Assignment 5 - Hierarchical Clustering

Akhil Kornala

2023-12-03

```
knitr::opts_chunk$set(echo = TRUE, comment = NA)
```

```
getwd()
```

```
[1] "C:/Users/LENOVO/Desktop/KSU/Sem 1/FML/Assignment 5"
```

```
setwd("C:/Users/LENOVO/Desktop/KSU/Sem 1/FML/Assignment 5")
```

```
# 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 4.3.2
```

```r
library(dendextend)
```

```
---------------------
Welcome to dendextend version 1.17.1
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 core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.0
v lubridate 1.9.2     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.0
v readr     2.1.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x purrr::lift()   masks caret::lift()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```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
library(readr)
# To import the data collection "cereal"
Cereals <- Cereals <- read_csv("C:/Users/LENOVO/Desktop/KSU/Sem 1/FML/Assignment 5/Cereals.csv")
```

```
Rows: 77 Columns: 16
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): name, mfr, type
dbl (13): calories, protein, fat, sodium, fiber, carbo, sugars, potass, vita...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Getting the first few rows of the data collection using head
head(Cereals)
```

```
# A tibble: 6 x 16
  name       mfr   type  calories protein   fat sodium fiber carbo sugars potass
  <chr>      <chr> <chr>    <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>
1 100%_Bran  N     C          70       4     1    130  10     5        6    280
2 100%_Natu~ Q     C         120       3     5     15   2     8        8    135
3 All-Bran   K     C          70       4     1    260   9     7        5    320
4 All-Bran_~ K     C          50       4     0    140  14     8        0    330
5 Almond_De~ R     C         110       2     2    200   1    14        8     NA
6 Apple_Cin~ G     C         110       2     2    180   1.5  10.5     10     70
# i 5 more variables: vitamins <dbl>, shelf <dbl>, weight <dbl>, cups <dbl>,
#   rating <dbl>
```

