Assignment 5

Nithin varma

2022-12-01

```
 Cerealsdata <- read.csv("C:/Users/Kittu Varma/Downloads/Cereals.csv")
```

```
library(fastDummies)
```

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

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(car)
```

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

## Loading required package: carData

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

```
library(tidyverse)
```

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

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

## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.4

## Warning: package 'forcats' was built under R version 4.2.2

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

```
library(cluster)
library(stats)
library(factoextra)
```

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

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

```
library(cluster)
library(hrbrthemes)
```

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

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
library(GGally)
```

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

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(viridis)
```

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

```
## Loading required package: viridisLite
```

```
library(fpc)
```

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

```
library(caTools)
```

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

#DataPreprocessing

#Removing all cereals with missing values

```
Cerealsdata<-na.omit(Cerealsdata)
```

#Confirming that each record is unique

```
record<- as.data.frame(table(Cerealsdata[1]))
```

#Adding row names

```
row.names(Cerealsdata)<- Cerealsdata[,1]
```

#removing the name coloums

```
Cerealsdata<-Cerealsdata[,-1]
```

#reviewing data structure

```
str(Cerealsdata)
```

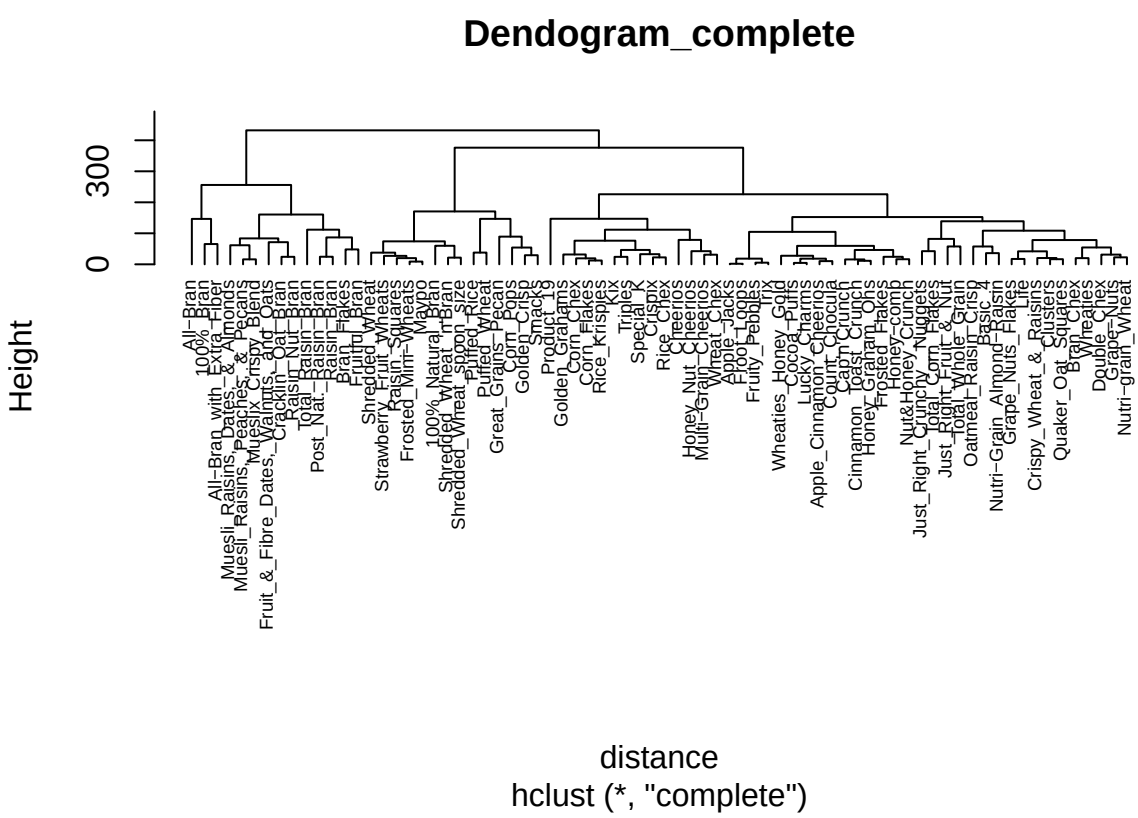
```
## 'data.frame':    74 obs. of  15 variables:
##  $ mfr     : chr  "N" "Q" "K" "K" ...
##  $ type    : chr  "C" "C" "C" "C" ...
##  $ calories: int  70 120 70 50 110 110 130 90 90 120 ...
##  $ protein : int  4 3 4 4 2 2 3 2 3 1 ...
##  $ fat     : int  1 5 1 0 2 0 2 1 0 2 ...
##  $ sodium  : int  130 15 260 140 180 125 210 200 210 220 ...
##  $ fiber   : num  10 2 9 14 1.5 1 2 4 5 0 ...
##  $ carbo   : num  5 8 7 8 10.5 11 18 15 13 12 ...
##  $ sugars  : int  6 8 5 0 10 14 8 6 5 12 ...
##  $ potass  : int  280 135 320 330 70 30 100 125 190 35 ...
##  $ vitamins: int  25 0 25 25 25 25 25 25 25 25 ...
##  $ shelf   : int  3 3 3 3 1 2 3 1 3 2 ...
##  $ weight  : num  1 1 1 1 1 1 1.33 1 1 1 ...
##  $ cups    : num  0.33 1 0.33 0.5 0.75 1 0.75 0.67 0.67 0.75 ...
##  $ rating  : num  68.4 34 59.4 93.7 29.5 ...
```

#Apply hierarchical clustering to the cereals data using Euclidean distance to the normalized measurements.

