  - K = Kelloggs
  - N = Nabisco
  - P = Post
  - Q = Quaker Oats
  - R = Ralston Purina
- type:
  - cold
  - hot
- calories: calories per serving
- protein: grams of protein
- fat: grams of fat
- sodium: milligrams of sodium
- fiber: grams of dietary fiber
- carbo: grams of complex carbohydrates
- sugars: grams of sugars
- potass: milligrams of potassium
- vitamins: vitamins and minerals - 0, 25, or 100, indicating the typical percentage of FDA recommended
- shelf: display shelf (1, 2, or 3, counting from the floor)
- weight: weight in ounces of one serving
- cups: number of cups in one serving
- rating: a rating of the cereals

Load the required libraries

```
library(readr)
library(tidyverse)
library(factoextra)
library(psych)
library(ggplot2)
library(ggpubr)
library(corrplot)
```

```
library(RColorBrewer)
library(data.table)
library(caret)
library(DMwR)
library(dendextend)
library(cluster)
library(gridExtra)
```

Reading and Understanding the Data

```
# Load the file
Cereals <- read.csv("Cereals.csv")

# Show the first 6 rows
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
```

```
# Show the last 6 rows
tail(Cereals)
```

```
##                    name mfr type calories protein fat sodium fiber carbo sugars
## 72   Total_Whole_Grain   G    C      100       3   1    200     3    16      3
## 73             Triples   G    C      110       2   1    250     0    21      3
## 74                Trix   G    C      110       1   1    140     0    13     12
## 75          Wheat_Chex   R    C      100       3   1    230     3    17      3
## 76            Wheaties   G    C      100       3   1    200     3    17      3
## 77 Wheaties_Honey_Gold   G    C      110       2   1    200     1    16      8
##    potass vitamins shelf weight cups   rating
## 72    110      100     3      1 1.00 46.65884
## 73     60       25     3      1 0.75 39.10617
## 74     25       25     2      1 1.00 27.75330
## 75    115       25     1      1 0.67 49.78744
## 76    110       25     1      1 1.00 51.59219
## 77     60       25     1      1 0.75 36.18756
```

It is important to run the head and tail of the dataframe to confirm that the dataframe is similar among its data points.

Data Exploration and Visualization

```
# To get the total number of rows and columns
dim(Cereals  )
```

```
## [1] 77 16
```

This output shows that the Cereals data frame has 77 data points and 16 variables.

```
# See the data frame structure
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 ...
```

This data frame has 5 numerical variables such calories, protein, fat, etc. and 3 categorical variables, which are following: * Name
* mfr
* type

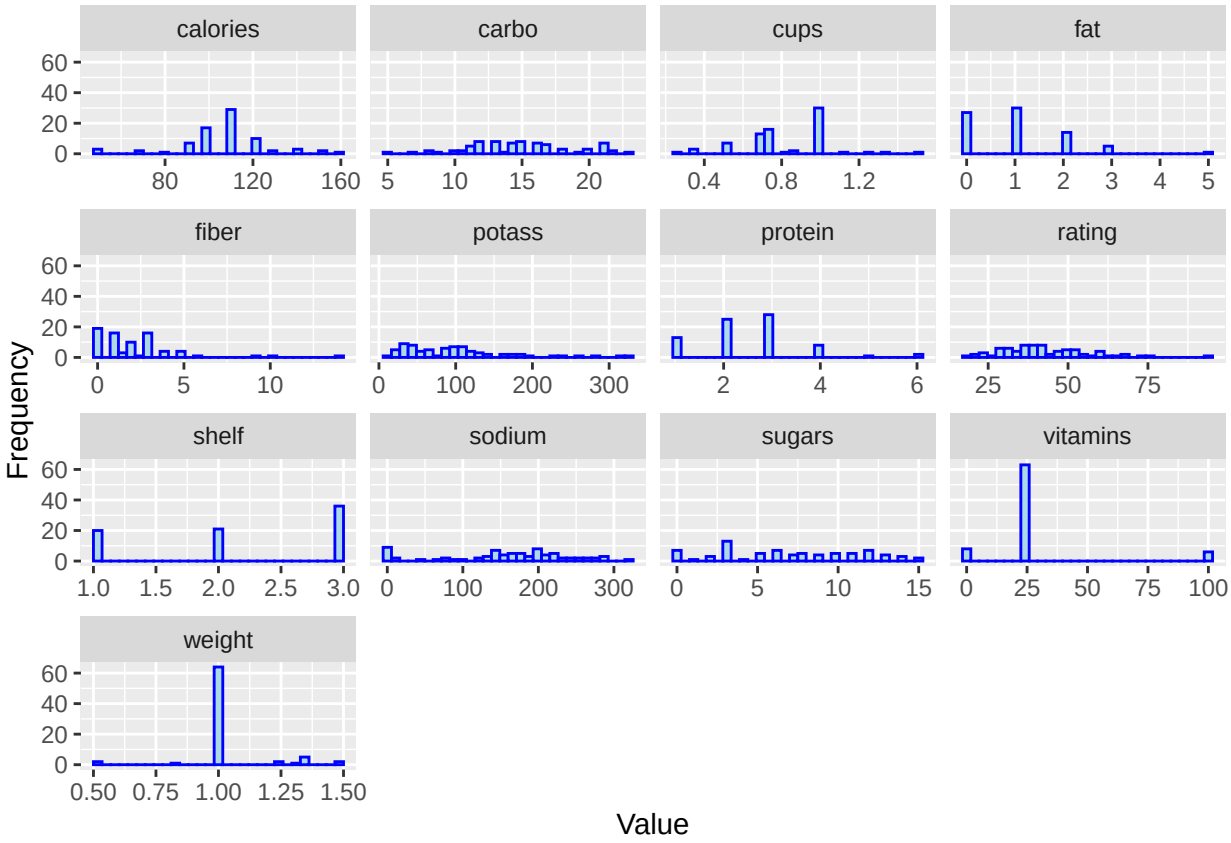
```
# To get descriptive statistics
summary(Cereals)
```

```
##      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
##
```

This help us to see some descriptive statistics and also to determine that variables carbo, sugars, potass have missing values.

```
# Lets visualize the data for each attribute
Cereals %>% gather(Attributes, value, 4:16) %>% ggplot(aes(x=value)) + geom_histogram(fill = "lightblue
```



This allows us to visualize the statistical distribution of the variables. Also, it appears that the shelf column in categorical.

Data Preparation

```
# Lets save the original dataset before making any changes for future reference
MasterCereals <- Cereals
# Convert the names of the breakfast cereals to the row names, as this will later help us in visualizin
```

```
rownames(Cereals) <- Cereals$name
# Drop the name, mfr and type column as they are not used. Also, drop Shelf variable because it's categ
Cereals <- Cereals[,c(-1,-2,-3,-13)]
# Make sure the columns are dropped
head(Cereals)
```

```
##                             calories protein fat sodium fiber carbo sugars potass
## 100%_Bran                         70       4   1    130  10.0   5.0      6    280
## 100%_Natural_Bran                120       3   5     15   2.0   8.0      8    135
## All-Bran                          70       4   1    260   9.0   7.0      5    320
## All-Bran_with_Extra_Fiber         50       4   0    140  14.0   8.0      0    330
## Almond_Delight                   110       2   2    200   1.0  14.0      8     NA
## Apple_Cinnamon_Cheerios          110       2   2    180   1.5  10.5     10     70
##                             vitamins weight cups   rating
## 100%_Bran                         25      1 0.33 68.40297
## 100%_Natural_Bran                  0      1 1.00 33.98368
## All-Bran                          25      1 0.33 59.42551
## All-Bran_with_Extra_Fiber         25      1 0.50 93.70491
## Almond_Delight                    25      1 0.75 34.38484
## Apple_Cinnamon_Cheerios           25      1 0.75 29.50954
```

This shows that the names of the breakfast cereals are converted to the row names and also, the name, type, mfr and shelf columns are dropped.

Scaling or Normalization

The data must be scaled, before measuring any type of distance metric as the variables with higher ranges will significantly influence the distance.

```
# Normalize the data using the scale function
Cereals <- scale(Cereals)

# See the first 6 rows
head(Cereals)
```

```
##                               calories    protein         fat     sodium
## 100%_Bran                   -1.8929836  1.3286071 -0.01290349 -0.3539844
## 100%_Natural_Bran            0.6732089  0.4151897  3.96137277 -1.7257708
## All-Bran                    -1.8929836  1.3286071 -0.01290349  1.1967306
## All-Bran_with_Extra_Fiber   -2.9194605  1.3286071 -1.00647256 -0.2346986
## Almond_Delight               0.1599704 -0.4982277  0.98066557  0.4810160
## Apple_Cinnamon_Cheerios      0.1599704 -0.4982277  0.98066557  0.2424445
##                                   fiber      carbo     sugars     potass
## 100%_Bran                    3.29284661 -2.5087829 -0.2343906  2.5753685
## 100%_Natural_Bran           -0.06375361 -1.7409943  0.2223705  0.5160205
## All-Bran                     2.87327158 -1.9969238 -0.4627711  3.1434645
## All-Bran_with_Extra_Fiber    4.97114672 -1.7409943 -1.6046739  3.2854885
## Almond_Delight              -0.48332864 -0.2054171  0.2223705         NA
## Apple_Cinnamon_Cheerios     -0.27354112 -1.1011705  0.6791317 -0.4071355
##                               vitamins     weight       cups     rating
## 100%_Bran                   -0.1453172 -0.1967771 -2.1100340  1.8321876
## 100%_Natural_Bran           -1.2642598 -0.1967771  0.7690100 -0.6180571
## All-Bran                    -0.1453172 -0.1967771 -2.1100340  1.1930986
## All-Bran_with_Extra_Fiber   -0.1453172 -0.1967771 -1.3795303  3.6333849
## Almond_Delight              -0.1453172 -0.1967771 -0.3052601 -0.5894990
```

```
## Apple_Cinnamon_Cheerios   -0.1453172 -0.1967771 -0.3052601 -0.9365625
```

