Assignment_5

Saipriya Gourineni

2022-11-28

```
getwd()
```

```
## [1] "C:/Users/Saipr/OneDrive/Desktop"
```

```
setwd("C:/Users/Saipr/OneDrive/Desktop")
```

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

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

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

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

```r
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.16.0
## 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
```

```r
library(tidyverse)
```

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

```
## v tibble  3.1.8     v purrr   0.3.4
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```r
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
```

```r
# To import the data collection "cereal"
Cereals <- read.csv("Cereals.csv")
# Getting the first few rows of the data collection using head
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
```

```r
# Analyzing the data set's structure with str
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 ...
```

```r
# Analyzing the data set's summary utilizing the summary
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  
## 
```

Now I am scaling the data to remove NA values from the data set.

```r
# I'm making a duplicate of this data set here for preparation.
Scaled_Cereals <- Cereals
# To fit the data set into a clustering technique, I am currently scaling it.
Scaled_Cereals[ , c(4:16)] <- scale(Cereals[ , c(4:16)])
# Here, I'm using the omit function to remove the NA values from the data set.
Preprocessed_Cereal <- na.omit(Scaled_Cereals)
# After deleting NA, using head to display the top few rows
head(Preprocessed_Cereal)
```

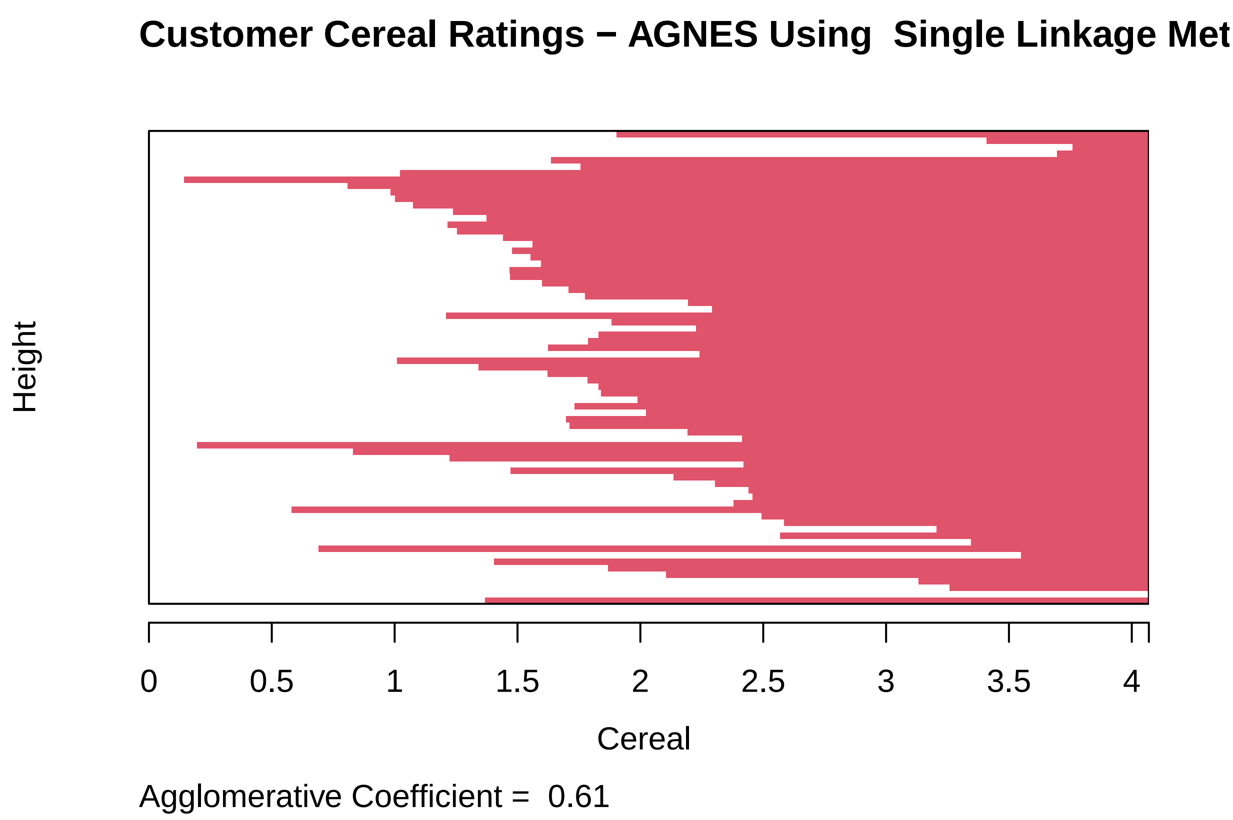
```
##                        name mfr type   calories    protein         fat
## 1                 100%_Bran   N    C -1.8929836  1.3286071 -0.01290349
## 2         100%_Natural_Bran   Q    C  0.6732089  0.4151897  3.96137277
## 3                  All-Bran   K    C -1.8929836  1.3286071 -0.01290349
## 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
```