```
distance <- dist(Cerealsdata, method = "euclidean")
```

```
## Warning in dist(Cerealsdata, method = "euclidean"): NAs introduced by coercion
```

```
# Hierarchical clustering using Complete Linkage
hc1 <- hclust(distance, method = "complete")
# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1, main = "Dendogram_complete")
```



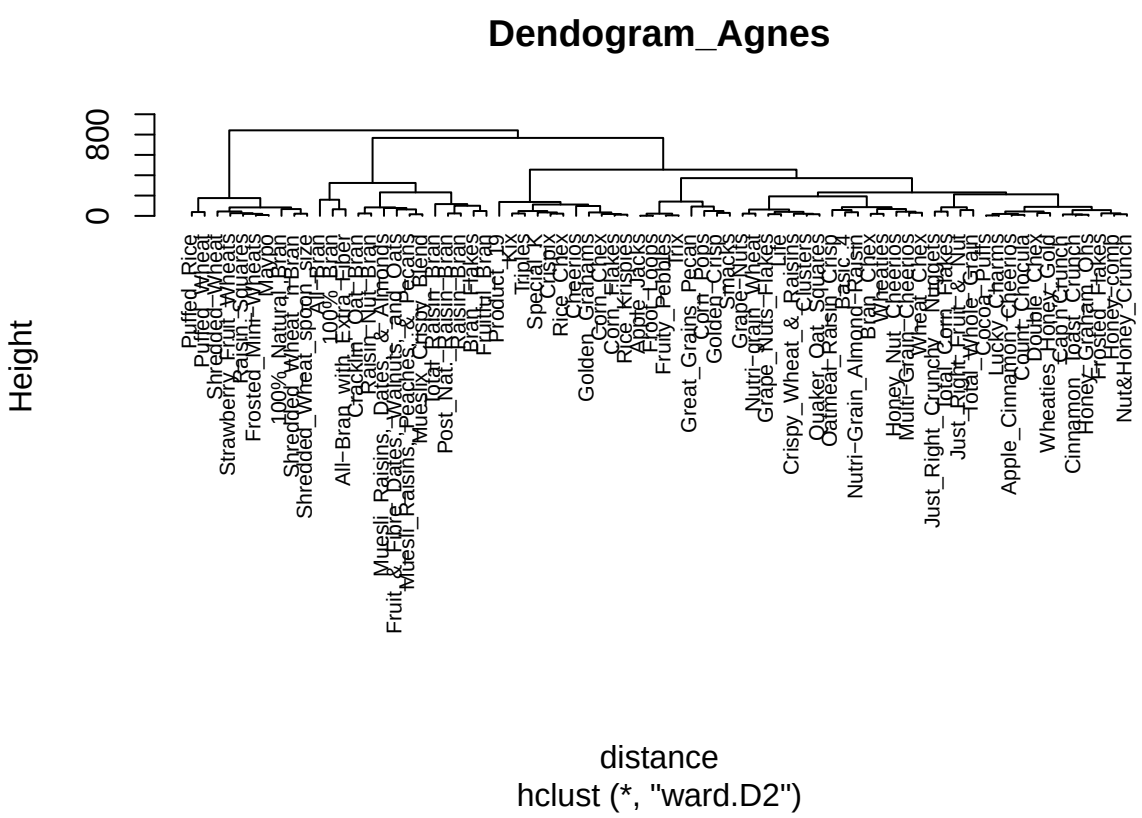
#Use Agnes to compare the clustering from single linkage, complete linkage, average linkage, and Ward. Choose the best method.

