Assignment 5

MUNERAH

11/23/2021

```
## Load the file
library(readxl)
Cereals <- read.csv("C:/Users/mnooo/Desktop/Datasets/Cereals.csv")
```

```
# Loading the libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.1.2
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --
## v broom        0.7.9      v rsample      0.1.1
## v dials        0.0.10     v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.4
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.1.7      v yardstick    0.0.9
## v recipes      0.1.17
```

```
## Warning: package 'dials' was built under R version 4.1.2
```

```
## Warning: package 'infer' was built under R version 4.1.2
```

```
## Warning: package 'modeldata' was built under R version 4.1.2
```

```
## Warning: package 'parsnip' was built under R version 4.1.2
```

```
## Warning: package 'rsample' was built under R version 4.1.2
```

```
## Warning: package 'tune' was built under R version 4.1.2
```

```
## Warning: package 'workflows' was built under R version 4.1.2
```

```
## Warning: package 'workflowsets' was built under R version 4.1.2
```

```
## Warning: package 'yardstick' was built under R version 4.1.2
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

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

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

```r
library(knitr)
```

```
## Data preprocessing
# Transfer the cereal name into a row name
Cereals <- data.frame(Cereals,row.names ='name')

# Remove missing observations with unused features
Cereals <- Cereals %>%
        filter(complete.cases(.)) %>%
        select(-c(mfr,type,shelf))

#Scale Numeric variables [Necessary for clustering]
scaled <-data.frame(scale(Cereals))
```

Apply hierarchical clustering to the data using Euclidean distance to the normalised mesurements.

In this part, we shall be using AGNES to compare the clustering from single linkage, complete linkage, average linkage, and Ward. We shall choose the best method.

```
#Hierarchical Clustering
metd<-c('single','complete','average','ward')
names(metd)<- c('single','complete','average','ward')

# we write a function which computes agglomerative clustering
AGNES<- function(x){
        agnes(scaled, method = x)$ac
}
```

```
#calculate agglomerative coefficient for each clustering linkage method
sapply(metd, AGNES)
```

```
##    single  complete   average      ward
## 0.6072384 0.8469328 0.7881955 0.9087265
```

We can see that Ward's minimum variance method provides the highest agglomerative coefficient. We will use it as our final model for hierarchical clustering.

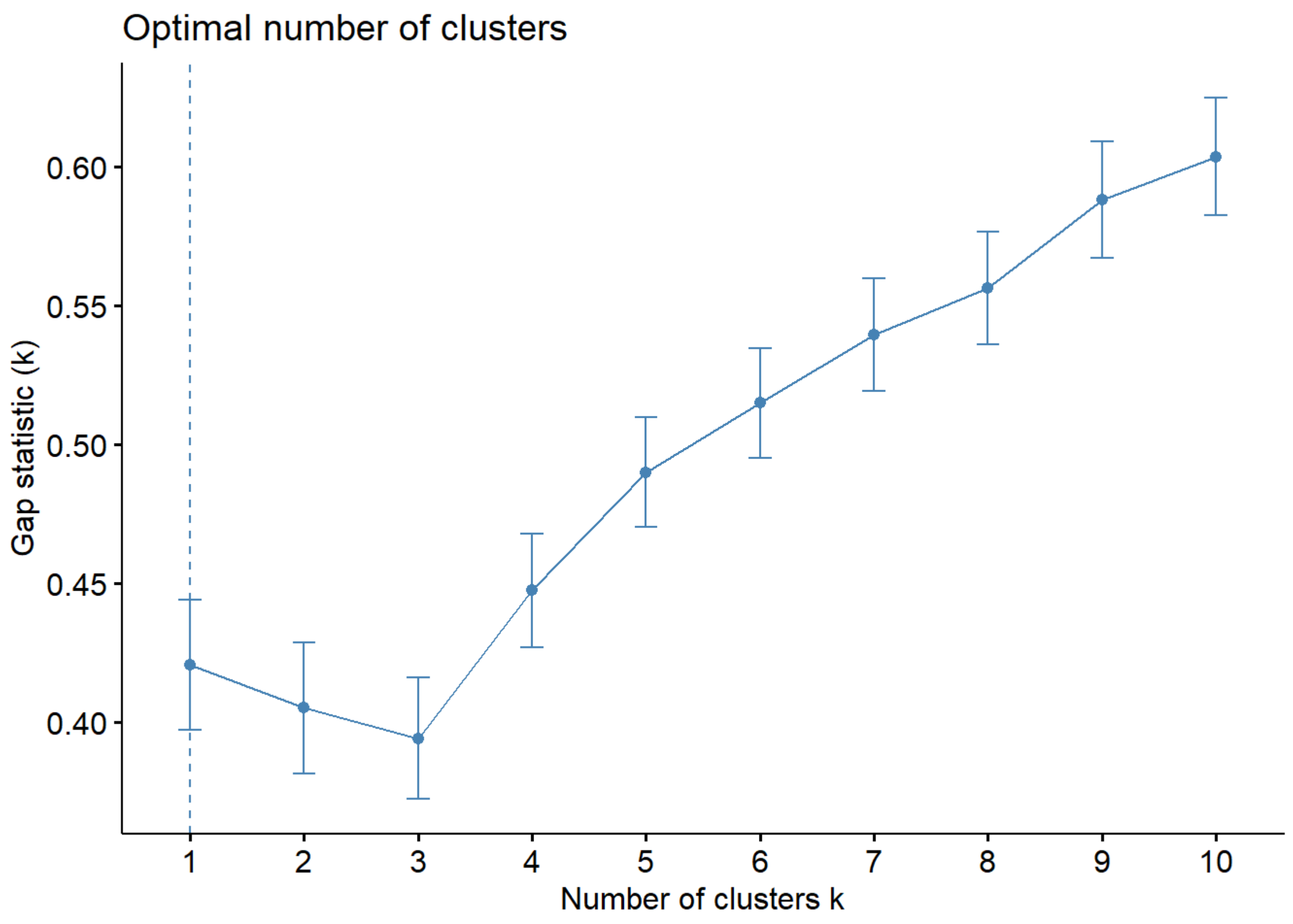
```
set.seed(100)
#Compute th euclidean distance matrix
dm<- dist(scaled, method = "euclidean")

