Homework 8

Federico Veronesi

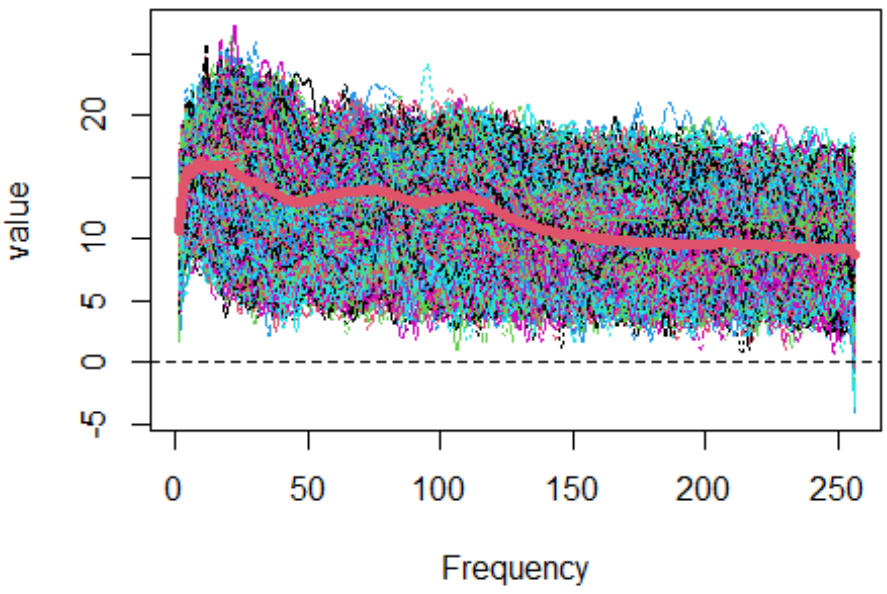
2023-12-02

```
## Data prep
setwd("C:/Users/Veronesi/Desktop/uniBo/Magistrale/Modern Statistics and Big Data Analytics")
phonemes <- read.table("phonemes1000.dat", sep = " ", header = T)
phonemes256 <- as.matrix(phonemes[,1:256])
```

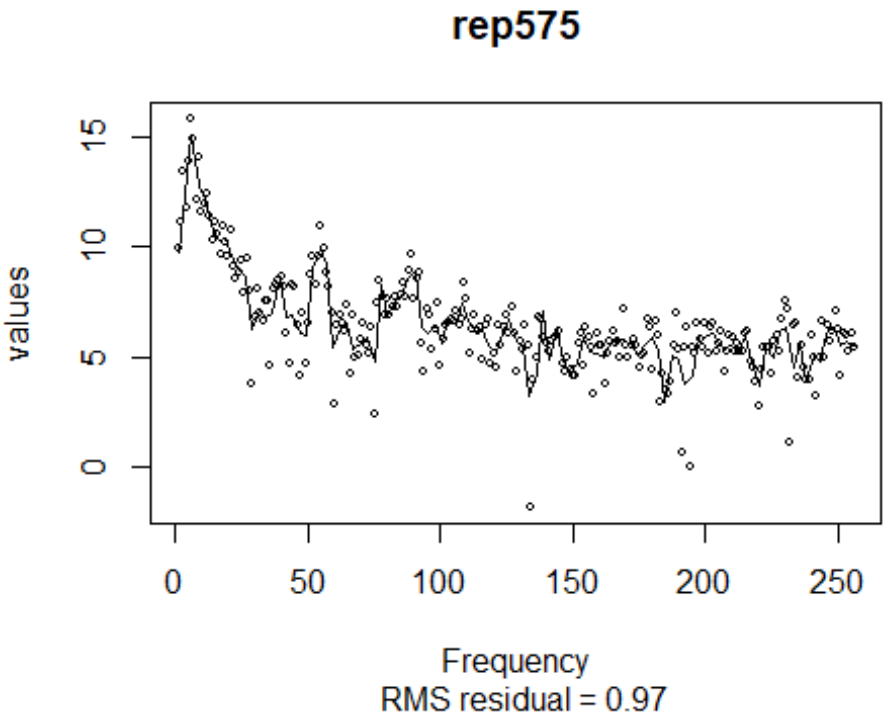
Exercise 1

B-spline

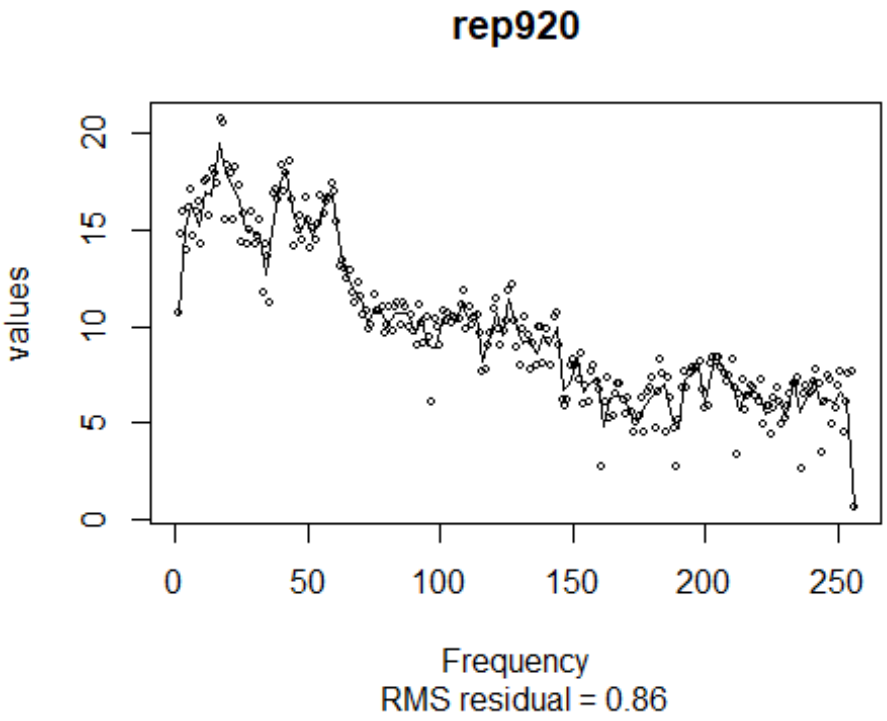
```
library(fda)

bbasis <- create.bspline.basis(c(1,256),nbasis=100) # same with p=100
fdphonemes <- Data2fd(1:256,y=t(as.matrix(phonemes256)),basisobj=bbasis)
plot(fdphonemes, xlab = "Frequency")

mphon <- mean.fd(fdphonemes)
lines(mphon,col=2,lwd=5)
```




```
plotfit.fd(t(phonemes256),1:256,fdphonemes,index=sample(1:1000,
1),cex.pch=0.5,residual= F, xlab = "Frequency")
```




```
plotfit.fd(t(phonemes256),1:256,fdphonemes,index=sample(c(1:341, 343:1000),
1),cex.pch=0.5,residual= F, xlab = "Frequency")
```

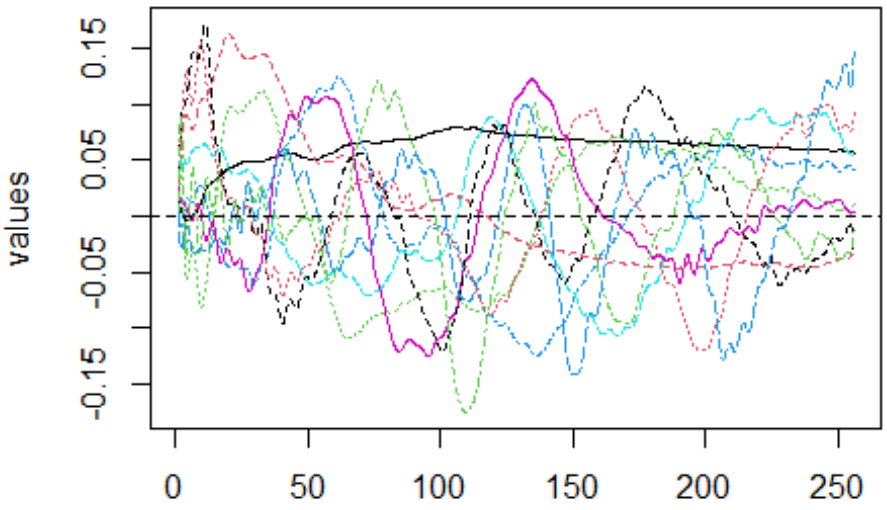


The fit seems quite good. That's because we choose an high number of basis (100), so the model fits well these data (potential overfitting?).

