

FML5

Vijay

2022-04-16

```
# installing required 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(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.1.3
```

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

```
library(ggplot2)
```

```
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(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3
```

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

```
## v tibble  3.1.6      v purrr   0.3.4
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
## Warning: package 'forcats' was built under R version 4.1.3
```

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

```
library(cluster)
library(NbClust)
library(ppclust)
```

```
## Warning: package 'ppclust' was built under R version 4.1.3
```

```
library(dendextend)
```

```
## Warning: package 'dendextend' was built under R version 4.1.3
```

```
##
## -----
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
## To suppress this message use: suppressPackageStartupMessages(library(dendextend))
## -----
##
## Attaching package: 'dendextend'
##
## The following object is masked from 'package:stats':
##
##   cutree
```

```
# Importing cereal dataset
cereals <- read.csv("cereals.csv")
## Reviewing the Dataset
# Reviewing first five rows of the dataset
head(cereals)
```

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

```
# checking the structure of the dataset
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 ...
## $ 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 ...
```

```
# summary of the dataset
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
##
```

The data has to be scaled before removing the NA values from the data set.

```
# Creating duplicate dataset for preprocessing
cereal_data <- cereals
# Scale the dataset before placing that into a clustering algorithm
cereal_data[, c(4:16)] <- scale(cereals[, c(4:16)])
# Removing Null values from dataset
cereal_pre <- na.omit(cereal_data)
# Review the first five rows of scaled data set after removing Null values
head(cereal_pre)
```

```
##           name mfr type  calories  protein      fat
## 1      100%_Bran   N    C -1.8929836  1.3286071 -0.01290349
## 2  100%_Natural_Bran Q    C  0.6732089  0.4151897  3.96137277
## 3      All-Bran    K    C -1.8929836  1.3286071 -0.01290349
## 4 All-Bran_with_Extra_Fiber K    C -2.9194605  1.3286071 -1.00647256
## 6  Apple_Cinnamon_Cheerios G    C  0.1599704 -0.4982277  0.98066557
## 7      Apple_Jacks  K    C  0.1599704 -0.4982277 -1.00647256
##      sodium      fiber      carbo      sugars      potass      vitamins      shelf
## 1 -0.3539844  3.29284661 -2.5087829 -0.2343906  2.5753685 -0.1453172  0.9515734
## 2 -1.7257708 -0.06375361 -1.7409943  0.2223705  0.5160205 -1.2642598  0.9515734
## 3  1.1967306  2.87327158 -1.9969238 -0.4627711  3.1434645 -0.1453172  0.9515734
## 4 -0.2346986  4.97114672 -1.7409943 -1.6046739  3.2854885 -0.1453172  0.9515734
## 6  0.2424445 -0.27354112 -1.1011705  0.6791317 -0.4071355 -0.1453172 -1.4507595
## 7 -0.4136273 -0.48332864 -0.9732057  1.5926539 -0.9752315 -0.1453172 -0.2495930
##      weight      cups      rating
## 1 -0.1967771 -2.1100340  1.8321876
```

```
## 2 -0.1967771  0.7690100 -0.6180571
## 3 -0.1967771 -2.1100340  1.1930986
## 4 -0.1967771 -1.3795303  3.6333849
## 6 -0.1967771 -0.3052601 -0.9365625
## 7 -0.1967771  0.7690100 -0.6756899
```

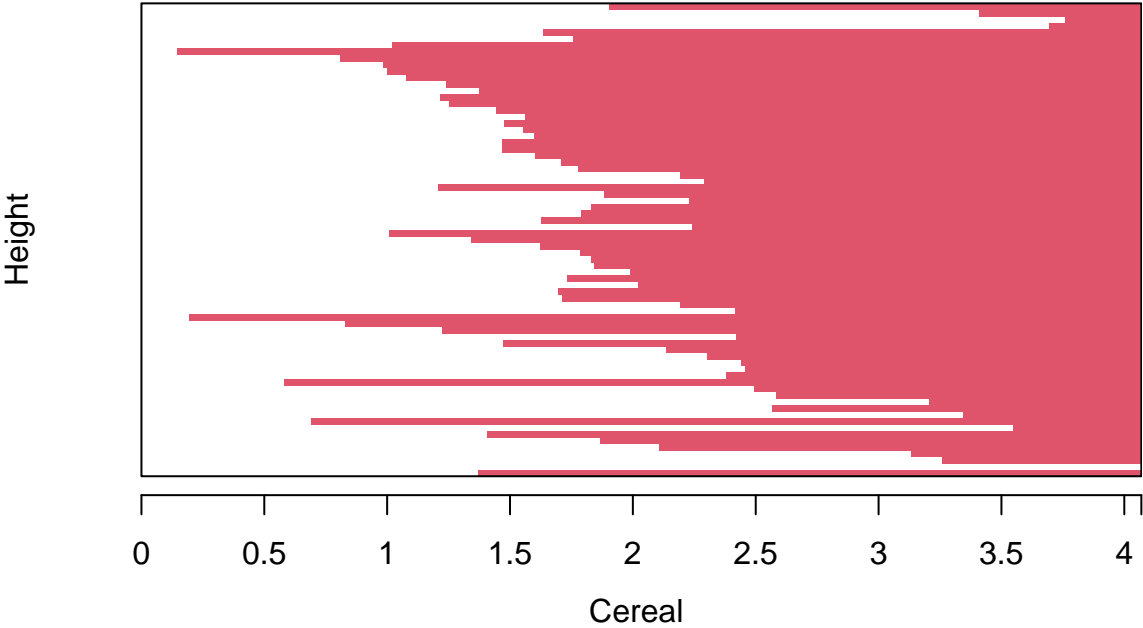
After preprocessing 3 null values have been found in the dataset

