Exercise 4

Riccardo Petrella

2023-10-30

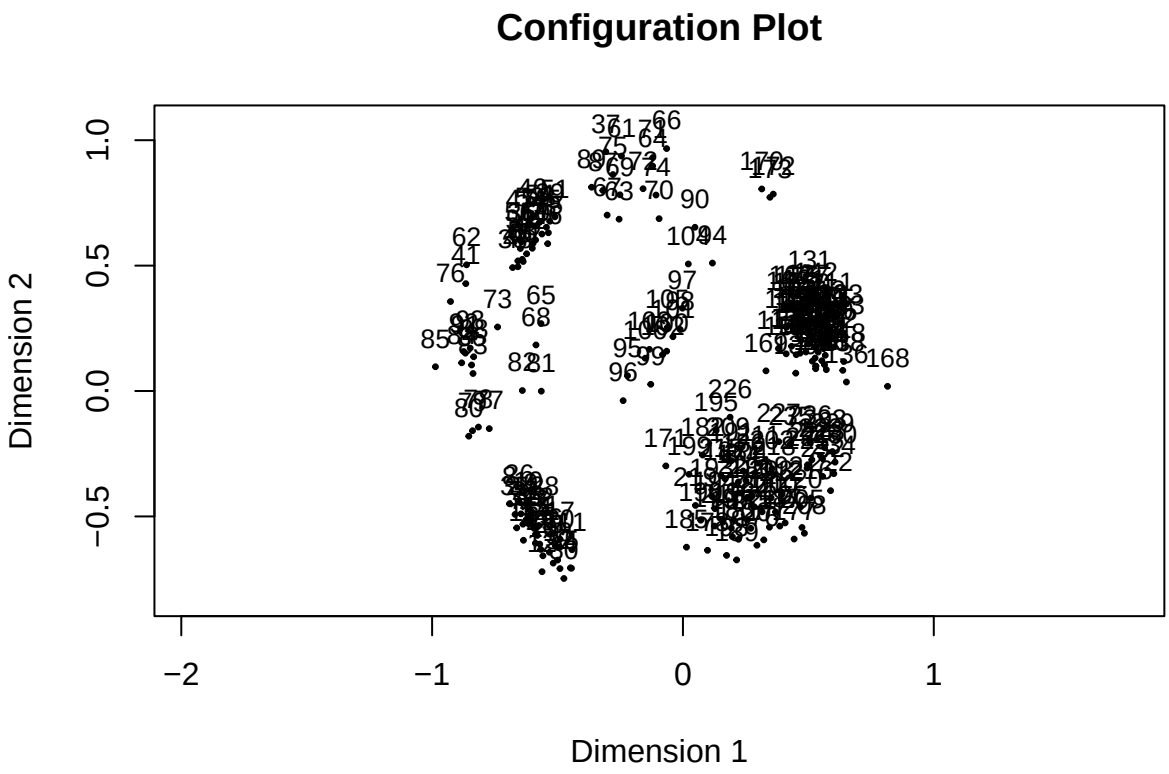
1.a

```
data(tetragonula)

ta <- alleleconvert(strmatrix=tetragonula)

tai <- alleleinit(allelematrix=ta)
```

Data preparation:

Using tai$distmat, compute an MDS and show the MDS plot. Try out different dissimilarity-based cluster analysis methods and decide which one you think is best here. Also choose a number of clusters and visualise your final clustering using the MDS. Give reasons for your choices

```
tai_mds <- mds(tai$distmat)

plot(tai_mds, asp=1)
```



By plotting the data using MDS, we immediately notice that the data points are well separated and visually grouped in several clusters. Just by a rapid look, we may say that K might be equal to $8, 9$ or sligthly more.

Let us compute clustering analysis with 3 different options:

- PAM
- Hierarchical clustering with Euclidean Distance
- Hierarchical clustering with Correlation Dissimilarity.

We will eventually choose the number of clusters K using the Average Silhouette Width (ASW).

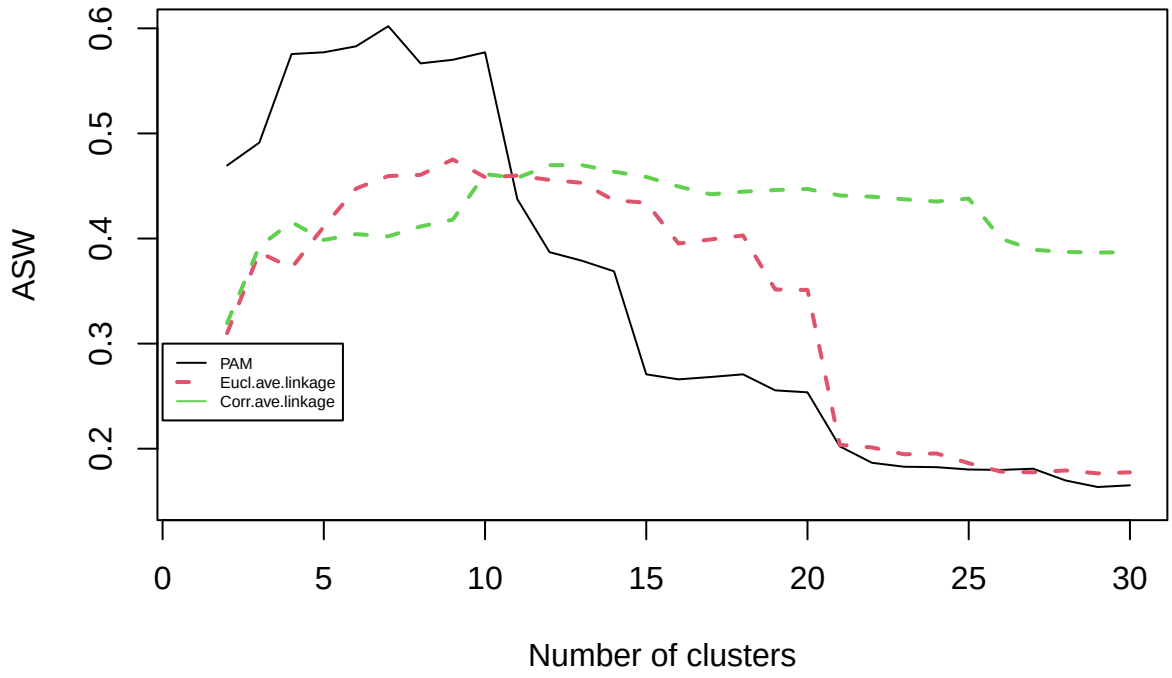
Respectively we get that:

- $K = 7$ with PAM
- $K = 9$ with Hier.Clust. with Euclidean distance
- $K = 13$ with Hier.Clust. with Correlation dissimilarity

These results comply with our visual prediction of K.

```
# PAM:
pasw <- NA
pclusk <- list()
psil <- list()
```

```r
for (k in 2:30){
pclusk[[k]] <- pam(tai$distmat,k)
psil[[k]] <- silhouette(pclusk[[k]])
pasw[k] <- summary(psil[[k]])$avg.width
}
plot(1:30,pasw,type="l",xlab="Number of clusters",ylab="ASW",ylim=c(0.15,0.6))


cor_tai <- cor(tai$distmat)
cordistai <- 0.5 - cor_tai/2 # Correlation dissimilarity
cordistai <- as.dist(cordistai)
# Average Linkage
tai_clust1 <- hclust(dist(tai$distmat, method = "euclidean"),method="average")
tai_clust2 <- hclust(cordistai,method="average")

tasw1 <- NA
tasw2 <- NA
tclusk1 <- list()
tclusk2 <- list()
tsil1 <- list()
tsil2 <- list()
for (k in 2:30){
tclusk1[[k]] <- cutree(tai_clust1,k)
tclusk2[[k]] <- cutree(tai_clust2,k)
tsil1[[k]] <- silhouette(tclusk1[[k]],dist=tai$distmat)
tsil2[[k]] <- silhouette(tclusk2[[k]],dist=tai$distmat)
tasw1[k] <- summary(silhouette(tclusk1[[k]],dist=tai$distmat))$avg.width
tasw2[k] <- summary(silhouette(tclusk2[[k]],dist=tai$distmat))$avg.width
}
points(1:30,tasw1,type="l",col=2,lty=2,lwd=2)
points(1:30,tasw2,type="l",col=3,lty=2,lwd=2)

# ...plots...
legend(0,0.30,legend=c("PAM","Eucl.ave.linkage", "Corr.ave.linkage"),col=c(1,2,3), cex = 0.5, lwd=c(1,2
```




```
which.max(pasw)
```

```
## [1] 7
```