```r
# Analyzing the data set's structure with str
str(Cereals)
```

```
spc_tbl_ [77 x 16] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ name    : chr [1:77] "100%_Bran" "100%_Natural_Bran" "All-Bran" "All-Bran_with_Extra_Fiber" ...
 $ mfr     : chr [1:77] "N" "Q" "K" "K" ...
 $ type    : chr [1:77] "C" "C" "C" "C" ...
 $ calories: num [1:77] 70 120 70 50 110 110 110 130 90 90 ...
 $ protein : num [1:77] 4 3 4 4 2 2 2 3 2 3 ...
 $ fat     : num [1:77] 1 5 1 0 2 2 0 2 1 0 ...
 $ sodium  : num [1:77] 130 15 260 140 200 180 125 210 200 210 ...
 $ fiber   : num [1:77] 10 2 9 14 1 1.5 1 2 4 5 ...
 $ carbo   : num [1:77] 5 8 7 8 14 10.5 11 18 15 13 ...
 $ sugars  : num [1:77] 6 8 5 0 8 10 14 8 6 5 ...
 $ potass  : num [1:77] 280 135 320 330 NA 70 30 100 125 190 ...
 $ vitamins: num [1:77] 25 0 25 25 25 25 25 25 25 25 ...
 $ shelf   : num [1:77] 3 3 3 3 3 1 2 3 1 3 ...
 $ weight  : num [1:77] 1 1 1 1 1 1 1 1.33 1 1 ...
 $ cups    : num [1:77] 0.33 1 0.33 0.5 0.75 0.75 1 0.75 0.67 0.67 ...
 $ rating  : num [1:77] 68.4 34 59.4 93.7 34.4 ...