Functional PCA

Here, initially, I choose an high number of principal component. Then I will decide how many of them it's worth keeping, looking at the cumulative proportion of explained variance.

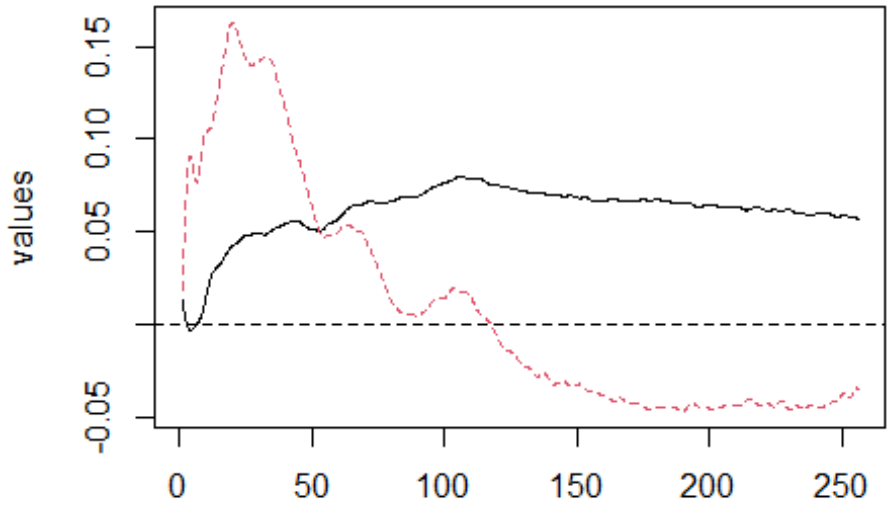
```
phonpca <- pca.fd(fdphonemes, nharm = 10)
plot(phonpca$harmonics) # PCs phi_k
```




```
## [1] "done"

cumsum(phonpca$varprop) # Cumulative percentage of variance

##  [1] 0.5763016 0.7799804 0.8390231 0.8603975 0.8747653 0.8868737 0.8963245
##  [8] 0.9047813 0.9114037 0.9165083
```

It's sufficient to keep the first two principal components (almost 80% of variance it's explained).

```
phonpca <- pca.fd(fdphonemes, nharm = 2)
plot(phonpca$harmonics) # PCs phi_k
```




```
## [1] "done"

library(funFEM)

## Warning: il pacchetto 'funFEM' è stato creato con R versione 4.2.3

## Caricamento del pacchetto richiesto: elasticnet

## Caricamento del pacchetto richiesto: lars

## Loaded lars 1.3

set.seed(11111)
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=2:10 # 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(fdphonemes,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] "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))
```

```
femmodels[besti]
```

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

```
bestk[besti]
```

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

AjBk with 9 components is chosen:

```
best_fem_model <- femresults[besti]
trueclass <- phonemes[,257]
table(best_fem_model[[1]]$cls, trueclass)
```

```
##    trueclass
##      aa  ao dcl  iy  sh
##   1   0   0   0   1  86
##   2   0   0   0   1 109
##   3  22 104   0   0   0
##   4   0   0  11  89   0
##   5   0   0 148   0   0
##   6   0   1   1 147   0
##   7  52  17   0   0   0
##   8  82 102   0   0   0
##   9   4   2   1  20   0
```

```
library(mclust)
```

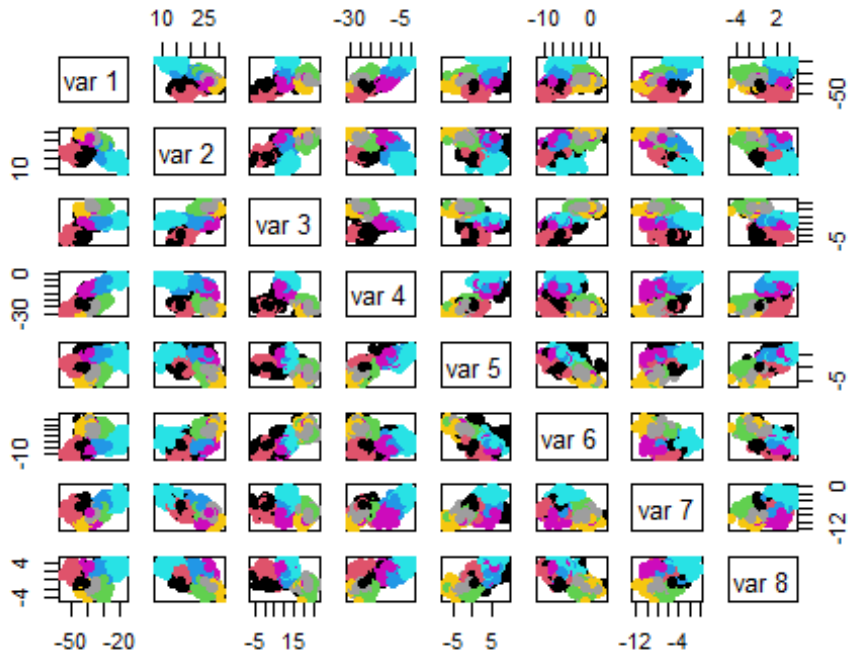
```
## Warning: il pacchetto 'mclust' è stato creato con R versione 4.2.3
```

```
## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.

adjustedRandIndex(best_fem_model[[1]]$cls, trueclass)

## [1] 0.5350016
```

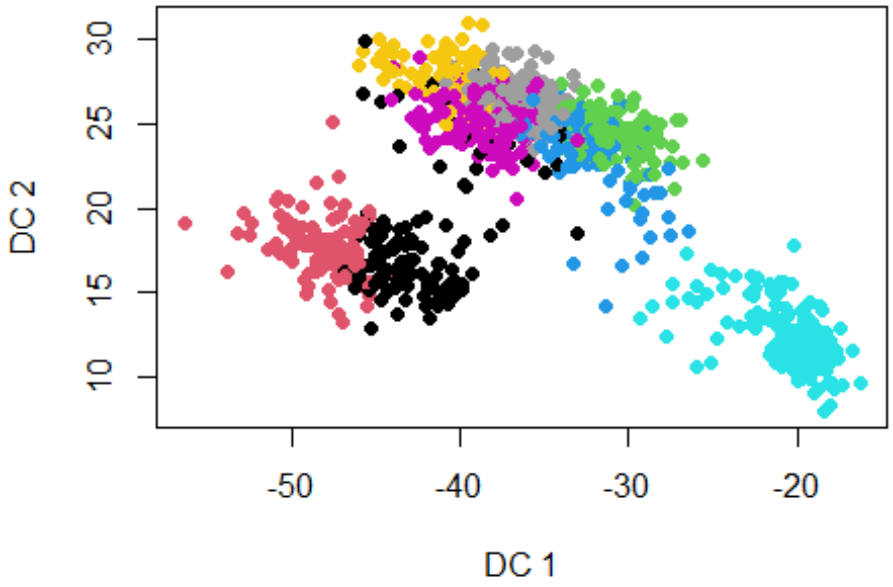
ARI is quite good: despite k is different from the true n°of clusters, the method is able to find a partition the reproduces quite well the true clustering.