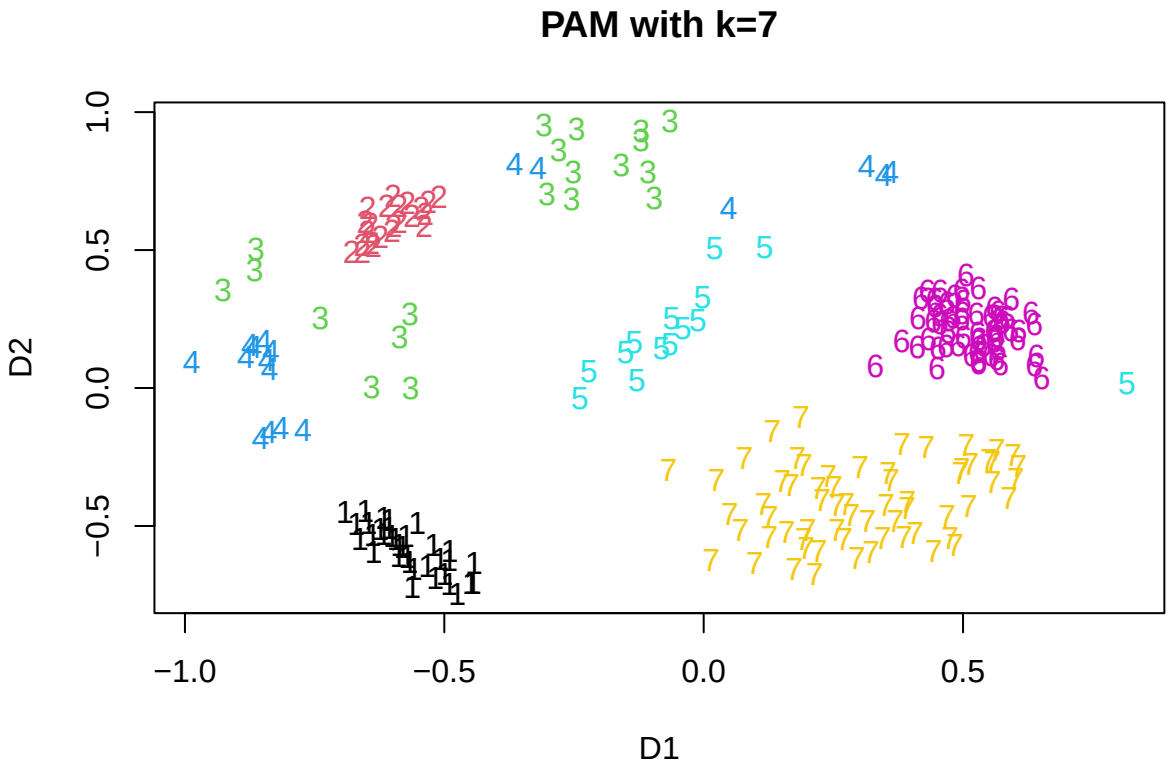
```
which.max(tasw1)
```

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

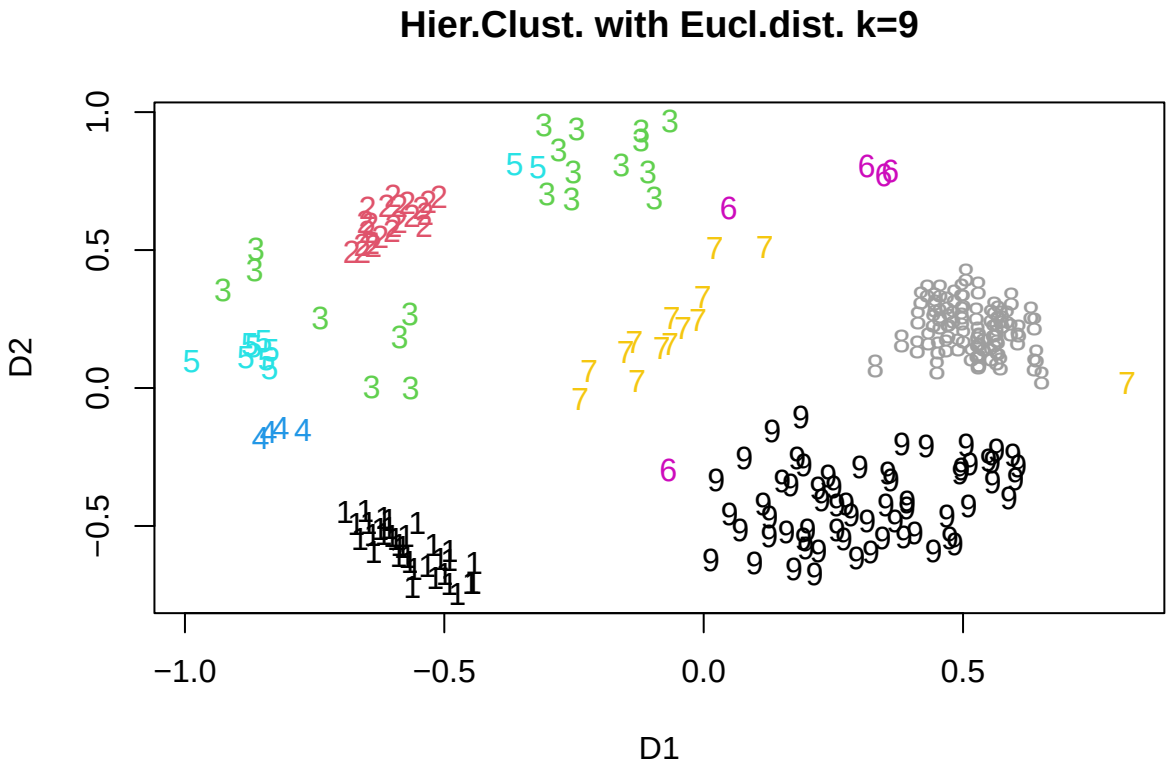
```
which.max(tasw2)
```

```
## [1] 13
```

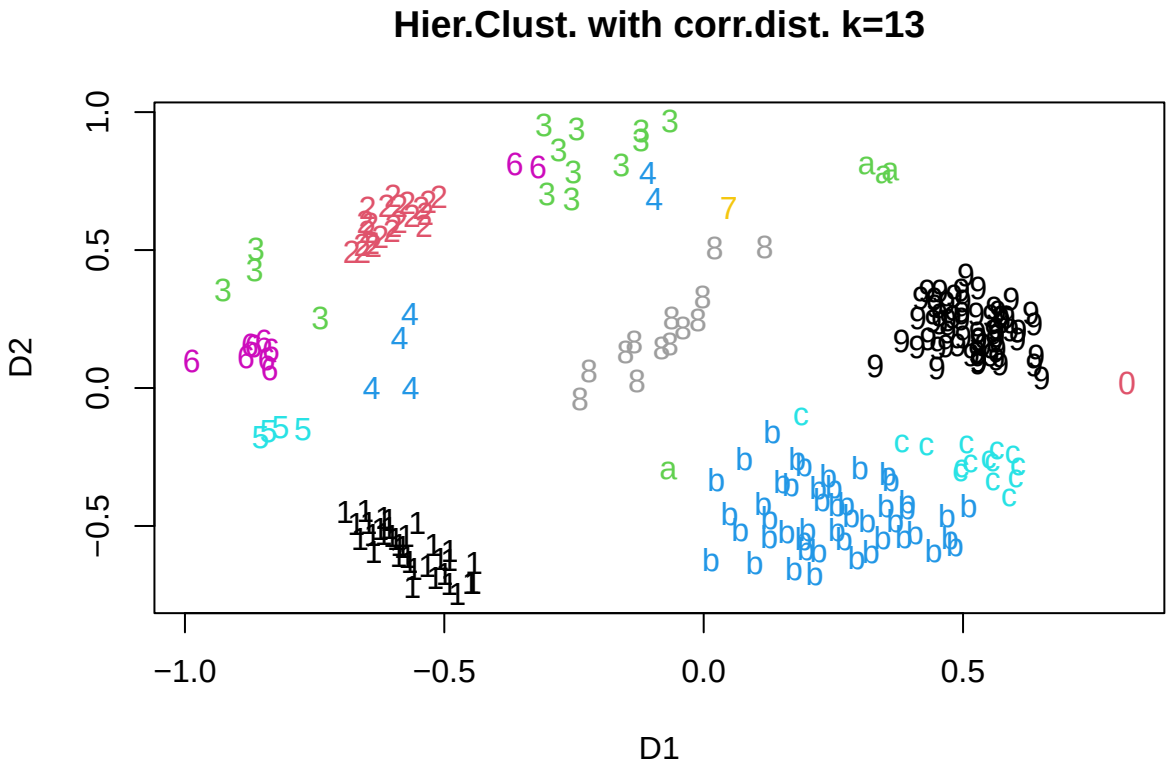
```
# MDS-plot of the final clusterings:
plot(tai_mds$conf,col=pclusk[[7]]$cluster,pch=clusym[pclusk[[7]]$cluster])
title(main="PAM with k=7")
```



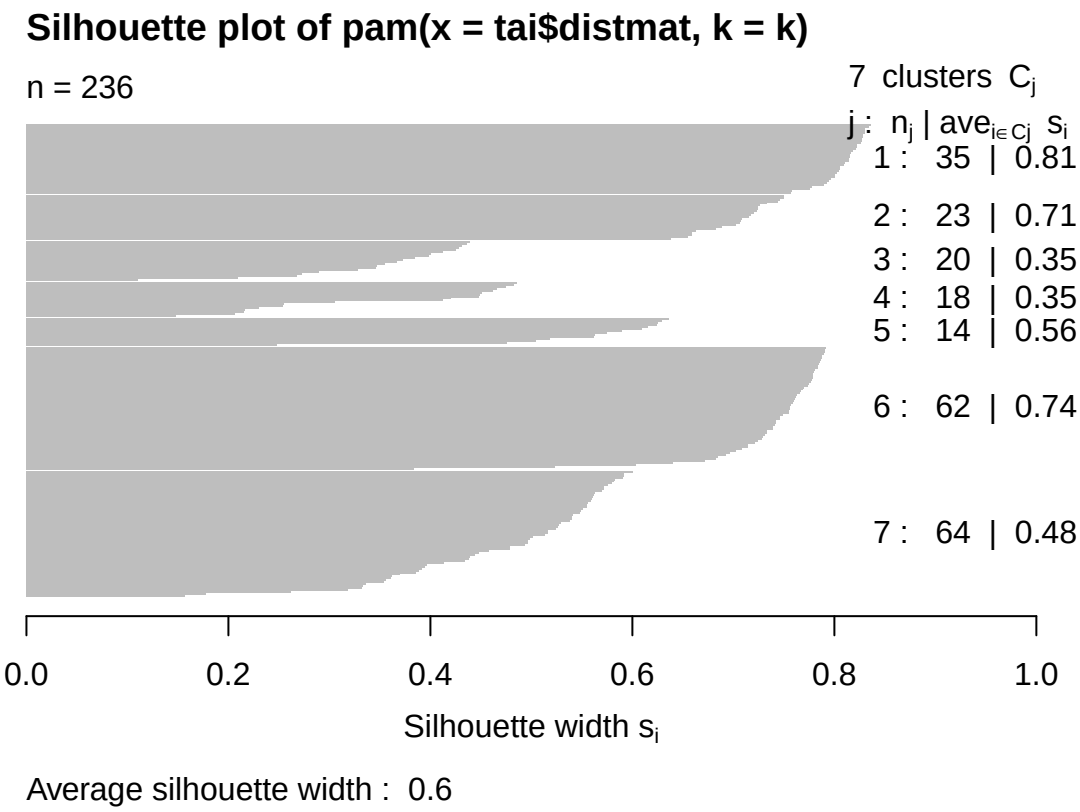

```
plot(tai_mds$conf, col=tclusk1[[9]],pch=clusym[tclusk1[[9]] ])
title(main="Hier.Clust. with Eucl.dist. k=9")
```



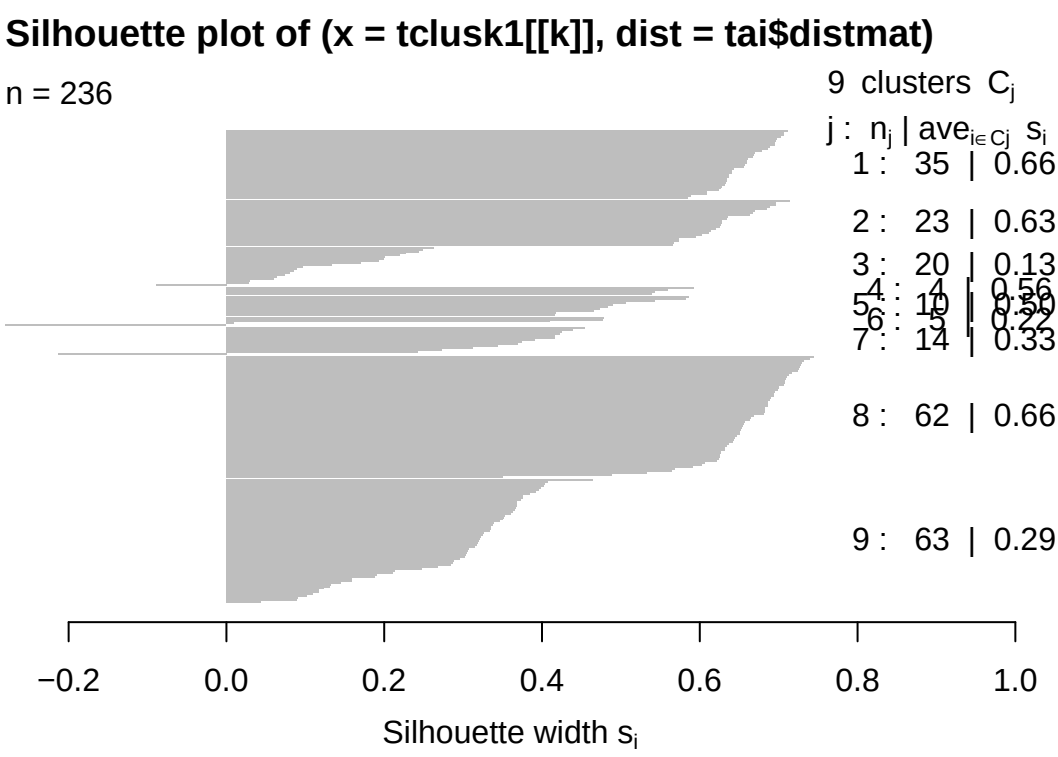

```
plot(tai_mds$conf,col=tclusk1[[13]],pch=clusym[tclusk1[[13]] ])
title(main="Hier.Clust. with corr.dist. k=13")
```




```
# Silhouette plots:
plot(psil[[7]])
```