 - attr(*, "spec")=
  .. cols(
  ..   name = col_character(),
  ..   mfr = col_character(),
```

```
  ..   type = col_character(),
  ..   calories = col_double(),
  ..   protein = col_double(),
  ..   fat = col_double(),
  ..   sodium = col_double(),
  ..   fiber = col_double(),
  ..   carbo = col_double(),
  ..   sugars = col_double(),
  ..   potass = col_double(),
  ..   vitamins = col_double(),
  ..   shelf = col_double(),
  ..   weight = col_double(),
  ..   cups = col_double(),
  ..   rating = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
#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)
```

```
# A tibble: 6 x 16
  name  mfr   type  calories protein     fat sodium   fiber  carbo sugars potass
  <chr> <chr> <chr>    <dbl>   <dbl>   <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
1 100%~ N     C       -1.89    1.33  -0.0129 -0.354  3.29   -2.51  -0.234  2.58
2 100%~ Q     C        0.673   0.415  3.96   -1.73  -0.0638 -1.74   0.222  0.516
3 All-~ K     C       -1.89    1.33  -0.0129  1.20   2.87   -2.00  -0.463  3.14
4 All-~ K     C       -2.92    1.33  -1.01   -0.235  4.97   -1.74  -1.60   3.29
5 Appl~ G     C        0.160  -0.498  0.981   0.242 -0.274  -1.10   0.679 -0.407
6 Appl~ K     C        0.160  -0.498 -1.01   -0.414 -0.483  -0.973  1.59  -0.975
# i 5 more variables: vitamins <dbl>, shelf <dbl>, weight <dbl>, cups <dbl>,
#   rating <dbl>
```

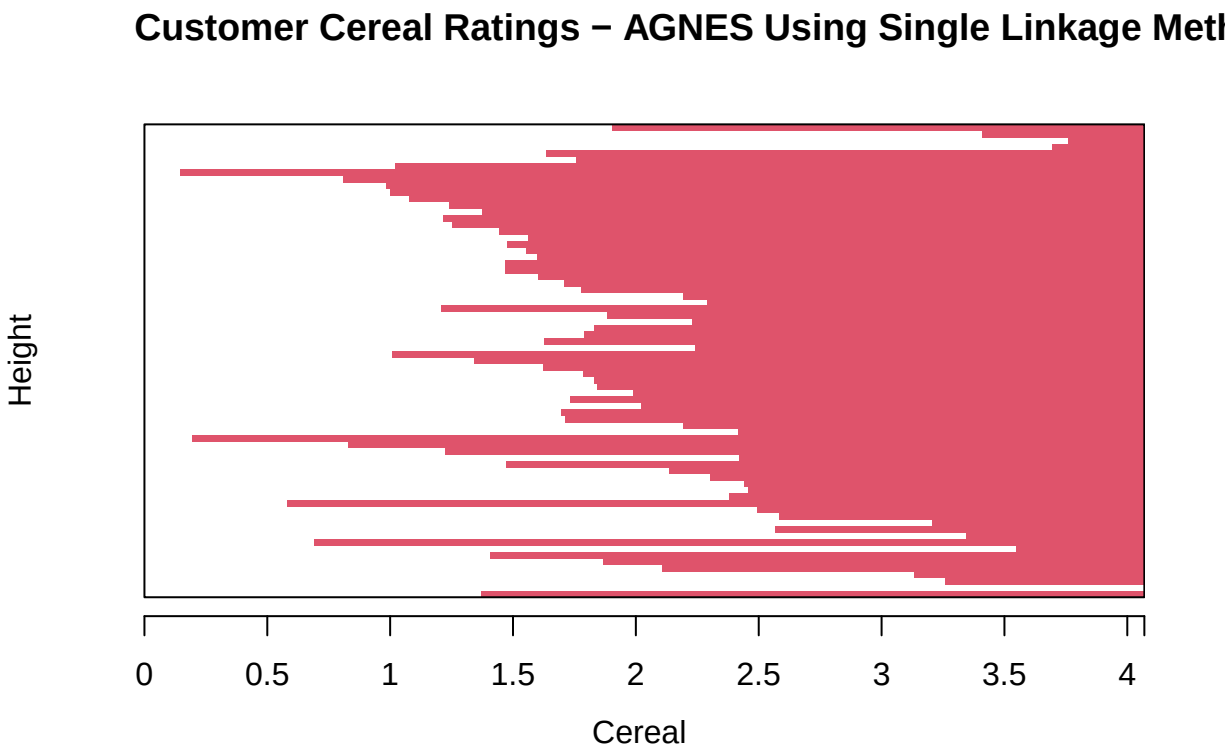
Following pre-processing and scaling, there were 74 observations overall as opposed to 77 before. There were just 3 records with the value "NA."

Question 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.

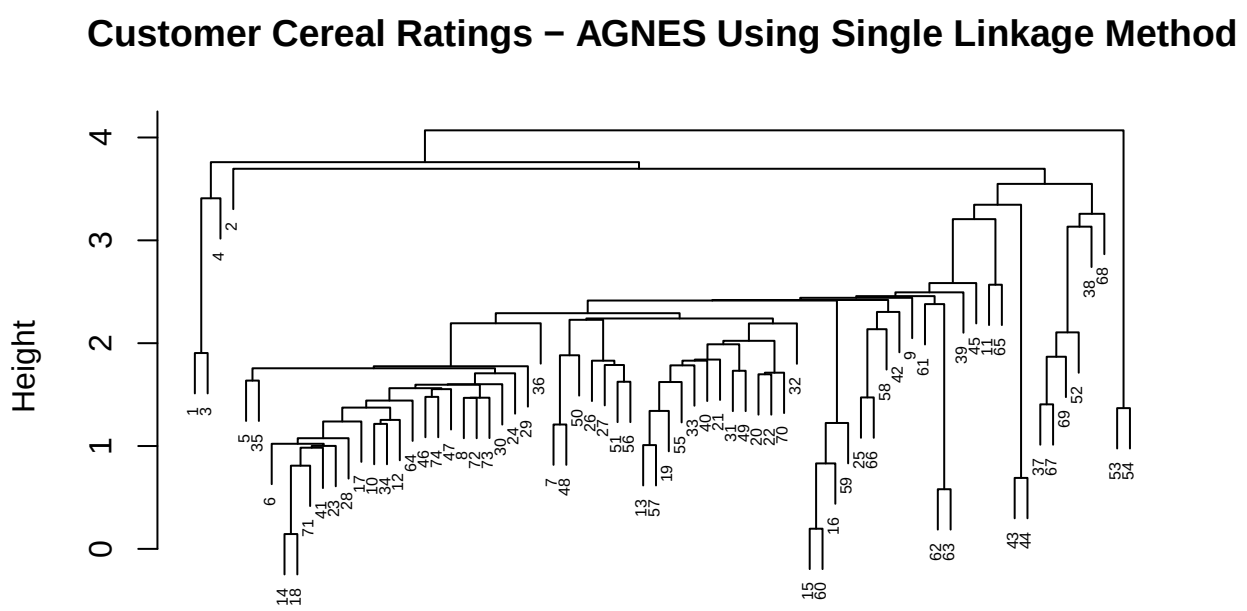
Solution

Single Linkage:

```r
# 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)
```

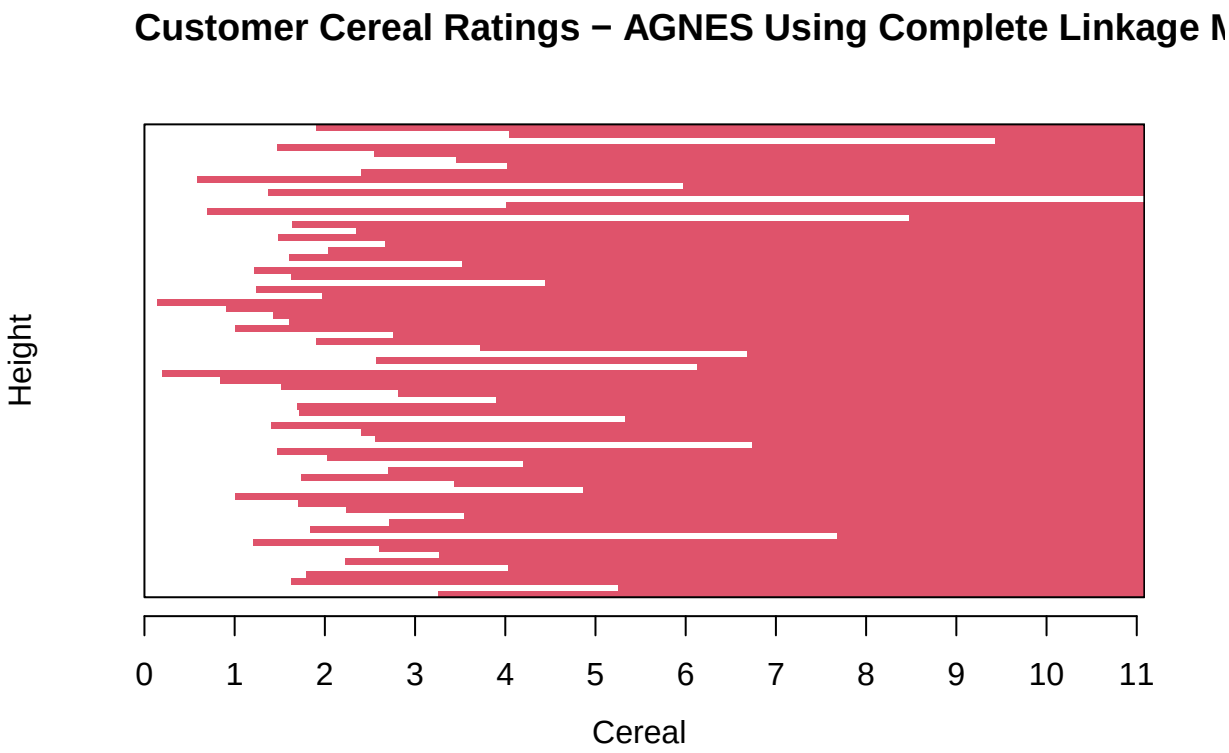


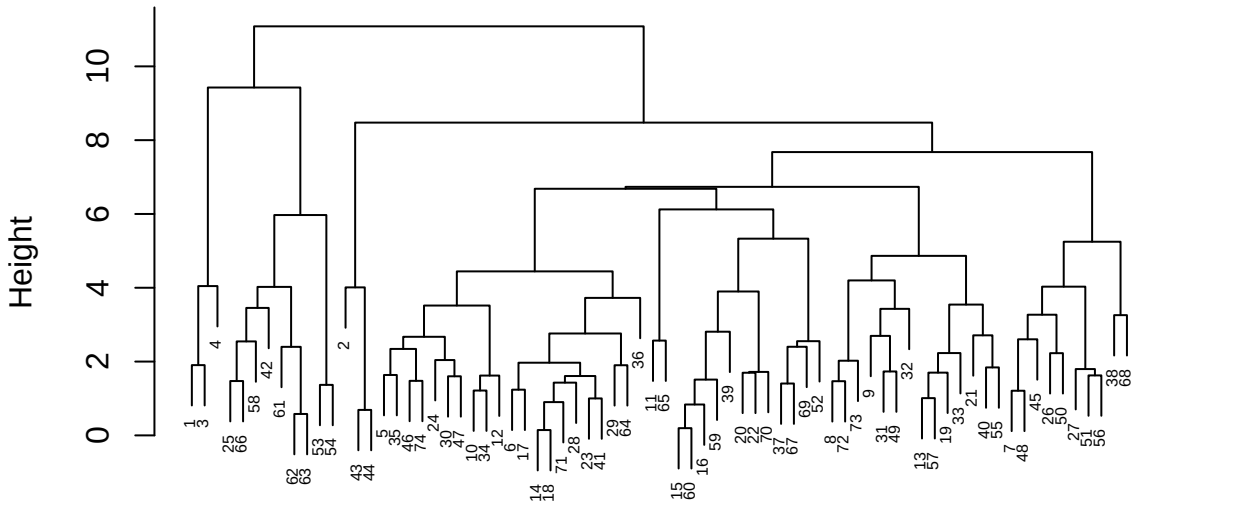
Agglomerative Coefficient = 0.61



Complete Linkage:

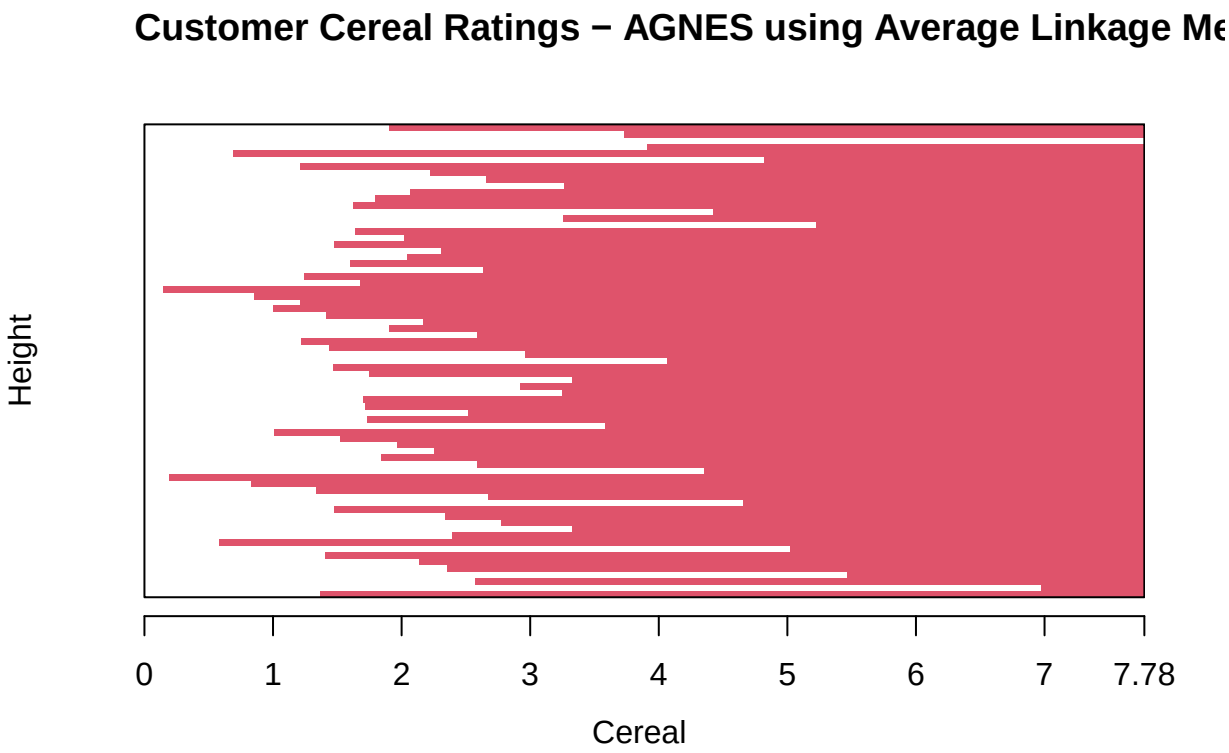
```
# Making use of the entire linkage approach to perform hierarchical clustering
HC_Complete <- agnes(Cereal_Euclidean, method = "complete")
