**Session22\_Assignment\_modified.R**

2. Perform the below given activities:

a. apply K-means clustering to identify similar recipies

b. apply K-means clustering to identify similar attributes

c. how many unique recipies that people order often

d. what are their typical profiles

Discussion:-

Based on the assumption that o and 1 are indication of people order recipes, we have modified

the data base , based on higher no of 1 and sorted then named it as epir\_1 and the cluster

analysis is performed.

Based on the analysis the aggregate group and cluster are given below.

head(df\_train)

## rating calories protein fat sodium cl

## 175 3.125 259 3 22 164 3

## 868 3.750 619 3 9 255 3

## 850 5.000 587 7 26 172 3

## 1369 3.750 203 6 11 1040 3

## 1185 4.375 408 9 20 461 3

## 889 4.375 188 2 1 10 3

# profiles of clusters

aggregate(df\_train[,1:5],list(df\_train[,6]),mean)

## Group.1 rating calories protein fat sodium

## 1 1 0.8088235 214.7353 3.647059 8.50000 205.0588

## 2 2 3.4134615 1891.3462 81.346154 108.53846 2303.0769

## 3 3 4.1368626 315.8584 9.114731 16.76204 280.6119

setwd("C:/Users/krish/Desktop/sv R related/acadgild/assignments/session 22/e

picurious")

library(readr)

epi\_r1 <- read.csv("epi\_r1.csv")

View(epi\_r1)

df<-epi\_r1

df[df==""] <- NA

df1<-na.exclude(df)

View(df1)

str(df1)

## 'data.frame': 15864 obs. of 681 variables:

## $ title : Factor w/ 17736 levels "'Wichcraft's Roasted

Turkey, Avocado, Bacon, Onion Relish, & AÃ¯oli on Ciabatta ",..: 2728 12026 7

098 12233 4953 16811 5964 5951 4907 13864 ...

## $ rating : num 4.38 4.38 4.38 5 4.38 ...

## $ calories : int 148 274 466 150 208 512 438 338 215 247

...

