3. DA\_Cluster

Xin Tan, 583833

2019/3/11

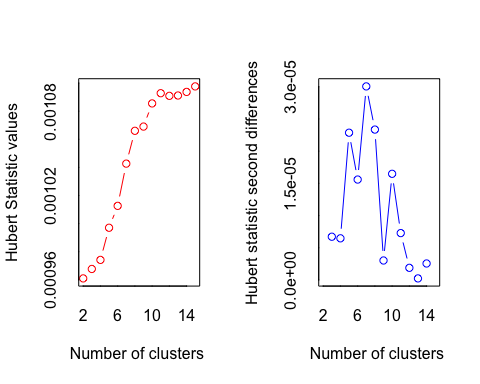
In the cluster, NA means the respondents have no experence about the rating objective, which should not be imputate with another number, otherwise we will lose this important information. Here I will replace the NA with zero, on the one hand, the information has been kept, on the other hand, the orienginal scale and the distribution of data is not influenced too much.

# prepare for the attribute table 结果虽好，但有问题：一旦有个没填，整个attribute都是0  
library(readxl)  
dat <- read\_excel("Data\_Chocolate\_allinterviews.xlsx", sheet = "AttributeRatingsStacked")  
dat<-as.data.frame(dat)  
  
attribute.rating<- aggregate(dat[,-c(1,2)], by=list(dat$Person),mean, na.rm=F)  
attribute.rating[is.na(attribute.rating)] <- 0  
colnames(attribute.rating)[1] <- "Person"

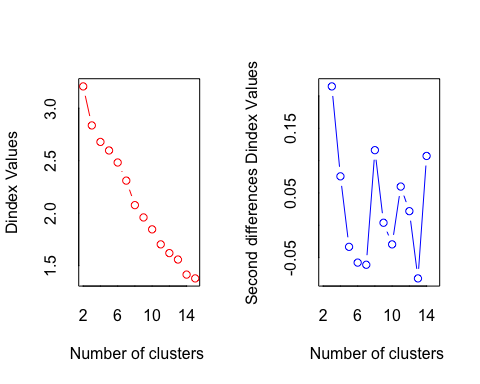
# Cluster only use attribute

# -------------------------------- kmeans  
library(cluster)  
library(NbClust)  
  
# Calinski and Harabasz (suggest 2 cluster)  
NbClust(attribute.rating[-1], method="kmeans")

## Warning in pf(beale, pp, df2): NaNs produced  
  
## Warning in pf(beale, pp, df2): NaNs produced



## \*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a   
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.   
##



