Exercise 8

Riccardo Petrella

2023-12-06

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)
library(poLCA)
library(fda)
library(dplyr)
library(funFEM)
library(splines)
```

1

Represent the data in terms of a suitable B-spline basis. Show how well two exemplary observations are approximated in this way. Run a functional principal component analysis, run a funFEM-clustering, and for comparison run a cluster analysis of your choice on the functional principal component scores. Visualise your results suitably, and compare the clustering results with the true phonem classes. You are asked to make your own decisions about tuning choices such as the size of the B-spline basis, number of principal components etc.

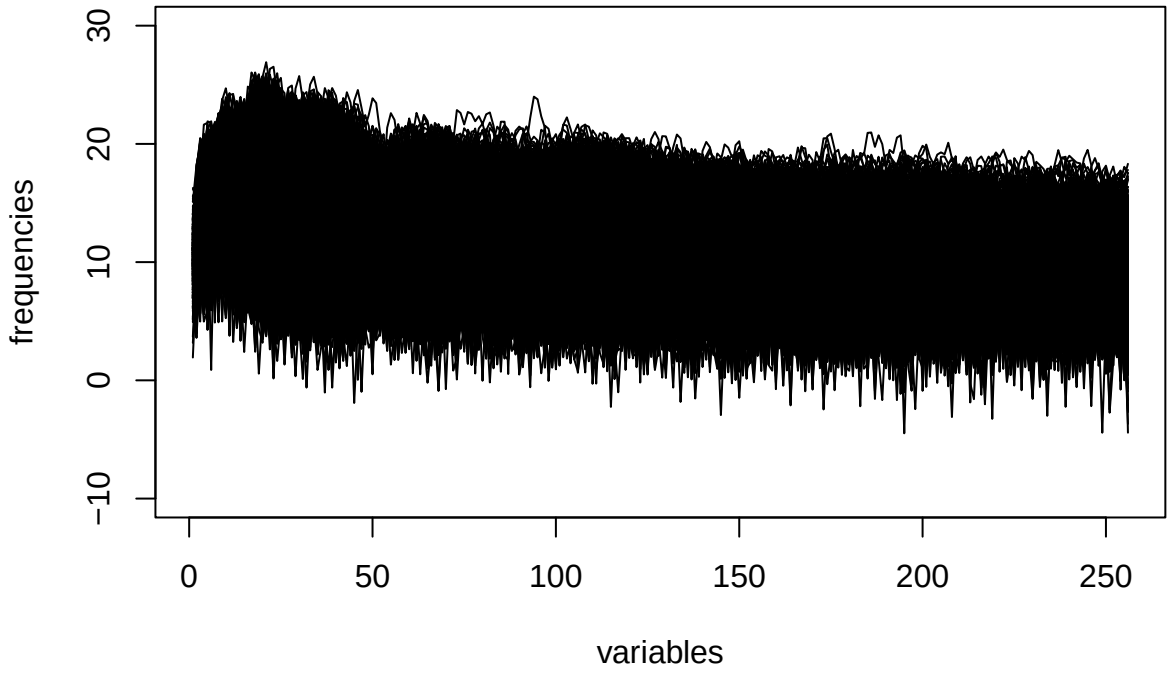
Firstly, we upload the data, we create the vector of true clusters and then we plot the functional data.

```
#setwd("C:/Users/...)

phonemes1000 <- read.table("phonemes1000.dat",header=TRUE)

phonemes256 <- as.matrix(phonemes1000[,1:256])
phonemes257 <- phonemes1000[,257]
```

```r
levels(as.factor(phonemes257))
```

```
## [1] "aa"  "ao"  "dcl" "iy"  "sh"
```

```r
true_clusters <- as.numeric(as.factor(phonemes257))


plot(1:256, phonemes256[1,1:256], type = "l", ylim=c(-10,30),
     xlab = "variables", ylab = "frequencies")
for(i in 2:1000) # the remaining observations
  points(1:256,phonemes256[i,],type="l")
```



As expected, the plot is pointless since 1000 observations (functions) all plotted on the same chart are completely overlapped.

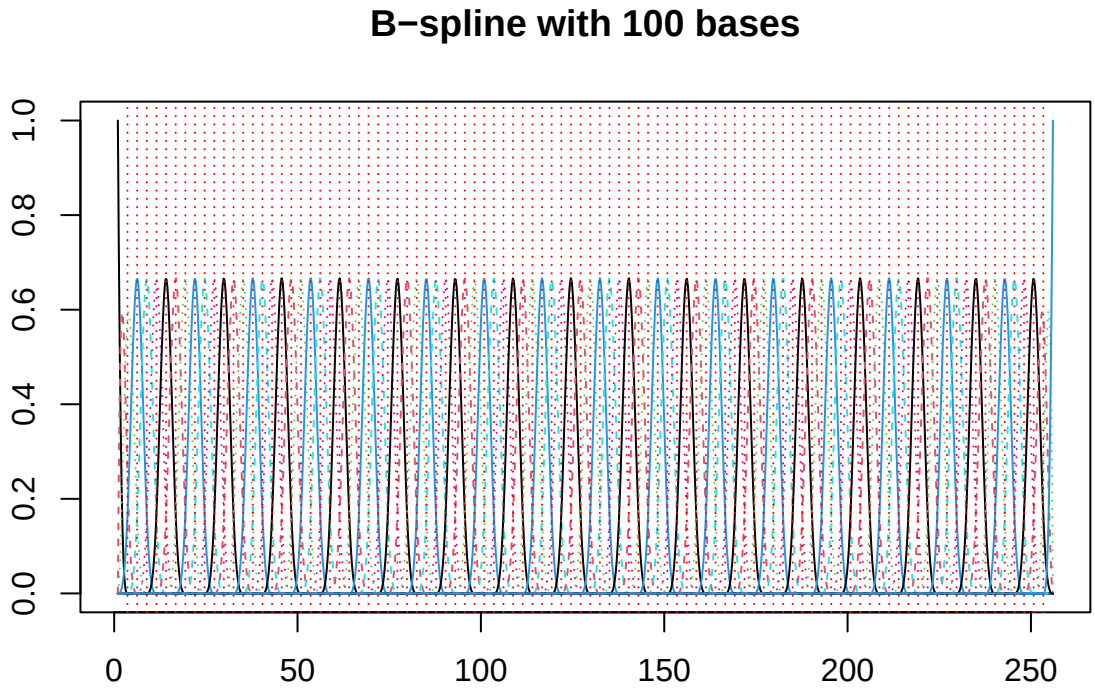
Let us compute the B-Splines as follows.

Before that, we need to choose the number of bases p.

- p too large: high dimensional computations, maybe unstable.
- p too small: functions X_i are not well represented/approximated.

We try with $p = 100$.

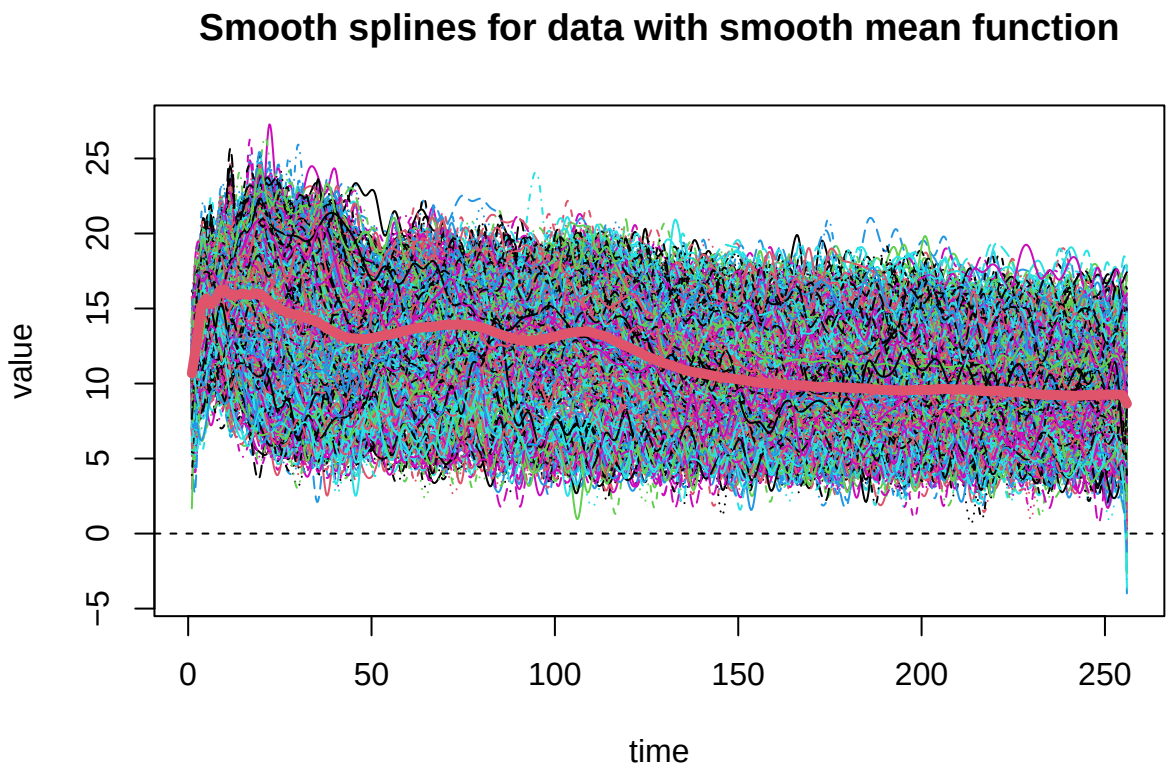
```r
bbasis <- create.bspline.basis(c(1,256),nbasis=100)
fd_pho <- Data2fd(1:256,y=t(as.matrix(phonemes256)),basisobj=bbasis)