Q) Apply hierarchical clustering to the data using Euclidean distance to the normalized measurements. Use Agnes to compare the clustering from single linkage, complete linkage, average linkage, and Ward. Choose the best method.

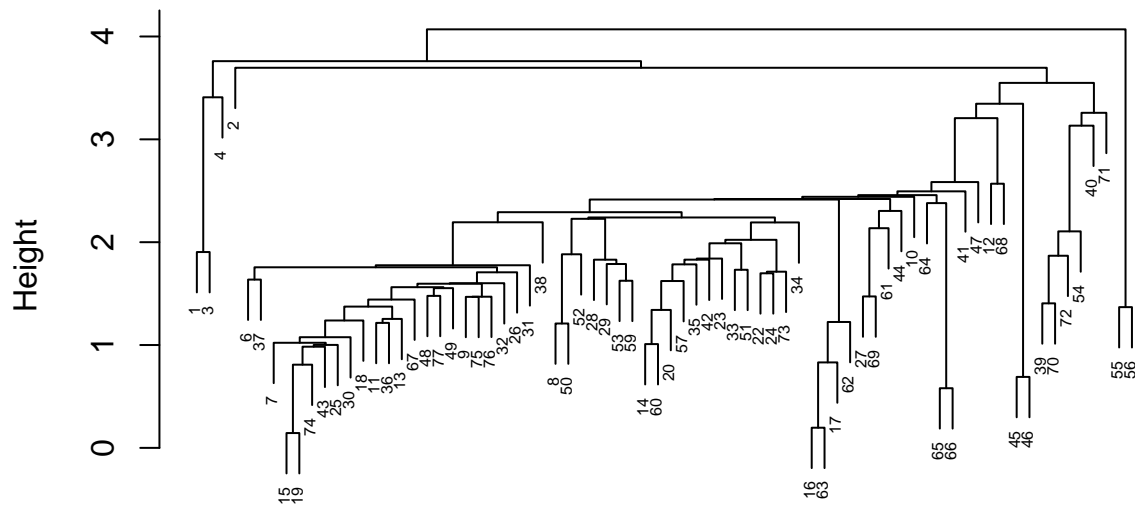
Single Linkage:

```
# Creating dissimilarity matrix for the numeric values in the dataset using Euclidean distance
cereal_EUC <- dist(cereal_pre[, c(4:16)], method = "euclidean")
# Performing hierarchical clustering using single linkage
agnes_hc_single <- agnes(cereal_EUC, method = "single")
# Plot the results of the different methods
plot(agnes_hc_single,
     main = "Single Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```

Single Linkage Method



Single Linkage Method

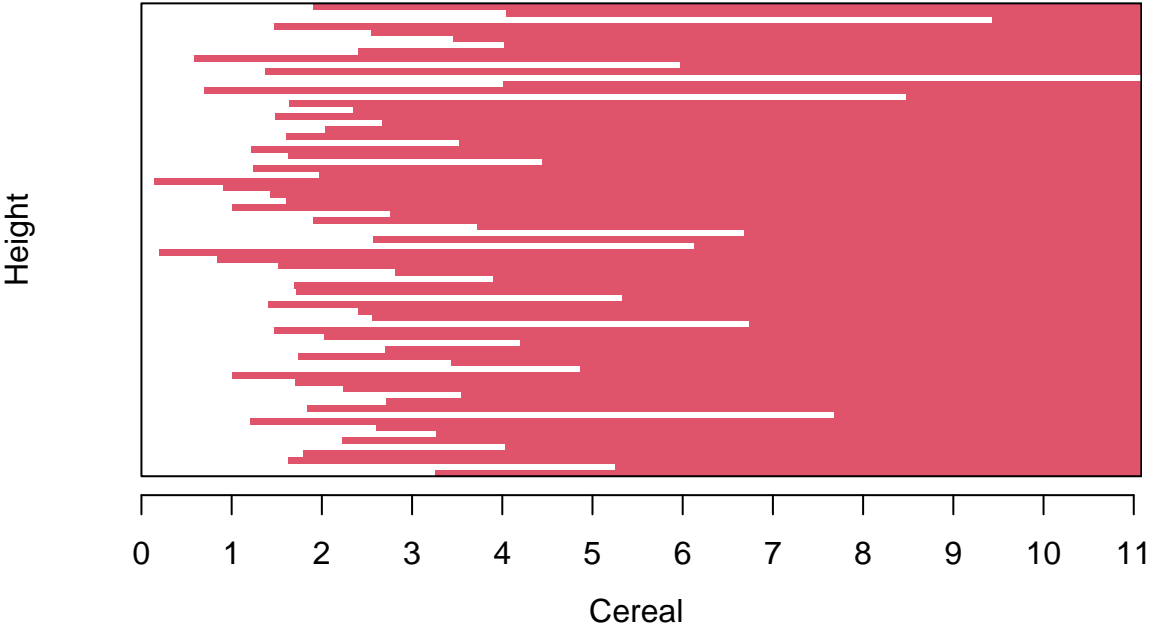


Cereal
Agglomerative Coefficient = 0.61

Complete Linkage:

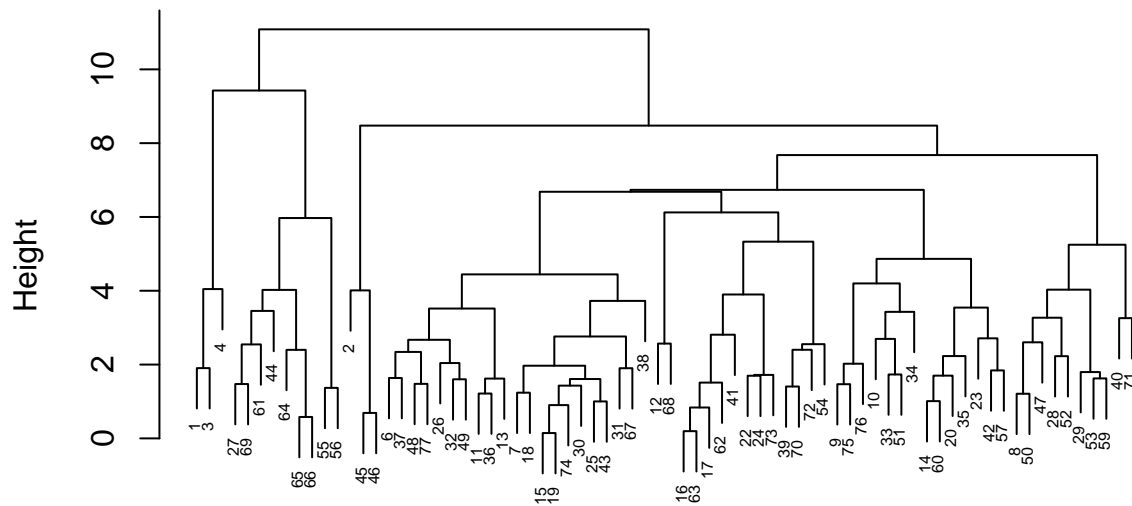
```
# Performing hierarchical clustering using complete linkage method
agnes_hc_complete <- agnes(cereal_EUC, method = "complete")
# Plot the results of the different methods
plot(agnes_hc_complete,
     main = "Complete Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```