```
# vector of methods to compare
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")
# function to compute coefficient
ac <- function(x) {
agnes(Cerealsdata, method = x)$ac
}
map_dbl(m, ac)
```

```
##   average    single  complete      ward
## 0.8786692 0.7297141 0.9225732 0.9595040
```

#ward linkage has the strongest clustering structure

```
hc3<-hclust(distance,method = "ward.D2")
plot(hc3, cex = 0.7, hang = -1, main = "Dendogram_Agnes")
```



#differnece between Hieracial clustering and K means is that: K-Means is that it needs us to pre-enter the number of clusters (K) but Hierarchical clustering has no such requirements to do so.

#How many clusters would you choose? #based on data exploration we have 7 clusters that appear to be common among the paired variables #Cut them into 7 clusters

```
clusters <- cutree(hc3, k = 7)
#number of cereals in each cluster
table(clusters)
```

```
## clusters
##  1  2  3  4  5  6  7
##  3 10 31  8  5 11  6
```

#cluster data with k=4

```
clusters1 <- cutree(hc3, k = 4)
#number of cereals in each cluster
table(clusters1)
```

```
## clusters1
##  1  2  3  4
## 14 10 39 11
```

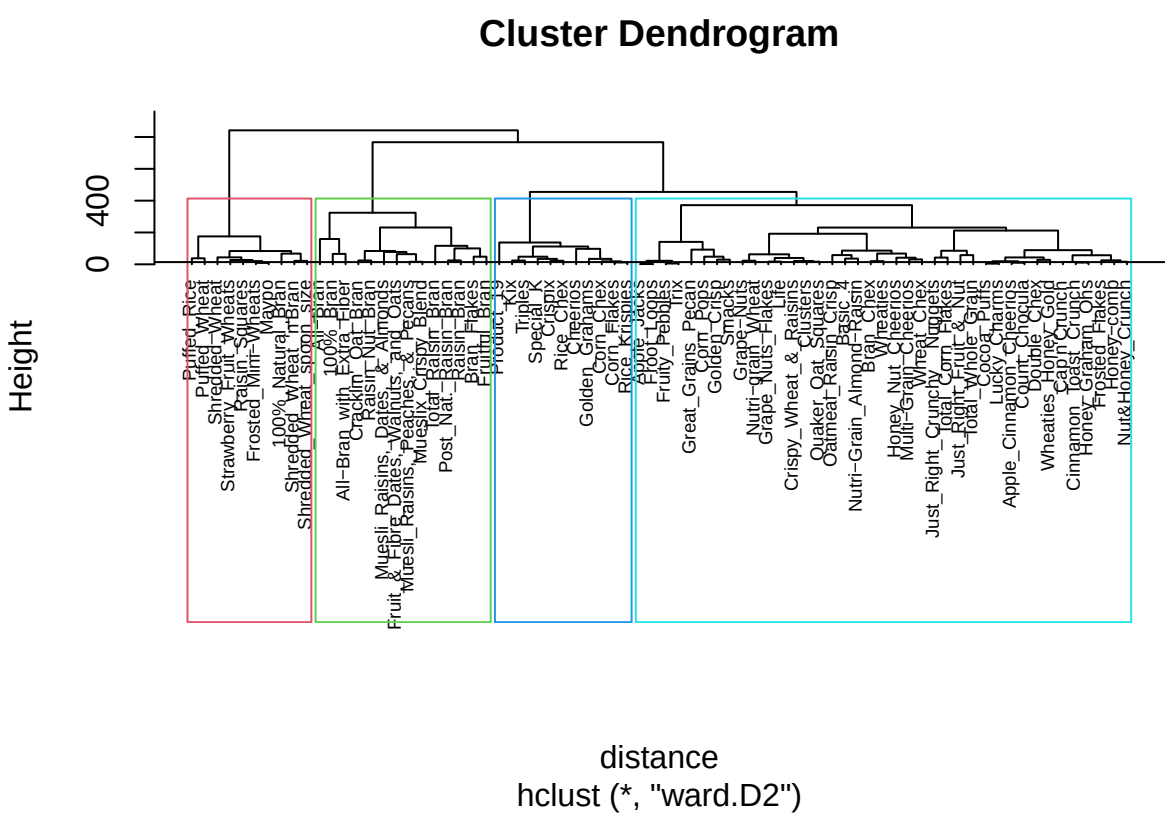
```r
#cluster centroids
centroids_cereals <- aggregate(Cerealsdata, by=list(cluster=clusters1), mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```r