## $ protein : int 2 10 48 0 5 14 12 2 6 6 ...

## $ fat : int 10 0 28 0 17 47 40 1 20 15 ...

## $ sodium : int 57 28 998 1 347 562 868 33 250 418 ...

## $ X.cakeweek : int 0 0 0 0 0 0 0 0 0 0 ...

## $ X.wasteless : int 0 0 0 0 0 0 0 0 0 0 ...

## $ X22.minute.meals : int 0 0 0 0 0 0 0 0 0 0 ...

## $ X3.ingredient.recipes : int 0 0 0 0 0 0 0 0 0 0 ...

## $ X30.days.of.groceries : int 0 0 0 0 0 0 0 0 0 0 ...

## $ advance.prep.required : int 0 1 0 0 0 0 1 0 1 0 ...

## $ alabama : int 0 0 0 0 0 0 0 0 0 0 ...

## $ alaska : int 0 0 0 0 0 0 0 0 0 0 ...

## $ alcoholic : int 0 1 0 1 0 0 0 0 0 0 ...

## $ almond : int 0 0 0 0 0 0 0 0 0 0 ...

## $ amaretto : int 0 0 0 0 0 0 0 0 0 0 ...

## $ anchovy : int 0 0 0 0 0 1 0 0 0 0 ...

## $ anise : int 0 0 0 0 0 0 0 0 0 0 ...

## $ anniversary : int 0 1 0 0 0 0 0 0 0 0 ...

## $ anthony.bourdain : int 0 0 0 0 0 0 0 0 0 0 ...

## $ aperitif : int 0 0 0 0 0 0 0 0 0 0 ...

## $ appetizer : int 0 0 0 0 1 0 1 0 0 0 ...

## $ apple : int 1 0 0 0 0 0 0 0 0 0 ...

## $ apple.juice : int 0 0 0 0 0 0 0 0 0 0 ...

## $ apricot : int 0 0 0 0 0 0 0 0 0 0 ...

## $ arizona : int 0 0 0 0 0 0 0 0 0 0 ...

## $ artichoke : int 0 0 0 0 0 0 0 0 0 0 ...

## $ arugula : int 0 0 0 0 0 0 0 0 0 0 ...

## $ asian.pear : int 0 0 0 0 0 0 0 0 0 0 ...

## $ asparagus : int 0 0 0 0 0 0 0 0 0 0 ...

## $ aspen : int 0 0 0 0 0 0 0 0 0 0 ...

## $ atlanta : int 0 0 0 0 0 0 0 0 0 0 ...

## $ australia : int 0 0 0 0 0 0 0 0 0 0 ...

## $ avocado : int 0 0 0 0 0 0 0 0 0 0 ...

## $ back.to.school : int 0 0 0 0 0 0 0 0 0 0 ...

## $ backyard.bbq : int 1 0 1 0 1 1 0 1 0 0 ...

## $ bacon : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bake : int 0 0 0 0 0 0 0 0 0 1 ...

## $ banana : int 0 0 0 0 0 0 0 0 0 0 ...

## $ barley : int 0 0 0 0 0 0 0 0 0 0 ...

## $ basil : int 0 0 0 0 0 0 1 0 1 0 ...

## $ bass : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bastille.day : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bean : int 0 0 0 0 0 0 0 0 0 0 ...

## $ beef : int 0 0 0 0 0 0 0 0 0 0 ...

## $ beef.rib : int 0 0 0 0 0 0 0 0 0 0 ...

## $ beef.shank : int 0 0 0 0 0 0 0 0 0 0 ...

## $ beef.tenderloin : int 0 0 0 0 0 0 0 0 0 0 ...

## $ beer : int 0 0 0 0 0 0 0 0 0 0 ...

## $ beet : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bell.pepper : int 0 0 0 0 0 0 0 0 0 0 ...

## $ berry : int 0 1 0 0 0 0 0 0 0 0 ...

## $ beverly.hills : int 0 0 0 0 0 0 0 0 0 0 ...

## $ birthday : int 0 1 0 0 0 0 0 1 0 0 ...

## $ biscuit : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bitters : int 0 0 0 0 0 0 0 0 0 0 ...

## $ blackberry : int 0 0 0 0 0 0 0 0 0 0 ...

## $ blender : int 0 0 0 0 0 0 0 0 0 0 ...

## $ blue.cheese : int 0 0 0 0 0 0 0 0 0 0 ...

## $ blueberry : int 0 0 0 0 0 0 0 0 0 0 ...

## $ boil : int 0 1 0 0 0 0 0 0 1 0 ...

## $ bok.choy : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bon.appÃ.tit : int 1 1 1 0 1 1 0 0 0 1 ...

## $ bon.appï..ï..tit : int 0 0 0 0 0 0 0 0 0 0 ...

## $ boston : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bourbon : int 0 0 0 1 0 0 0 0 0 0 ...

## $ braise : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bran : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brandy : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bread : int 0 0 0 0 0 0 0 0 0 0 ...

## $ breadcrumbs : int 0 0 0 0 0 0 0 0 0 0 ...

## $ breakfast : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brie : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brine : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brisket : int 0 0 0 0 0 0 0 0 0 0 ...

## $ broccoli : int 0 0 0 0 0 0 0 0 0 0 ...

## $ broccoli.rabe : int 0 0 0 0 0 0 0 0 0 0 ...

## $ broil : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brooklyn : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brown.rice : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brownie : int 0 0 0 0 0 0 0 0 0 0 ...

