Assignment-5

TarakRam Nunna

29/11/2021

```
# 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(tidyverse)

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

## v tibble  3.1.4     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

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

library(cluster)
library(factoextra)

## 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(NbClust)
library(ppclust)

## Warning: package 'ppclust' was built under R version 4.1.2

library(dendextend)

## Warning: package 'dendextend' was built under R version 4.1.2

##
## ---------------------
## 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

# Now, import the "cereal" data set into the RStudio environment.
cereals <- read.csv("cereals.csv")
## Reviewing the Data Set
# Review first few rows of the data set
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
```

```
# analyse the structure of the data set
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 ...
```

```
# analyse the summary of the data set
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 should be scaled prior to removing the NA values from the data set.

```
# Create duplicate of data set for preprocessing
cereal_scaled <- cereals
# Scale the data set prior to placing it into a clustering algorithm
cereal_scaled[ , c(4:16)] <- scale(cereals[ , c(4:16)])
# Remove NA values from data set
cereal_preprocessed <- na.omit(cereal_scaled)
# Review the scaled data set with NA's removed
head(cereal_preprocessed)
```

```
##                        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
```

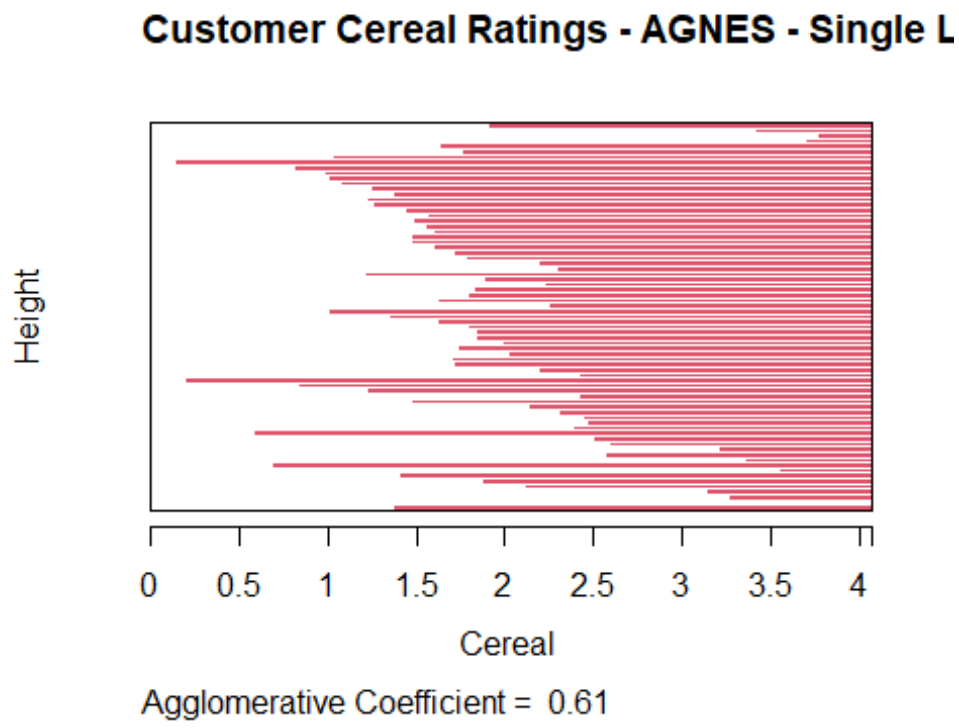
```
0.9515734
## 3  1.1967306  2.87327158 -1.9969238 -0.4627711  3.1434645 -0.1453172
0.9515734
## 4 -0.2346986  4.97114672 -1.7409943 -1.6046739  3.2854885 -0.1453172
0.9515734
## 6  0.2424445 -0.27354112 -1.1011705  0.6791317 -0.4071355 -0.1453172 -
1.4507595
## 7 -0.4136273 -0.48332864 -0.9732057  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
```

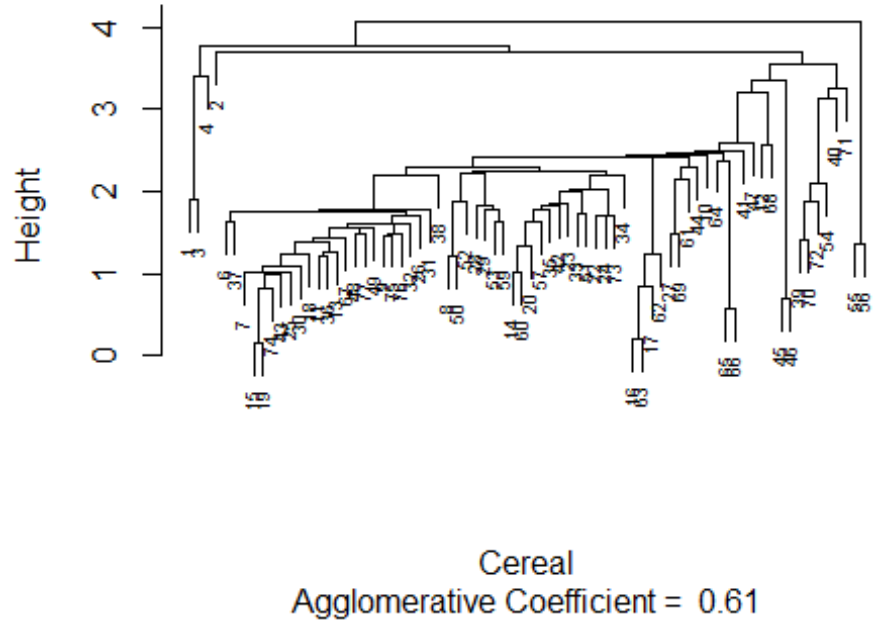
After pre-processing and scaling the data, the total number of observations went from 77 to 74. Therefore, there were only 3 records with "NA" value.