#We perform hierarchical clustering using ward's method
wClust<- hclust(dm, method = "ward.D2")
wClust
```

```
##
## Call:
## hclust(d = dm, method = "ward.D2")
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 74
```

How many clusters will you choose?

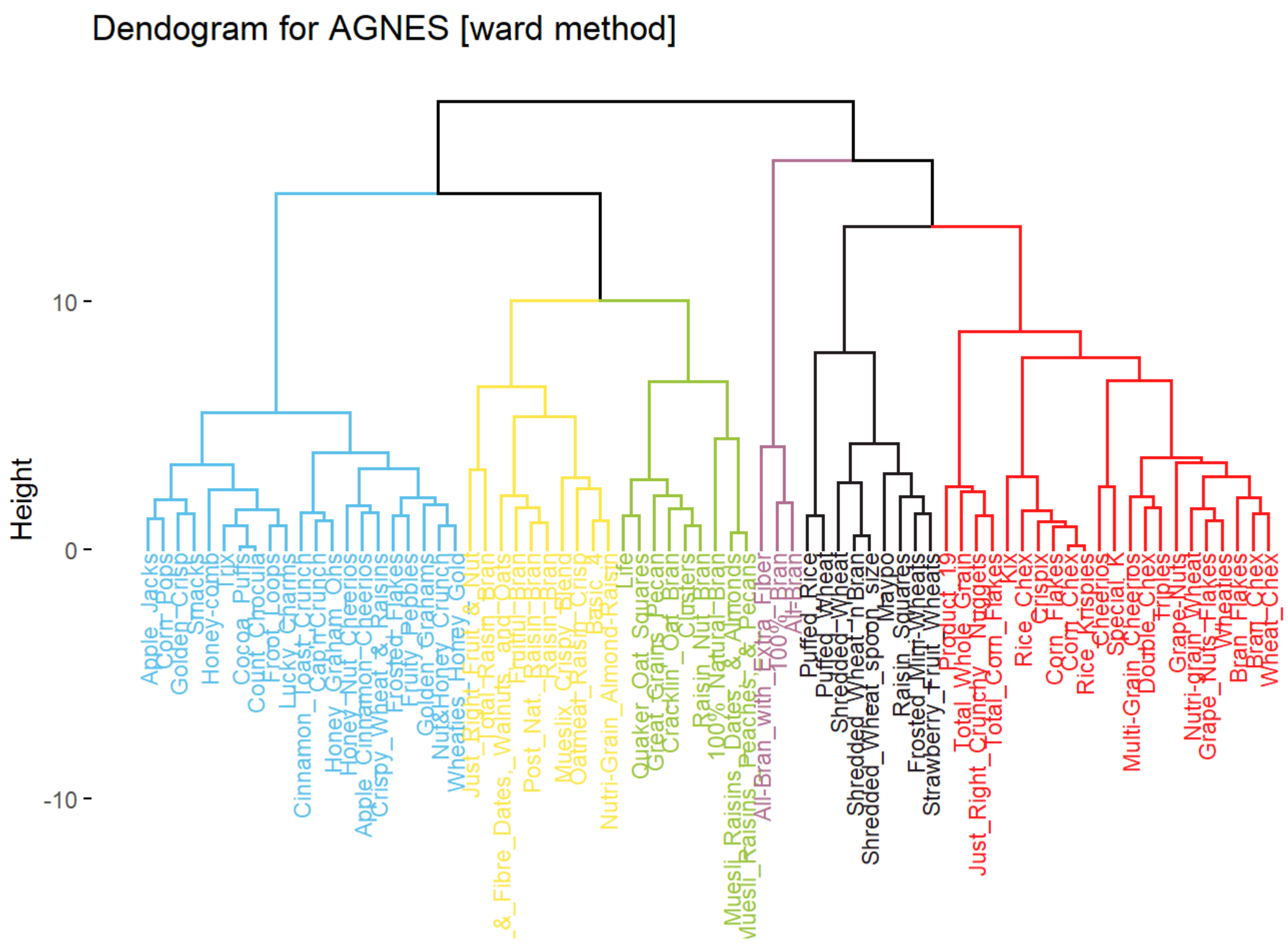
```
#Selecting  number of clusters K using gap-stat
fviz_nbclust(scaled, hcut, method = "gap_stat") # K = 7
```



The gap-statistic does not suggest an ideal number of cluster as it indicates its optimal number to be 1. We will not use this but we will rather use 6 clusters.