Remove missing values
Remember that normalizing the data first and then removing the missing values will help us to find the true mean and true standard deviation to accurately run the model.

```
# Find the number of missing values
sum(is.na(Cereals))
```

```
## [1] 4
```

There are 4 missing values in dataset. Lets remove them.

```
# Remove missing values
Cereals <- na.omit(Cereals)
# To get the total number of rows and columns
dim(Cereals)
```

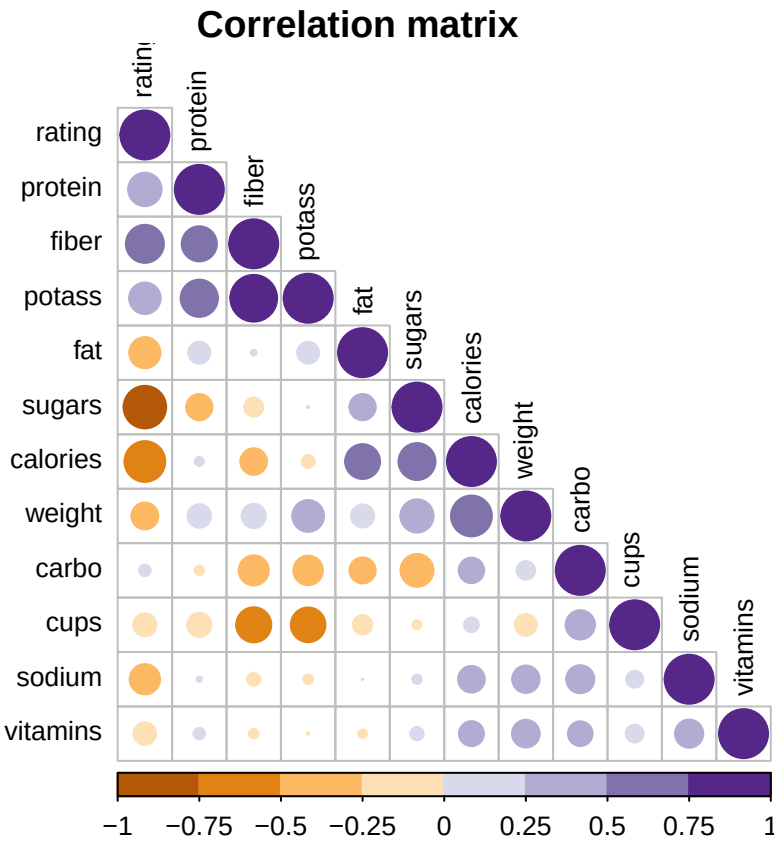
```
## [1] 74 12
```

Here we can see that 3 rows were removed and we only have 12 variables.

Correlation Matrix

What's the relationship between the different scaled attributes? Use `corrplot()` to create correlation matrix.

```
# Multivariate correlation matrix
corrplot(cor(Cereals), type = "lower",main="Correlation matrix", order = "hclust", mar=c(0,0,1,0),tl.ce
```



So, this plot will help us to analyze how data is correlated and it might help us to get some insights. Here we see the following patterns:
* There is a positive correlation between:

Calories and Weight
Calories and Fat
Sugars and Calories
* There is a negative correlation between:
Potassium and Fiber
Sugars and Rating

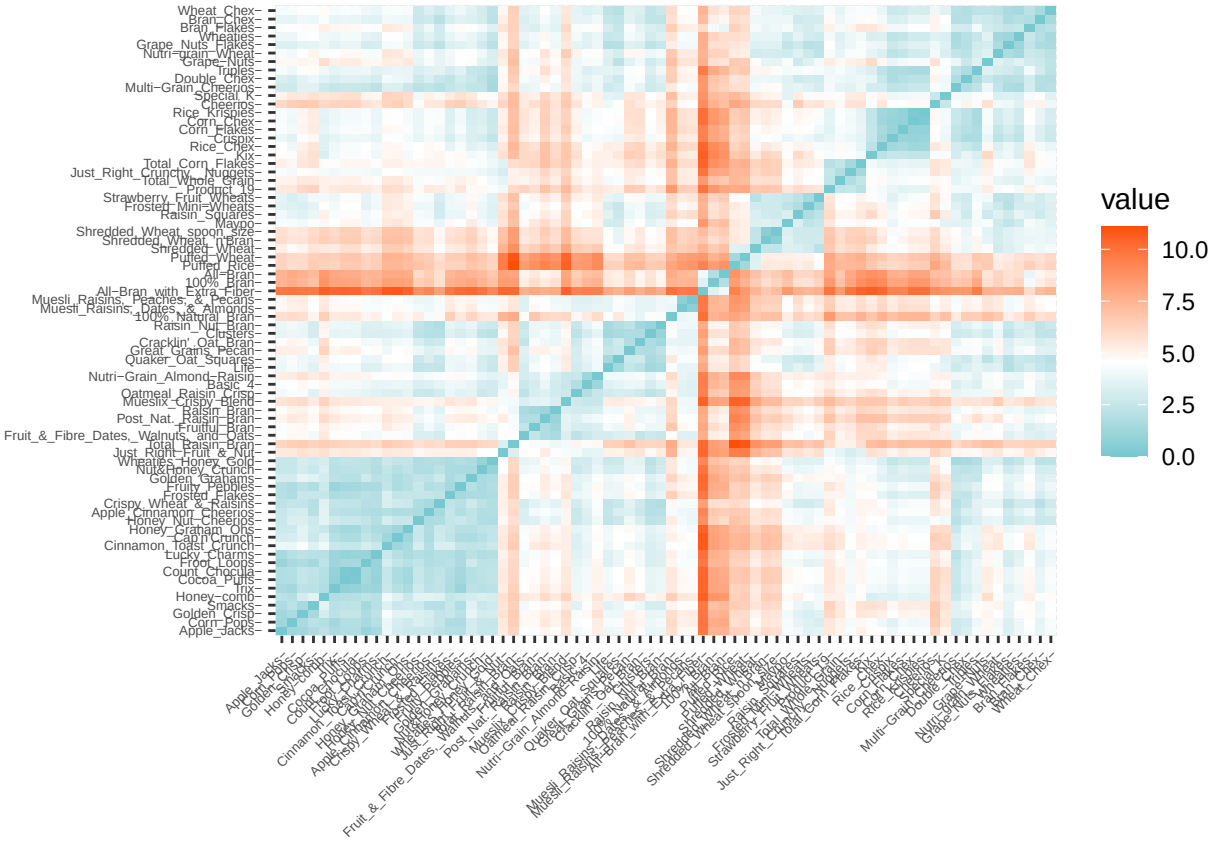
Computing Distance

For computing the distance, we are going to use the get_dist function from the factoextra package in R. The get_dist() function computes a distance matrix between the rows of a data matrix and it uses the Euclidean distance as default.

```
# Computing the distance
dist <- get_dist(Cereals)

# See the first 6 rows
head(dist)
```

```
## [1] 7.546309 1.904228 3.408981 6.651946 7.462129 7.372052
```

```
#  Let's visualize our distances. The fviz_dist() function visualizes a distance matrix
fviz_dist(dist, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07",lab_size = 0.1), lab_s
```