## $ brunch : int 0 0 0 1 0 0 0 0 0 0 ...

## $ brussel.sprout : int 0 0 0 0 0 0 0 0 0 0 ...

## $ buffalo : int 0 0 0 0 0 0 0 0 0 0 ...

## $ buffet : int 1 0 0 0 1 1 0 0 0 0 ...

## $ bulgaria : int 0 0 0 0 0 0 0 0 0 0 ...

## $ bulgur : int 0 0 0 0 0 0 0 0 0 0 ...

## $ burrito : int 0 0 0 0 0 0 0 0 0 0 ...

## $ butter : int 0 0 0 0 0 0 0 0 0 0 ...

## $ buttermilk : int 0 0 0 0 0 0 0 0 0 0 ...

## $ butternut.squash : int 0 0 0 0 0 0 0 0 0 0 ...

## $ butterscotch.caramel : int 0 0 0 0 0 0 0 0 0 0 ...

## $ cabbage : int 0 0 0 0 0 0 0 0 0 0 ...

## $ cake : int 0 0 0 0 0 0 0 0 0 0 ...

## $ california : int 0 0 0 0 0 0 0 0 0 0 ...

## $ calvados : int 0 0 0 0 0 0 0 0 0 0 ...

## $ cambridge : int 0 0 0 0 0 0 0 0 0 0 ...

## $ campari : int 0 0 0 0 0 0 0 0 0 0 ...

## [list output truncated]

## - attr(\*, "na.action")= 'exclude' Named int 1 3 11 14 19 21 25 26 31 35

...

## ..- attr(\*, "names")= chr "1" "3" "11" "14" ...

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

## 2 Celery, App

le, and Fennel Slaw

## 4 Prosec

co-Raspberry GelÃ©e

## 5 Grilled Lemon-Oregano

Chicken Drumsticks

## 6

Rabbit Punch

## 7 Cucumber, To

mato and Feta Salad

## 8 Tusc

an Kale Caesar Slaw

## 9 Fresh Herb Plat

ter (Sabzi Khordan)

## 10 Fresh Fruit Ice Trio: Lime, Wat

ermelon & Pineapple

## 12 Radic

chio with Garlic

## 1363 FrisÃ©e and Celery Salad with Toasted F

ennel-Seed Dressing

## 1364 B

ourbon Creamed Corn

## 1365 Romaine wit

h Parmesan Dressing

## 1367

Plum Applesauce

## 1368 Gratin Dauphinoise (

Scalloped Potatoes)

## 1369 Quinoa and Bul

gur Salad with Feta

## 1370 Crab and Cucumber Pastries

with Mustard Sauce

## rating calories protein fat sodium

## 2 4.375 148 2 10 57

## 4 4.375 274 10 0 28

## 5 4.375 466 48 28 998

## 6 5.000 150 0 0 1

## 7 4.375 208 5 17 347

## 8 4.375 512 14 47 562

## 9 0.000 438 12 40 868

## 10 4.375 338 2 1 33

## 12 3.750 215 6 20 250

## 13 4.375 247 6 15 418

## 15 3.750 295 5 16 480

## 16 3.750 324 11 19 618

## 17 3.125 83 1 7 11

## 18 4.375 196 5 10 400

## 20 3.125 83 1 7 11

## 22 3.125 627 1 61 81

## 23 4.375 142 2 1 14

## 24 5.000 503 6 23 430

## 27 4.375 375 18 26 578

## 28 4.375 391 6 21 19

## 29 4.375 431 33 17 135

## 30 4.375 138 0 0 5

## 32 4.375 221 6 17 52

## 33 2.500 179 1 0 32

## 1344 3.750 186 2 10 118

## 1345 3.750 248 0 27 321

## 1347 0.000 419 19 15 328

## 1349 4.375 738 42 43 209

## 1350 4.375 1051 13 72 518

## 1351 3.125 176 11 14 35

## 1352 2.500 188 1 0 4

## 1353 0.000 222 2 1 3

## 1354 3.750 414 6 30 29

## 1355 4.375 859 8 50 486

## 1356 4.375 84 1 8 40

## 1357 4.375 2330 31 94 992

## 1358 3.125 102 4 4 48

## 1359 3.750 412 7 26 901

## 1360 4.375 123 4 9 404

## 1362 3.125 67 3 3 12

## 1363 5.000 81 1 7 329

## 1364 3.750 414 6 30 29

## 1365 3.750 249 11 21 399

## 1367 5.000 94 1 0 1

## 1368 3.750 228 6 8 42

## 1369 3.750 203 6 11 1040

## 1370 4.375 453 13 35 621

View(df)

head(df[, 1:6])