# Plot basis
plot(bbasis)
title(main= "B-spline with 100 bases")
```



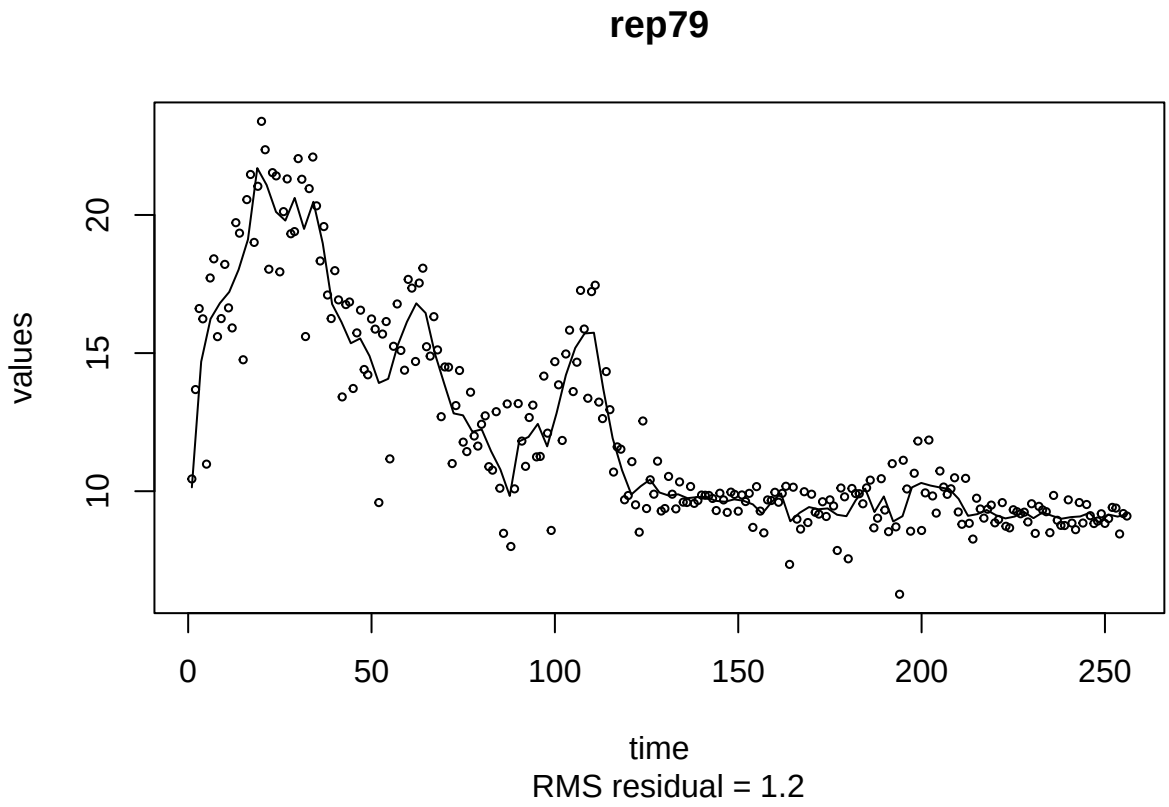

```r
plot(fd_pho)
```

```
## [1] "done"
```

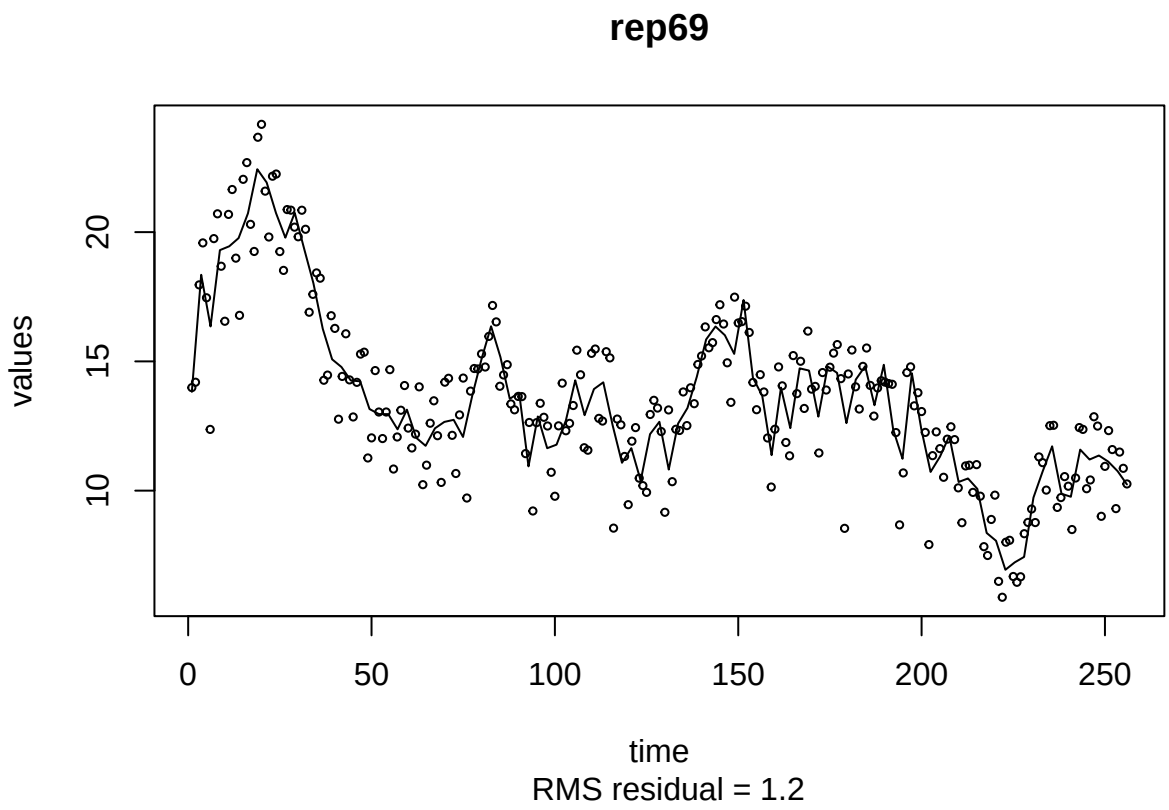
```r
m_pho <- mean.fd(fd_pho)
lines(m_pho,col=2,lwd=5)
title(main = "Smooth splines for data with smooth mean function")
```



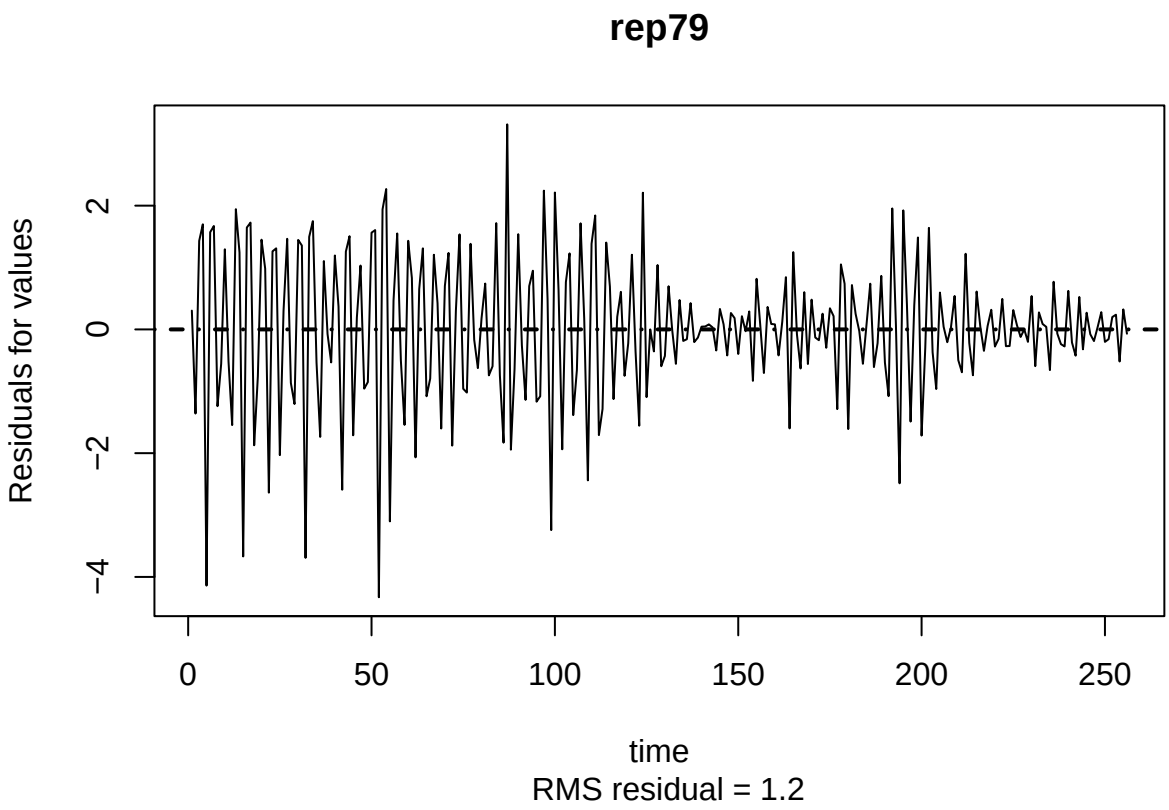

```
# Show residual plots of individual observations
plotfit.fd(t(phonemes256),1:256,fd_pho,index=79,cex.pch=0.5)
```



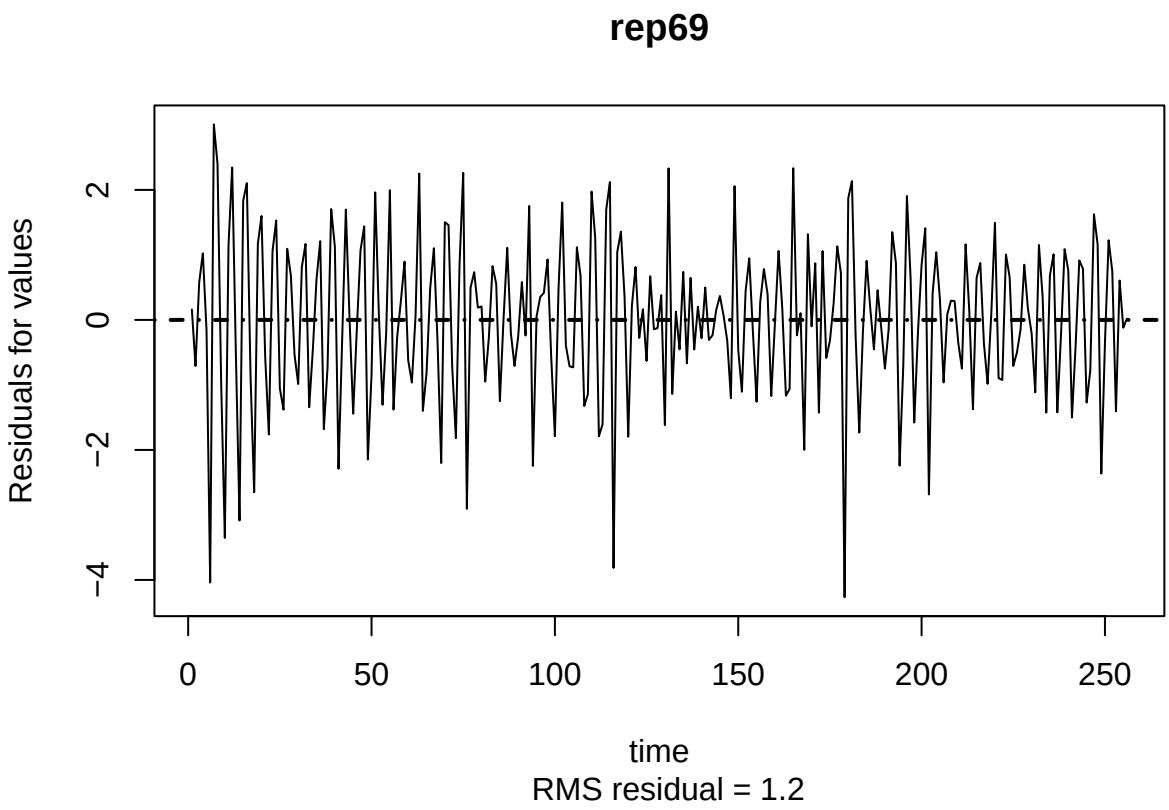

```
plotfit.fd(t(phonemes256),1:256,fd_pho,index=69,cex.pch=0.5)
```




```
plotfit.fd(t(phonemes256),1:256,fd_pho,index=79,residual = T,cex.pch=0.5)
```




```
plotfit.fd(t(phonemes256),1:256,fd_pho,index=69,residual = T,cex.pch=0.5)
```



Remark: Since from the plots "rep69" and "rep79" (i.e. two observations randomly taken as example) we notice that the splines correctly approximate the real observations, I have decided to keep the number of basis equal to 100 as used before in the lectures.

If we look at the residual plots, we notice that the values of the residuals are between -4 and +2 in both the observations. As the values along the x-axis increase, the variability of the residuals seems to decrease. This detail is also noticeable from the smoothing splines plot with the mean (i.e. the red line) because the functions are less variable.

We now proceed with the computation of the functional principal component analysis.

Remark: I decided to set the number of components equal to 5 to facilitate the comparision with the clustering accomplished with "FunFEM" as 5 is the number of clusters chosen by it.

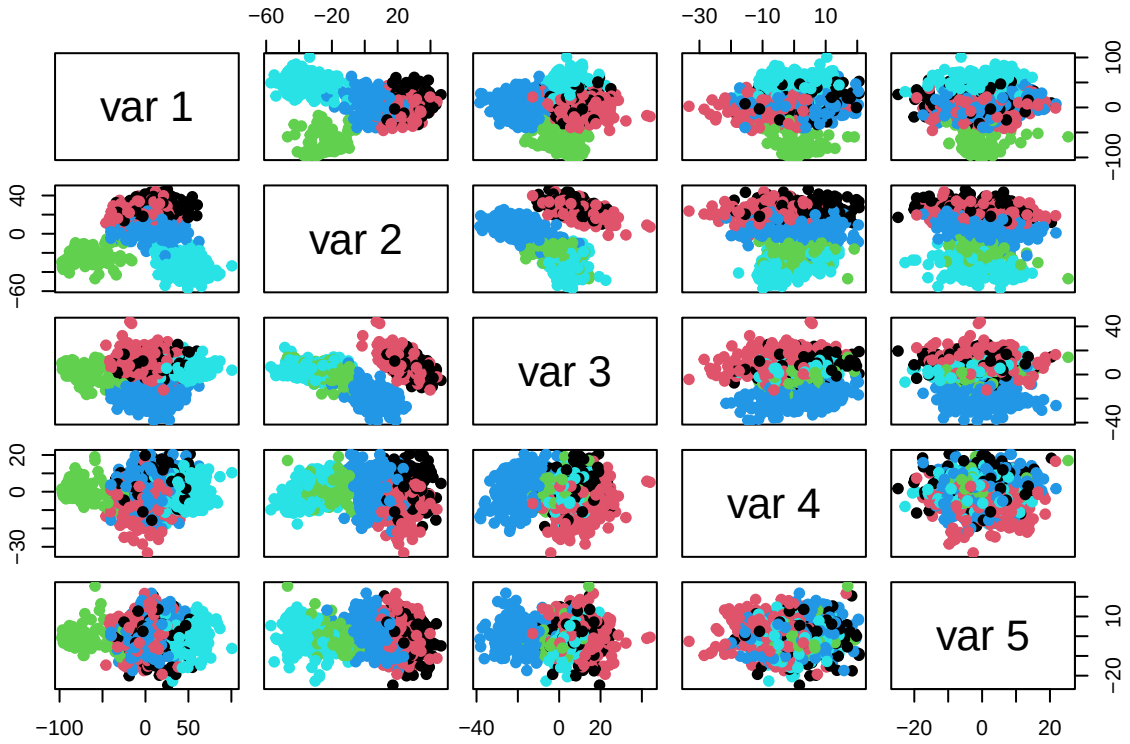
```
pho_pca <- pca.fd(fd_pho, nharm = 5)


pho_pca$varprop # Percentage of variance
```

```
## [1] 0.57630155 0.20367882 0.05904273 0.02137439 0.01436779
```

```
cumsum(pho_pca$varprop) # Cumulative percentage of variance
```

```
## [1] 0.5763016 0.7799804 0.8390231 0.8603975 0.8747653
```

```r
pairs(pho_pca$scores,col=true_clusters,pch=19)
```



Remark: The pairsplot of the first 5 components labeled with the true clusters reveals that the scores are naturally grouped in different clusters without even performing a certain clustering.

Here is attached the clustering with funFEM:

```r
pho_cluster <- funFEM(fd_pho,K=2:5)

# Try out all models and find the best.
# Unfortunately just submitting model="all" to funFEM doesn't work

set.seed(1234567)
femmodels <- c("DkBk", "DkB", "DBk",
"DB", "AkjBk", "AkjB", "AkB", "AkBk", "AjBk", "AjB", "ABk",
"AB")
nmodels <- length(femmodels)
femresults <- list() # Save output for all models in femmodels
bestk <- bestbic <- numeric(0)
# bestk: vector of best K for each model.
# bestbic: Best BIC value for each model.
K=2:5 # Numbers of clusters K to try out.
fembic <- matrix(NA,nrow=nmodels,ncol=max(K))
# fembic will hold all BIC values for models (rows) and K (columns);
# NA for those that cannot be fitted.
for (i in 1:nmodels){ # This takes a long time!!
print(femmodels[i])
```

```
femresults[[i]] <- funFEM(fd_pho,model=femmodels[i],K=K)
fembic[i,K] <- femresults[[i]]$allCriterions$bic
bestk[i] <- which(fembic[i,]==max(fembic[i,K],na.rm=TRUE))
bestbic[i] <- max(fembic[i,K],na.rm=TRUE)
}
```