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:

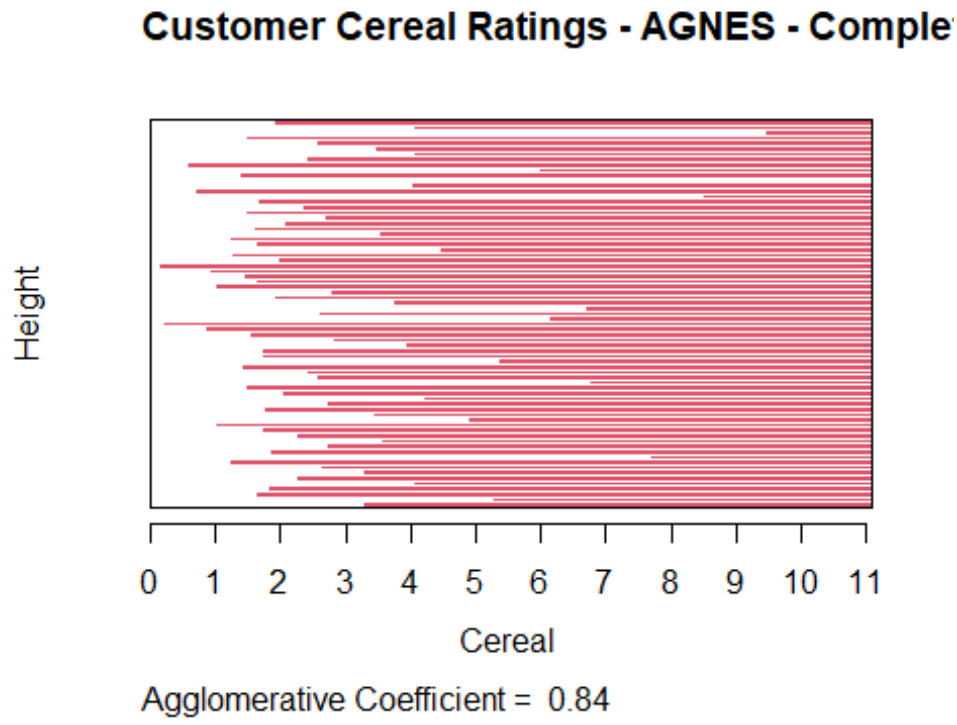
```
# Create the dissimilarity matrix for the numeric values in the data set via
Euclidean distance measurements
cereal_d_euclidean <- dist(cereal_preprocessed[ , c(4:16)], method =
"euclidean")
# Perform hierarchical clustering via the single linkage method
ag_hc_single <- agnes(cereal_d_euclidean, method = "single")
# Plot the results of the different methods
plot(ag_hc_single,
     main = "Customer Cereal Ratings - AGNES - Single Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55)
```

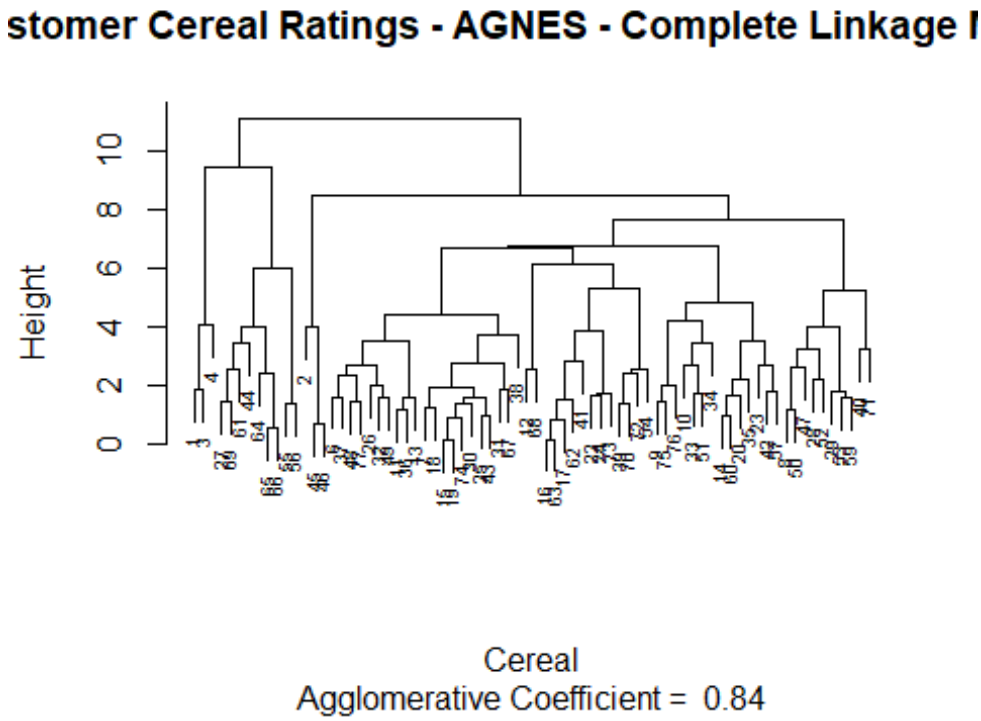

Customer Cereal Ratings - AGNES - Single L
Height
0
0.5
1
1.5
2
2.5
3
3.5
4
Cereal
Agglomerative Coefficient = 0.61


ustomer Cereal Ratings - AGNES - Single Linkage M
Height
0
1
2
3
4
Cereal
Agglomerative Coefficient = 0.61

Complete Linkage:

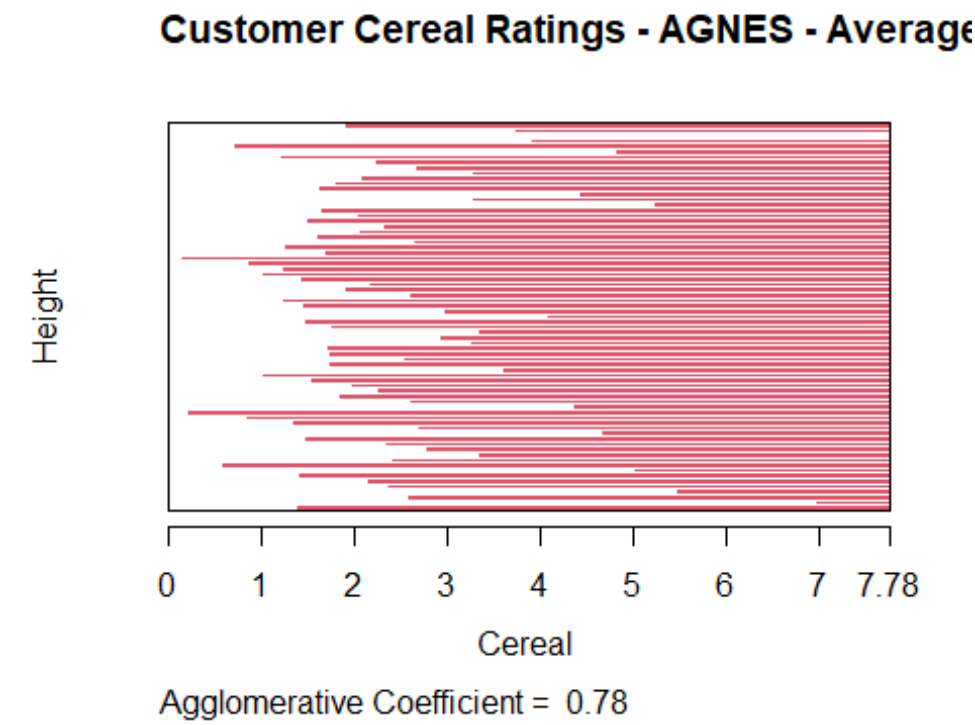
```
# Perform hierarchical clustering via the complete linkage method
ag_hc_complete <- agnes(cereal_d_euclidean, method = "complete")
# Plot the results of the different methods
plot(ag_hc_complete,
     main = "Customer Cereal Ratings - AGNES - Complete Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55)
```

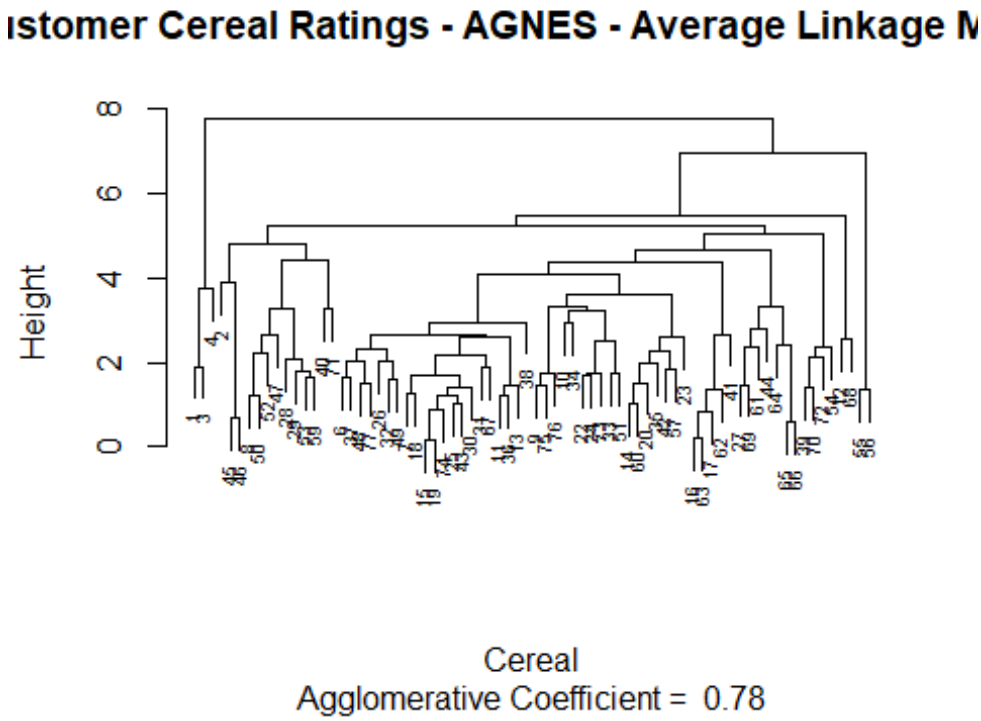

Customer Cereal Ratings - AGNES - Comple
Height
0 1 2 3 4 5 6 7 8 9 10 11
Cereal
Agglomerative Coefficient = 0.84


stomer Cereal Ratings - AGNES - Complete Linkage I
Height
0 2 4 6 8 10
Cereal
Agglomerative Coefficient = 0.84

Average Linkage:

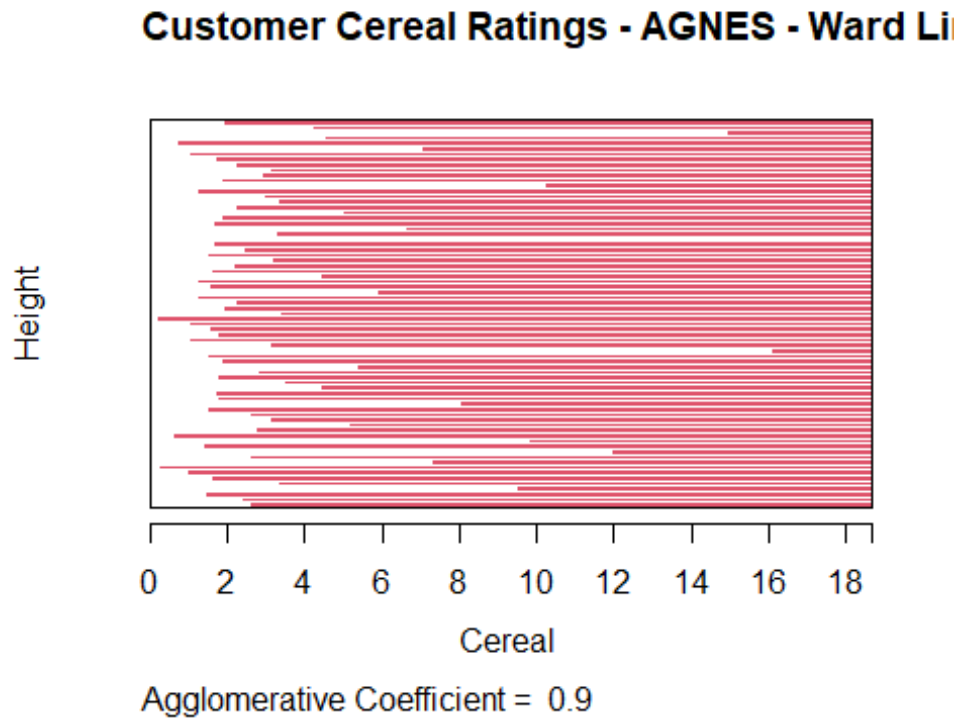
```
# Perform hierarchical clustering via the average linkage method
ag_hc_average <- agnes(cereal_d_euclidean, method = "average")
# Plot the results of the different methods
plot(ag_hc_average,
     main = "Customer Cereal Ratings - AGNES - Average Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55)
```

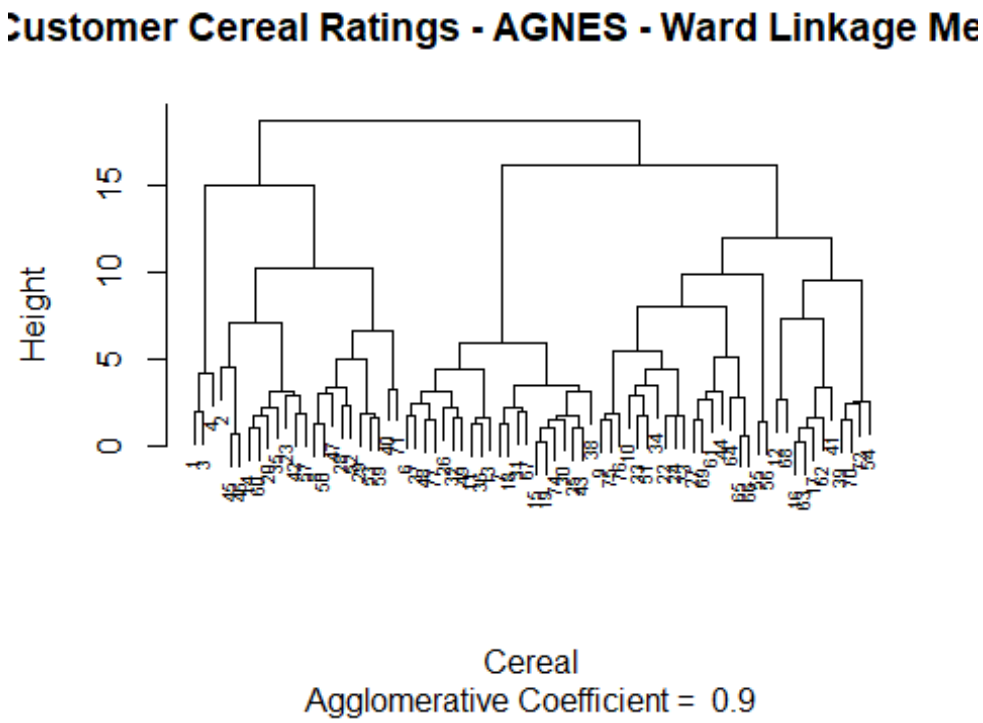

Customer Cereal Ratings - AGNES - Average
Height
0
1
2
3
4
5
6
7
7.78
Cereal
Agglomerative Coefficient = 0.78


istomer Cereal Ratings - AGNES - Average Linkage M
Height
0
2
4
6
8
Cereal
Agglomerative Coefficient = 0.78

Ward Method:

```
# Perform hierarchical clustering via the ward linkage method
ag_hc_ward <- agnes(cereal_d_euclidean, method = "ward")
# Plot the results of the different methods
plot(ag_hc_ward,
     main = "Customer Cereal Ratings - AGNES - Ward Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55)
```





The best clustering method would be based on the agglomerative coefficient that is returned from each method. The close the value is to 1.0, the closer the clustering structure is. Therefore, the method with the value closest to 1.0 will be chosen.

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

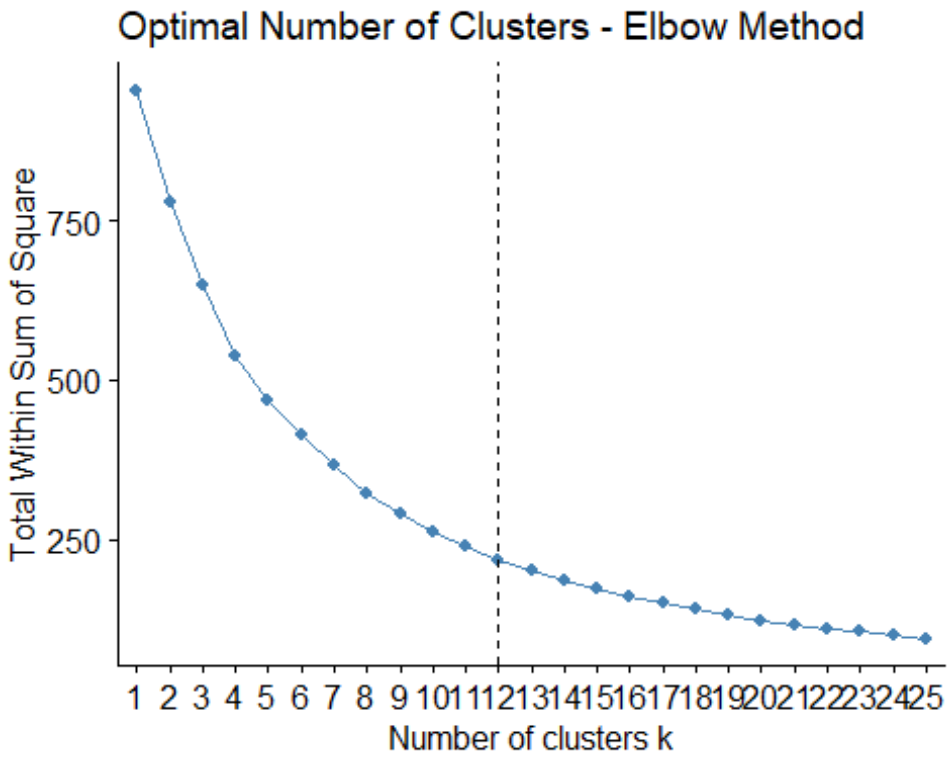
From the result, the Ward method will be chosen as the best clustering model.

Q) How many clusters would you choose?

To determine the appropriate number of clusters, we will use the elbow and silhouette methods.

Elbow Method:

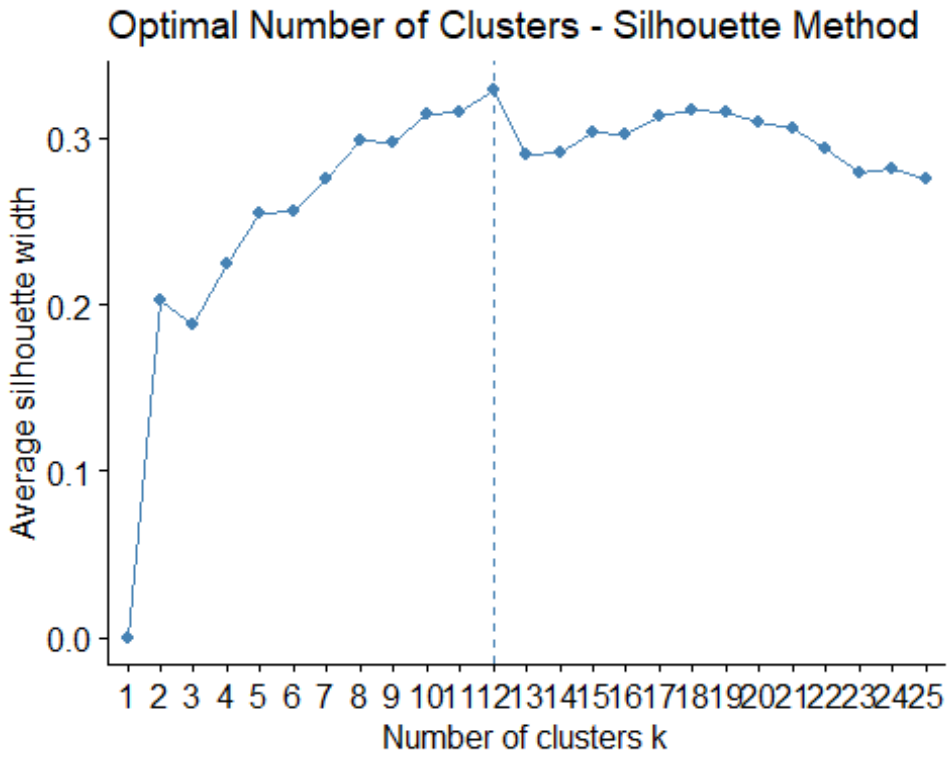
```
# Determine the optimal number of clusters for the dataset via the Elbow method
fviz_nbclust(cereal_preprocessed[ , c(4:16)], hcut, method = "wss", k.max = 25) +
  labs(title = "Optimal Number of Clusters - Elbow Method") +
  geom_vline(xintercept = 12, linetype = 2)
```



Silhouette Method:

```
# Determine the optimal number of clusters for the dataset via the silhouette method
fviz_nbclust(cereal_preprocessed[ , c(4:16)],
                                  hcut,
```

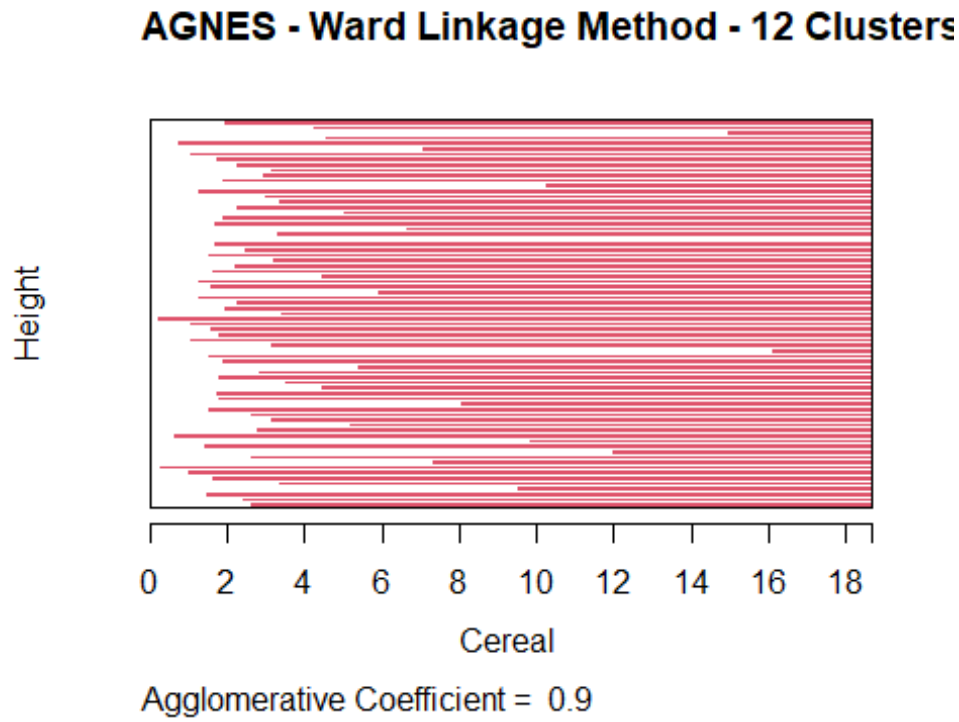
```
                                method = "silhouette",
                                k.max = 25) +
  labs(title = "Optimal Number of Clusters - Silhouette Method")
```



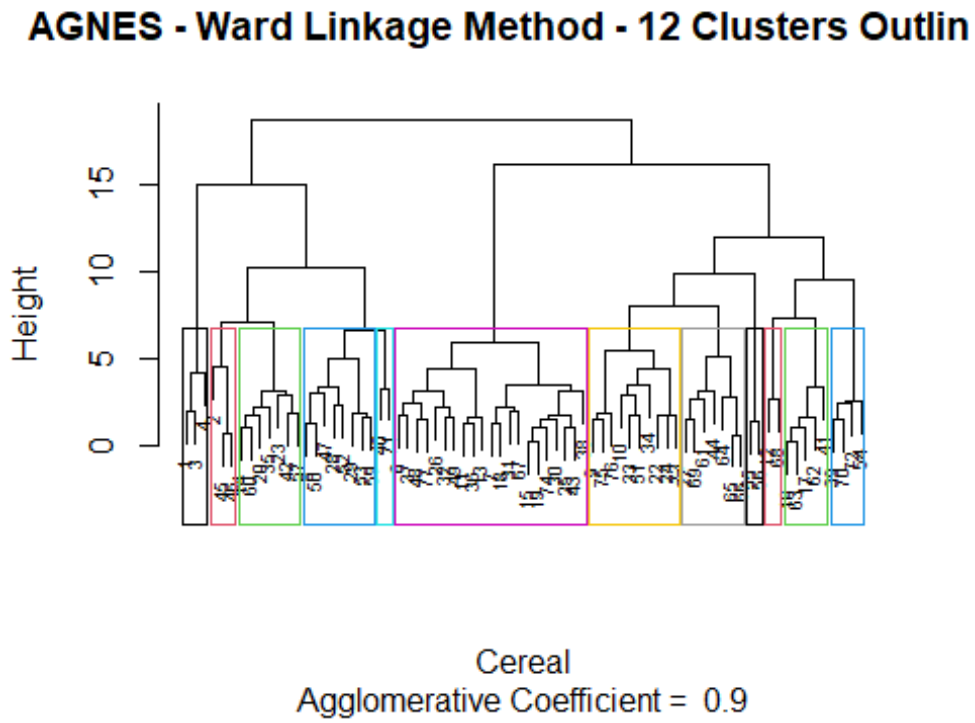
From the results of the elbow and silhouette methods, the number of clusters would be 12.

Now, we will outline the 12 clusters on the hierarchical tree

```
# Plot of the Ward hierarchical tree with the 12 clusters outlined for
reference
plot(ag_hc_ward,
     main = "AGNES - Ward Linkage Method - 12 Clusters Outlined",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55,)
```




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



Q) Comment on the structure of the clusters and on their stability. Hint: To check stability, partition the data and see how well clusters formed based on one part apply to the other part. To do this:

A) Cluster partition A

B) Use the cluster centroids from A to assign each record in partition B (each record is assigned to the cluster with the closest centroid).

C) Assess how consistent the cluster assignments are compared to the assignments based on all the data.

```
# Dividing the tree into 12 clusters for analysis
ward_clusters_12 <- cutree(ag_hc_ward, k = 12)
# Add the assigned cluster to the preprocessed data set
cereal_preprocessed_1 <- cbind(cluster = ward_clusters_12,
cereal_preprocessed)
cereal_preprocessed_1
```

```
##    cluster                                    name mfr type   calories
## 1        1                               100%_Bran   N    C -1.8929836
## 2        2                       100%_Natural_Bran   Q    C  0.6732089
## 3        1                                 All-Bran   K    C -1.8929836
## 4        1                All-Bran_with_Extra_Fiber   K    C -2.9194605
## 6        3                  Apple_Cinnamon_Cheerios   G    C  0.1599704
## 7        3                              Apple_Jacks   K    C  0.1599704
## 8        4                                  Basic_4   G    C  1.1864474
## 9        5                                Bran_Chex   R    C -0.8665066
## 10       5                              Bran_Flakes   P    C -0.8665066
## 11       3                             Cap'n'Crunch   Q    C  0.6732089
## 12       6                                 Cheerios   G    C  0.1599704
## 13       3                    Cinnamon_Toast_Crunch   G    C  0.6732089
## 14       7                                 Clusters   G    C  0.1599704
## 15       3                              Cocoa_Puffs   G    C  0.1599704
## 16       8                                Corn_Chex   R    C  0.1599704
## 17       8                              Corn_Flakes   K    C -0.3532681
## 18       3                                Corn_Pops   K    C  0.1599704
## 19       3                             Count_Chocula   G    C  0.1599704
## 20       7                       Cracklin'_Oat_Bran   K    C  0.1599704
## 22       5                                  Crispix   K    C  0.1599704
## 23       7                    Crispy_Wheat_&_Raisins   G    C -0.3532681
## 24       5                              Double_Chex   R    C -0.3532681
## 25       3                              Froot_Loops   K    C  0.1599704
## 26       3                            Frosted_Flakes   K    C  0.1599704
## 27       9                      Frosted_Mini-Wheats   K    C -0.3532681
```