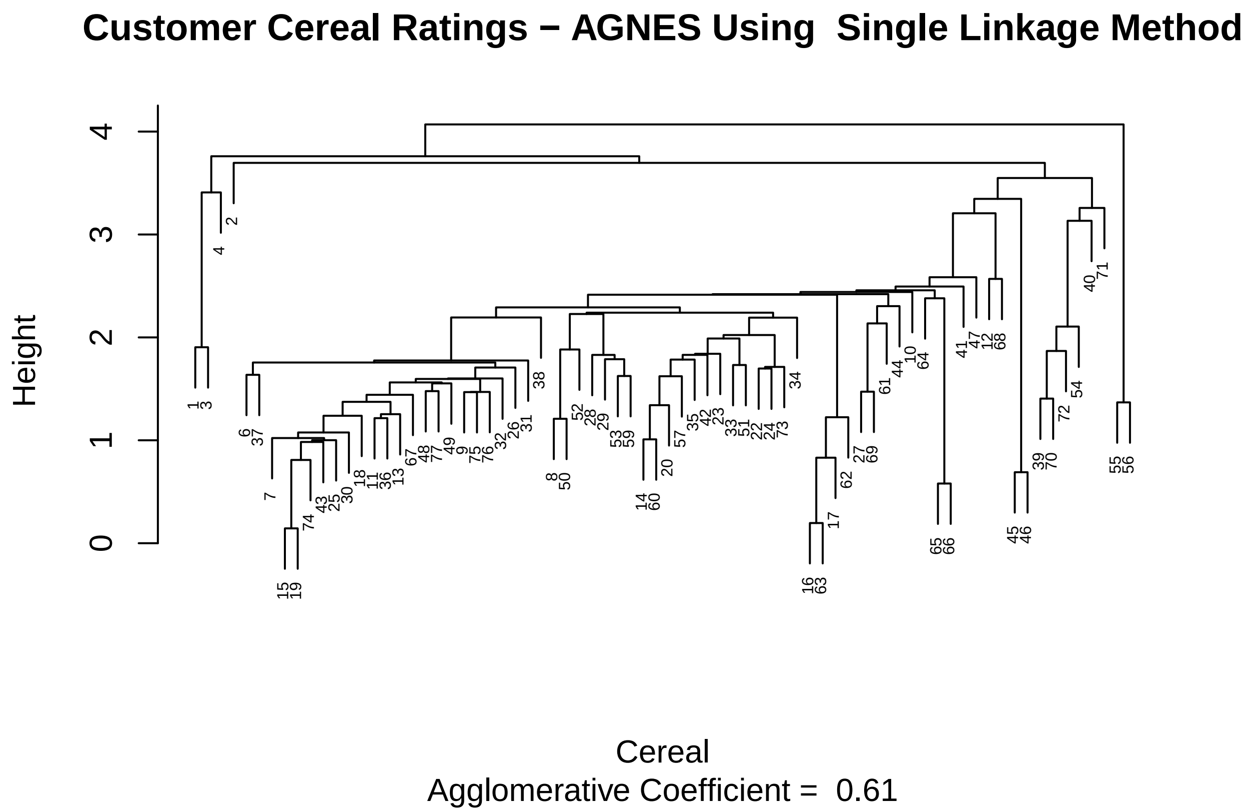
```
## 3 -0.1967771 -2.1100340  1.1930986
## 4 -0.1967771 -1.3795303  3.6333849
## 6 -0.1967771 -0.3052601 -0.9365625
## 7 -0.1967771  0.7690100 -0.6756899
```

After pre-processing and scaling the data, the total number of observations decreased from 77 to 74. Only 3 records had "NA" as their value. ## 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:

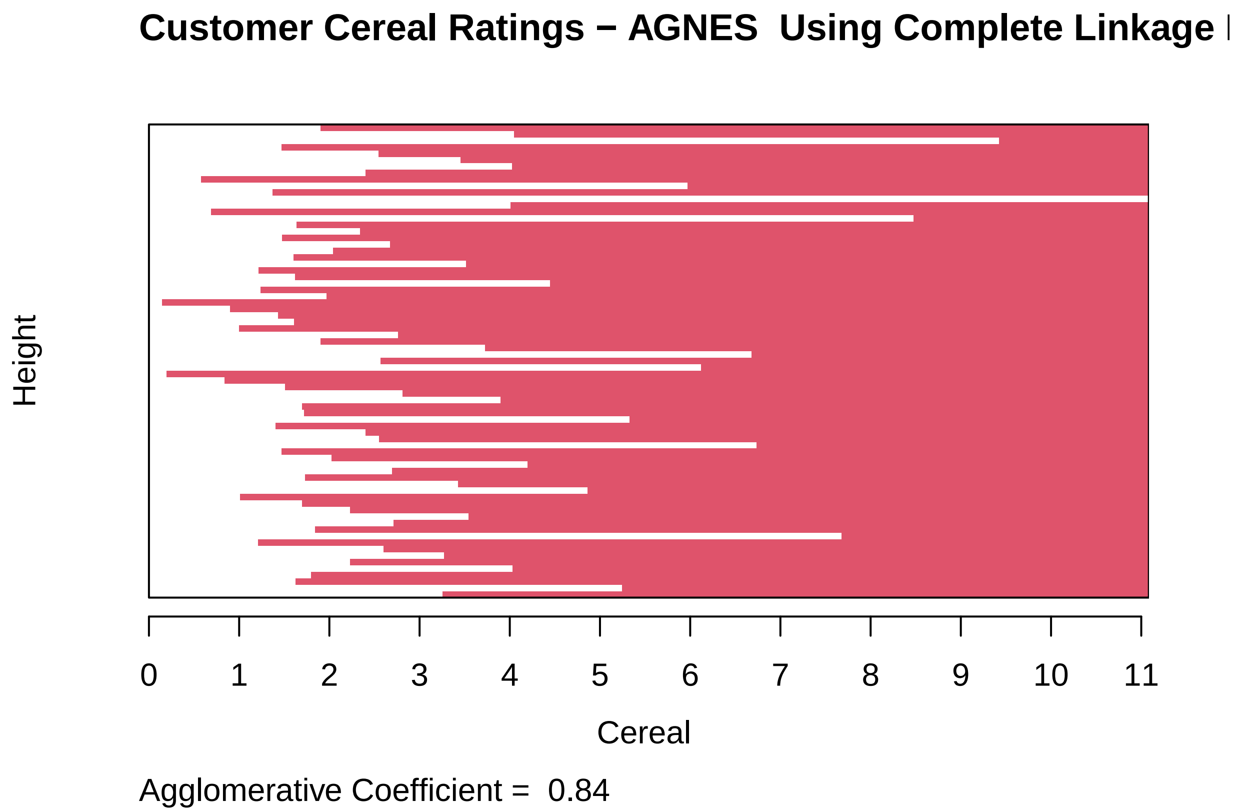
```
# Euclidean distance measurements are used to create the dissimilarity matrix for all the numerical val
Cereal_Euclidean <- dist(Preprocessed_Cereal[ , c(4:16)], method = "euclidean")
# The single linkage approach is used to perform a hierarchical clustering.
HC_Single <- agnes(Cereal_Euclidean, method = "single")
# I'm plotting the outcomes of the various techniques here.
plot(HC_Single,
     main = "Customer Cereal Ratings - AGNES Using  Single Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```

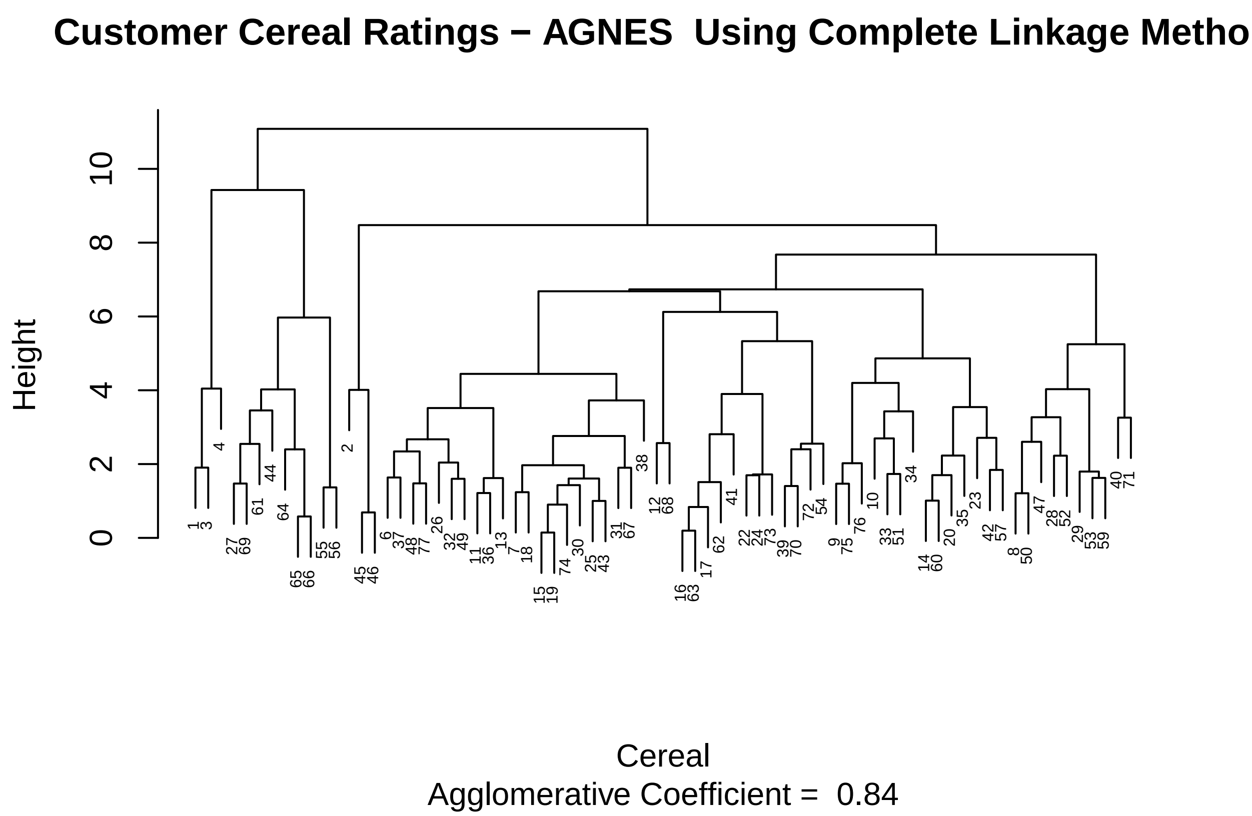




Complete Linkage:

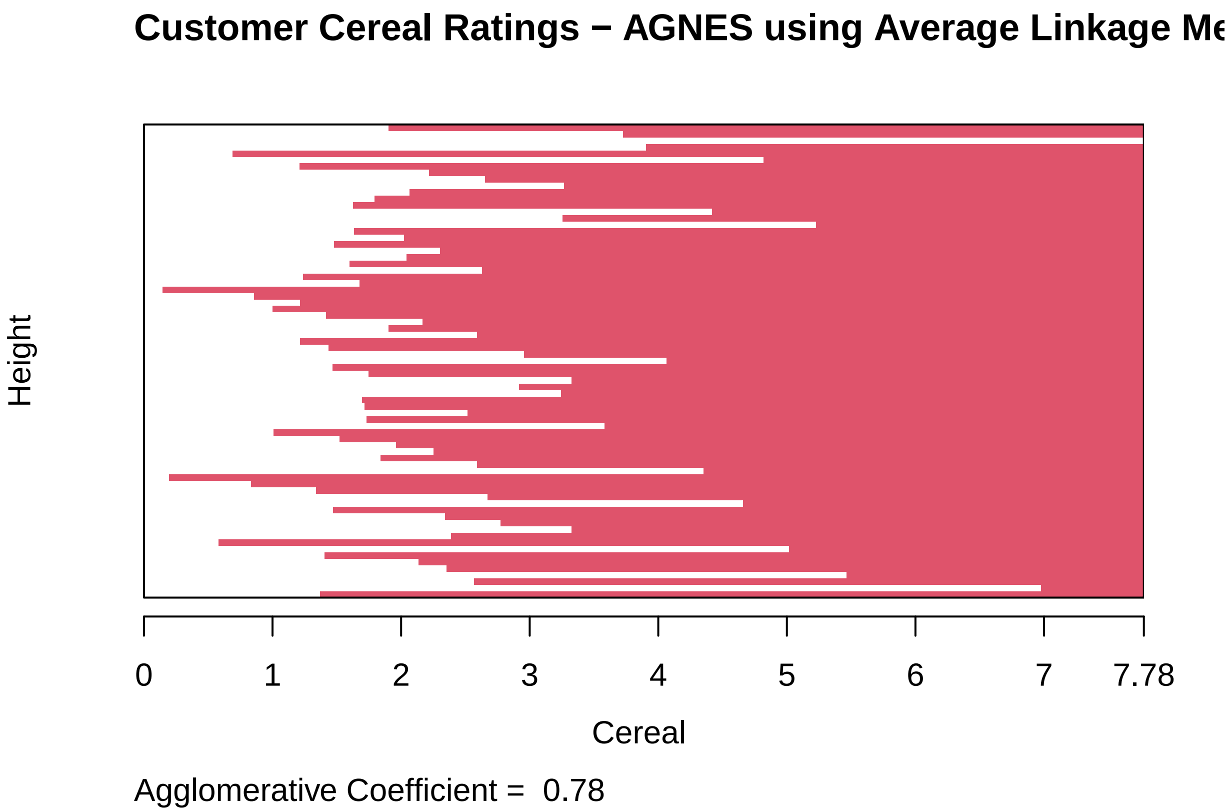
```r
# Making use of the entire linkage approach to perform hierarchical clustering
HC_Complete <- agnes(Cereal_Euclidean, method = "complete")
# I'm plotting the outcomes of the various techniques here.
plot(HC_Complete,
     main = "Customer Cereal Ratings - AGNES  Using Complete Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```