centroids_cereals
```

```
##   cluster mfr type calories  protein       fat   sodium     fiber    carbo
## 1       1  NA   NA 112.1429 3.357143 1.4285714 172.5000 5.6071429 12.03571
## 2       2  NA   NA  86.0000 2.500000 0.6000000   3.0000 2.1000000 14.60000
## 3       3  NA   NA 110.2564 2.179487 1.1025641 169.7436 1.4230769 14.26923
## 4       4  NA   NA 108.1818 2.636364 0.4545455 268.1818 0.5454545 19.90909
##     sugars    potass vitamins    shelf   weight      cups   rating
## 1 9.142857 217.14286 30.35714 2.928571 1.160000 0.6614286 46.01532
## 2 2.900000  95.00000 10.00000 2.100000 0.883000 0.8640000 60.11492
## 3 8.564103  71.66667 32.69231 2.128205 1.031026 0.8082051 36.55985
## 4 3.181818  45.90909 31.81818 1.727273 1.000000 1.0345455 42.21039
```

```r
#putting the data all together to identify which cluster each cereal belongs to
cereal.cluster <- cbind(clusters1, Cerealsdata)
#plot cluster
plot(hc3, cex= 0.6, hang = -1)
#Plot clusters with borders
rect.hclust(hc3, k = 4, border = 2:7)
abline(h = 14, col = "black")
```



#cluster centroids

```
centroids_cerealsdata <- aggregate(Cerealsdata, by=list(cluster=clusters1), mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
```