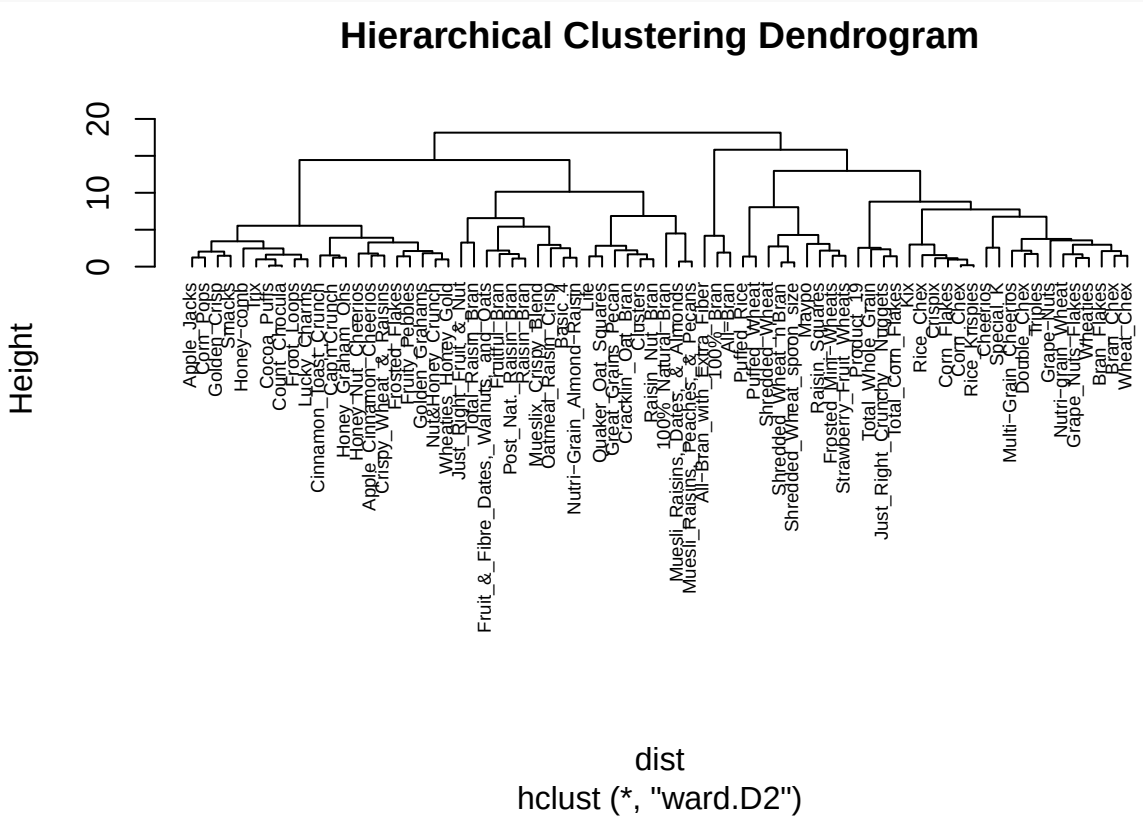


This graph is a distance matrix. As we can see, the diagonal values with blue line are zeros because it is showing the distance between any point against itself. The orange represents the furthest distance between any pair of observations.

Hierarchical Clustering

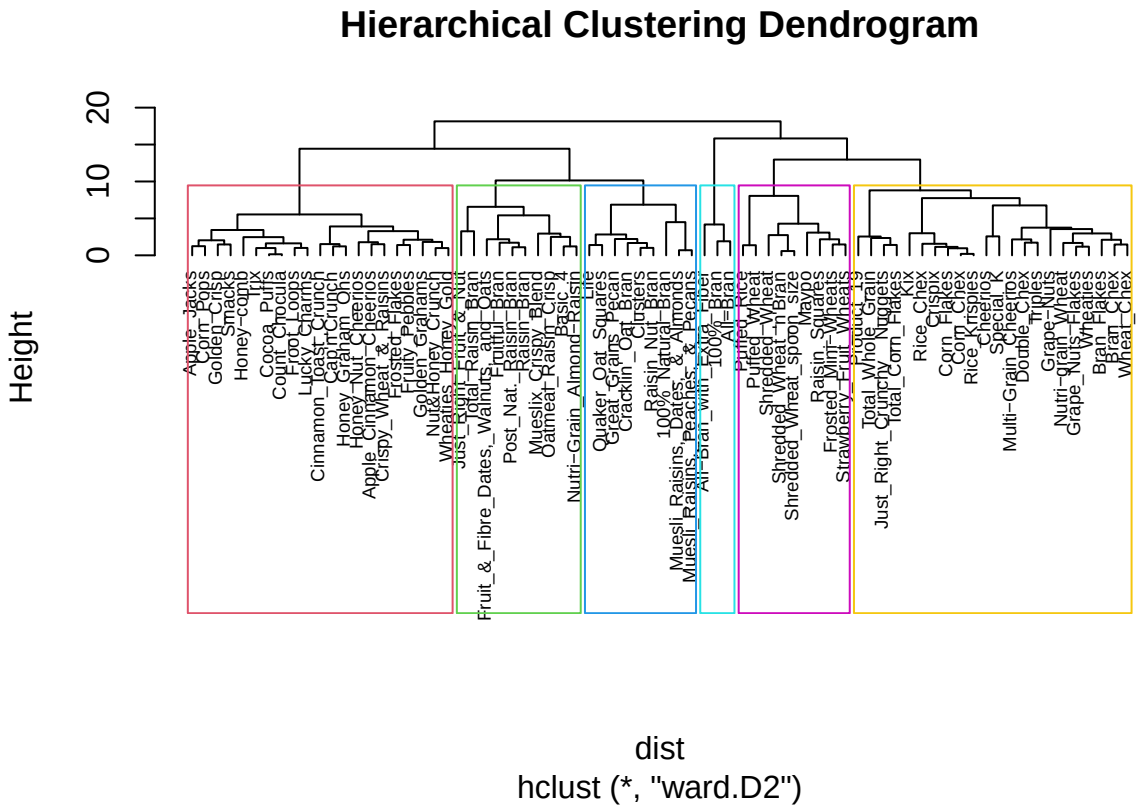
Let's now perform hierarchical clustering using the hclust() function, for which we'll first need to calculate the distance measures using Euclidean method which is already calculated above.

```
# Run the hierarchical clustering using Ward method
hc_fit <- hclust(dist, method = "ward.D2")

# We can display the dendrogram for hierarchical clustering, using the plot() function
plot(hc_fit,cex = 0.6, hang = -1, main = "Hierarchical Clustering Dendrogram")
```



Based on this dendrogram, cluster size 6 seems appropriate.
Let's visualize the 6 clusters.

```
# Plot a new dendrogram, with each of the clusters being displayed in a different, using the A2Rplot()
plot(hc_fit, cex = 0.6, hang = -1, main = "Hierarchical Clustering Dendrogram")
rect.hclust(hc_fit, k = 6, border = 2:10)
```



We can see the 6 clusters are formed in the above dendrogram.

Add Cluster numbers to the dataframe.

```
# Lets Cut the tree to 6 clusters, using the cutree() function
points_hc <- cutree(hc_fit, k = 6)

# Store the clusters in a data frame along with the cereals data
Cereals_Cluster <- cbind(points_hc, Cereals)

# Name the cluster number column as 'Cluster'
colnames(Cereals_Cluster)[1] <- "Cluster"

# Have a look at the head of the new data frame
head(Cereals_Cluster)
```

```
##                            Cluster   calories    protein         fat     sodium
## 100%_Bran                        1 -1.8929836  1.3286071 -0.01290349 -0.3539844
## 100%_Natural_Bran                2  0.6732089  0.4151897  3.96137277 -1.7257708
## All-Bran                         1 -1.8929836  1.3286071 -0.01290349  1.1967306
## All-Bran_with_Extra_Fiber        1 -2.9194605  1.3286071 -1.00647256 -0.2346986
## Apple_Cinnamon_Cheerios          3  0.1599704 -0.4982277  0.98066557  0.2424445
## Apple_Jacks                      3  0.1599704 -0.4982277 -1.00647256 -0.4136273
##                                      fiber      carbo     sugars     potass
## 100%_Bran                       3.29284661 -2.5087829 -0.2343906  2.5753685
## 100%_Natural_Bran              -0.06375361 -1.7409943  0.2223705  0.5160205
## All-Bran                        2.87327158 -1.9969238 -0.4627711  3.1434645
## All-Bran_with_Extra_Fiber       4.97114672 -1.7409943 -1.6046739  3.2854885
## Apple_Cinnamon_Cheerios        -0.27354112 -1.1011705  0.6791317 -0.4071355
```

```
## Apple_Jacks                -0.48332864 -0.9732057  1.5926539 -0.9752315
##                              vitamins     weight       cups     rating
## 100%_Bran                  -0.1453172 -0.1967771 -2.1100340  1.8321876
## 100%_Natural_Bran          -1.2642598 -0.1967771  0.7690100 -0.6180571
## All-Bran                   -0.1453172 -0.1967771 -2.1100340  1.1930986
## All-Bran_with_Extra_Fiber -0.1453172 -0.1967771 -1.3795303  3.6333849
## Apple_Cinnamon_Cheerios    -0.1453172 -0.1967771 -0.3052601 -0.9365625
## Apple_Jacks                -0.1453172 -0.1967771  0.7690100 -0.6756899
```

The Cluster column is added to the dataframe.

AGNES

Lets use Agnes to compare the clustering from single linkage, complete linkage, average linkage, and Ward methods and Choose the best method out of these.

```
# Run agnes with single linkgae
hc_single <- agnes(Cereals, method = "single")

# Run agnes with complete linkgae
hc_complete <- agnes(Cereals, method = "complete")

# Run agnes with average linkgae
hc_average <- agnes(Cereals, method = "average")

# Run agnes with ward method
hc_ward <- agnes(Cereals, method = 'ward')

# Compare Agglomerative Coefficients for each agnes method
m <- c( "single", "complete", "average", "ward")
names(m) <- c( "Single", "Complete", "Average", "Ward")

# function to compute coefficients for all the methods
ac <- function(x) { agnes(Cereals, method = x)$ac}
map_dbl(m, ac)
```

```
##    Single  Complete   Average      Ward
## 0.6091225 0.8508357 0.7888569 0.9088247
```

