# Exercise 6

Riccardo Petrella

2023-11-22

These are the packages to be installed and loaded in order to run the code.

```
library(fpc)
library(smacof)
library(pdfCluster)
library(cluster)
library(reshape2)
library(prabclus)
library(teigen)
library(psych)
library(dbscan)
library(clusterSim)
library(mixsmsn)
library(DescTools)
library(flexmix)
library(stringdist)
library(nomclust)
```

## 1

**Cluster these data using Gaussian mixtures, t-mixtures, skew-normal mixtures, and skew-t mixtures (using smsn.search you will have to decide what number of degrees of freedom to use for the t-distributions), and decide which clustering you find most convincing, with reasons.**

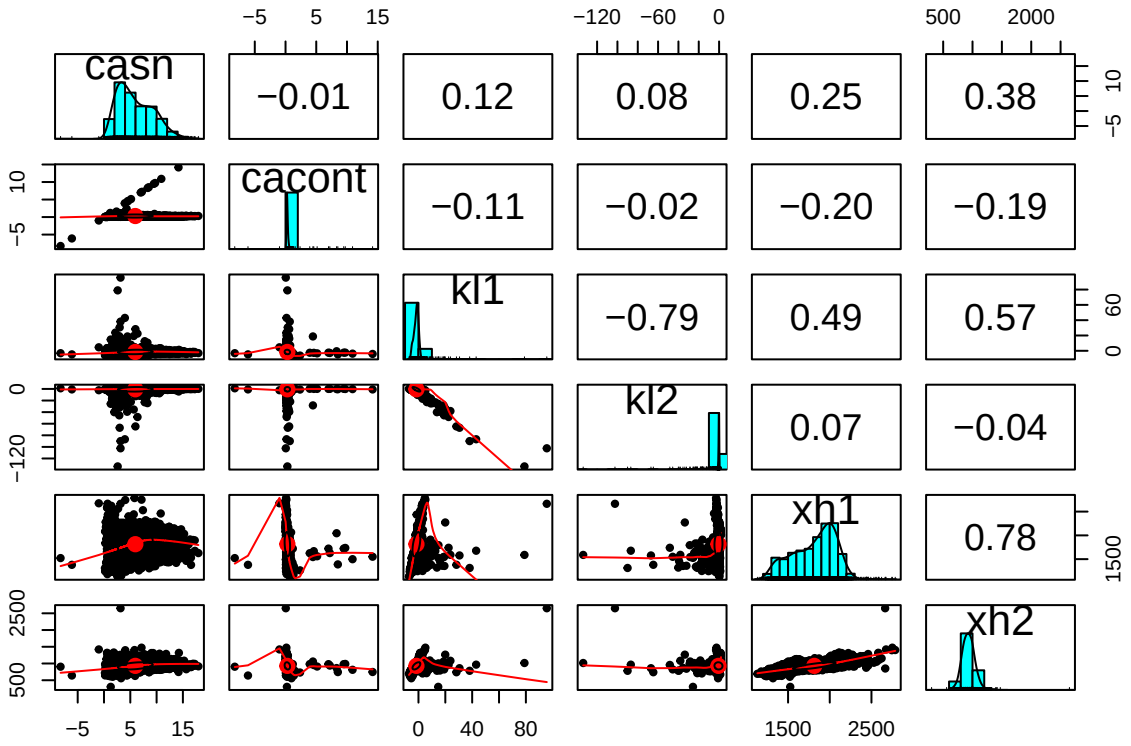
Three clustering methods have been used on this dataset with $K = 1, ..., 5$ (K is smaller this time due to the increased computational demand required to accomplish clustering on a 5000-observations dataset). Respectively the methods are:

- Gaussian Mixture Model
- Mixture of skewed and Heavy tailed distributions clustering
- Scale Mixture of Skew-Normal/t clustering

```
stars5000 <- read.table("stars5000.dat",header=TRUE)


#stars_st <- data.Normalization(stars5000, type = "n2")
pairs.panels(stars5000)
```

**Important Remark: after some trials, I have come to the conclusion that it is better to not standardize the data, despite the professor's advice to do so. The main reason why I have chosen to not scale the data is because if I do it, the smsn function returns an error everytime I try it (the same error discussed in the forum on Virtuale). Moreover, the clustering methods used in this exercise yield very similar clusters with and without standardization. For the sake of experimentation, in the code below there is written as comment the standardized dataset used for the comparison: the standardization made with the Median and MAD (i.e. type "n2" in data.normalization).**




```
pc_stars <- princomp(stars5000)
```

From the "pairs.panels" function, we notice thatthe distributions of "casn", "kl1","kl2", and "xh1" have very skewed histograms.

"cacont" has small variance with a high density peak on just one rectangle. The remaining points creates two wide tails (particularly the right one). However, these points are only a few and they can be considered outliers. "kl1" and "kl2" have very long tails, the former to the right and the latter to the left.

"casn" and "xh1" seem to have two modes and "xh2" might be t-distributed, since there are a couple of outliers on both the tails.

The linear correlation coefficients are pretty low, meaning that there should not be collinearity.

### GMM

Let us proceed with the Gaussian Mixture Model clustering.

```r
set.seed(1234)
gmm <- Mclust(stars5000,G=1:5)
time_gmm <- system.time(Mclust(stars5000,G=1:5))
time_gmm
```

**Remark: this method is evaluated with maximum BIC (the higher the better).**

```
##    user  system elapsed
##    8.76    0.07    9.76
```