```
# Plot Dendrogram of AGNES[Ward method]
fviz_dend(wClust, k = 6,
          main = "Dendogram for AGNES [ward method]",
          cex = .6,
          k_colors = c('#5BC0EB','#FDE74C','#9BC53D',
                       '#B27092','#211A1E','#FF1B1C'),
          color_labels_by_k = TRUE,
          labels_track_height = 14,
          ggtheme = theme_classic())
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



we can notice how these clusters are being partitioned with different colours from the dendrogram above.

Comment on the structure of the clusters and their stability.

In order to understand the structure of the cluster, we split the data into two. We perform Hierarchical Clustering on both partitions using AGNES `Ward method` and plotted the dendrograms.

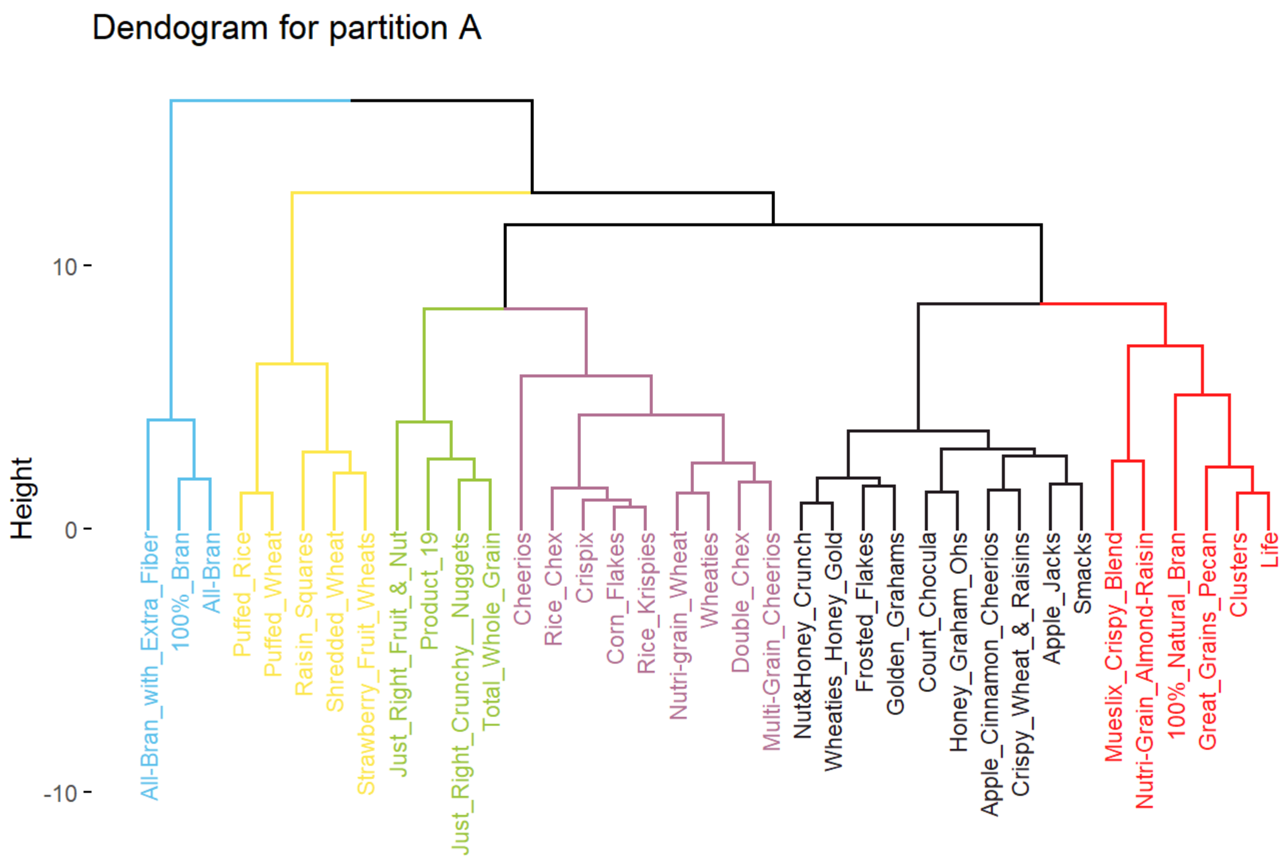
```
#Partitioning the data randomly.
index<-sample(1:nrow(scaled),
                              .5 * nrow(scaled))
partA<- scaled[-index,]
partB<-scaled[index,]
```

```
#For partA
set.seed(100)
dmA<- dist(partA, method = 'euclidean')
aClust<- hclust(dmA, method = 'ward.D2')
```