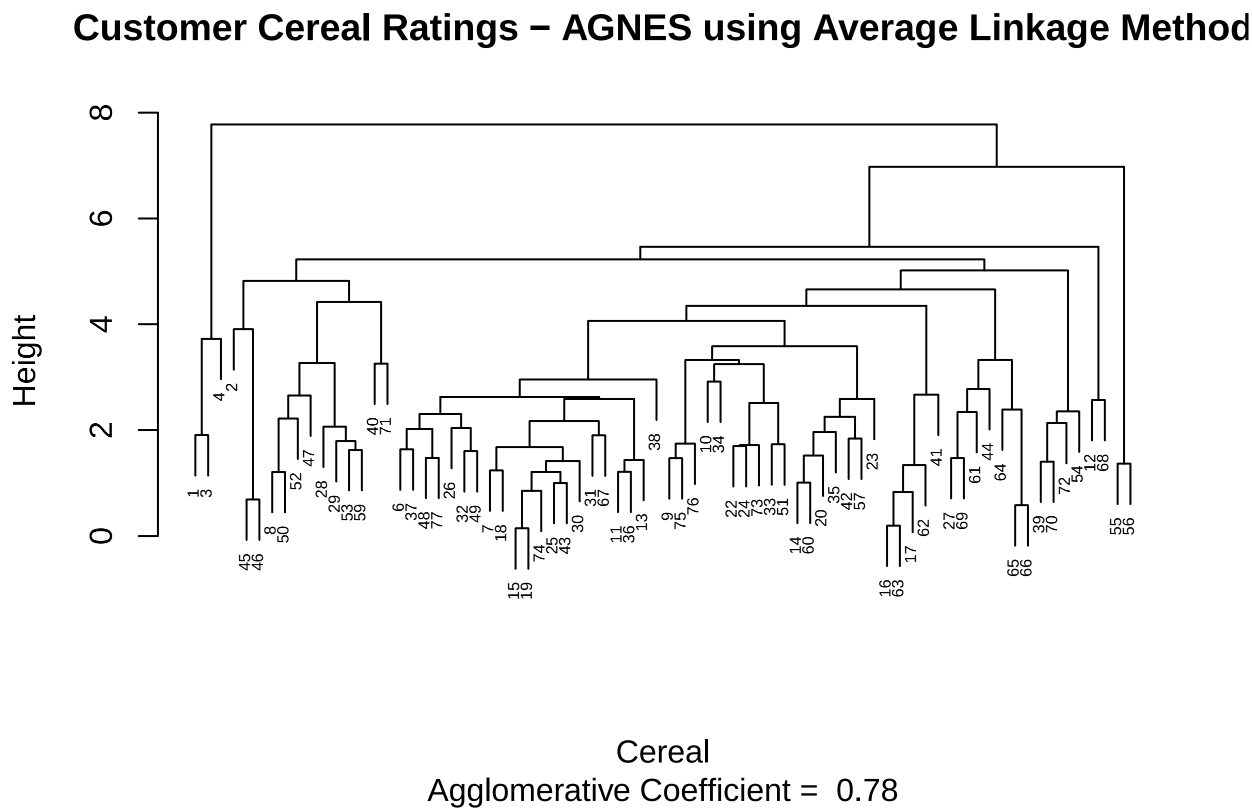

Customer Cereal Ratings − AGNES Using Complete Linkage
Height
0
1
2
3
4
5
6
7
8
9
10
11
Cereal
Agglomerative Coefficient = 0.84