# I'm plotting the outcomes of the various techniques here.
plot(HC_Complete,
main = "Customer Cereal Ratings - AGNES Using Complete Linkage Method",
xlab = "Cereal",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


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

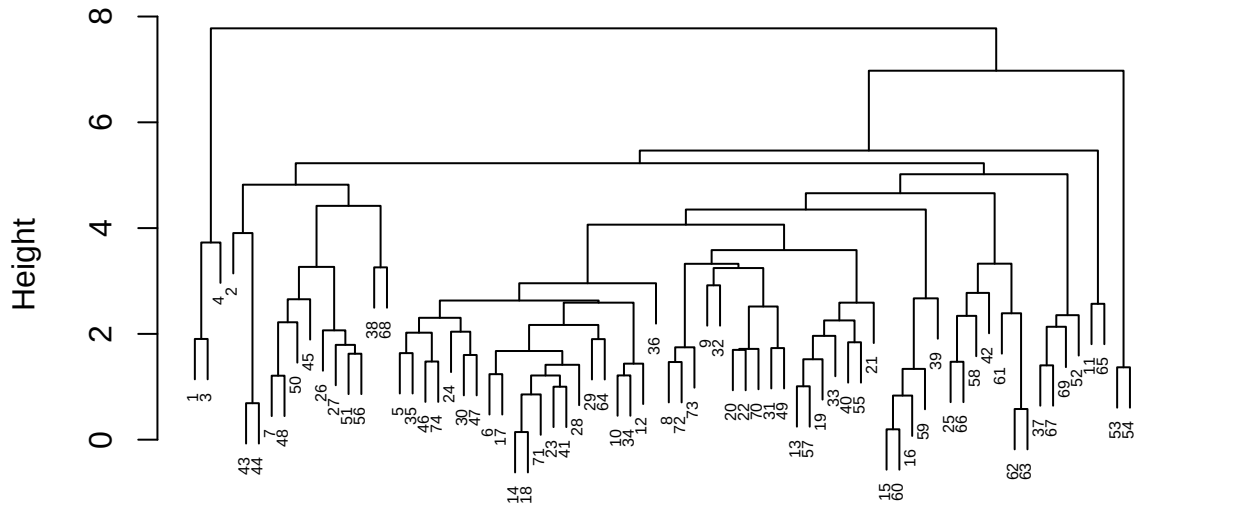


Average Linkage:

```
# 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)
```

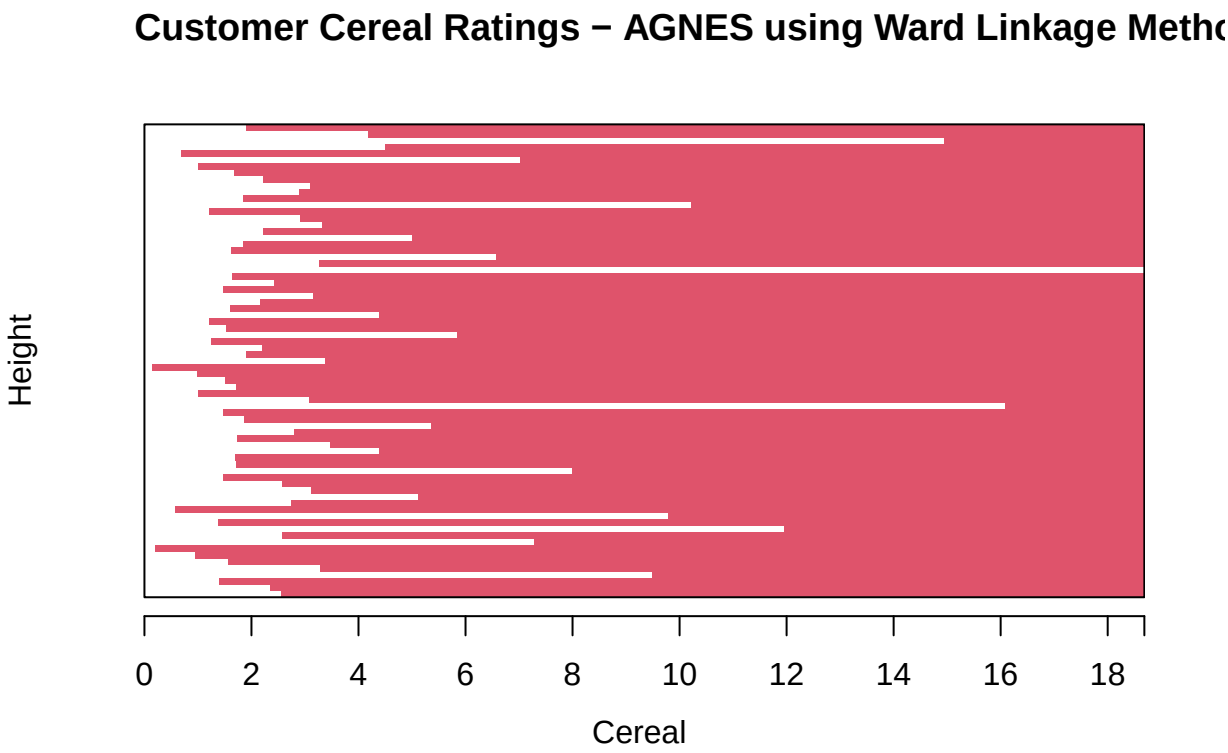


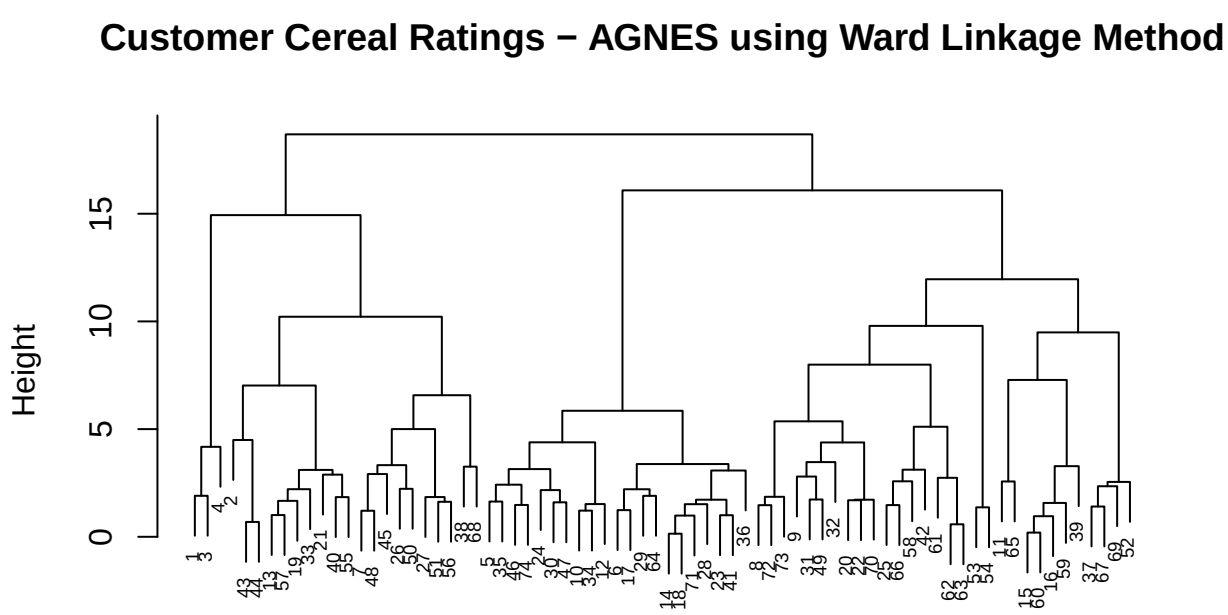
Agglomerative Coefficient = 0.78



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

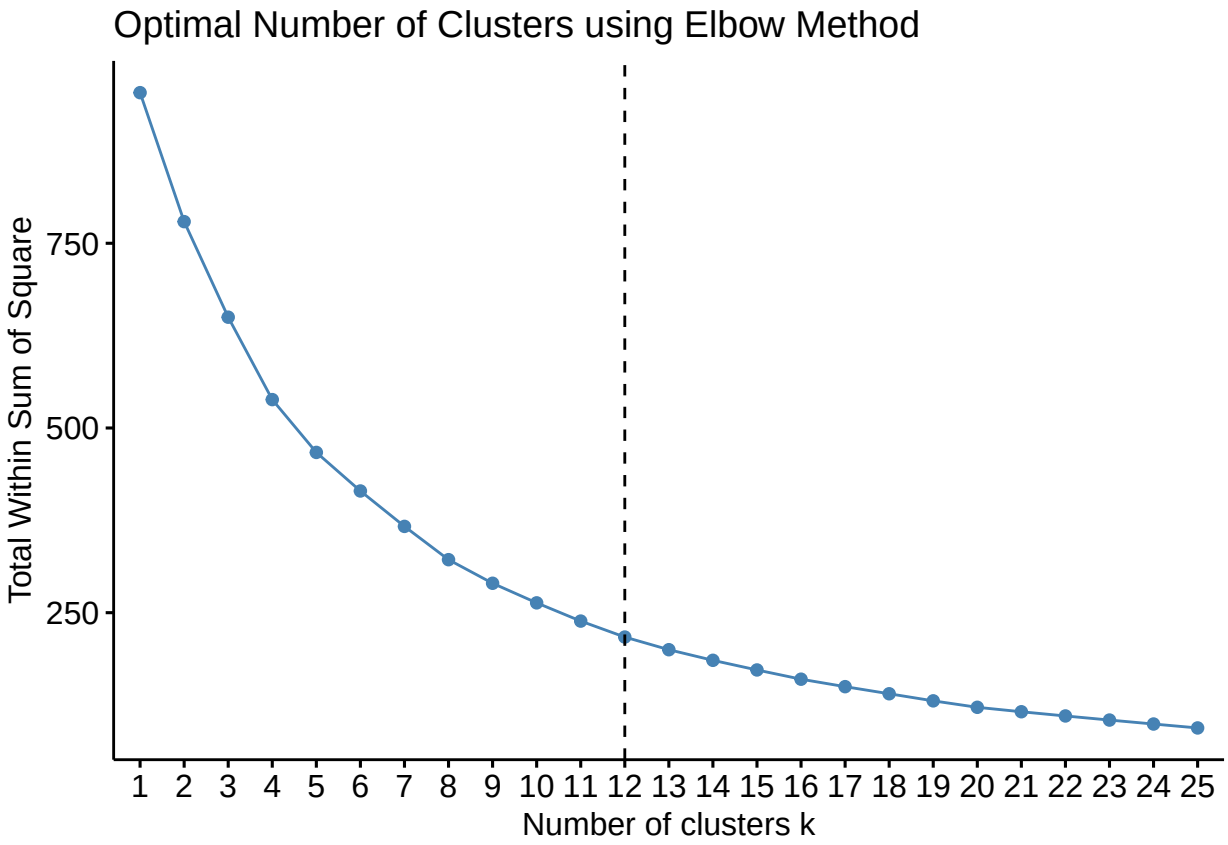


The closer the clustering structure, the closer the value is to 1.0. Consequently, the strategy that has the value closest to 1.0 will be chosen. Linkage Only: 0.61 Total Connection: 0.84 Linkage on average: 0.78 Ward Approach: 0.90 Here The Ward approach is the most effective clustering model based on the results.

Question How many clusters would you choose?