We then plot a dendrogram for the first partition.

```
# Dendrogram for first partition
fviz_dend(aClust, k = 6,
          main = "Dendogram for partition A",
          cex = .6,
          k_colors = c('#5BC0EB','#FDE74C','#9BC53D',
                       '#B27092','#211A1E','#FF1B1C'),
          color_labels_by_k = TRUE,
          labels_track_height = 14,
          ggtheme = theme_classic())
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



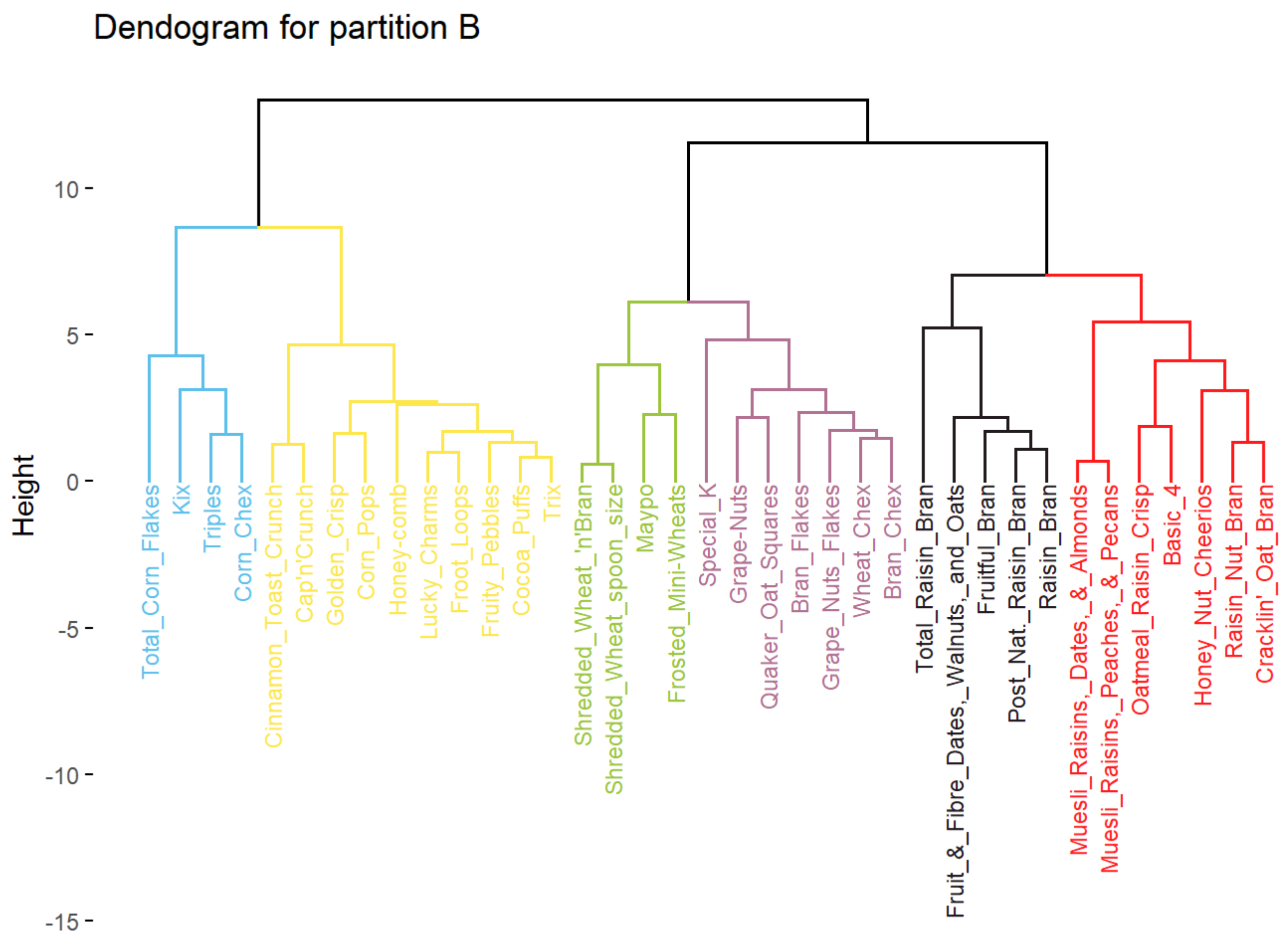
We do a similar clustering on the second partition.