## \*\*\* : The D index is a graphical method of determining the number of clusters.   
## In the plot of D index, we seek a significant knee (the significant peak in Dindex  
## second differences plot) that corresponds to a significant increase of the value of  
## the measure.   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## \* Among all indices:   
## \* 9 proposed 2 as the best number of clusters   
## \* 7 proposed 3 as the best number of clusters   
## \* 1 proposed 10 as the best number of clusters   
## \* 1 proposed 11 as the best number of clusters   
## \* 1 proposed 12 as the best number of clusters   
## \* 3 proposed 14 as the best number of clusters   
## \* 2 proposed 15 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 2   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## $All.index  
## KL CH Hartigan CCC Scott Marriot TrCovW  
## 2 7.5067 83.8665 15.5234 8.0232 267.6280 9.582151e+18 4619.9354  
## 3 2.6561 61.9385 7.0823 5.0338 357.0975 3.601831e+18 2565.7535  
## 4 2.8523 48.8142 -0.0195 4.5585 432.7966 1.408921e+18 1811.4313  
## 5 0.3905 35.7948 5.7164 3.0340 453.0248 1.468948e+18 2342.6837  
## 6 1.1582 32.6742 5.1216 3.0155 559.5705 2.511452e+17 1829.3550  
## 7 0.5444 30.5412 9.6125 3.1472 641.1974 6.680613e+16 1324.4585  
## 8 1.6617 32.6266 6.5756 4.6697 749.0790 1.008678e+16 923.0276  
## 9 1.8176 33.0341 4.0346 5.6091 847.1322 1.796299e+15 649.2693  
## 10 0.5182 31.9039 8.0826 5.8416 881.2995 1.119746e+15 512.6137  
## 11 1.4125 34.4407 6.4664 7.3592 996.1750 1.361789e+14 395.4782  
## 12 9.4244 36.1379 1.0943 8.4488 1080.4324 3.004941e+13 289.5879  
## 13 0.1095 33.2723 6.9226 7.7966 1117.0910 1.694136e+13 276.4250  
## 14 4.3940 35.9920 1.8811 9.3818 1210.4183 3.038692e+12 203.5956  
## 15 0.1272 34.3212 -8.2907 9.0424 1275.8048 9.433506e+11 197.3841  
## TraceW Friedman Rubin Cindex DB Silhouette Duda Pseudot2  
## 2 701.2186 585.0722 9.5317 0.4110 0.6297 0.6148 0.6921 16.0131  
## 3 529.8598 642.9725 12.6143 0.3316 1.2477 0.4554 0.8335 2.7960  
## 4 460.4722 716.2712 14.5151 0.3909 1.2747 0.4185 4.7663 -21.3352  
## 5 460.6677 708.6422 14.5089 0.3325 1.5443 0.2397 0.8437 2.4081  
## 6 408.7441 908.3555 16.3520 0.3883 1.5427 0.2420 2.4467 -5.9128  
## 7 366.1267 942.2373 18.2554 0.3069 1.4097 0.2647 0.7392 5.2917  
## 8 299.2340 1123.5740 22.3363 0.3217 1.2151 0.2921 0.8535 1.8877  
## 9 258.7270 1377.5219 25.8334 0.3054 1.1564 0.3115 0.5733 14.8836  
## 10 235.5478 1364.7490 28.3755 0.2710 1.1107 0.3224 1.0233 -0.0228  
## 11 195.9528 1553.7318 34.1092 0.3078 0.9853 0.3520 2.0045 -7.0158  
## 12 168.0836 1759.3607 39.7647 0.3025 0.9311 0.3648 12.2626 0.0000  
## 13 163.3786 1984.2909 40.9098 0.3018 0.8620 0.3884 1.9100 0.0000  
## 14 137.6286 2178.4997 48.5640 0.3826 0.7684 0.4250 0.9703 0.3668  
## 15 130.7942 2473.4893 51.1016 0.3528 0.8893 0.4043 1.3645 -1.8699  
## Beale Ratkowsky Ball Ptbiserial Frey McClain Dunn Hubert  
## 2 3.8459 0.5514 350.6093 0.8651 1.7789 0.2305 0.6671 0.0010  
## 3 1.5529 0.4813 176.6199 0.7812 0.5648 0.5025 0.3033 0.0010  
## 4 -5.2665 0.4269 115.1181 0.7844 -7.1921 0.5265 0.3781 0.0010  
## 5 1.3169 0.3830 92.1335 0.5015 0.1110 1.4420 0.1144 0.0010  
## 6 -4.7289 0.3553 68.1240 0.5105 0.0842 1.4694 0.1426 0.0010  
## 7 2.9390 0.3330 52.3038 0.5276 0.1680 1.4749 0.1520 0.0010  
## 8 1.1437 0.3184 37.4042 0.5327 0.0626 1.4969 0.1944 0.0011  
## 9 6.1407 0.3036 28.7474 0.5377 0.6127 1.4733 0.1944 0.0011  
## 10 -0.1011 0.2905 23.5548 0.5026 0.0111 1.7024 0.1859 0.0011  
## 11 -4.0821 0.2806 17.8139 0.5057 -0.9545 1.6719 0.2181 0.0011  
## 12 0.0000 0.2708 14.0070 0.5181 0.9818 1.5801 0.2181 0.0011  
## 13 0.0000 0.2608 12.5676 0.5168 0.0458 1.5881 0.2181 0.0011  
## 14 0.2507 0.2529 9.8306 0.5178 0.7351 1.5771 0.2804 0.0011  
## 15 -2.1580 0.2450 8.7196 0.4876 4.6122 1.7635 0.2804 0.0011  
## SDindex Dindex SDbw  
## 2 0.6262 3.2080 0.4255  
## 3 0.8924 2.8361 0.4331  
## 4 0.9210 2.6789 0.4305  
## 5 1.3230 2.5975 0.3060  
## 6 1.2827 2.4827 0.3431  
## 7 1.2790 2.3101 0.3748  
## 8 1.2576 2.0765 0.2882  
## 9 1.2108 1.9591 0.2708  
## 10 1.2383 1.8455 0.2464  
## 11 1.1215 1.7025 0.1993  
## 12 1.0813 1.6193 0.2022  
## 13 1.0525 1.5579 0.1800  
## 14 0.9402 1.4144 0.1176  
## 15 1.1919 1.3781 0.1179  
##   
## $All.CriticalValues  
## CritValue\_Duda CritValue\_PseudoT2 Fvalue\_Beale  
## 2 0.7539 11.7537 0.0000  
## 3 0.5459 11.6457 0.1140  
## 4 0.4075 39.2588 1.0000  
## 5 0.4549 15.5808 0.2337  
## 6 0.5846 7.1064 1.0000  
## 7 0.6563 7.8570 0.0006  
## 8 0.4075 15.9943 0.3548  
## 9 0.6372 11.3864 0.0000  
## 10 0.2493 3.0108 1.0000  
## 11 0.6139 8.8034 1.0000  
## 12 0.0916 0.0000 NaN  
## 13 0.0916 0.0000 NaN  
## 14 0.6262 7.1631 0.9964  
## 15 0.6002 4.6632 1.0000  
##   
## $Best.nc  
## KL CH Hartigan CCC Scott Marriot  
## Number\_clusters 12.0000 2.0000 15.0000 14.0000 11.0000 3.000000e+00  
## Value\_Index 9.4244 83.8665 10.1718 9.3818 114.8754 3.787409e+18  
## TrCovW TraceW Friedman Rubin Cindex DB  
## Number\_clusters 3.000 3.0000 15.0000 14.0000 10.000 2.0000  
## Value\_Index 2054.182 101.9712 294.9896 -5.1165 0.271 0.6297  
## Silhouette Duda PseudoT2 Beale Ratkowsky Ball  
## Number\_clusters 2.0000 3.0000 3.000 3.0000 2.0000 3.0000  
## Value\_Index 0.6148 0.8335 2.796 1.5529 0.5514 173.9893  
## PtBiserial Frey McClain Dunn Hubert SDindex Dindex  
## Number\_clusters 2.0000 2.0000 2.0000 2.0000 0 2.0000 0  
## Value\_Index 0.8651 1.7789 0.2305 0.6671 0 0.6262 0  
## SDbw  
## Number\_clusters 14.0000  
## Value\_Index 0.1176  
##   
## $Best.partition  
## [1] 1 1 2 1 2 1 1 2 1 1 1 1 1 1 2 1 1 2 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1  
## [36] 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1