```
## returning NA
```

```
centroids_cerealsdata
```

```
##   cluster mfr type calories  protein       fat   sodium     fiber    carbo
## 1       1  NA   NA 112.1429 3.357143 1.4285714 172.5000 5.6071429 12.03571
## 2       2  NA   NA  86.0000 2.500000 0.6000000   3.0000 2.1000000 14.60000
## 3       3  NA   NA 110.2564 2.179487 1.1025641 169.7436 1.4230769 14.26923
## 4       4  NA   NA 108.1818 2.636364 0.4545455 268.1818 0.5454545 19.90909
##     sugars    potass vitamins    shelf   weight      cups   rating
## 1 9.142857 217.14286 30.35714 2.928571 1.160000 0.6614286 46.01532
## 2 2.900000  95.00000 10.00000 2.100000 0.883000 0.8640000 60.11492
## 3 8.564103  71.66667 32.69231 2.128205 1.031026 0.8082051 36.55985
## 4 3.181818  45.90909 31.81818 1.727273 1.000000 1.0345455 42.21039
```

```
# partition data into A and B - 50% (data has 74rows)
set.seed(123)
A<-Cerealsdata[1:37,]
B<-Cerealsdata[38:74,]
```

#clustering partition A # Apply hierarchical clustering using Euclidean distance

```
distanceA <- dist(A, method = "euclidean")
```

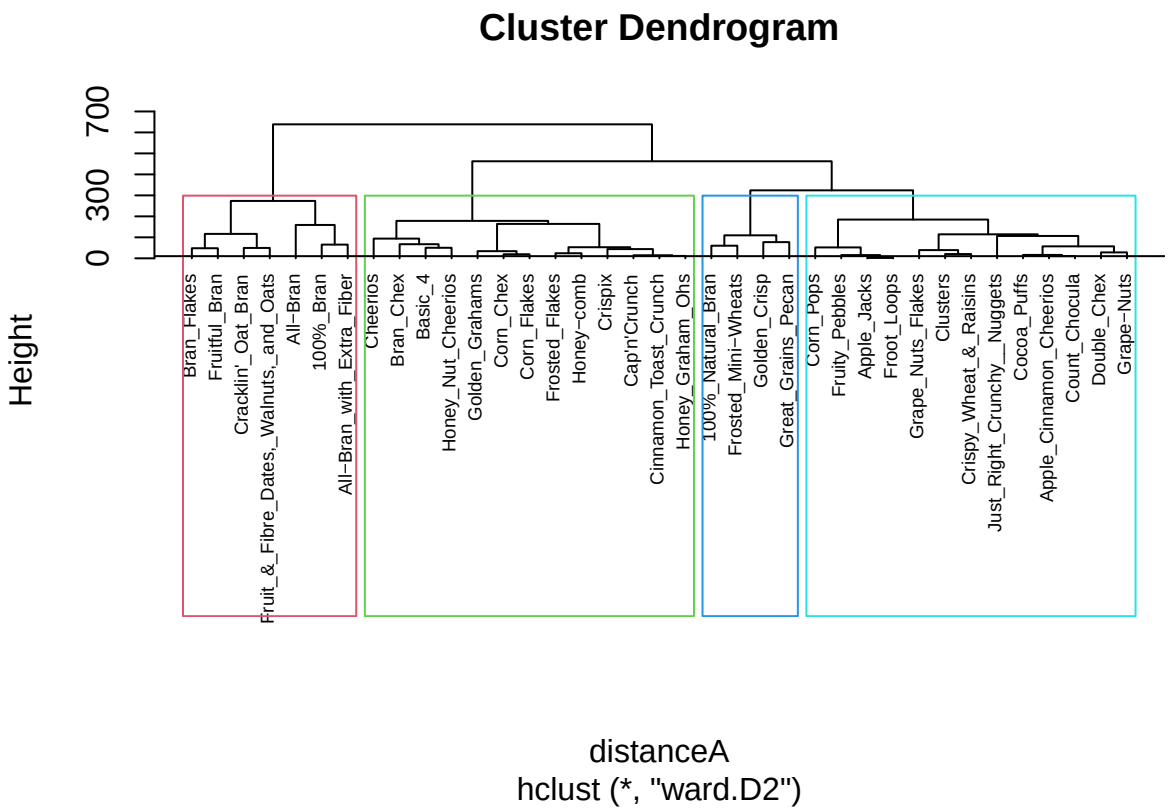
```
## Warning in dist(A, method = "euclidean"): NAs introduced by coercion
```