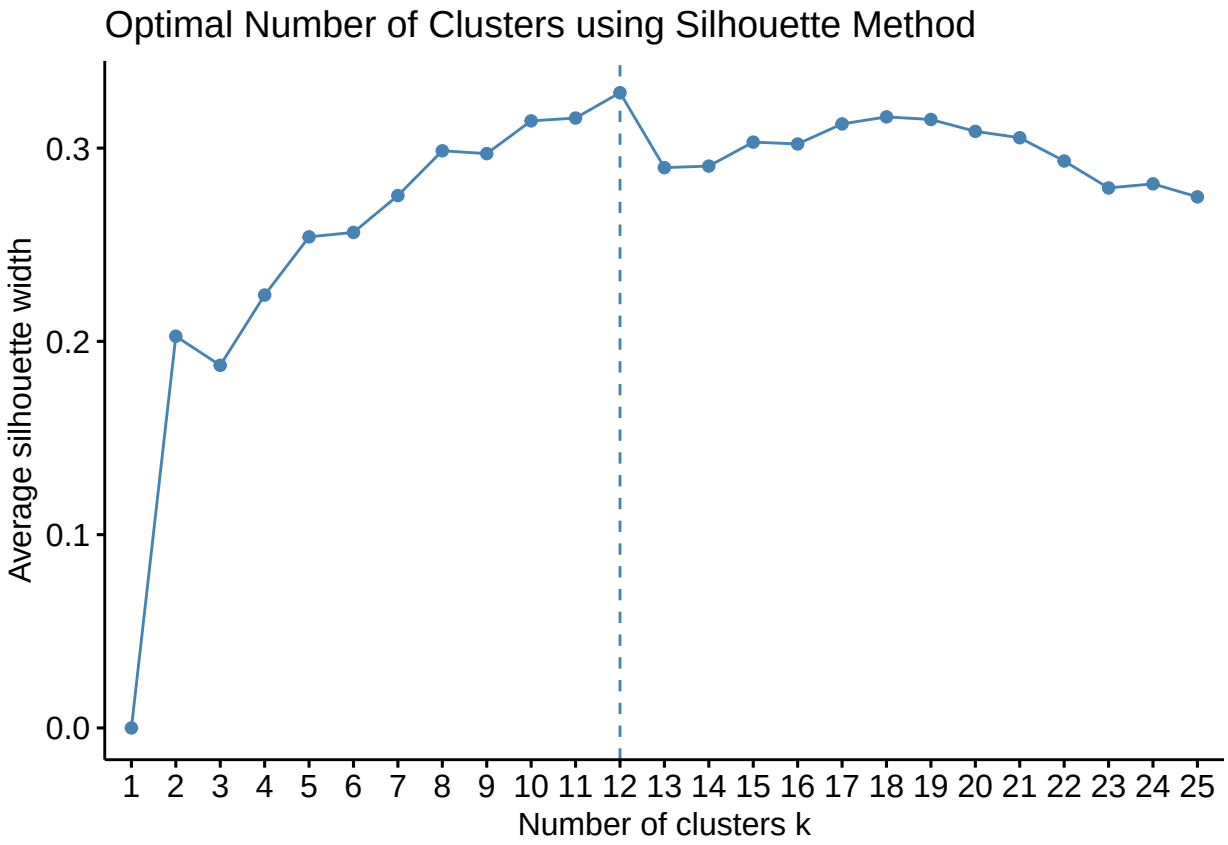
Here I am using elbow and silhouette methods to determine the appropriate number of clusters. Elbow Method:

```
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)
```



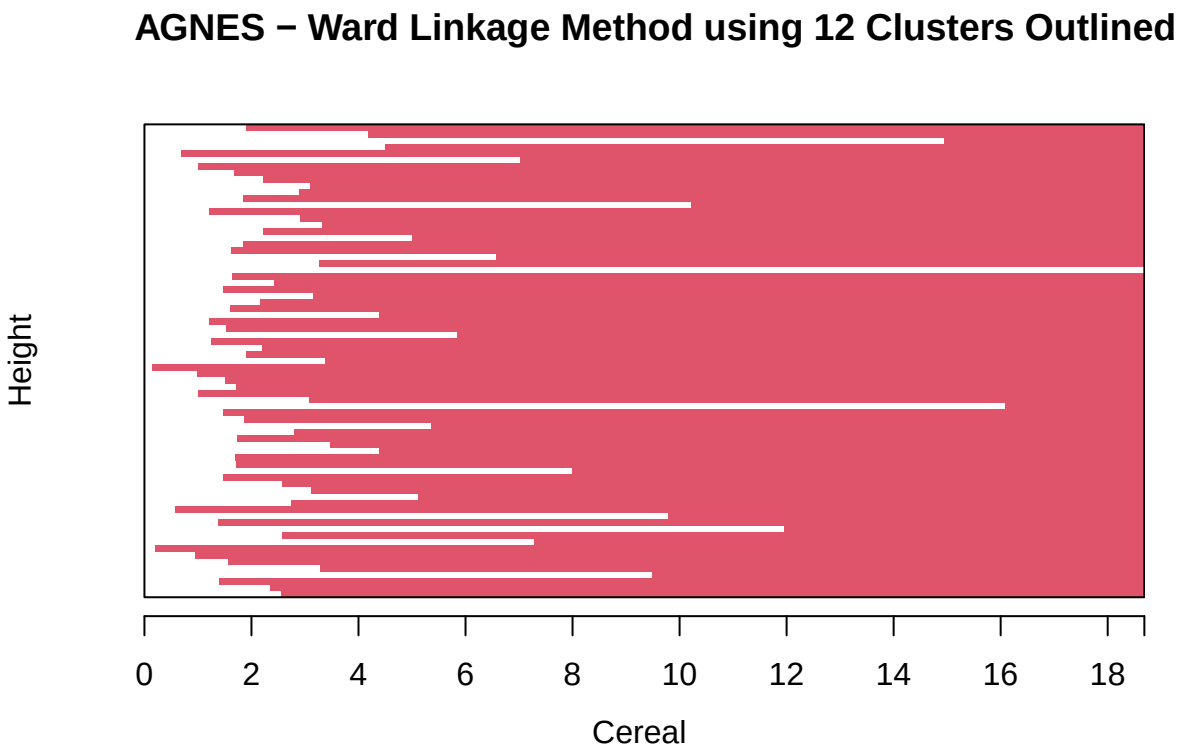
##Silhouette Method:

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

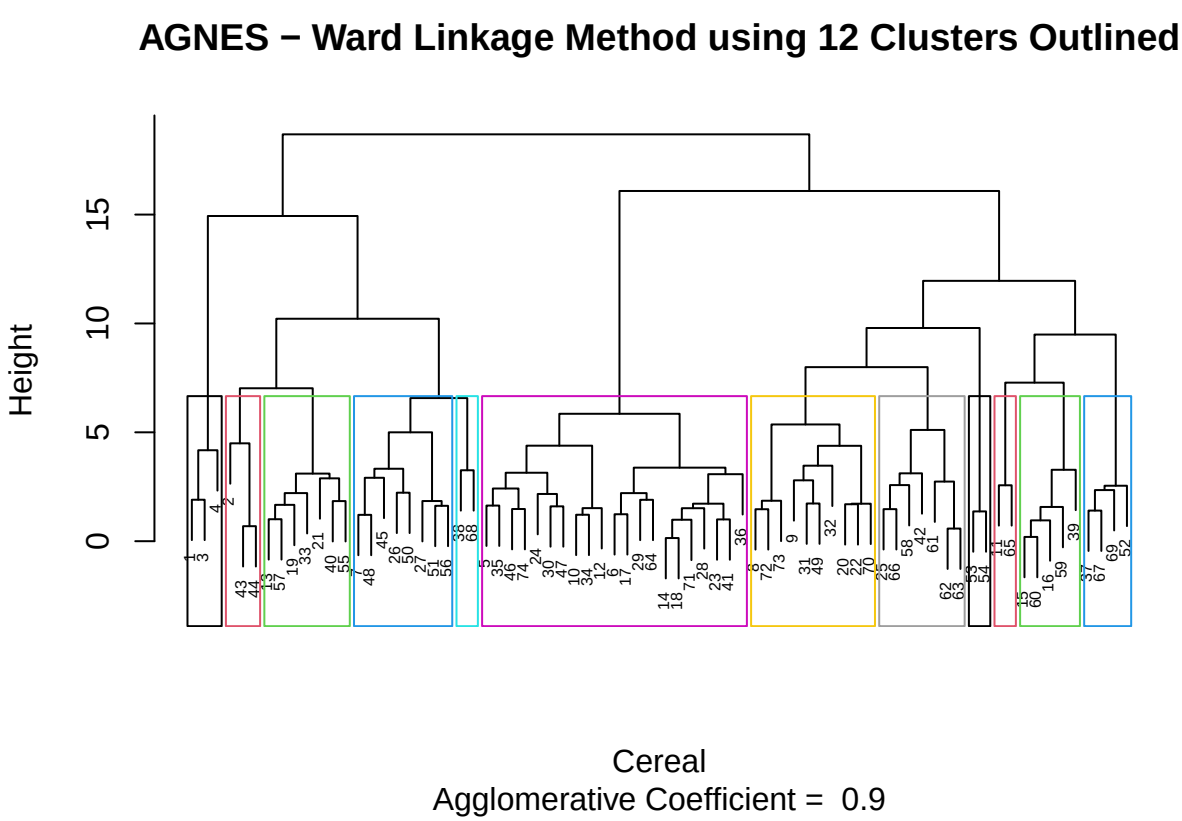


The findings of the elbow and silhouette approaches show that 12 clusters would be the ideal quantity.

```
#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)
```



Question 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?

In this case, standardizing the data would be improper because cereal's nutritional information varies according on the sample that is being studied. Thus, the cereals that could be included in the data collection were limited to those with a high sugar content and low levels of fiber, iron, or other nutritional information. Predicting the amount of nutrition a child will receive from cereal becomes challenging once it has been homogenized across the sample set. That means that a cereal with an iron level of 0.999 can be the best of the worst in the sample set and offer no nutritional value. A cereal with an iron content of 0.999 is likely to have almost all of its nutritional benefits. It is reasonable to presume that a cereal with an iron content of 0.999 will supply almost all of a child's nutritional iron needs. Making the data into a ratio of a child's daily recommended amounts of calories, fiber, carbohydrates, and other nutrients would be a better way to preprocess the data. By doing this, analysts would be able to make more informed cluster judgments during the review phase because a limited number of crucial criteria would not be able to override distance estimates. When analyzing the clusters, an analyst can look at the cluster average to find out what proportion of a student's daily nutritional needs XX cereal would satisfy. Employees would be able to choose "healthy" cereal clusters with knowledge thanks to this.