## title rating calories protein fat

## 2 Celery, Apple, and Fennel Slaw 4.375 148 2 10

## 4 Prosecco-Raspberry GelÃ©e 4.375 274 10 0

## 5 Grilled Lemon-Oregano Chicken Drumsticks 4.375 466 48 28

## 6 Rabbit Punch 5.000 150 0 0

## 7 Cucumber, Tomato and Feta Salad 4.375 208 5 17

## 8 Tuscan Kale Caesar Slaw 4.375 512 14 47

## sodium

## 2 57

## 4 28

## 5 998

## 6 1

## 7 347

## 8 562

# 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

## Classic Red Rice : 3 Min. :0.000

## Amaretto Olive Oil Cake : 2 1st Qu.:3.750

## Apple and Celery Salad : 2 Median :4.375

## Arugula Salad with Lemon-Pepper Dressing : 2 Mean :3.834

## Asian Cabbage Salad : 2 3rd Qu.:4.375

## Avocado Salsa : 2 Max. :5.000

## (Other) :987

## calories protein fat sodium

## Min. : 3.0 Min. : 0.00 Min. : 0.00 Min. : 1.0

## 1st Qu.: 146.8 1st Qu.: 2.00 1st Qu.: 5.00 1st Qu.: 32.0

## Median : 247.0 Median : 5.00 Median : 12.00 Median : 152.0

## Mean : 352.9 Mean : 10.96 Mean : 18.79 Mean : 359.6

## 3rd Qu.: 426.0 3rd Qu.: 11.00 3rd Qu.: 23.25 3rd Qu.: 405.0

## Max. :4562.0 Max. :348.00 Max. :460.00 Max. :15061.0

##

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] 0.000 2.500 2.500 0.000 1.250 2.500 0.000 1.875 0.000 2.500 0.000