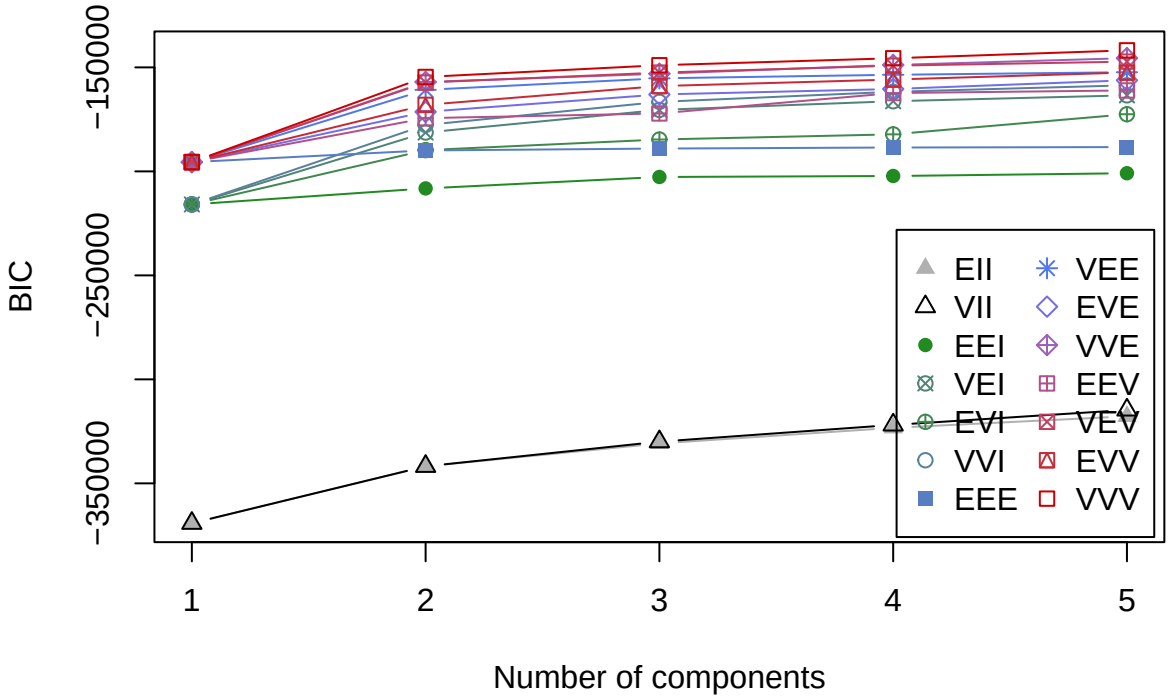
```r
summary(gmm)
```

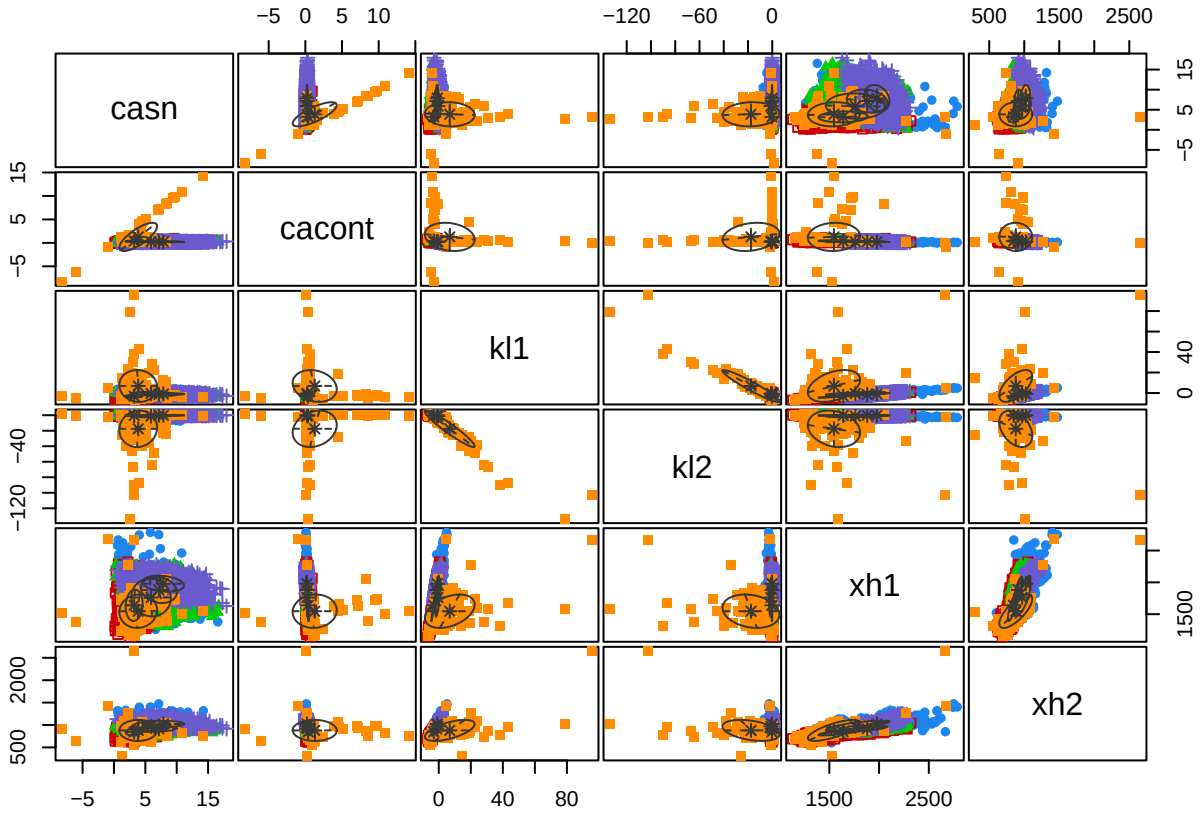
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model with 5
## components:
##
##  log-likelihood    n  df       BIC     ICL
##        -70311.4 5000 139 -141806.7 -142567
##
## Clustering table:
##    1    2    3    4    5
##  530 1520 1481 1380   89
```

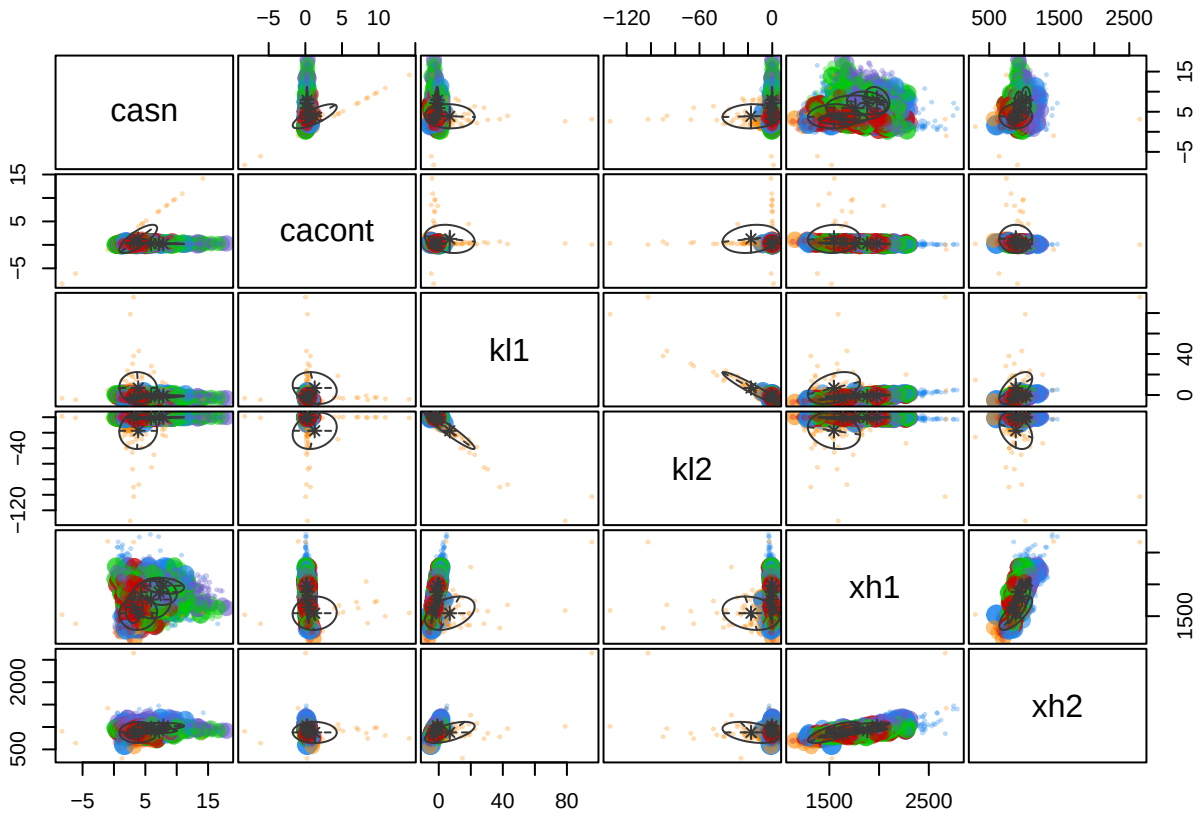
```r
summary(gmm$BIC)
```

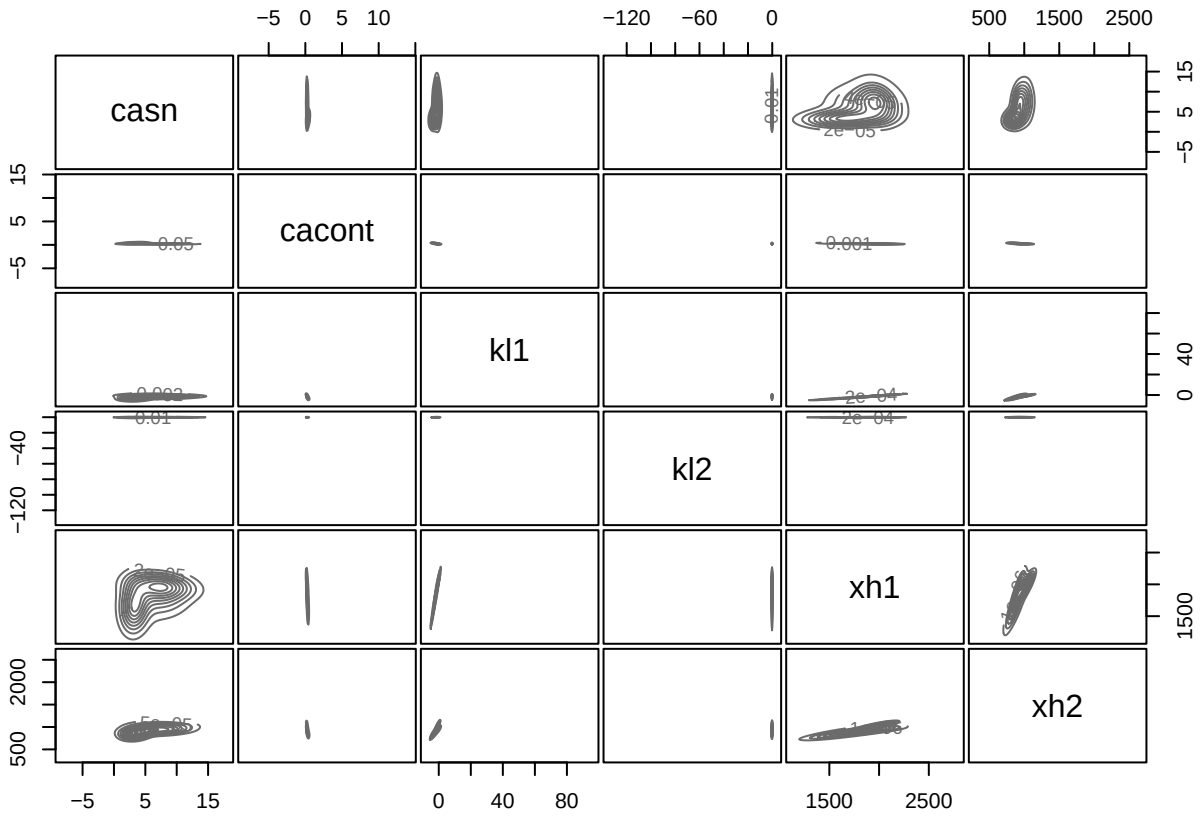
```
## Best BIC values:
##              VVV,5       VVE,5       VVV,4
## BIC      -141806.7 -145554.368 -145609.278
## BIC diff       0.0   -3747.682   -3802.592
```