set.seed(2)  
  
seg.k <- kmeans(attribute.rating[-c(1)], centers= 3)  
  
attribute.rating$kmeans <- seg.k$cluster  
clusplot(attribute.rating[-c(1)], seg.k$cluster, color=TRUE, shade=TRUE,labels= , lines=0, main="K-means cluster")  
  
  
table(seg.k$cluster)

##   
## 1 2 3   
## 29 12 9

table(seg.k$cluster)/50

##   
## 1 2 3   
## 0.58 0.24 0.18

# fit statistics  
seg.k$betweenss

## [1] 1396.54

seg.k$totss

## [1] 1926.4

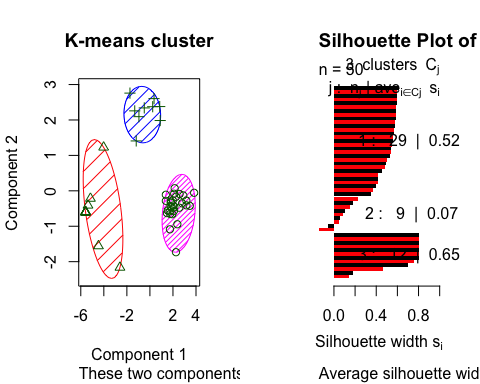
seg.k$betweenss/seg.k$totss

## [1] 0.7249482

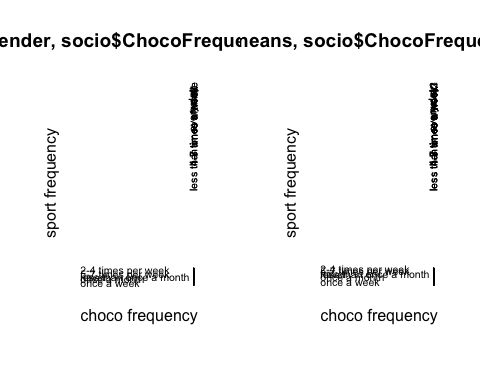
# kmeans Silhouette  
  
library("cluster")  
d <- dist(attribute.rating[-c(1)])  
set.seed(6)  
seg.k <- kmeans(attribute.rating[-c(1)] , centers= 3)  
seg.k

## K-means clustering with 3 clusters of sizes 29, 9, 12  
##   
## Cluster means:  
## crunchy creamy sweet chocolaty healthful calorie rich  
## 1 3.503448 3.372414 4.5448276 4.079310 1.4379310 4.1137931 3.393103  
## 2 2.844444 2.755556 4.2444444 3.655556 1.2333333 4.3222222 2.455556  
## 3 0.000000 0.000000 0.4083333 0.000000 0.3333333 0.4166667 0.000000  
## addiction accessible handy wrapping image commercial kmeans  
## 1 3.206897 4.0620690 4.1344828 3.689655 3.9862069 3.8103448 1  
## 2 1.366667 2.2444444 3.3000000 2.544444 2.4555556 0.3666667 3  
## 3 0.000000 0.3666667 0.6833333 0.275000 0.3833333 0.5750000 2  
##   
## Clustering vector:  
## [1] 1 1 3 1 3 1 2 3 2 1 1 1 2 1 3 2 1 3 1 3 1 1 1 1 3 3 1 1 1 1 1 1 1 3 2  
## [36] 1 2 2 1 1 3 1 3 1 3 1 2 1 1 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 194.4890 190.7533 144.6175  
## (between\_SS / total\_SS = 72.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

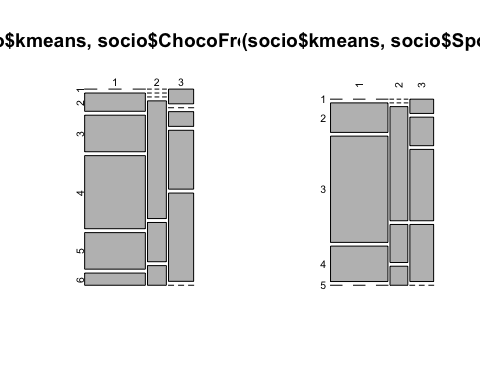
memb2 <- seg.k$cluster  
  
s2 <- silhouette(memb2, d)  
plot(s2, col=1:2, border=NA, main = "Silhouette Plot of the Kmeans Cluster")



# add kmeans in the table  
library(readxl)  
socio<-read\_excel("Data\_Chocolate\_allinterviews.xlsx", sheet = "info\_xin")  
ab <- read\_excel("Data\_Chocolate\_allinterviews.xlsx", sheet = "attri\_xin")  
socio<-as.data.frame(socio)  
ab <- as.data.frame(ab)  
  
socio$kmeans <- seg.k$cluster  
bars<-merge(ab,socio,by="Person")  
  
  
  
write.csv(bars, file = "kmeans3.csv")  
# Mosaic  
  
plot(table(socio$Gender, socio$ChocoFrequency, socio$Sport), las =2, xlab = "choco frequency", ylab = "sport frequency") # 女的比男的爱吃巧克力（结论能不能乱写）  
  
plot(table(socio$kmeans, socio$ChocoFrequency, socio$Sport), las =2, xlab = "choco frequency", ylab = "sport frequency") # 分类为爱吃巧克力和不爱的



