session_22_segment___clusterings.R

Seshan

Mon Aug 20 14:22:42 2018

2. Perform the below given activities:
a. apply K-means clustering to identify similar recipes
b. apply K-means clustering to identify similar attributes
c. how many unique recipes that people order often
d. what are their typical profiles

```
setwd("C:/Users/Seshan/Desktop/sv R related/acadgild/assignments/session 22")

library(readr)

epi_r <- read.csv("epi_r.csv")

View(epi_r)

df<-epi_r

df[df==""] <- NA

df1<-na.exclude(df)

View(df1)

str(df1)

library(factoextra)

library("factoextra")

df <- df1[1:1000, 1:6]

na.exclude(df)

View(df)

head(df[, 1:6])

# Prepare Data

df <- na.omit(df) # listwise deletion of missing

#df <- scale(df) # standardize variables
```

```
View(df)

set.seed(1234)

ind = sample(1:nrow(df),0.8*nrow(df),replace = F)

df_train =df[ind,-1]

df_test = df[-ind,-1]

summary(df)

dim(df)

# outlier definition

# x > Q3+1.5*IQR - positive side outlier

# x < Q1-1.5*IQR - negative or lower side outlier

par(mfrow=c(2,3))

(boxplot(df1$rating)$out);(boxplot(df1$calories)$out);(boxplot(df1$protein)$out);(boxplot(df1$fat)$out);(boxplot(df1$sodium)$out)

apply(df,2,range)

apply(df,2,summary)

# KMeans - comes from Rcmdr library

# Kmeans- from amap library

# kmeans- from stats library

# steps in k-means clustering

#1- preprocessing the data (impute missing values, remove outliers, feature trasnformation)

#2- scaling or standardization of data set

#3- decide the number of clusters (value of K)

#4- iterate over the samples to create clusters

#5- decide the distance measure

#6- calculate the group accuracy

# scaling of data

df_train1 <- scale(df_train)

head(df_train1)
```

```
class(df_train1)
# screeplot approach to decide the number of clusters
km = kmeans(df_train1,1)
km$withinss
km$tot.withinss
km = kmeans(df_train1,2)
km$withinss
km$tot.withinss
km = kmeans(df_train1,3)
km$withinss
km$tot.withinss
km = kmeans(df_train1,4)
km$withinss
km$tot.withinss
km = kmeans(df_train1,5)
km$withinss
km$tot.withinss
km = kmeans(df_train1,6)
km$withinss
km$tot.withinss
km = kmeans(df_train1,7)
km$withinss
km$tot.withinss
km = kmeans(df_train1,8)
km$withinss
km$tot.withinss
km = kmeans(df_train1,9)
km$withinss
```

```
km$tot.withinss

km = kmeans(df_train1,10)

km$withinss

km$tot.withinss

dev.off()

sumsq=NULL

for (i in 1:25)

  sumsq[i] = sum(kmeans(df_train,centers=i, iter.max = 1000, nstart=i,
algorithm='Forgy')$withinss)

plot(1:25,sumsq,type='b', main='Screeplot showing within group sum of squares')

km = kmeans(df_train1,3)

km$withinss

km$tot.withinss

class(km$cluster)

summary(km)

km$centers

as.numeric(km$cluster)

length(km$cluster)

dim(df_train)

class(df_train)

df_train$cl <- km$cluster

head(df_train)

# profiles of clusters

aggregate(df_train[,1:5],list(df_train[,6]),mean)

table(df1$rating)

table(df1$calories)

table(df1$X22.minute.meals)

table(df1$sodium)
```

```
library(cluster)
clusplot(df_train,df_train$cl,cex=0.9,color=T,shade=T, labels=4,lines=0)
#HC clustering or Hierarchical Clustering
# distance (euclidean, manhattan, cosine distance)
# Divisive method (top down)
# Agglomorative method (bottom up)
df_train = df_train[,-5]
head(df_train)
str(df_train)
# compute the distance metrix
d1 <- dist(df_train,method='euclidean')
summary(d1)
# HC
fit <- hclust(d1,method = 'ward.D2')
plot(fit)
# single, double, average, ward, ward.D2
# agglomorative method
fit <- agnes(d1,metric='euclidean',method = 'ward')
plot(fit)
# divisive method
fit <- diana(d1,metric='euclidean')
plot(fit)
```

```
setwd("C:/Users/Seshan/Desktop/sv R related/acadgild/assignments/session 22")
library(readr)
epi_r <- read.csv("epi_r.csv")
#setwd("C:/Users/Seshan/Desktop/sv R related/acadgild/assignments/session 22"
)
```

```r
#library(readr)
#epi_r <- read.csv("C:/Users/Seshan/Desktop/sv R related/acadgild/assignments
/session22/epi_r.csv",header=T, na.strings=c("","NA"))
View(epi_r)
df<-epi_r
df[df==""] <- NA
df1<-na.exclude(df)
View(df1)


str(df1)
```

```
## 'data.frame':    15864 obs. of  680 variables:
##  $ title                   : Factor w/ 17736 levels "'Wichcraft's Roasted
Turkey, Avocado, Bacon, Onion Relish, & AÃ¯oli on Ciabatta ",..: 8782 1738 11
861 15252 16218 8349 7499 17591 1005 1270 ...
##  $ rating                  : num  2.5 4.38 3.75 3.12 4.38 ...
##  $ calories                : num  426 403 165 547 948 170 602 256 766 174
...
##  $ protein                 : num  30 18 6 20 19 7 23 4 12 11 ...
##  $ fat                     : num  7 23 7 32 79 10 41 5 48 12 ...
##  $ sodium                  : num  559 1439 165 452 1042 ...
##  $ X.cakeweek              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X.wasteless             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X22.minute.meals        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X3.ingredient.recipes   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X30.days.of.groceries   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ advance.prep.required   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ alabama                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ alaska                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ alcoholic               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ almond                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ amaretto                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ anchovy                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ anise                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ anniversary             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ anthony.bourdain        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ aperitif                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ appetizer               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ apple                   : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ apple.juice             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ apricot                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ arizona                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ artichoke               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ arugula                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ asian.pear              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ asparagus               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ aspen                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ atlanta                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ australia               : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ avocado                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ back.to.school         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ backyard.bbq           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bacon                  : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ bake                   : num  0 1 0 1 0 0 0 0 1 0 ...
##  $ banana                 : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ barley                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ basil                  : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ bass                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bastille.day           : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ bean                   : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ beef                   : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ beef.rib               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ beef.shank             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ beef.tenderloin        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ beer                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ beet                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bell.pepper            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ berry                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ beverly.hills          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ birthday               : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ biscuit                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bitters                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ blackberry             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ blender                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ blue.cheese            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ blueberry              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ boil                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bok.choy               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bon.appÃ.tit           : num  0 1 0 1 1 0 0 1 1 1 ...
##  $ bon.appï..ï..tit       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ boston                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bourbon                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ braise                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bran                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brandy                 : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ bread                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ breadcrumbs            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ breakfast              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brie                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brine                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brisket                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ broccoli               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ broccoli.rabe          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ broil                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brooklyn               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brown.rice             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brownie                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brunch                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ brussel.sprout         : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ buffalo                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ buffet                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bulgaria                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bulgur                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ burrito                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ butter                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ buttermilk              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ butternut.squash        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ butterscotch.caramel    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cabbage                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cake                    : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ california              : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ calvados                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cambridge               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ campari                 : num  0 0 0 0 0 0 0 0 0 0 ...
##   [list output truncated]
##  - attr(*, "na.action")= 'exclude' Named int  4 7 8 12 22 23 24 31 32 35 .
..