```
## [1] "DkBk"
## [1] "DkB"
## [1] "DBk"
## [1] "DB"
## [1] "AkjBk"
## [1] "AkjB"
## [1] "AkB"
## [1] "AkBk"
## [1] "AjBk"
## [1] "AjB"
## [1] "ABk"
## [1] "AB"
```

```
besti <- which(bestbic==max(bestbic,na.rm=TRUE))
```

```
besti  # 9th model
```

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

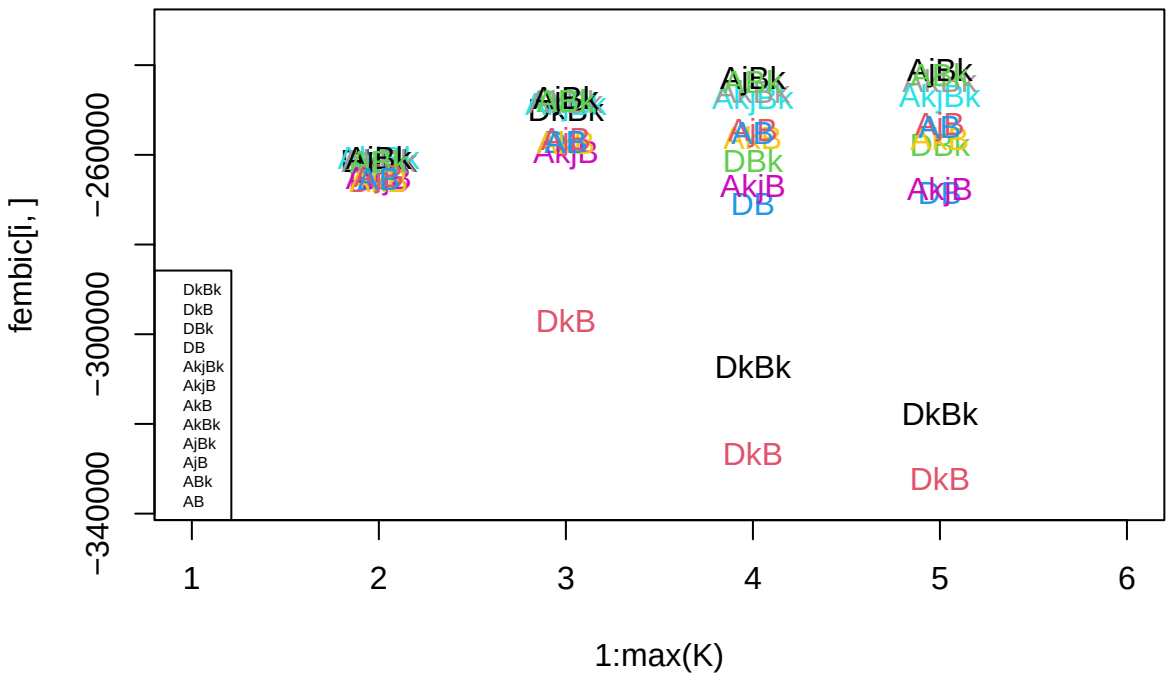
```
femmodels[besti] #"AjBk"
```

```
## [1] "AjBk"
```

```
# Sigma_j is diagonal and can vary between variables. beta_k can vary between clusters.


bestk # K optimal = 5 for "AjBk"
```

```
##  [1] 3 2 3 3 5 3 4 5 5 5 5 5
```

```
# Can reproduce results faster (up to random initialisation):
# femresult9 <- funFEM(fd_pho,model="AjBk",K=5)
femresult9 <- femresults[[9]]
# Plot BIC values for all models and K:
i <- 1
plot(1:max(K),fembic[i,],col=i,pch=i,
  xlim = c(1,6),
  ylim=c(min(fembic,na.rm=TRUE)-15000,max(fembic,na.rm=TRUE))+10000,type ="n")
for(i in 1:nmodels)
  text(1:max(K),fembic[i,],femmodels[i],col=i)
legend("bottomleft", legend = femmodels, col =nmodels, cex=0.5)
```



As written in the comments of the above chunk of code, the best model is "AjBk" with 5 clusters: Sigma_j is diagonal and can vary between variables. beta_k can vary between clusters.

Remark: The higher the BIC the better the model.

We compare the clusters from FunFEM with the true clusters using the adjusted Rand Index.

```
# Clustering is in cls component of output.
table(femresult9$cls,true_clusters)
```

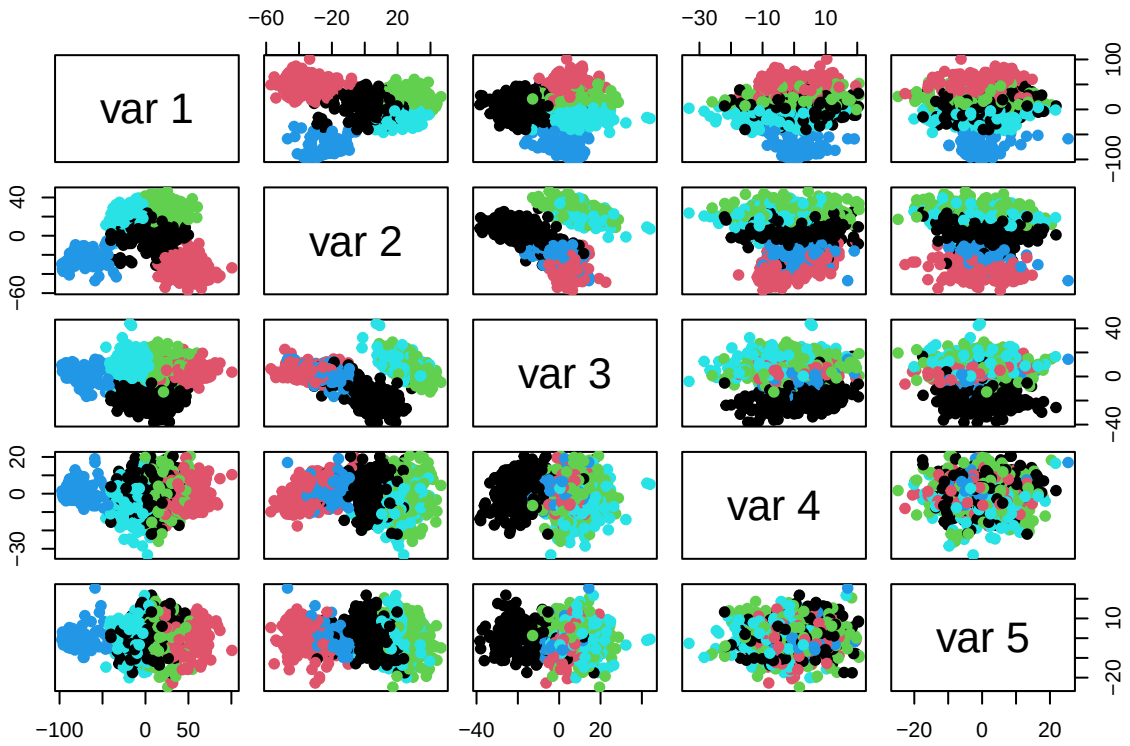
```
##    true_clusters
##       1   2   3   4   5
##   1   4   5   8 254   6
##   2   0   0   0   3 189
##   3 123  79   0   0   0
##   4   0   0 152   0   0
##   5  33 142   1   1   0
```

```
adjustedRandIndex(femresult9$cls,true_clusters)
```

```
## [1] 0.7447376
```

74.47% shows that our clustering model is very good !

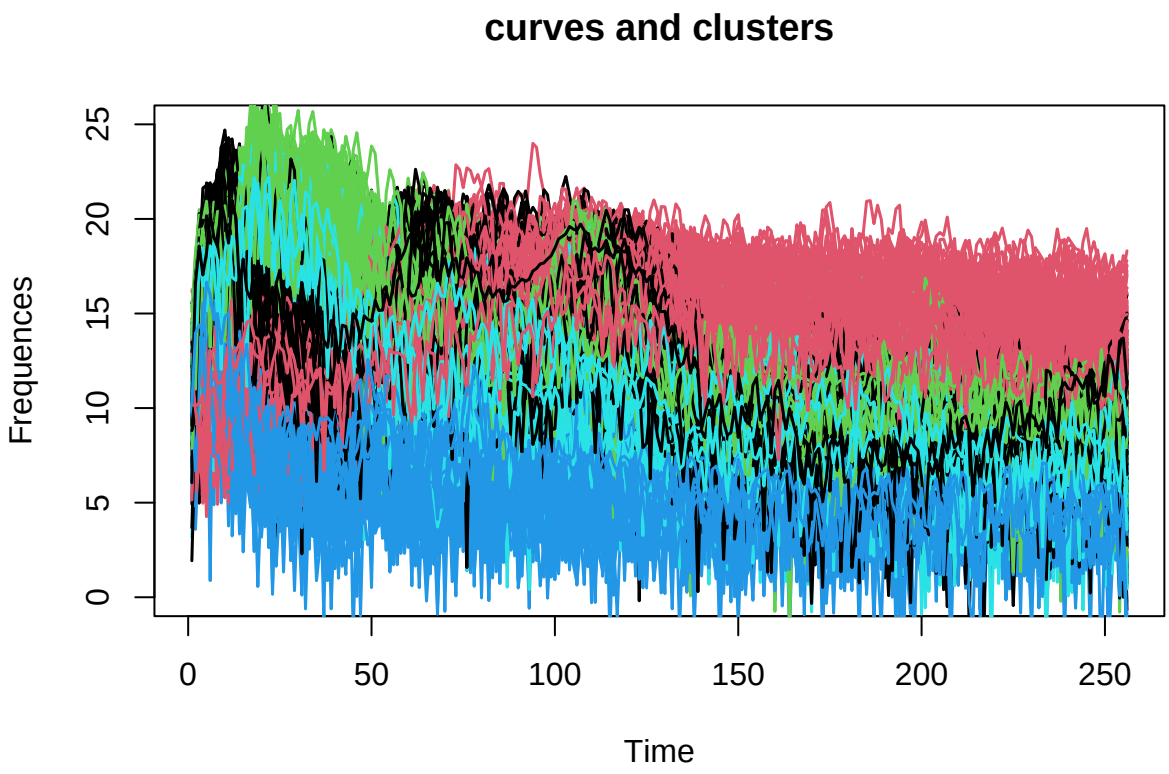
```
# Clusters on principal components
pairs(pho_pca$scores,col=femresult9$cls,pch=19)
```



In the pairsplot with the scores of the principal components, the units labeled with the clusters from FunFEM are pretty homogeneous.

Moreover, the chunk of code below plots the functional data clustered with FunFEM.

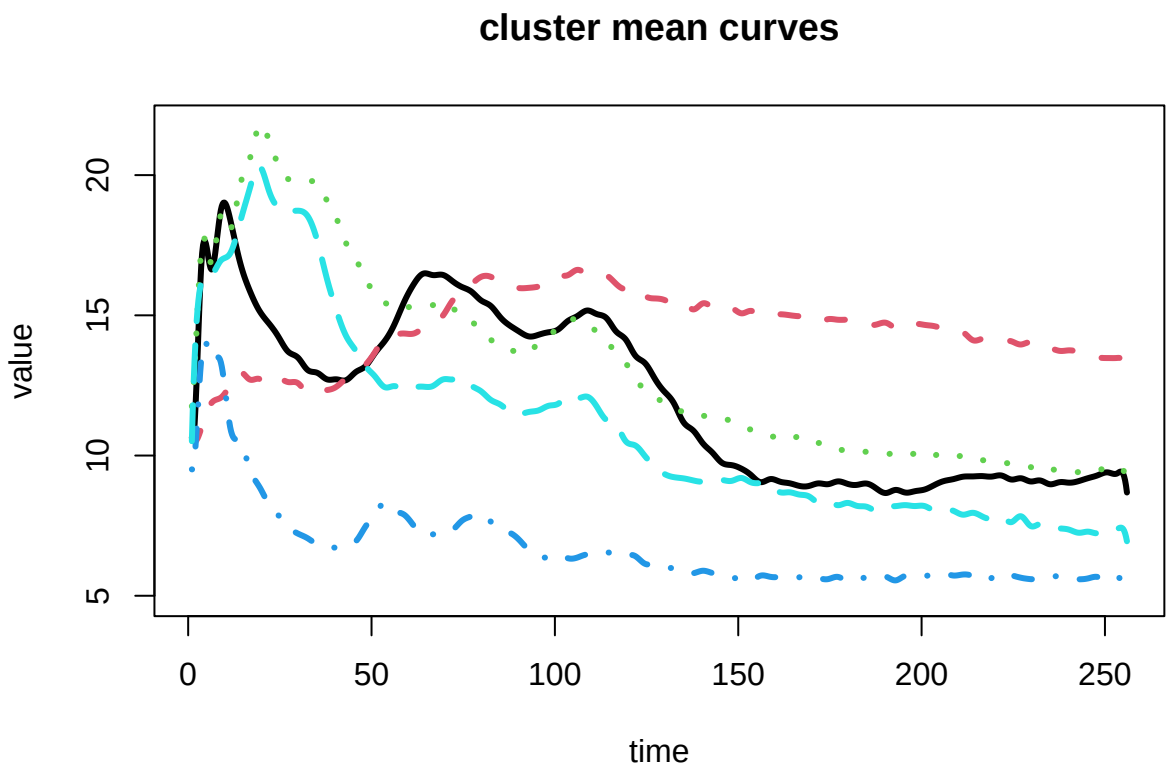
```
# Plot the curves and clusters
plot(1:256,phonemes256[1,],type="l",ylim=c(0,25),
  ylab="Frequences",
  xlab="Time",lwd=1.5,col=femresult9$cls[1])