```
## 28       4 Fruit_&_Fibre_Dates,_Walnuts,_and_Oats   P    C  0.6732089
## 29       4                            Fruitful_Bran   K    C  0.6732089
## 30       3                           Fruity_Pebbles   P    C  0.1599704
## 31       3                             Golden_Crisp   P    C -0.3532681
## 32       3                           Golden_Grahams   G    C  0.1599704
## 33       5                        Grape_Nuts_Flakes   P    C -0.3532681
## 34       5                                Grape-Nuts   P    C  0.1599704
## 35       7                       Great_Grains_Pecan   P    C  0.6732089
## 36       3                         Honey_Graham_Ohs   Q    C  0.6732089
## 37       3                       Honey_Nut_Cheerios   G    C  0.1599704
## 38       3                                Honey-comb   P    C  0.1599704
## 39      10              Just_Right_Crunchy__Nuggets   K    C  0.1599704
## 40      11                   Just_Right_Fruit_&_Nut   K    C  1.6996859
## 41       8                                      Kix   G    C  0.1599704
## 42       7                                     Life   Q    C -0.3532681
## 43       3                              Lucky_Charms   G    C  0.1599704
## 44       9                                     Maypo   A    H -0.3532681
## 45       2         Muesli_Raisins,_Dates,_&_Almonds   R    C  2.2129244
## 46       2        Muesli_Raisins,_Peaches,_&_Pecans   R    C  2.2129244
## 47       4                    Mueslix_Crispy_Blend   K    C  2.7261629
## 48       3                    Multi-Grain_Cheerios   G    C -0.3532681
## 49       3                        Nut&Honey_Crunch   K    C  0.6732089
## 50       4                Nutri-Grain_Almond-Raisin   K    C  1.6996859
## 51       5                        Nutri-grain_Wheat   K    C -0.8665066
## 52       4                     Oatmeal_Raisin_Crisp   G    C  1.1864474
## 53       4                   Post_Nat._Raisin_Bran   P    C  0.6732089
## 54      10                                Product_19   K    C -0.3532681
## 55      12                               Puffed_Rice   Q    C -2.9194605
## 56      12                              Puffed_Wheat   Q    C -2.9194605
## 57       7                       Quaker_Oat_Squares   Q    C -0.3532681
## 59       4                               Raisin_Bran   K    C  0.6732089
## 60       7                            Raisin_Nut_Bran   G    C -0.3532681
## 61       9                             Raisin_Squares   K    C -0.8665066
## 62       8                                  Rice_Chex   R    C  0.1599704
## 63       8                              Rice_Krispies   K    C  0.1599704
## 64       9                             Shredded_Wheat   N    C -1.3797451
## 65       9                  Shredded_Wheat_'n'Bran   N    C -0.8665066
## 66       9                Shredded_Wheat_spoon_size   N    C -0.8665066
## 67       3                                    Smacks   K    C  0.1599704
## 68       6                                 Special_K   K    C  0.1599704
## 69       9                  Strawberry_Fruit_Wheats   N    C -0.8665066
## 70      10                        Total_Corn_Flakes   G    C  0.1599704
## 71      11                        Total_Raisin_Bran   G    C  1.6996859
## 72      10                        Total_Whole_Grain   G    C -0.3532681
## 73       5                                  Triples   G    C  0.1599704
## 74       3                                     Trix   G    C  0.1599704
## 75       5                               Wheat_Chex   R    C -0.3532681
## 76       5                                  Wheaties   G    C -0.3532681
## 77       3                      Wheaties_Honey_Gold   G    C  0.1599704
##       protein         fat       sodium       fiber       carbo
```

```
sugars
## 1   1.3286071 -0.01290349 -0.353984399  3.29284661 -2.50878291 -
0.234390576
## 2   0.4151897  3.96137277 -1.725770770 -0.06375361 -1.74099432
0.222370547
## 3   1.3286071 -0.01290349  1.196730628  2.87327158 -1.99692385 -
0.462771138
## 4   1.3286071 -1.00647256 -0.234698628  4.97114672 -1.74099432 -
1.604673946
## 6  -0.4982277  0.98066557  0.242444457 -0.27354112 -1.10117049
0.679131670
## 7  -0.4982277 -1.00647256 -0.413627285 -0.48332864 -0.97320572
1.592653916
## 8   0.4151897  0.98066557  0.600301771 -0.06375361  0.81830100
0.222370547
## 9  -0.4982277 -0.01290349  0.481016000  0.77539645  0.05051241 -
0.234390576
## 10  0.4151897 -1.00647256  0.600301771  1.19497147 -0.46134666 -
0.462771138
## 11 -1.4116451  0.98066557  0.719587543 -0.90290366 -0.71727619
1.135892793
## 12  3.1554419  0.98066557  1.554587942 -0.06375361  0.56237147 -
1.376293384
## 13 -1.4116451  1.97423464  0.600301771 -0.90290366 -0.46134666
0.450751108
## 14  0.4151897  0.98066557 -0.234698628 -0.06375361 -0.46134666 -
0.006010015
## 15 -1.4116451 -0.01290349  0.242444457 -0.90290366 -0.71727619
1.364273355
## 16 -0.4982277 -1.00647256  1.435302171 -0.90290366  1.84201913 -
0.919532261
## 17 -0.4982277 -1.00647256  1.554587942 -0.48332864  1.58608960 -
1.147912823
## 18 -1.4116451 -1.00647256 -0.831127485 -0.48332864 -0.46134666
1.135892793
## 19 -1.4116451 -0.01290349  0.242444457 -0.90290366 -0.71727619
1.364273355
## 20  0.4151897  1.97423464 -0.234698628  0.77539645 -1.22913525 -
0.006010015
## 22 -0.4982277 -1.00647256  0.719587543 -0.48332864  1.58608960 -
0.919532261
## 23 -0.4982277 -0.01290349 -0.234698628 -0.06375361 -0.97320572
0.679131670
## 24 -0.4982277 -1.00647256  0.361730229 -0.48332864  0.81830100 -
0.462771138
## 25 -0.4982277 -0.01290349 -0.413627285 -0.48332864 -0.97320572
1.364273355
## 26 -1.4116451 -1.00647256  0.481016000 -0.48332864 -0.20541712
0.907512232
## 27  0.4151897 -1.00647256 -1.904699427  0.35582142 -0.20541712 -
```

```
0.006010015
## 28  0.4151897  0.98066557  0.003872915  1.19497147 -0.71727619
0.679131670
## 29  0.4151897 -1.00647256  0.958159085  1.19497147 -0.20541712
1.135892793
## 30 -1.4116451 -0.01290349 -0.294341514 -0.90290366 -0.46134666
1.135892793
## 31 -0.4982277 -1.00647256 -1.367913456 -0.90290366 -0.97320572
1.821034478
## 32 -1.4116451 -0.01290349  1.435302171 -0.90290366  0.05051241
0.450751108
## 33  0.4151897 -0.01290349 -0.234698628  0.35582142  0.05051241 -
0.462771138
## 34  0.4151897 -1.00647256  0.123158686  0.35582142  0.56237147 -
0.919532261
## 35  0.4151897  1.97423464 -1.010056142  0.35582142 -0.46134666 -
0.691151699
## 36 -1.4116451  0.98066557  0.719587543 -0.48332864 -0.71727619
0.907512232
## 37  0.4151897 -0.01290349  1.077444857 -0.27354112 -0.84524095
0.679131670
## 38 -1.4116451 -1.00647256  0.242444457 -0.90290366 -0.20541712
0.907512232
## 39 -0.4982277 -0.01290349  0.123158686 -0.48332864  0.56237147 -
0.234390576
## 40  0.4151897 -0.01290349  0.123158686 -0.06375361  1.33016007
0.450751108
## 41 -0.4982277 -0.01290349  1.196730628 -0.90290366  1.58608960 -
0.919532261
## 42  1.3286071  0.98066557 -0.115412857 -0.06375361 -0.71727619 -
0.234390576
## 43 -0.4982277 -0.01290349  0.242444457 -0.90290366 -0.71727619
1.135892793
## 44  1.3286071 -0.01290349 -1.904699427 -0.90290366  0.30644194 -
0.919532261
## 45  1.3286071  1.97423464 -0.771484599  0.35582142  0.30644194
0.907512232
## 46  1.3286071  1.97423464 -0.115412857  0.35582142  0.30644194
0.907512232
## 47  0.4151897  0.98066557 -0.115412857  0.35582142  0.56237147
1.364273355
## 48 -0.4982277 -0.01290349  0.719587543 -0.06375361  0.05051241 -
0.234390576
## 49 -0.4982277 -0.01290349  0.361730229 -0.90290366  0.05051241
0.450751108
## 50  0.4151897  0.98066557  0.719587543  0.35582142  1.58608960 -
0.006010015
## 51  0.4151897 -1.00647256  0.123158686  0.35582142  0.81830100 -
1.147912823
## 52  0.4151897  0.98066557  0.123158686 -0.27354112 -0.33338189
```