##   ..- attr(*, "names")= chr  "4" "7" "8" "12" ...
```

```
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at http
s://goo.gl/13EFCZ
```

```
library("factoextra")
```

```
df <- df1[1:1000, 1:6]
na.exclude(df)
```

```
##
title
## 1                                                                    Lenti
l, Apple, and Turkey Wrap
## 2                                                 Boudin Blanc Terr
ine with Red Onion Confit
## 3                                                                     Pot
ato and Fennel Soup Hodge
## 5
Spinach Noodle Casserole
## 6
The Best Blts
## 9
Korean Marinated Beef
## 10                                       Ham Persillade with Mustard Pot
ato Salad and Mashed Peas
## 11                                               Yams Braised with C
ream, Rosemary and Nutmeg
```

```
## 13                                                 Banana-Chocolate Chip Cake Wi
th Peanut Butter Frosting
## 14                                                                 Beef Tenderlo
in with Garlic and Brandy
## 15
Peach Mustard
## 16
Raw Cream of Spinach Soup
## 17                                                                         Swee
t Buttermilk Spoon Breads
## 18                                                                           Cr
isp Braised Pork Shoulder
## 19                                                  Mozzarella-Topped Peppers
with Tomatoes and Garlic
## 20                    Tuna, Asparagus, and New Potato Salad with Chive Vin
aigrette and Fried Capers
## 21                                                Asian Pear and Watercress S
alad with Sesame Dressing
## 25
Sea Salt-Roasted Pecans
## 26
Garlic Baguette Crumbs
## 27
Cucumber-Basil Egg Salad
## 28
Dried Pear Crisps
## 29                          Green Bean, Red Onion, and Roast Potato Salad
with Rosemary Vinaigrette
## 30
Apricot-Cherry Shortcakes
## 33                                                  Roasted Sweet-Potato Spea
rs with Bacon Vinaigrette
## 34
Deviled Ham
## 36
Aztec Chicken
## 38
Sauteed Broccoli Rabe
## 39                                                                         Grou
per with Tomato and Basil
## 40                                                                          Bet
ter-Than-Pita Grill Bread
## 41                                                                           Co
conut-Key Lime Sheet Cake
## 42                                                Baked Halibut with Orzo, Spi
nach, and Cherry Tomatoes
## 46
Pickled Red Onions
## 47                                                                           S
picy Black Beans and Rice
```

```
## 49
Mexican Lime Soup
## 50                                                                    Cit
rus Salad with Mint Sugar
## 51                                                                  Mexica
n Chile and Mushroom Soup
## 52                                                                    Pea
nut Butter-Banana Muffins
## 54                                                  Pancetta Roast Chi
cken with Walnut Stuffing
## 55                                                                    197
7 Coconut Angel Food Cake
## 57                                                  Veal Burgers Stuff
ed with Mozzarella Cheese
## 58
Pumpkin Muffins
## 59
Orange Balsamic Glaze
## 60                                        Roasted Eggplant and Olive Spr
ead with Pita Bread Chips
## 61                                                                     P
ecan Blue Cheese Crackers
## 62                    Romaine, Grilled Avocado, and Smoky Corn Salad with
Chipotle-Caesar Dressing
## 63                                      Southwest Corn Bread Stuffing wi
th Corn and Green Chilies
## 64                                              Colin Perryâ\200\231s S
orghum and Apple Sticky Pudding
## 65
Mixed Berry Pavlovas
## 67                                                                    Sca
rborough Fair Tofu Burger
## 68
Italian Vinaigrette
## 69                                       White Chocolate Tartlets with
Strawberries and Bananas
## 70                                                  Tomato-Infused ## 1
253
Preakness
##      rating calories protein fat sodium
## 1     2.500      426      30   7    559
## 2     4.375      403      18  23   1439
## 3     3.750      165       6   7    165
## 5     3.125      547      20  32    452
## 6     4.375      948      19  79   1042
## 9     4.375      170       7  10   1272
## 10    3.750      602      23  41   1696
## 11    3.750      256       4   5     30
## 13    4.375      766      12  48    439
## 14    4.375      174      11  12    176
```

```
## 15    3.125      134       4   3   1394
## 16    4.375      382       5  31    977
## 17    1.875      146       4   5    160
## 18    4.375      890      59  68   1027
## 19    5.000      107       5   7    344
## 20    5.000      421      10  33    383
## 21    4.375      345      11  19    423
## 25    3.750      279       3  30    206
## 26    0.000       95       1   7    103
## 27    3.750      215       6  20    250
## 28    2.500       14       0   0      0
## 29    4.375      351       6  19     79
## 30    4.375      311       5   5    226
## 33    4.375      376       7  18    604
## 34    3.125      185      10  13    765
## 36    3.750      625      39  44   1248
## 38    4.375      107       4  10    329
## 39    4.375      336      44  16    413
## 40    2.500      145       3   6    208
## 41    4.375      483       