Average Linkage:

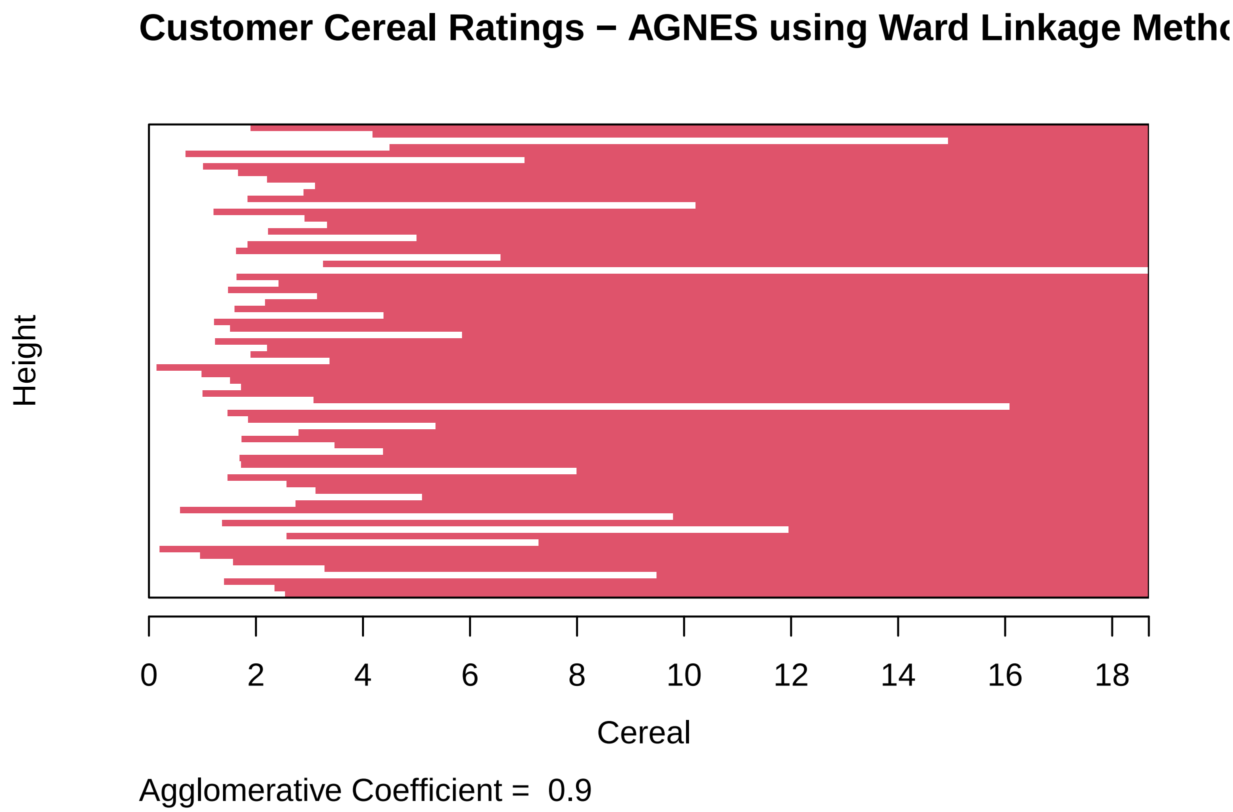
```r
# Performing the average linkage method for hierarchical clustering
HC_Average <- agnes(Cereal_Euclidean, method = "average")
# Here I am Plotting the results of the different methods
plot(HC_Average,
     main = "Customer Cereal Ratings - AGNES using Average Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50)
```


Customer Cereal Ratings − AGNES using Average Linkage Me
Height
0
1
2
3
4
5
6
7
7.78
Cereal
Agglomerative Coefficient = 0.78

