Exercise 5

Riccardo Petrella

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)
```

1

Task: The task is to cluster the glass splinters into different types of glass. Specifically, these are from crime scenes, and it is of interest how many different types of glass there are, and which splinters belong together.

Find a good clustering, i.e., one of which you think that it captures in the best possible manner a really meaningful grouping. Try out at least two clusterings, of which at least one is based on a Gaussian mixture, and at least one is from a different approach.

Compare the clusterings and comment on how meaningful and useful you think they are. Select one clustering that you prefer. Discuss in particular whether the Gaussian mixture is a good method for these data in your view, and what might be potential problems with it.

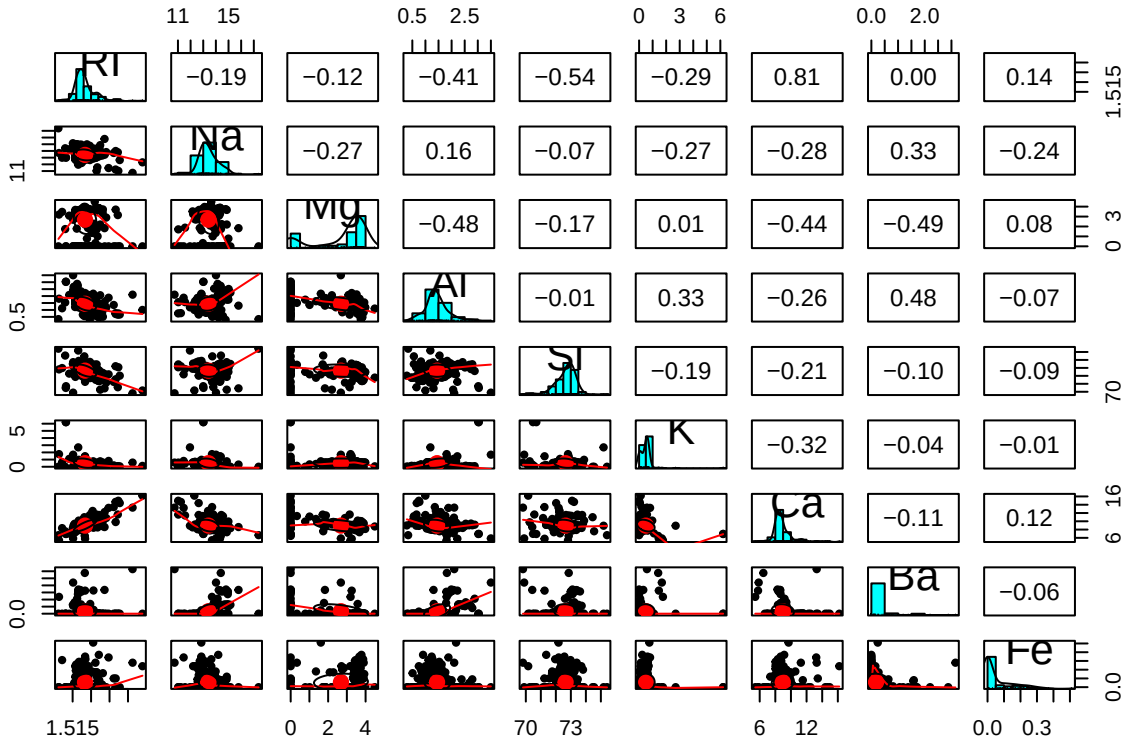
Produce at least one visualisation each for at least two clusterings. Interpret the clusters of the chosen clustering (you can use all given variables and your visualisations).

Comment on other aspects of the data set that you find out as far as you think they could be relevant.

Answer: We upload the dataset amd we convert it into a data matrix.

Remark: since the variables are all expressed as percentages of elements from the periodic table, standardizing the data is not very useful.

```
#setwd("C:/Users/...")
glass <- read.table("glass.dat",header=TRUE)
mglass <- as.matrix(glass)

pairs.panels(mglass)
```




```
# mds_glass <- mds(mglass)
pc_glass <- princomp(mglass)
```

We graphically explore the data using the function "pairs.panels". From it, we can notice that CA and RI are highly correlated (81 %) but the other coefficients of correlation are low and there should not be collinearity in our data. By looking at the histograms, it is easy to see that most of the variables have skewed distributions and MG is clearly bi-modal. Hence, it might make sense to do clustering with "Mixtures of skew and heavy-tailed distributions".

Moreover, we compute the Principal Component Analysis as our chosen method to later plot the data (could have been used MDS as well). Let us test different clustering methods with $K = 1, ..., 10$:

- GMM
- K-Means
- PAM
- Mixtures of skew and heavy-tailed distributions.

GMM:

Let us start with the Gaussian Mixture Model

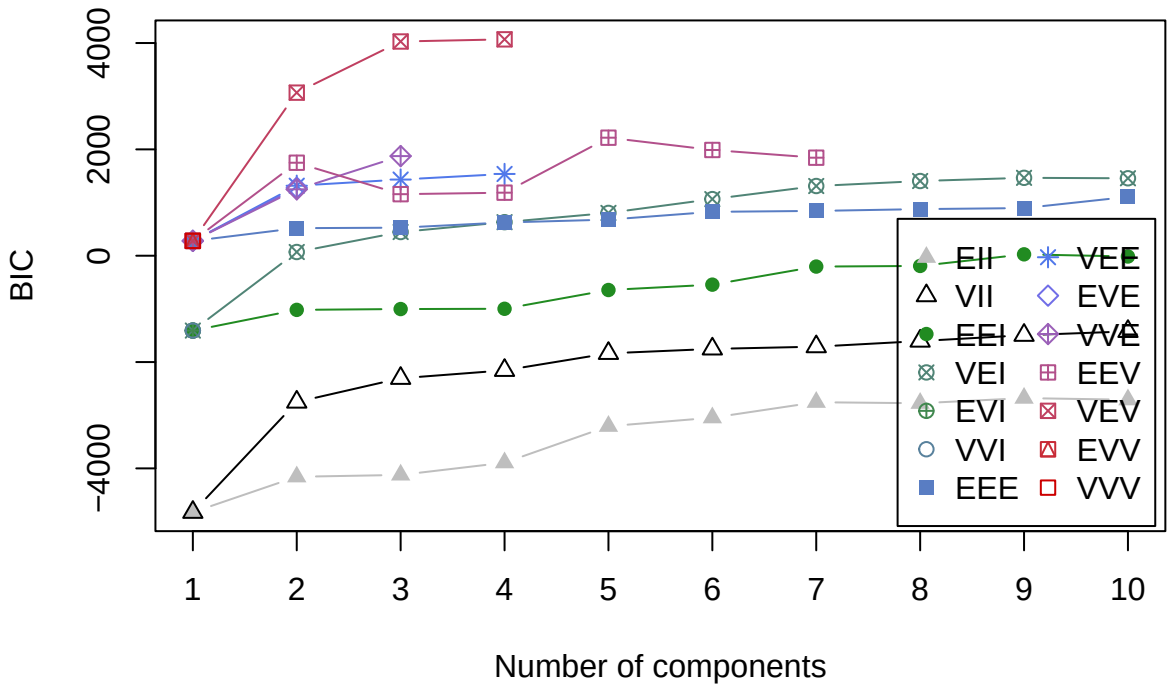
```r
set.seed(1234)
gmm <- Mclust(mglass,G=1:10)
summary(gmm)
```

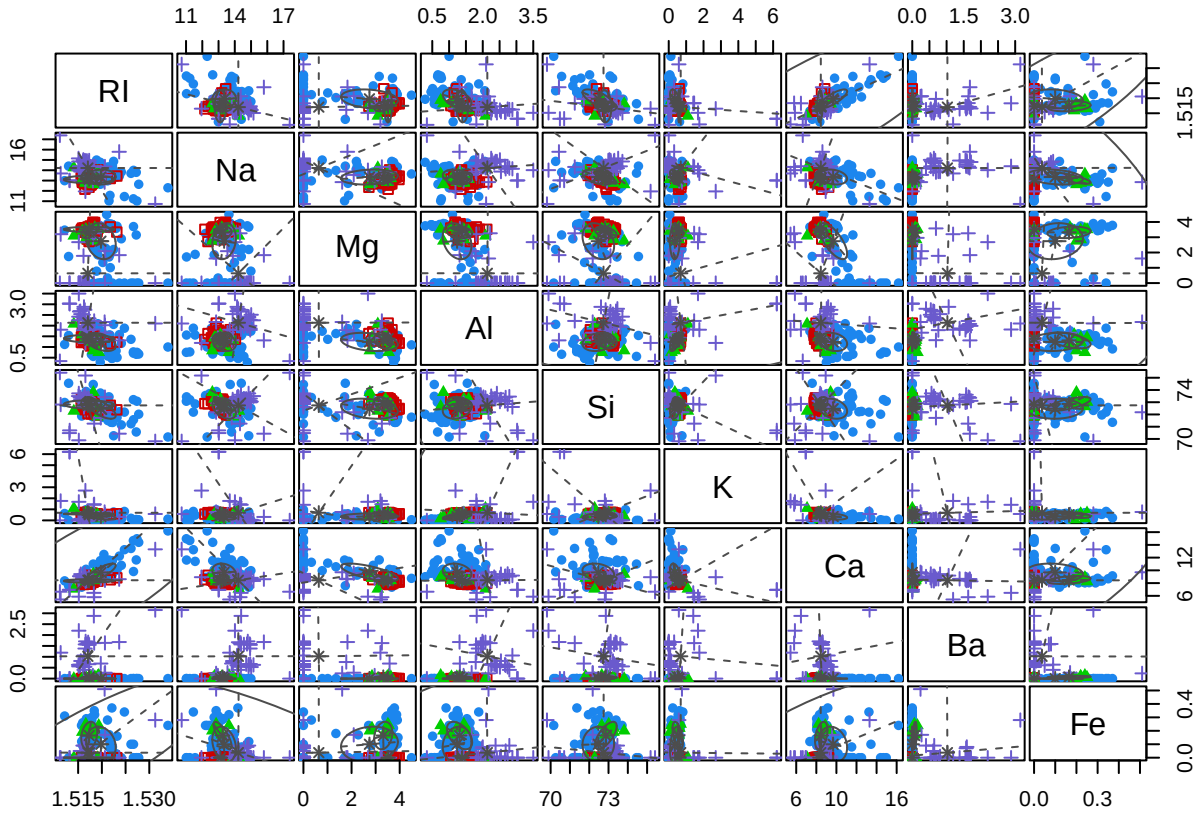
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 4 components:
##
##  log-likelihood   n  df      BIC      ICL
##        2557.753 214 195 4069.141 4068.502
##
## Clustering table:
##  1  2  3  4
## 94 75  9 36
```