5  35    100
## 42    4.375      634      44  31    181
## 46    4.375       90       2   0    881
## 47    3.750      202      19   8    815
## 49    4.375      338      14  21    174
## 50    4.375      191       3   1      4
## 51    3.125      166       8  12    508
## 52    3.750      275       6  13    242
## 54    5.000     1203      89  87    583
## 55    3.750      266       4   7    148
## 57    4.375      904      38  70   1413
## 58    4.375      223       4  10    211
## 59    3.750      194       2   3    697
## 60    3.750      177       5   7    116
## 61    3.750       70       2   6     60
## 62    4.375      368       6  32    112
## 63    5.000      293       7  15    565
## 64    0.000      523       8  19    694


## 1187  3.125      224       6  17    120
## 1188  3.750      244       7  21    236
## 1190  4.375      199       3   9     14
## 1191  3.750      137      11   8     78
## 1193  5.000      195       1   3     15
## 1194  4.375     1311      81  85   1222
## 1196  4.375      326       6  11    336
## 1197  4.375      111       2   8    170
## 1199  4.375      507      20  27    957
## 1200  4.375      625      42  30   1642
## 1201  3.750      799      19  44    351
```

```
## 1202  0.000      162       2   0   2872
## 1203  4.375      766      36  43   1330
## 1204  4.375      177       3  11     12
## 1205  5.000      396      10  24    607
## 1206  4.375      312       6   2    255
## 1207  4.375      510      51  20    926
## 1209  3.750     1193      43  99   1384
## 1210  3.125      631       9  37    307
## 1211  3.750      651       5  24    249
## 1212  4.375      611      15  34    391
## 1214  4.375      598       9  37    196
## 1215  3.125      300      15  15     94
## 1216  4.375      261       6  17    173
## 1221  3.750      135       2   5     71
## 1222  0.000      138       0   0      2
## 1223  4.375      296       9  23    283
## 1224  3.750      505       6  29    216
## 1225  3.750       92       7   3     39
## 1226  4.375      126       4   9    142
## 1227  3.750      331       8  10     93
## 1228  3.125      328      38  16    555
## 1230  2.500      378      18  31    489
## 1232  4.375      668      53  32   1393
## 1233  4.375      149       9   8     49
## 1234  4.375      135       1   0      1
## 1235  3.750      321      12  18    537
## 1236  3.750      168       9  13    213
## 1237  3.750      246       2   1     17
## 1238  4.375      380       7   6    363
## 1239  4.375      831      10  66    212
## 1240  4.375      563      30  42   1414
## 1241  4.375      418       5  11     27
## 1242  0.000      562       1   1     46
## 1243  3.750      507      30  38    982
## 1245  4.375      351      23  24   1826
## 1246  4.375      880      69  56    250
## 1247  5.000      639      35  28   1155
## 1248  3.750      457      10  24    499
## 1249  3.750      475      21  29    510
## 1251  5.000     1405      17  96    597
## 1252  0.000      145       0   0     10
## 1253  0.000      136       0   0      2

View(df)
head(df[, 1:6])

##                                        title rating calories protein fat
## 1             Lentil, Apple, and Turkey Wrap   2.500      426      30   7
## 2 Boudin Blanc Terrine with Red Onion Confit   4.375      403      18  23
## 3               Potato and Fennel Soup Hodge   3.750      165       6   7
```

```
## 5                   Spinach Noodle Casserole   3.125      547      20  32
## 6                             The Best Blts   4.375      948      19  79
## 9                Korean Marinated Beef   4.375      170       7  10
##   sodium
## 1    559
## 2   1439
## 3    165
## 5    452
## 6   1042
## 9   1272
```

```
# Prepare Data
df <- na.omit(df) # listwise deletion of missing
#df <- scale(df) # standardize variables
View(df)
set.seed(1234)
ind = sample(1:nrow(df),0.8*nrow(df),replace = F)
df_train =df[ind,-1]
df_test = df[-ind,-1]

summary(df)
```

```
##                            title         rating         calories
##  Pastry Dough                 :  4   Min.   :0.000   Min.   :   2.0
##  Chicken Stock                :  3   1st Qu.:3.750   1st Qu.: 177.0
##  Balsamic Vinaigrette         :  2   Median :4.375   Median : 305.0
##  Blackberry-Raspberry Sauce   :  2   Mean   :3.812   Mean   : 449.1
##  Blue Cheese Coleslaw         :  2   3rd Qu.:4.375   3rd Qu.: 564.8
##  Caramel Macadamia Nut Crunch :  2   Max.   :5.000   Max.   :8603.0
##  (Other)                      :985
##     protein            fat              sodium
##  Min.   :  0.00   Min.   :  0.00   Min.   :    0.0
##  1st Qu.:  3.00   1st Qu.:  7.00   1st Qu.:   78.0
##  Median :  7.00   Median : 17.00   Median :  242.0
##  Mean   : 18.21   Mean   : 25.91   Mean   :  759.5
##  3rd Qu.: 23.00   3rd Qu.: 31.00   3rd Qu.:  657.5
##  Max.   :470.00   Max.   :923.00   Max.   :97225.0
##
```

```
dim(df)
```

```
## [1] 1000    6
```

```
# outlier definition
# x > Q3+1.5*IQR - positive side outlier
# x < Q1-1.5*IQR - negative or lower side outlier
par(mfrow=c(2,3))
(boxplot(df1$rating)$out);(boxplot(df1$calories)$out);(boxplot(df1$protein)$o
ut);(boxplot(df1$fat)$out);(boxplot(df1$sodium)$out)
```

```