```
#For partB
set.seed(100)
dmB<- dist(partB, method = 'euclidean')
bClust<- hclust(dmB, method = 'ward.D2')
```

We then plot a dendrogram for the first partition.

```
# Dendrogram for second partition
fviz_dend(bClust, k = 6,
          main = "Dendogram for partition B",
          cex = .6,
          k_colors = c('#5BC0EB','#FDE74C','#9BC53D',
                       '#B27092','#211A1E','#FF1B1C'),
          color_labels_by_k = TRUE,
          labels_track_height = 14,
          ggtheme = theme_classic())
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



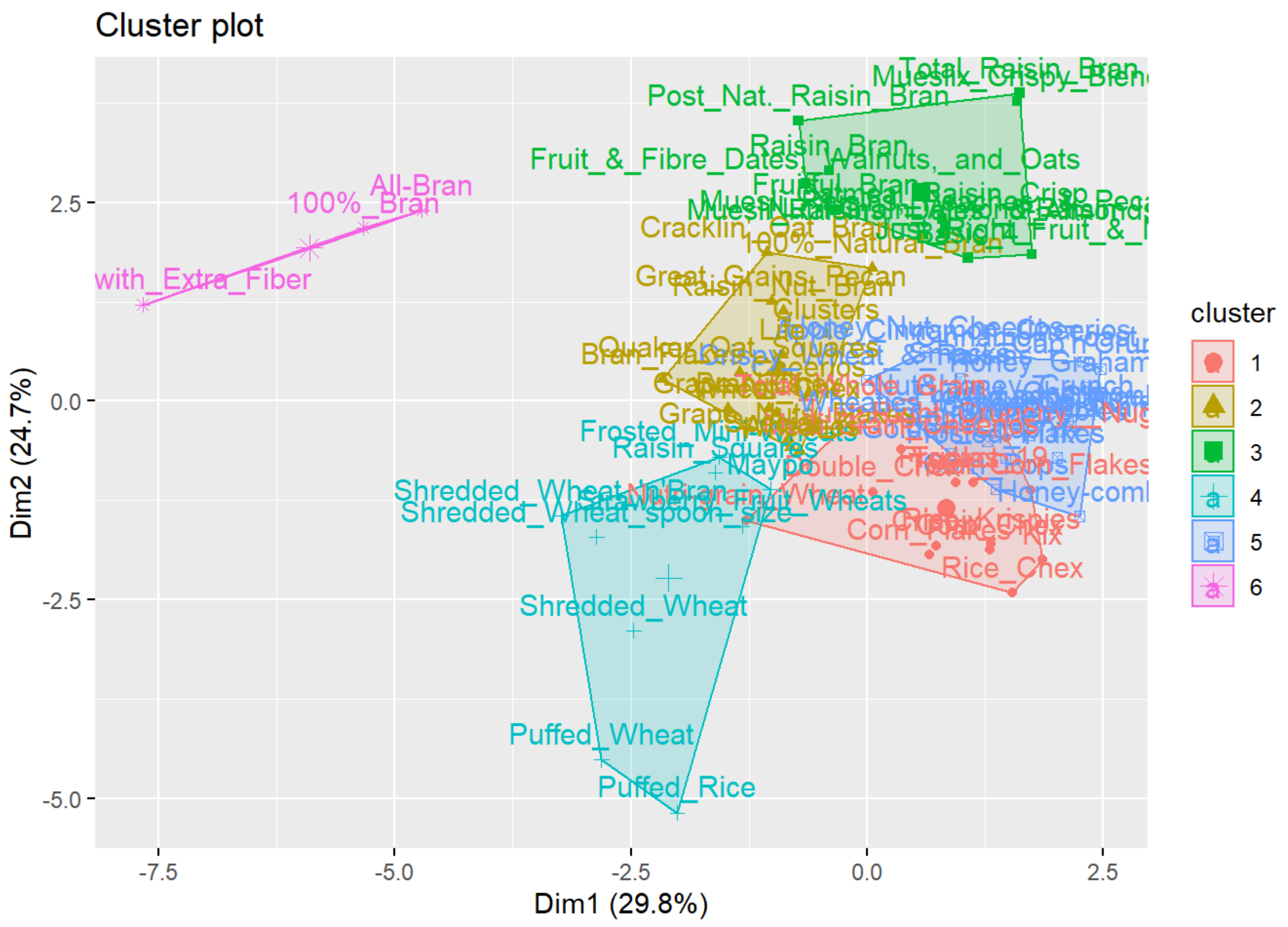
We discovered that the clusters maintain similar structure for more than 90% even when the data was randomly split. We may observe this by looking into the dendrogram for partition A and Partition B.