Complete Linkage Method



Agglomerative Coefficient = 0.84

Complete Linkage Method

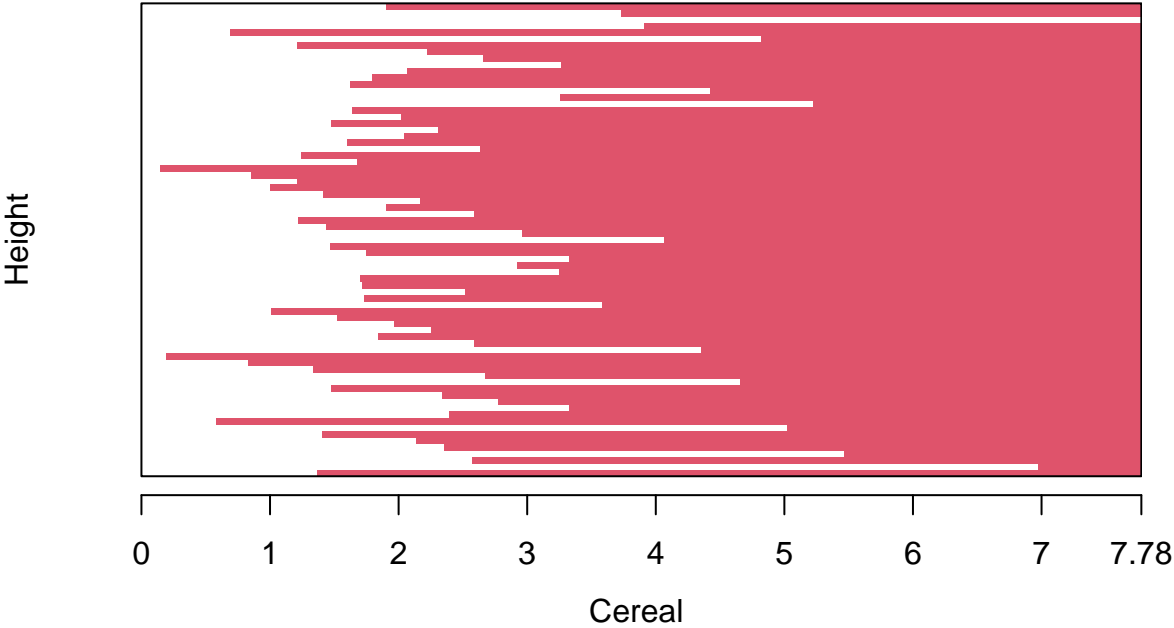


Cereal
Agglomerative Coefficient = 0.84

Average Linkage:

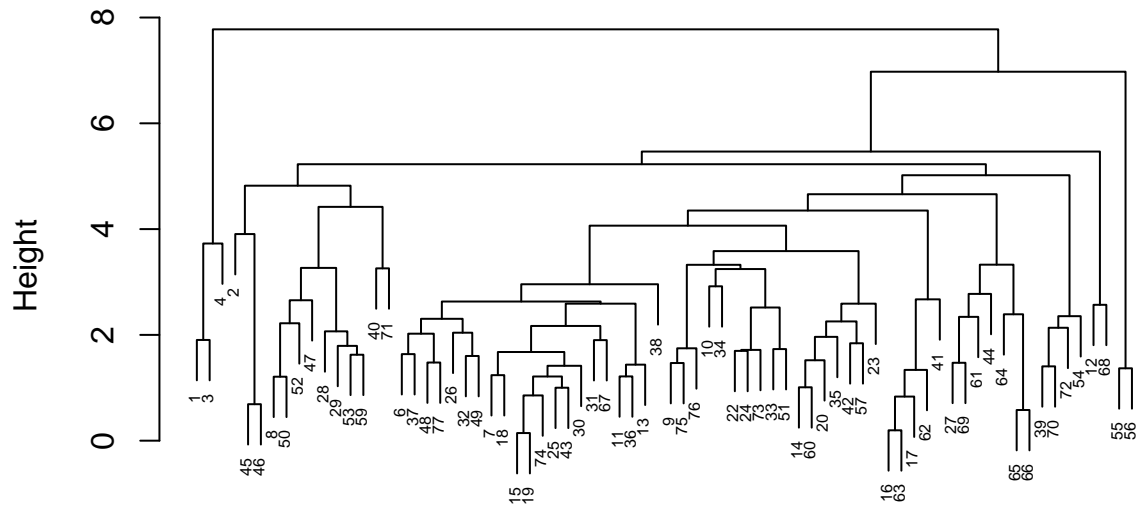
```
# Performing hierarchical clustering using average linkage method
agnes_hc_average<- agnes(cereal_EUC, method = "average")
# Plot the results of the different methods
plot(agnes_hc_average,
     main = "Average Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```

Average Linkage Method



Agglomerative Coefficient = 0.78

Average Linkage Method

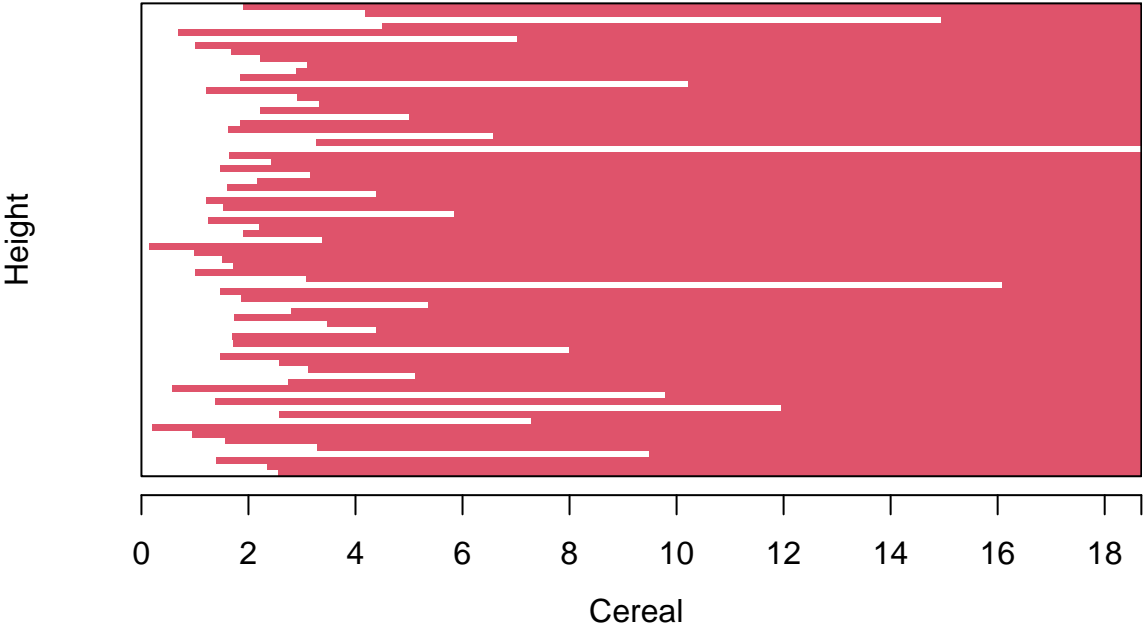


Cereal
Agglomerative Coefficient = 0.78

Ward Method:

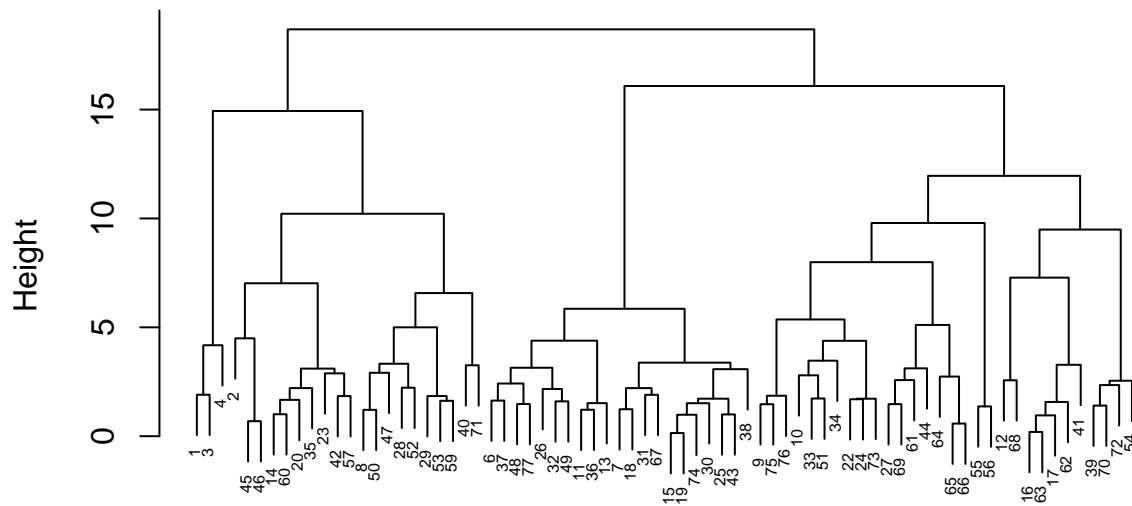
```
# Performing hierarchical clustering using ward linkage method
agnes_hc_ward <- agnes(cereal_EUC, method = "ward")
# Plot the results of the different methods
plot(agnes_hc_ward,
     main = "Ward Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```

Ward Linkage Method



Agglomerative Coefficient = 0.9

Ward Linkage Method



Cereal

Agglomerative Coefficient = 0.9

Based on the agglomerative coefficient the best clustering method is that which returned from the each method The close the value is to 1.0, the closer the clustering structure will be. So, the method with the value closer to 1.0 will be chosen

Single Linkage: 0.61 Complete Linkage: 0.84 Average Linkage: 0.78 Ward Method: 0.90