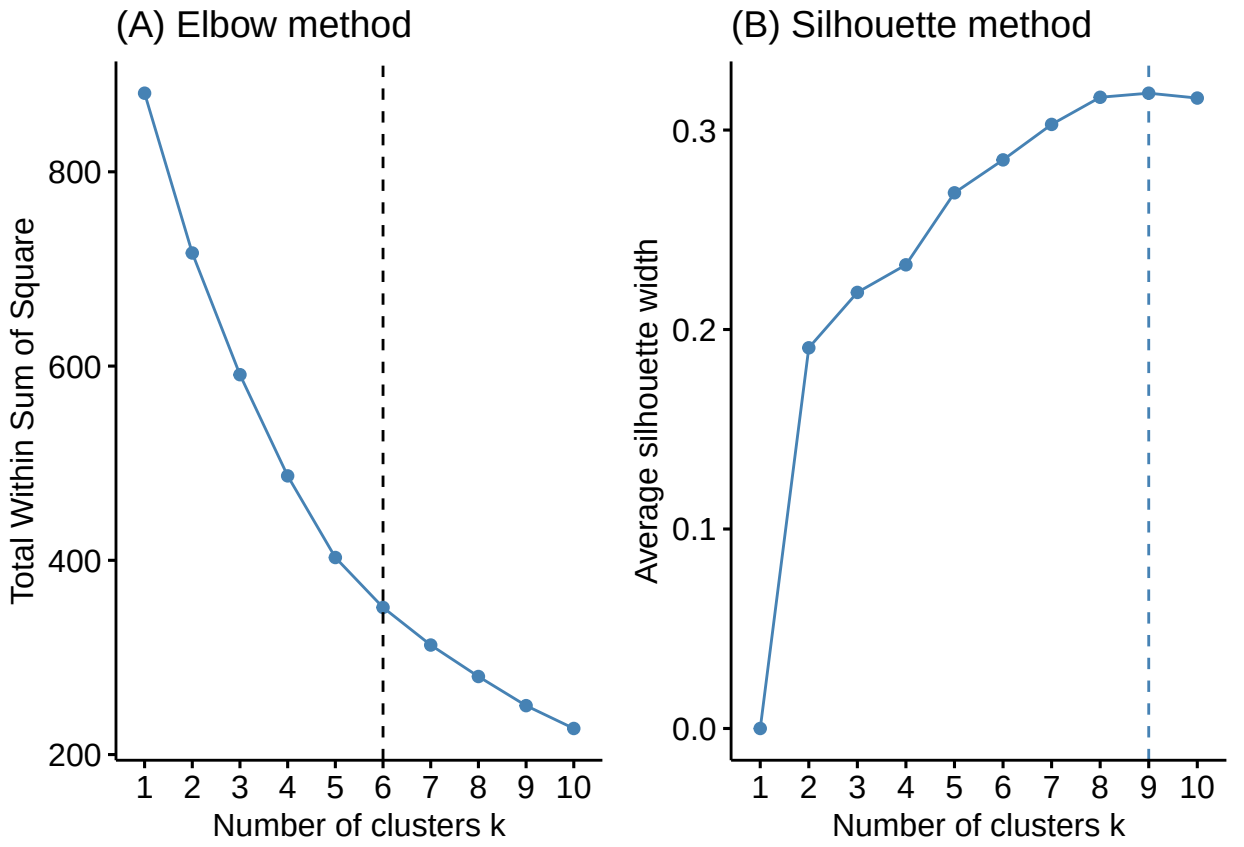
The above table shows the Agglomerative coefficients for Single linkage, Complete linkage, Average linkage and Ward Methods. The Ward method has the highest vale of 0.9088 i.e. the accuracy of 91% and hence it is the best method.

Optimal Clusters

Lets run Elbow and Silhouette Methods to find the optimal values of K for Hierarchical clustering

```
# Elbow method
p1 <- fviz_nbclust(Cereals, FUN = hcut , method = "wss", k.max = 10) +
  ggtitle("(A) Elbow method") + geom_vline(xintercept = 6, linetype = 2)

# Silhouette Method
p2 <- fviz_nbclust(Cereals, FUN = hcut , method = "silhouette", k.max = 10) +
  ggtitle("(B) Silhouette method")
```

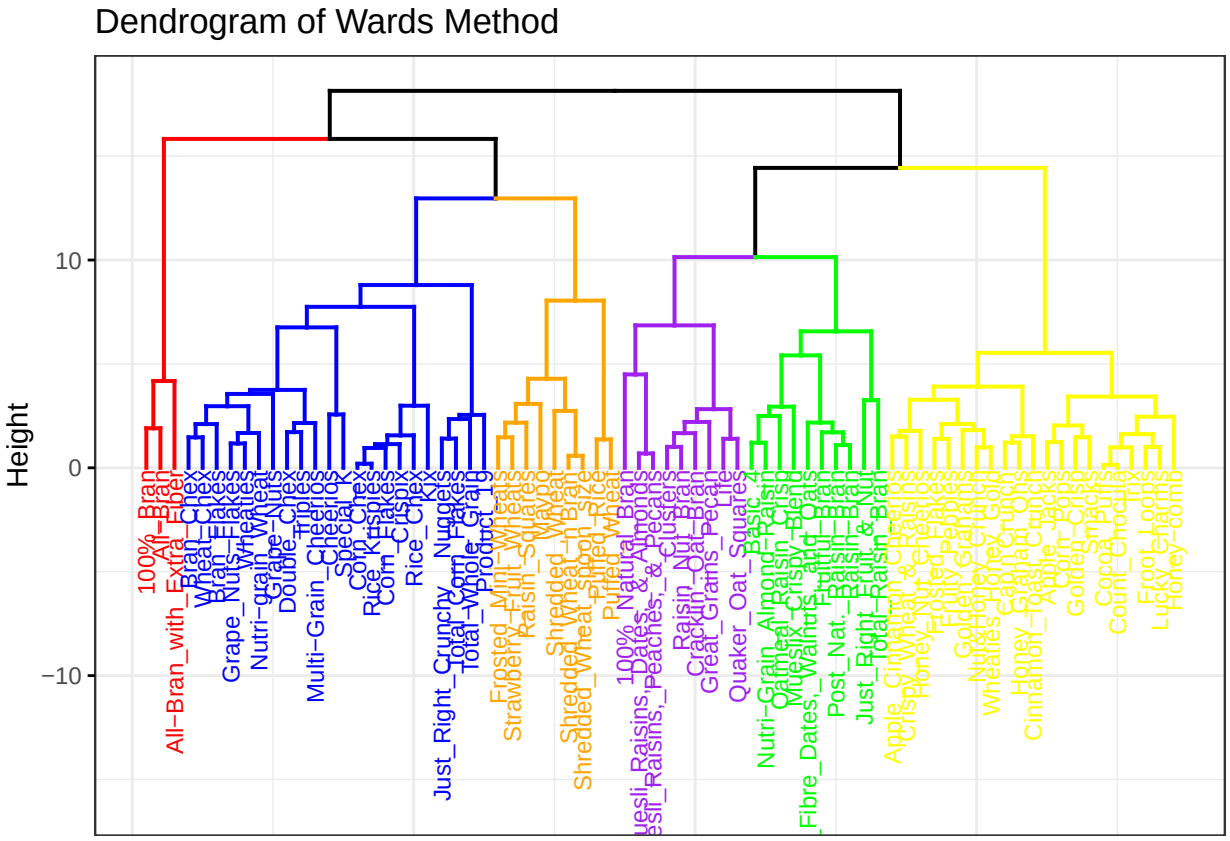
```
# Display plots side by side
grid.arrange(p1, p2, nrow = 1)
```



Here, we can see that the Elbow and Silhouette methods have different values. The Elbow Method shows k=6 whereas the Silhouette Method shows k=9 as the optimal value. Since our Cereal data structure is small, I am choosing k=6 as the optimal value of k to keep the clusters compact.

Now, lets visualize the dendrogram of the best method which is ward

```
fviz_dend(hc_ward, k = 6, main = "Dendrogram of Wards Method", cex = 0.6,
k_colors = c( "red", "blue", "orange", "purple", "green", "yellow"), color_labels_by_k = TRUE,
labels_track_height = 16, ggtheme = theme_bw())
```