Plot in two dimensions

```
# Visualisation of discriminative subspace U,
# projection of observations on U-space:
fdproj <- t(fdphonemes$coefs) %*% best_fem_model[[1]]$U
pairs(fdproj,col=best_fem_model[[1]]$cls,pch=19)
```

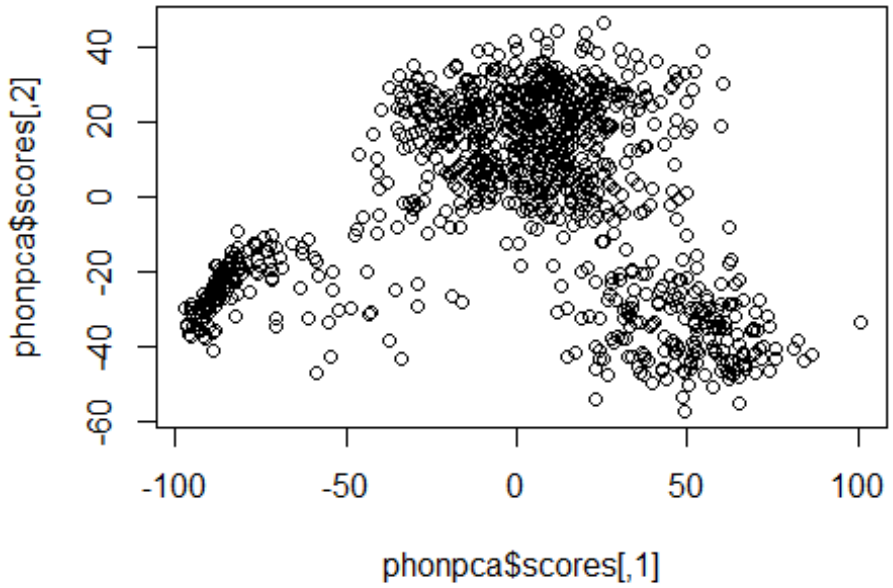



```
plot(fdproj,col=best_fem_model[[1]]$cls,pch=19,xlab="DC 1",ylab="DC 2")
```



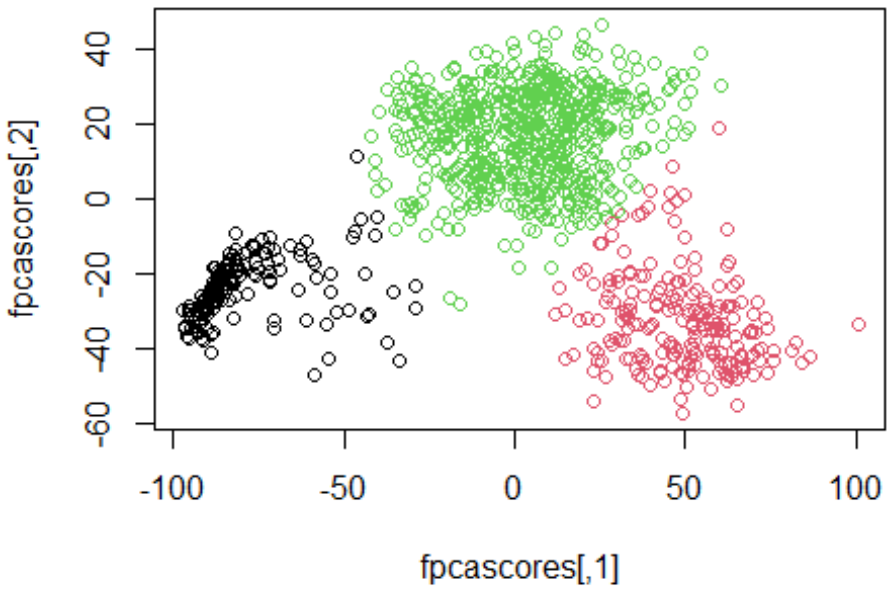
Other methods applied on functional PCA scores

```
plot(phonpca$scores)
```



K-means

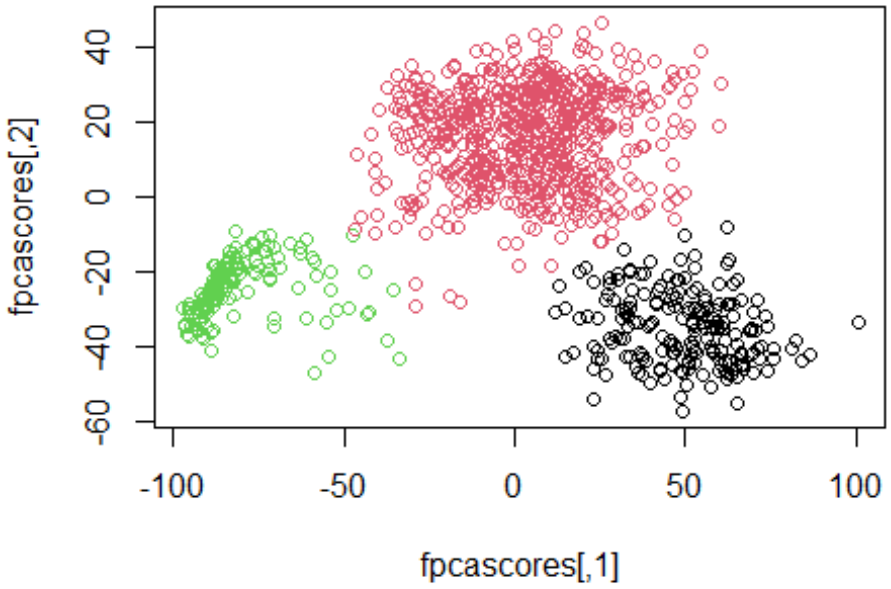
```
set.seed(125)
fpcascores <- phonpca$scores
km3 <- kmeans (fpcascores, centers = 3, nstart = 100)
plot(fpcascores, col = km3$cluster)
```



The clusters are well separated and spherical (except for the "black" one), so the k-means with K=3 seems reasonable. Let's try a Gaussian mixture, or a t mixture (since some outliers are present) with 3 components.

Gaussian mixture

```
library(mclust)
gaussmixture <- Mclust(fpcascores, G=3)
plot(fpcascores, col = gaussmixture$classification)
```

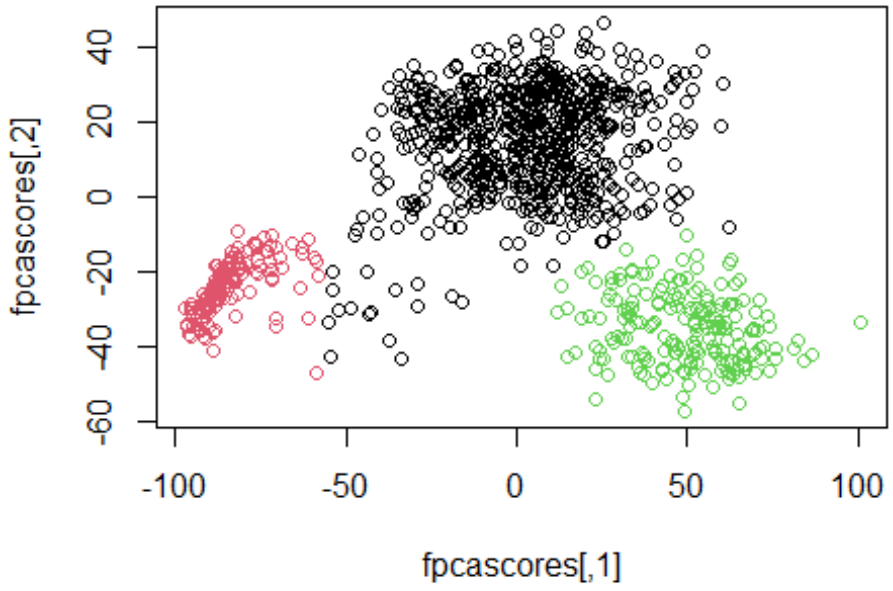


The Gaussian mixture creates cluster even more homogeneous than k-means. However, we will decide what's better by looking at ARI and comparing them to the true clustering.