```r
plot(gmm)
```

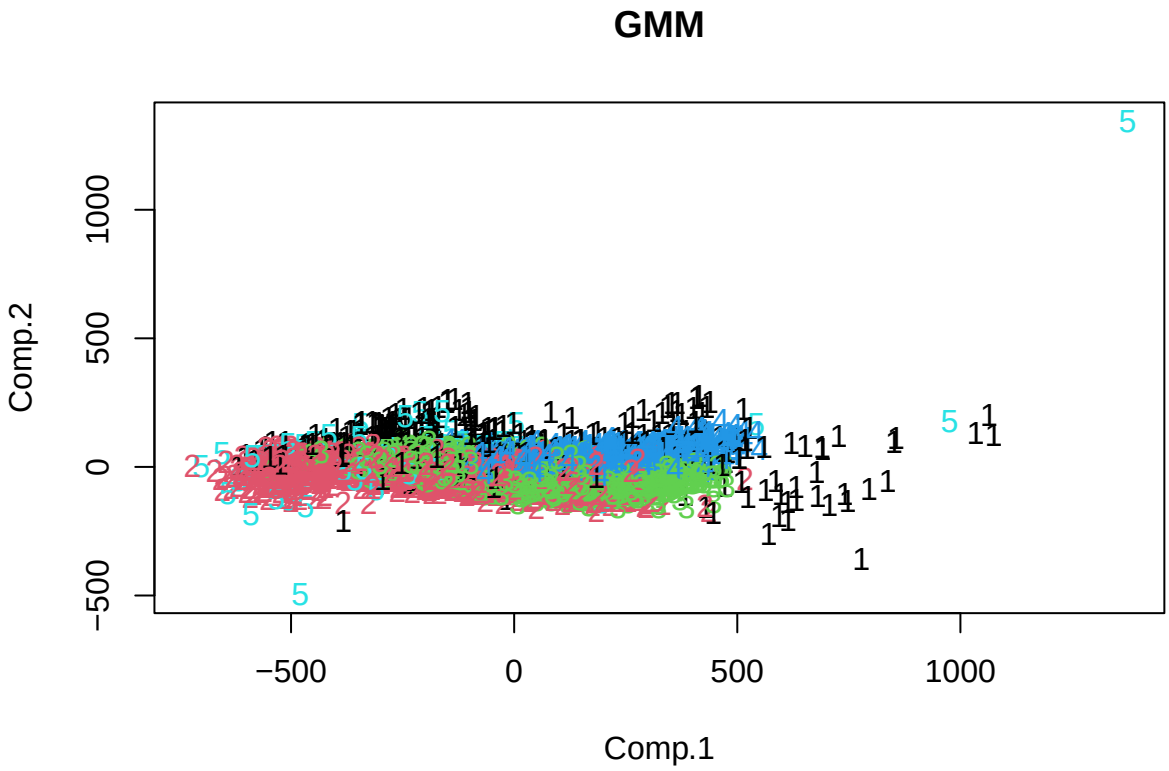

BIC
−150000
−250000
−350000
1
2
3
4
5
Number of components
EII
VII
EEI
VEI
EVI
VVI
EEE
VEE
EVE
VVE
EEV
VEV
EVV
VVV


casn
cacont
kl1
kl2
xh1
xh2
−5 0 5 10
−120 −60 0
500 1500 2500
−5 5 15
0 40
−40 −120
1500
2000 500
−5 5 15
0 40 80
1500 2500


casn
cacont
kl1
kl2
xh1
xh2
−5 0 5 10
−120 −60 0
500 1500 2500
−5 5 15
0 40
−40 −120
1500
2000 500
−5 5 15
0 40 80
1500 2500




```
plot(pc_stars$scores, col=gmm$classification, pch=clusym[gmm$classification])
title(main="GMM")
```




The best chosen model is VVV (ellipsoidal, varying volume, shape, and orientation) with 5 components.

Sadly the plots are not very informative. In both the paired scatterplots, the orange cluster has many sparse units while the others almost overlap with each other and the peaks of density are very close.

If we look at the plot of the level curves, in some paired plots 2 peaks are visible, but never 5. Thus, the 5 clusters are not easily determined with the naked eye.

The plot made with the PCA is not that different. However, we can see that the clusters occupy sligthly differnt positions on the cartesian chart. For instance, cluster number 1 is the most North-East one, cluster number 2 is the most South-West one, while cluster number 5 has the most sparse points and clusters number 3 and 4 are in the middle.

### Mixture of skew-heavy-tailed distribution

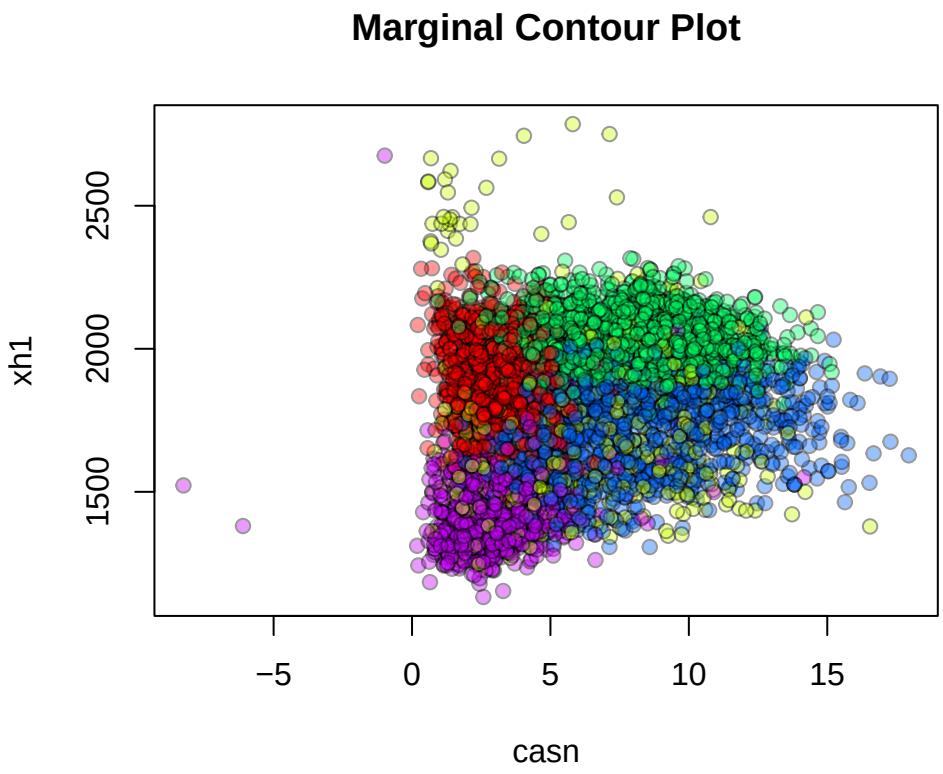
We move on with the second method:

```
set.seed(2423423)
t_stars <- teigen(stars5000, Gs=1:5, scale = FALSE)
```

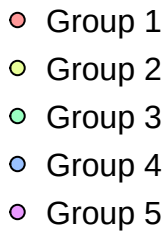
```
t_stars$bestmodel
```

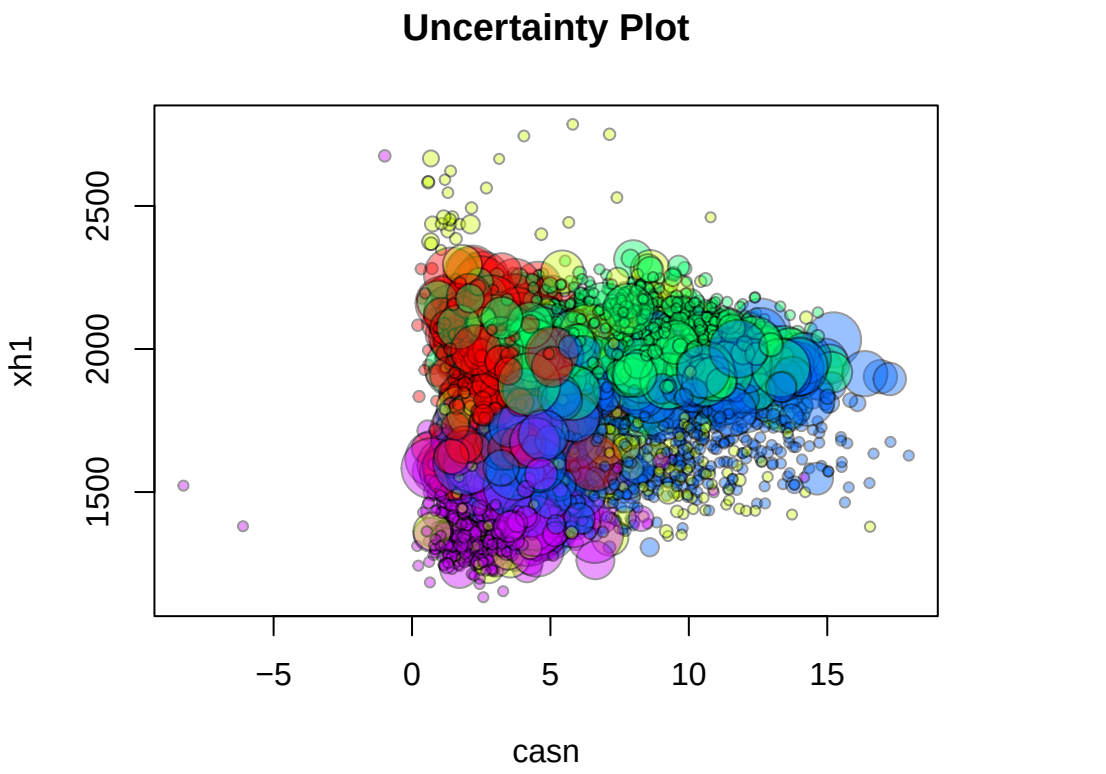
```
## [1] "The best model (BIC of -142724.98) is UUUU with G=5"
```