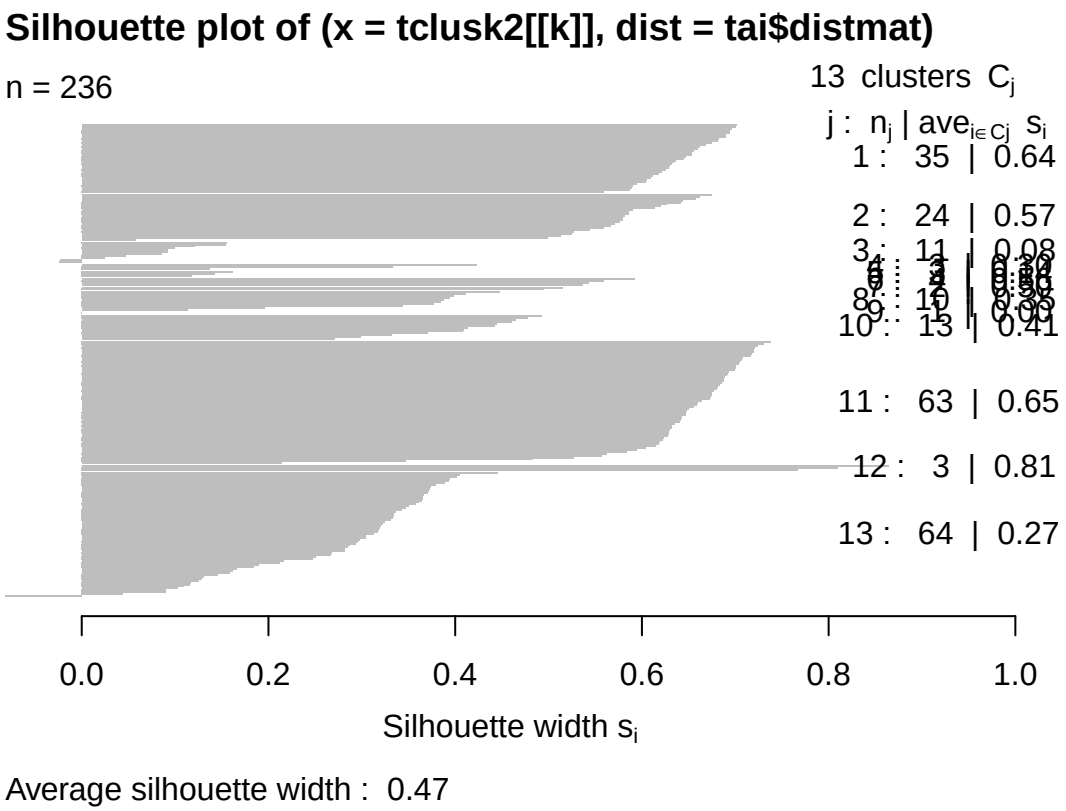

Silhouette plot of pam(x = tai$distmat, k = k)
n = 236
7 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 35 | 0.81
2 : 23 | 0.71
3 : 20 | 0.35
4 : 18 | 0.35
5 : 14 | 0.56
6 : 62 | 0.74
7 : 64 | 0.48
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.6

```
plot(tsil1[[9]])
```



Average silhouette width : 0.48

```
plot(tsil2[[13]])
```



As we can see from the MDS plots, the clusters appear to have many scattered data points. For example in the PAM plot, clusters 1,2,6,7 are well defined. However, clusters 3,4 and 5 seem to be very sparse.

There is the same problem with the other two plots but it is less serious. Perhaps, increasing the number of clusters improves the homogeneity of them. Despite that, in the hierarchical clustering with correlation, the cluster 0 contains just one unit.

Remark: This is certainly a problem caused by representing a multidimensional dataset in 2D. A further investigation is required. A MDS set to 3D may be a solution to the issue.

In the end, we choose the model with hierarchical clustering with correlation dissimilarity because a bigger K appear to match the underlying natural clusters of the dataset.

Indeed, K should always be chosen depending on the data dominion with experts of the field, rather than with statistical models which may not comply with the real life situation. Here the data give pairs of alleles for 13 diploid microsatellite loci (these can be thought of as positions of genes, each characterised by two alleles).

Hence, our third clustering perfectly find 13 varieties of bees by 13 similar genetic makeups. As a consequence, we ignore the fact that the Silhouette plots suggest us to choose $K = 7$ since the Silhouette width is the highest.

1.b

Try out the following alternative way of clustering the data: Take the points generated by the MDS and apply k-means clustering, Ward's method, or fitting a Gaussian mixture to them. Again choose a number of clusters. What are advantages and disadvantages of this approach compared to the hierarchical clustering? Do you think that this clustering is ultimately better? Do you think it would be better, for this task, to produce an MDS solution with p > 2?

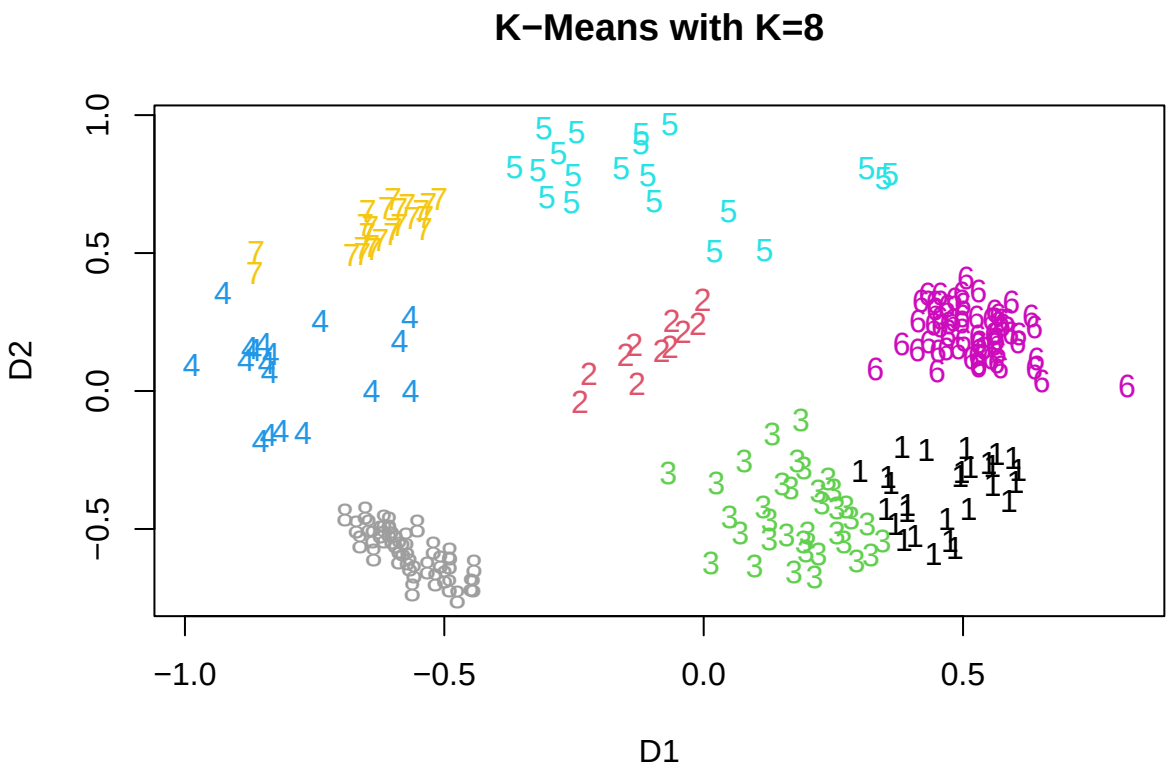
Remark: We are going to use the output of MDS rather than the data points of the original datasets as required by the task.

This time, we may use the "gap" function with "clusGap" to select the optimal K for K-Means. It yields a similar result to what we obtained using ASW, since $K = 8$.

```
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
}

gap_tai <- gap(tai_mds$conf,nstart = 100)
gap_tai$nc
```

```
## [1] 8
```

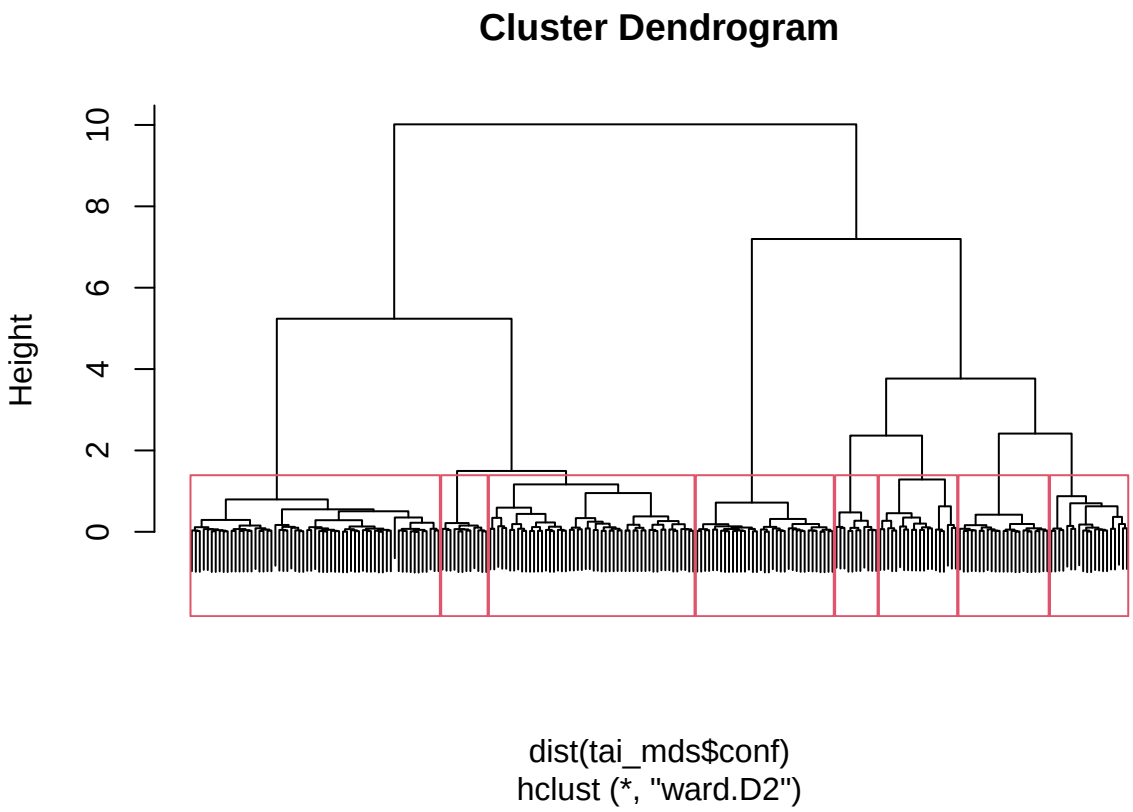
```
km_gap_tai <- gap_tai$kmopt

plot(tai_mds$conf,col=km_gap_tai$cluster,pch=clusym[km_gap_tai$cluster])
title(main="K-Means with K=8")
```