for(i in 2:1000)
  points(1:256,phonemes256[i,],type="l",col=femresult9$cls[i],lwd=1.5)
title(main="curves and clusters")
```



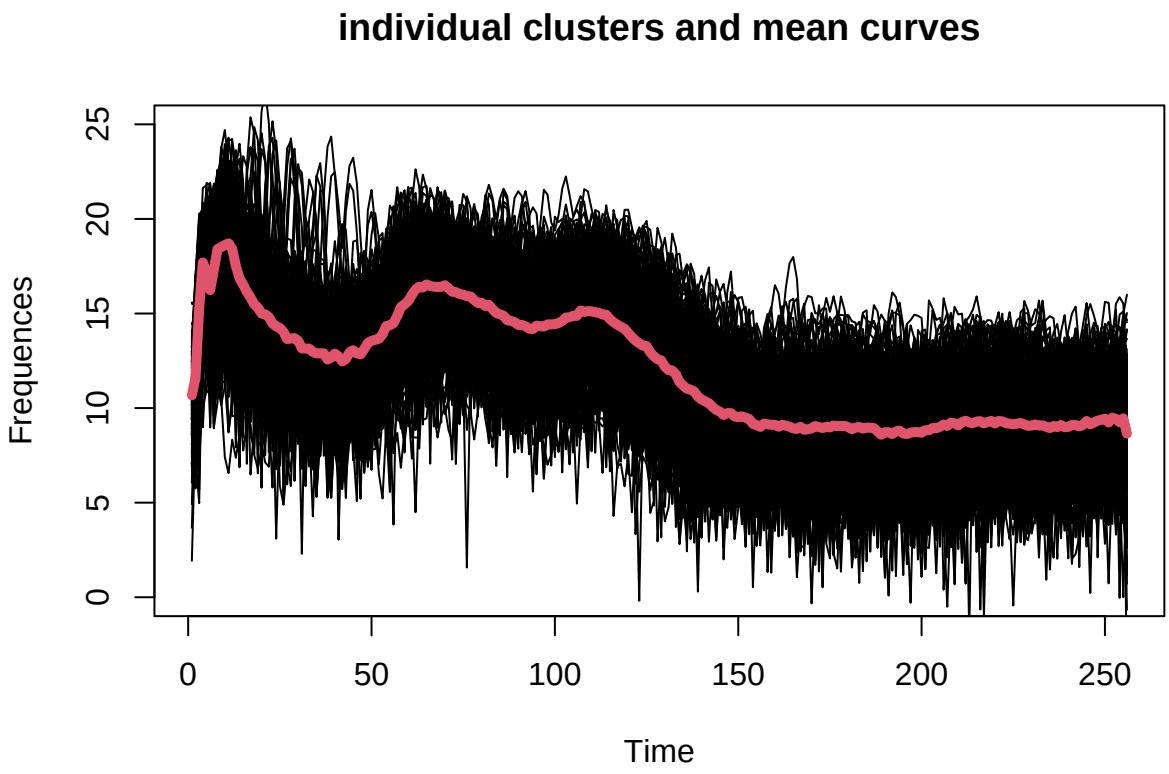

```
# Plot the cluster mean curves
clmeans <- fd_pho; clmeans$coefs <- t(femresult9$prms$my)
plot(clmeans,lwd=3) # col doesn't seem to work here, neither lwd
```

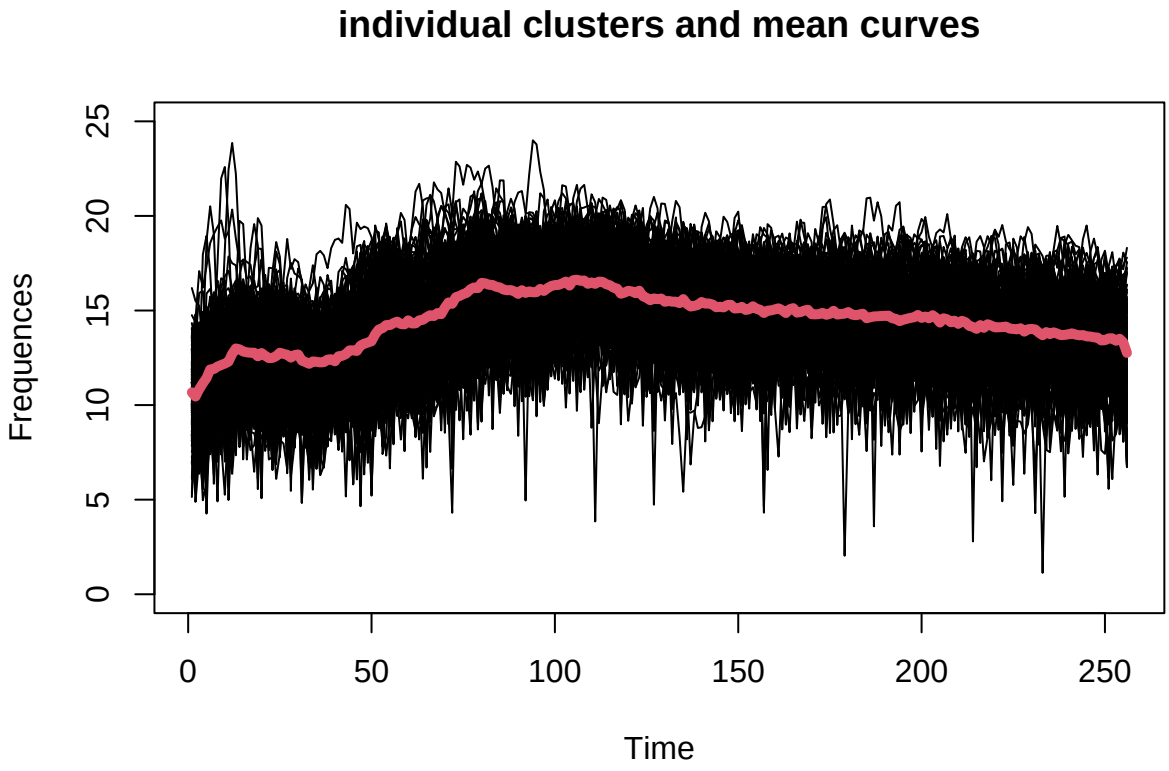
```
## [1] "done"
```

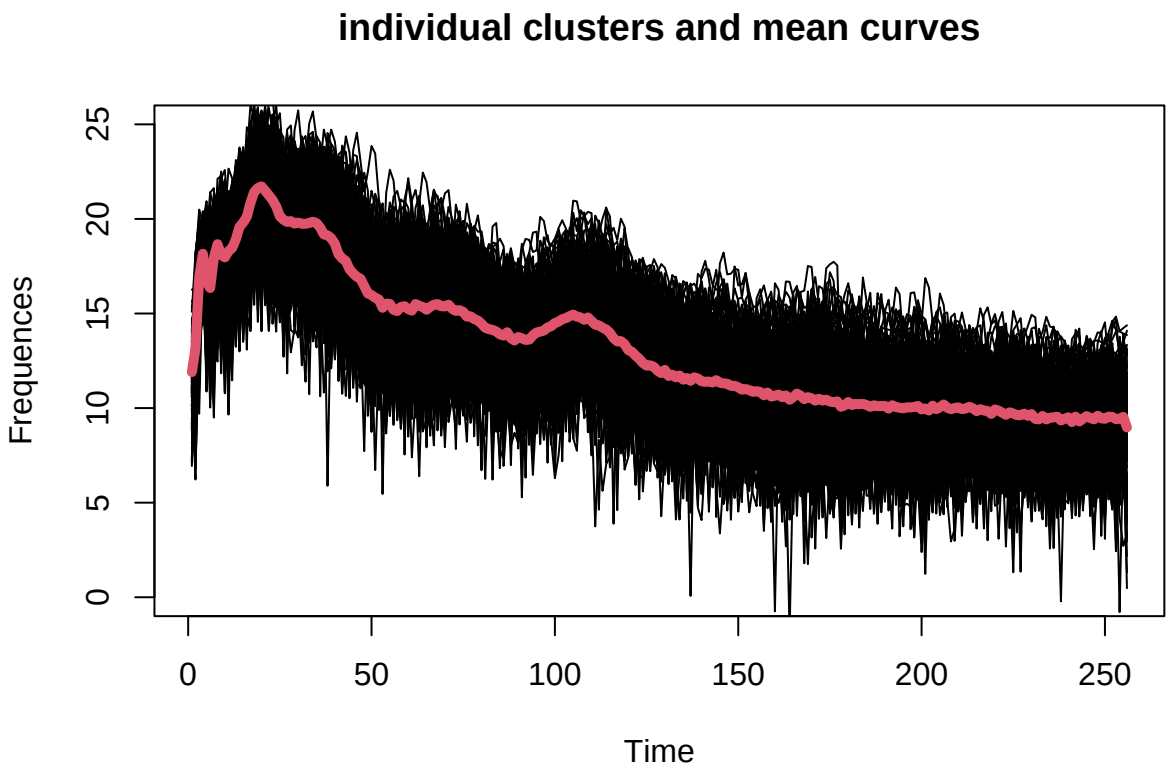
```
title(main="cluster mean curves")
```

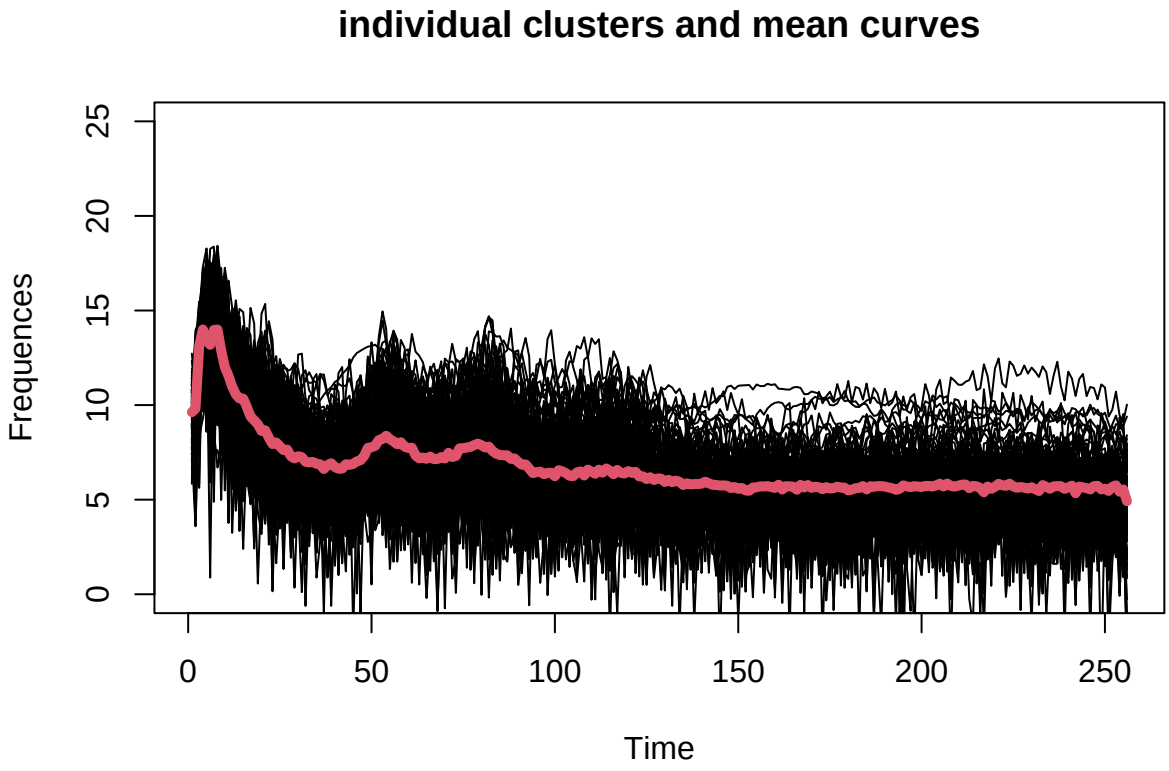


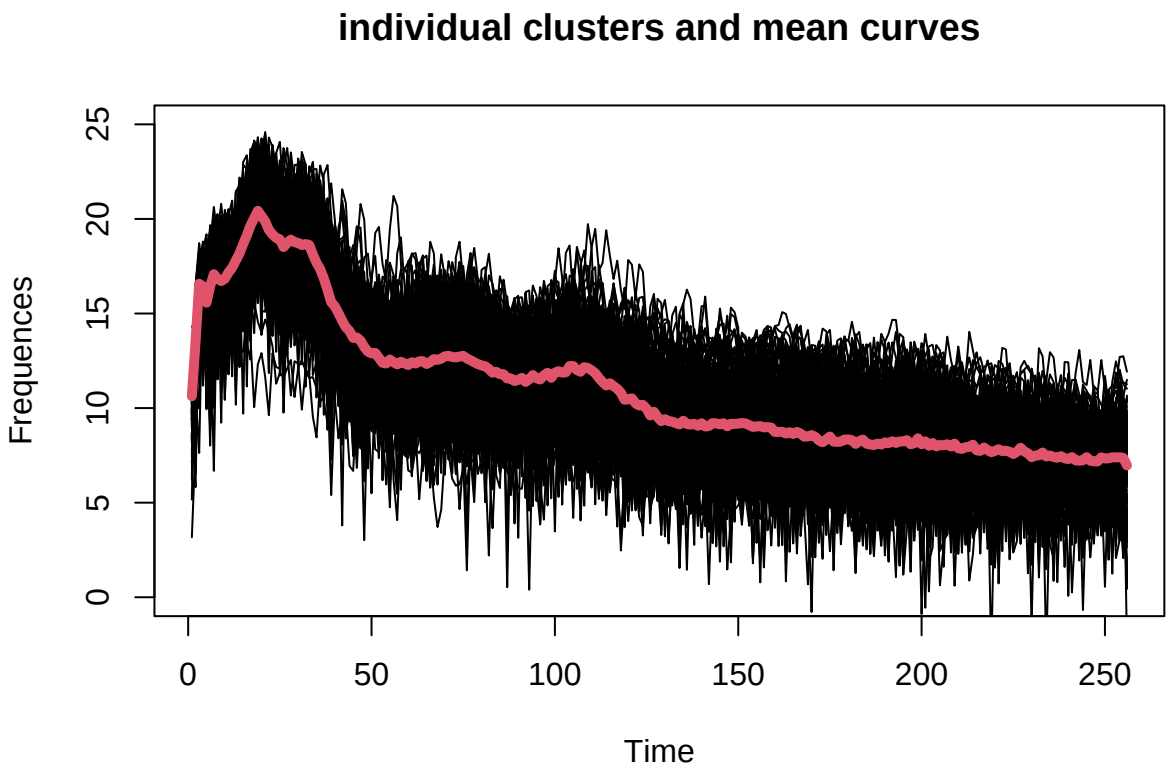