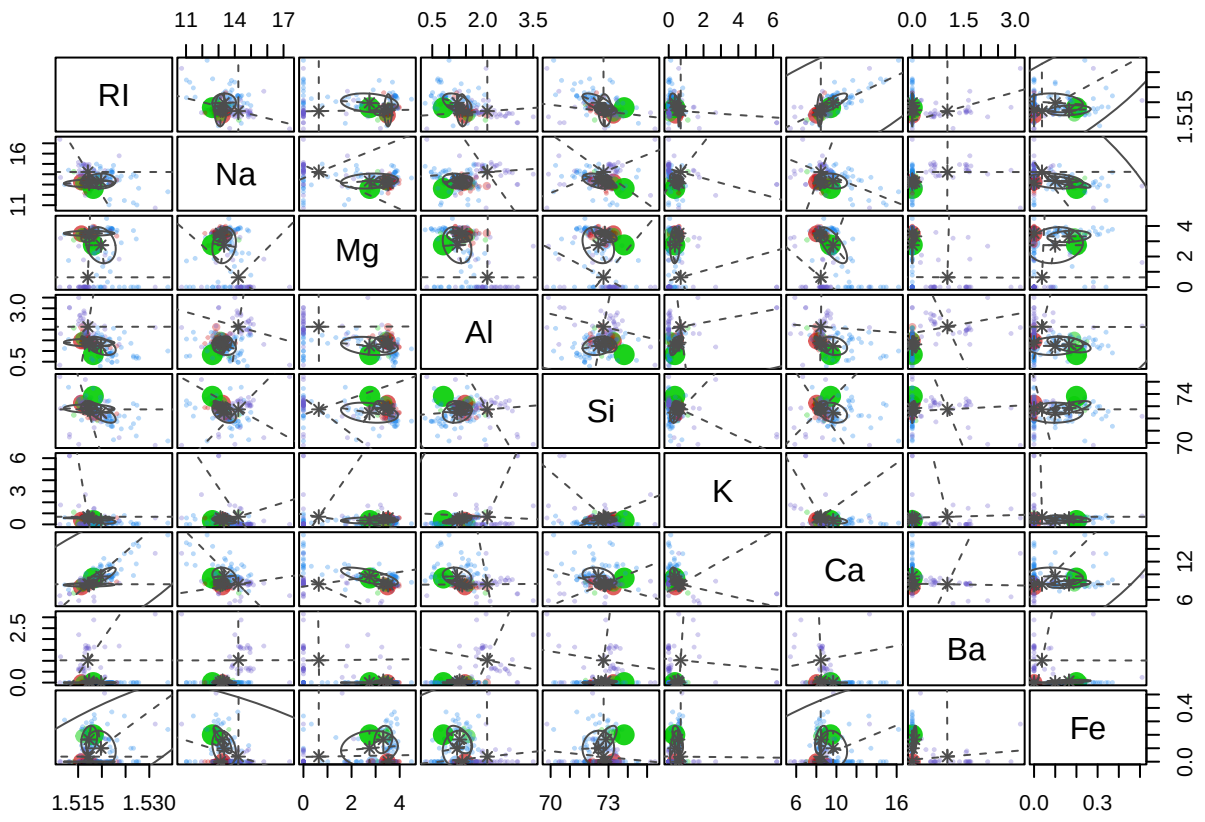
```r
summary(gmm$BIC)
```

```
## Best BIC values:
##             VEV,4      VEV,3     VEV,2
## BIC      4069.141 4028.56162  3066.734
## BIC diff    0.000  -40.57902 -1002.407
```