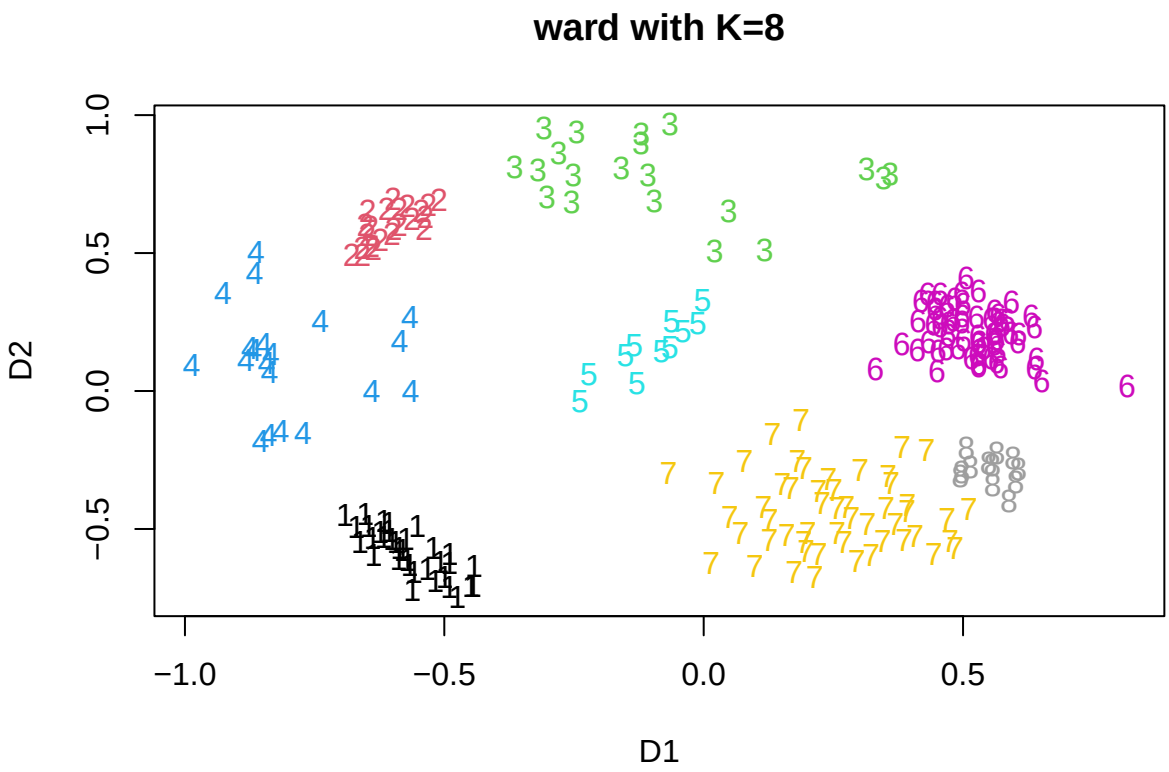
From the plot, we immediately notice that the clusters are well defined and the points are not too sparse on the plot.

Let us apply the Ward's method using the same number of clusters.

```
# Ward's method

ward_tai <- hclust(dist(tai_mds$conf), method="ward.D2")
plot(ward_tai, labels=FALSE)
rect.hclust(ward_tai, k= 8)
```




```
ward8_tai <- cutree(ward_tai,8)
plot(tai_mds$conf,col=ward8_tai,pch=clusym[ward8_tai])
title(main="ward with K=8")
```



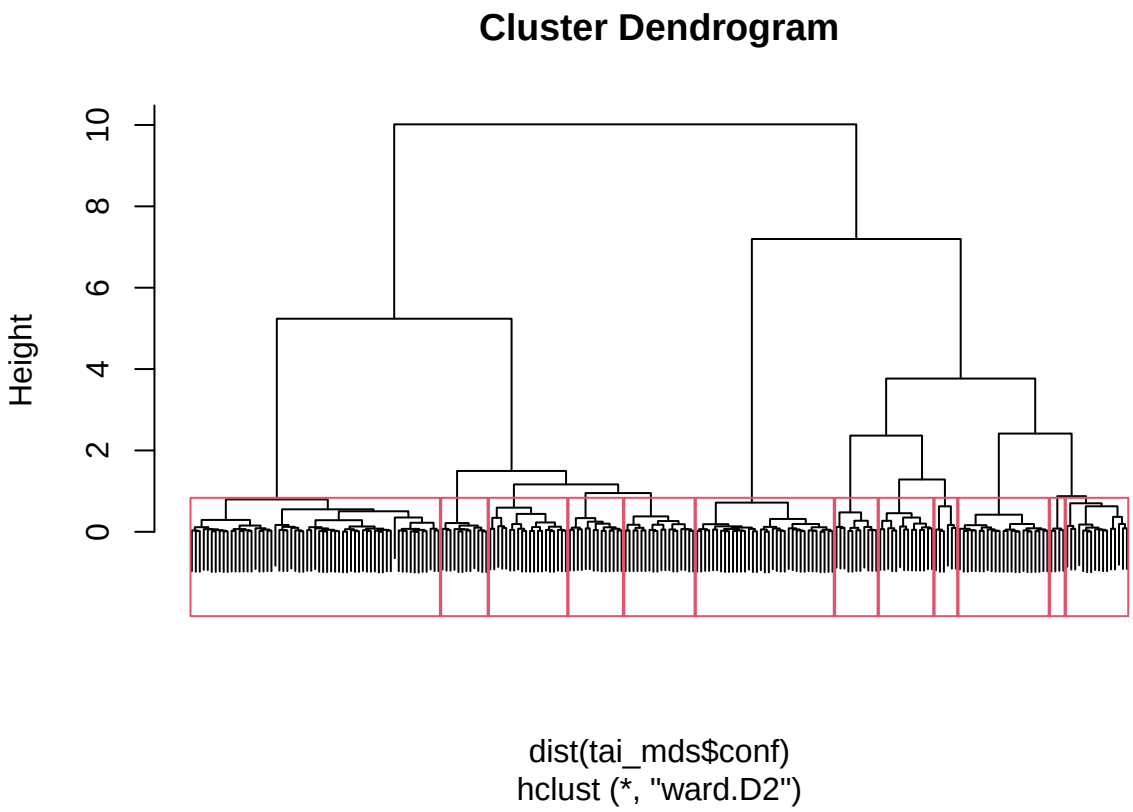
From the plots with MDS, we notice that there is a difference in the number of points contained in some clusters.

For instance, in K-means, clusters 6 and 7 are more regular and homogeneous than their counterparts in ward's (i.e. clusters 7 and 8). In facts, cluster 7 in Ward's is much bigger than cluster 6 of K-Means and cluster 8 from Ward's is much smaller than 7. The same phenomenon involves clusters 3-5 in K-Means vs clusters 2-4 in Ward's.

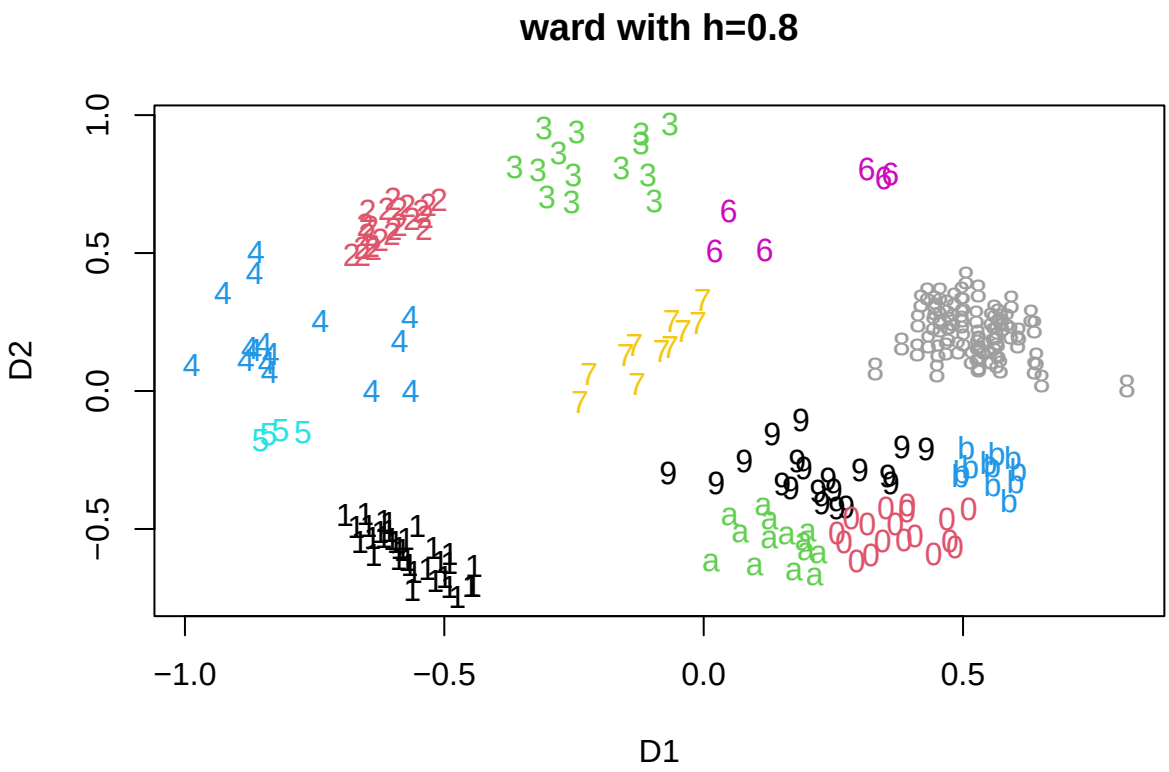
The clusters do not have sparse points as seen in K-Means.