```
# Plot individual clusters and mean curves
par(ask=TRUE)
for (k in 1:femresult9$K){
plot(1:256,phonemes256[1,],type="l",ylim=c(0,25),
  ylab="Frequences",xlab="Time",col=as.integer(femresult9$cls [1]==k))

  for(i in 2:1000)
points(1:256,phonemes256[i,],type="l",col=as.integer(femresult9$cls[i]==k))
meank <- colMeans(phonemes256[femresult9$cls==k,])
points(1:256,meank,type="l",lwd=5,col=2)
title(main = "individual clusters and mean curves")
}
```


individual clusters and mean curves
Frequences
0
5
10
15
20
25
0
50
100
150
200
250
Time


individual clusters and mean curves
Frequences
0
5
10
15
20
25
0
50
100
150
200
250
Time


individual clusters and mean curves
Frequences
0
5
10
15
20
25
0
50
100
150
200
250
Time


individual clusters and mean curves
Frequences
0
5
10
15
20
25
0
50
100
150
200
250
Time




```
par(ask=FALSE)
```

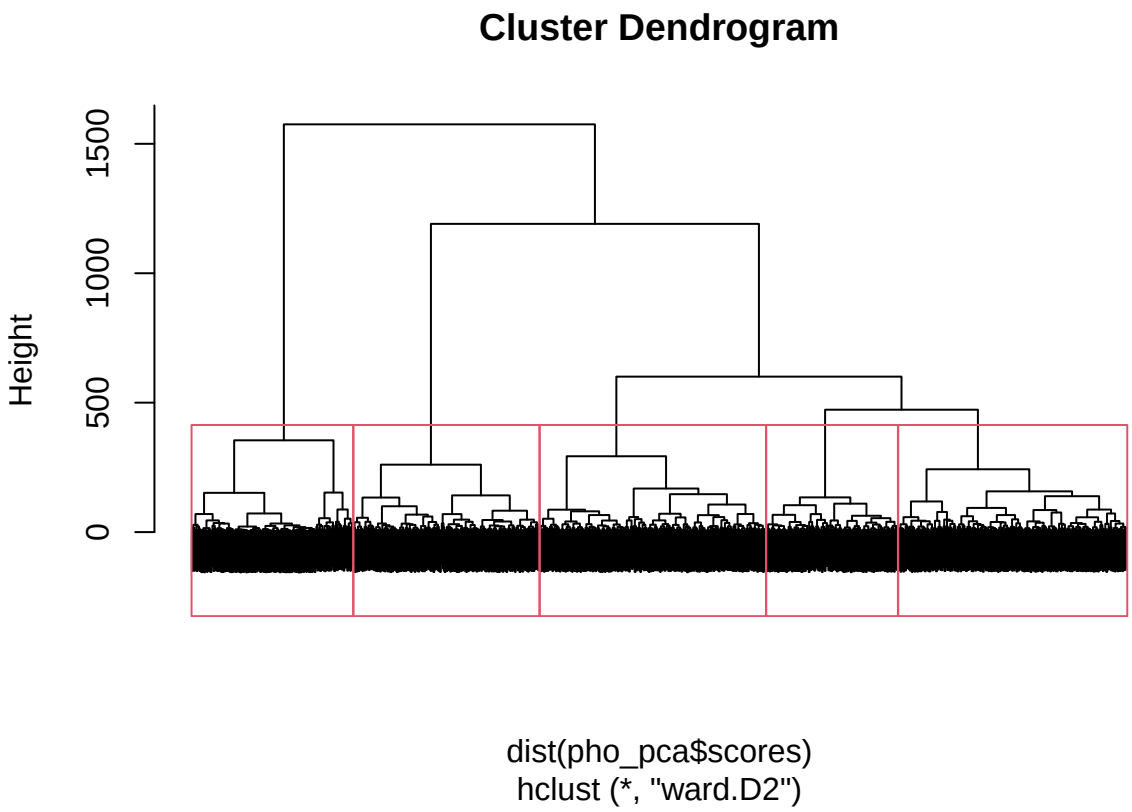
The first plot shows the original functions labeled with the assigned cluster. It is easy to notice the patterns and we can confirm what the Adj.Rand.Ind. already revealed.

The other clusters are self explainatory with their titles.

Clustering of my choice

Here we perform heirarchical clustering with Euclidean Distance and the Ward's method on the scores of the functional PCA.

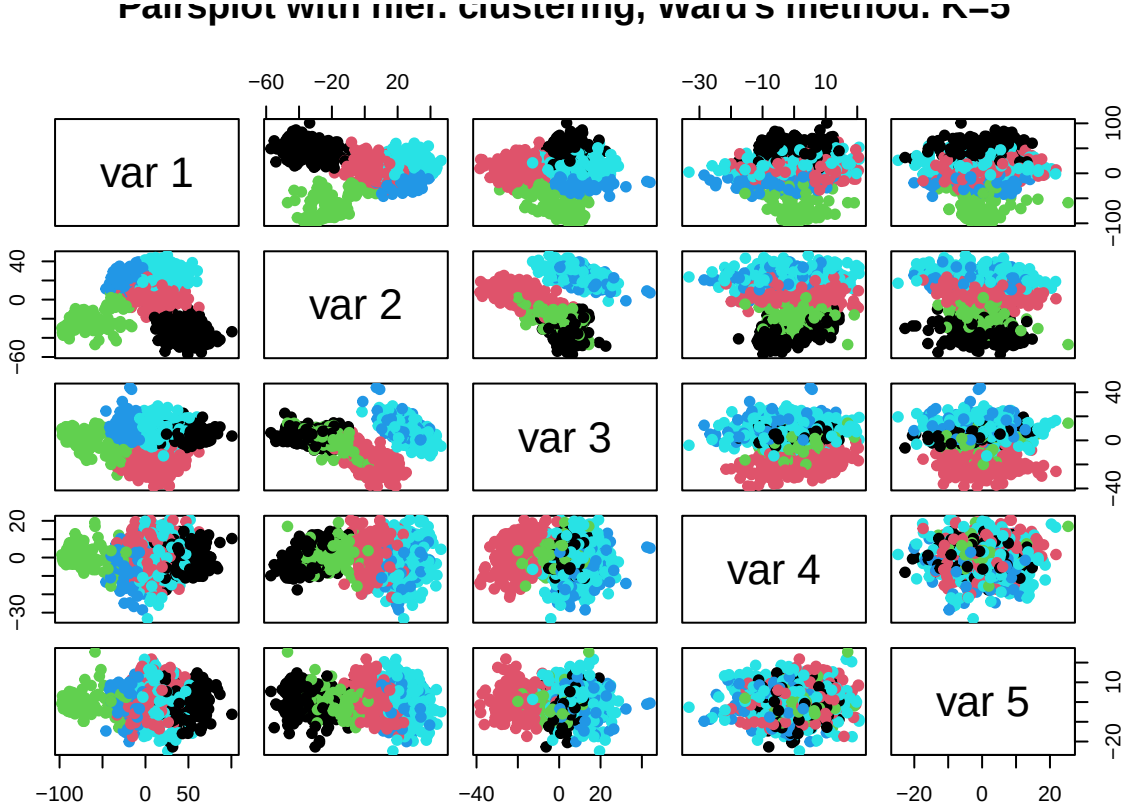
```
hier_elec <- hclust(dist(pho_pca$scores),method="ward.D2")

# Plot the dendrogram:

plot(hier_elec, labels = FALSE)

rect.hclust(hier_elec, k = 5)
```




```
hclust_clust <- cutree(hier_elec, k=5)
table(hclust_clust)
```

```
## hclust_clust
##   1   2   3   4   5
## 199 242 173 141 245
```

```
pairs(pho_pca$scores,col=hclust_clust,
  pch=19)

title(main = "Pairsplot with hier. clustering, Ward's method. K=5", outer = TRUE)
```




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

```
## [1] 0.7310852
```

This hierarchical clustering model seems to work very well. Both the dendrogram and the pairsplot have grouped the units in a homogeneous way. As happened before, only the scatterplot of the 4th PC vs the 5th PC look a bit confusing.

Regardless the visual aspect, the Adjusted Rand Index scores 73.11% which is very good.

2

Task:

Consider knots at $s_1 = 1, s_2 = 2, s_3 = 3, s_4 = 4$. Define a B-spline B of order $d = 3$ so that $B(1) = 0, B(4) = 0, B(x) = 0$ for $x \leq 1$ and $x \geq 4$, but $B(x) \geq 0$ for $x \in (1, 4)$, also $max_x B(x) = 1$. This means that you are asked to specify the second order polynomials between any two of the knots that define B.

Solution:

I decided to try to solve the problem through R rather then using the manual computations to see what I could do with it.

I was not sure on how to select the data (i.e. "x") and I have just decided to use a sequence from 1 to 4 as the knots.

Remark: In the "bs" functions there are both the parameters "df" and "degree". From the helper of R, it seems that "df" stands for "order".

```
x <- 1:4
# Define knots
knots <- c(1, 2, 3, 4)

# Define the B-spline basis functions
bspline <- bs(x, knots = knots, df = 3, intercept = FALSE)


# Normalize to ensure max_x B(x) = 1
bspline <- bspline / max(bspline)

# Set conditions
bspline[1, ] <- 0   # B(1) = 0
bspline[4, ] <- 0   # B(4) = 0

bspline
```

```
##      1    2         3         4    5 6 7
## [1,] 0 0.00 0.0000000 0.0000000 0.00 0 0
## [2,] 0 0.25 0.5833333 0.1666667 0.00 0 0
## [3,] 0 0.00 0.1666667 0.5833333 0.25 0 0
## [4,] 0 0.00 0.0000000 0.0000000 0.00 0 0
## attr(,"degree")
## [1] 3
## attr(,"knots")
## [1] 1 2 3 4
## attr(,"Boundary.knots")
## [1] 1 4
## attr(,"intercept")
## [1] FALSE
## attr(,"class")
## [1] "bs"     "basis"  "matrix"
```

The coefficients of the second order polynomials are shown in the table.

3

Task:

(a) Explain in your own words what discriminant coordinates ("dc") and asymmetric weighted discriminant coordinates ("awc") are, and how they work.

Solution:

DC Basically, Discriminant coordinates are linear combinations of the original variables derived through LDA. LDA aims to find a set of these discriminant coordinates in such a way that the ratio of the between-classes covariance matrix to the within-classes covariance matrix is maximized. In other words, it seeks to maximize the separation between different classes in the data.

The discriminant coordinates effectively transform the data from its original space into a new space where the classes are well-separated. These coordinates are chosen to maximize the differences between the means of different classes while minimizing the variation within each class.

AWC: Asymmetric Weighted Discriminant Coordinates are an extension of the traditional discriminant coordinates. In some cases, classes may not contribute equally to the analysis, and assigning equal importance to all classes may not be appropriate. Asymmetric weighting allows you to assign different weights to different classes, reflecting their relative importance in the analysis.

Asymmetric weighted discriminant coordinates as defined in Hennig (2003). Regardless of the number of desired clusters, Asymmetric discriminant projection means that there are two classes, one of which is treated as the homogeneous class (i.e., it should appear homogeneous and separated in the resulting projection). The homogeneous class is the cluster we want to define with its units that must have a certain characteristic and distance. All the remaining units are linked to the second class. They may be heterogeneous units, not showing a specific characteristic or simply being sparse. The principle is to maximize the ratio between the projection of a between classes separation matrix and the projection of the covariance matrix within the homogeneous class. Points are weighted according to their (robust) Mahalanobis distance to the homogeneous class.

Task:

(b) For the optimal 10-clusters Gaussian mixture clustering of the olive oil data (Example 6.5 on the course slides) show 2-dimensional discriminant coordinates, and asymmetric weighted discriminant coordinates for all clusters. Comment on how these plots compare to the principal components plot in terms of showing the separation of the clusters.

Solution:

Remark: we do not scale the "olive" dataset since in the Example 6.5 the unscaled data are used and we want to remain faithful to the example as required by the task with $K = 1, ..., 10$.