```
plot(t_stars,xmarg=1,ymarg=5,what="contour")
```

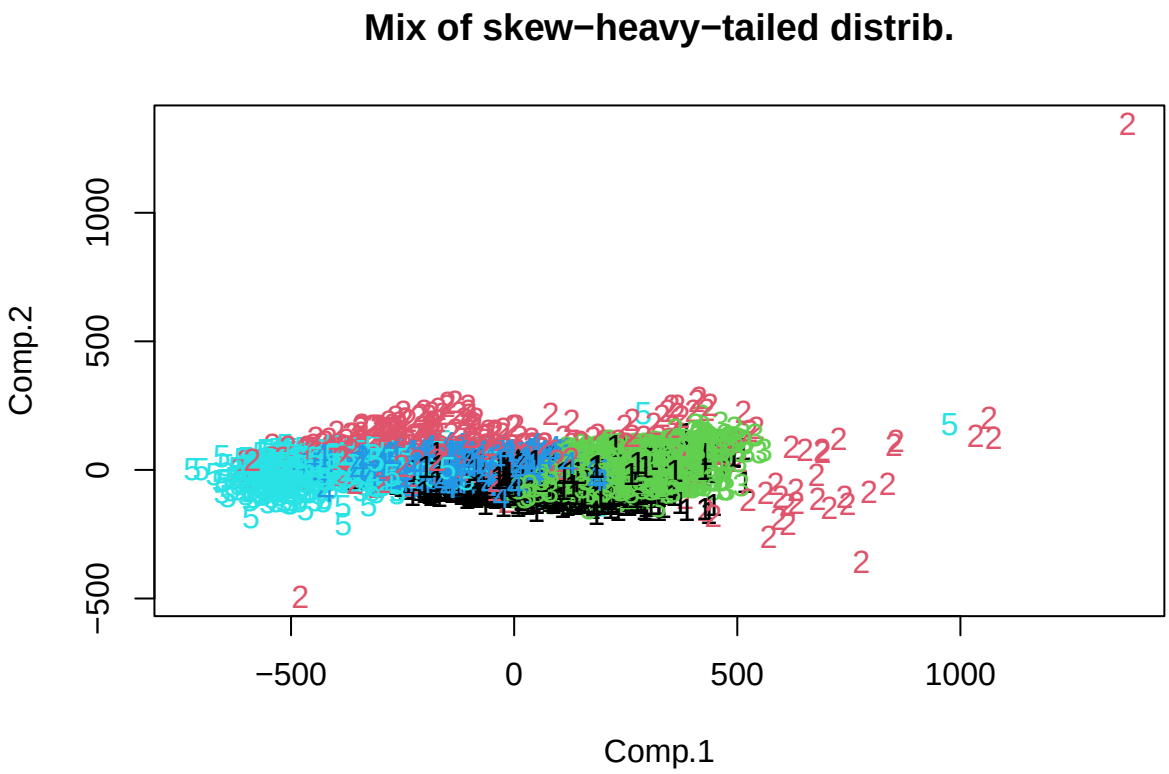



```
plot(t_stars,xmarg=1,ymarg=5,what="uncertainty")
```






```
# The plot command for teigen allows for two kinds of plots.
# Both are done on two variables, and xmarg, ymarg specify
# the numbers of the variables used.
plot(pc_stars$scores,col=t_stars$classification,pch=clusym[t_stars$classification])
title(main="Mix of skew-heavy-tailed distrib.")
```




Interestingly, we notice from both the marginal contour plot and the uncertainty plot, that the clusters are well distinguished from one to another, despite not being clearly detached.

When we observe the PC plot, we notice the same thing. Clusters number 1,3 and 4 are partially overlapped and are disposed in the middle of the other clusters, while cluster number 2 is the most sparse being the Northern-most and the Southern-most one at the same time. They all are disposed along the horizontal line equal to 0.

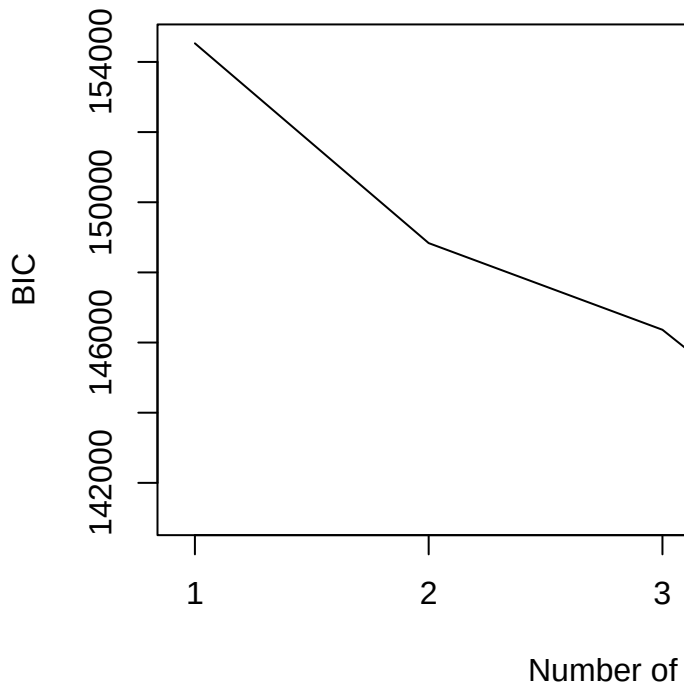
### Scale Mixture of Skew-Normal/t clustering

The last clustering method is as follows:

**Remark: nu=2 has been chosen to avoid the error of the singular matrix**

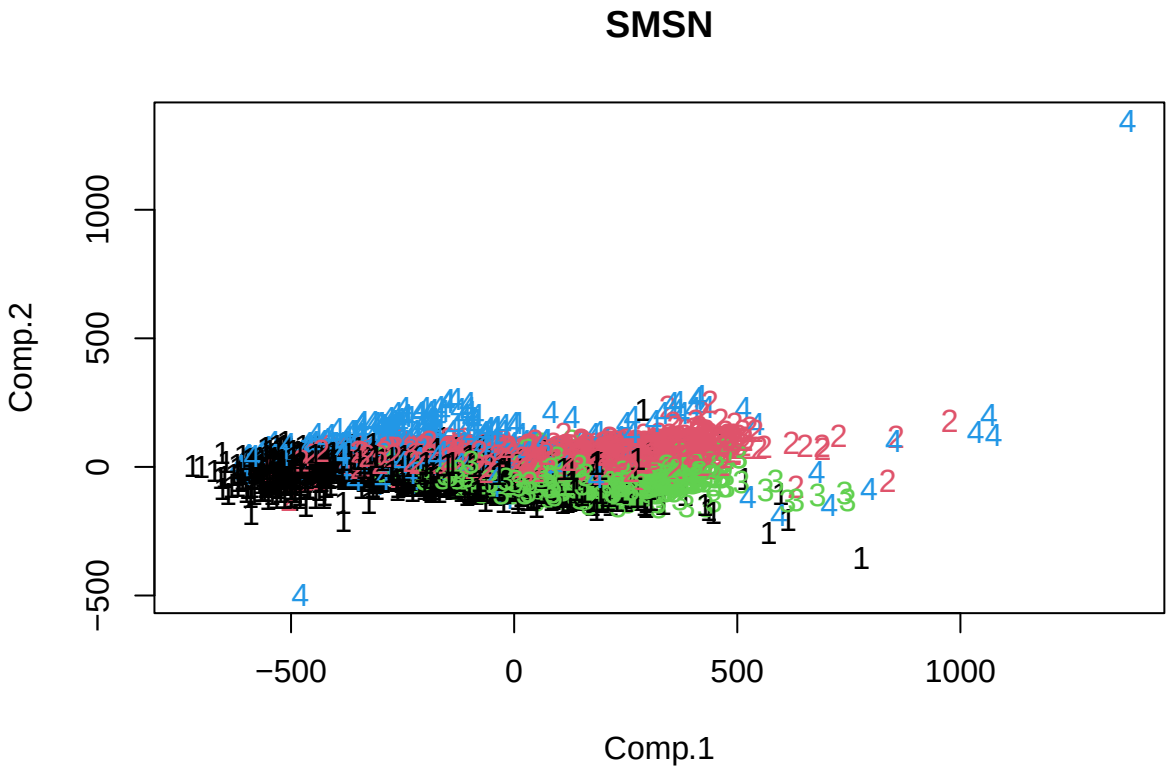
```
set.seed(67543)
smsn_stars <- smsn.search(stars5000, nu=2,g.min=1,g.max=5,family="Skew.t")
#smsn_stars$best.model

plot(1:5,smsn_stars$criteria,type="l",ylab="BIC",xlab="Number of clusters")
```




**Remark 2: here the lower the BIC, the better the model**

```
plot(pc_stars$scores,col=smsn_stars$best.model$group,pch=clusym[smsn_stars$best.model$group])
title(main="SMSN")
```




As we can see, the best model has $K = 4$ and the PC plot is very similar to the previous ones. It is easy to distinguish the groups but they all are very close and horizontally distributed along PC2=0.

## 2

**Implement the big data method explained above, and apply it to the stars5000 data from Exercise 1 above. Use ns = 1000 (for a data set of size 5000, using ns = 2000 will not win that much time). Take the time (this can be done using the system.time command in R). Also take the time for running Mclust on those data. Compare the running times and the results of the big data method and of standard Mclust (it would be a good result for the big data method if results are very similar, but the run time is much faster).**

The chunk of code below serves the purpose of computing the subset of 1000 random units from stars5000 and apply Mclust on them.