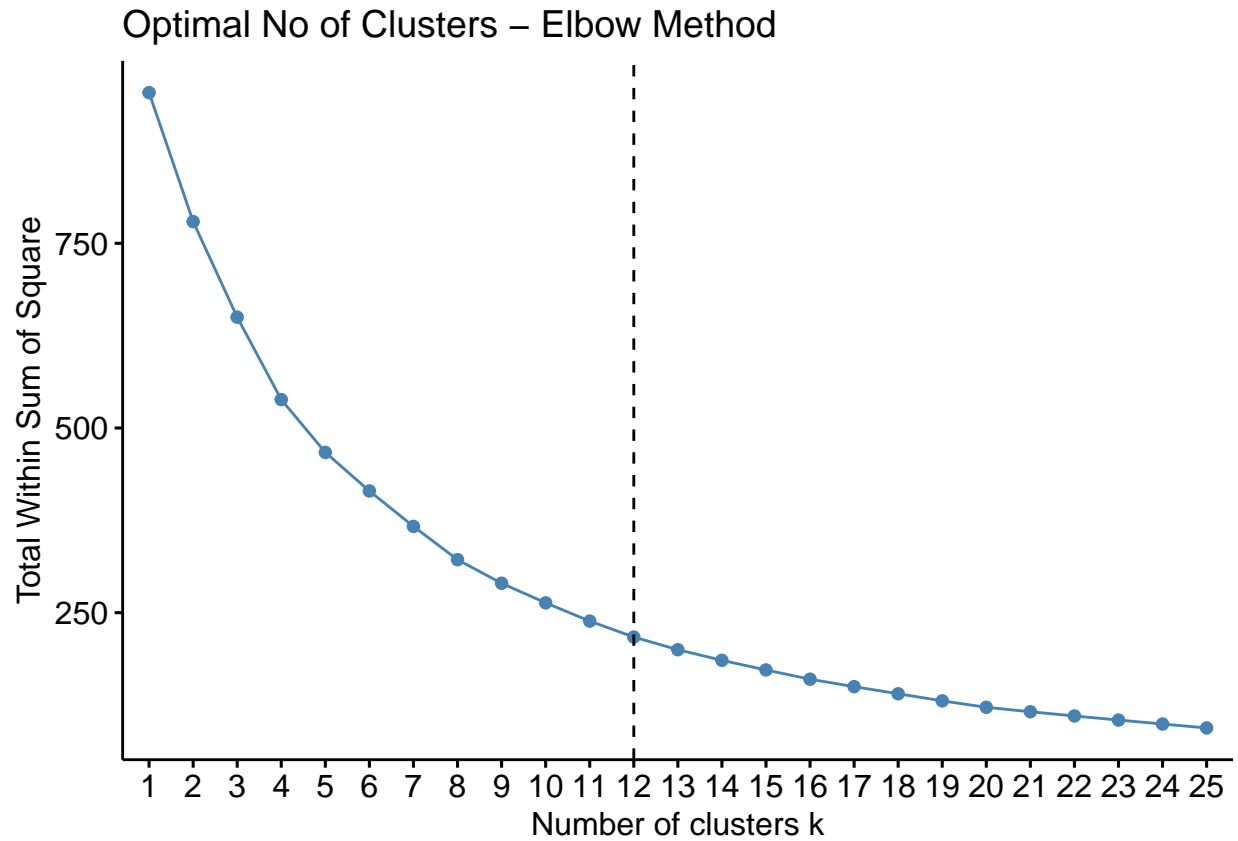
The Ward method will be chosen as the best clustering model based on the result.

Q) How many clusters would you choose?

To determine Appropriate number of Clusters we use the elbow method and silhouette method.