## [12] 0.000 1.250 0.000 0.000 0.000 0.000 1.250 2.500 0.000 2.500 0.000

## [23] 0.000 0.000 1.250 0.000 2.500 0.000 0.000 0.000 0.000 2.500 2.500

## [34] 1.875 0.000 0.000 0.000 0.000 0.000 2.500 2.500 0.000 0.000 0.000

## [45] 0.000 0.000 0.000 0.000 0.000 0.000 2.500 0.000 0.000 2.500 2.500

## [56] 1.875 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 2.500 1.250

## [67] 0.000 0.000 2.500 2.500 0.000 1.875 2.500 0.000 0.000 1.875 2.500

## [78] 2.500 2.500 2.500 0.000 2.500 0.000 1.250 1.250 0.000 0.000 1.250

## [89] 0.000 1.250 0.000 1.875 2.500 2.500 2.500 1.875 0.000 2.500 0.000

## [100] 1.250 0.000 0.000 0.000 2.500 0.000 0.000 0.000 1.250 0.000 2.500

## [111] 0.000 2.500 1.250 0.000 0.000 0.000 0.000 2.500 2.500 0.000 0.000

## [122] 0.000 2.500 2.500 1.250 0.000 1.250 1.250 0.000 2.500 1.875 0.000

## [133] 1.250 2.500 0.000 0.000 2.500 2.500 0.000 2.500 0.000 0.000 2.500

## [144] 0.000 0.000 0.000 0.000 1.250 0.000 2.500 0.000 0.000 1.250 0.000

## [155] 1.875 2.500 0.000 0.000 2.500 1.875 1.875 2.500 2.500 0.000 2.500

## [166] 0.000 1.250 0.000 1.250 2.500 1.875 2.500 0.000 2.500 0.000 0.000

## [177] 2.500 1.250 0.000 2.500 0.000 2.500 2.500 2.500 0.000 2.500 2.500

## [188] 0.000 0.000 0.000 0.000 1.875 0.000 0.000 2.500 2.500 0.000 2.500

## [199] 1.875 0.000 1.875 0.000 0.000 1.250 2.500 0.000 0.000 0.000 2.500

## [210] 0.000 0.000 0.000 0.000 2.500 1.250 2.500 0.000 2.500 0.000 2.500

## [1079] 1731 2178 2244 1792 1918 2155 1914

## [1086] 2715 11462 1914 1945 1844 1745 1731

## [1093] 2320 3525 22859 3196 3525 1792 2475

## [1100] 2102 2492 1867 1663 2492 3604 2310

## [1107] 1729 2509 2434 1814 2320 1844 2934

## [1114] 2861 2725 2773 5757 2391 1786 3196

## [1121] 2419 3526 1786 2866 4092 4646 2312

## [1128] 2724 2505 2079 2505 3340 1809 2938

## [1135] 3715 2866 1706 1706 1865 2032 2295

## [1142] 1701 4595 2293 2883 2032 1738 2509

## [1149] 2875 6502 2377 2420 2377 1780 2420

apply(df,2,range)

## title rating calories protein fat

## [1,] "\"Cannoli\" Ice Cream Sandwiches " "0.000" " 3" " 0" " 0"

## [2,] "Zucchini, Tomato, and Corn Salad " "5.000" "4562" "348" "460"

## sodium

## [1,] " 1"

## [2,] "15061"

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

## 175 -0.61680701 -0.2289947 -0.38653872 0.10460888 -0.3003865

## 868 -0.07035562 0.5998100 -0.38653872 -0.35521036 -0.1450024

## 850 1.02254716 0.5261385 -0.19320894 0.24609172 -0.2867264

## 1369 -0.07035562 -0.3579199 -0.24154139 -0.28446894 1.1953990

## 1185 0.47609577 0.1140383 -0.09654406 0.03386746 0.2067463

## 889 0.47609577 -0.3924534 -0.43487116 -0.63817604 -0.5633443

class(df\_train1)

## [1] "matrix"

# screeplot approach to decide the number of clusters

km = kmeans(df\_train1,1)

km$withinss

## [1] 3995

km$tot.withinss

## [1] 3995

km = kmeans(df\_train1,2)

km$withinss

## [1] 1782.1249 992.6804

km$tot.withinss

## [1] 2774.805

km = kmeans(df\_train1,3)

km$withinss

## [1] 72.89837 1166.04827 992.68042

km$tot.withinss

## [1] 2231.627

km = kmeans(df\_train1,4)

km$withinss

## [1] 451.6621 837.1245 148.6584 486.3105

km$tot.withinss

## [1] 1923.755

km = kmeans(df\_train1,5)

km$withinss

## [1] 58.43936 178.41998 206.90469 743.44042 352.97788

km$tot.withinss

## [1] 1540.182

km = kmeans(df\_train1,6)

km$withinss

## [1] 148.65838 122.95590 451.66212 383.87247 121.73544 69.63142

km$tot.withinss

## [1] 1298.516

km = kmeans(df\_train1,7)

km$withinss

## [1] 62.80186 174.34790 384.84071 223.46541 26.14207 214.57696 148.65838

km$tot.withinss

## [1] 1234.833

km = kmeans(df\_train1,8)

km$withinss

## [1] 41.97872 183.68202 180.83736 88.90418 185.51602 159.69707 148.65838

## [8] 89.09993

km$tot.withinss

## [1] 1078.374

km = kmeans(df\_train1,9)

km$withinss

## [1] 27.30353 47.43438 85.55081 142.64389 145.19211 246.45766 148.65838

## [8] 176.23962 41.44880

km$tot.withinss

## [1] 1060.929

km = kmeans(df\_train1,10)

km$withinss

## [1] 73.22619 68.49062 124.91473 114.14763 0.00000 110.33940 148.65838

## [8] 81.46543 106.42891 27.30353

km$tot.withinss

## [1] 854.9748

dev.off()