```
#Hierarchical clustering using Ward (we had determined that ward had the strongest clustering structure
hc_A <- hclust(distanceA, method = "ward.D2")
# Cut tree into 4 groups(we had determined that optimal k =4)
clust_A <- cutree(hc_A, k = 4)
# Number of members in each cluster
table(clust_A)
```

```
## clust_A
##  1  2  3  4
##  7  4 13 13
```

#putting all the data together to identify which cluster each cereal belongs to

```
clust_A1 <- cbind(clust_A, A)
plot(hc_A, cex= 0.6, hang = -1)
#Plot clusters with borders
rect.hclust(hc_A, k = 4, border = 2:7)
abline(h = 10.3, col = "black")
```



calculating centroids of partition A

```
A<-as.data.frame(A)
centroids_A <- aggregate(A, by=list(cluster=clust_A), mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

#assign each record in partition B to the cluster with the closest centroid).

```
Assign<-data.frame(observations=seq(1,37,1),cluster=rep(0,37))
for(i in 0:37)
{
x<-as.data.frame(rbind(centroids_A[,-1],B[i,]))
y<-as.matrix(get_dist(x))
Assign[i,2]<-which.min(y[4,-4])
}
```

```
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```

```
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```

```
rownames(Assign) <-rownames(B)
table(Assign)
```

```
##              cluster
## observations 3 4
##           1  1 0
##           2  0 1
##           3  1 0
##           4  0 1
##           5  1 0
##           6  1 0
##           7  1 0
##           8  1 0
##           9  0 1
##           10 0 1
##           11 0 1
##           12 0 1
```

```
##           13 1 0
##           14 1 0
##           15 1 0
##           16 1 0
##           17 1 0
##           18 1 0
##           19 1 0
##           20 1 0
##           21 1 0
##           22 0 1
##           23 0 1
##           24 1 0
##           25 1 0
##           26 1 0
##           27 1 0
##           28 0 1
##           29 1 0
##           30 1 0
##           31 1 0
##           32 1 0
##           33 0 1
##           34 1 0
##           35 0 1
##           36 0 1
##           37 0 1
```

```
cbind(cluster_partition=Assign$cluster,cluster_complete=cereal.cluster[38:74,1])
```

```
##       cluster_partition cluster_complete
##  [1,]                 3                3
##  [2,]                 4                4
##  [3,]                 3                3
##  [4,]                 4                3
##  [5,]                 3                2
##  [6,]                 3                1
##  [7,]                 3                1
##  [8,]                 3                1
##  [9,]                 4                3
## [10,]                 4                3
## [11,]                 4                3
## [12,]                 4                3
## [13,]                 3                3
## [14,]                 3                1
## [15,]                 3                4
## [16,]                 3                2
## [17,]                 3                2
## [18,]                 3                3
## [19,]                 3                1
## [20,]                 3                1
## [21,]                 3                2
## [22,]                 4                4
## [23,]                 4                4
## [24,]                 3                2
## [25,]                 3                2
```

```
## [26,]                3               2
## [27,]                3               3
## [28,]                4               4
## [29,]                3               2
## [30,]                3               3
## [31,]                3               1
## [32,]                3               3
## [33,]                4               4
## [34,]                3               3
## [35,]                4               3
## [36,]                4               3
## [37,]                4               3
```

```
table(Assign$cluster==cereal.cluster[38:74,1])
```

```
##
## FALSE  TRUE
##    24    13
```

#59% of data was assigned to the same cluster with both partitioned data and complete data.this representsthe stability of the cluster, which is not too high. however, centroids were used for assigning clusters in partitioned dataset whereas in the complete dataset, total figures were used.

*** Cluster stability assessment ***

Cluster method: hclust/cutree

Full clustering results are given as parameter result

of the clusterboot object, which also provides further statistics

of the resampling results.