```
set.seed(565564)
stars1000 <- stars5000[sample(5000,1000),]


set.seed(1234)
gmm2 <- Mclust(stars1000,G=1:5)

time_gmm2 <- system.time(Mclust(stars1000,G=1:5))
time_gmm2
```

```
##    user  system elapsed
##    1.57    0.00    1.56
```

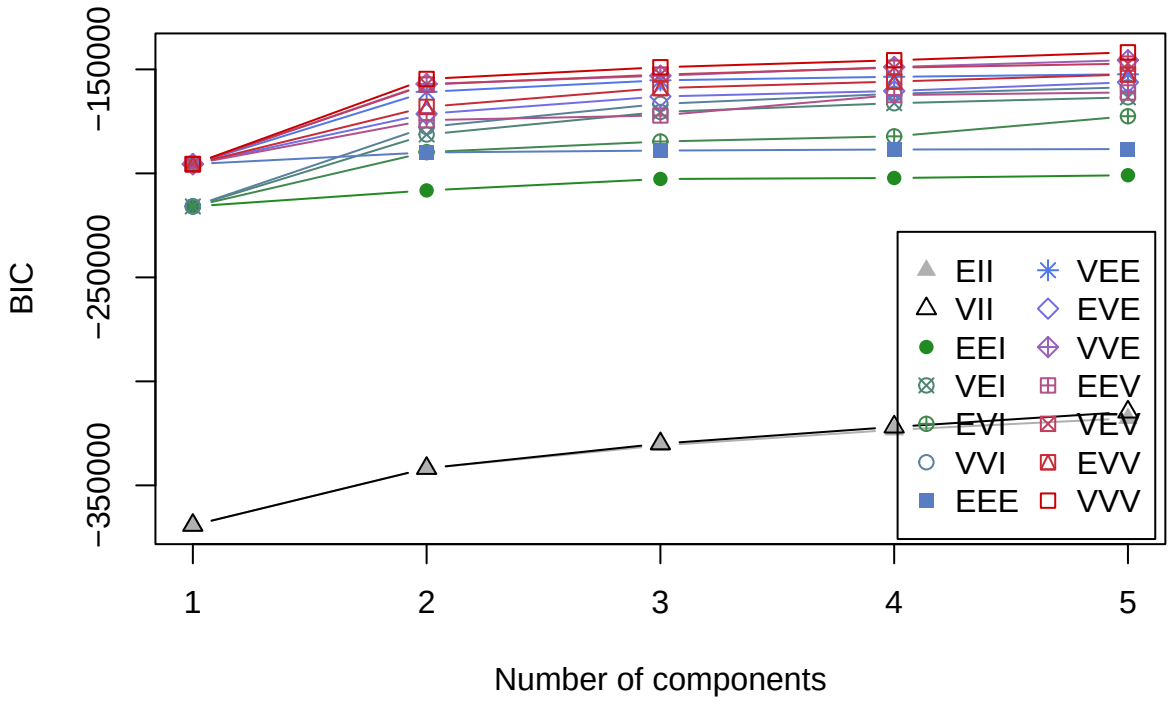
```
summary(gmm2)
```

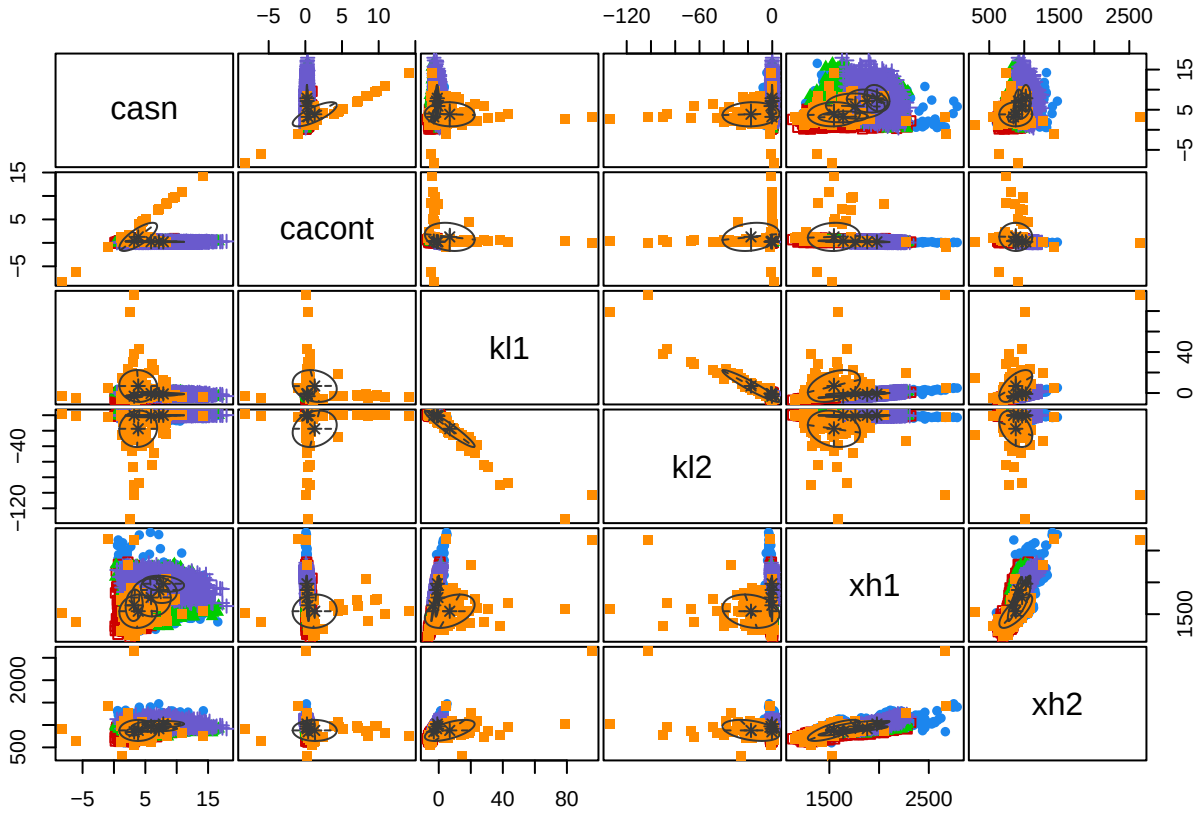
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVE (ellipsoidal, equal orientation) model with 5 components:
##
##  log-likelihood    n df       BIC       ICL
##       -14317.53 1000 79 -29180.78 -29281.03
##
## Clustering table:
##   1   2   3   4   5
## 371 204  98 297  30
```

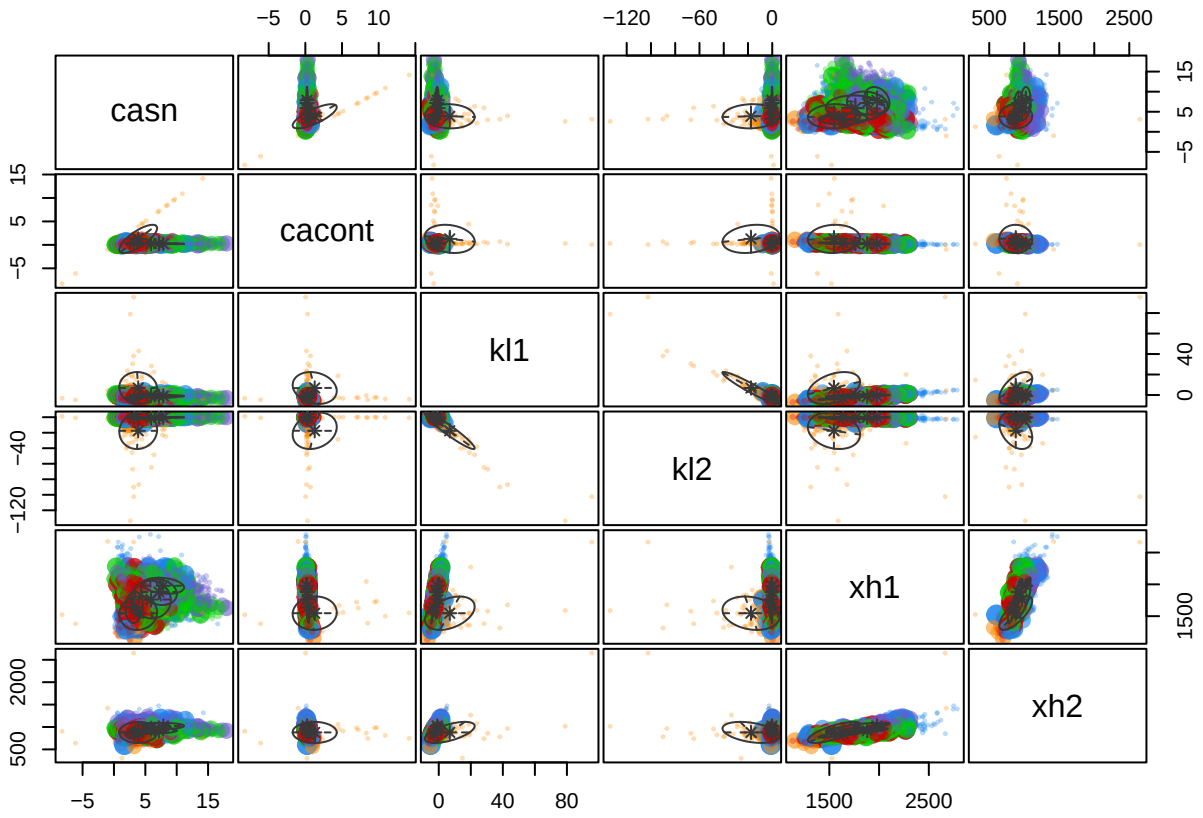
```
summary(gmm2$BIC)
```

```
## Best BIC values:
##              VVE,5       VVV,5       VVV,4
## BIC      -29180.78 -29344.6989 -29485.1599
## BIC diff      0.00   -163.9225   -304.3836
```