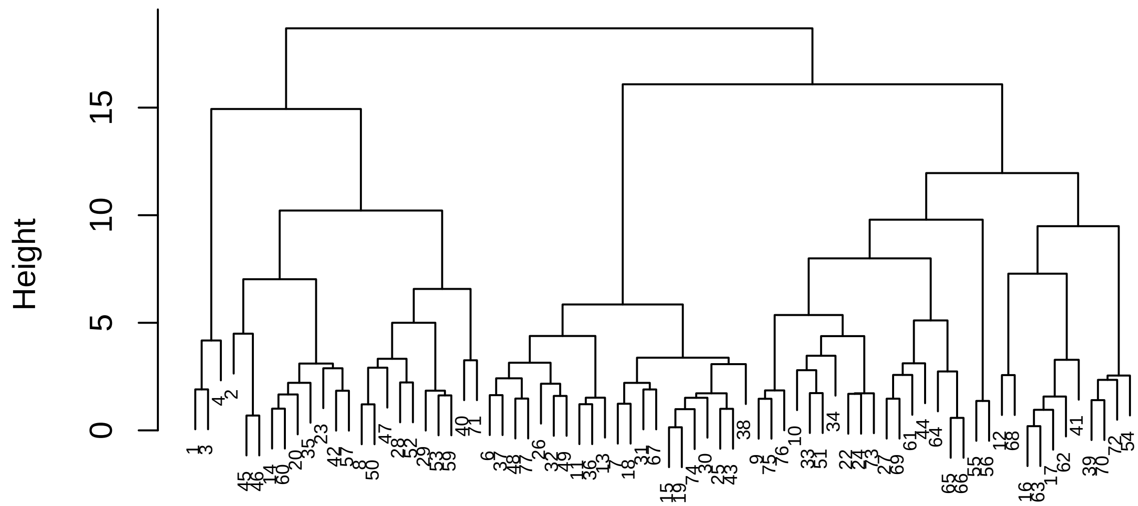


Ward Method:

```
# Performing the ward linkage method for hierarchical clustering
HC_Ward <- agnes(Cereal_Euclidean, method = "ward")
# I am  Plotting the outcomes of the different methods
plot(HC_Ward,
     main = "Customer Cereal Ratings - AGNES using Ward Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55)
```


Customer Cereal Ratings − AGNES using Ward Linkage Metho
Height
0
2
4
6
8
10
12
14
16
18
Cereal
Agglomerative Coefficient = 0.9

Customer Cereal Ratings − AGNES using Ward Linkage Method



Cereal
Agglomerative Coefficient = 0.9

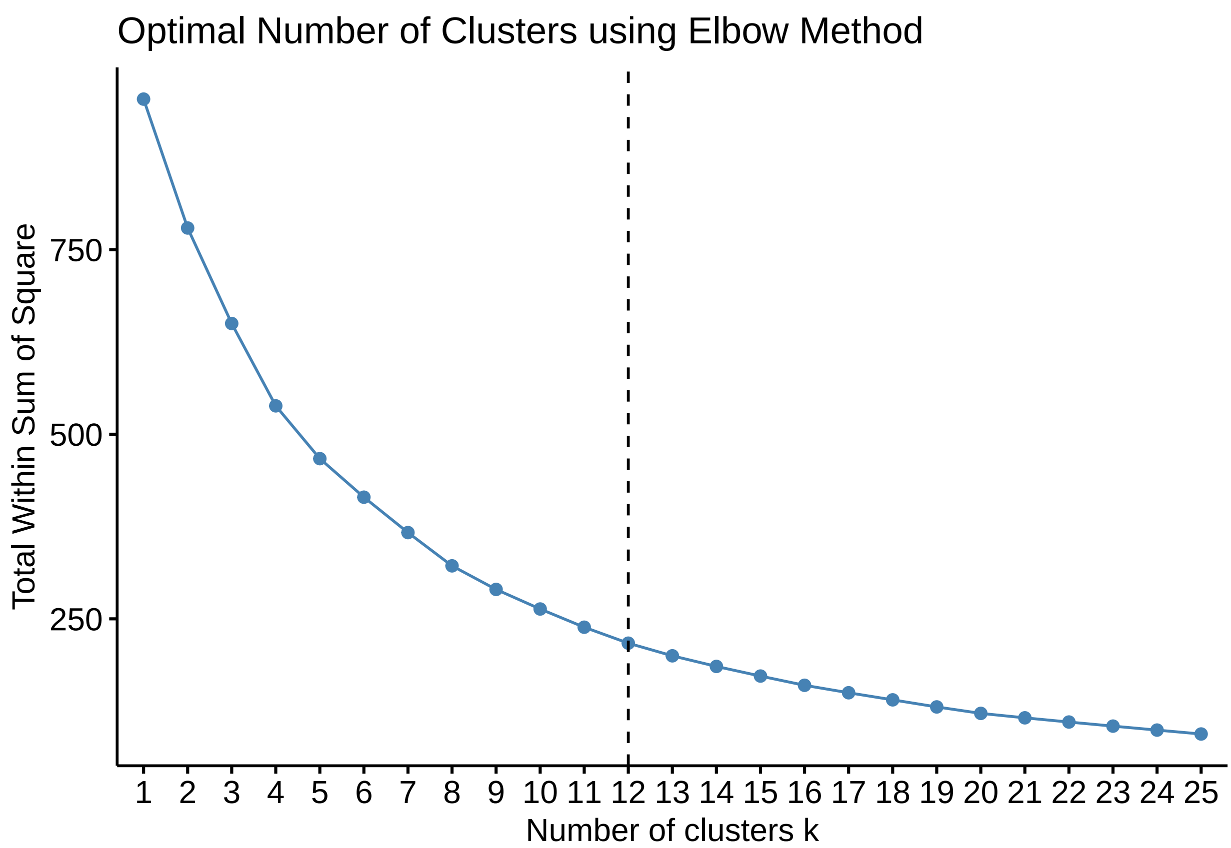
If the value is near to 1.0, the clustering structure is closer. As a result, the approach with the value that is most similar to 1.0 will be selected. Single Linkage: 0.61 Complete Linkage: 0.84 Average Linkage: 0.78 Ward Method: 0.90 Here From the result, The best clustering model is the Ward method.

Q) How many clusters would you choose?

Here I am using elbow and silhouette methods to determine the appropriate number of clusters.