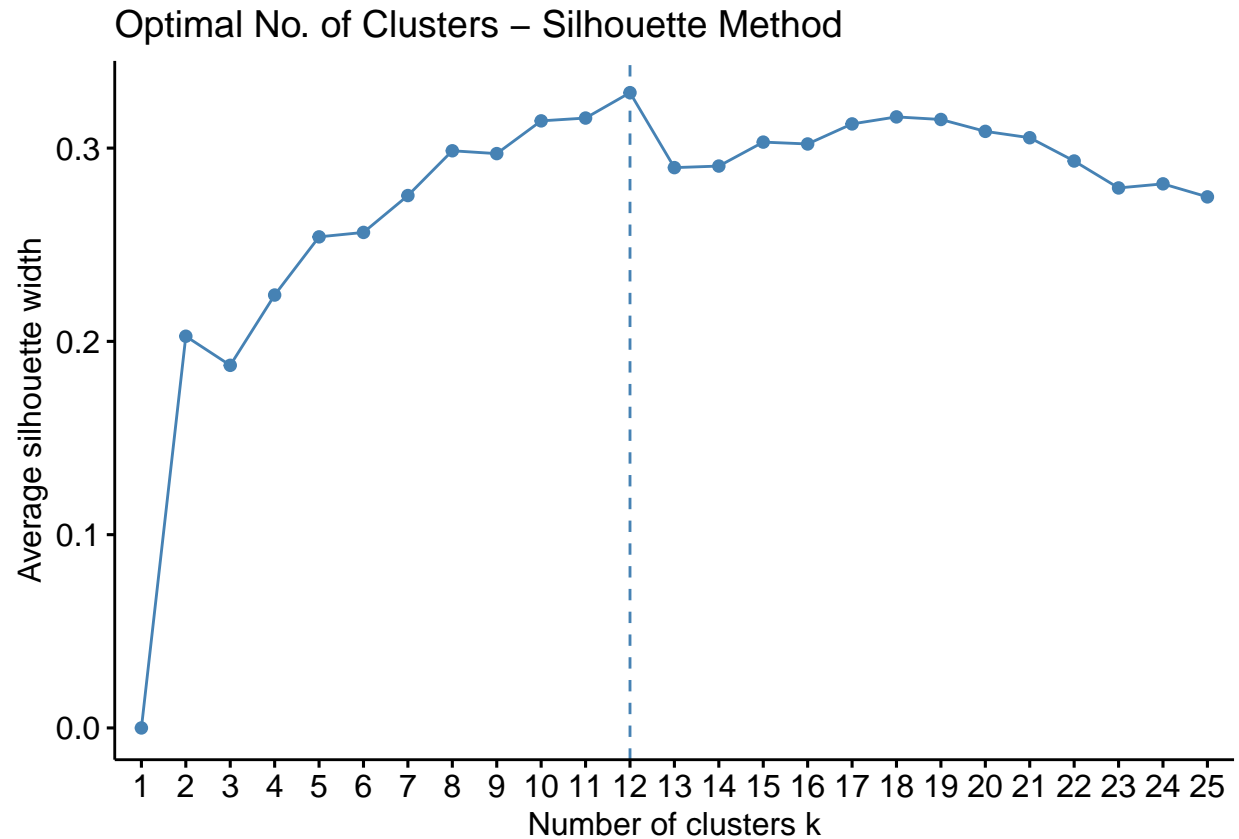
Elbow Method:

```
# To Determine the optimal number of clusters using the Elbow method
fviz_nbclust(cereal_pre[, c(4:16)], hcut, method = "wss", k.max = 25) +
  labs(title = "Optimal No of Clusters - Elbow Method") +
  geom_vline(xintercept = 12, linetype = 2)
```



Silhouette Method:

```
# Determine the optimal number of clusters for the dataset via the silhouette method
fviz_nbclust(cereal_pre[, c(4:16)],
             hcut,
             method = "silhouette",
             k.max = 25) +
labs(title = "Optimal No. of Clusters - Silhouette Method")
```

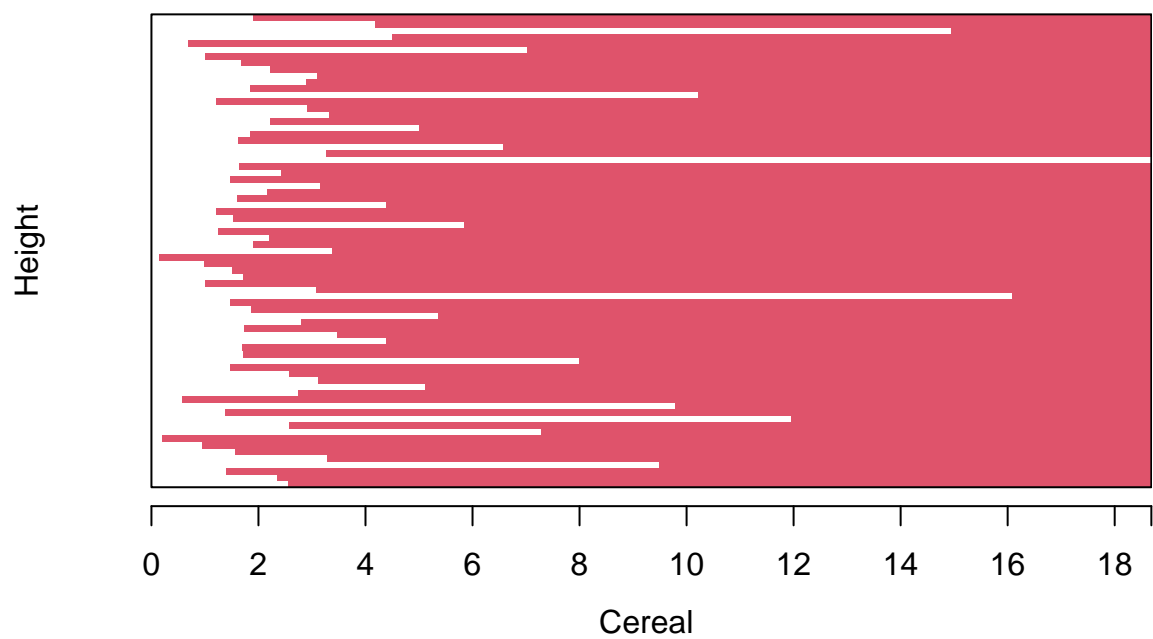


Based on the results of the elbow and silhouette methods, the number of clusters are 12.