```
oliveoil <- read.table("oliveoil.dat",header=TRUE, stringsAsFactors = TRUE)
olive <- oliveoil[,3:10]


molive <- Mclust(olive,G=1:10)
summary(molive)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVE (ellipsoidal, equal orientation) model with 10 components:
##
##  log-likelihood   n  df       BIC       ICL
##       -20448.03 572 197 -42146.84 -42193.07
##
## Clustering table:
##   1   2   3   4   5   6   7   8   9  10
##  30  96 110  37  50  64  34  49  45  57
```

```r
summary(molive$BIC)
```

```
## Best BIC values:
##             VVE,10        VVV,6       VVE,9
## BIC      -42146.84 -42158.49981 -42195.7529
## BIC diff      0.00    -11.65605    -48.9091
```

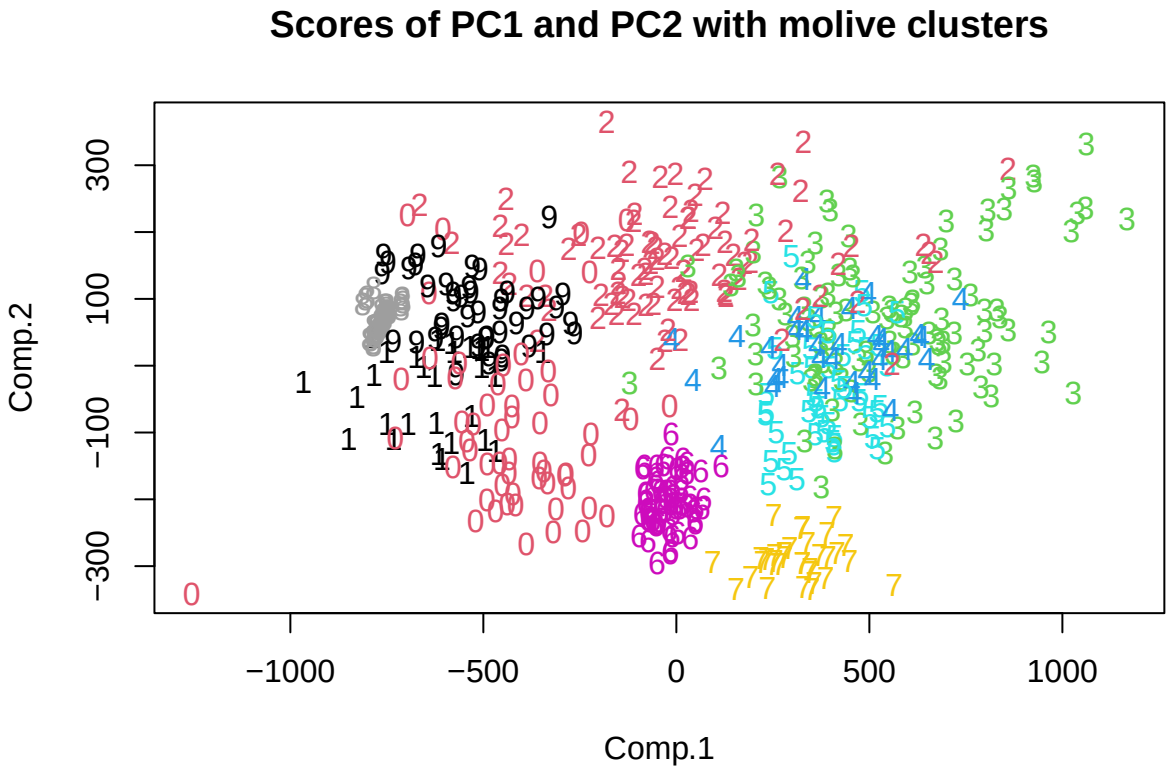
```r
adjustedRandIndex(molive$classification,oliveoil$region)
```

```
## [1] 0.5914548
```

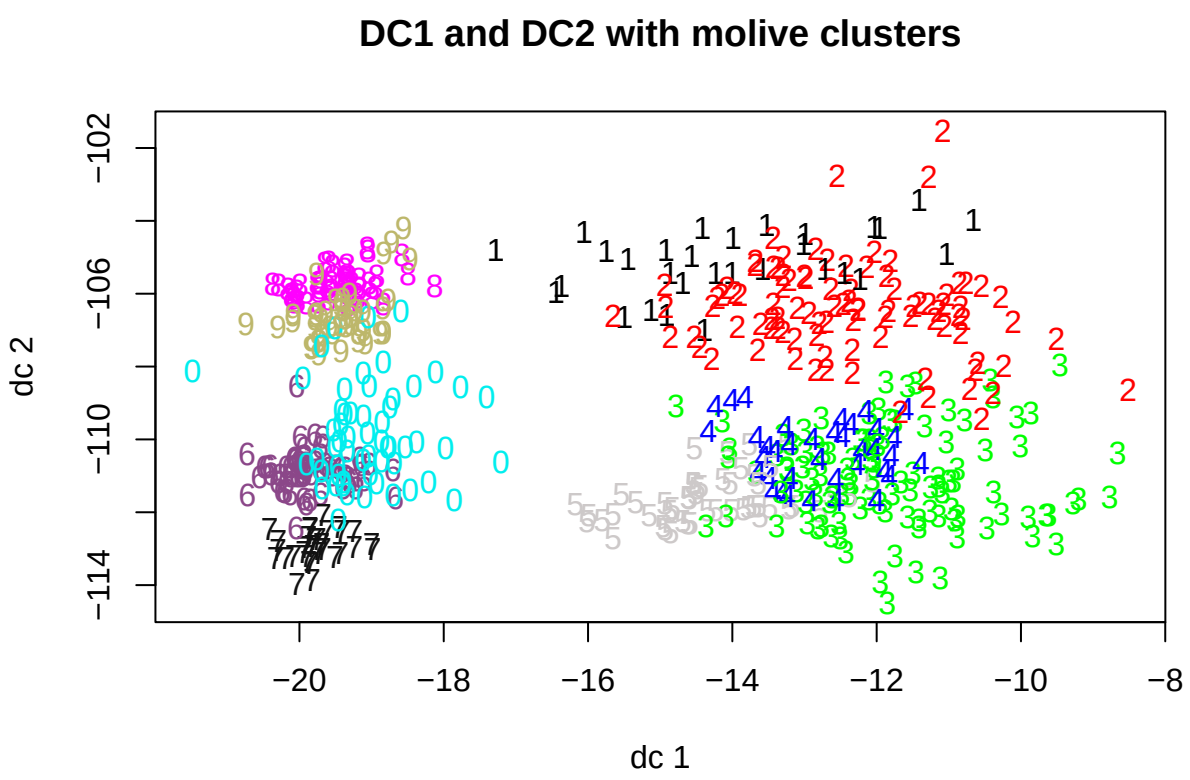
```r
prolive <- princomp(olive)
summary(prolive)
```

```
## Importance of components:
##                             Comp.1       Comp.2       Comp.3       Comp.4
## Standard deviation     479.7299024 150.82827868 45.394449751 27.522646558
## Proportion of Variance   0.8970072   0.08866821  0.008031707  0.002952451
## Cumulative Proportion    0.8970072   0.98567544  0.993707152  0.996659603
##                             Comp.5       Comp.6       Comp.7       Comp.8
## Standard deviation     24.78169442 1.196956e+01 7.1390744088 6.9756965249
## Proportion of Variance  0.00239367 5.584168e-04 0.0001986489 0.0001896608
## Cumulative Proportion   0.99905327 9.996117e-01 0.9998103392 1.0000000000
```

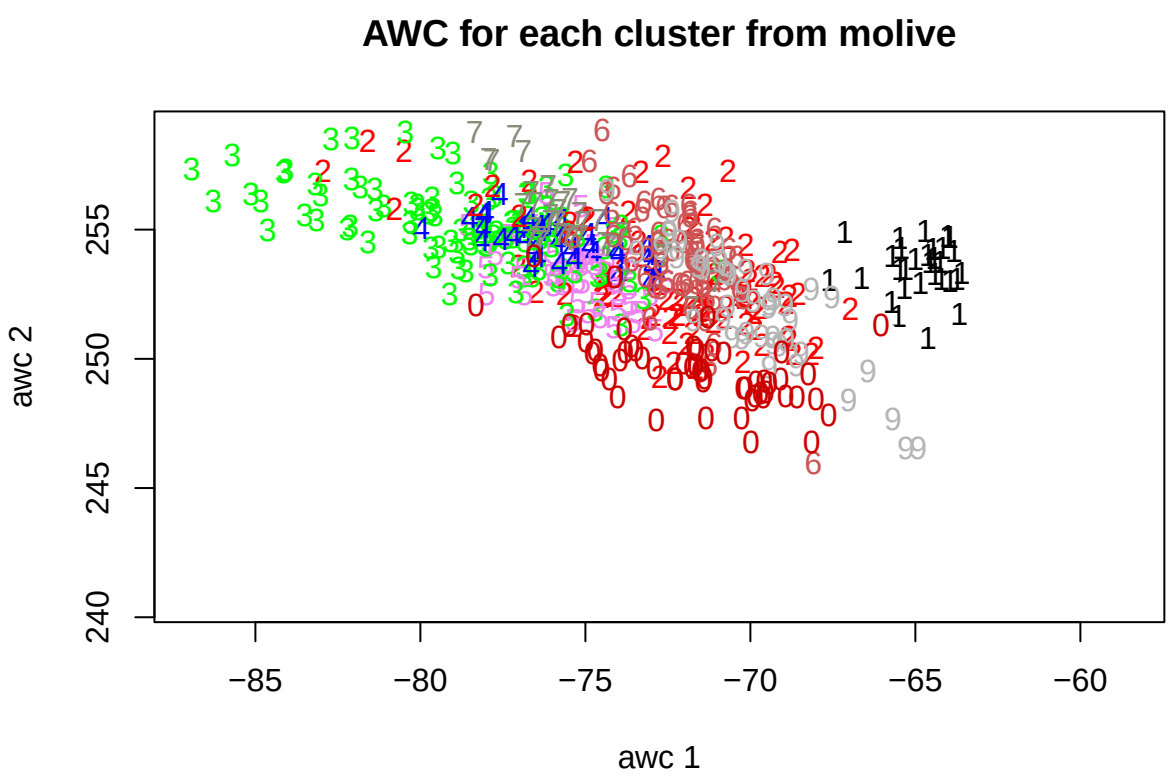
```r
plot(prolive$scores,col=molive$classification,
pch=clusym[molive$classification])
title(main= "Scores of PC1 and PC2 with molive clusters")
```

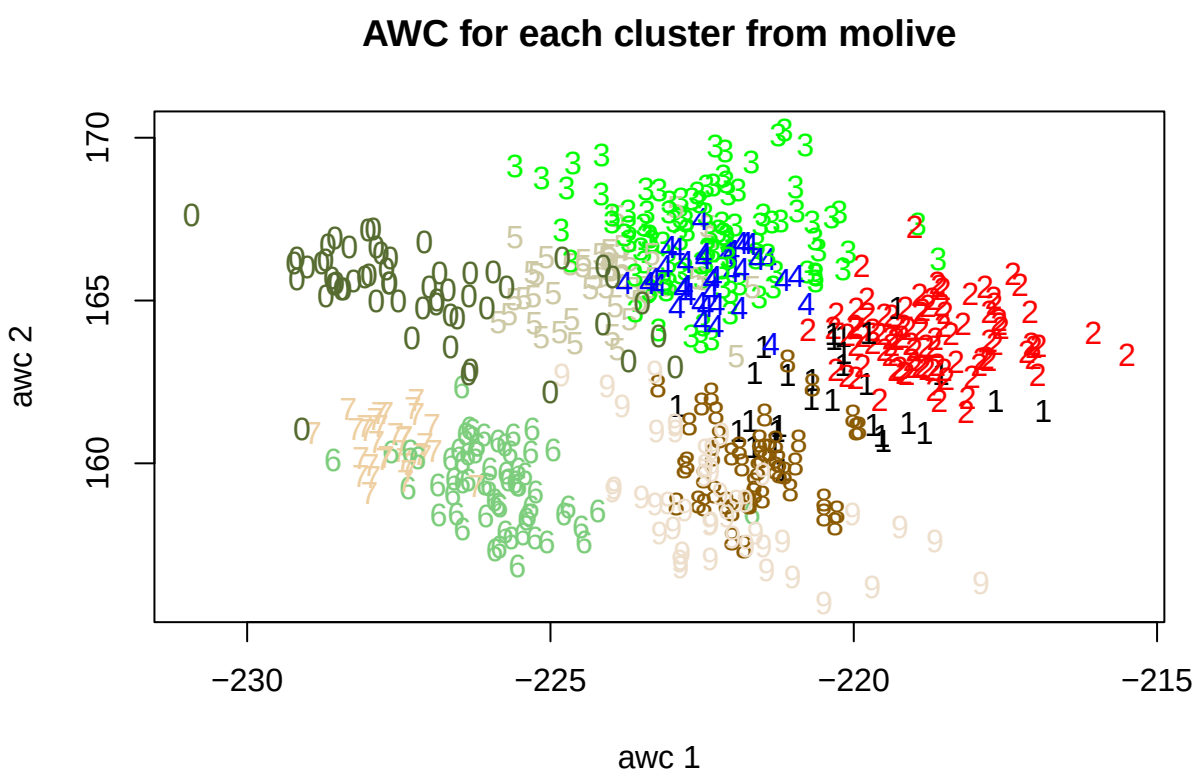


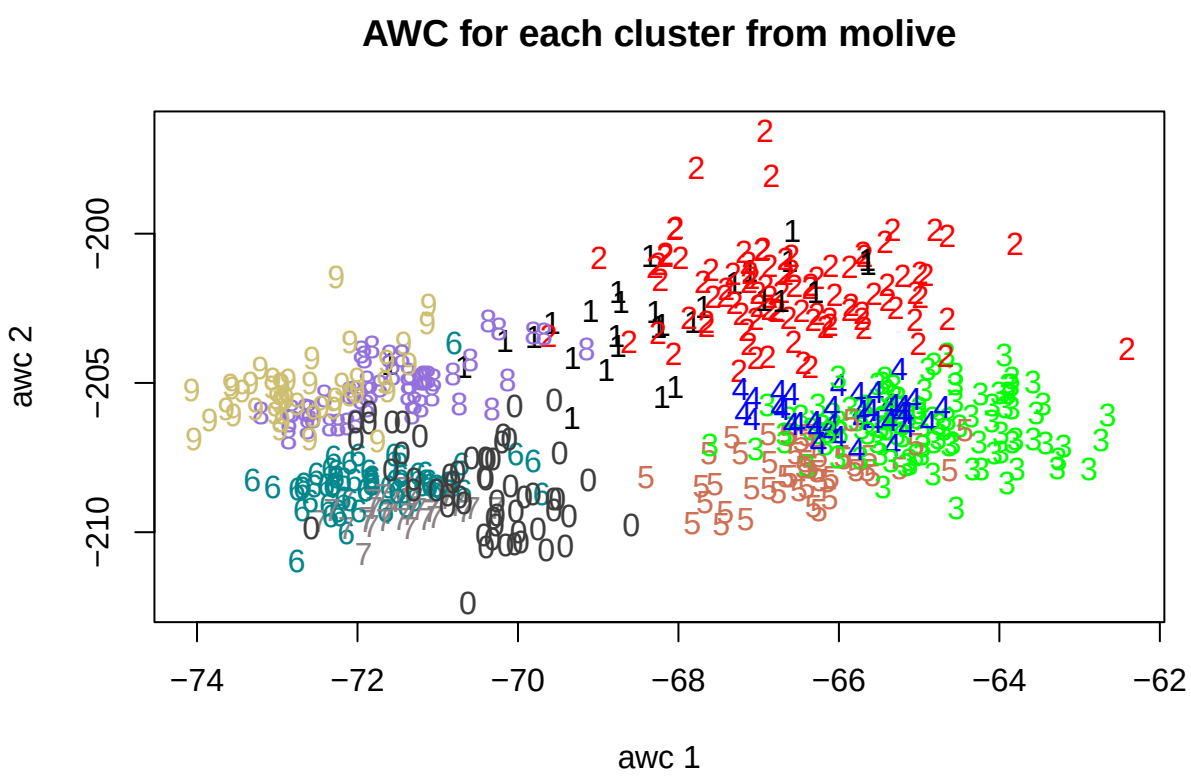