t-mixture

```
library(teigen)
set.seed(333)
tmixture <- teigen(fpcascores, Gs=3)

## Time taken:???  |  Approx. remaining:???  |  0% completeTime taken:   0.6 secs
|  Approx. remaining:  16.3 secs  |     4% completeTime taken:   1.6 secs  |
Approx. remaining:  20.2 secs  |     7% completeTime taken:   1.6 secs  |  Approx.
remaining:  13.4 secs  |    11% completeTime taken:   1.7 secs  |  Approx.
remaining:   9.9 secs  |    14% completeTime taken:   1.7 secs  |  Approx.
remaining:   7.8 secs  |    18% completeTime taken:   1.7 secs  |  Approx.
remaining:   6.4 secs  |    21% completeTime taken:   1.8 secs  |  Approx.
remaining:   5.3 secs  |    25% completeTime taken:   1.8 secs  |  Approx.
remaining:   4.5 secs  |    29% completeTime taken:   1.8 secs  |  Approx.
remaining:   3.9 secs  |    32% completeTime taken:   1.9 secs  |  Approx.
remaining:   3.4 secs  |    36% completeTime taken:   1.9 secs  |  Approx.
remaining:   2.9 secs  |    39% completeTime taken:   1.9 secs  |  Approx.
remaining:   2.6 secs  |    43% completeTime taken:   3.9 secs  |  Approx.
remaining:   4.5 secs  |    46% completeTime taken:     4 secs  |  Approx.
remaining:     4 secs  |    50% completeTime taken:     4 secs  |  Approx.
remaining:   3.5 secs  |    54% completeTime taken:     4 secs  |  Approx.
remaining:     3 secs  |    57% completeTime taken:     6 secs  |  Approx.
remaining:   3.9 secs  |    61% completeTime taken:     6 secs  |  Approx.
remaining:   3.3 secs  |    64% completeTime taken:   9.7 secs  |  Approx.
remaining:   4.6 secs  |    68% completeTime taken:   9.8 secs  |  Approx.
```

```
remaining:   3.9 secs  |    71% completeTime taken:  13.8 secs  |  Approx.
remaining:   4.6 secs  |    75% completeTime taken:  13.9 secs  |  Approx.
remaining:   3.8 secs  |    79% completeTime taken:  16.1 secs  |  Approx.
remaining:   3.5 secs  |    82% completeTime taken:  16.1 secs  |  Approx.
remaining:   2.7 secs  |    86% completeTime taken:  18.5 secs  |  Approx.
remaining:   2.2 secs  |    89% completeTime taken:  18.5 secs  |  Approx.
remaining:   1.4 secs  |    93% completeTime taken:  18.6 secs  |  Approx.
remaining:   0.7 secs  |    96% completeTime taken:  18.6 secs  |  Approx.
remaining:     0 secs  |   100% complete

plot(fpcascores, col = tmixture$classification)
```



Evaluation

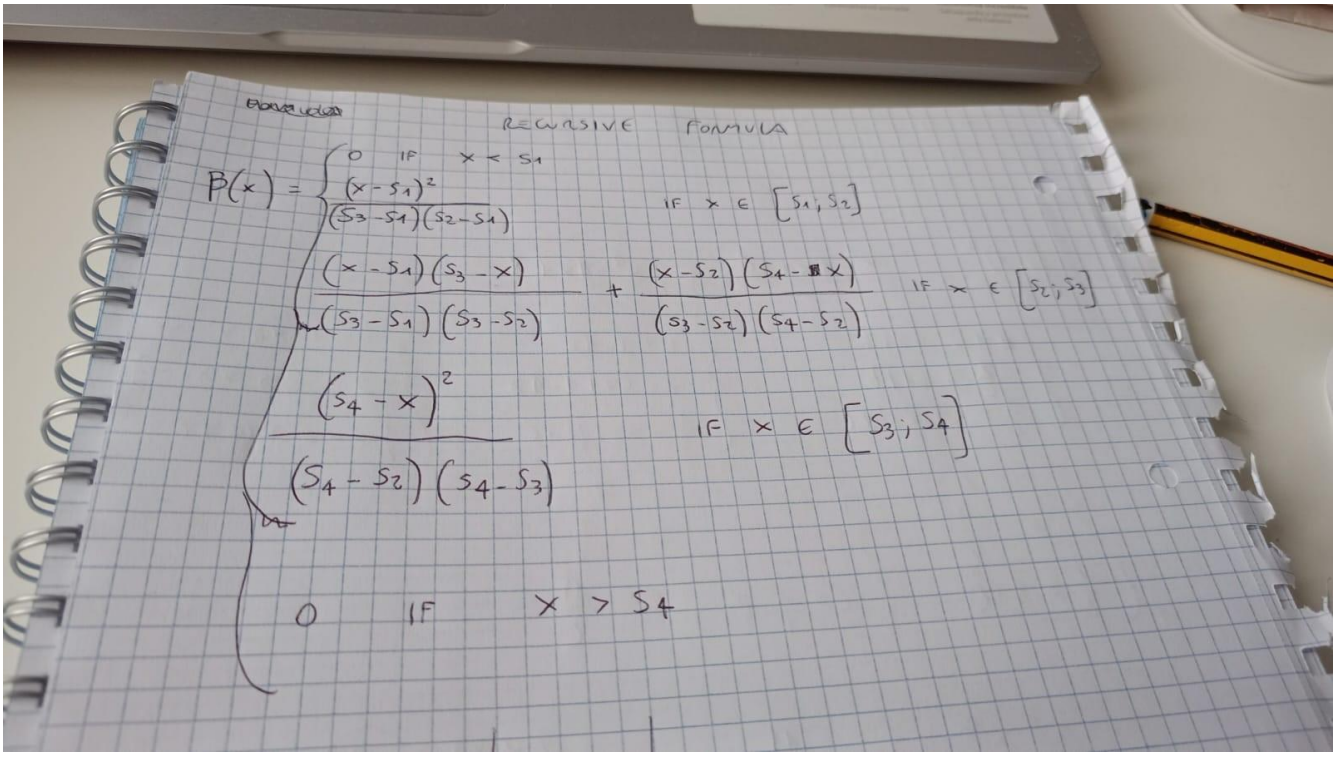
The black cluster obtained here is a little too sparse and heterogeneous. We should prefere include in the "red" cluster the extreme points in the lower part of the plot . Let's exclude the t (since we want homogenous clusters) and compare k-means and gaussian mixture (both with k=3), basing our final conclusion on the ARI.