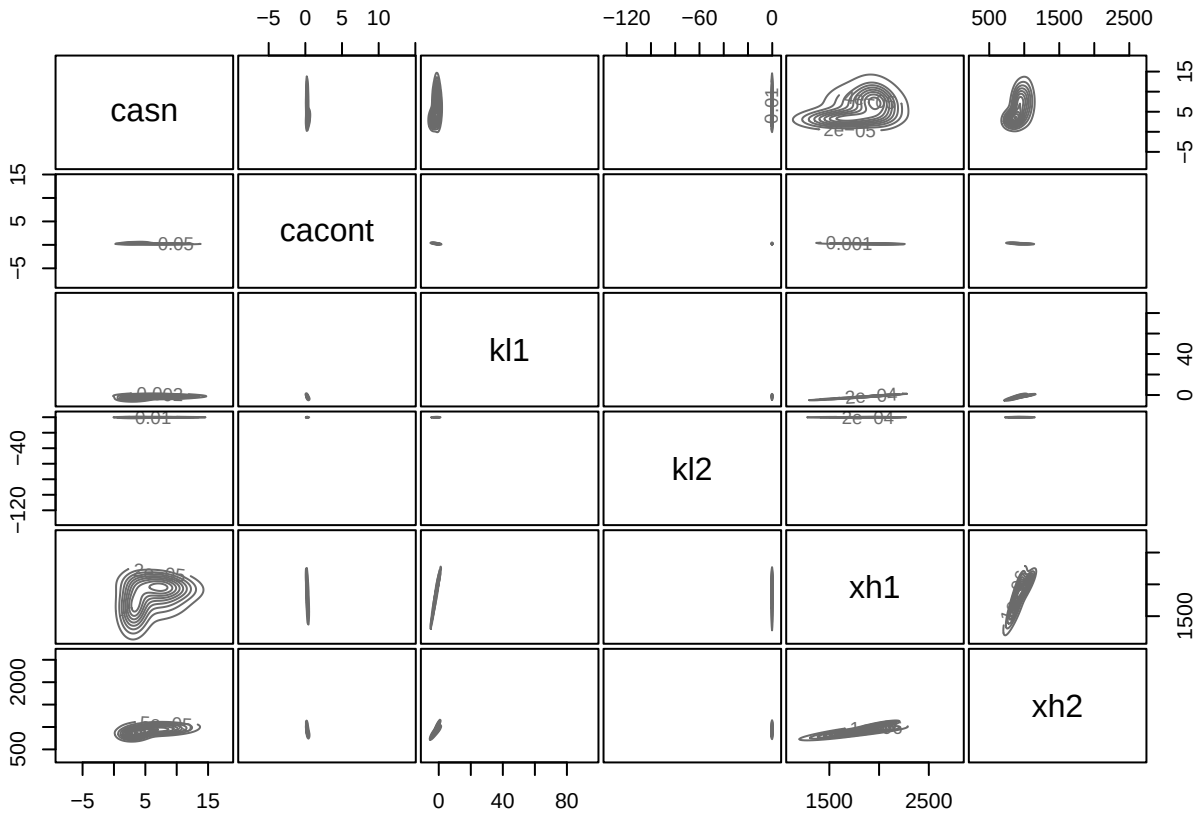
```
plot(gmm)
```



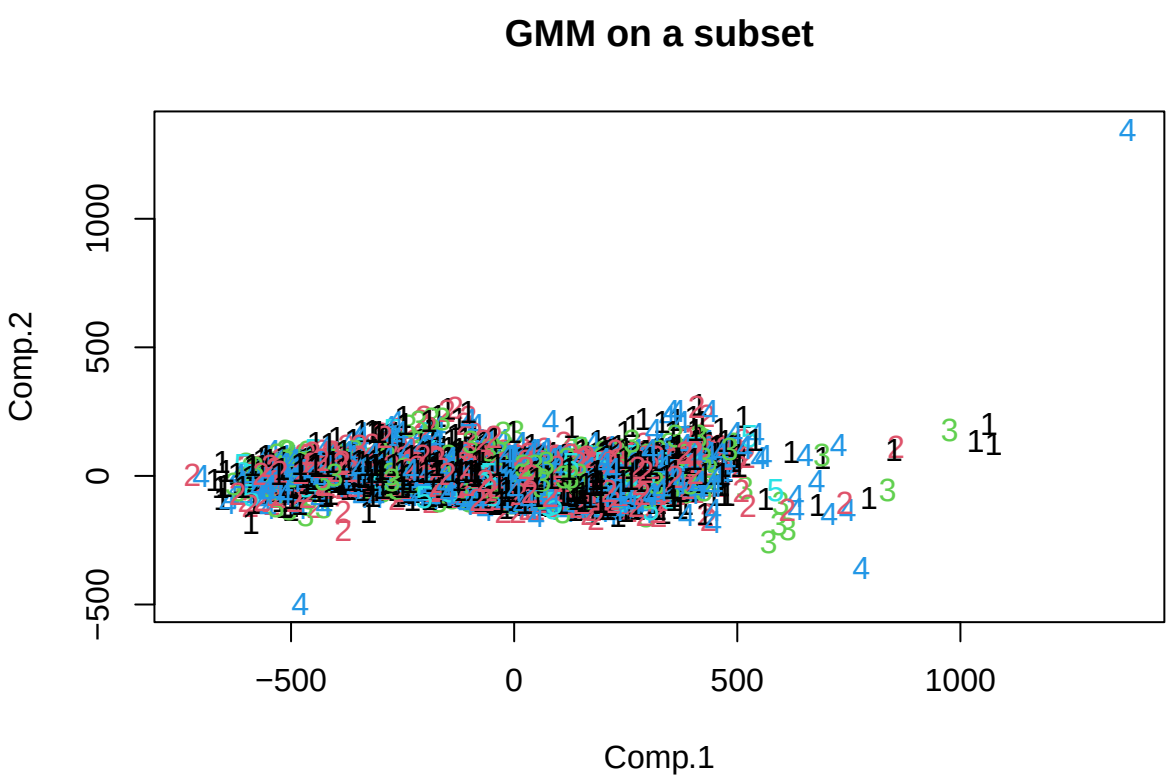

casn
cacont
kl1
kl2
xh1
xh2
−5 0 5 10
−120 −60 0
500 1500 2500
−5 5 15
−40 0 40
1500 2500
500 2000
−5 5 15
0 40 80




```
legend(10, -1000, 0.3, legend = "", cex = 0.3)
```




```
plot(pc_stars$scores, col=gmm2$classification, pch=clusym[gmm2$classification])
title(main="GMM on a subset")
```




The best selected model VVE (ellipsoidal, equal orientation) model with 5 clusters.

Unfortunately, the GMM clustering on the dataset seems to not work properly, because the clusters look completely undistinguishable.

We now apply the predict.Mclust function to extend the clusterization to the full dataset.

```
set.seed(39835)
pred <- predict.Mclust(gmm2, newdata = stars5000)
time_pred_gmm2 <- system.time(predict.Mclust(gmm2, newdata = stars5000))

time_pred_gmm2
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
adj.rand.index(gmm$classification ,pred$classification)
```

```
## [1] 0.5801302
```

Here we can compute the total amount of time spent on the GMM applied on the subset. It consists of $time_gmm2 + time_pred_gmm2 = 1.49 + 0.01 = 1.50$

This time is much smaller than $time_gmm = 6.71$, meaning that we managed to get a reduction of 77% of the time circa. Moreover, the Adjusted Rand Index reveals that the prediction is pretty decent with a 58% of similarity with respect to the GMM applied on the full model.

## 3

In a situation with 10 variables and 4 mixture components, what is the number of free parameters for:

- (a) a "VVV" Gaussian mixture model assuming fully flexible covariance matrices

**Answer:** The number of parameters is:

Means (**a**): For each of the 10 variables in each of the 4 mixture components, there are 10 means. So, $10 \times 4 = 40$ parameters.

Covariance matrices ($\sum$): For each of the 10 variables, there is a fully flexible covariance matrix in each of the 4 mixture components. The number of parameters for a fully flexible d-dimensional covariance matrix is $d \times (d+1)/2$. So, for 10 variables, it is $10 \times (10+1)/2 = 55$ parameters for each component. Therefore, $55 \times 4 = 220$ parameters.

Mixture weights ($\pi$): There are 4 mixture components, so there are 3 free parameters for the mixture weights (assuming they sum to 1, and the fourth can be determined).

Now, summing these up:

$$40 + 220 + 3 = 263$$

- (b) a "VII" Gaussian mixture model assuming spherical covariance matrices with potentially differing volumes

**Answer:** The number of parameters is computed as follows:

Means (**a**): For each of the 10 variables in each of the 4 mixture components, there are 10 means. So, $10 \times 4 = 40$ parameters.

Shared Covariance Matrix ($\sum$): There is one shared covariance matrix for all components, and it is spherical (isotropic). For a spherical covariance matrix, there is a single variance parameter for each dimension, so in total, there is only 1 variance as parameter for 10 variables. However, there are 4 mixtures and then we get $1 \times 4 = 4$.

Mixture weights ($\pi$): There are 4 mixture components, so there are 3 free parameters for the mixture weights (assuming they sum to 1, and the fourth can be determined).

Now, summing these up:

$$40 + 4 + 3 = 47$$

- (c) a fully flexible skew-normal mixture,

**Answer:** In this model the parameters are obtained as shown below:

Vector of mean **a**: that is $10 \times 4 = 40$ mixture components.

Covariance matrix $\sum$: as done before, we have $10 \times (10+1)/2 = 55$ $55 \times 4 = 220$

The Skewness parameter $\lambda$: $10 \times 4 = 40$.