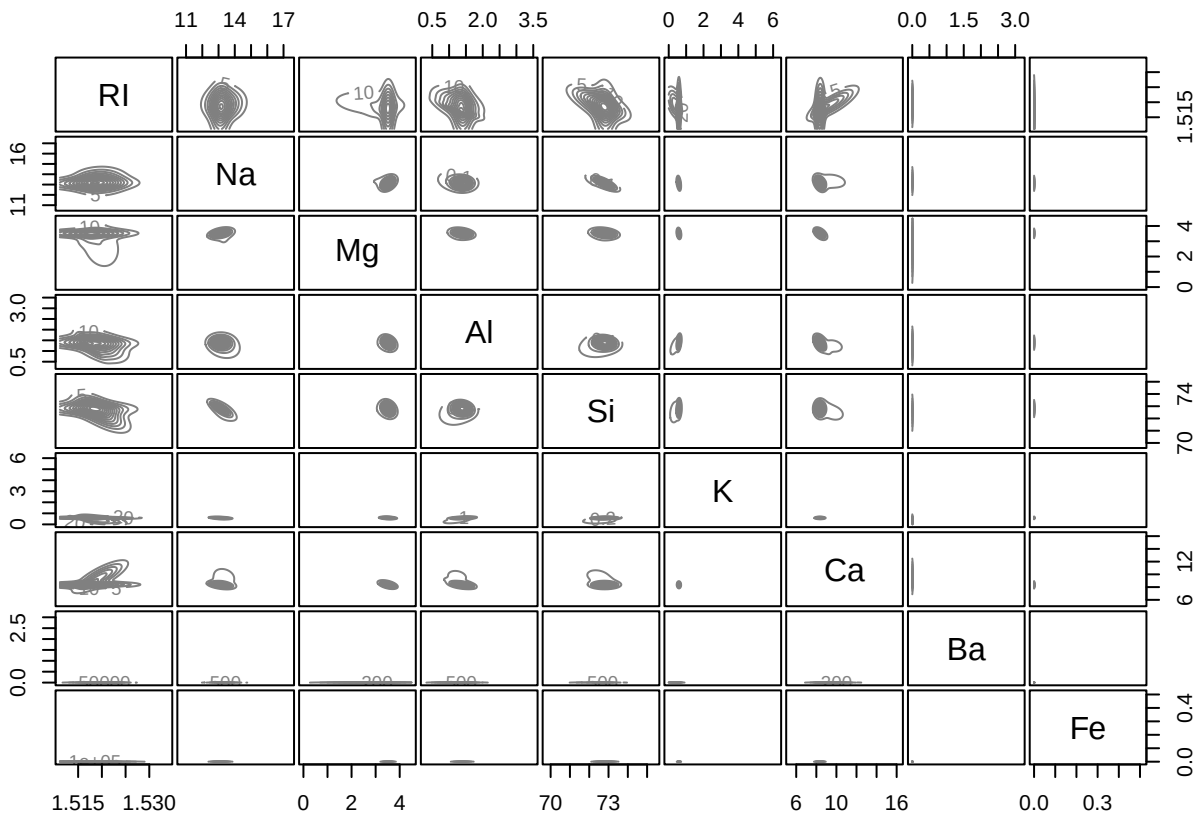
```r
plot(gmm)
```



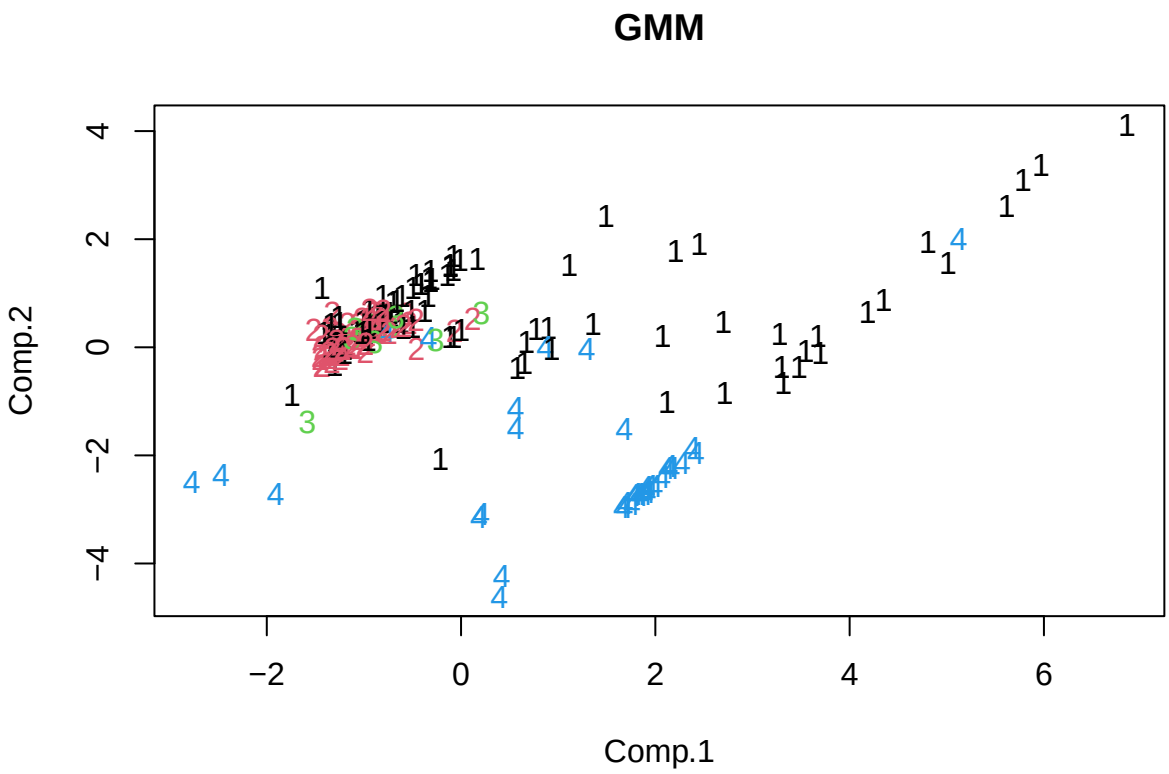

RI
Na
Mg
Al
Si
K
Ca
Ba
Fe




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




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



The selected model by MClust is VEV (ellipsoidal, equal shape) model with $K = 4$, since the highest value of BIC is reached with these settings.

Unfortunately, the paired scatterplots do not reveal much. However, it is noticeable how in both the tables of paired scatterplots, 2 clusters are always very close and overlapping with similar means and variances, while the other two usually have sparse units and peaks of density that are clearly distant from the peaks of the first 2. This detail might indicate that K should be smaller than 4. Perhaps $K = 2$ or $K = 3$ is better. The level curves plot is not meaningful and the peaks of density does not comply with the choice of $K = 4$.

The bi-dimensional plot obtained with PC's confirms our comments, as cluster 2 and 3 are very close and almost overlapping, while cluster 1 and 4 have many sparse units. There might be a problem not just with the chosen number of clusters but also with the association of the units with the clusters. For instance, it may be better that clusters 2 and 3 would become one single cluster and cluster 1 and 4 should be split in more clusters, as many as we visually see their peaks of density.

K-Means:

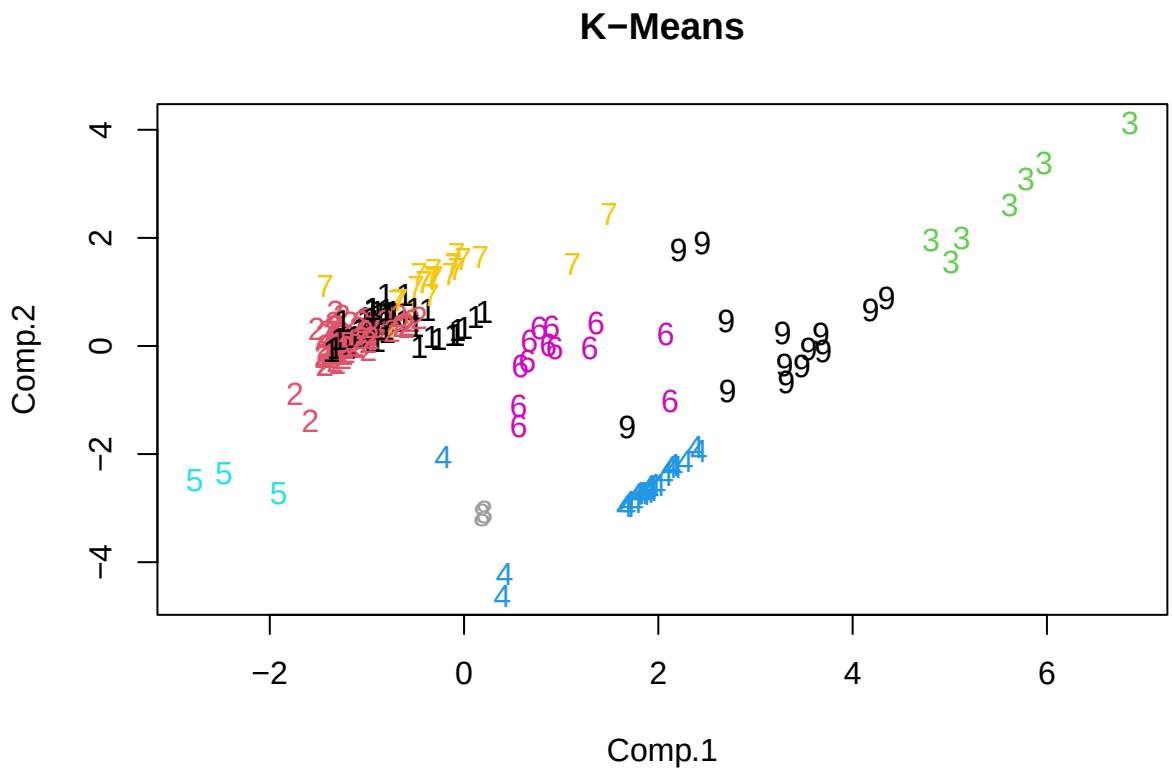
Another clustering method to apply is the K-Means through the "clusGap" function.

```
gap <- function(data,FUNcluster=kmeans, K.max=10, B = 50, d.power = 2,
                spaceH0 ="scaledPCA",method ="globalSEmax", SE.factor = 2,...){

          gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)

          nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)
```

```
          kmopt <- kmeans(data,nc,...)
          out <- list()
          out$gapout <- gap1
          out$nc <- nc
          out$kmopt <- kmopt
          out
}
km_gap_glass <- gap(mglass, nstart=1)
km_gap_glass$nc
```

```
## [1] 9
```

```
km_glass <- km_gap_glass$kmopt

plot(pc_glass$scores,col=km_glass$cluster,pch=clusym[km_glass$cluster])
title(main="K-Means")
```

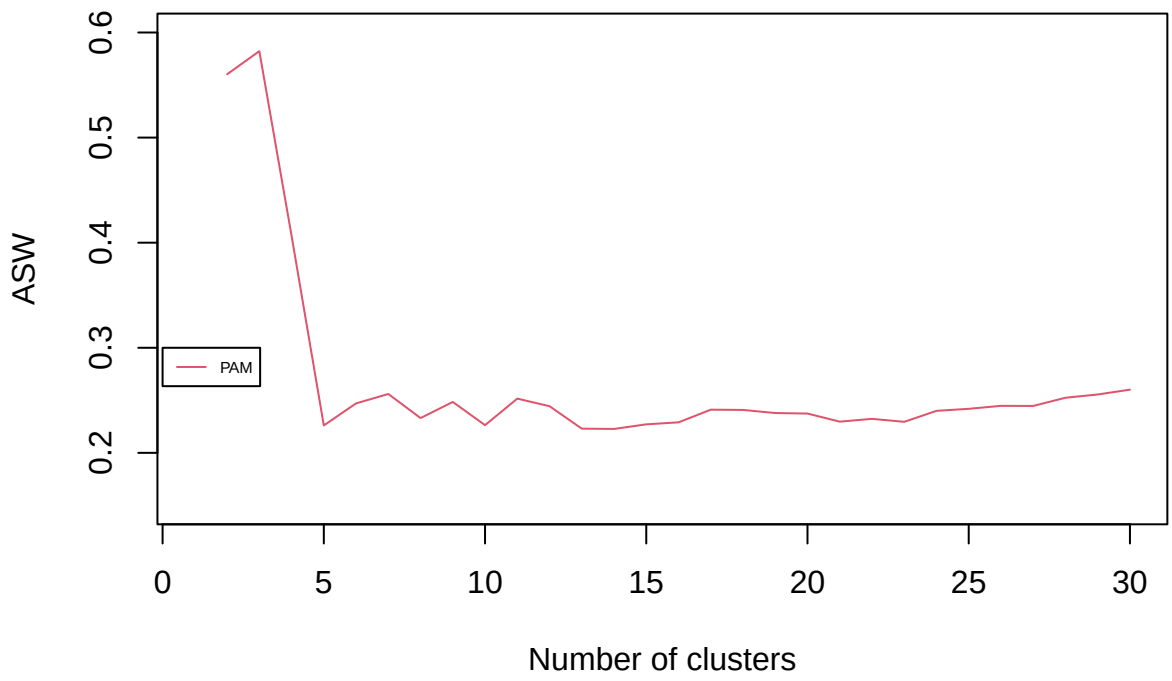


The ClusGap function has selected $K = 9$, which is much higher than GMM and goes against our assumption that K should be smaller. As a consequence, there are many less sparse units and the clusters look more regular, even though cluster 5 and 8 contain very few points and cluster 1,2,4 and 7 are composed of many units. There is still a problem with overlapping because cluster 1 and 2 are very close to eachother. Let us keep in mind that we are using a 2D plot to represent a 9D dataset and there is inevitably a loss of information. Also cluster 4 seems to have three outliers

PAM

The following code applies the Partitioning Around Medoids method:

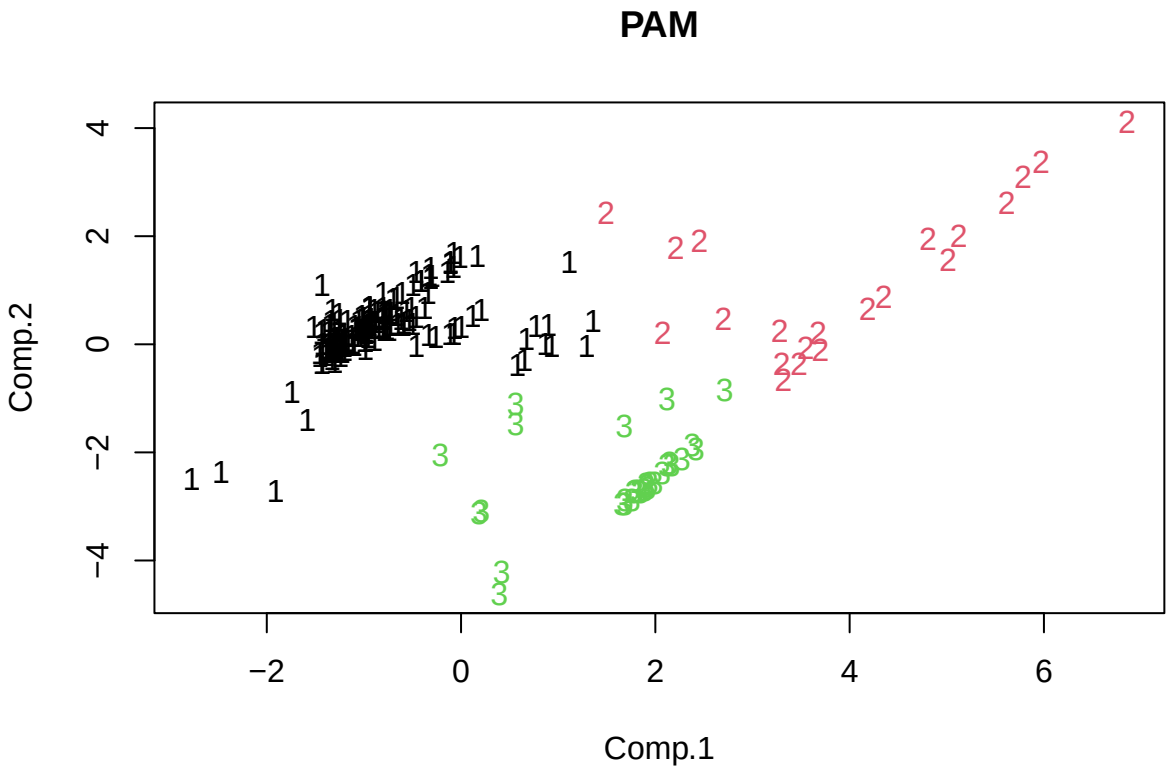
```
# PAM:
pam_sw <- NA
pam_clust <- list()
pam_sil <- list()
for (k in 2:30){
pam_clust[[k]] <- pam(mglass,k)
pam_sil[[k]] <- silhouette(pam_clust[[k]])
pam_sw[k] <- summary(pam_sil[[k]])$avg.width
}
plot(1:30,pam_sw,type="l",col = 2, xlab="Number of clusters",ylab="ASW",ylim=c(0.15,0.6))
legend(0,0.30,legend=c("PAM"),col=2, cex = 0.5, lwd=c(1,2),lty=c(1,2))
```




```
which.max(pam_sw)
```

```
## [1] 3
```

```
plot(pc_glass$scores,col=pam_clust[[3]]$clustering,
     pch=clusym[pam_clust[[3]]$clustering])
title(main="PAM")
```