##    [1] 2.500 1.875 0.000 2.500 2.500 0.000 0.000 0.000 0.000 2.500 0.000
##   [12] 0.000 2.500 0.000 0.000 0.000 0.000 0.000 0.000 2.500 0.000 0.000
##   [23] 0.000 1.250 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
##   [34] 0.000 2.500 0.000 0.000 0.000 0.000 2.500 0.000 0.000 2.500 0.000
##   [45] 2.500 0.000 2.500 0.000 0.000 0.000 0.000 0.000 0.000 0.000 2.500
##   [56] 2.500 2.500 1.250 2.500 0.000 0.000 2.500 1.250 2.500 0.000 0.000
##   [67] 0.000 0.000 2.500 2.500 0.000 0.000 0.000 1.250 0.000 0.000 1.875
##   [78] 0.000 0.000 2.500 1.250 2.500 0.000 0.000 0.000 0.000 2.500 0.000
##   [89] 2.500 0.000 2.500 2.500 0.000 2.500 2.500 1.250 0.000 2.500 0.000
##  [100] 0.000 0.000 0.000 0.000 2.500 0.000 0.000 0.000 2.500 0.000 0.000
##  [111] 0.000 0.000 0.000 0.000 0.000 2.500 2.500 0.000 0.000 0.000 0.000
##  [122] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
##  [133] 1.250 2.500 2.500 0.000 0.000 0.000 2.500 0.000 1.875 0.000 0.000
##  [144] 0.000 0.000 1.875 0.000 0.000 0.000 0.000 0.000 1.250 0.000 0.000
##  [155] 0.000 0.000 2.500 0.000 1.250 2.500 2.500 1.250 0.000 0.000 0.000
##  [166] 2.500 2.500 2.500 2.500 0.000 0.000 0.000 0.000 1.250 2.500 0.000
##  [177] 2.500 0.000 2.500 0.000 0.000 0.000 0.000 0.000 0.000 1.875 0.000
##  [188] 0.000 0.000 0.000 0.000 0.000 1.875 0.000 0.000 1.250 0.000 1.250
##  [199] 0.000 1.875 1.250 0.000 2.500 0.000 2.500 0.000 0.000 0.000 0.000
##  [210] 0.000 0.000 0.000 0.000 2.500 0.000 0.000 0.000 0.000 0.000 0.000
##  [221] 2.500 2.500 0.000 0.000 0.000 0.000 0.000 2.500 0.000 0.000 0.000
##  [232] 0.000 0.000 2.500 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
##  [243] 0.000 0.000 1.875 0.000 0.000 0.000 0.000 0.000 2.500 2.500 0.000
##  [254] 2.500 0.000 2.500 0.000 2.500 0.000 0.000 0.000 0.000 0.000 0.000
##  [265] 0.000 0.000 2.500 0.000 0.000 2.500 0.000 0.000 0.000 0.000 1.250
##  [276] 0.000 2.500 0.000 0.000 0.000 0.000 2.500 0.000 0.000 0.000 1.875
##  [287] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 2.500 2.500 0.000
##  [298] 2.500 2.500 0.000 2.500 2.500 0.000 1.875 2.500 2.500 0.000 0.000
##  [309] 0.000 0.000 1.875 2.500 0.000 2.500 0.000 1.250 0.000 0.000 0.000
##  [320] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 2.500 0.000 0.000 2.500
##  [331] 0.000 2.500 2.500 2.500 2.500 0.000 2.500 0.000 0.000 0.000 0.000
##  [342] 0.000 0.000 2.500 0.000 2.500 0.000 1.875 0.000 1.875 0.000 0.000
##  [353] 0.000 0.000 2.500 0.000 0.000 0.000 0.000 0.000 2.500 0.000 2.500
##  [364] 0.000 0.000 0.000 2.500 0.000 0.000 0.000 2.500 0.000 1.250 2.500
##  [375] 0.000 0.000 0.000 1.250 0.000 0.000 0.000 0.000 0.000 0.000 0.000
##  [386] 2.500 0.000 0.000 0.000 0.000 1.250 0.000 2.500 0.000 0.000 1.250
##  [397] 0.000 0.000 0.000 2.500 0.000 0.000 0.000 0.000 0.000 1.875 2.500
##  [408] 0.000 0.000 0.000 2.500 2.500 0.000 0.000 0.000 1.250 0.000 2.500
##  [419] 2.500 0.000 0.000 2.500 0.000 0.000 0.000 2.500 0.000 0.000 0.000


##  [722]     5967     1669     2422    16988     2009     8187     2217
##  [729]     3097     1945     2267     2529     4386     2075     3824
##  [736]     1775     2505     2881    11349     2079     2492     2421
##  [743]     1867     2276     1954     2013     1687     2783     1959
##  [750]     4000     2918     2125     7695     1811     3004     1804
##  [757]     2118     5265     2046     1910     1701     3926     2556
##  [764]     4303     5638    66833     2507     4595     2302     4331
##  [771]     2714    15061     1689     2544     2885     2834     2177
##  [778]     3690     1727     2362    11846     2721     2505     1758
```

```
##  [785]     1865    10635     3334     1693     1853     2817     2395
##  [792]     1690     1885     4353     3613     9792     3204     8644
##  [799]    15300     2013     3070     2183     1770     3657     1931
##  [806]     1703     4121     2400    11628     1725     1992     2789
##  [813]     1763     2092     2960     2116     5695     6052     2316
##  [820]     1747     2762     2283     3593     8014     1916     7032
##  [827]     1667     1888     9792     4770     1768     2008     2090
##  [834]     1663     2303     2492     2493     1788     1676     2254
##  [841]     1780     1749     2871     1821     5349     3443     4204
##  [848]     2398     2007     1863     2388     2029     3604     2018
##  [855]     2735     4186     2374     5638     2584    11451     2220
##  [862]     1741     1957     2663     2357     7955     1792     2763
##  [869]     1952    10042     2978     1926     4648     2434     1793
##  [876]    45407     2831     5661     5263     1986     3614     1737
##  [883]     5166     2702     4050     1713     3434     3340     2652
##  [890]    13806    23273     2314     4603     1666     1809     1717
##  [897]     4856    13447     1790     2920     1875     5197     2310
##  [904]     2711     3684     2955     2420     2736     2858     3833
##  [911]    34351     2938     2878     3603     1933    13820     4051
##  [918]    15065     2873     4580     1995     2045     2953     1729
##  [925]     1857     3175     1916     2734     3000     2112     2453
##  [932]     4145     3167     2871   132025     3983     2335     2865
##  [939]     3990     2078    11670     1795     2788     3773     2798
##  [946]    16443     4584     3128     1957     9465     1871     3094
##  [953]     2190     1846     1712    11428     2724    67253     8197
##  [960]     2161     1937     2110     2106     1903     2152     3715
##  [967]     1776     1772     2495     1705     2343     5915     2866
##  [974]    22932     6677     2559     1751     2707     1759     1711
##  [981]     1715     1872     2058     1775     2006     2121     2630
##  [988]     2255     2293     1786     1933     3445     2509    15350
##  [995]     