Outline the 12 clusters on the hierarchical tree

```
# Plot of the Ward hierarchical tree with the 12 clusters outlined for reference
plot(agnes_hc_ward,
     main = "AGNES - Ward Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50,)
```

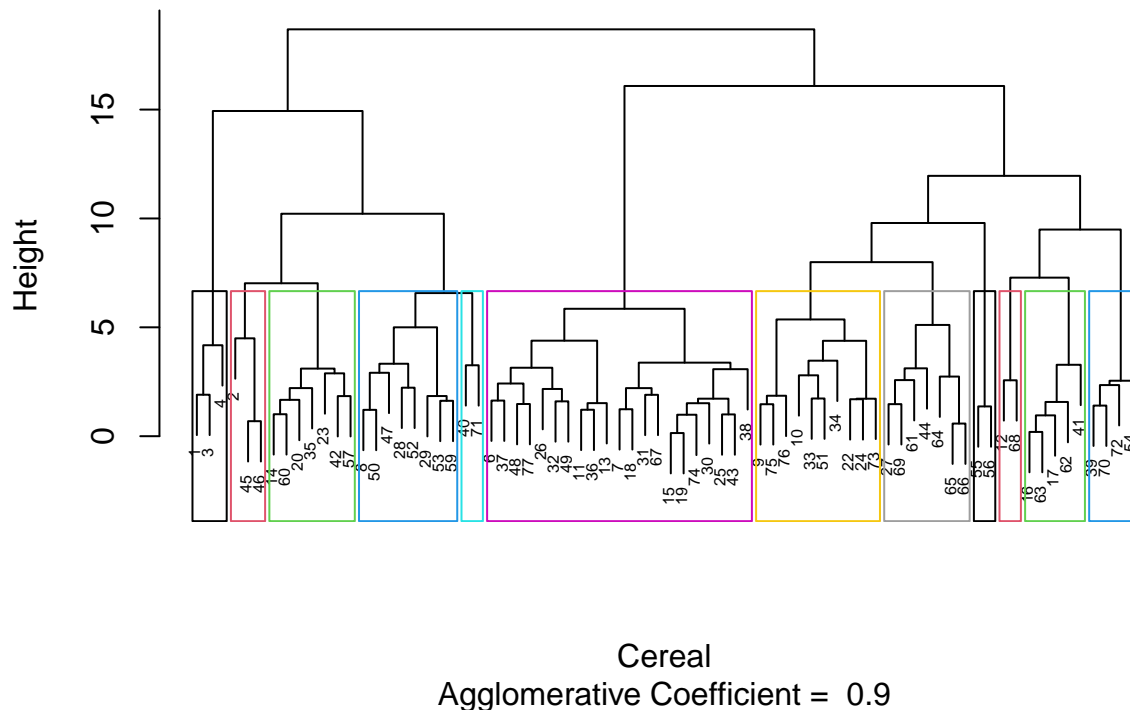
AGNES – Ward Linkage Method



Agglomerative Coefficient = 0.9

```
rect.hclust(agnes_hc_ward, k = 12, border = 1:12)
```


AGNES – Ward Linkage Method



Q) 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.” Should the data be normalized? If not, how should they be used in the cluster analysis?

Ans: Normalizing the data is not a good idea in this situation because of Nutritional information of cereals is normalized based on cereal evaluated so the collected data can only have cereals with high sugar quantity and less iron, fiber and other nutrition data. So it's not possible to tell nourishment the cereal provides to a child once it's normalized throughout sample. We may infer that a cereal with iron quantity of 0.99 means it is sufficient for a child yet, it simply has no nutritional value. As a result, a best way to preprocess the data would be to convert it to a ratio of daily recommended calories, fiber, carbohydrates, and other nutrients for a child. This makes analysts take more informed decisions on clusters during reviewing and also preventing a few larger variables from overriding the distance estimates. By looking at the clusters the analyst may have to look at the cluster average to see what percentage of a student's daily required nutrition that would come from cereal.