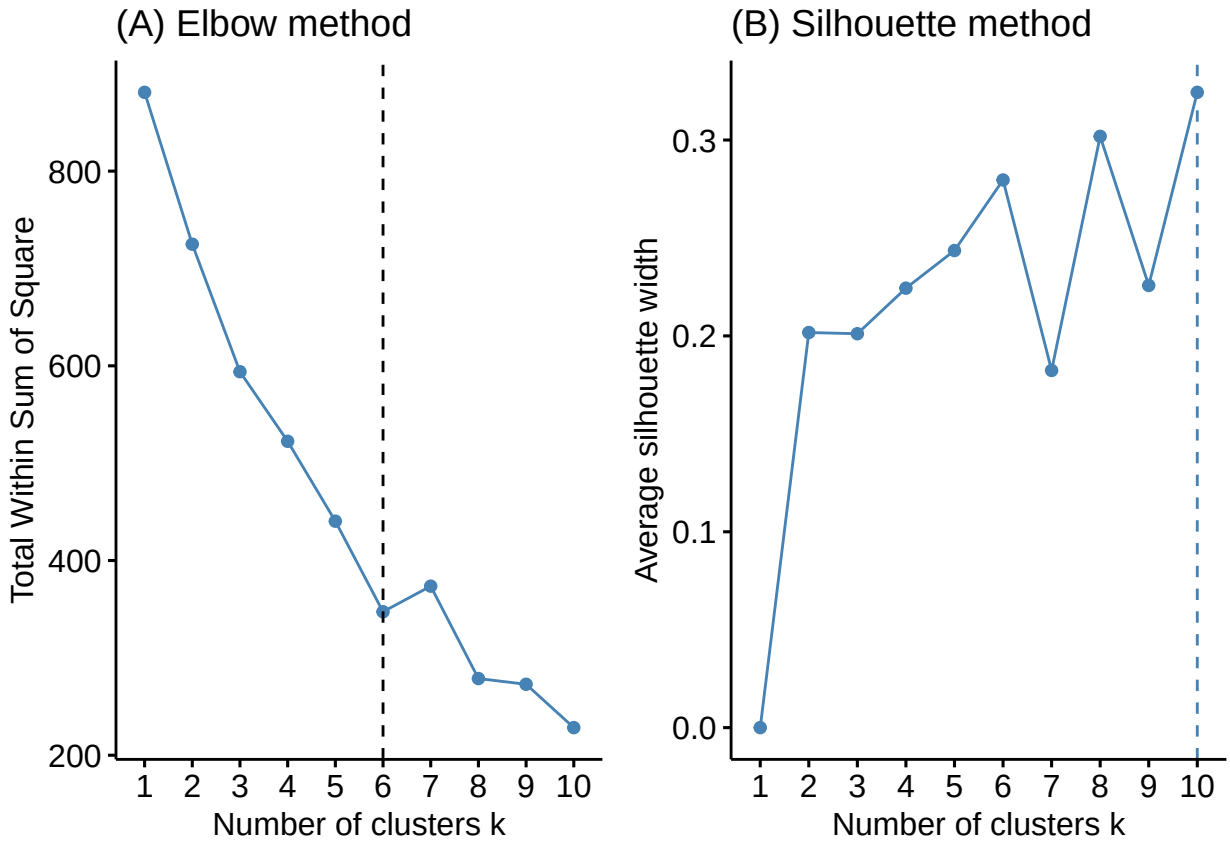


K-Means Clustering

- Lets now run the K-Means clustering and see the difference between the K-Means and the Hierarchical Clustering We are going to run methods to choose our optimal k.

Lets run Elbow and Silhouette Methods to find the optimal values of K for kmeans clustering

```
# Elbow method
p1 <- fviz_nbclust(Cereals, kmeans , method = "wss", k.max = 10) +
  ggtitle("(A) Elbow method") + geom_vline(xintercept = 6, linetype = 2)

# Silhouette Method
p2 <- fviz_nbclust(Cereals, kmeans , method = "silhouette", k.max = 10) +
  ggtitle("(B) Silhouette method")

# Display plots side by side
grid.arrange(p1, p2, nrow = 1)
```



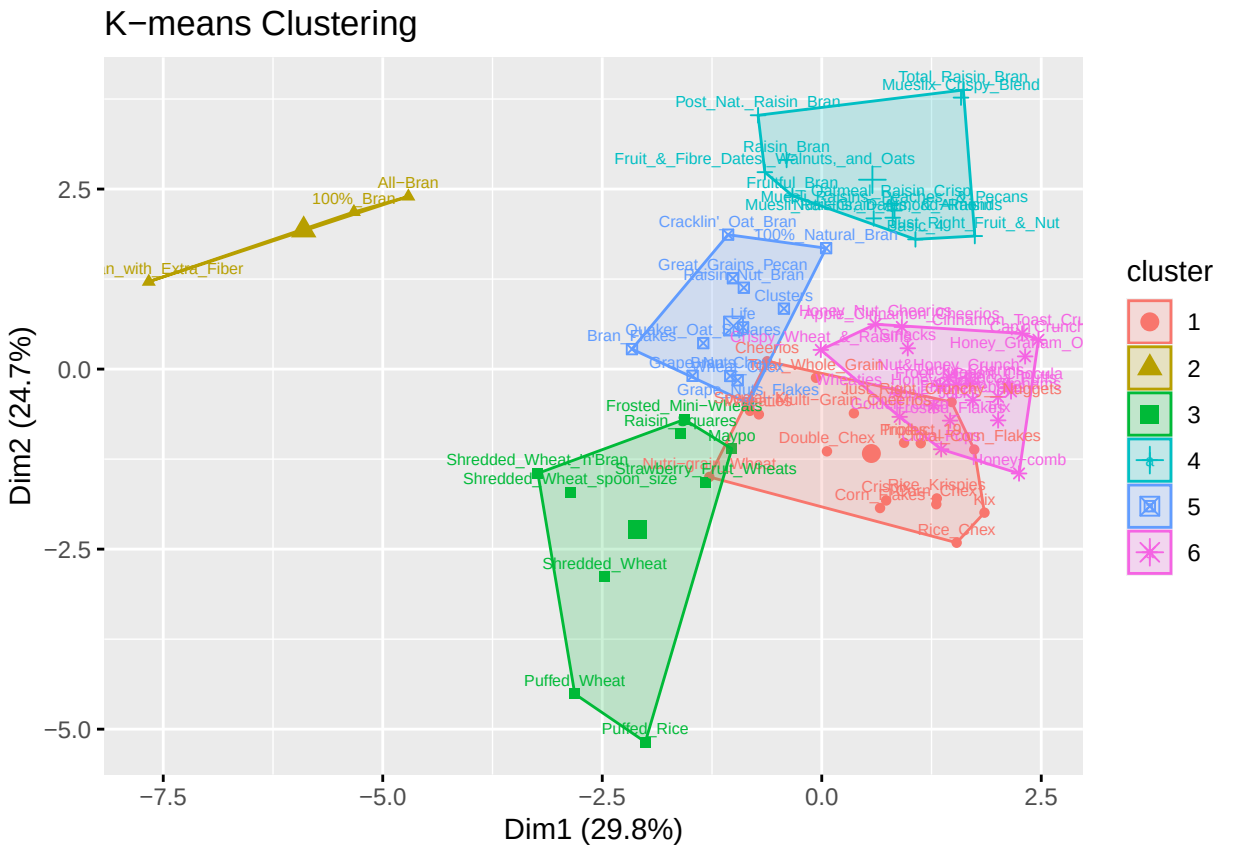
Again, the above graphs show the different values. The Elbow Method shows k=6 whereas the Silhouette Method shows k=10 as the optimal value. I am choosing the same value k=6 as the optimal value of k again.

Run the kmeans

```
# Set seed for reproducibility
set.seed(123)

# Run kmeans model
km_clust <- kmeans(Cereals, 6, nstart = 10)

# See the size of clusters
km_clust$size
```

```
## [1] 17  3  9 12 12 21
```

```
# Visualize the Clusters
fviz_cluster(km_clust, Cereals, main = "K-means Clustering", labelsize = 6)
```



Difference between Hierarchical and K-means clustering

In hierarchical clustering, observations are sequentially grouped to create clusters, based on distances between observations and distances between clusters.It also produces a useful graphical display of the clustering process and results, called a dendrogram.In k -means clustering, observations are allocated to one of a pre-specified set of clusters, according to their distance from each cluster.

Compare Hierarchical and K-means clustering

```
# Following table shows the number of observations in each clusters for Hierarchical clustering
table(points_hc, dnn = "Hierarchical Clusters")
```

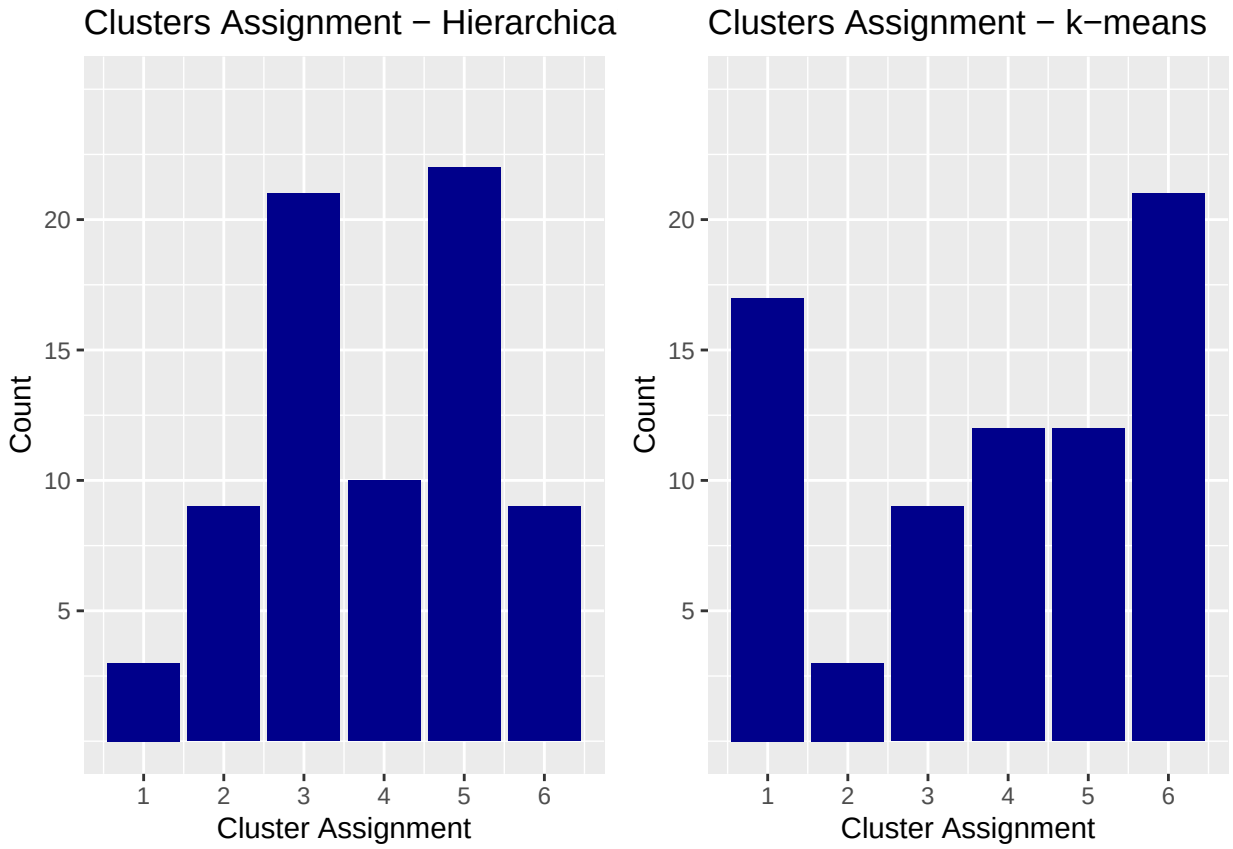
```
## Hierarchical Clusters
##  1  2  3  4  5  6
##  3  9 21 10 22  9
```

```
# Following table shows the number of observations in each clusters for kmeans clustering
table(km_clust$cluster, dnn = "k-means Clusters")
```

```
## k-means Clusters
##  1  2  3  4  5  6
## 17  3  9 12 12 21
```

```
# Lets visualize both the clustering methods
PlotCereals <- na.omit(MasterCereals)

# Plot of hierarchical clustering clusters
pl1 <- ggplot(data = PlotCereals, aes(points_hc)) + geom_bar(fill = "blue4") +
```

```
labs(title="Clusters Assignment - Hierarchical") + labs(x="Cluster Assignment", y="Count") +
guides(fill=FALSE) +
scale_x_continuous(breaks=c(1:6)) + scale_y_continuous(breaks=c(5,10,15,20), limits = c(0,25))

# Plot of K-means clustering clusters
pl2 <- ggplot(data = PlotCereals, aes(km_clust$cluster)) + geom_bar(fill = "blue4") +
labs(title="Clusters Assignment - k-means") + labs(x="Cluster Assignment", y="Count") +
guides(fill=FALSE) +
scale_x_continuous(breaks=c(1:6)) + scale_y_continuous(breaks=c(5,10,15,20), limits = c(0,25))

# plot the graphs for both the clustering methods side by side
grid.arrange(pl1, pl2, nrow = 1)
```