The Average Silhouette Width (ASW) reaches its highest value at $K = 3$. The PAM method apparently yields the best clusterisation by looking just at the plot made with the PC's, since all the three clusters are pretty regular, homogeneous and their points are not too sparse.

Mixtures of skew and heavy-tailed distributions

Let us test the last clustering method with the code written below.

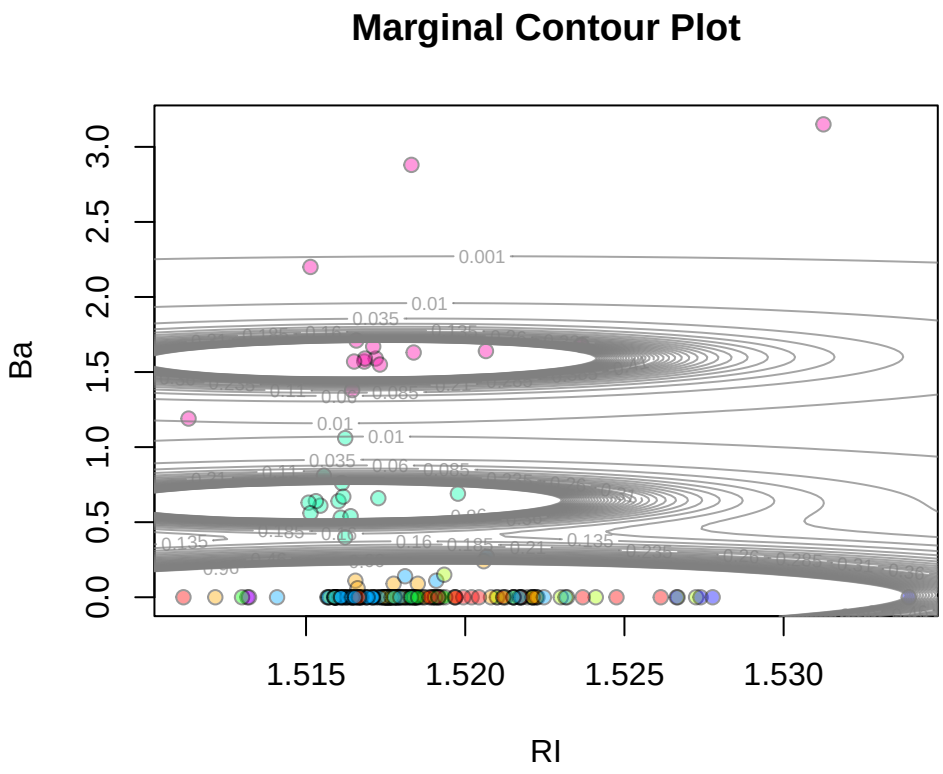
Remark: We have to set scale=FALSE since the default option standardises the data.

```
set.seed(2423423)
t_glass <- teigen(mglass, Gs=1:10, scale = FALSE)
```

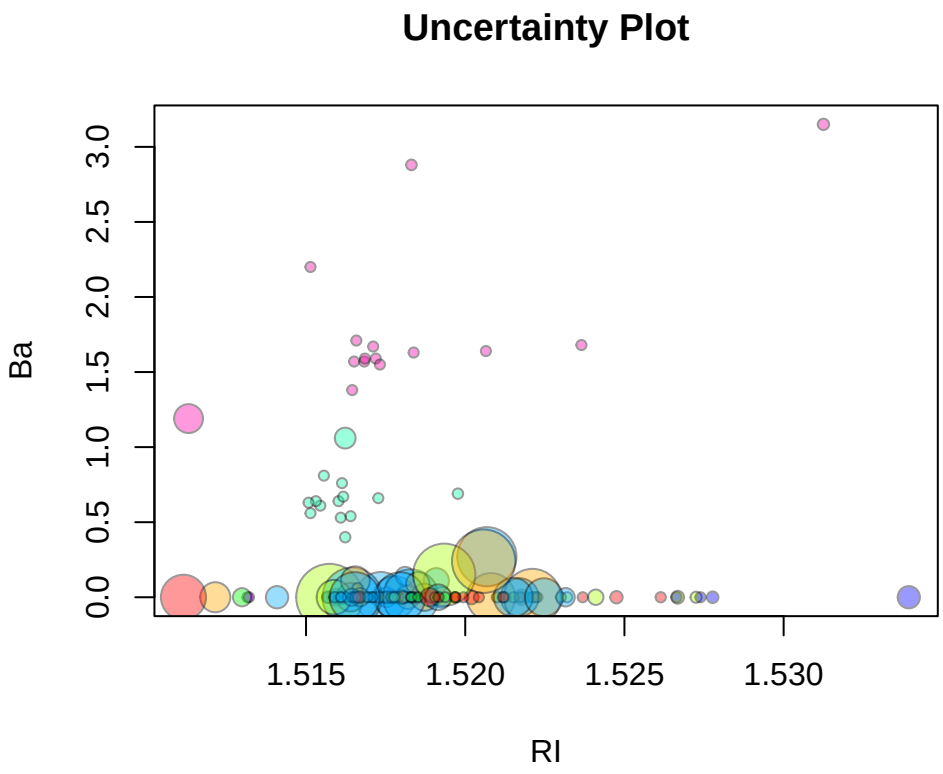
```
t_glass$bestmodel
```

```
## [1] "The best model (BIC of 2564.89) is CCCC with G=9"
```