```
0.679131670
## 53  0.4151897 -0.01290349  0.481016000  1.61454650 -0.97320572
1.592653916
## 54  0.4151897 -1.00647256  1.912445256 -0.48332864  1.33016007 -
0.919532261
## 55 -1.4116451 -1.00647256 -1.904699427 -0.90290366 -0.46134666 -
1.604673946
## 56 -0.4982277 -1.00647256 -1.904699427 -0.48332864 -1.22913525 -
1.604673946
## 57  1.3286071 -0.01290349 -0.294341514 -0.06375361 -0.20541712 -
0.234390576
## 59  0.4151897 -0.01290349  0.600301771  1.19497147 -0.20541712
1.135892793
## 60  0.4151897  0.98066557 -0.234698628  0.14603391 -1.10117049
0.222370547
## 61 -0.4982277 -1.00647256 -1.904699427 -0.06375361  0.05051241 -
0.234390576
## 62 -1.4116451 -1.00647256  0.958159085 -0.90290366  2.09794866 -
1.147912823
## 63 -0.4982277 -1.00647256  1.554587942 -0.90290366  1.84201913 -
0.919532261
## 64 -0.4982277 -1.00647256 -1.904699427  0.35582142  0.30644194 -
1.604673946
## 65  0.4151897 -1.00647256 -1.904699427  0.77539645  1.07423054 -
1.604673946
## 66  0.4151897 -1.00647256 -1.904699427  0.35582142  1.33016007 -
1.604673946
## 67 -0.4982277 -0.01290349 -1.069699027 -0.48332864 -1.48506478
1.821034478
## 68  3.1554419 -1.00647256  0.838873314 -0.48332864  0.30644194 -
0.919532261
## 69 -0.4982277 -1.00647256 -1.725770770  0.35582142  0.05051241 -
0.462771138
## 70 -0.4982277 -0.01290349  0.481016000 -0.90290366  1.58608960 -
0.919532261
## 71  0.4151897 -0.01290349  0.361730229  0.77539645  0.05051241
1.592653916
## 72  0.4151897 -0.01290349  0.481016000  0.35582142  0.30644194 -
0.919532261
## 73 -0.4982277 -0.01290349  1.077444857 -0.90290366  1.58608960 -
0.919532261
## 74 -1.4116451 -0.01290349 -0.234698628 -0.90290366 -0.46134666
1.135892793
## 75  0.4151897 -0.01290349  0.838873314  0.35582142  0.56237147 -
0.919532261
## 76  0.4151897 -0.01290349  0.481016000  0.35582142  0.56237147 -
0.919532261
## 77 -0.4982277 -0.01290349  0.481016000 -0.48332864  0.30644194
0.222370547
##         potass   vitamins      shelf     weight        cups      rating
```

```
## 1   2.57536849 -0.1453172  0.9515734 -0.1967771 -2.11003399  1.83218758
## 2   0.51602052 -1.2642598  0.9515734 -0.1967771  0.76901001 -0.61805706
## 3   3.14346448 -0.1453172  0.9515734 -0.1967771 -2.11003399  1.19309856
## 4   3.28548848 -0.1453172  0.9515734 -0.1967771 -1.37953029  3.63338491
## 6  -0.40713546 -0.1453172 -1.4507595 -0.1967771 -0.30526014 -0.93656251
## 7  -0.97523145 -0.1453172 -0.2495930 -0.1967771  0.76901001 -0.67568989
## 8   0.01893653 -0.1453172  0.9515734  1.9962520 -0.30526014 -0.40058570
## 9   0.37399653 -0.1453172 -1.4507595 -0.1967771 -0.64902659  0.45948710
## 10  1.29715251 -0.1453172  0.9515734 -0.1967771 -0.64902659  0.75801873
## 11 -0.90421945 -0.1453172 -0.2495930 -0.1967771 -0.30526014 -1.75285455
## 12  0.08994853 -0.1453172 -1.4507595 -0.1967771  1.84328015  0.57657347
## 13 -0.76219545 -0.1453172 -0.2495930 -0.1967771 -0.30526014 -1.62608831
## 14  0.08994853 -0.1453172  0.9515734 -0.1967771 -1.37953029 -0.16127646
## 15 -0.62017146 -0.1453172 -0.2495930 -0.1967771  0.76901001 -1.41872637
## 16 -1.04624345 -0.1453172 -1.4507595 -0.1967771  0.76901001 -0.08689833
## 17 -0.90421945 -0.1453172 -1.4507595 -0.1967771  0.76901001  0.22763247
## 18 -1.11725545 -0.1453172 -0.2495930 -0.1967771  0.76901001 -0.48998167
## 19 -0.47814746 -0.1453172 -0.2495930 -0.1967771  0.76901001 -1.44292556
## 20  0.87108052 -0.1453172  0.9515734 -0.1967771 -1.37953029 -0.15781928
## 22 -0.97523145 -0.1453172  0.9515734 -0.1967771  0.76901001  0.30112138
## 23  0.30298453 -0.1453172  0.9515734 -0.1967771 -0.30526014 -0.46197591
## 24 -0.26511146 -0.1453172  0.9515734 -0.1967771 -0.30526014  0.11853896
## 25 -0.97523145 -0.1453172 -0.2495930 -0.1967771  0.76901001 -0.74449406
## 26 -1.04624345 -0.1453172 -1.4507595 -0.1967771 -0.30526014 -0.79942345
## 27  0.01893653 -0.1453172 -0.2495930 -0.1967771 -0.09040611  1.11618949
## 28  1.43917651 -0.1453172  0.9515734  1.4646086 -0.64902659 -0.12448367
## 29  1.29715251 -0.1453172  0.9515734  1.9962520 -0.64902659 -0.11747555
## 30 -1.04624345 -0.1453172 -0.2495930 -0.1967771 -0.30526014 -1.04218972
## 31 -0.83320745 -0.1453172 -1.4507595 -0.1967771  0.25336034 -0.52773607
## 32 -0.76219545 -0.1453172 -0.2495930 -0.1967771 -0.30526014 -1.34272615
## 33 -0.19409946 -0.1453172  0.9515734 -0.1967771  0.25336034  0.66996501
## 34 -0.12308746 -0.1453172  0.9515734 -0.1967771 -2.45380043  0.76209027
## 35  0.01893653 -0.1453172  0.9515734 -0.1967771 -2.11003399  0.22395859
## 36 -0.76219545 -0.1453172 -0.2495930 -0.1967771  0.76901001 -1.48031505
## 37 -0.12308746 -0.1453172 -1.4507595 -0.1967771 -0.30526014 -0.82531855
## 38 -0.90421945 -0.1453172 -1.4507595 -0.1967771  2.18704660 -0.99117283
## 39 -0.54915946  3.2115106  0.9515734 -0.1967771  0.76901001 -0.43723896
## 40 -0.05207547  3.2115106  0.9515734  1.7968857 -0.30526014 -0.44095292
## 41 -0.83320745 -0.1453172 -0.2495930 -0.1967771  2.91755030 -0.24379018
## 42 -0.05207547 -0.1453172 -0.2495930 -0.1967771 -0.64902659  0.18952903
## 43 -0.62017146 -0.1453172 -0.2495930 -0.1967771  0.76901001 -1.13411138
## 44 -0.05207547 -0.1453172 -0.2495930 -0.1967771  0.76901001  0.86744227
## 45  1.01310452 -0.1453172  0.9515734 -0.1967771  0.76901001 -0.39358784
## 46  1.01310452 -0.1453172  0.9515734 -0.1967771  0.76901001 -0.60694559
## 47  0.87108052 -0.1453172  0.9515734  3.1259942 -0.64902659 -0.87934079
## 48 -0.12308746 -0.1453172 -1.4507595 -0.1967771  0.76901001 -0.18222306
## 49 -0.83320745 -0.1453172 -0.2495930 -0.1967771 -0.64902659 -0.90703767
## 50  0.44500853 -0.1453172  0.9515734  1.9962520 -0.64902659 -0.14048156
## 51 -0.12308746 -0.1453172  0.9515734 -0.1967771  0.76901001  1.20857002
## 52  0.30298453 -0.1453172  0.9515734  1.4646086 -1.37953029 -0.86955299
```