The above table and display shows the number of data points per cluster in Hierarchical and kmeans clustering respectively.

Lets see the cluster of each observation in both the clustering methods

```
# Store the clusters in a data frame along with the cereals data
Cereals_Cluster <- cbind(points_hc, km_clust$cluster, PlotCereals[c(0)])

# Name the cluster number column as 'Cluster'
colnames(Cereals_Cluster)[1] <- "H Cluster"
colnames(Cereals_Cluster)[2] <- "kmeans Cluster"

D1 <- filter(Cereals_Cluster, `H Cluster`==1)
D2 <- filter(Cereals_Cluster, `H Cluster`==2)
D3 <- filter(Cereals_Cluster, `H Cluster`==3)
```

```r
D4 <- filter(Cereals_Cluster, `H Cluster`==4)
D5 <- filter(Cereals_Cluster, `H Cluster`==5)
D6 <- filter(Cereals_Cluster, `H Cluster`==6)

rbind(D1, D2, D3, D4, D5, D6)
```

```
##                                       H Cluster kmeans Cluster
## 100%_Bran                                     1               2
## All-Bran                                      1               2
## All-Bran_with_Extra_Fiber                     1               2
## 100%_Natural_Bran                             2               5
## Clusters                                      2               5
## Cracklin'_Oat_Bran                            2               5
## Great_Grains_Pecan                            2               5
## Life                                          2               5
## Muesli_Raisins,_Dates,_&_Almonds              2               4
## Muesli_Raisins,_Peaches,_&_Pecans             2               4
## Quaker_Oat_Squares                            2               5
## Raisin_Nut_Bran                               2               5
## Apple_Cinnamon_Cheerios                       3               6
## Apple_Jacks                                   3               6
## Cap'n'Crunch                                  3               6
## Cinnamon_Toast_Crunch                         3               6
## Cocoa_Puffs                                   3               6
## Corn_Pops                                     3               6
## Count_Chocula                                 3               6
## Crispy_Wheat_&_Raisins                        3               6
## Froot_Loops                                   3               6
## Frosted_Flakes                                3               6
## Fruity_Pebbles                                3               6
## Golden_Crisp                                  3               6
## Golden_Grahams                                3               6
## Honey_Graham_Ohs                              3               6
## Honey_Nut_Cheerios                            3               6
## Honey-comb                                    3               6
## Lucky_Charms                                  3               6
## Nut&Honey_Crunch                              3               6
## Smacks                                        3               6
## Trix                                          3               6
## Wheaties_Honey_Gold                           3               6
## Basic_4                                       4               4
## Fruit_&_Fibre_Dates,_Walnuts,_and_Oats        4               4
## Fruitful_Bran                                 4               4
## Just_Right_Fruit_&_Nut                        4               4
## Mueslix_Crispy_Blend                          4               4
## Nutri-Grain_Almond-Raisin                     4               4
## Oatmeal_Raisin_Crisp                          4               4
## Post_Nat._Raisin_Bran                         4               4
## Raisin_Bran                                   4               4
## Total_Raisin_Bran                             4               4
## Bran_Chex                                     5               5
## Bran_Flakes                                   5               5
## Cheerios                                      5               1
## Corn_Chex                                     5               1
```

```
## Corn_Flakes                          5           1
## Crispix                              5           1
## Double_Chex                          5           1
## Grape_Nuts_Flakes                    5           5
## Grape-Nuts                           5           5
## Just_Right_Crunchy__Nuggets          5           1
## Kix                                  5           1
## Multi-Grain_Cheerios                 5           1
## Nutri-grain_Wheat                    5           1
## Product_19                           5           1
## Rice_Chex                            5           1
## Rice_Krispies                        5           1
## Special_K                            5           1
## Total_Corn_Flakes                    5           1
## Total_Whole_Grain                    5           1
## Triples                              5           1
## Wheat_Chex                           5           5
## Wheaties                             5           1
## Frosted_Mini-Wheats                  6           3
## Maypo                                6           3
## Puffed_Rice                          6           3
## Puffed_Wheat                         6           3
## Raisin_Squares                       6           3
## Shredded_Wheat                       6           3
## Shredded_Wheat_'n'Bran               6           3
## Shredded_Wheat_spoon_size            6           3
## Strawberry_Fruit_Wheats              6           3
```

On comparing the k-means clusters with that of the Hierarchical clusters, we can say that the optimal number of clusters are quite same in both the methods.
By looking at both plots, the clusters were classified very similar. By looking at the each observation, we can figure out following
* Cluster 1 in hierarchical is similar to cluster 2 in kmeans. And by the table we can confirm that both have same 3 cereals in it.
* Similarly, cluster 3 from hierarchical is similar to cluster 6 from kmeans and vice versa. Again, both have same cereals.
* Cluster 4 is almost similar in both the clustering methods except for 2 more cereals added in kmeans cluster 4.
* Therefore, we can say that the cluster similarity is almost 74%.

Final thoughts and advantages of hierarchical clustering compared to k-means:
Hierarchical clustering may have some benefits over k-means such as not having to pre-specify the number of clusters and the fact that it can produce a nice hierarchical illustration of the clusters (that's useful for smaller data sets). However, from a practical perspective, hierarchical clustering analysis still involves a number of decisions that can have large impacts on the interpretation of the results.
First, like k-means, you still need to make a decision on the dissimilarity measure to use.
Second, you need to make a decision on the linkage method. Each linkage method has different systematic tendencies (or biases) in the way it groups observations and can result in significantly different results.
Third, although we do not need to pre-specify the number of clusters, we often still need to decide where to cut the dendrogram in order to obtain the final clusters to use. So it is still on us to decide the number of clusters.

Cluster Stability

To check the stability of clusters, run the heirachical clustering on Partitioned data set and check the structure of datasets