2373     1951     1866     2715     2292     2434     1809
## [1002]    13430     4520     2853     2217     2883     1973     1690
## [1009]     1918     1778     1951     3506     2053     2157    62368
## [1016]     3636     1779     1706     3418     2369     1706     1716
## [1023]     3588     2498     3169     1765     3648     1871     2345
## [1030]     2830     2980     1814     3032     3022     2422     2377
## [1037]     2426     2713     1868     2320     6927     7887     1926
## [1044]     1742     2874     2410     1844     1844     1920     4029
## [1051]     1709     1989     1749     3597     2248     1763     1916
## [1058]     2030     1790     4927     2205     1719     1975     2018
## [1065]     3771     2918     2000     2591     1865     1831     2751
## [1072]     1870     3886     4819     1884     2495     2168     2497
## [1079]     2337     2281     1676     2012     3065     5106     1825
## [1086]     6267     2012     2183     2032     2149     3136     2039
## [1093]     1738     2934     1717     2291     1695     2511     4382
## [1100]     3711     4018     1672     3923     2861     3591     3777
## [1107]     5980     1980     1959     1800     2064     9286     2811
## [1114]     2579     2139     4830     3548     2509     1750     2528
## [1121]    15416     2023     4240     2665     6046     2133     2206
## [1128]     1828     1986     2446     2316 12005810     1741    45240
```

```
## [1135]     2072     1940     2369     2865     2912     1747     1904
## [1142]     2725     1663     1737     2805     2340     3217     3875
## [1149]     5753     3339     2745     2292     5684     2027     3698
```

```
apply(df,2,range)
```

```
##      title
## [1,] "'Wichcraft's Roasted Turkey, Avocado, Bacon, Onion Relish, & AÃ¯oli
on Ciabatta "
## [2,] "Zucchini with Vinegar and Mint "
##      rating  calories protein fat   sodium
## [1,] "0.000" "   2"    "  0"    "  0" "    0"
## [2,] "5.000" "8603"   "470"   "923" "97225"
```

```
apply(df,2,summary)
```

```
##        title       rating      calories    protein     fat
## Length "1000"      "1000"      "1000"      "1000"      "1000"
## Class  "character" "character" "character" "character" "character"
## Mode   "character" "character" "character" "character" "character"
##        sodium
## Length "1000"
## Class  "character"
## Mode   "character"
```

```
# KMeans - comes from Rcmdr library
# Kmeans- from amap library
# kmeans- from stats library

# steps in k-means clustering
#1- preprocessing the data (impute missing values, remove outliers, feature t
rasnformation)
#2- scaling or standardization of data set
#3- decide the number of clusters (value of K)
#4- iterate over the samples to create clusters
#5- decide the distance measure
#6- calculate the group accuracy

# scaling of data
df_train1 <- scale(df_train)
```

```
head(df_train1)
```

```
##           rating    calories    protein         fat      sodium
## 146  -0.07554974 -0.65481746 -0.5359008 -0.46718457 -0.17777045
## 785   0.96651565 -0.38626336 -0.5040280 -0.26121760 -0.17753437
## 769   0.44548296 -0.03097129 -0.3765369 -0.09644403 -0.18650555
## 1252 -3.20174592 -0.52471051 -0.5996463 -0.54957135 -0.19099114
## 1074 -0.59658244  0.49612868  1.1214833  0.19190972  0.02762239
## 803   0.44548296 -0.29118520 -0.5040280 -0.50837796 -0.18579730
```

```r
class(df_train1)
```

```
## [1] "matrix"
```

```r
# screeplot approach to decide the number of clusters
km = kmeans(df_train1,1)
km$withinss
```

```
## [1] 3995
```

```r
km$tot.withinss
```

```
## [1] 3995
```

```r
km = kmeans(df_train1,2)
km$withinss
```

```
## [1]  781.8523 2580.4197
```

```r
km$tot.withinss
```

```
## [1] 3362.272
```

```r
km = kmeans(df_train1,3)
km$withinss
```

```
## [1]  612.91771   39.87556 1797.64218
```

```r
km$tot.withinss
```

```
## [1] 2450.435
```

```r
km = kmeans(df_train1,4)
km$withinss
```

```
## [1]   39.87556  202.40093 1151.34612  405.29139
```

```r
km$tot.withinss
```

```
## [1] 1798.914
```

```r
km = kmeans(df_train1,5)
km$withinss
```

```
## [1] 440.03536  39.87556 102.13818 396.48642 202.40093
```

```r
km$tot.withinss
```

```
## [1] 1180.936
```

```r
km = kmeans(df_train1,6)
km$withinss
```

```
## [1] 438.01940 173.08807 121.65624 202.40093 102.13818  37.65537
```

```r
km$tot.withinss
```

```
## [1] 1074.958
```

```r
km = kmeans(df_train1,7)
km$withinss
```

```
## [1] 102.13818 202.40093  18.03701  37.65537 276.95475 140.32102 145.09666
```

```r
km$tot.withinss
```

```
## [1] 922.6039
```

```r
km = kmeans(df_train1,8)
km$withinss
```

```
## [1] 125.26434  37.65537  62.97005 141.34211  64.68009 102.13818 149.93891
## [8]  76.93294
```

```r
km$tot.withinss
```

```
## [1] 760.922
```

```r
km = kmeans(df_train1,9)
km$withinss
```

```
## [1] 102.13818  88.72301 141.34211  47.65000  56.04750  37.65537  90.78147
## [8]  68.16672  62.97005
```

```r
km$tot.withinss
```

```
## [1] 695.4744
```

```r
km = kmeans(df_train1,10)
km$withinss
```

```
##  [1]  48.03969 102.13818 141.34211  55.96067  41.80232  37.65537  67.54641
##  [8]  42.58579  62.97005  53.27464
```

```r
km$tot.withinss
```

```
## [1] 653.3152
```

```r
dev.off()
```

```
## null device
##           1
```

```r
sumsq=NULL
for (i in 1:25)
  sumsq[i] = sum(kmeans(df_train,centers=i,
                        iter.max = 1000,
                        nstart=i,
                        algorithm='Forgy')$withinss)
```

```r
plot(1:25,sumsq,type='b', main='Screeplot showing within group sum of squares