Stability of the cluster

A cluster is said to be stable if the partitioning remains similar when the dataset are randomised or the clustering process changes.

We use K means clustering to assess stability of the clusters.

```
set.seed(100)
KM<-kmeans(scaled,6, nstart = 25)
#Plot the clusters
fviz_cluster(KM, data = scaled)
```



We may see an exact clustering pattern if we examine both the dendrogram and the cluster plot for the K means algorithm.

Finding a cluster of “healthy cereals.”

To identify the cluster with healthy cereals, we may be interested in using rating but we have to identify the features which influences rating through our correlation plot.

From our correlation plot, sugar and calories are having significant negative effect on ratings and protein and fiber has positive effect on rating. We may use these variables or rating to select the cluster with the best nutritional content.

```
# assign cluster to data
cuts<-cutree(wClust, k = 6)
clusterData<- cbind(Cereals, cluster = cuts)

centroids<- clusterData %>%
        group_by(cluster) %>%
        summarise_all(funs(mean))

kable(round(centroids,2))
```

cluster	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	weight	cups	rating
1	63.33	4.00	0.67	176.67	11.00	6.67	3.67	310.00	25.00	1.00	0.39	73.84
2	117.78	3.44	2.67	115.56	2.61	12.50	7.56	131.67	22.22	1.00	0.67	40.72
3	110.95	1.52	1.00	168.57	0.57	12.43	11.48	47.38	25.00	1.00	0.88	28.66
4	132.00	3.00	1.40	192.00	3.65	15.55	10.90	172.50	40.00	1.34	0.71	36.26
5	103.64	2.68	0.55	225.91	1.68	18.32	3.45	74.55	38.64	1.00	0.93	46.18
6	82.22	2.44	0.11	1.67	2.11	15.33	2.33	90.56	11.11	0.87	0.85	63.02

Cluster 1 has the highest rating with an average of 73.8. This cluster has cereals with the highest Protein and fibre but with the least calories. It has almost close to the least sugar on average too. Cluster 1 has `100% Bran`, `All Bran` and `All Bran with extra fibre` as the cereals in it.

Should the data be normalised

For this goal, we use the main dataset which is not normalised so as to be able to properly interpret the result but we use normalised data to obtain clusters since each variable has different measurement scale.

How should they be used in the cluster analysis?

In order to use this in the cluster analysis, we assigned the cluster values to each cereal in the original dataset and computed the mean for each features based on the clusters assigned. This makes it easy to analyse and advise on which cluster has the healthy cereals.