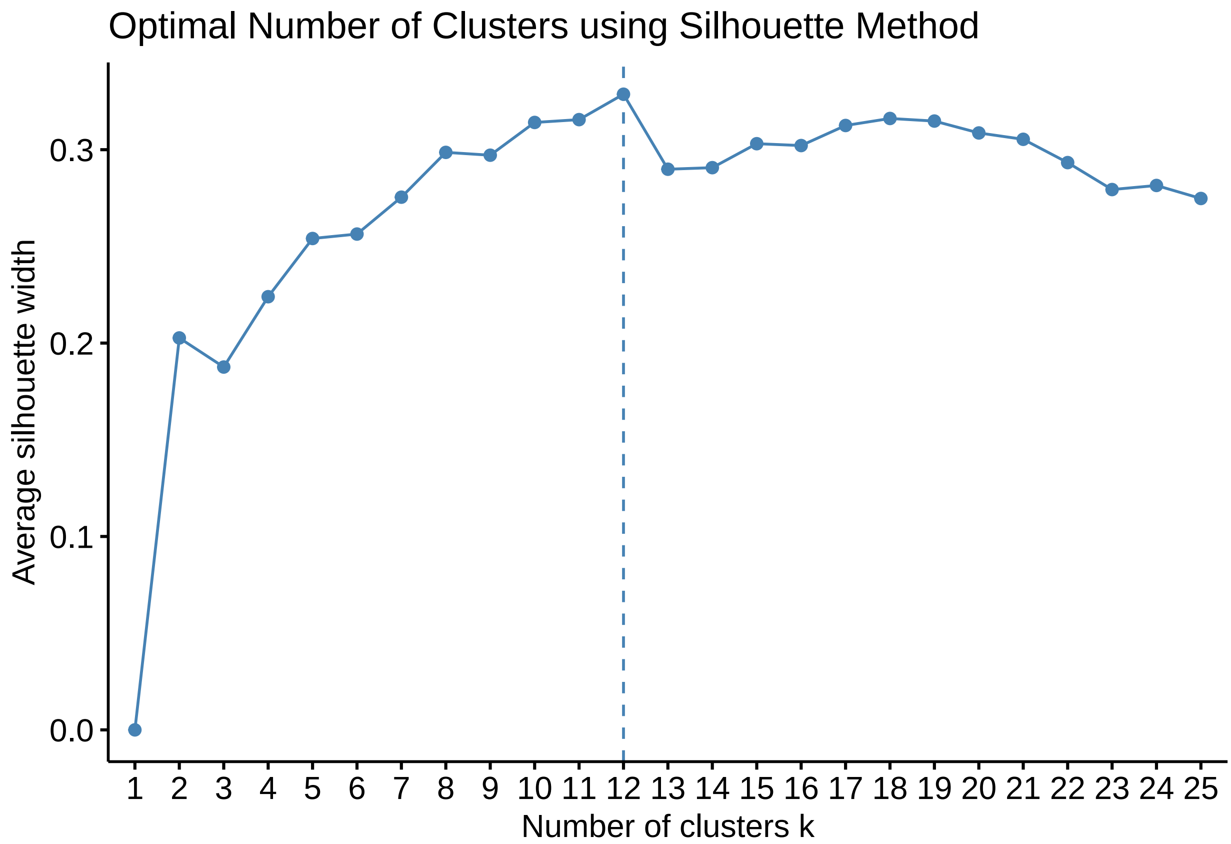
Elbow Method:

```
fviz_nbclust(Preprocessed_Cereal[ , c(4:16)], hcut, method = "wss", k.max = 25) +
  labs(title = "Optimal Number of Clusters using Elbow Method") +
  geom_vline(xintercept = 12, linetype = 2)
```



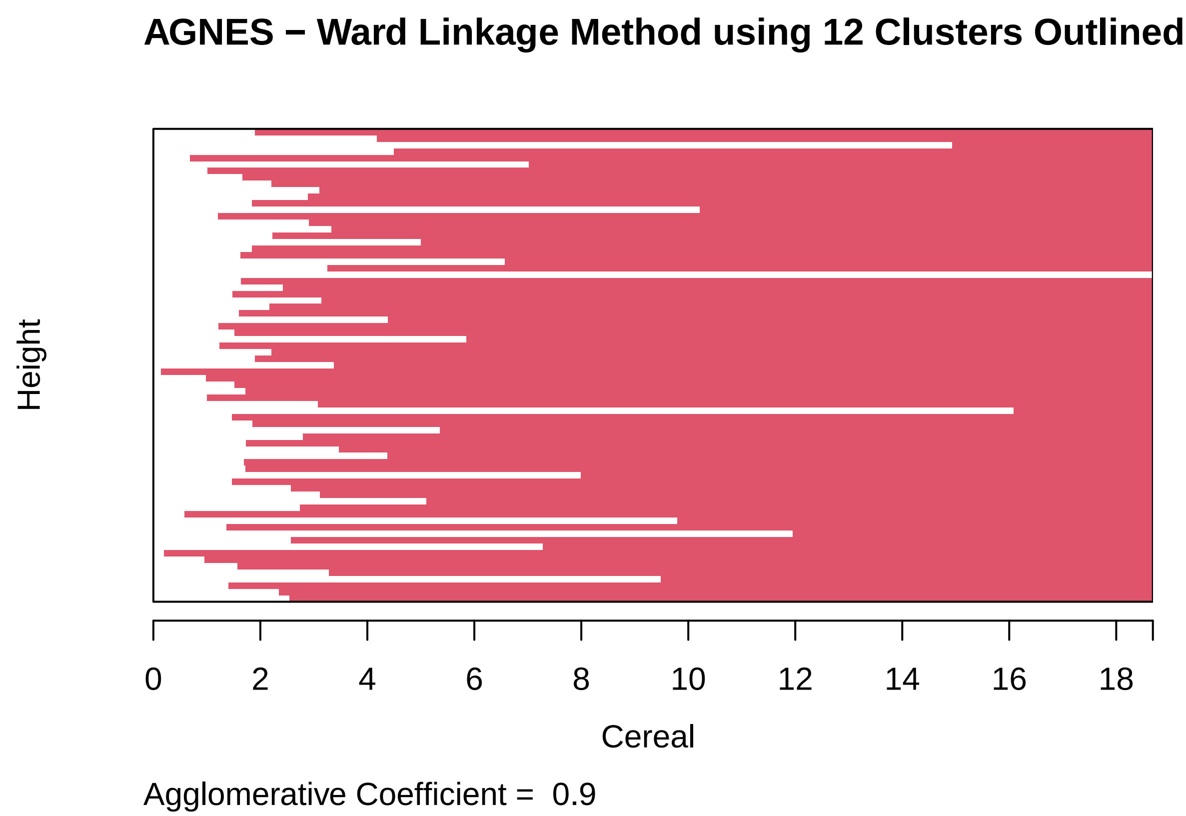
##Silhouette Method:

```
fviz_nbclust(Preprocessed_Cereal[ , c(4:16)],
                               hcut,
                               method = "silhouette",
                               k.max = 25) +
  labs(title = "Optimal Number of Clusters using Silhouette Method")
```