# 巧克力的消费程度  
plot(table(socio$kmeans,socio$ChocoFrequency\_Num))  
  
# 运动  
plot(table(socio$kmeans,socio$Sport\_Num), las =2)



# brand  
brand <- read\_excel("Data\_Chocolate\_allinterviews.xlsx", sheet = "Direct Preference Rating")  
brand$kmeans <- seg.k$cluster  
  
seg.summ <- function (data, groups)   
{aggregate (data , list(groups), function (x) mean(as.numeric(x)))}  
  
seg.summ(brand, brand$kmeans)

## Group.1 Person Snickers (b1) KinderBueno (b2) Twix (b3) Mars (b4)  
## 1 1 25.51724 5.896552 5.275862 5.310345 3.965517  
## 2 2 28.00000 6.666667 5.888889 5.777778 4.000000  
## 3 3 23.58333 3.833333 5.333333 4.416667 3.666667  
## KitKat (b5) Bounty (b6) Kinderriegel (b7) BalistoKornMix(b8) Lion (b9)  
## 1 5.000000 4.206897 5.965517 4.482759 4.724138  
## 2 4.555556 3.888889 6.000000 4.555556 4.555556  
## 3 5.500000 3.416667 5.500000 3.750000 3.916667  
## Duplo (bb) kmeans  
## 1 4.448276 1  
## 2 4.555556 2  
## 3 5.083333 3

# 每个人对attribute的打分, omit NA  
dat <- read\_excel("Data\_Chocolate\_allinterviews.xlsx", sheet = "AttributeRatingsStacked")  
id.attr<- aggregate(dat,by=list(dat$Person),mean, na.rm=T)

## 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  
  
## 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  
  
## 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  
  
## 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  
  
## 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  
  
## 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  
  
## 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

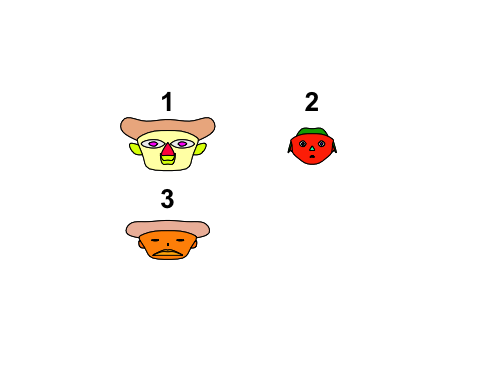
id.attr$kmeans<- seg.k$cluster  
  
tmp <- aggregate(id.attr, by=list(id.attr$kmeans),mean, na.rm=T);tmp

## Group.1 Group.1 Person Product crunchy creamy sweet chocolaty  
## 1 1 25.51724 25.51724 NaN 3.503448 3.372414 4.544828 4.079310  
## 2 2 28.00000 28.00000 NaN 3.227160 3.027160 4.244444 3.655556  
## 3 3 23.58333 23.58333 NaN 3.179067 3.307540 4.519444 3.911706  
## healthful calorie rich addiction accessible handy wrapping  
## 1 1.634483 4.427969 3.577778 3.321839 4.062069 4.134483 3.689655  
## 2 1.522222 4.322222 3.514286 3.091319 3.886111 3.712500 3.237500  
## 3 1.801389 4.270337 3.137446 3.072222 4.188961 4.063955 3.626479  
## image commercial kmeans  
## 1 3.986207 3.810345 1  
## 2 3.741049 3.757540 2  
## 3 3.651356 3.707275 3

#Chernoff faces  
library("MASS")  
library("aplpack")

## Loading required package: tcltk

tmp <- tmp[-c(1:4,18)]  
x <- scale(tmp)  
faces(x, scale=F)



## effect of variables:  
## modified item Var   
## "height of face " "crunchy"   
## "width of face " "creamy"   
## "structure of face" "sweet"   
## "height of mouth " "chocolaty"   
## "width of mouth " "healthful"   
## "smiling " "calorie"   
## "height of eyes " "rich"   
## "width of eyes " "addiction"   
## "height of hair " "accessible"  
## "width of hair " "handy"   
## "style of hair " "wrapping"   
## "height of nose " "image"   
## "width of nose " "commercial"  
## "width of ear " "crunchy"   
## "height of ear " "creamy"