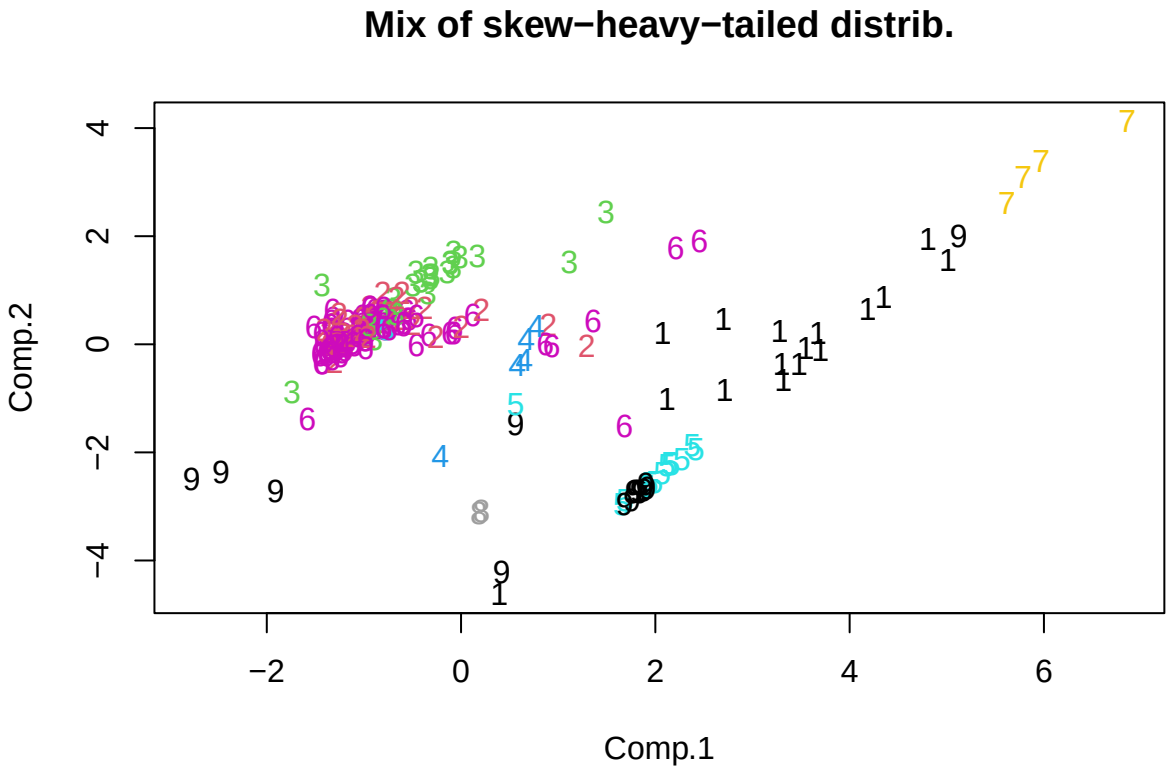
```
plot(t_glass,xmarg=1,ymarg=8,what="contour")
```




```
plot(t_glass,xmarg=1,ymarg=8,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_glass$scores,col=t_glass$classification,pch=clusym[t_glass$classification])
title(main="Mix of skew-heavy-tailed distrib.")
```



Remark: the countour and uncertainty plots are just scatterplots of two variables of the original variables.

As we mentioned before, it may be interesting to try clustering with mixtures of heavy tailed distributions with the "teigen" function. The selected model is "CCCC" with $K = 9$ (i.e. constrained volume, shape, orientation and equal degrees of freedom). By looking at the countour and uncertainty plots we notice that the level curves indicate a distribution with three modes. The two smaller peaks are represented by the pink and violet clusters (though there are some outliers), while the main peak is composed of all the remaining clusters. This result might reveal that the decision to select $K = 9$ by the teigen function is probably not precise. Once again it is useful to rely on the visualization of the data and notice that $K = 3$ is probably a better choice, as claimed in the clusterisation with PAM.

Remark: using other pairs of variables yields always a distribution with three modes.

Finally from the plot of PC's we kind of confirm that $K = 9$ is not a good choice, since clusters 2,3 and 6 are very close and could be grouped together, while cluster 1 and 9 are quite sparse. Clusters 4,5,7 and 8 seem to be homogeneous.

Remark: It is very difficult to determine the true underlying clusters in the data, since the degrees of freedom of the distributions may vary and BIC is not enough to determine the real K and the correct labeling of the units to the actual clusters. We have used the plots as an extra help to chose our clusterisation but the domain knowledge is fundamental.

2

(a)

Task: Show that for $k = 1, ..., K$, regardless of $a_1, ..., a_K$, $\sigma_1^2, ..., \sigma_K^2$, (1) is maximised by choosing π_k as:

$$\pi_k^* = \frac{1}{n} \sum_{i=1}^{n} p_{ik}$$

Solution: To maximize the expression E_η, with respect to π_k subject to the constraint $\sum_{k=1}^{K} \pi_k = 1$, we can use the method of Lagrange multipliers. The Lagrangian is expressed as follows:

$$L(\pi_1, \pi_2, \ldots, \pi_K, \lambda) = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \left(\log \pi_k + \log \varphi_{a_k, \sigma_k^2}(x_i) \right) - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

Now we need to compute the partial derivatives of L with respect to π_k $(k = 1, ..., K)$ and λ. Let us start with the former:

$$\frac{dL}{d\pi_k} = \frac{d}{d\pi_k} \left(\sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \pi_k + p_{ik} \log \varphi_{a_k, \sigma_k^2}(x_i) - \lambda \left(\sum_{k=1}^{K} \pi_{k-1} \right) \right)$$

$$= \frac{d}{d\pi_k} \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \pi_k - \frac{d}{d\pi_k} \lambda \left(\sum_{n=1}^{K} \pi_n - 1 \right)$$

Then, for a given k, we set the partial derivative equal to 0:

$$= \sum_{i=1}^{n} \frac{p_{ik}}{\pi_k} - \lambda = 0$$

Finally, we want to isolate π_k as follows:

$$\pi_k = \sum_{i=1}^{n} \frac{p_{ik}}{\lambda}$$

We compute the second partial derivative:

$$\frac{dL}{d\lambda} = \frac{d}{d\lambda} \left(\sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \pi_k + p_{ik} \log \varphi_{a_k, \sigma_k^2}(x_i) - \lambda \left(\sum_{k=1}^{K} \pi_{k-1} \right) \right)$$

which basically reduces to:

$$-\sum_{k=1}^{K} \pi_k + 1 = 0$$

$$\sum_{k=1}^{K} \pi_k = 1$$

Let us substitute π_k with its expression from the first derivative:

$$\sum_{k=1}^{K}\sum_{i=1}^{n}\frac{p_{ik}}{\lambda}=\sum_{k=1}^{n}\sum_{i=1}^{K}\frac{p_{ik}}{\lambda}=1$$

By the professor's hint, we know that $\sum_{i=1}^{n}\sum_{k=1}^{K}p_{ik}=n$. Hence:

$$\frac{n}{\lambda}=1$$

And:

$$\lambda=n$$

Finally, by using the last result of the first derivative, we obtain:

$$\pi_k=\sum_{i=1}^{n}\frac{p_{ik}}{\lambda}\Rightarrow\pi_n^*=\sum_{i=1}^{n}\frac{p_{ik}}{n}=\frac{1}{n}\sum_{i=1}^{n}p_{ik}$$

(b)

Task: Show that for $k=1,...,K$, regardless of $\pi_1,...,\pi_K$, (1) is maximised by choosing a_k as

$$a_k^*=\frac{1}{\sum_{i=1}^{n}p_{ik}}\sum_{i=1}^{n}p_{ik}x_i$$

and σ_k^2

$$\sigma_k^{2*}=\frac{1}{\sum_{i=1}^{n}p_{ik}}\sum_{i=1}^{n}p_{ik}\left(x_i-a_k\right)^2.$$

Solution: Once again, we can maximize the following expression (regardless $\pi_1,...,\pi_K$)

$$E_\eta=\sum_{i=1}^{n}\sum_{k=1}^{K}p_{ik}\left(\log\pi_k+\log\varphi_{a_k,\sigma_k^2}\left(x_i\right)\right)$$

by computing the partial derivatives for a given k and with respect to a_k and σ_k^2 from the Lagrangian.