From the dendrogram, we see that a higher number of clusters can be chosen as well. Let us try and see:

```
ward_tai2 <- hclust(dist(tai_mds$conf), method="ward.D2")
plot(ward_tai2, labels=FALSE)
rect.hclust(ward_tai2, h=0.8)
```




```
ward8_tai2 <- cutree(ward_tai2, h=0.8)
plot(tai_mds$conf,col=ward8_tai2,pch=clusym[ward8_tai2])
title(main="ward with h=0.8")
```



Visually, if we set $h = 0.8$, we get 12 clusters and they all look spherical and homogeneous.

Our final choice goes to $K = 12$ because, as we mentioned before, we already know that the number of pairs of genes is 13. Thus, setting the number of species of bees to 12 is very close to the desired outcome, while $K = 8$ is not.

Gaussian Mixure model

Hierarchical Clustering:

Pros:

- Interpretability: Hierarchical clustering provides a visual representation of the data in the form of a dendrogram, which makes it easy to understand the hierarchy of clusters and how data points are grouped.
- No need to specify the number of clusters: Hierarchical clustering does not require you to specify the number of clusters in advance. You can choose the number of clusters at a later stage by cutting the dendrogram at an appropriate level.
- Agglomerative and divisive approaches: Hierarchical clustering supports both agglomerative (bottom-up) and divisive (top-down) approaches, allowing you to choose the most suitable method for your data.
- Cluster linkage options: You can use different linkage criteria (e.g., single, complete, average) to control how clusters are merged, offering flexibility in cluster shape and size.

Cons:

- Computationally expensive: Hierarchical clustering can be computationally expensive, especially for large datasets, as it involves the pairwise comparison of all data points.
- Lack of scalability: It may not be suitable for very large datasets due to its high computational complexity.
- Sensitivity to noise: It can be sensitive to noise and outliers, leading to suboptimal cluster assignments.

Gaussian Mixture Models:

Pros:

- Soft clustering: GMMs allow for soft clustering, meaning each data point can belong to multiple clusters with different probabilities, providing a more nuanced view of cluster assignments.
- Model flexibility: GMMs are based on a probabilistic framework and can capture complex data distributions, allowing for more flexible cluster shapes.
- Efficient parameter estimation: Expectation-Maximization (EM) algorithm can efficiently estimate the model parameters (means, covariances, and mixing coefficients) for GMMs. Suitable for density estimation: GMMs can also be used for density estimation, which is useful for anomaly detection.

Cons:

- Sensitive to initialization: The performance of GMMs can be highly dependent on the initial parameter values, making it essential to use good initialization strategies.
- Requires specifying the number of components: Unlike hierarchical clustering, you need to specify the number of components (clusters) in advance for GMMs, which can be challenging when the true number of clusters is unknown.
- Local optima: EM algorithm can get stuck in local optima, so multiple initializations and variations of the algorithm may be required to improve results.
- May not handle non-Gaussian data well: GMMs assume that data in each cluster follows a Gaussian distribution, which may not hold true for all types of data.

Before applying the Gaussian Mixture model with the EM algorithm, it is important to note that there is no such a thing like the best clustering method. It all depends on the desired outcome and on the context.

```
set.seed(1234)
m_tai <- Mclust(tai_mds$conf,G=1:10)
summary(m_tai)
```

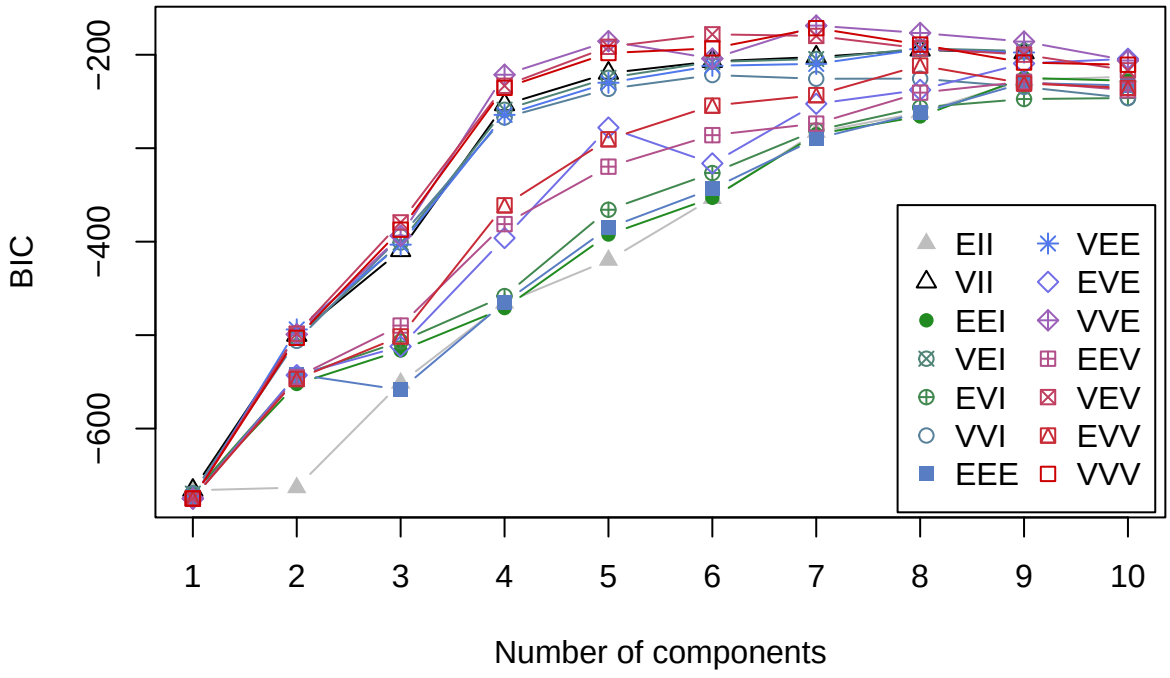
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVE (ellipsoidal, equal orientation) model with 7 components:
##
##  log-likelihood   n df       BIC       ICL
```

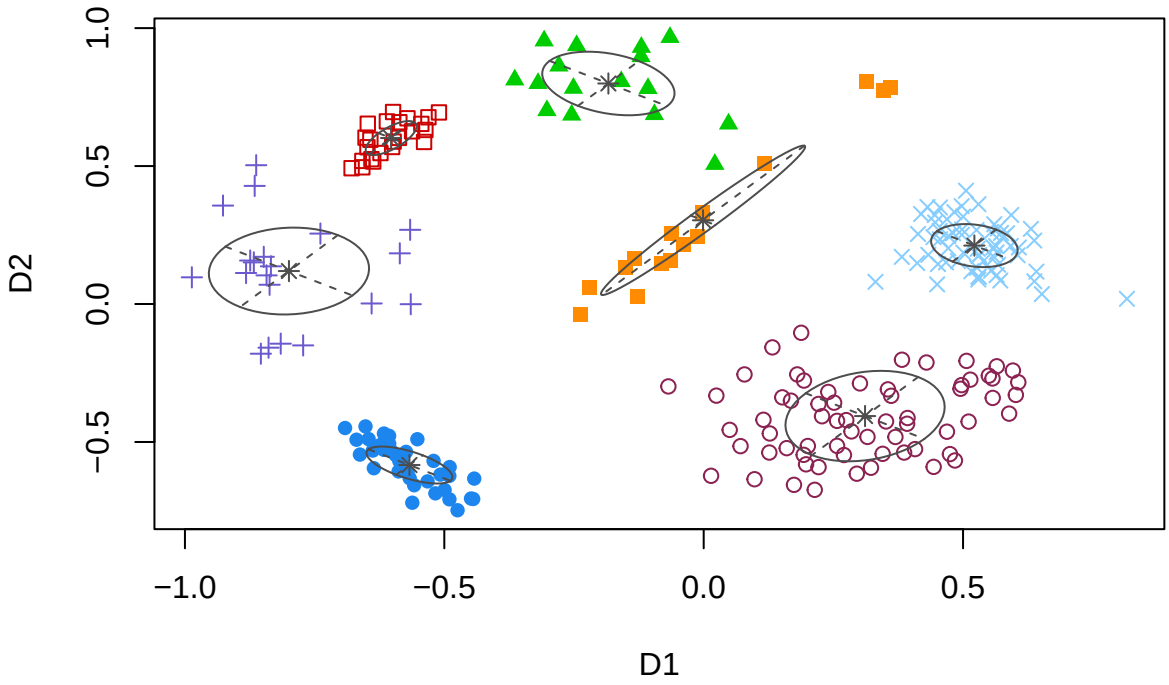
```
##        11.20011 236 35 -168.8339 -171.0657
##
## Clustering table:
##  1  2  3  4  5  6  7
## 35 23 16 20 15 63 64
```

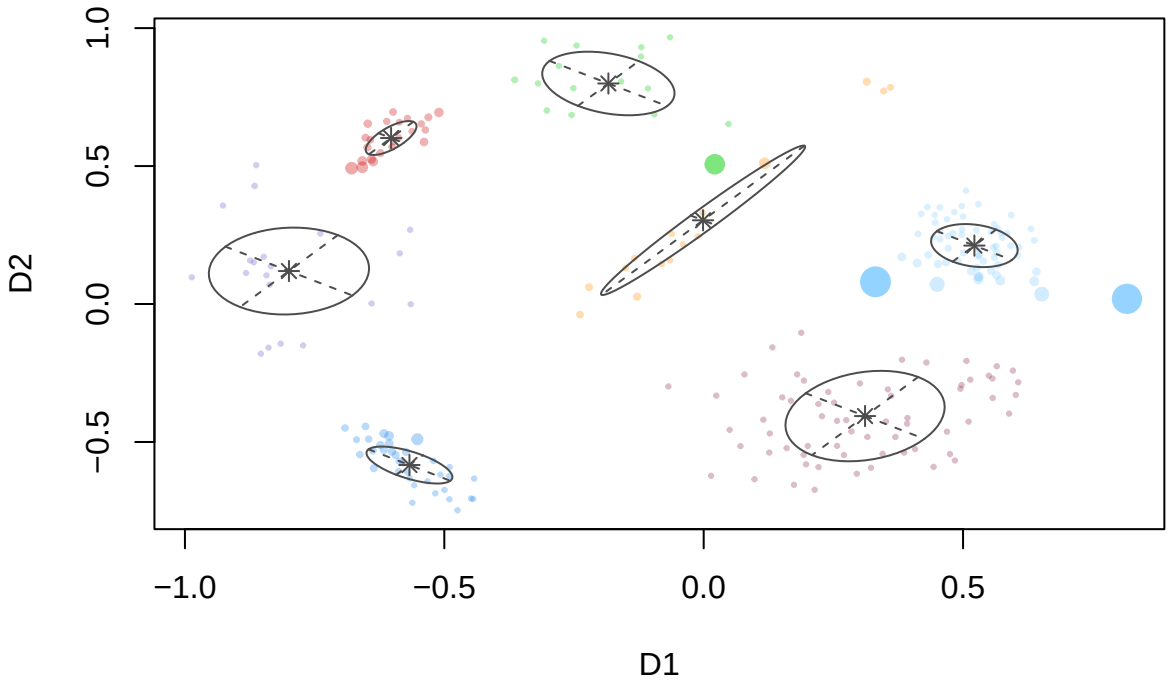
```
summary(m_tai$BIC)
```