#  
aggregate(bars, by=list(bars$Occupation),mean, na.rm=F)

## 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  
  
## 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  
  
## 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  
  
## 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  
  
## 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

## Group.1 Person Product crunchy creamy sweet  
## 1 At home for Babys 12.00000 NA 3.700000 3.600000 5.000000  
## 2 I am a student. 27.69231 NA 3.350394 3.227838 4.428772  
## 3 I am an employee. 23.66667 NA 3.398263 3.275100 4.485958  
## 4 I am self-employed. 23.00000 NA 4.323256 3.845349 4.952273  
## chocolaty healthful calorie rich addiction accessible handy  
## 1 3.700000 4.300000 4.500000 4.200000 3.800000 4.700000 5.000000  
## 2 3.899534 1.662249 4.316887 3.172405 3.185148 3.977002 3.962336  
## 3 4.001679 1.436371 4.395041 3.794356 3.249507 4.081940 4.121096  
## 4 4.593182 1.835040 4.904651 4.071053 3.553830 4.614359 3.900000  
## wrapping image commercial Gender Age Occupation Marital Status  
## 1 5.000000 4.900000 4.400000 NA 25.00000 NA NA  
## 2 3.619784 3.792769 3.708021 NA 24.30769 NA NA  
## 3 3.535607 3.848082 3.699395 NA 27.28571 NA NA  
## 4 3.448232 4.088254 4.225279 NA 24.00000 NA NA  
## Children City States Sport Sport\_Num ChocoFrequency ChocoFrequency\_Num  
## 1 NA NA NA NA 2.000000 NA 5.000000  
## 2 NA NA NA NA 3.115385 NA 4.038462  
## 3 NA NA NA NA 3.190476 NA 4.000000  
## 4 NA NA NA NA 2.500000 NA 5.000000  
## kmeans  
## 1 1.000000  
## 2 1.846154  
## 3 1.380952  
## 4 2.500000

aggregate(brand, by=list(socio$Occupation),mean, na.rm=F) # 学生最爱kr, 上班族最爱snicker

## Group.1 Person Snickers (b1) KinderBueno (b2) Twix (b3)  
## 1 At home for Babys 12.00000 7.000000 7.000000 7.000000  
## 2 I am a student. 27.69231 5.192308 5.423077 4.961538  
## 3 I am an employee. 23.66667 5.761905 5.142857 5.190476  
## 4 I am self-employed. 23.00000 7.000000 7.000000 7.000000  
## Mars (b4) KitKat (b5) Bounty (b6) Kinderriegel (b7) BalistoKornMix(b8)  
## 1 7.000000 6.000000 6.000000 7.000000 3.000000  
## 2 3.461538 5.038462 4.000000 5.884615 4.115385  
## 3 4.238095 5.000000 3.619048 5.666667 4.428571  
## 4 4.500000 5.000000 6.000000 7.000000 6.500000  
## Lion (b9) Duplo (bb) kmeans  
## 1 4.000000 7.000000 1.000000  
## 2 4.192308 4.307692 1.846154  
## 3 4.714286 4.761905 1.380952  
## 4 6.500000 6.000000 2.500000

aggregate(brand, by=list(socio$Gender),mean, na.rm=F) # 男女最爱kr，

## Group.1 Person Snickers (b1) KinderBueno (b2) Twix (b3) Mars (b4)  
## 1 female 23.48276 5.551724 5.793103 4.896552 3.620690  
## 2 male 28.28571 5.523810 4.857143 5.571429 4.285714  
## KitKat (b5) Bounty (b6) Kinderriegel (b7) BalistoKornMix(b8) Lion (b9)  
## 1 4.793103 3.827586 6.000000 4.206897 4.206897  
## 2 5.380952 4.142857 5.666667 4.476190 4.904762  
## Duplo (bb) kmeans  
## 1 4.413793 1.586207  
## 2 4.904762 1.761905

## remove the respondents with 0 rating

attribute.rating<-attribute.rating[!(attribute.rating$kmeans==3),]