')

km = kmeans(df_train1,3)
km$withinss

## [1]  612.91771   39.87556 1797.64218

km$tot.withinss

## [1] 2450.435

class(km$cluster)

## [1] "integer"

summary(km)

##              Length Class  Mode
## cluster      800    -none- numeric
## centers       15    -none- numeric
## totss          1    -none- numeric
## withinss       3    -none- numeric
## tot.withinss   1    -none- numeric
## betweenss      1    -none- numeric
## size           3    -none- numeric
## iter           1    -none- numeric
## ifault         1    -none- numeric

km$centers

##       rating   calories    protein        fat      sodium
## 1  0.2462425 -0.1470263 -0.1454249 -0.1255707 -0.08042799
## 2 -3.0692800 -0.2659102 -0.3538479 -0.2144387 -0.11265136
## 3  0.2175312  2.4495400  2.5345094  2.0765074  1.29964639

as.numeric(km$cluster)

##   [1] 1 1 1 2 1 1 1 1 1 1 3 1 1 1 1 1 1 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 1
1
##  [36] 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 3 1 1 2 1 1 1 1 1
2
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 3 1 1 1 1 1 1 1
3
## [106] 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 3 3 1 1 1 1 1 1 2 1
1
## [141] 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 3 1 1 1 1 1 1 1 1 1
2
## [176] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1
1
## [211] 1 1 1 1 2 1 1 3 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 2 1 1 1
1
```

```
## [246] 2 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1
## [281] 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1 3 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2
1
## [316] 3 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
1
## [351] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 3 3 3 1 1 2 1 1 1 1
1
## [386] 1 1 1 1 3 1 1 1 1 1 1 1 2 1 1 3 1 1 1 1 1 3 1 1 1 3 1 1 1 1 1 1 1
1
## [421] 1 1 3 1 1 3 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1
3
## [456] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1
1
## [491] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 1 1 1 1 1 3 1 1
1
## [526] 1 1 1 1 2 1 2 3 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1
1
## [561] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1
1
## [596] 1 1 1 1 1 1 1 2 3 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 3 1 1 1 1
1
## [631] 3 1 1 1 1 1 1 1 1 1 3 1 1 1 2 1 1 1 1 2 1 1 1 3 1 2 1 1 1 1 1 1 1 1
2
## [666] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1
1
## [701] 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 3 1 1 1
1
## [736] 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 3 1 2 1 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1
2
## [771] 1 3 2 1 1 1 1 1 1 1 1 1 1 1 2 3 3 1 1 3 1 1 1 1 1 1 1 1 1 3
```

```
length(km$cluster)
```

```
## [1] 800
```

```
dim(df_train)
```

```
## [1] 800   5
```

```
class(df_train)
```

```
## [1] "data.frame"
```

```
df_train$cl <- km$cluster

head(df_train)
```

```
##      rating calories protein fat sodium cl
## 146   3.750       67       2   4     66  1
## 785   5.000      228       3  14     67  1
## 769   4.375      441       7  22     29  1
```

```
## 1252  0.000      145       0   0     10  2
## 1074  3.125      757      54  36    936  1
# profiles of clusters
aggregate (df_train [, 1:5], list(df_train[,6]),mean)
  Group.1 rating    calories   protein    fat    sodium
1       1 4.1360029 371.4242 14.251082 20.58586  478.3218
2       2 0.1588983 300.1525 7.711864 16.27119   341.8305
3       3 4.1015625 1928.0833 98.333333 127.50000 6324.0208
```

```
table(df1$rating)
```

```
##
##     0  1.25 1.875   2.5 3.125  3.75 4.375     5
##  1296   123    81   405  1165  4136  6552  2106
```

```
table(df1$calories)
```

```
##
##        0        1        2        3        4        5        6        7
##        8        4       11        7        7        1        9        5
##        8        9       10       11       12       13       14       15
##        5        6        8        9        9       12       10       12
##       16       17       18       19       20       21       22       23
##       13        9       13       21       18       18       15       19
##       24       25       26       27       28       29       30       31
##        9       10        7       19       18        6        4        7
##       32       33       34       35       36       37       38       39
##       17        7        7       13        9       11       21       10
##       40       41       42       43       44       45       46       47
##        7       23       11       18       19       14       12       14
##       48       49       50       51       52       53       54       55
##       14       10       17       14       18       15       17       10
##       56       57       58       59       60       61       62       63
##       13       20       16       19       22       19       29       12
##       64       65       66       67       68       69       70       71
##       19       18       14       21       16       14       17       11
##       72       73       74       75       76       77       78       79
##        6       12       23       10       13       19       11       19
##       80       81       82       83       84       85       86       87
##       16       12       20       14       17       13       18       16
##       88       89       90       91       92       93       94       95

##        1        1        1        1        1        1        1        1
##     6996     7141     7202     7469     7576     8179     8275     8406
##        1        1        1        1        1        1        1        1
##     8414     8603     8624     8844     8858     9101     9799     9811
##        1        1        1        1        1        1        1        1
##     9831    11453    12010    12213    12824    16050    16761    19576
##        1        1        1        1        1        1        1        1
##    22312    24117    54512  3358029  3358273  4157357  4518216 13062948
##        3        2        1        1        1        2        1        1
```