```
## Best BIC values:
##              VVE,7       VVV,7       VVE,8
## BIC      -168.8339 -171.559535 -176.686823
## BIC diff    0.0000   -2.725638   -7.852926
```

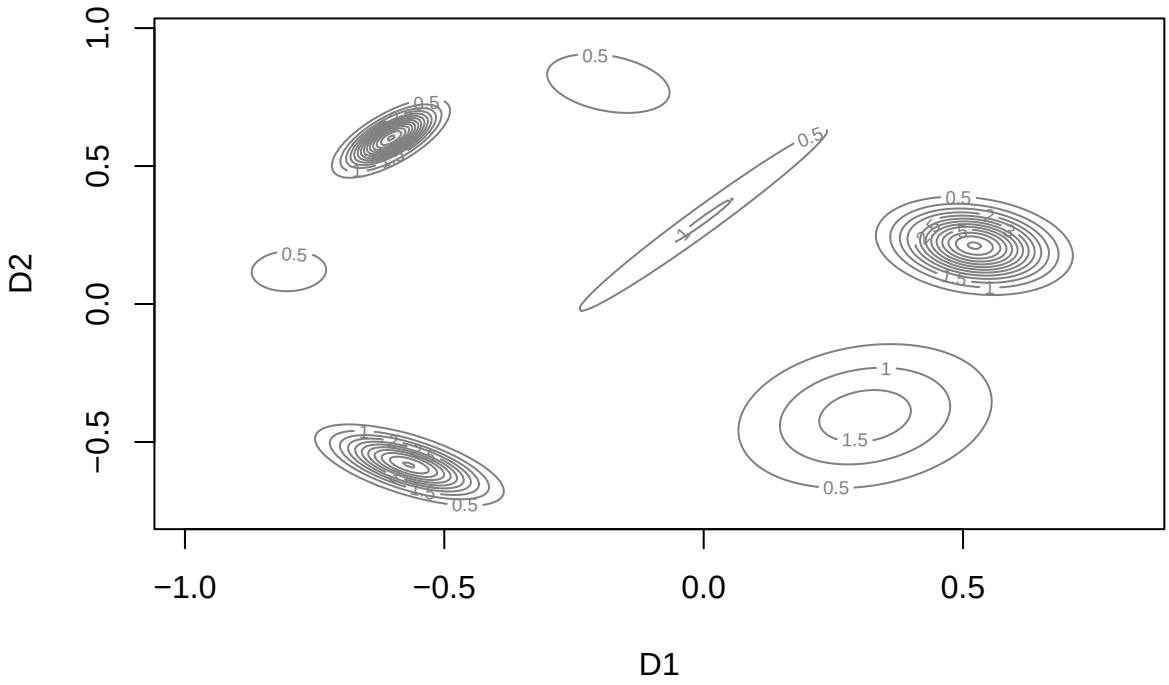
```
plot(m_tai)
```




D2
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
D1




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



As we can see from the plot of the BIC function vs the number of clusters, we notice that the chosen model is a "VVE" (i.e. the one with the highest value of BIC) with $K = 7$. Once again, thanks to the knowledge of the dominion of our data, we know that the ideal number of cluster is 13. Therefore we may discard this clustering.

Moreover, the plot showing the level curves of the gaussian distributions of the clusters reveals that all these clusters are well visible with an elliptical shape and distant from one to another.

Let us answer the last question. We may use can_mds$stress to check the loss of information caused by representing our data with multidimensional scaling.

```
tai_mds$stress
```

```
## [1] 0.2619208
```

26.19% is not an irrelevant loss of information, meaning that the 2D representation may not be enough. We check the stress level with 3D:

```
tai_mds2 <- mds(tai$distmat, ndim = 3)
tai_mds2$stress
```

```
## [1] 0.1685292
```

Here we can see that moving to 3 dimensions may be a good choice, since the loss information is only 16.85%.

2

Compute different clusterings of the data (use at least two different approaches including a Gaussian mixture model and try out numbers of clusters up to 10) and compare them first without using the information about benign vs. malign cancers in the diagnosis variable wdbcdiag (this also means you should ignore the information that there are 2 classes). Which clustering do you think is best?

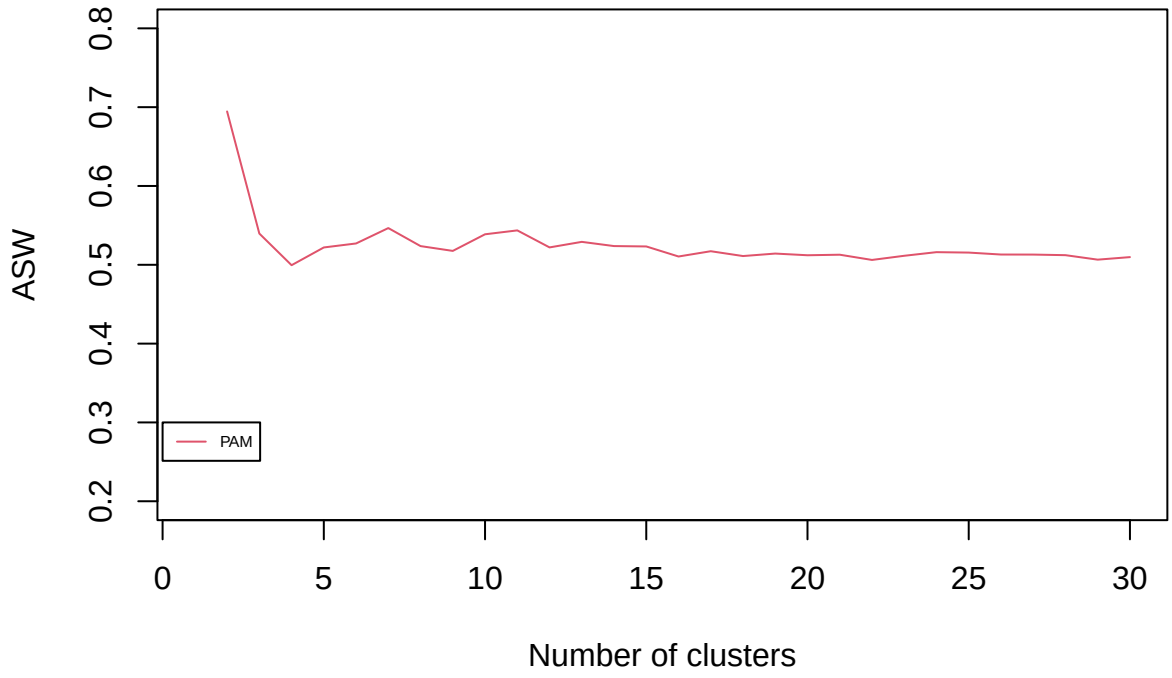
We upload the data.

```
# setwd("C:/Users/...")
wdbc <- read.csv("wdbc.data",header=FALSE)
# The variables I ask you to use are variables 3 to 12,
# so wbbcc is the data set to be clustered:
wdbcc <- wdbc[,3:12]
# There is also a diagnosis whether cancer is benign or malign as variable 2
# in the data set:
wdbcdiag <- as.integer(as.factor(wdbc[,2]))
```

We try 3 different clustering methods here:

- PAM
- K-Means
- Gaussian Mixture Model.

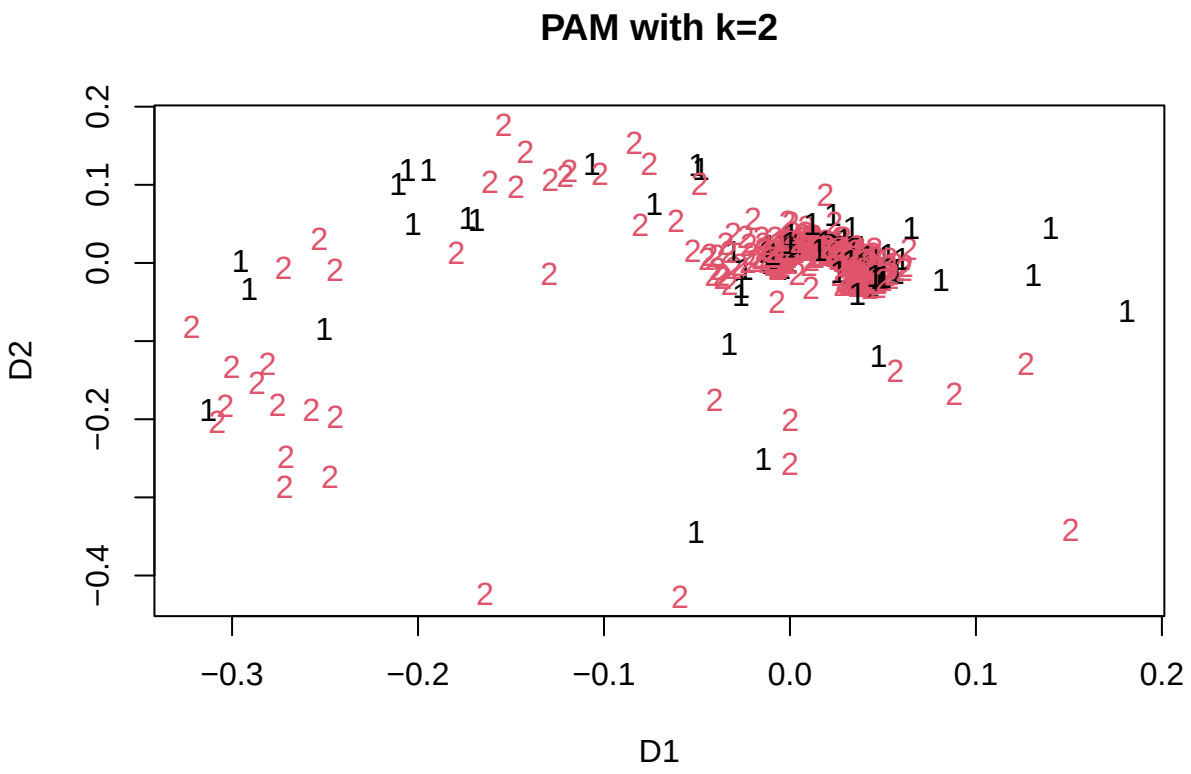
```
# PAM:
pam_sw <- NA
pam_clust <- list()
pam_sil <- list()
for (k in 2:30){
pam_clust[[k]] <- pam(wdbcc,k)
pam_sil[[k]] <- silhouette(pam_clust[[k]])
pam_sw[k] <- summary(pam_sil[[k]])$avg.width
}
plot(1:30,pam_sw,type="l",col = 2, xlab="Number of clusters",ylab="ASW",ylim=c(0.2,0.8))