Finally:

$$40 + 220 + 40 + 3 = 303$$

- (d) a fully flexible mixture of multivariate t distributions

**Answer:** Means **a**: For each of the 10 variables in each of the 4 mixture components, there are 10 means. So, 10×4=40 parameters.

Covariance matrices ($\sum$): For each of the 10 variables, there is a fully flexible covariance matrix in each of the 4 mixture components. The number of parameters for a fully flexible d-dimensional covariance matrix is

$d \times (d+1)/2$. So, for 10 variables, it is $10 \times (10+1)/2 = 55$ parameters for each component. Therefore, $55 \times 4 = 220$ parameters.

Degrees of freedom $\nu$: For each of the 4 mixture components, there is a single degrees of freedom parameter (i.e 4 of them).

Mixture weights ($\pi$): There are 4 mixture components, so there are 3 free parameters for the mixture weights (assuming they sum to 1, and the fourth can be determined).

Now, summing these up:

$$40 + 220 + 4 + 3 = 267$$

- (e) a mixture of skew-t distributions where skewness parameters, degrees of freedom and $\sum$-matrices are assumed equal in all mixture components?

**Answer:** Means **a**: For each of the 10 variables in each of the 4 mixture components, there are 10 means. So, $10 \times 4 = 40$ parameters.

Degrees of freedom $\nu$: There is a single degrees of freedom parameter for the entire mixture (i.e we have totally 4 df).

Skewness $\lambda$: There is a single skewness parameter for the entire mixture (i.e. 4 of them).

Covariance matrix $\sum$: For each of the 10 variables, there is a covariance matrix in each of the 4 mixture components. The number of parameters for a fully flexible d-dimensional covariance matrix is $10\times(10+1)/2 = 55$ parameters for the entire mixture.

Mixture weights $\pi$: There are 4 mixture components, so there are 3 free parameters for the mixture weights (assuming they sum to 1).

Now, summing these up:

$$40 + 1 + 10 + 55 + 3 = 109$$

**4**

**Task:** Estimate a latent class mixture model using the flexmixedruns-function, including estimating the number of clusters (it is enough to go up to 5). Compare the results with the truth that you know about these data (this can concern the clustering vector, but also the estimated parameters). Also compute the simple matching distance and cluster the data using any distance based clustering method (just one), and compare the clustering with the same number of clusters with the optimal clustering from flexmixedruns. You should find that Average Linkage with Simple Matching is somewhat problematic here. Why is this not a good method for these data?

The code below is copied from the task:

```
prodcat <- c("PR","MB","AB","N") # product categories
eventcat <- c("S","M","C","P","N") # event categories
prodp <- eventp <- list()
# Category probabilities within the two mixture components (clusters)
```

```
prodp[[1]] <- c(0.4,0.4,0.1,0.1)
prodp[[2]] <- c(0.2,0.1,0.4,0.3)
eventp[[1]] <- c(0.5,0.2,0.1,0.1,0.1)
eventp[[2]] <- c(0.1,0.1,0.1,0.3,0.4)

# The first 400 observations from component 1, then 600 from component 2:
n1 <- 400
n2 <- 600
n <- n1+n2
# Initialisation (provide a data frame of the correct type and size
# without already having the data):
consumers <- data.frame(prod=factor(c(prodcat,rep(NA,n-4)),levels=prodcat),
event=factor(c(eventcat,rep(NA,n-5)),levels=eventcat))
# Generation of the data; you may want to set a seed here.
consumers$prod[1:n1] <-  sample(prodcat,n1,replace=TRUE,prob=prodp[[1]])

consumers$event[1:n1] <- sample(eventcat,n1,replace=TRUE,prob=eventp[[1]])

consumers$prod[(n1+1):n] <- sample(prodcat,n2,replace=TRUE,prob=prodp[[2]])

consumers$event[(n1+1):n] <- sample(eventcat,n2,replace=TRUE,prob=eventp[[2]])
```

The table below shows the number of observations in each categorical variable:

```
table(consumers)
```

```
##     event
## prod   S   M   C   P   N
##   PR  97  47  34  54  63
##   MB  86  26  25  36  34
##   AB  38  24  36  81 108
##   N   37  22  27  56  69
```

Finally we apply the mixture model for categorical data:

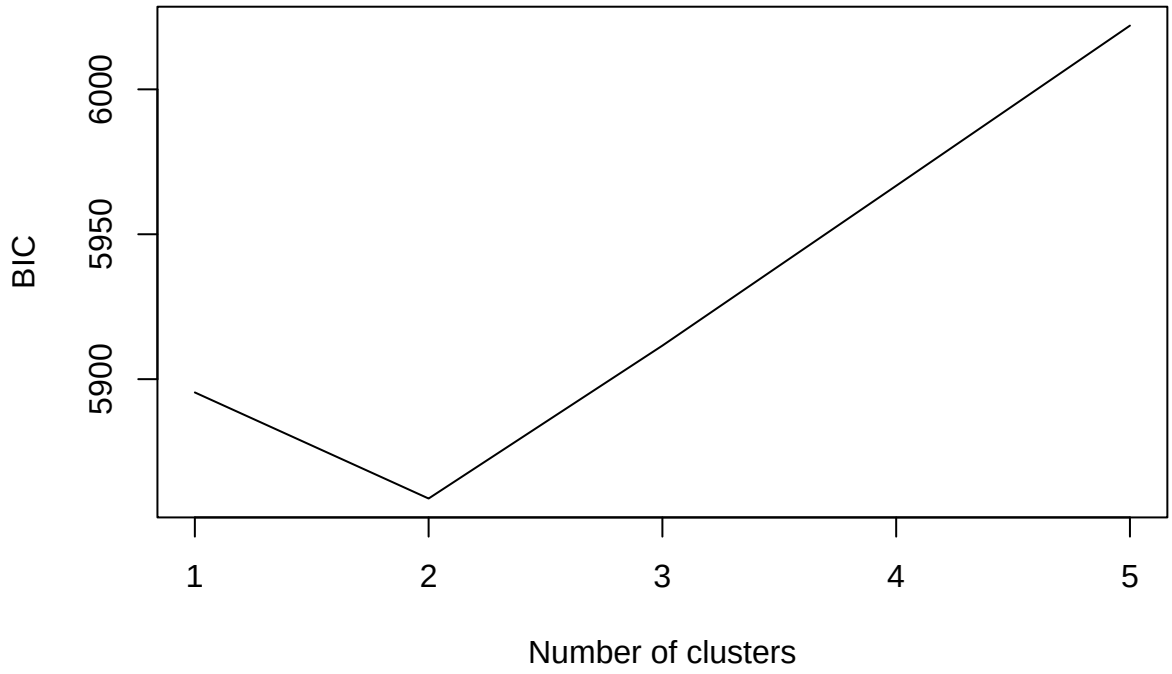
```
set.seed(887766)
flex_cons <-
flexmixedruns(consumers,continuous=0,discrete=2,n.cluster=1:5)
```

```
## k=  1  new best fit found in run  1
## k=  1  BIC=  5895.426
## k=  2  new best fit found in run  1
## k=  2  new best fit found in run  2
## k=  2  new best fit found in run  3
## k=  2  new best fit found in run  4
## Nonoptimal or repeated fit found in run  5
## k=  2  new best fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## k=  2  new best fit found in run  10
## Nonoptimal or repeated fit found in run  11
```

```
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  2  BIC=  5858.823
## k=  3  new best fit found in run  1
## k=  3  new best fit found in run  2
## k=  3  new best fit found in run  3
## k=  3  new best fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## k=  3  new best fit found in run  18
## k=  3  new best fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  3  BIC=  5911.577
## k=  4  new best fit found in run  1
## k=  4  new best fit found in run  2
## Nonoptimal or repeated fit found in run  3
## Nonoptimal or repeated fit found in run  4
## k=  4  new best fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## k=  4  new best fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  4  BIC=  5966.732
## k=  5  new best fit found in run  1
## Nonoptimal or repeated fit found in run  2
```

```
## k=  5  new best fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## k=  5  new best fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  5  BIC=  6021.991
```

```
plot(1:5,flex_cons$bicvals,typ="l",
xlab="Number of clusters",ylab="BIC")
```




```
flex_cons$flexout[[2]]
```

```
##
```

```
## Call:
## flexmix(formula = x ~ 1, k = k, cluster = initial.cluster, model = lcmixed(continuous = continuous,
##     discrete = discrete, ppdim = ppdim, diagonal = diagonal),
##     control = control)
##
## Cluster sizes:
##   1   2
## 460 540
##
## convergence after 29 iterations
```

**Remark: The lower the BIC the better the model.** $K = 2$ has been chosen as the best model. Let us compare it with the true clusters.

Here below are attached the values of the parameters.

```
true_clusters <- c(rep(1, n1), rep(2, n2))

adj.rand.index(flex_cons$flexout[[2]]@cluster, true_clusters)
```

```
## [1] 0.3355955
```

```
flex_cons$flexout[[2]]@components[[2]][[1]]@parameters$pp
```

```
## [[1]]
## [1] 0.46385157 0.06854033 0.29199760 0.17561050
##
## [[2]]
## [1] 0.12866469 0.07349001 0.42137822 0.31056457 0.06590251
```

36% is a positive result, since it shows that we get a mild similarity between our model and the true clusters.