```
## 29997918 30111218
##        1        1

table(df1$X22.minute.meals)

##
##     0     1
## 15849    15

table(df1$sodium)

##
##        0        1        2        3        4        5        6        7
##       52      141      172      160      152      116      108      114
##        8        9       10       11       12       13       14       15
##       91       83       93       76       79       78       74       61
##       16       17       18       19       20       21       22       23
##       36       71       58       50       43       42       50       61
##       24       25       26       27       28       29       30       31
##       37       33       62       36       31       34       43       44
##       32       33       34       35       36       37       38       39
##       42       34       55       45       39       36       28       20
##       40       41       42       43       44       45       46       47
##       42       34       40       34       37       30       35       38
##       48       49       50       51       52       53       54       55
##       29       38       35       28       34       20       34       26
##       56       57       58       59       60       61       62       63
##       28       37       24       27       36       29       23       30
##       64       65       66       67       68       69       70       71
##       28       25       41       29       23       22       25       42
##       72       73       74       75       76       77       78       79
##       31       31       31       29       23       22       34       26
##       80       81       82       83       84       85       86       87
##       19       34       19       25       23       17       26       26
##       88       89       90       91       92       93       94       95
##       24       31       21       28       28       23       21       25
##       96       97       98       99      100      101      102      103
##       16       22       17       27       26       28       23       20
##      104      105      106      107      108      109      110      111
##       12       33       31       18       31       35       26       19
##      112      113      114      115      116      117      118      119
##       23       30       14       22       23       18       28       18
##      120      121      122      123      124      125      126      127
##       26       23       12       31       32       19       22       15
##      128      129      130      131      132      133      134      135
##       23       27       25       19       18       15       25       26
##      136      137      138      139      140      141      142      143
##       25       21       13       26       16       15       30       20
##      144      145      146      147      148      149      150      151
##       16       12       22       25       21       29       25       25
##      152      153      154      155      156      157      158      159
```

```
##       16       27       24       26       30       20       11       23
##      160      161      162      163      164      165      166      167
##       25       17       28       28       18       18       15       15
##      168      169      170      171      172      173      174      175
##       22       18       28       15       19       19       19       15
##      176      177      178      179      180      181      182      183
##       16       16       16       16       18       13       24       10
##      184      185      186      187      188      189      190      191
##       29       13       13       14       16       16       13       20
##      192      193      194      195      196      197      198      199
##        6       15       23       11       21       15       26       24
##      200      201      202      203      204      205      206      207
##       21       21       26       25       26       15       24       13
##      208      209      210      211      212      213      214      215
##       19       15       17       19       20       18       14       15
##      216      217      218      219      220      221      222      223
##       17       13       14       19       23       14       12       10
##      224      225      226      227      228      229      230      231
##       15       14       18        9       14       16       21       27
##      232      233      234      235      236      237      238      239
##       19       13       16       16       14       22       12       17
##      240      241      242      243      244      245      246      247
##       15       22       21       25       18       17       14       10
##      248      249      250      251      252      253      254      255
##       12       