```r
plotcluster(olive, molive$classification, method = "dc")
title(main= "DC1 and DC2 with molive clusters")
```

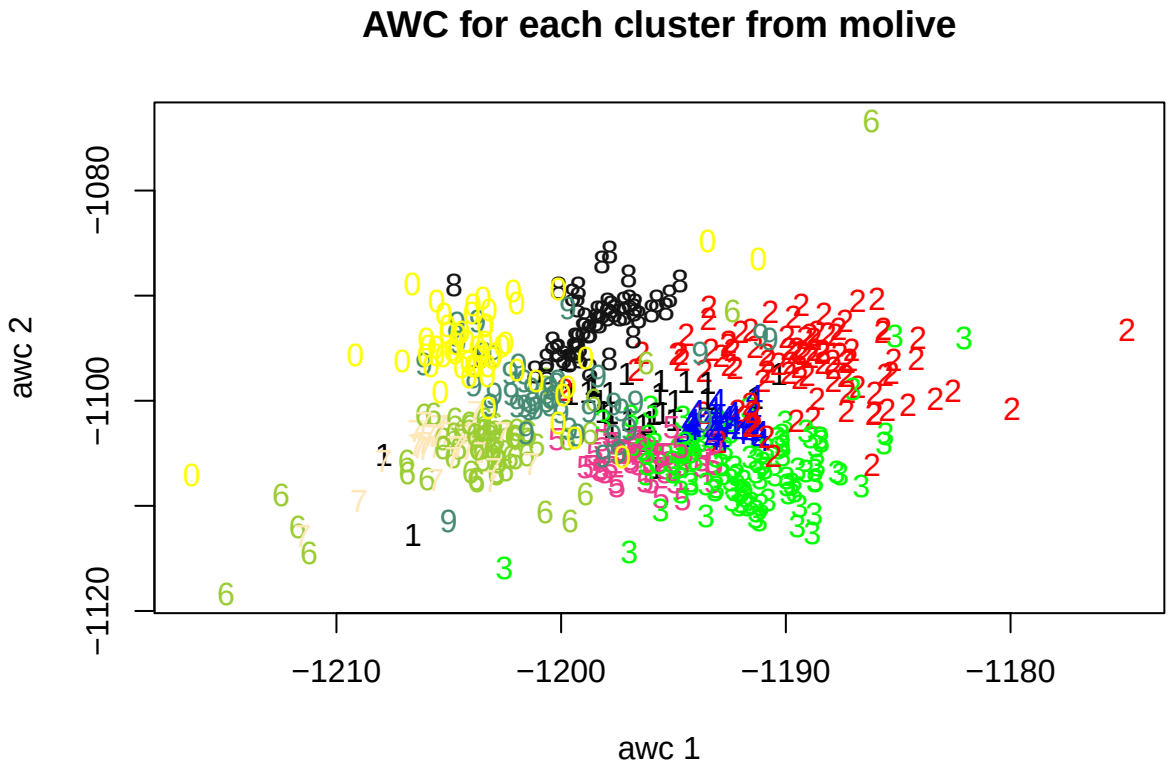


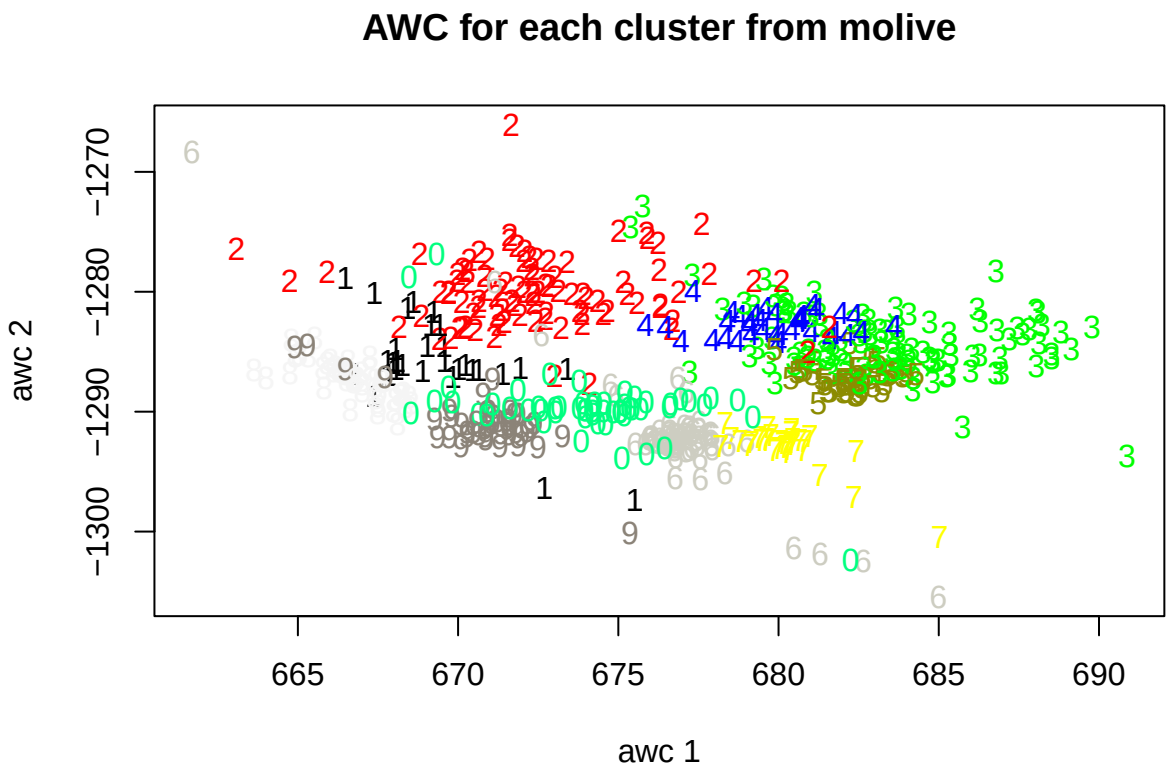