```
## 53  2.29132050 -0.1453172  0.9515734  1.9962520 -0.64902659 -0.34349055
## 54 -0.76219545  3.2115106  0.9515734 -0.1967771  0.76901001 -0.08273233
## 55 -1.18826745 -1.2642598  0.9515734 -3.5195485  0.76901001  1.28782197
## 56 -0.69118346 -1.2642598  0.9515734 -3.5195485  0.76901001  1.44796198
## 57  0.16096053 -0.1453172  0.9515734 -0.1967771 -1.37953029  0.48736586
## 59  2.00727250 -0.1453172 -0.2495930  1.9962520 -0.30526014 -0.24250288
## 60  0.58703252 -0.1453172  0.9515734 -0.1967771 -1.37953029 -0.21088091
## 61  0.16096053 -0.1453172  0.9515734 -0.1967771 -1.37953029  0.90177096
## 62 -0.97523145 -0.1453172 -1.4507595 -0.1967771  1.32763048 -0.04746624
## 63 -0.90421945 -0.1453172 -1.4507595 -0.1967771  0.76901001 -0.14988985
## 64 -0.05207547 -1.2642598 -1.4507595 -1.3265194  0.76901001  1.82029290
## 65  0.58703252 -1.2642598 -1.4507595 -0.1967771 -0.64902659  2.26429773
## 66  0.30298453 -1.2642598 -1.4507595 -0.1967771 -0.64902659  2.14533086
## 67 -0.83320745 -0.1453172 -0.2495930 -0.1967771 -0.30526014 -0.81408243
## 68 -0.62017146 -0.1453172 -1.4507595 -0.1967771  0.76901001  0.74502768
## 69 -0.12308746 -0.1453172 -0.2495930 -0.1967771  0.76901001  1.18871964
## 70 -0.90421945  3.2115106  0.9515734 -0.1967771  0.76901001 -0.27236281
## 71  1.86524850  3.2115106  0.9515734  3.1259942  0.76901001 -1.00182464
## 72  0.16096053  3.2115106  0.9515734 -0.1967771  0.76901001  0.28426404
## 73 -0.54915946 -0.1453172  0.9515734 -0.1967771 -0.30526014 -0.25339630
## 74 -1.04624345 -0.1453172 -0.2495930 -0.1967771  0.76901001 -1.06158592
## 75  0.23197253 -0.1453172 -1.4507595 -0.1967771 -0.64902659  0.50698324
## 76  0.16096053 -0.1453172 -1.4507595 -0.1967771  0.76901001  0.63545985
## 77 -0.54915946 -0.1453172 -1.4507595 -0.1967771 -0.30526014 -0.46116700
```

The assigned clusters for all data sets will be in “cereal_preprocessed_1”

Partition Data:

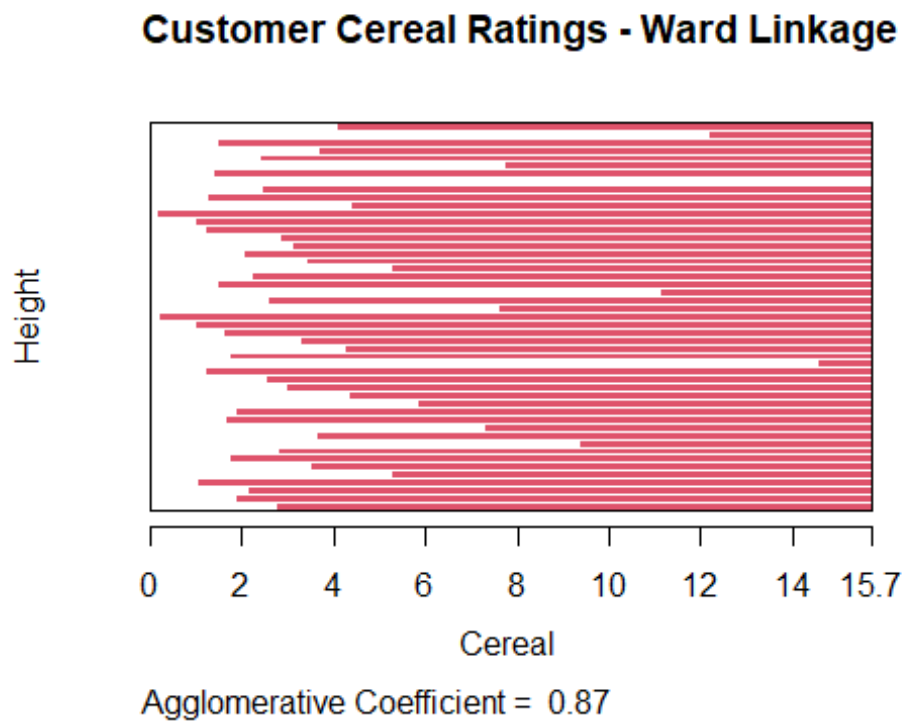
To check stability of clusters, the data set will be split into a 70% and 30%. The 70% will be used to create cluster assignments again, and then the remaining 30% will be assigned based on their closest centroid.

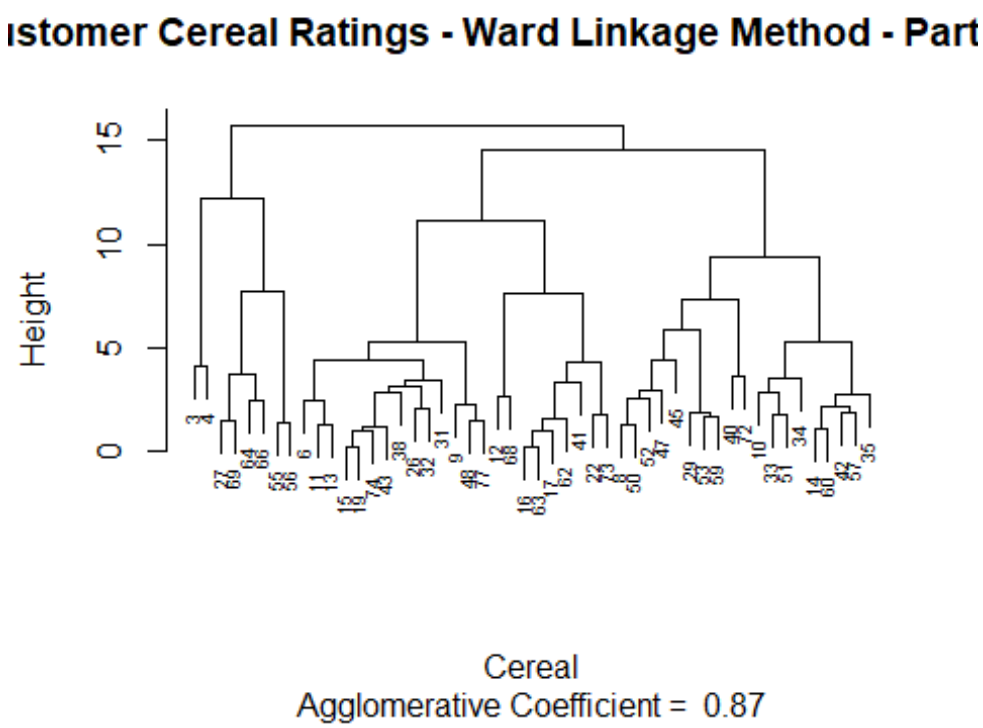
```
set.seed(300)
# Split the data into 70% partition A and 30% partition B
cerealIndex <- createDataPartition(cereal_preprocessed$protein, p=0.3, list =
F)
cereal_preprocessed_PartitionB <- cereal_preprocessed[cerealIndex, ]
cereal_preprocessed_PartitionA <- cereal_preprocessed[-cerealIndex,]
```

Re-Run Clustering with Partitioned Data:

For the purposes of this task, we will assume the same K value (12) and ward clustering method to determine the stability of the clusters. We will then assign clusters to the nearest points in Partition B (for clusters 1 to 12).

```
# Create the dissimilarity matrix for the numeric values in the partitioned
data set via Euclidean distance measurements
cereal_d_euclidean_A <- dist(cereal_preprocessed_PartitionA[ , c(4:16)],
method = "euclidean")
# Perform hierarchical clustering via the ward linkage method on partitioned
data
ag_hc_ward_A <- agnes(cereal_d_euclidean_A, method = "ward")
# Plot the results of the different methods
plot(ag_hc_ward_A,
     main = "Customer Cereal Ratings - Ward Linkage Method - Partition A",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55)
```






```
# Cut the tree into 12 clusters for analysis
ward_clusters_12_A <- cutree(ag_hc_ward_A, k = 12)
# Add the assigned cluster to the preprocessed data set
```

```
cereal_preprocessed_A <- cbind(cluster = ward_clusters_12_A,
cereal_preprocessed_PartitionA)
```

The centroids for each of the clusters will need to be calculated, so we can find the closest centroid for the data points in partition B.