Partition the data set

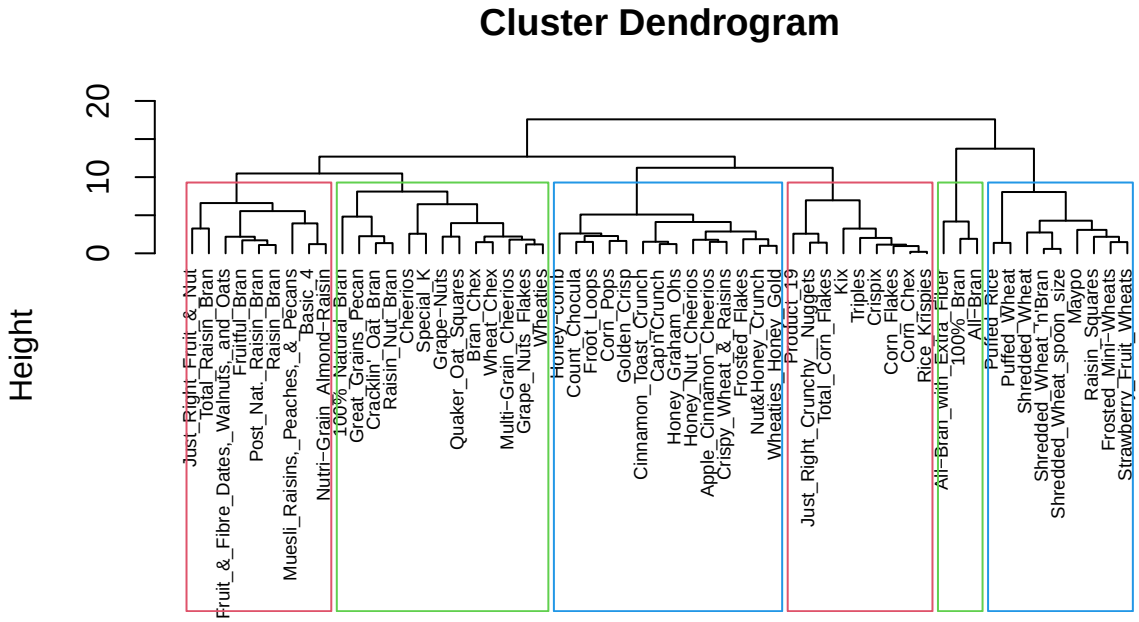
```
# Set seed for reproducibility
set.seed(123)

# Partition the data set
Cereals_index <- createDataPartition(Cereals[,'protein'], p=0.75, list=FALSE)

# Partition A : Train data, Partition B: Test Data
train_data<-Cereals[Cereals_index,] # Partition A
test_data<-Cereals[-Cereals_index,] # Partition B

# Run all the modes and find the best model
hc11<- agnes(train_data,method = "ward")
hc12<-agnes(train_data,method="average")
hc13<-agnes(train_data,method="complete")
hc14<-agnes(train_data,method="single")

cbind(ward=hc11$ac,average=hc12$ac,complete=hc13$ac,single=hc14$ac)
```

```
##           ward   average  complete    single
## [1,] 0.8975757 0.7705437 0.8361074 0.6659841
```

Again, we can see that the ward method is the best method.

Run the Hierarchical model using Ward method

```
# Calculate the distance using the Euclidean method
Newdist <- dist(train_data, method = "euclidean")

# Run Hierarchical clustering using the distance calculated
hc_fit1 <- hclust(Newdist, method = "ward.D2")

#We can display the dendrogram for hierarchical clustering, using the plot() function
plot(hc_fit1,cex = 0.6, hang = -1)
rect.hclust(hc_fit1, k = 6, border = 2:4)
```




```
# Cut the tree into 6 clusters for analysis
clust2<-cutree(hc_fit1, k=6)

# Add the assigned cluster to the pre-processed data set
result<-as.data.frame(cbind(train_data,clust2))

# Determine centroids for all 6 clusters
centroid1<-data.frame(column=seq(1,12,1),mean=rep(0,12))
centroid2<-data.frame(column=seq(1,12,1),mean=rep(0,12))
centroid3<-data.frame(column=seq(1,12,1),mean=rep(0,12))
centroid4<-data.frame(column=seq(1,12,1),mean=rep(0,12))
centroid5<-data.frame(column=seq(1,12,1),mean=rep(0,12))
centroid6<-data.frame(column=seq(1,12,1),mean=rep(0,12))
for(i in 1:12)
{
centroid1[i,2]<-mean(result[result$clust2==1,i])
centroid2[i,2]<-mean(result[result$clust2==2,i])
centroid3[i,2]<-mean(result[result$clust2==3,i])
centroid4[i,2]<-mean(result[result$clust2==4,i])
centroid5[i,2]<-mean(result[result$clust2==5,i])
centroid6[i,2]<-mean(result[result$clust2==6,i])
}

# Combine the centroids of all the variables
centroidResult<-t(cbind(centroid1$mean,centroid2$mean,centroid3$mean,centroid4$mean,centroid5$mean,centr
colnames(centroidResult)<-colnames(Cereals)

# Assign the clusters to the test data based on the minimum distance to cluster centers
```

```
Dumm1 <- data.frame(data=seq(1,17,1), cluster=rep(0,17))
for(i in 1:17)
{
R <- as.data.frame(rbind(centroidResult,test_data[i,]))
U <- as.matrix(get_dist(R))
Dumm1[i,2] <- which.min(U[7,-7])
}

# Combine partitions for comparison to original clusters
result1<-as.data.frame(cbind(test_data,Dumm1$cluster))
colnames(result1)[13] <- "clust2"
Finalresult <- rbind(result,result1)

# Compare the number of matching assignments to see the stability of the clusters.
table(points_hc == Finalresult$clust2)
```

```
##
## FALSE  TRUE
##    48    26
```

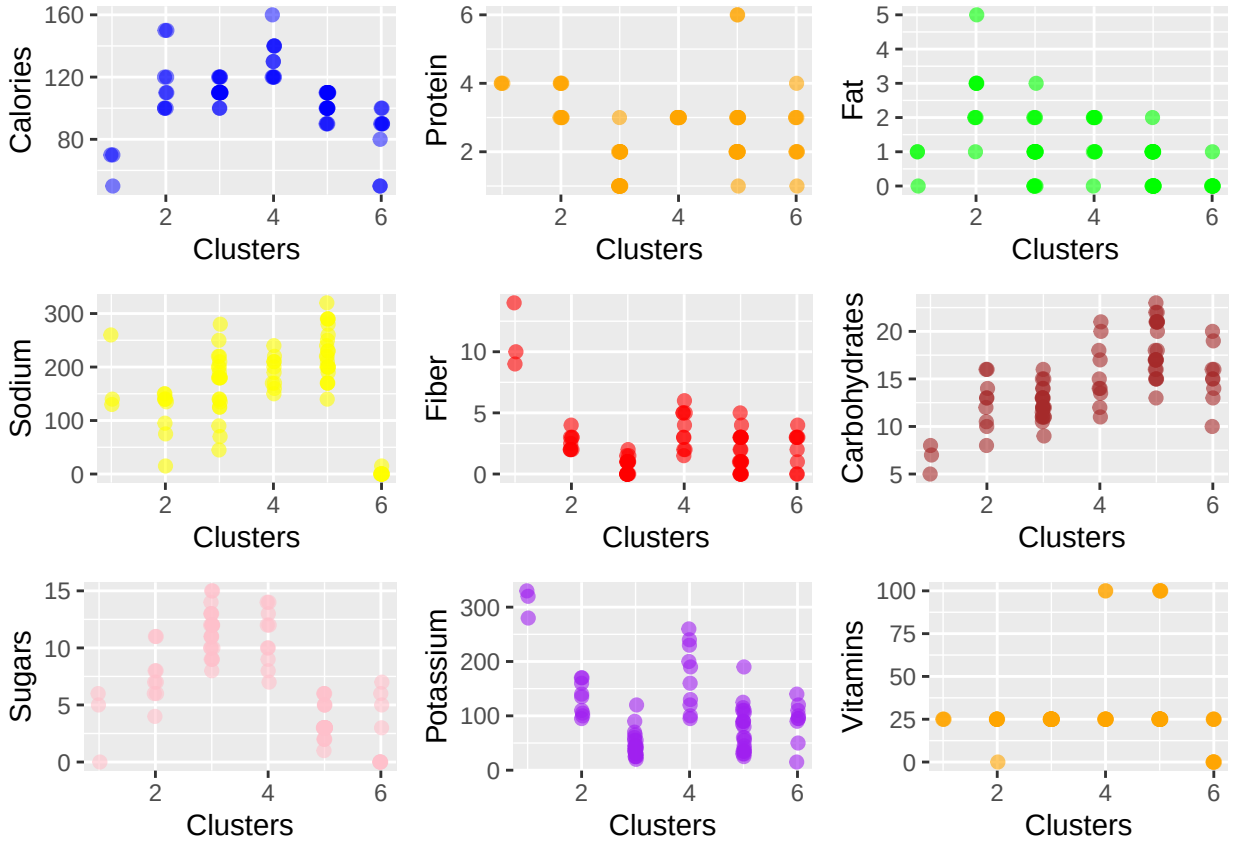
From this result, it can be stated that the clusters are not very stable. With 75% of the data available, the resulting assignments were only identical for 26 out of the 74 observations. This results in a 35% similarity of cluster assignment.

Find the cluster of Healthy Cereals
In this case, normalizing the data would not be appropriate because the normalizing of the cereal nutritional information is based on the sample of cereal being analyzed. Therefore, the gathered dataset could include only cereals with very high sugar content and very low fiber, iron, and other nutritional information. Once it is normalized across the sample set, it is impossible to state how much nutrition the cereal will give a child.