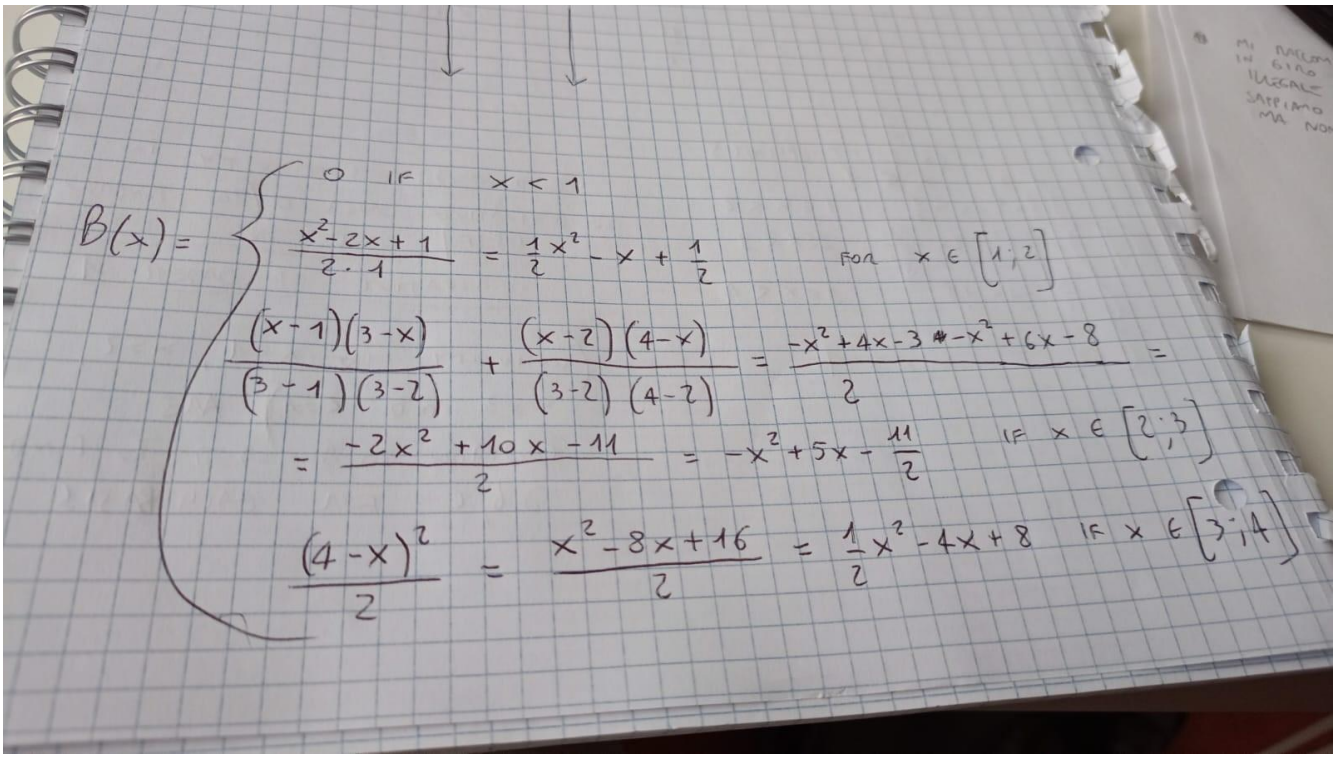
```
adjustedRandIndex(trueclass, gaussmixture$classification)

## [1] 0.4014239

adjustedRandIndex(trueclass, km3$cluster)

## [1] 0.4005486
```

Results are very similar, we can say that the Gauss.mixture is slightly better. However, the funFEM clustering has an higher Adjusted Rand Index. That's probably because the

Gauss,mixture is applied to the first two principal component, which are able to capture about an 80% of the variability.

Exercise 2





These polynomials respect the constraints requested by the exercise and B(x) is continuous up to the first derivative (B-splines have to be continuous up the d-2th derivative. Here d=3)

Exercise 3

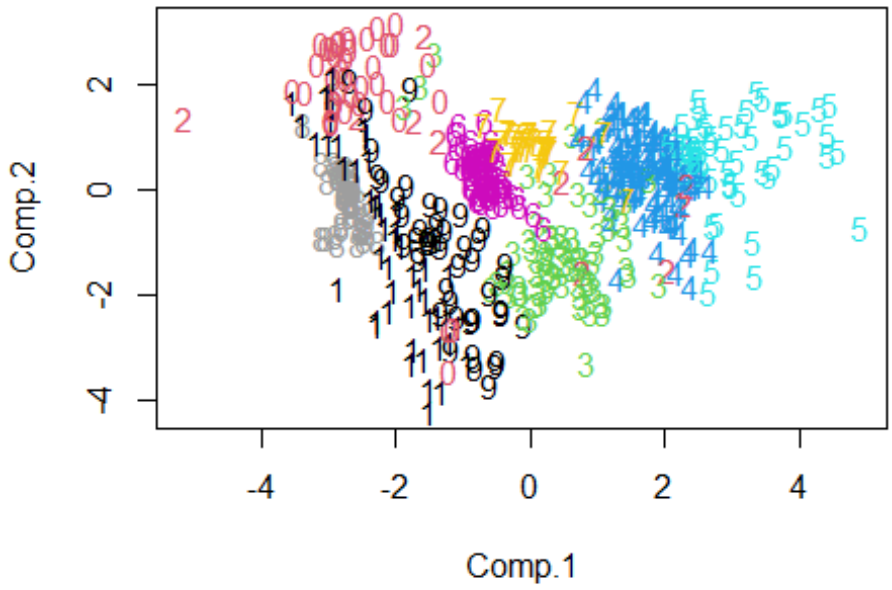
Point a

DC (discriminating coordinates): The aim is simply projecting high-dimensional data, already grouped in a given way, in a lower-dimensional subspace. Here the concept of homogeneity isn't mentioned (there isn't necessarily a more homogenous class and a non-homogeneous one). Speaking in mathematical terms, DC finds the first k projections vectors (usually 2) $C_1, \dots C_k$ that maximize $F_c = \frac{c'Qc}{c'Rc}$ where Q=B and R=W: W is the pooled within-groups scatter matrix and B is the between-groups scatter matrix. W is an adequate measure of the covariance structure. Later W has been replaced with $W_0 = \frac{1}{2}(S_1 + S_2)$.

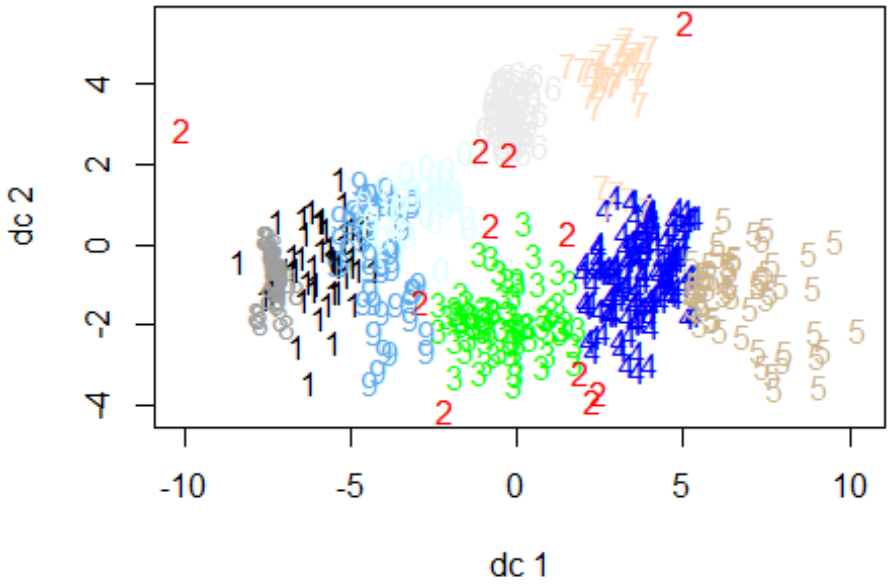
AWC: it's a sort of "improvement" from ADC and DC. Here the aim is to explicitely and adequately accounting for non homogeneity. AWC weights the object of the non-homogeneous class basing on the Mahalanobis distance to the homogeneous class. This feature is implemented in order to not give too much importance to extreme objects of the non-homogeneous class. This for example happens in ADC, and the risk is to excessively show the distance between the H-class and only the outliers (=extreme objects) of the NH-class. This approach is particularly useful when dealing to a problem of presence/absence of a certain feature (for example, a disease): the objects that shows the feature are likely to be very similar (and to create an homogeneous group), while the class characterized by the absence of the peculiarity could be very heterogeneous.