## null device

## 1

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

## [1] 115.3271 992.6804 1127.6139

km$tot.withinss

## [1] 2235.621

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 -2.6418916 -0.33090248 -0.35526478 -0.3728957 -0.2302779

## 2 -0.3645987 3.52905020 3.40012235 3.1655357 3.3521253

## 3 0.2678870 -0.09809339 -0.09099883 -0.0806615 -0.1012696

as.numeric(km$cluster)

## [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3

## [36] 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 2 3 3 3 2 3 3 3 1 1 3 3 3 3 3 3

3

## [71] 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3

3

## [106] 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3

## [141] 3 3 2 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3

## [176] 3 3 3 3 3 3 3 3 1 3 3 2 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 1 3

3

## [211] 3 3 3 3 3 3 2 3 3 3 1 1 3 3 3 3 2 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3

## [246] 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

2

## [281] 3 1 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3

3

## [316] 3 2 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3

3

## [351] 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3 1 3 3 3

3

## [386] 1 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3

## [421] 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 1 3 1 3 3 2

3

## [456] 3 2 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3

3

## [491] 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 2 2 3 1 3 3 1 3 3 3 3

3

## [526] 3 3 3 2 3 3 1 3 3 3 3 3 3 1 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3

## [561] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 1 3 3 3

3

## [596] 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 1 3 3

3

## [631] 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3

1

## [666] 3 1 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3

3

## [701] 3 1 3 1 3 3 3 3 2 3 3 3 3 3 1 3 3 3 3 1 3 3 1 3 1 1 3 3 3 3 2 3 3 3

3

## [736] 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3

3

## [771] 3 3 3 3 3 2 3 3 3 1 3 3 1 3 3 3 3 1 3 1 3 3 3 3 3 1 3 3 3 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

## 175 3.125 259 3 22 164 3

## 868 3.750 619 3 9 255 3

## 850 5.000 587 7 26 172 3

## 1369 3.750 203 6 11 1040 3

## 1185 4.375 408 9 20 461 3

## 889 4.375 188 2 1 10 3

# profiles of clusters

aggregate(df\_train[,1:5],list(df\_train[,6]),mean)

## Group.1 rating calories protein fat sodium

## 1 1 0.8088235 214.7353 3.647059 8.50000 205.0588

## 2 2 3.4134615 1891.3462 81.346154 108.53846 2303.0769

## 3 3 4.1368626 315.8584 9.114731 16.76204 280.6119

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

## 6370 6694 6836 6841 6857 6912 6927 6929

## 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

## 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

## 175 3.125 259 3 22 3

## 868 3.750 619 3 9 3

## 850 5.000 587 7 26 3

## 1369 3.750 203 6 11 3

## 1185 4.375 408 9 20 3

## 889 4.375 188 2 1 3

str(df\_train)

## 'data.frame': 800 obs. of 5 variables:

## $ rating : num 3.12 3.75 5 3.75 4.38 ...

## $ calories: int 259 619 587 203 408 188 247 35 57 101 ...

## $ protein : int 3 3 7 6 9 2 6 1 1 1 ...

## $ fat : int 22 9 26 11 20 1 15 1 0 7 ...

## $ cl : int 3 3 3 3 3 3 3 3 3 3 ...

# compute the distance metrix

d1 <- dist(df\_train,method='euclidean')

summary(d1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## 0.00 81.81 185.29 324.25 373.77 4560.68

# HC

fit <- hclust(d1,method = 'ward.D2')

plot(fit)