Remark: If we change φ_{a_k,σ_k^2} with the full expression of the Gaussian density, we get the following expressions to set equal to 0:

$$\frac{d}{d\hat{a_k}}\left(\sum_{i=1}^{n}p_{ik}\log\left[\sum_{n=1}^{k}\hat{\pi}_k\frac{1}{\sqrt{2\pi\hat{\sigma}_k^2}}\exp\left(-\frac{\left(x_i-\hat{a_k}\right)^2}{2\hat{\sigma_k^2}}\right)\right]-\lambda\left(\sum_{k=1}^{K}\pi_{k-1}\right)\right)=0$$

$$\frac{d}{d\hat{\sigma_k^2}}\left(\sum_{i=1}^{n}p_{ik}\log\left[\sum_{n=1}^{k}\hat{\pi}_k\frac{1}{\sqrt{2\pi\hat{\sigma}_k^2}}\exp\left(-\frac{\left(x_i-\hat{a_k}\right)^2}{2\hat{\sigma_k^2}}\right)\right]-\lambda\left(\sum_{k=1}^{K}\pi_{k-1}\right)\right)=0$$

By doing some algebraic modifications from these two expressions, we could arrive to the thesis of our proof as we previously did in point (a).

However, if we use the theoretical knowledge about ML of a Gaussian distribution, we can use a shortcut as shown below. In facts, the ML estimators for a Gaussian distribution (given k) are:

$$\hat{a_k} = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\hat{\sigma_k}^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \hat{a_k})^2$$

It simply remains to introduce the weigths p_{ik} to x_i and $(x_i - \hat{a_k})^2$.

Remark: since we are using weigthed estimators, we do not divide by the sample size n anymore. We should use instead the sum of weigths.

The final result of the proof is shown below:

$${a_k}^* = \frac{1}{\sum_{i=1}^{n} p_{ik}} \sum_{i=1}^{n} p_{ik}\ x_i$$
$$\sigma_k^{2*} = \frac{1}{\sum_{i=1}^{n} p_{ik}} \sum_{i=1}^{n} p_{ik}\,(x_i - \hat{a_k})^2$$

3

Task:

Read the paper and answer the following questions:

- (a) Summarize in your own words what the DBSCAN method does
- (b) What are the advantages, according to the authors, of their DBSCAN method compared with other clustering methods, particularly those that you already know? Do you think that the authors' arguments are convincing?
- (c) Find out how to run DBSCAN in R and apply it to the Glass data from question 1 (you may also try it out on other datasets). You will need to make some decisions (and or experiments) about tuning parameters. Comment on the results.

Solution:

a) DBSCAN estimates the density around each point by counting the number of points in a neighborhood (eps) specified by the user, and applies the thresholds $minPts$ to identify the "core", "border" and "noise" points.

In the second step, the core points are gathered into a cluster, if they are "density-reachable", i.e. if there is a chain of core points in which each point falls within the eps-neighborhood of the following one.

Finally the border points are assigned to the clusters. The algorithm requires only the parameters eps And $minPts$.

The points are classified as "core points", density-reachable or outliers, according to the following rules:

A point p is a core point if at least $minPts$ points are within the distance eps (i.e. the maximum radius for the neighborhood of p) from it (including p itself). These points are called "directly reachable" by p. A

point q is directly reachable from p if the point q is within distance *eps* from point p, with p being a core point.

A point q is reachable from p if there exists a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where every $p_i + 1$ is directly reachable from p_i (all points in the path must be core, with the only possible exception of q). All points that cannot be reached from other points are outliers.

Now, if p is a core point, then this forms a cluster together with all the other points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge" (i.e. they are border points), since they cannot be used to reach other points.

The DBSCAN algorithm can be exemplified with the following steps:

- 1) Find the eps-neighbors of each point, and identifies core points with more than minPts neighbors.
- 2) Find the connected components of the core points in the neighbor graph, ignoring all non-core points.
- 3) Assign each non-core point to a neighboring cluster if the cluster is a eps close, otherwise assign it to noise.

b) The DBSCAN brings some advantages over traditional clustering methods. In particular:

- it requires less domain knowledge which is fundamental to determine the input parameters in other clustering algorithms
- it needs just one initial parameter to be used and the user is guided to choose it
- it finds clusters with arbitrary shape. Non-spherical and non-convex clusters included
- it is computationally more efficient on large datasets
- it works with any distance function.

There are also specific advantages if DBSCAN is compared to PAM. In facts, PAM requires that K is known in advance, and that the clusters are roughly spherical. Moreover, PAM is not very good at dealing with outliers because they influence the medoids.

Furthermore, if we compare DBSCAN with GMM, we can say that GMM may not handle correctly cluster whose shape is not elliptical. GMM is also not robust to outliers and requires to set a range of K in advance. The tuning of parameters is harder as well.

If on one hand hierarchical clustering is good at handling non convex clusters, it is computationally more demanding than DBSCAN on large datasets.

Same things can be said when DBSCAN is compared with K-Means.

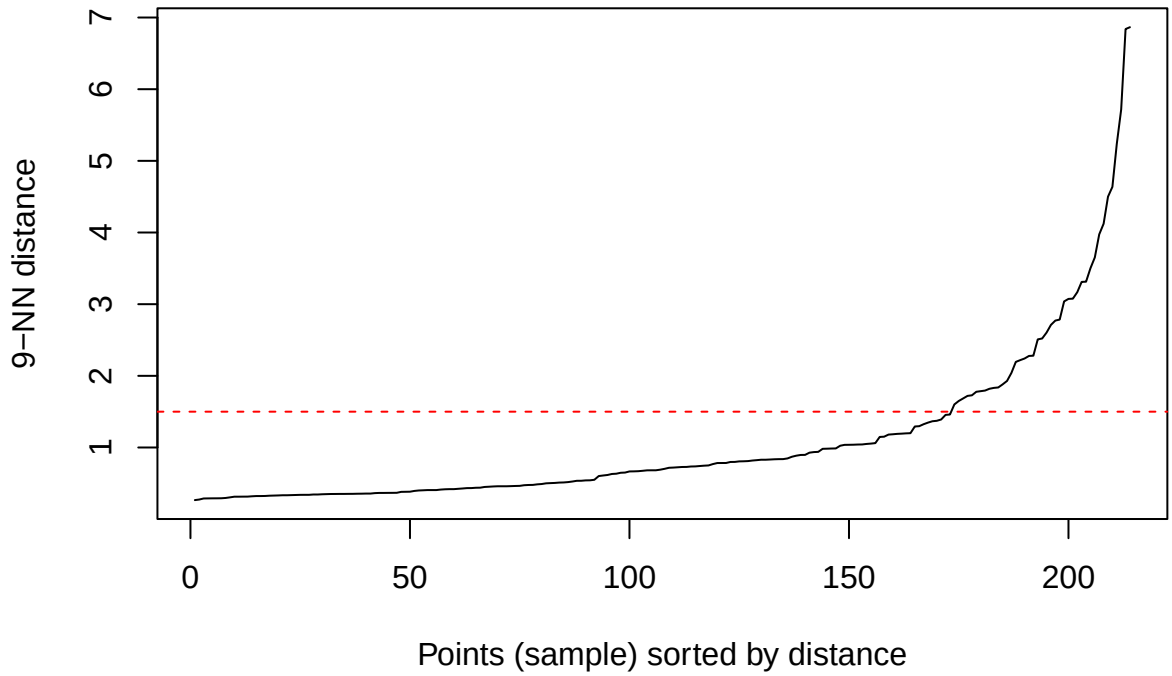
c) In order to apply "DBSCAN" in R to the glass dataset, we just need to install the "dbscan" package and upload it via the library function. We shall use the KNN algorithm to euristically chose the parameters to use in "dbscan" by looking at the elbow plot.