Point b

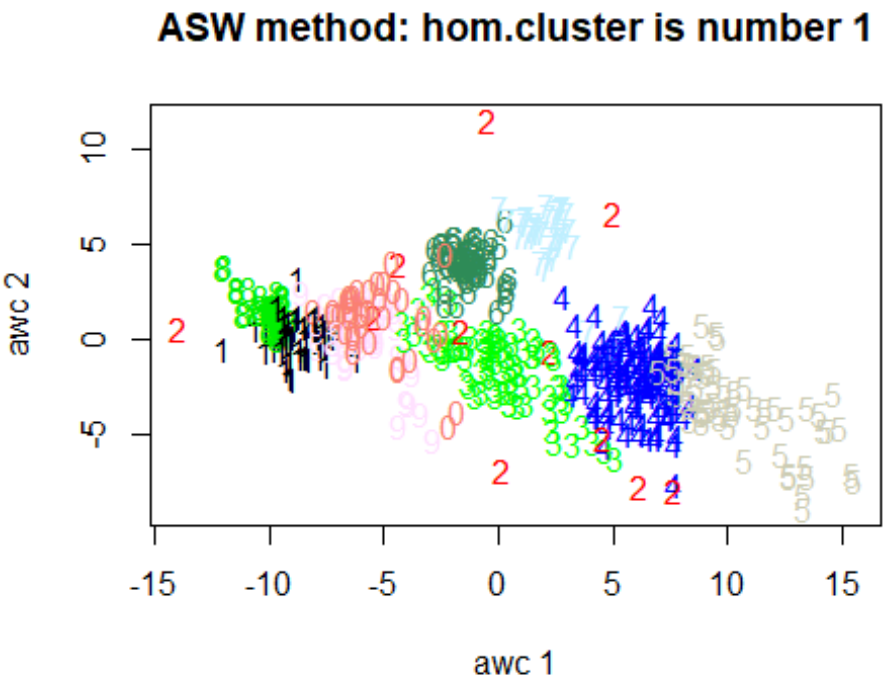
```
setwd("C:/Users/Veronesi/Desktop/uniBo/Magistrale/Modern Statistics and Big Data Analytics")
oliveoil <- read.table("oliveoil.DAT", header = T)
library(fpc)

## Warning: il pacchetto 'fpc' è stato creato con R versione 4.2.3

library(mclust)
molive <- Mclust(oliveoil,G=10)
solive <- scale(oliveoil[,3:10], T, T)
sprolive <- princomp(solive)
plot(sprolive$scores,col=molive$classification,
pch=clusym[molive$classification])
```

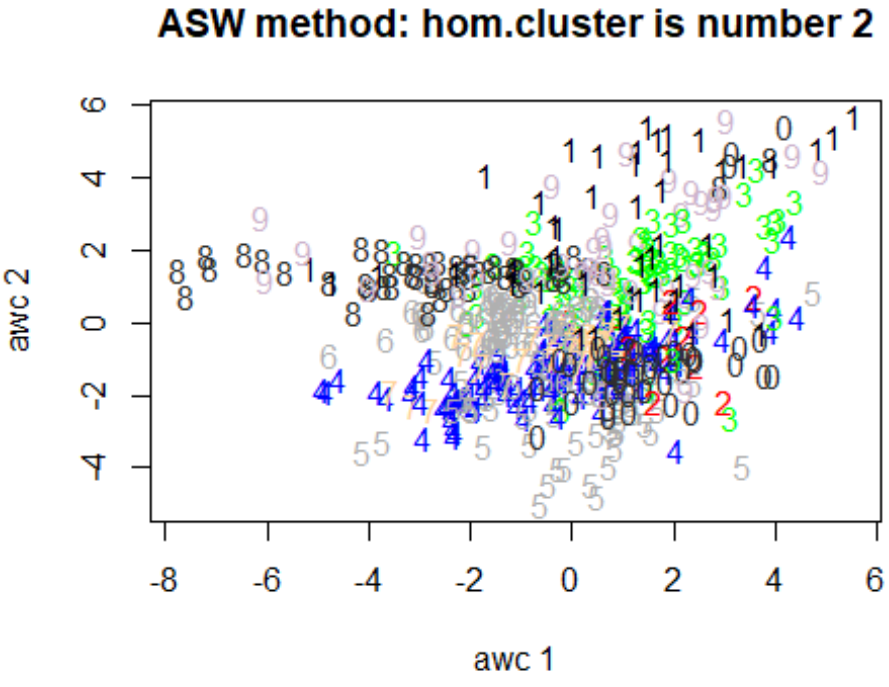


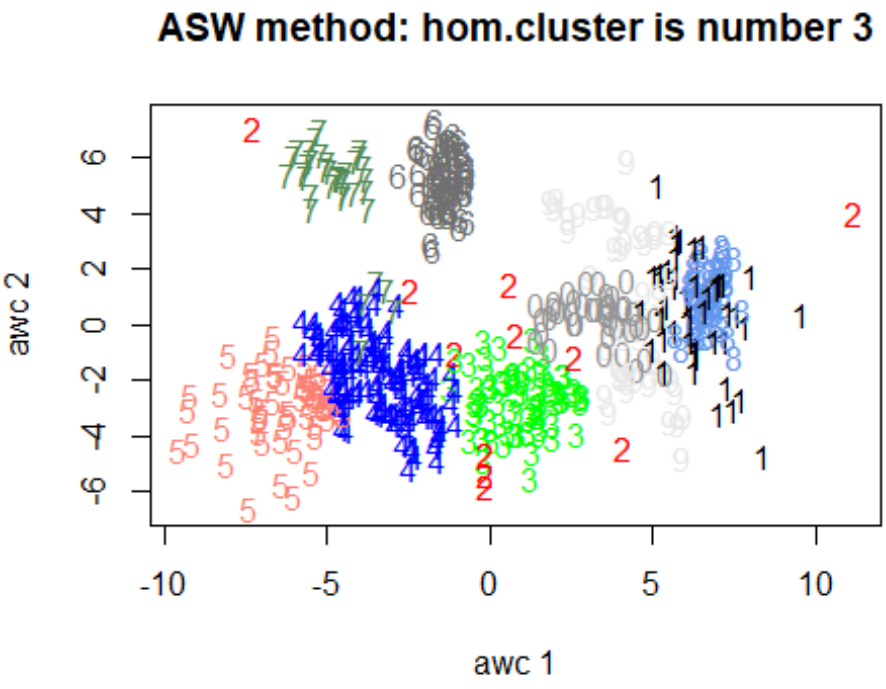

```
plotcluster(x=as.matrix(solive), clvecd = molive$classification, method = "dc")
```