17       19       16       20       15       14       14
##      256      257      258      259      260      261      262      263
##       18       10       14        8       20       10       10       10
##      264      265      266      267      268      269      270      271
##       12       12       19       17       15       12       14       11
##      272      273      274      275      276      277      278      279
##       21        8       10       11       12        6       11       11
##      280      281      282      283      284      285      286      287
##       10       14       10       13       13       11       10       10
##      288      289      290      291      292      293      294      295
##       11       10       15        9       14       18       16       19
##      296      297      298      299      300      301      302      303
##       20       22       19       17       12       13       19       14
##      304      305      306      307      308      309      310      311
##       10       19       12       18       10       15       10       11
##      312      313      314      315      316      317      318      319

##        1        2        1        1        1        1        2        1
##     7224     7273     7279     7302     7546     7666     7695     7707
##        1        1        1        1        1        1        1        1
##     7887     7955     8014     8023     8112     8187     8197     8470
##        1        1        1        1        1        1        1        1
##     8644     8748     8945     9040     9286     9465     9478     9573
##        1        1        2        1        1        1        1        1
##     9792    10042    10231    10543    10635    10672    11150    11298
##        2        1        1        1        1        1        2        1
```

```
##    11306    11349    11416    11428    11451    11462    11628    11670
##        1        1        1        1        1        1        1        1
##    11779    11846    11919    12450    12845    12862    13006    13430
##        1        1        1        2        1        1        1        1
##    13447    13767    13805    13806    13820    13869    13875    13999
##        1        1        1        1        3        1        1        1
##    14276    15061    15065    15300    15350    15416    15804    16056
##        1        1        1        1        1        1        1        1
##    16104    16443    16813    16984    16988    17544    18212    18898
##        1        1        1        2        1        1        1        1
##    19149    19986    20492    22579    22583    22593    22859    22932
##        1        1        2        1        1        1        1        1
##    23061    23273    23361    24382    30466    34351    37191    45166
##        1        1        1        1        1        1        2        1
##    45240    45351    45407    45573    55097    55369    62059    62368
##        1        1        1        1        1        1        1        1
##    66833    67253    67615    67884    67909    90572    97225   116178
##        1        1        1        1        1        1        1        1
##   132025   132220  3134853  3449373  3449512  7540990 12005810 27570999
##        1        1        2        1        1        1        1        1
## 27675110
##        1
```

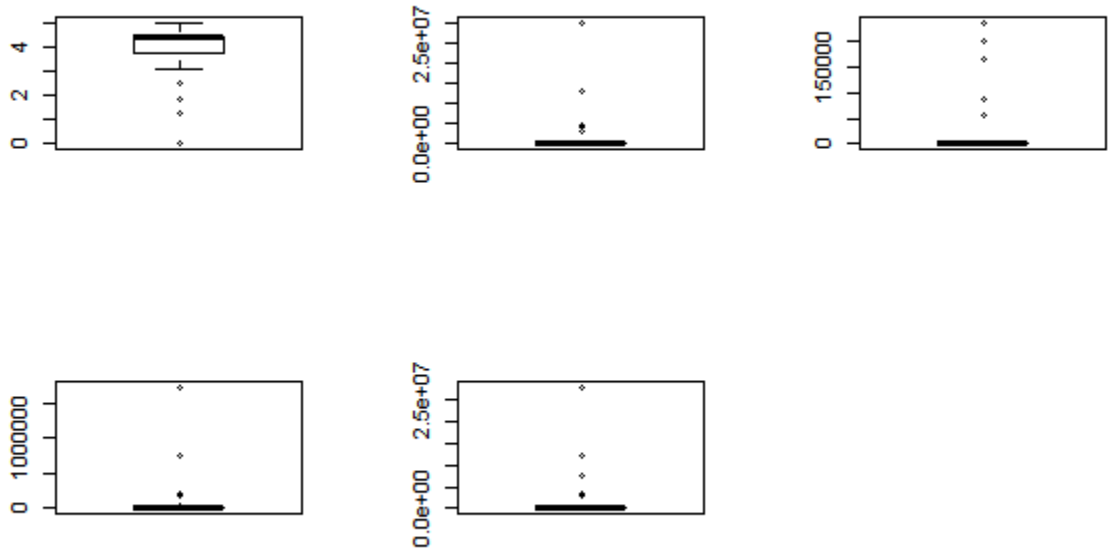
```
library(cluster)

clusplot(df_train,df_train$cl,cex=0.9,color=T,shade=T, labels=4,lines=0)

#HC clustering or Hierarchical Clustering
# distance (euclidean, manhattan, cosine distance)

# Divisive method (top down)
# Agglomorative method (bottom up)

df_train = df_train[,-5]
head(df_train)
```

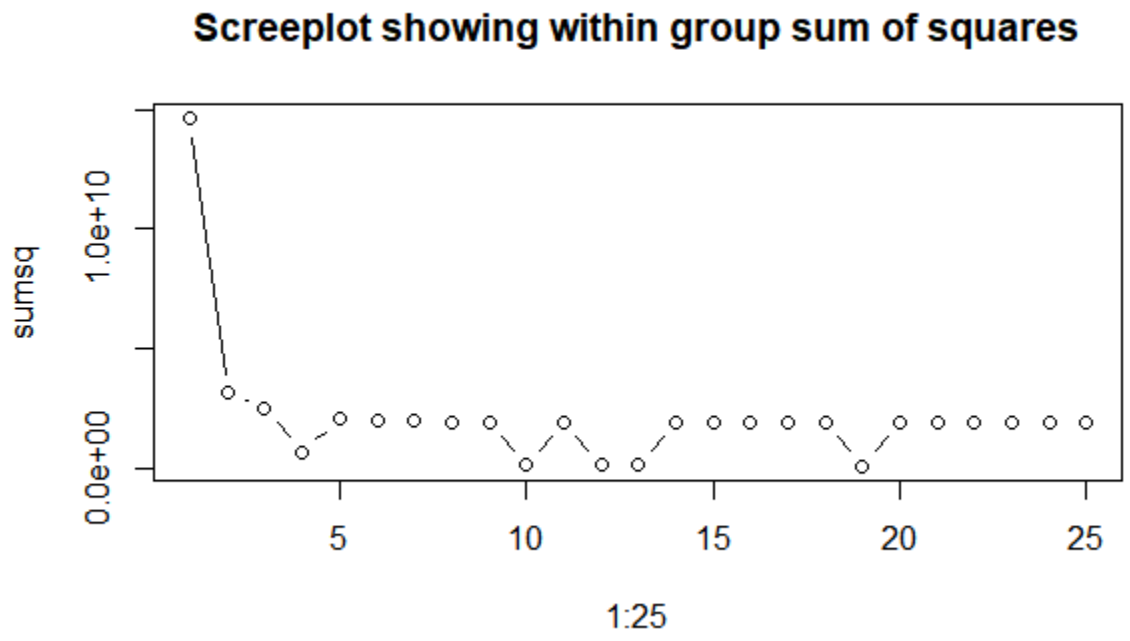
```
##      rating calories protein fat cl
## 146   3.750       67       2   4  1
## 785   5.000      228       3  14  1
## 769   4.375      441       7  22  1
## 1252  0.000      145       0   0  2
## 1074  3.125      757      54  36  1
## 803   4.375      285       3   2  1
```

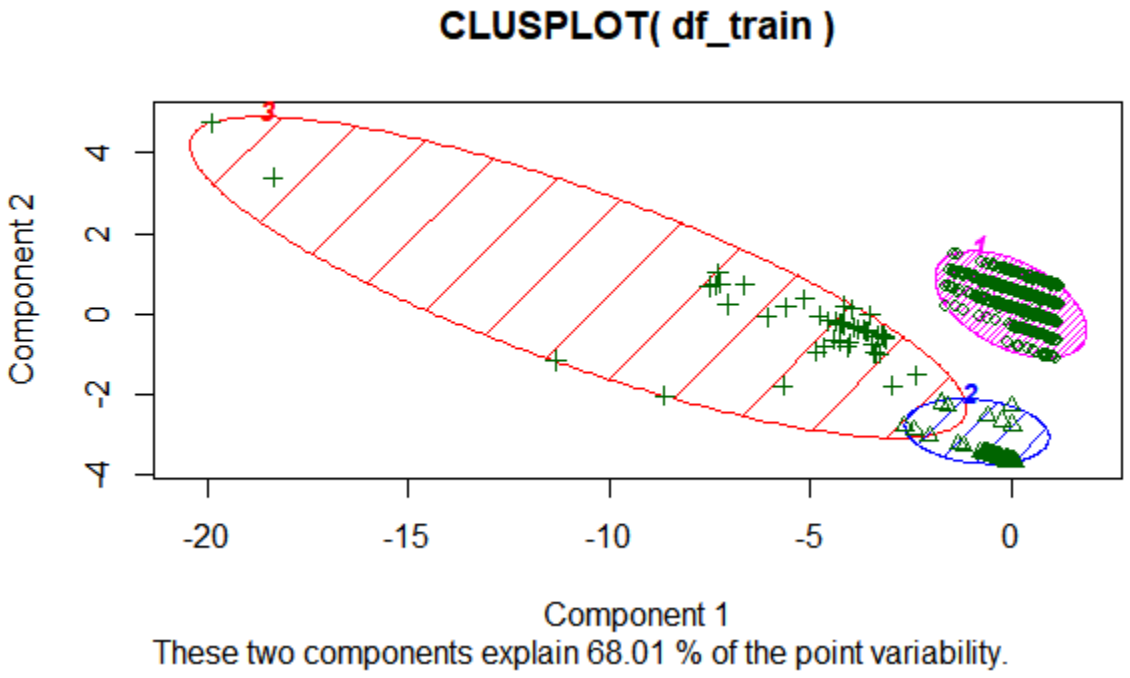
```
# compute the distance metrix
d1 <- dist(df_train,method='euclidean')
summary(d1)
```

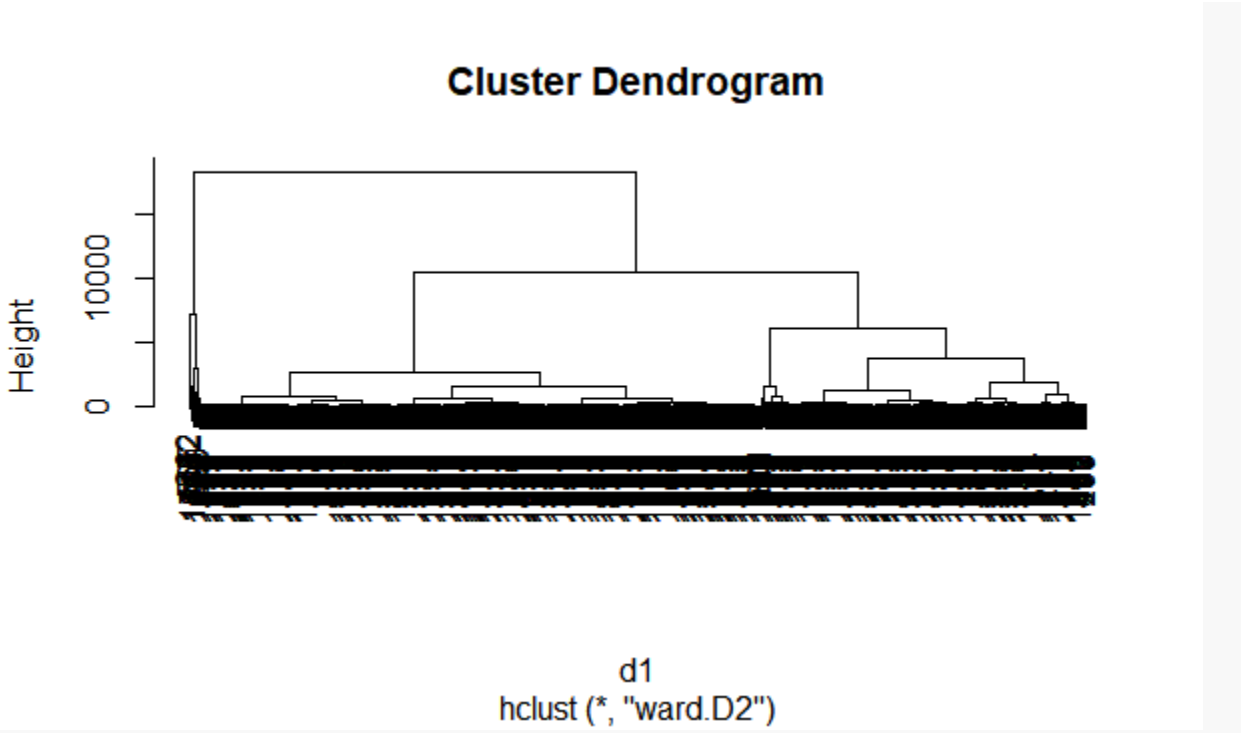
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   111.2   255.8   437.8   510.4  8650.4
```

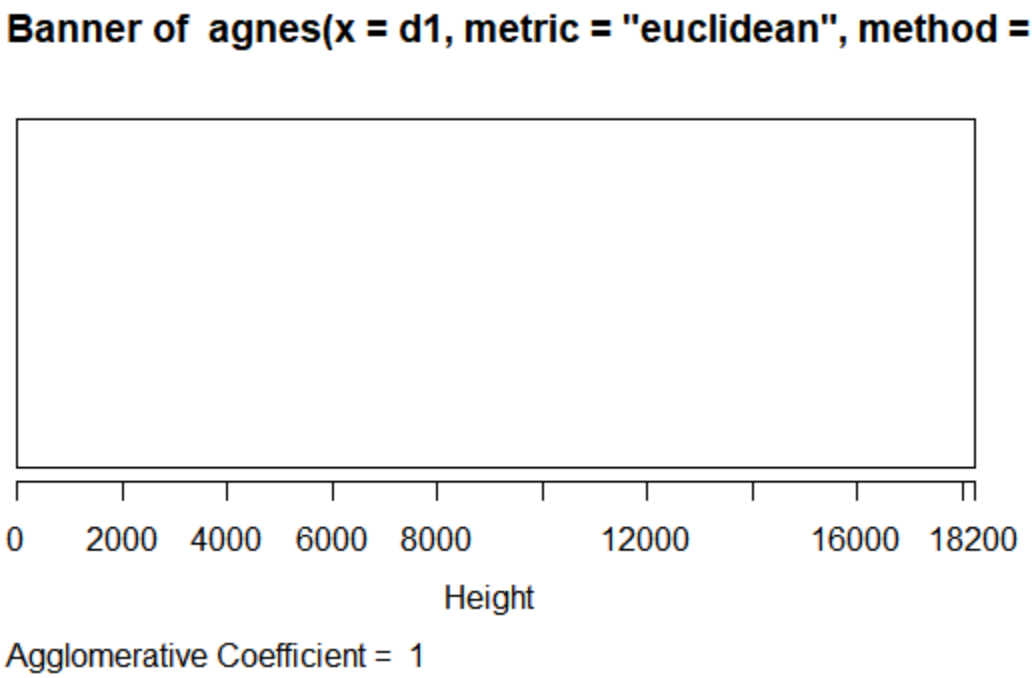
```r
# HC
fit <- hclust(d1,method = 'ward.D2')
plot(fit)

# single, double, average, ward, ward.D2

# agglomorative method
fit <- agnes(d1,metric='euclidean',method = 'ward')
plot(fit)

# divisive method
fit <- diana(d1,metric='euclidean')
plot(fit)
```

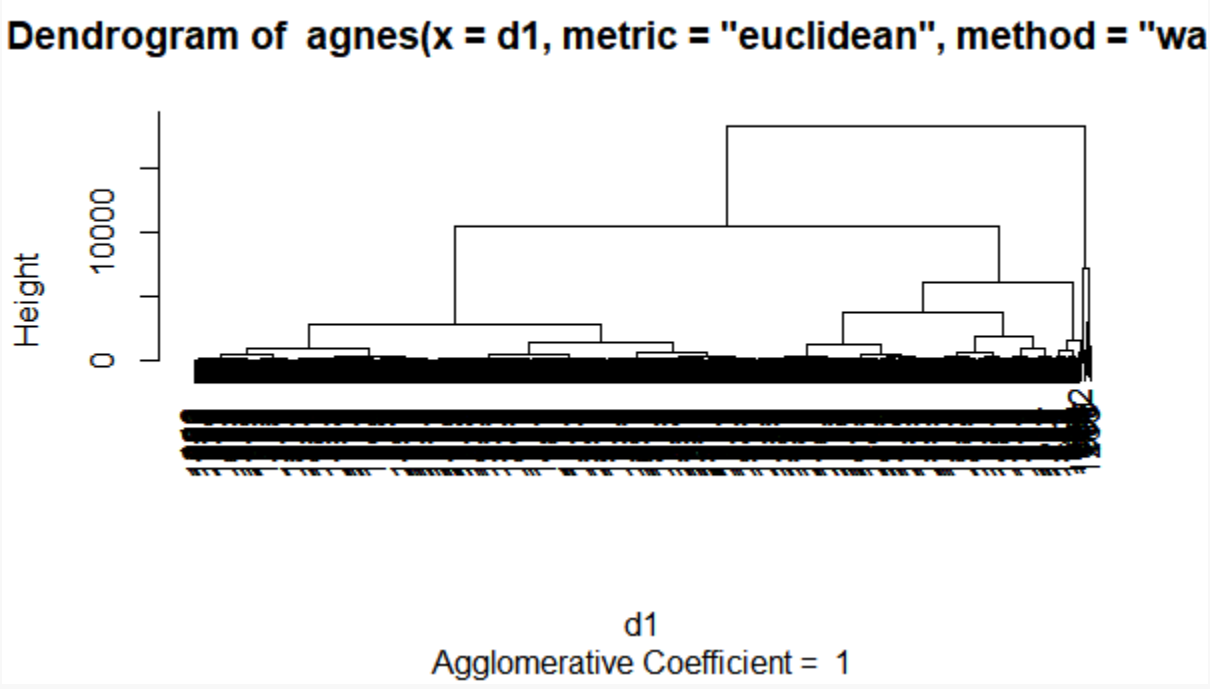


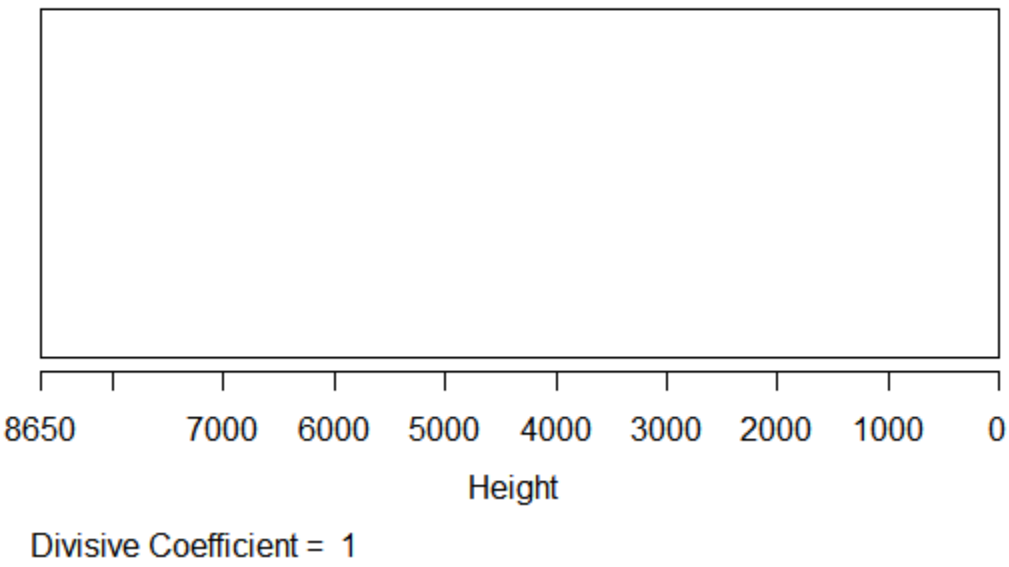

Screeplot showing within group sum of squares
sumsq
0.0e+00
1.0e+10
5
10
15
20
25
1:25

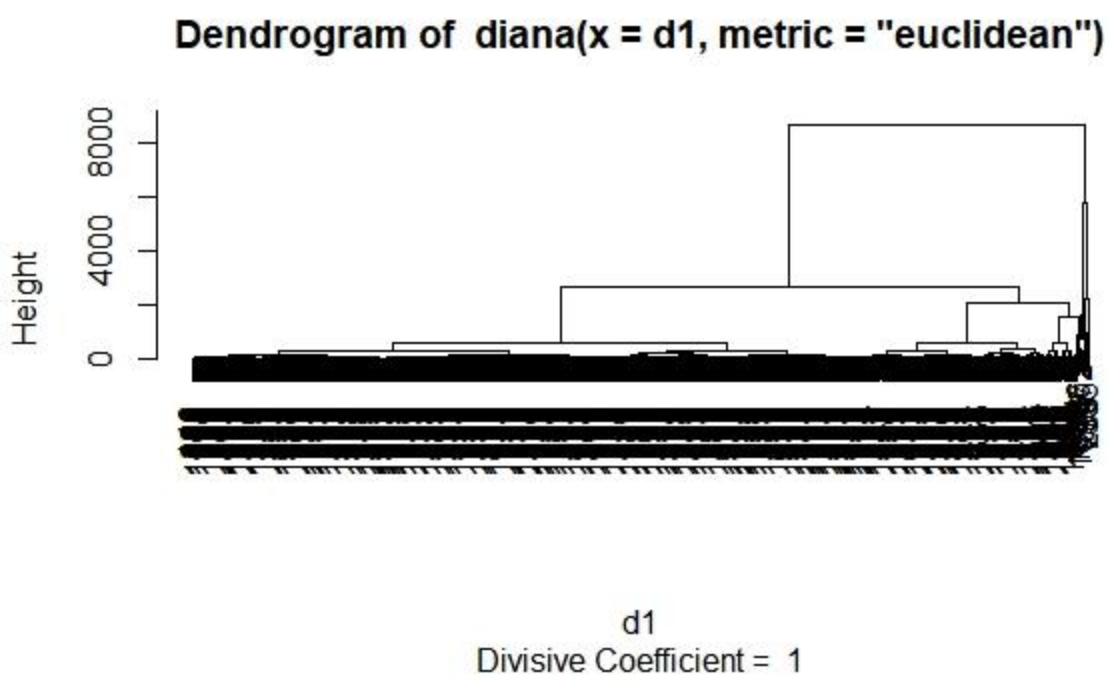

CLUSPLOT(df_train)
Component 2
4
2
0
-2
-4
-20
-15
-10
-5
0
Component 1
These two components explain 68.01 % of the point variability.


Cluster Dendrogram
Height
0
10000
d1
hclust (*, "ward.D2")


Banner of agnes(x = d1, metric = "euclidean", method = "
0
2000
4000
6000
8000
12000
16000
18200
Height
Agglomerative Coefficient = 1


Dendrogram of agnes(x = d1, metric = "euclidean", method = "wa
Height
10000
0
d1
Agglomerative Coefficient = 1


Banner of diana(x = d1, metric = "euclidean")
8650
7000
6000
5000
4000
3000
2000
1000
0
Height
Divisive Coefficient = 1


Dendrogram of diana(x = d1, metric = "euclidean")
Height
0
4000
8000
d1
Divisive Coefficient = 1