Remark: In the KNN, as our rule of thumb, we set K $\geq$ *number of variables* = 9. It is in practice a popular rule but larger values may be necessary for large or noisy datasets or those containing many duplicates. With this dataset the final choise goes to K = 9 = $minPts$.

To choose the value of *eps*, a k-distance graph is plotted by ordering the distance to the K= $minPts - 1$ nearest neighbor from the largest to the smallest value. Good values of *eps* are where the plot shows an "elbow". If *eps* is too small, a large portion of the data will not be clustered. If *eps* is too high, clusters will merge, and most objects will be in the same cluster.

After some trials, the final parameters are shown below:

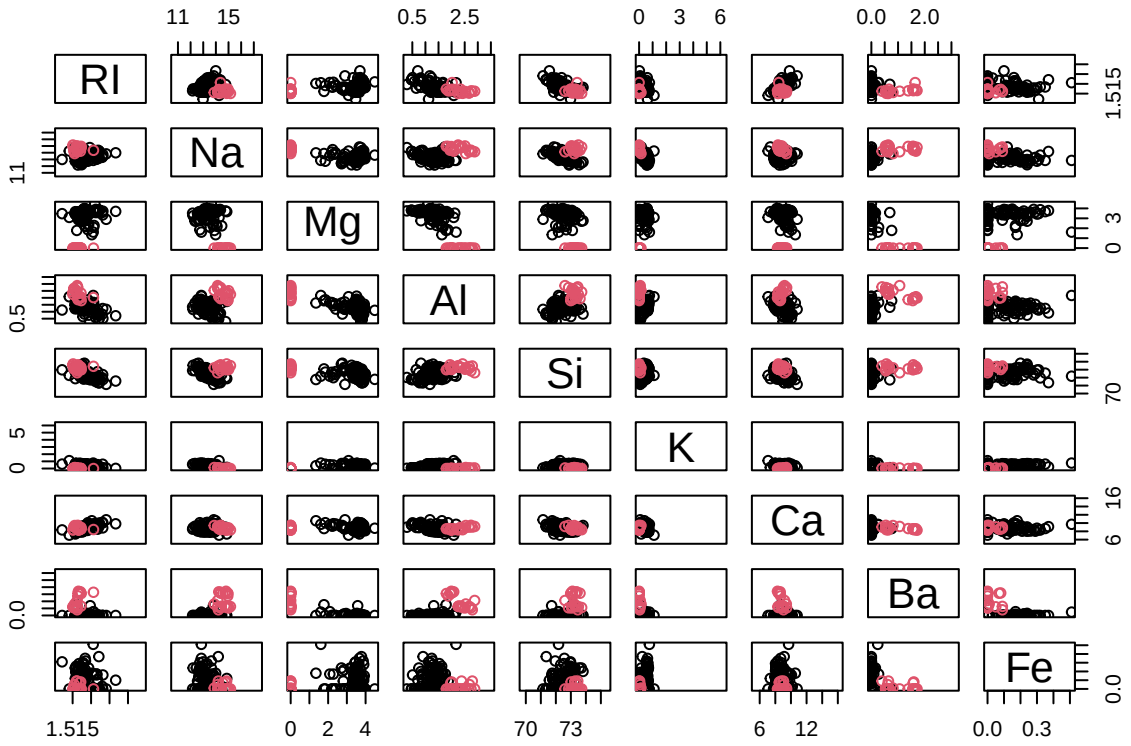
```
k <- 9
kNNdistplot(mglass, k)
abline(h=1.5, col = "red", lty=2)
```



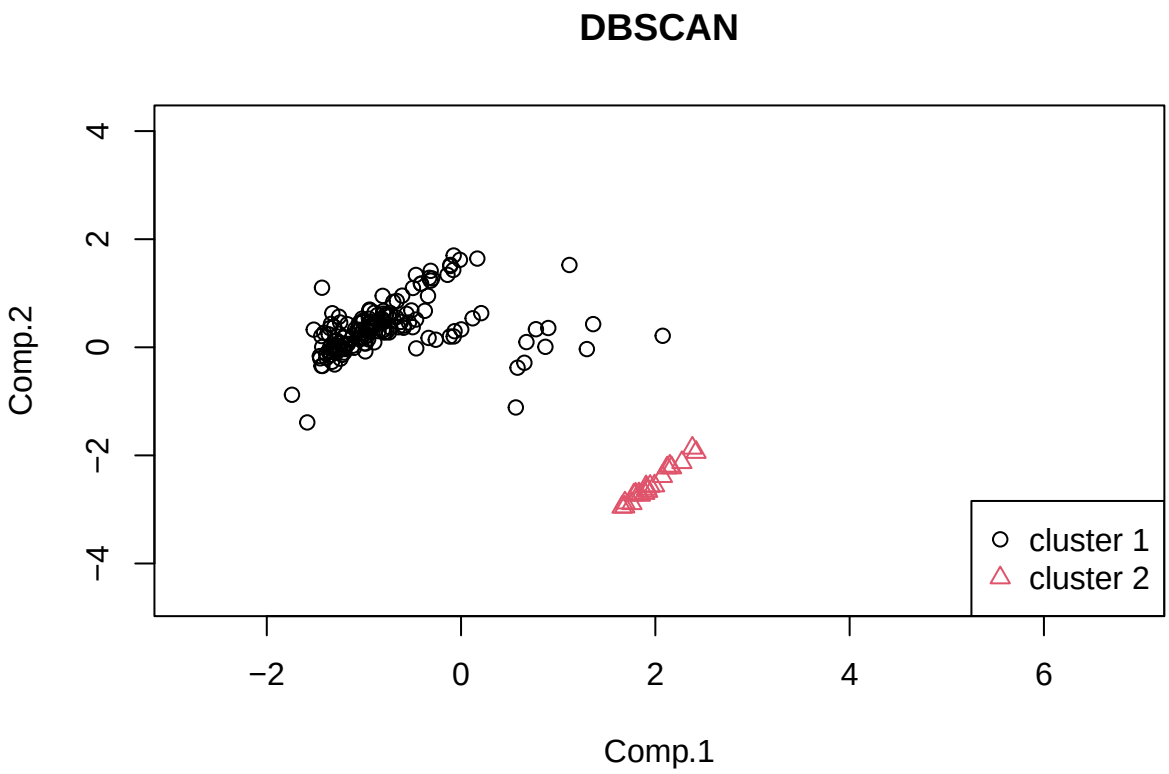

```
db_glass <- dbscan(mglass, eps = 1.5, minPts = 9)
db_glass
```

```
## DBSCAN clustering for 214 objects.
## Parameters: eps = 1.5, minPts = 9
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 2 cluster(s) and 33 noise points.
##
##   0   1   2
##  33 160  21
##
## Available fields: cluster, eps, minPts, dist, borderPoints
```

```
pairs(mglass, col = db_glass$cluster)
```




```
plot(pc_glass$scores,col=db_glass$cluster,pch=db_glass$cluster)
title(main="DBSCAN")
legend("bottomright", legend=c("cluster 1", "cluster 2"), col = c(1,2), pch=c(1,2))
```



Remark: the outliers are not shown in the plots.

From the plots and the console, we notice that "dbscan" has selected $K = 2$. The other units left out are outliers. Both in the paired plots and in the PC's plot, the clusters appear to be homogeneous and not sparse. This can be considered a good clustering for our data.