# ...plots...
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] 2
```

```
# MDS-plot of the final clusterings:
can_mds <- mds(wdbcc)
plot(can_mds$conf,col=pam_clust[[2]]$cluster,pch=clusym[pam_clust[[2]]$cluster])
title(main="PAM with k=2")
```



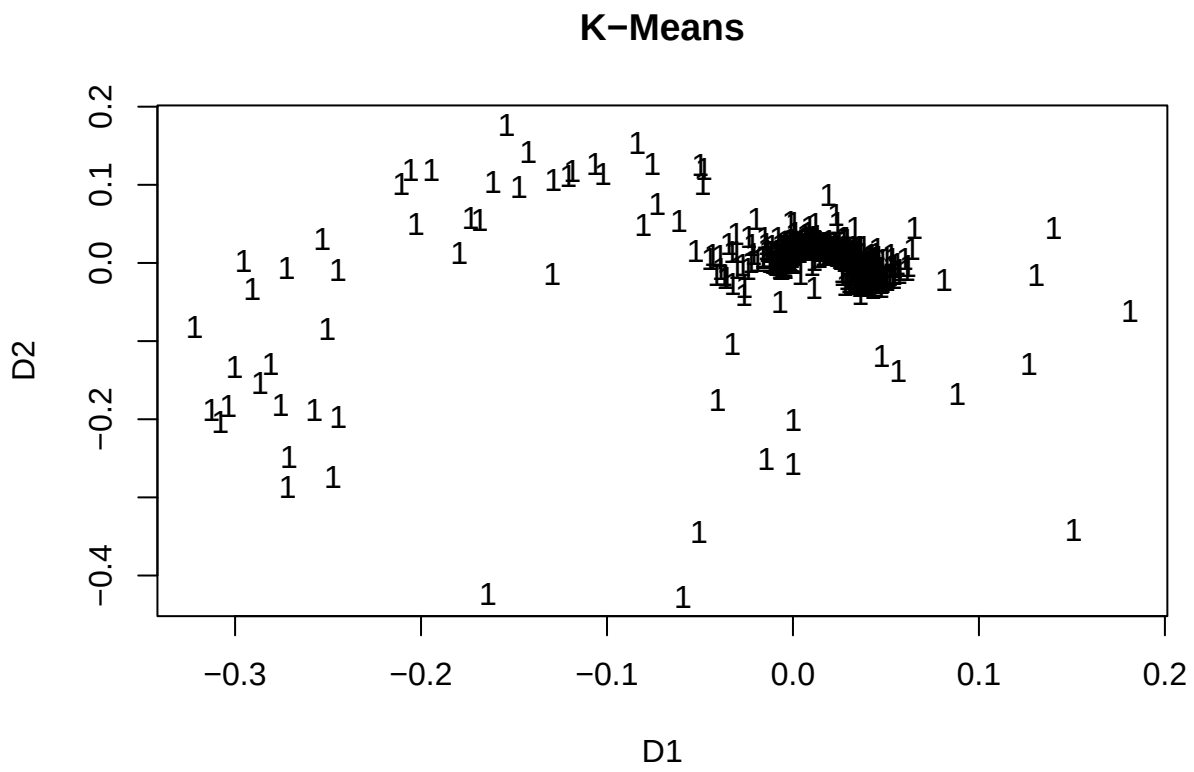
The ASW suggests to use $K = 2$ as often happens with this method. The assumption here is that there is no knowledge of the real clusters and we cannot choose our clustering as we did above.

The K-Means is computed as follows:

```
can_km <- gap(wdbcc, nstart=1)
can_km$nc
```

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

```
km_gap_can <- can_km$kmopt

plot(can_mds$conf,col=km_gap_can$cluster,pch=clusym[km_gap_can$cluster])
title(main="K-Means")
```



The K-Means seems to not work well with these data, since it is not able to detect any clusters at all. Points are all assigned to one single cluster. Perhaps the real number of cluster is not that big.

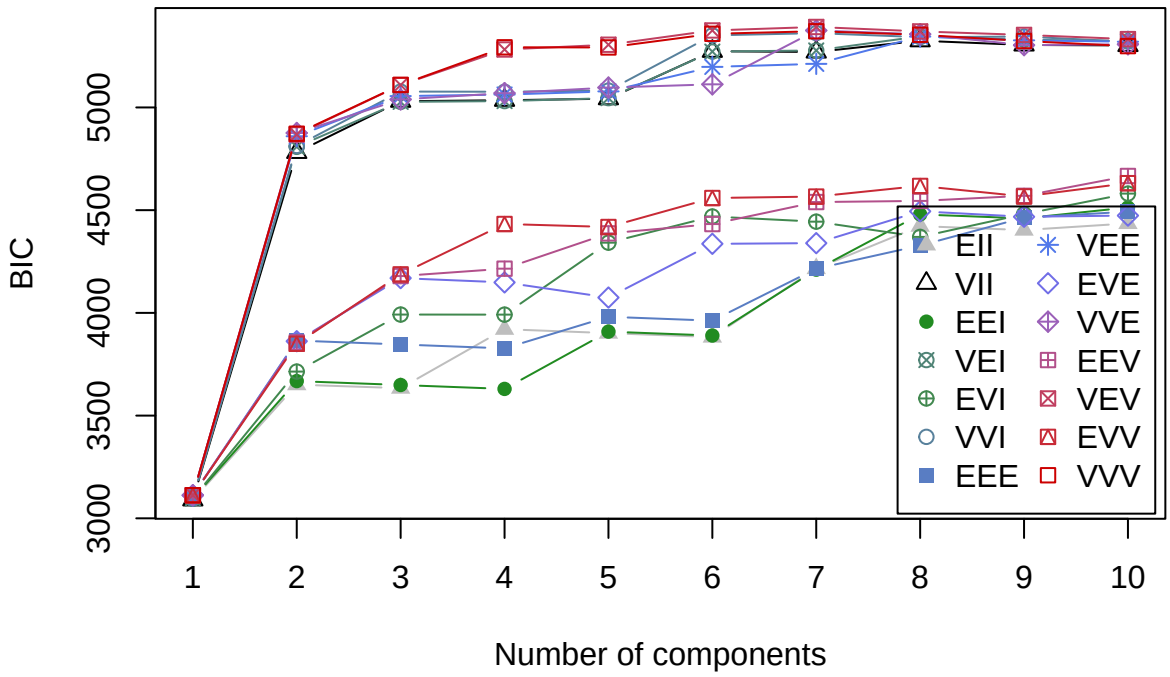
```
set.seed(1234)
m_can <- Mclust(can_mds$conf,G=1:10)
summary(m_can)
```

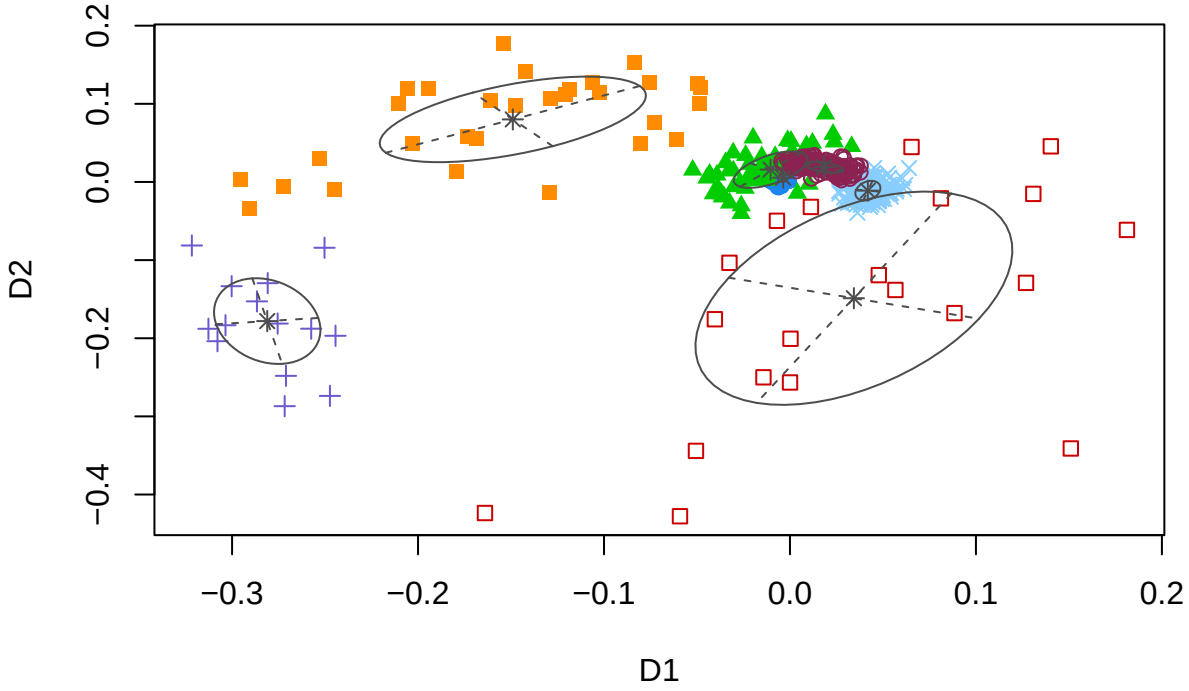
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 7 components:
##
##  log-likelihood   n df      BIC      ICL
##         2806.87 569 35 5391.705 5301.872
##
## Clustering table:
##   1   2   3   4   5   6   7
## 138  20  55  14  30 146 166
```