Healthy Cereal Cluster

```
# Read the dataset and remove any missing values and add the cluster column
Cereals_data <- MasterCereals
Cereals_data <-na.omit(Cereals_data)
Cereals_data<-cbind(Cereals_data,points_hc)

# Lets compare all the variables in each cluster
# cluster vs calories
plot1 <- Cereals_data %>%
    ggplot(aes(x = points_hc, y = calories)) +
    geom_jitter(width = .025, height = 0, size = 2, alpha = .5, color = "blue") +
  labs(x = "Clusters", y="Calories")

# cluster vs Protein
plot2 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = protein)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "orange") +
  labs(x = "Clusters", y="Protein")

# cluster vs fat
plot3 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = fat)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "green") +
```

```
  labs(x = "Clusters", y="Fat")

# cluster vs sodium
plot4 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = sodium)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "yellow") +
  labs(x = "Clusters", y="Sodium")

# cluster vs fiber
plot5 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = fiber)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "red") +
  labs(x = "Clusters", y="Fiber")

# cluster vs carbs
plot6 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = carbo)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "brown") +
  labs(x = "Clusters", y="Carbohydrates")

# cluster vs sugar
plot7 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = sugars)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "pink") +
  labs(x = "Clusters", y="Sugars")

# cluster vs potassium
plot8 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = potass)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "purple") +
  labs(x = "Clusters", y="Potassium")

# cluster vs vitamins
plot9 <-  Cereals_data %>%
  ggplot(aes(x = points_hc, y = vitamins)) +
    geom_jitter(width = .02, height = 0, size = 2, alpha = .6,  color = "orange") +
  labs(x = "Clusters", y="Vitamins")

# plot the graphs side by side
grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, plot7, plot8, plot9)
```

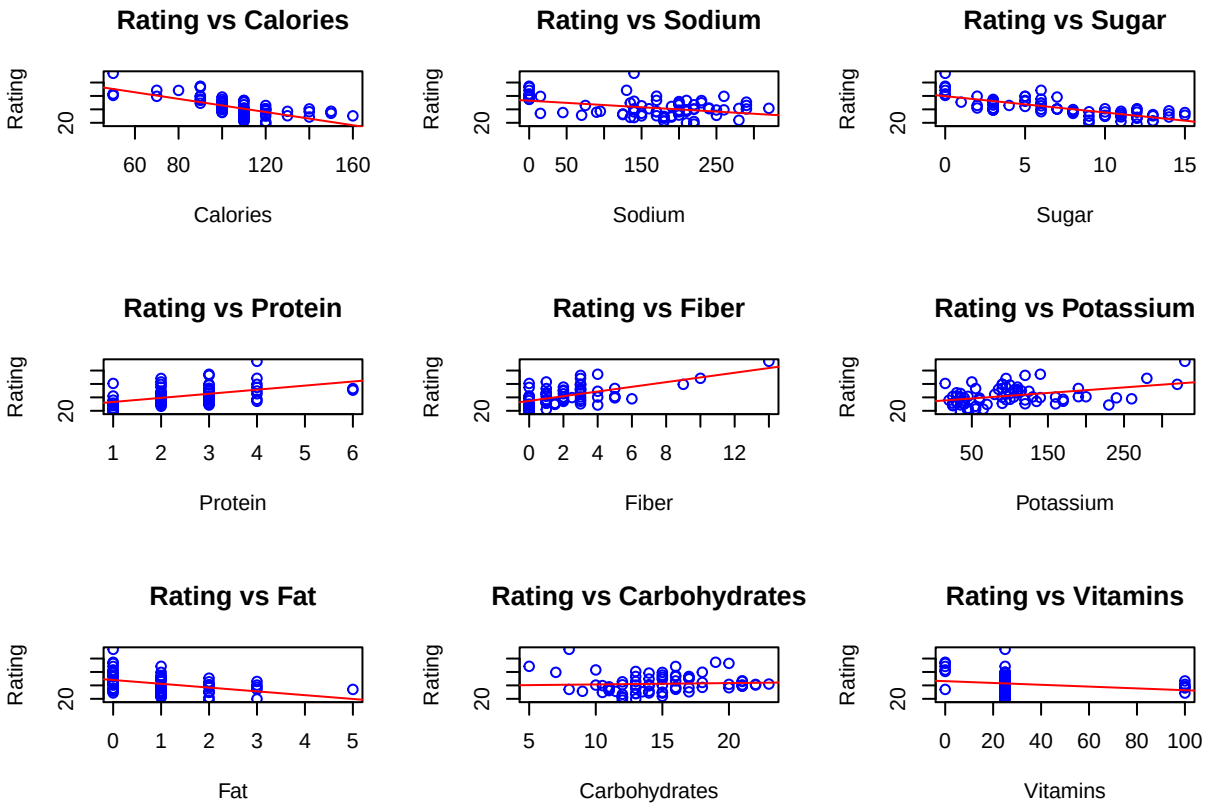


This output shows that the Cluster 1 has set of cereals which have less calories, less fat, more fiber, less carbs, less sugars as compared to other clusters. This cluster looks healthy option. Lets also check Which nutrients are essential for a nutritious breakfast per rating.

```
# Lets see the relation between ratings and all the nutrients
par(mfcol=c(3,3))

# rating vs calories
plot(Cereals_data$rating~calories,
     data = Cereals_data,
     xlab="Calories",
     ylab="Rating",
     main="Rating vs Calories",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$calories), col="red")

# rating vs protein
plot(Cereals_data$rating~protein,
     data = Cereals_data,
     xlab="Protein",
     ylab="Rating",
     main="Rating vs Protein",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$protein), col="red")

# rating vs fat
plot(Cereals_data$rating~fat,
     data = Cereals_data,
```

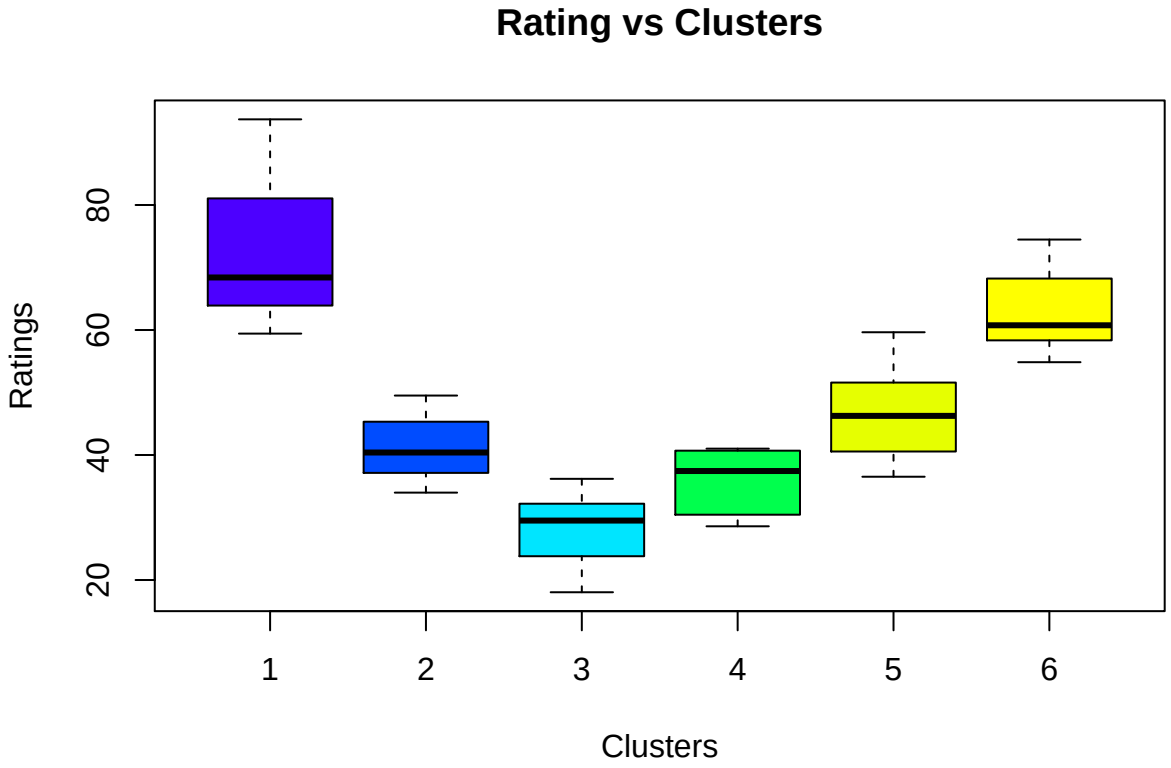
```
     xlab="Fat",
     ylab="Rating",
     main="Rating vs Fat",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$fat), col="red")

# rating vs sodium
plot(Cereals_data$rating~sodium,
     data = Cereals_data,
     xlab="Sodium",
     ylab="Rating",
     main="Rating vs Sodium",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$sodium), col="red")

# rating vs fiber
plot(Cereals_data$rating~fiber,
     data = Cereals_data,
     xlab="Fiber",
     ylab="Rating",
     main="Rating vs Fiber",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$fiber), col="red")

# rating vs carbo
plot(Cereals_data$rating~carbo,
     data = Cereals_data,
     xlab="Carbohydrates",
     ylab="Rating",
     main="Rating vs Carbohydrates",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$carbo), col="red")

# rating vs sugars
plot(Cereals_data$rating~sugars,
     data = Cereals_data,
     xlab="Sugar",
     ylab="Rating",
     main="Rating vs Sugar",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$sugars), col="red")

# rating vs potassium
plot(Cereals_data$rating~potass,
     data = Cereals_data,
     xlab="Potassium",
     ylab="Rating",
     main="Rating vs Potassium",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$potass), col="red")

# rating vs vitamins
plot(Cereals_data$rating~vitamins,
```

```
     data = Cereals_data,
     xlab="Vitamins",
     ylab="Rating",
     main="Rating vs Vitamins",
     col="blue")
abline(lm(Cereals_data$rating~Cereals_data$vitamins), col="red")
```



We can see that the rating is dependent on the nutrients of the cereals. For example:
* As the calories, sodium, sugar, fat content increases, the rating decreases.
* As the protein, fiber content increases, the rating also increases.
Therefore, we can say that there is significance of rating with reagrds to clusters.

```
# Lets see the cluster with the highest rating
boxplot(rating~points_hc,
        data = Cereals_data,
        xlab = "Clusters",
        ylab = "Ratings",
        main = "Rating vs Clusters",
        col = topo.colors(7))
```



Here also,from the above graph it appears that the Cluster 1 has highest ratings. So Cluster 1 is a cluster of "Healthy Cereals"