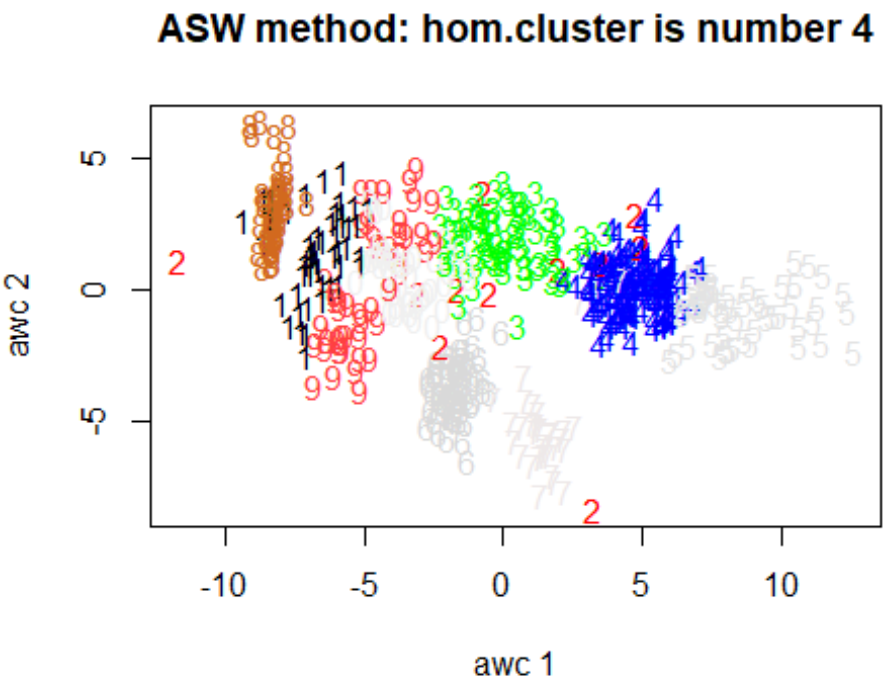


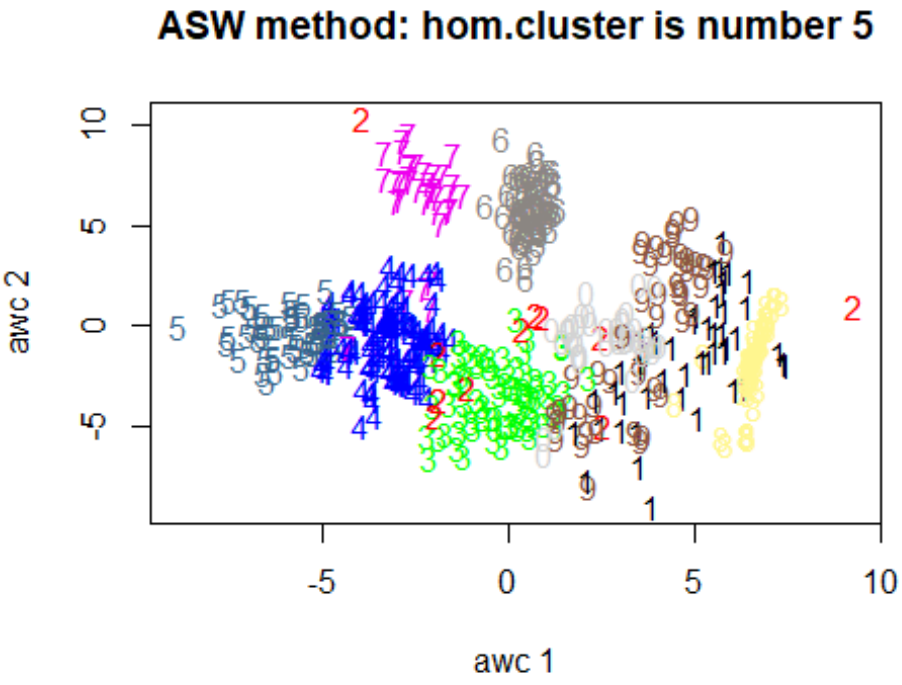

```
plotlist <- list()
for (i in 1:10) {
plotcluster(x=as.matrix(solive), clvecd = molive$classification, method = "awc",
clnum = i, main = paste("ASW method: hom.cluster is number", i))   ### visualize
asw with cluster n°k as homogenous class (-> 10 plots)
}
```

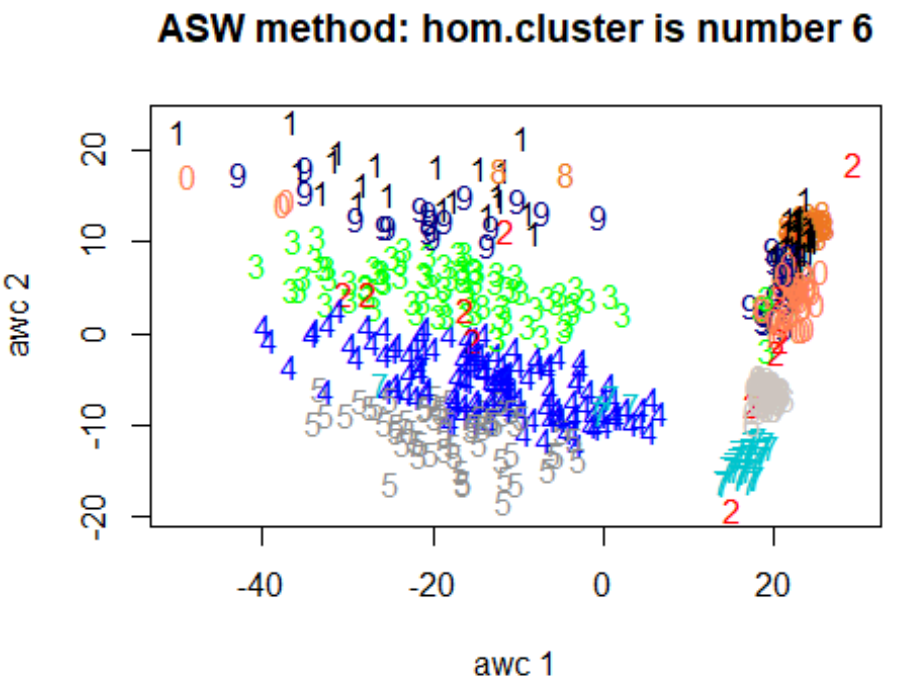

ASW method: hom.cluster is number 1
awc 2
awc 1
10
5
0
-5
-15
-10
-5
0
5
10
15

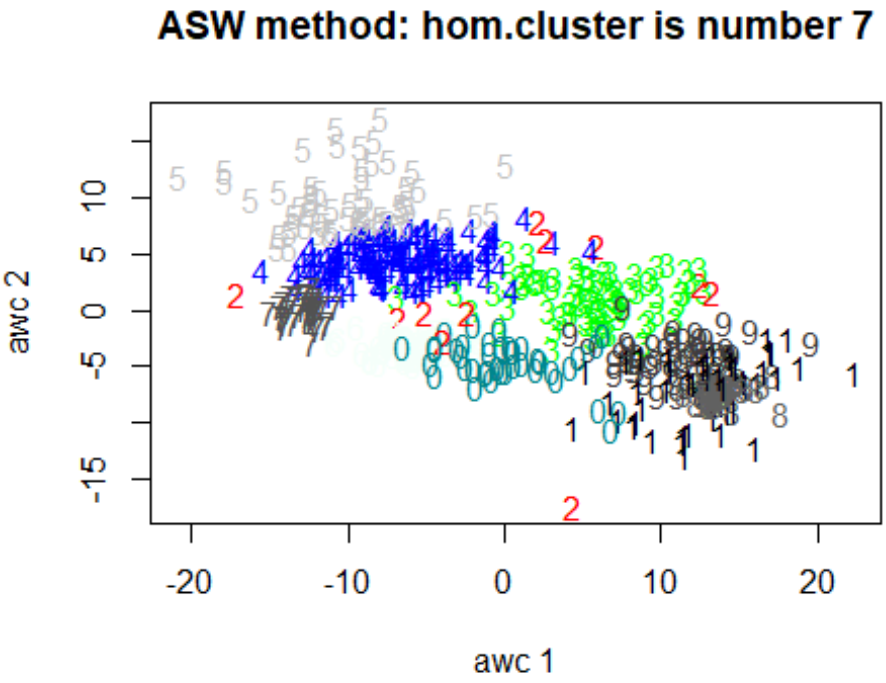

ASW method: hom.cluster is number 2
awc 2
awc 1
6
4
2
0
-2
-4
-8
-6
-4
-2
0
2
4
6