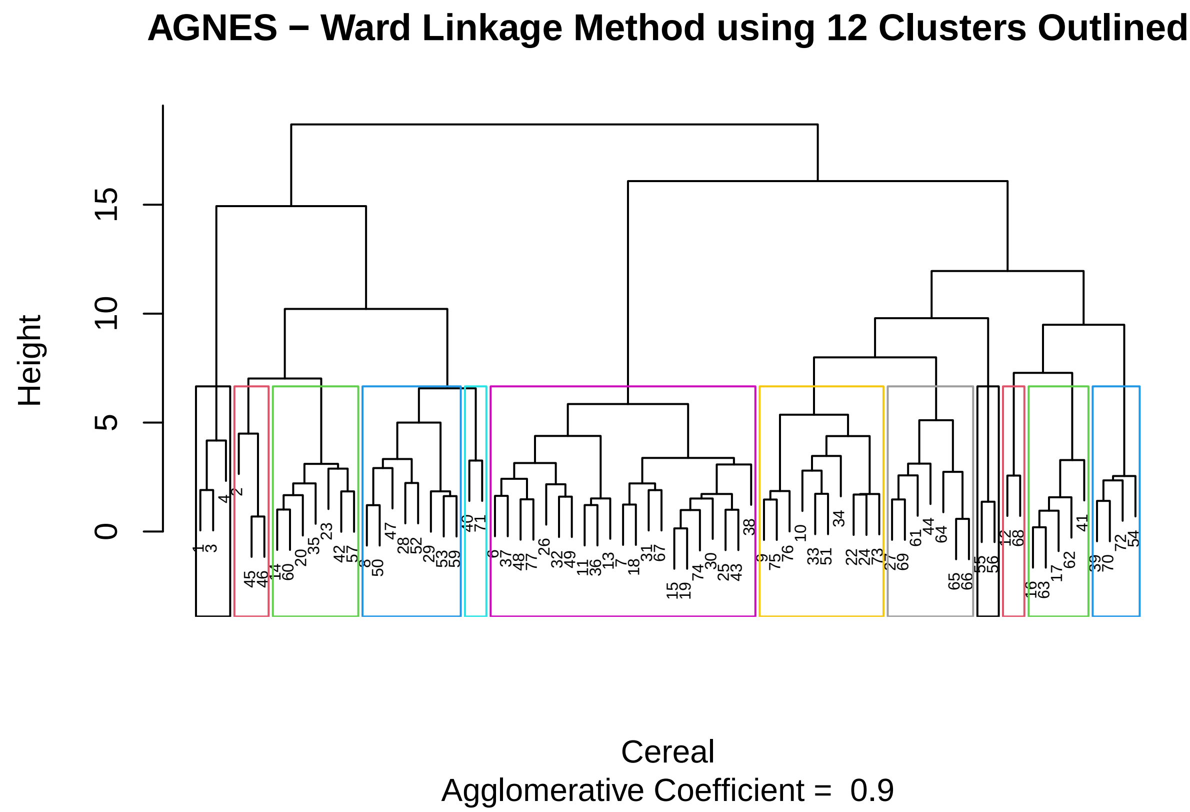


The findings of the elbow and silhouette approaches show that 12 clusters would be the ideal quantity.

```
# Here, I'm plotting the Ward hierarchical tree with the 12 groups highlighted for reference.
plot(HC_Ward,
     main = "AGNES - Ward Linkage Method using 12 Clusters Outlined",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.50,)
```




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



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?

Normalizing the data would not be suitable in this case because the nutritional information for cereal is standardized based on the sample of cereal being evaluated. As a result, only cereals with a very high sugar content and very little fiber, iron, or other nutritional information could be included in the data that was gathered. It is hard to predict how much nourishment the cereal will provide a child once it has been normalized throughout the sample set. However, it is possible that a cereal with an iron level of 0.999 is merely the best of the worst in the sample set and has no nutritional value. We might suppose that a cereal with an iron level of 0.999 contains practically all of the nutritional iron that a child needs. A better way to preprocess the data would be to convert it to a ratio of the daily recommended amounts of calories, fiber, carbohydrates, and other nutrients for a youngster. This would prevent a small number of significant variables from overriding the distance estimates and enable analysts to make more informed cluster decisions during the review phase. An analyst may look at the cluster average while looking at the clusters to figure out what proportion of a student's daily nutritional requirements would be satisfied by XX cereal. This would enable workers to make informed selections about which "healthy" cereal clusters to select.