```
for (i in 1:9){
  plotcluster(olive, molive$classification, clnum = i,method = "awc")
  title(main="AWC for each cluster from molive")
}
```

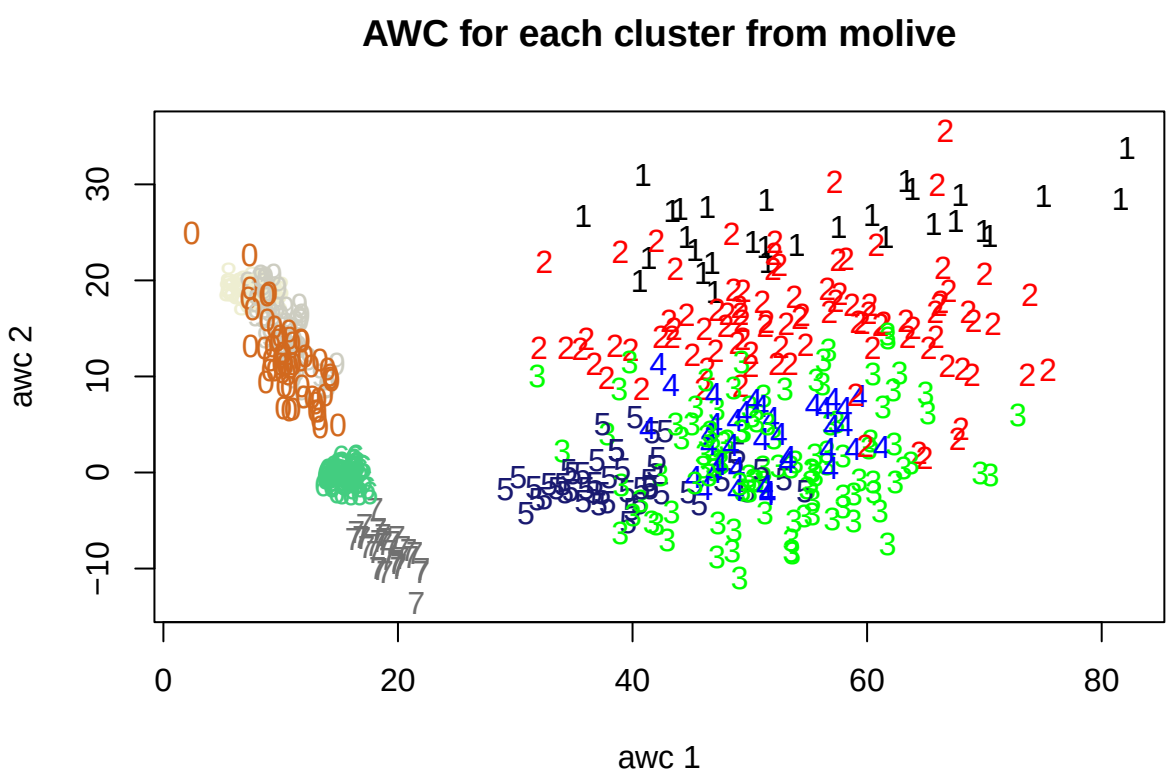

AWC for each cluster from molive
awc 2
240
245
250
255
−85
−80
−75
−70
−65
−60
awc 1

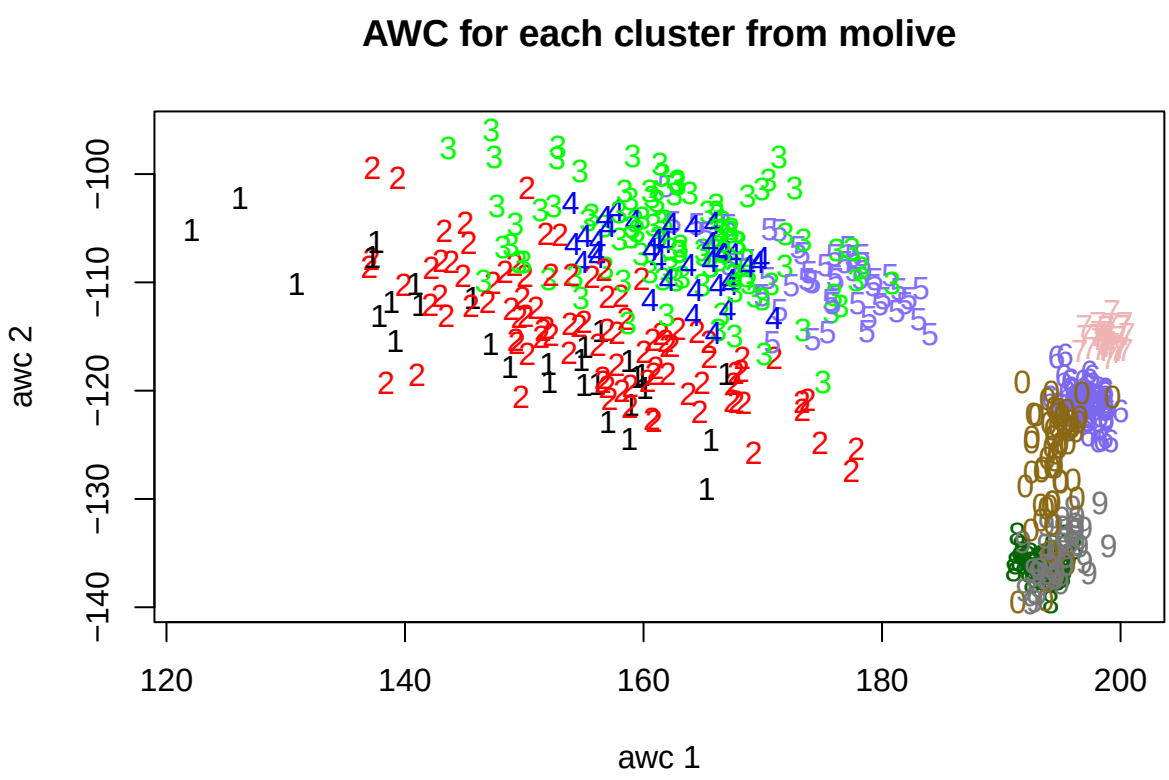

AWC for each cluster from molive
awc 2
170
165
160
−230
−225
−220
−215
awc 1

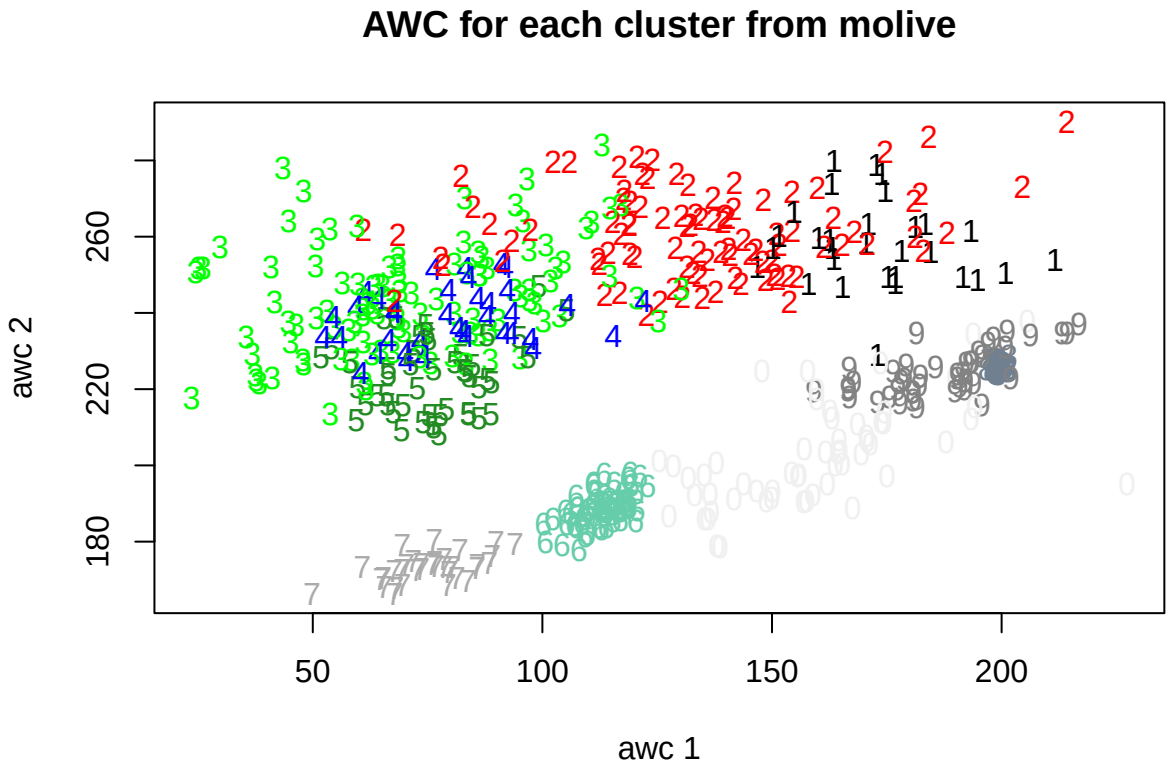

AWC for each cluster from molive
awc 2
−200
−205
−210
−74
−72
−70
−68
−66
−64
−62
awc 1

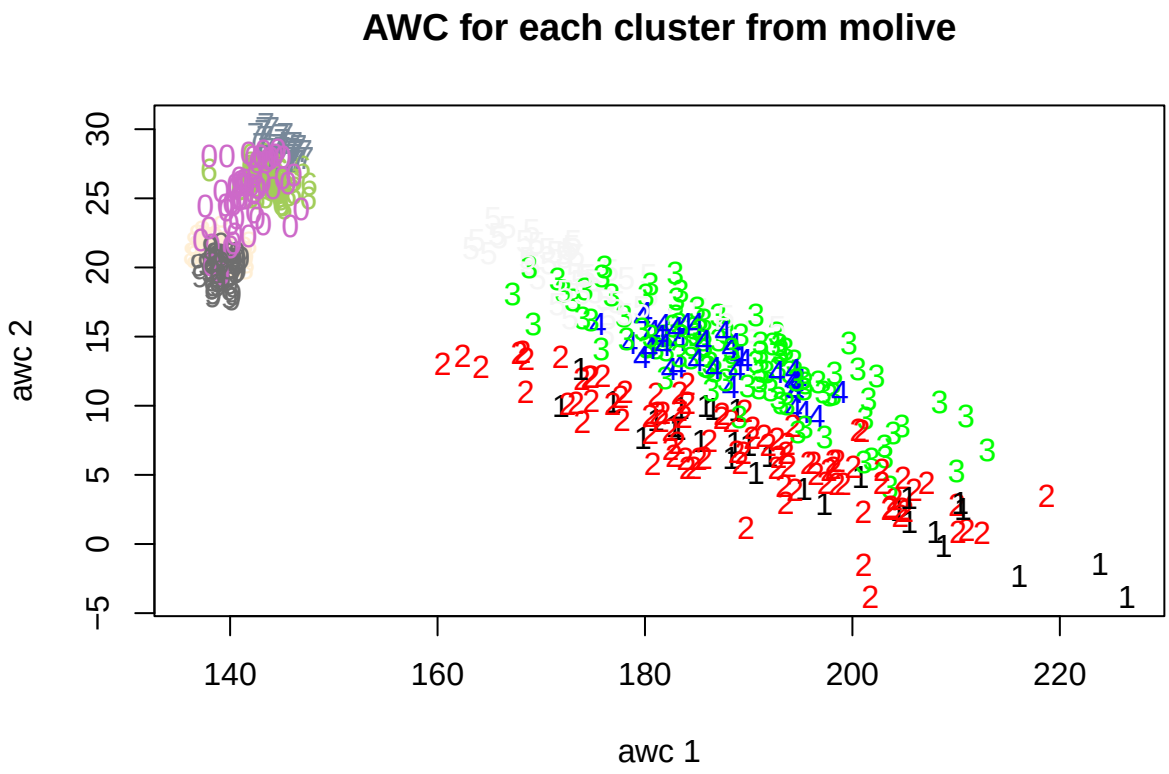

AWC for each cluster from molive
awc 2
−1080
−1100
−1120
−1210
−1200
−1190
−1180
awc 1


AWC for each cluster from molive
awc 2
−1270
−1280
−1290
−1300
665
670
675
680
685
690
awc 1


AWC for each cluster from molive
awc 2
30
20
10
0
−10
0
20
40
60
80
awc 1


AWC for each cluster from molive
awc 2
−100
−110
−120
−130
−140
120
140
160
180
200
awc 1


AWC for each cluster from molive
awc 2
180
220
260
50
100
150
200
awc 1



The best model selected by MClust is VVE with 10 clusters. The adjusted Rand Index is 59.14%, meaning that the clustering is fairly good.

The plot of PC1 and PC2 shows that clusters 6,7 and 8 are very dense, circular and homogeneous. Clusters 0 and 9 are fairly good while there seems to be heterogenity and overlapping in the other clusters.

The plot with the DC's seems to create a better separation between the clusters than PC's. Indeed, clusters 0,6,7,8 and 9 are very dense and homogeneous while there is less overlapping in clusters 3 and 4. Clusters 1 and 2 and 5 are still decent.

The remaining 9 plots are meant to isolate the ith cluster chosen in "clunm" from the rest, creating a homogeneous class against the rest which is heterogeneous.

- The first AWC performs greatly because the cluster 1 is easily distinguishable from the rest,
- the second is quite good,
- the third is not good since there is overlapping with cluster 4,
- the forth, fifth and ninth are quite dense and circular-shaped but the other clusters are not distant,
- the sixth, seventh and eith are very good.

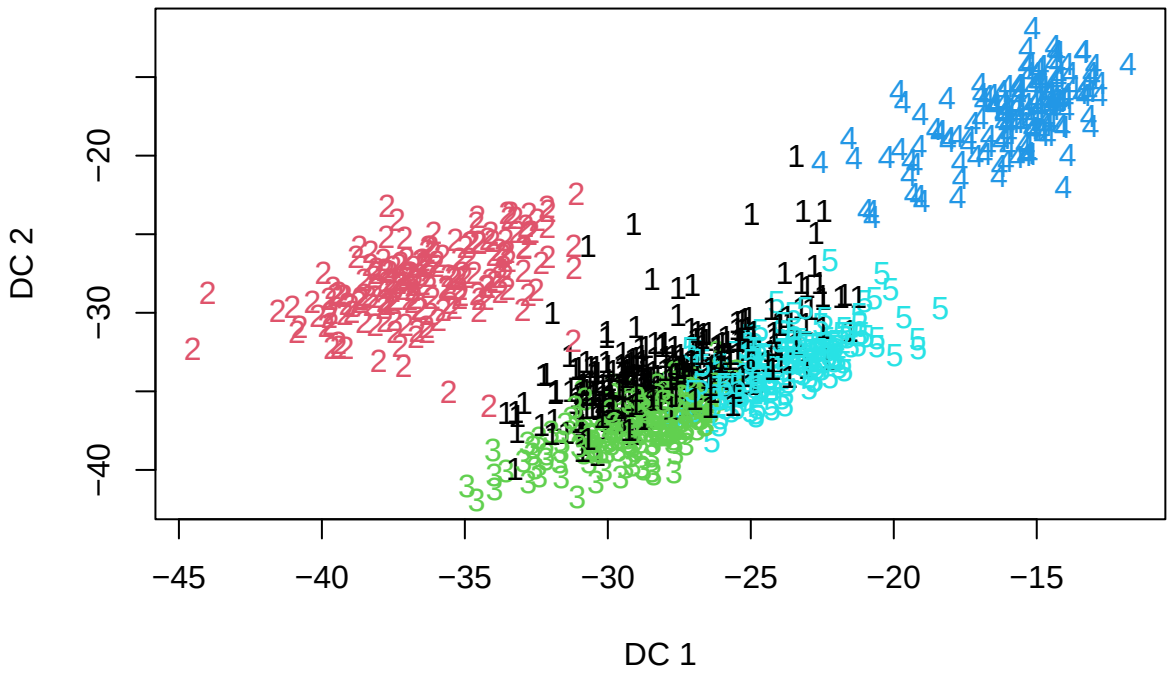
Task:

(c) Considering the phoneme data from question 1 and the funFEM-clustering, compare the plot of the first two dimensions of the Fisher discriminating subspace from funFEM with what you get when applying discriminant cordinates using plotcluster to the data set of the coefficients of the full dimensional B-spline basis and the funFEM-clustering.

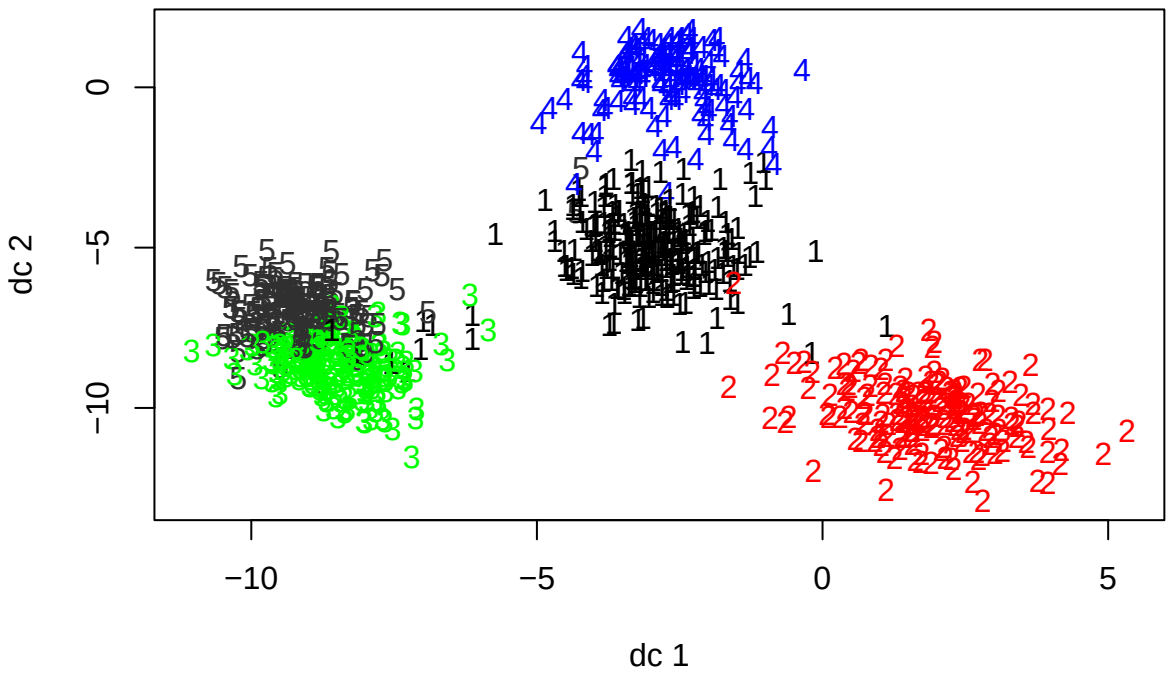
Solution:

As written below, we plot the DC's from FunFEM using the projections on U-Space and then with "plotcluster".

```
# Visualisation of discriminative subspace U,
# projection of observations on U-space:

fdproj <- t(fd_pho$coefs) %*% femresult9$U

plot(fdproj,col=femresult9$cls,pch=clusym[femresult9$cls],
    xlab="DC 1",ylab="DC 2")
```




```
plotcluster(fdproj, femresult9$cls, method = "dc")
```



Both the scatterplots show well defined and homogeneous clusters. However, the "plotcluster" with discriminant coordinates is able to visually create a better separation from one cluster to another. In facts, in the first plot, three clusters are attached while in the second only two of them.