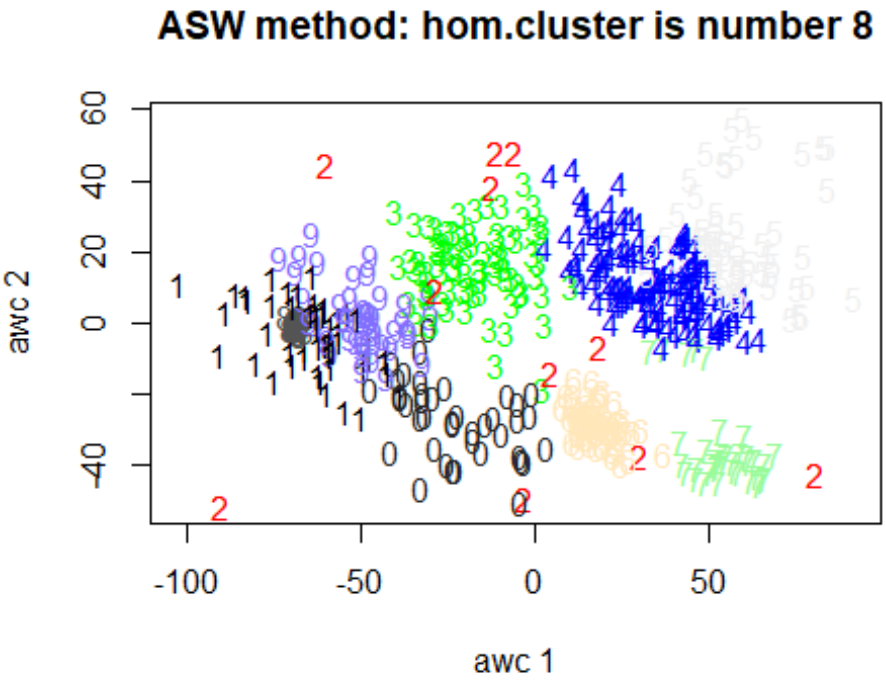

ASW method: hom.cluster is number 3
awc 2
awc 1
-10
-5
0
5
10
-6
-4
-2
0
2
4
6

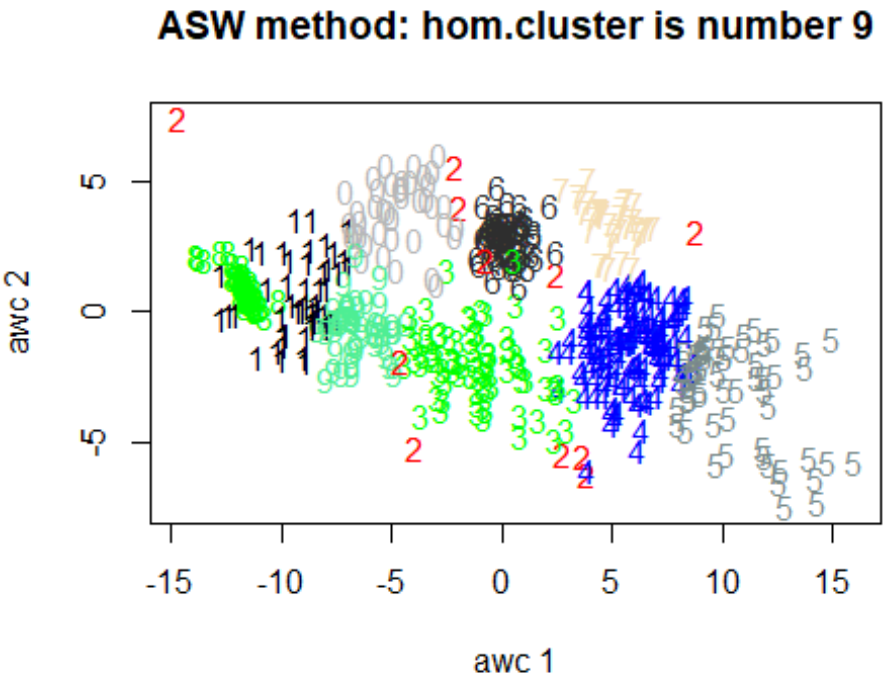

ASW method: hom.cluster is number 4
awc 2
awc 1
-10
-5
0
5
10
-5
0
5


ASW method: hom.cluster is number 5
awc 2
awc 1
-5
0
5
10

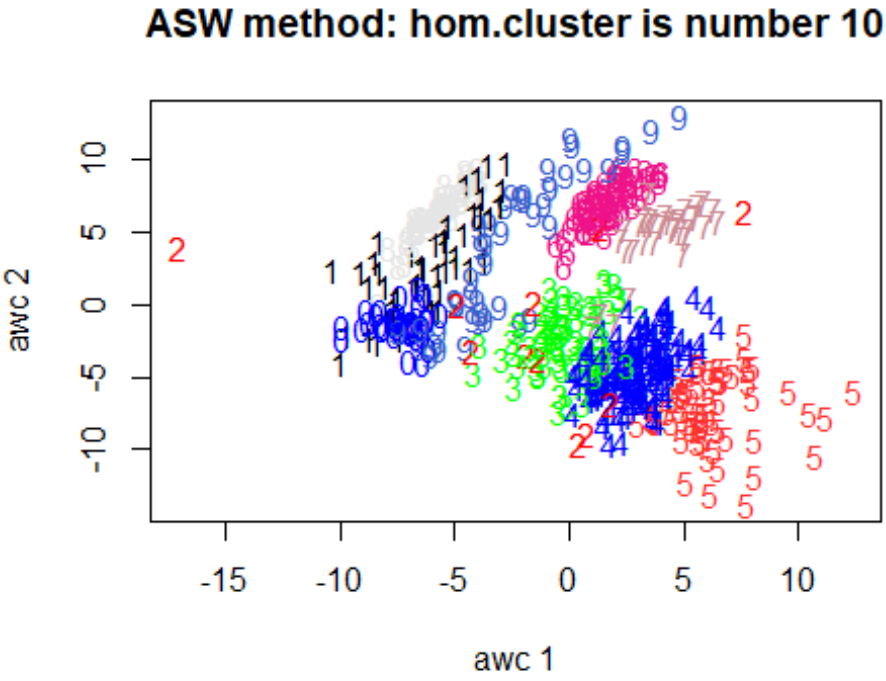

ASW method: hom.cluster is number 6
awc 2
awc 1
-40
-20
0
20


ASW method: hom.cluster is number 7
awc 2
awc 1
10
5
0
-5
-15
-20
-10
0
10
20


ASW method: hom.cluster is number 8
awc 2
awc 1
60
40
20
0
-40
-100
-50
0
50


ASW method: hom.cluster is number 9
awc 2
awc 1
-15
-10
-5
0
5
10
15
5
0
-5

Ff


ASW method: hom.cluster is number 10
awc 2
awc 1
-15
-10
-5
0
5
10
10
5
0
-5
-10

Here it is very difficult to represent separation between clusters since they are very close to each other. The DC plot produces more homogeneous cluster than the 2-principal components plot. Talking about ASW, it really depend on what cluster we treat as homogenous class: for example, it makes no sense to indicate cluster number 2 as the homogenous one. If we indicate an already homogenous cluster (such as cluster 6 or cluster 7) as H-class, ASW plot is quite good in terms of representing separation and homogeneity.

Point c

.....