```
# Find the centroids for the re-ran Ward hierarchical clustering
ward_Centroids_A <- aggregate(cereal_preprocessed_A[ , 5:17],
list(cereal_preprocessed_A$cluster), mean)
ward_Centroids_A <- data.frame(Cluster = ward_Centroids_A[ , 1], Centroid =
rowMeans(ward_Centroids_A[ , -c(1:4)]))
ward_Centroids_A <- ward_Centroids_A$Centroid

# Calculate Centers of Partition B data set
cereal_preprocessed_PartitionB_centers <-
data.frame(cereal_preprocessed_PartitionB[, 1:3], Center =
rowMeans(cereal_preprocessed_PartitionB[ , 4:16]))

# Calculate the distance between the centers of partition A and the values of
partition B
B_to_A_centers <- dist(ward_Centroids_A,
cereal_preprocessed_PartitionB_centers$Center, method = "euclidean")
# Assign the clusters based on the minimum distance to cluster centers
cereal_preprocessed_B <- cbind(cluster =
c(4,8,7,3,5,6,7,11,11,10,8,5,10,1,10,1,4,12,12,7,7,1,4,9),
cereal_preprocessed_PartitionB)
# Combine partitions A and B for comparision to original clusters
cereal_preprocessed_2 <- rbind(cereal_preprocessed_A, cereal_preprocessed_B)
cereal_preprocessed_1 <-
cereal_preprocessed_1[order(cereal_preprocessed_1$name), ]
cereal_preprocessed_2 <-
cereal_preprocessed_2[order(cereal_preprocessed_2$name), ]
```

Now that the data has been assigned by both methods (full data and partitioned data), we can compare the number of matching assignments to see the stability of the clusters.

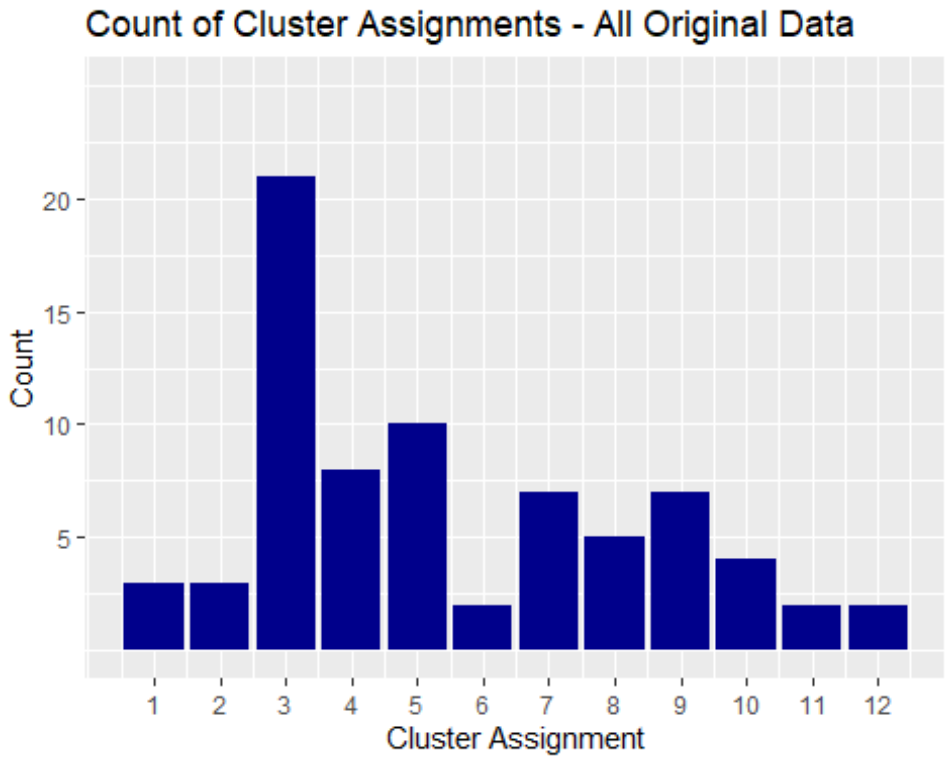
```
sum(cereal_preprocessed_1$cluster == cereal_preprocessed_2$cluster)

## [1] 27
```

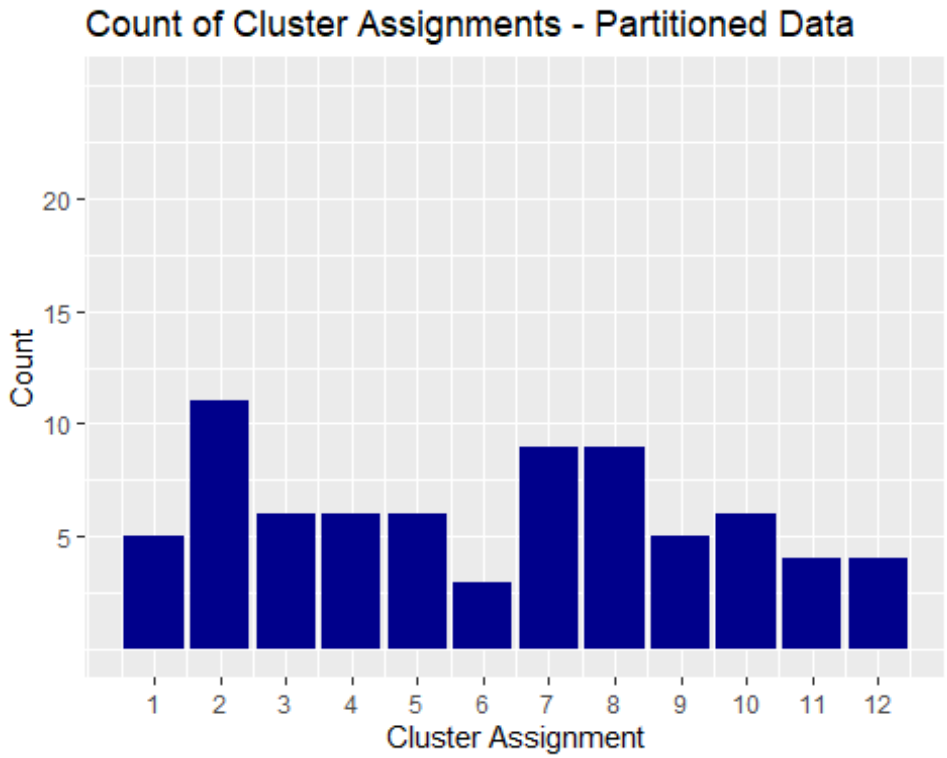
From this result, we can state that the clusters are not very stable. With 70% of the available data, the resulting assignments were only identical for 27 out of the 74 observations. This results in a 36% repeatability of assignment.

```
# Visualize the cluster assignments to see any difference between them
# Plot of original hierarchical clustering algorithm
ggplot(data = cereal_preprocessed_1, aes(cluster)) +
  geom_bar(fill = "blue4") +
  labs(title="Count of Cluster Assignments - All Original Data") +
  labs(x="Cluster Assignment", y="Count") +
```

```
  scale_x_continuous(breaks=c(1:12)) +
  scale_y_continuous(breaks=c(5,10,15,20), limits = c(0,25))
```




```
# Plot of algorithm that was partitioned prior to assigning the remaining
data
ggplot(data = cereal_preprocessed_2, aes(cluster)) +
  geom_bar(fill = "blue4") +
  labs(title="Count of Cluster Assignments - Partitioned Data") +
  labs(x="Cluster Assignment", y="Count") +
  scale_x_continuous(breaks=c(1:12)) +
  scale_y_continuous(breaks=c(5,10,15,20), limits = c(0,25))
```



From above graph, we can see that Cluster 3 significantly shrunk when using the partitioned data. As a result, several of the other clusters became larger as a result. From the chart, it appears the clusters are more evenly distributed across the 12 clusters when the data is partitioned.

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?

A) Normalizing the data would not be suitable in this scenario because the nutritional information for cereal is normalized based on the sample of cereal being evaluated. As a result, the collected data could only contain cereals with extremely high sugar content and very little fiber, iron, and other nutritional data. It's impossible to say how much nourishment the cereal will provide a child once it's been normalized throughout the sample set. We may infer that a cereal with an iron content of 0.999 means it contains virtually all of the nutrional iron a child need; yet, it could simply be the best of the worst in the sample set (having nearly no nutrional value).

As a result, a better 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 would allow analysts to make more informed decisions on clusters during review, while also preventing a few larger variables from overriding the distance estimates. When looking at the clusters, an analyst may look at the cluster average to see what percentage of a student's daily needed nutrition would come from XX cereal. This would enable the employees to make well-informed selections regarding which "healthy" cereal clusters to select.