Number of resampling runs: 100

Number of clusters found in data: 4

```
colnames(Cerealsdata)
```

```
##  [1] "mfr"      "type"     "calories" "protein"  "fat"      "sodium"
##  [7] "fiber"    "carbo"    "sugars"   "potass"   "vitamins" "shelf"
## [13] "weight"   "cups"     "rating"
```

#Healthy variables are Protein; Fiber; Potass; Vitamins. #Unhealthy variables are Sugars; calories;

```
centroids_cereals
```

```
##   cluster mfr type calories  protein       fat   sodium     fiber    carbo
## 1       1  NA   NA 112.1429 3.357143 1.4285714 172.5000 5.6071429 12.03571
## 2       2  NA   NA  86.0000 2.500000 0.6000000   3.0000 2.1000000 14.60000
## 3       3  NA   NA 110.2564 2.179487 1.1025641 169.7436 1.4230769 14.26923
## 4       4  NA   NA 108.1818 2.636364 0.4545455 268.1818 0.5454545 19.90909
##     sugars    potass vitamins    shelf   weight      cups   rating
## 1 9.142857 217.14286 30.35714 2.928571 1.160000 0.6614286 46.01532
## 2 2.900000  95.00000 10.00000 2.100000 0.883000 0.8640000 60.11492
## 3 8.564103  71.66667 32.69231 2.128205 1.031026 0.8082051 36.55985
## 4 3.181818  45.90909 31.81818 1.727273 1.000000 1.0345455 42.21039
```

#cluster 1 is highest in Protein; fiber; Potass and lowest in Sugar and calories. Cluster 1 has the healthy cereals.

```
summary(Cerealsdata)
```

```
##      mfr                type              calories      protein
##  Length:74          Length:74          Min.   : 50   Min.   :1.000
##  Class :character   Class :character   1st Qu.:100   1st Qu.:2.000
##  Mode  :character   Mode  :character   Median :110   Median :2.500
##                                        Mean   :107   Mean   :2.514
##                                        3rd Qu.:110   3rd Qu.:3.000
##                                        Max.   :160   Max.   :6.000
##       fat         sodium          fiber            carbo           sugars
##  Min.   :0   Min.   :  0.0   Min.   : 0.000   Min.   : 5.00   Min.   : 0.000
##  1st Qu.:0   1st Qu.:135.0   1st Qu.: 0.250   1st Qu.:12.00   1st Qu.: 3.000
##  Median :1   Median :180.0   Median : 2.000   Median :14.50   Median : 7.000
##  Mean   :1   Mean   :162.4   Mean   : 2.176   Mean   :14.73   Mean   : 7.108
##  3rd Qu.:1   3rd Qu.:217.5   3rd Qu.: 3.000   3rd Qu.:17.00   3rd Qu.:11.000
##  Max.   :5   Max.   :320.0   Max.   :14.000   Max.   :23.00   Max.   :15.000
##      potass          vitamins          shelf           weight
##  Min.   : 15.00   Min.   :  0.00   Min.   :1.000   Min.   :0.500
##  1st Qu.: 41.25   1st Qu.: 25.00   1st Qu.:1.250   1st Qu.:1.000
##  Median : 90.00   Median : 25.00   Median :2.000   Median :1.000
##  Mean   : 98.51   Mean   : 29.05   Mean   :2.216   Mean   :1.031
##  3rd Qu.:120.00   3rd Qu.: 25.00   3rd Qu.:3.000   3rd Qu.:1.000
##  Max.   :330.00   Max.   :100.00   Max.   :3.000   Max.   :1.500
##       cups            rating
##  Min.   :0.2500   Min.   :18.04
##  1st Qu.:0.6700   1st Qu.:32.45
##  Median :0.7500   Median :40.25
##  Mean   :0.8216   Mean   :42.37
##  3rd Qu.:1.0000   3rd Qu.:50.52
##  Max.   :1.5000   Max.   :93.70
```