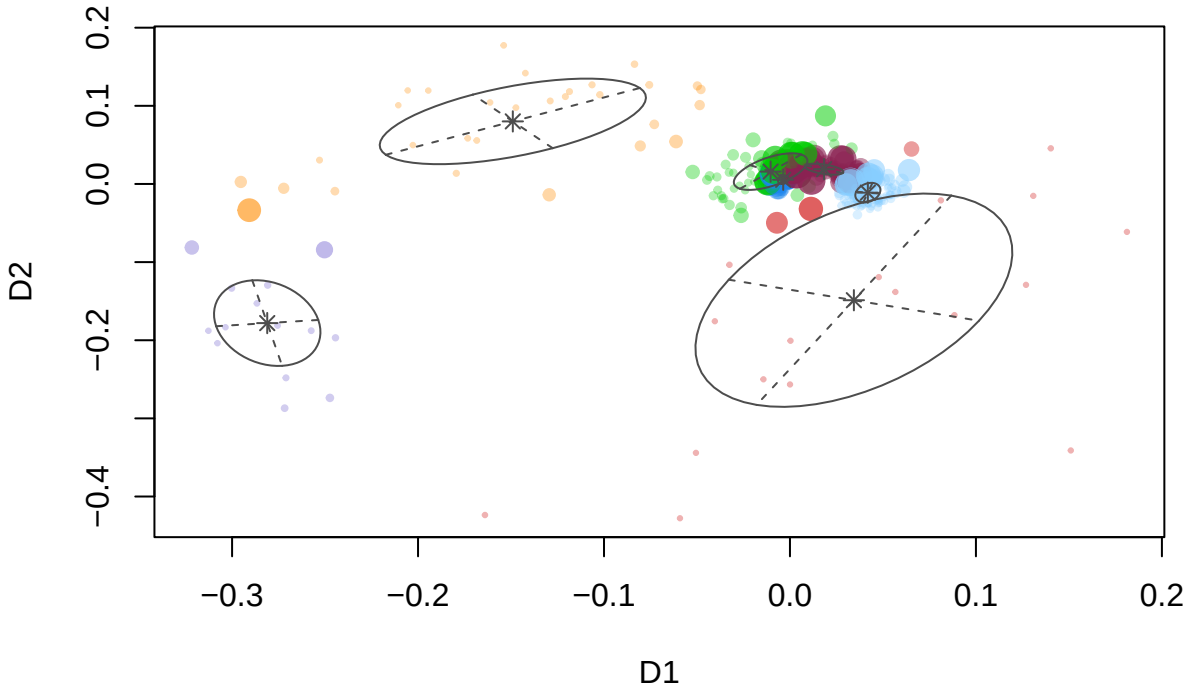
```
summary(m_can$BIC)
```

```
## Best BIC values:
##             VEV,7      VEV,6     VVE,7
## BIC      5391.705 5374.78317 5374.7628
## BIC diff    0.000  -16.92142  -16.9418
```

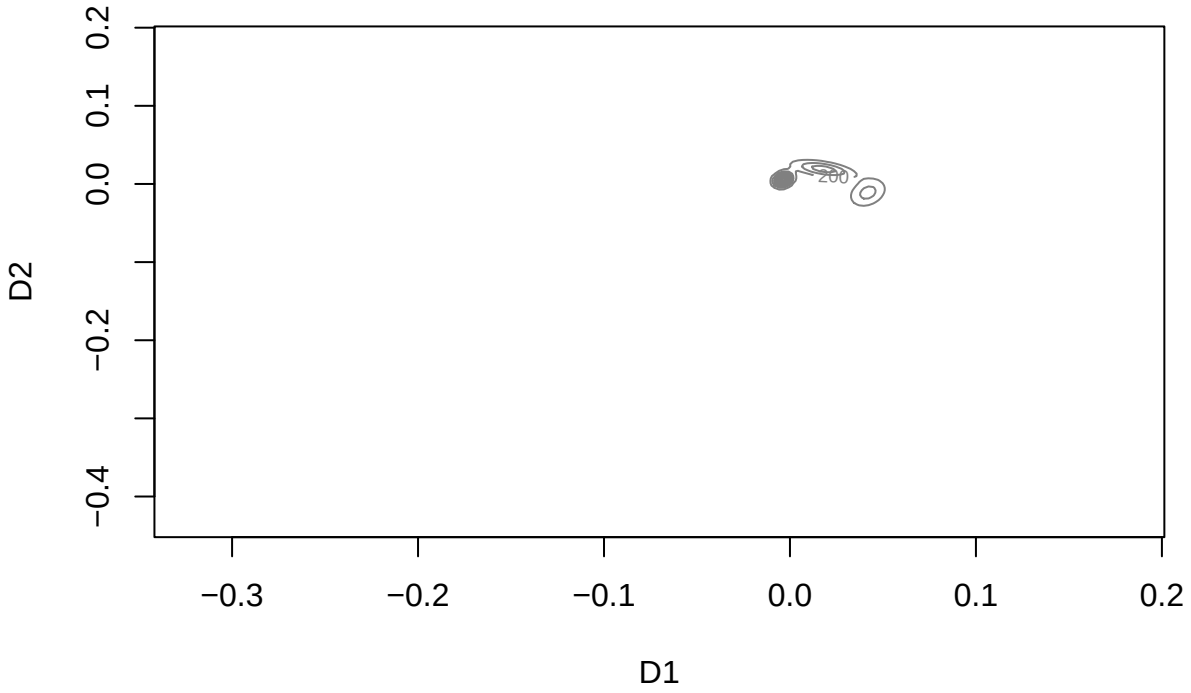
```
plot(m_can)
```


BIC
3000
3500
4000
4500
5000
1
2
3
4
5
6
7
8
9
10
Number of components
EII
VII
EEI
VEI
EVI
VVI
EEE
VEE
EVE
VVE
EEV
VEV
EVV
VVV


D2
0.2
0.1
0.0
−0.2
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
D1




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



The GMM suggests to use $K = 7$ with a "VEV" (i.e. ellipsoidal, equal shaped model).

Since the GMM requires certain assumptions to be working correctly and we assume to not know anything about the underlying distributions of the data and the real clusters, we decide to go with the PAM clustering because it is a non parametric method. Furthermore, from the density plot with level curves, only 3 clusters are plotted, revealing that $K = 7$ may be wrong. By checking the amount of level curves, there is just one high density peak and a second one which is moderate. The third one has a low density. This is probably another good hint that the real number of clusters might be $K = 2$ or $K = 3$.

For the above mentioned reasons, the PAM clustering with $K = 2$ has been chosen as the final clustering.

Let us apply the Adjusted Rand Index (ARI) between the real clusters and the ones chosen by our model.

```
adj.rand.index(pam_clust[[2]]$clustering, wdbcdiag)
```

```
## [1] 0.4513131
```

The index reveals a moderate similarity between the two, showing that our clustering has been able to "predict" the malign and benign tumors to a certain degree.

3

Task: With given $c(1), \ldots, c(n)$ and $\hat{\mathbf{m}}_k = \frac{1}{n_k} \sum_{c(i)=k} \mathbf{x}_i, n_k = |C_k|$,

$$S\left(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K\right) = \sum_{i=1}^{n} d_{L2}^2 \left(\mathbf{x}_i, \mathbf{m}_{c(i)}\right).$$

Show that

$$S\left(\mathcal{C}, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_K\right) = \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C_k} d_{L2}^2\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

Solution Let us formulate the first expression as follows:

$$\begin{aligned} S\left(\mathcal{C}, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_K\right) &= \sum_{i=1}^{n} d_{L2}^2\left(\mathbf{x}_i, \hat{\mathbf{m}}_{c(i)}\right) \\ &= \sum_{k=1}^{k} \sum_{c(i)=k} \left\|\mathbf{x}_i - \hat{\mathbf{m}}_k\right\|^2 \end{aligned}$$

Now, by the definition of the squared Euclidean distance, we get:

$$\begin{aligned} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} d_{L2}^2\left(\mathbf{x}_i, \mathbf{x}_j\right) &= \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 \\ = \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} \left\|\mathbf{x}_i - \hat{\mathbf{m}}_k\right\|^2 + \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} \left\|\hat{\mathbf{m}}_k - \mathbf{x}_j\right\|^2 + 2 \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} \left(\mathbf{x}_i - \hat{\mathbf{m}}_k\right)\left(\hat{\mathbf{m}}_k - \mathbf{x}_j\right) \\ = 2 \sum_{\mathbf{x}_j \in C(k)} \sum_{\mathbf{x}_i \in C(k)} \left\|\mathbf{x}_i - \hat{\mathbf{m}}_k\right\|^2 \\ = 2\ n_k \sum_{\mathbf{x}_i \in C(k)} \left\|\mathbf{x}_i - \hat{\mathbf{m}}_k\right\|^2 \end{aligned}$$

Hence:

$$\sum_{\mathbf{x}_i \in C(k)} \left\|\mathbf{x}_i - \hat{\mathbf{m}}_k\right\|^2 = \frac{1}{2n_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} d_{L2}^2\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

Finally, we obtain:

$$\begin{aligned} S\left(\mathcal{C}, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_K\right) &= \sum_{i=1}^{n} d_{L2}^2\left(\mathbf{x}_i, \hat{\mathbf{m}}_{c(i)}\right) \\ &= \sum_{k=1}^{k} \sum_{c(i)=k} \left\|\mathbf{x}_i - \hat{\mathbf{m}}_k\right\|^2 \\ &= \sum_{k=1}^{k} \frac{1}{2n_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C(k)} d_{L2}^2\left(\mathbf{x}_i, \mathbf{x}_j\right) \end{aligned}$$

4

a) Focusing on Complete Linkage clustering, what is the best distance in these experiments (including the standardisation method)? Explain how this can be seen from the plots regarding the Complete Linkage results (i.e., explain roughly what these plots show).

The plots shown in the paper represent the ARI on different types of standardisations between the estimated clusters and the true clusters listed as follows:

- Median Absolute Deviation (MAD)
- Unit Variance
- Unit Range
- Box Plot Transformation
- None.

By looking at them, it is obvious that the city block distance (L1) is the best of all.

As mentioned in the paper: "It is hardly ever beaten; only for PAM and complete linkage with range standardisation clustering in the simple normal (0.99) setup (Figure 3) and PAM clustering in the simple normal setup (Figure 2) some others are slightly better. Actually L1-aggregation delivers a good number of perfect results (i.e., ARI 1 or misclassification rate 0). This is in line with [7], who state that "the L1-metric is the only metric for which the absolute difference between nearest and farthest neighbor increases with the dimensionality." ".

b) Discuss how Complete Linkage and Partitioning Around Medoids clustering compare overall

By looking at the City Block distance, overall the ARI computed on complete linkage has higher low peaks and high peaks than the one computed with PAM despite some exceptions.

For instance PAM in the "normal, t and noise (0.1)" layout performs better with "MAD" and "none". There are also some severe drops in the "normal, t and noise (0.5)" and "normal, t and noise (0.9)" layouts layout once again with "MAD" and "none". This happens probably because the introduction of noise drastically reduces the performance of hierarchical clustering with complete linkage on unstandardised data and on stardardisation accomplished with MAD.

As stated in the paper: " Standardisation methods based on the central half of the observations such as MAD and boxplot transformation may suffer in presence of small classes that are well separated from the rest of the data on individual variables."

When it comes to the other distances (i.e. L2,L3,L4 and max), the performance of hierarchical clustering is usually worse than in PAM with some exceptions.

For example in the simple normal model L2,L3,L4 are equal to circa 0 for all types of standardisation, while in PAM only L2 is almost constant around 0, while in simple normal with noise=0.99 we notice how L2,L3,L4 perform better with unite range in Compl. Link and the rest is kind of the same (but L2 is sligthly better in complete linkage in "var").

By looking at the simple normal plot with noise = 0.1, we notice how in Comp.Link. L2,L3,L4 and Max are costant around 0, while in PAM they assume much higher values of ARI (except for L2 which is still constant around 0). The same things can be said for simple normal with noise 0.5.

The interesting comparison between PAM and Comp.Link is with the simple normal plot with noise = 0.9. In this scenario, L4 and Max behave exactly the same in both cases. However, L2 is better with unit variance standardisation in PAM, yet L2 and L3 perform better with Range in Comp.Link.