The last comparison is with the hierarchical clustering with average linkage adn with the Simple Matching distance:

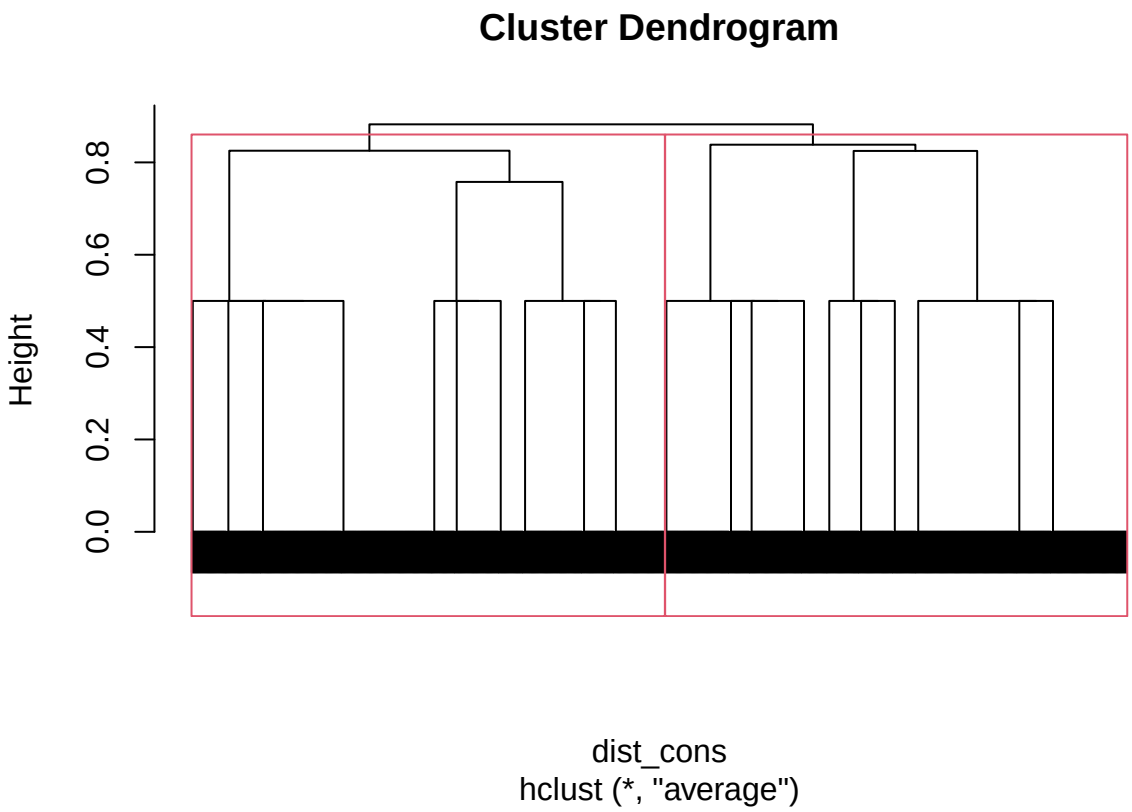
```
dist_cons <- sm(consumers)

hier_cons <- hclust(dist_cons,method="average")

# Plot the dendrogram:

plot(hier_cons, labels = FALSE)

rect.hclust(hier_cons, k = 2)
```




```
hclust_cons <- cutree(hier_cons, k=2)
table(hclust_cons)
```

```
## hclust_cons
##   1   2
## 506 494
```

```
adj.rand.index(flex_cons$flexout[[2]]@cluster, hclust_cons)
```

```
## [1] 0.2780654
```

```
adj.rand.index(hclust_cons, true_clusters)
```

```
## [1] 0.1928198
```

The flexible model for categorical data and the hierarchical clustering used in this exercise do not yield similar result. The similarity is positive but low (25.9%). Furthermore, the similarity between the hierarchical clustering and the true clusters is lower than the similarity between the flexible model and the real groups, meaning that"flexmixedruns" performs better.

The reasons behind this are several. Let us try to make some hypothesis:

combining Average Linkage with Simple Matching Distance may lead to some issues for certain types of datasets:

Sensitivity to Variable Frequencies:

Simple Matching Distance treats all variables (categories) equally. If there are imbalances in the frequencies of categories, this method might emphasize less frequent categories, leading to biased clustering results.

Elongated Clusters:

Average Linkage tends to form elongated clusters, especially when there are outliers or varying densities within the clusters. If your data has clusters with different densities or shapes, Average Linkage may not perform well in capturing these structures. Sensitivity to Noise:

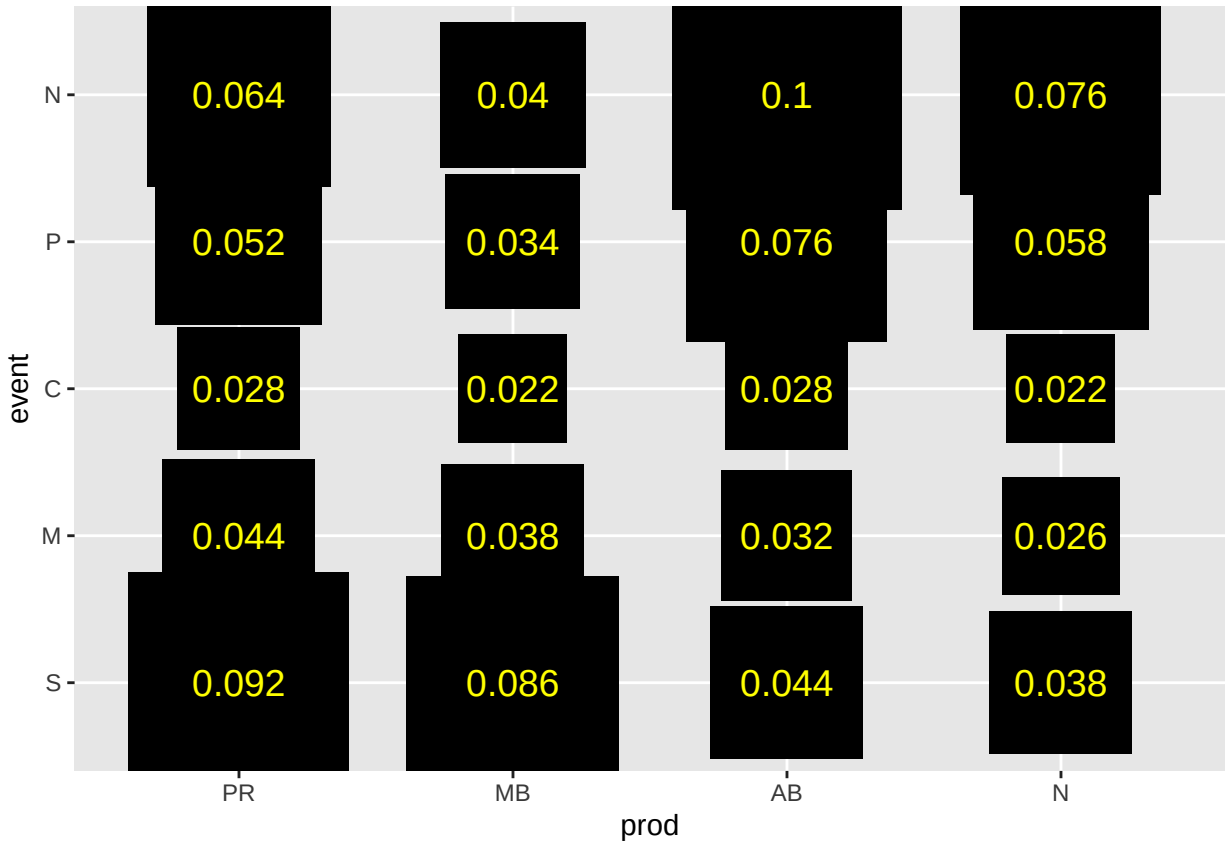
Simple Matching Distance, by counting mismatches without considering the magnitude, can be sensitive to noise or small variations in the data. This sensitivity might be exacerbated when used in conjunction with Average Linkage.

Suboptimal for Certain Data Structures:

The combination of Average Linkage with Simple Matching might not be the best choice for datasets where the underlying structure is not well-suited to the assumptions of these methods.

Here attached below there is an example on how to plot the data of the two mixture components:

```
# This makes use of objects defined above:
library(ggplot2)
pi1 <- n1/n
pi2 <- n2/n
thetable <- c1table <- c2table <- matrix(NA,nrow=4,ncol=5)
for(i in 1:4)
for(j in 1:5){
thetable[i,j] <- pi1*prodp[[1]][i]*eventp[[1]][j]+
pi2*prodp[[2]][i]*eventp[[2]][j]
c1table[i,j] <- prodp[[1]][i]*eventp[[1]][j]
c2table[i,j] <- prodp[[2]][i]*eventp[[2]][j]
}
# thetable is the table for the overall distribution,
# c1table and c2table for the two mixture components/clusters
thetabled <- as.table(thetable*1000) # Transform probabilities to counts here.
attr(thetabled,"dimnames") <- attr(table(consumers),"dimnames")
# This is not needed if you want to visualise the data in comsumers
# directly. attr makes sure that the category names are in the right place.
thetabled <- as.data.frame(thetabled) # ggplot needs a data frame.
ggplot(thetabled)+geom_point(mapping = aes(x = prod, y = event, size = Freq),
shape=15,show.legend=FALSE)+scale_size_area(max_size=40)+
geom_text(aes(x = prod, y = event, label=Freq/1000),color="yellow",size=5)
```


event
N
P
C
M
S
0.064
0.04
0.1
0.076
0.052
0.034
0.076
0.058
0.028
0.022
0.028
0.022
0.044
0.038
0.032
0.026
0.092
0.086
0.044
0.038
PR
MB
AB
N
prod