Homework7

2022-11-27

```
library(fpc)
library(smacof)
```

```
## Caricamento del pacchetto richiesto: plotrix

## Caricamento del pacchetto richiesto: colorspace

## Caricamento del pacchetto richiesto: e1071

##
## Caricamento pacchetto: 'smacof'

## Il seguente oggetto è mascherato da 'package:base':
##
##     transform
```

```
library(cluster)
library(pdfCluster)
```

```
## pdfCluster 1.0-3
```

```
library(prabclus)
```

```
## Warning: il pacchetto 'prabclus' è stato creato con R versione 4.2.2

## Caricamento del pacchetto richiesto: MASS

## Caricamento del pacchetto richiesto: mclust

## Warning: il pacchetto 'mclust' è stato creato con R versione 4.2.2

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

##
## Caricamento pacchetto: 'prabclus'

## Il seguente oggetto è mascherato da 'package:fpc':
##
##     con.comp
```

```
library(mclust)
library(teigen)
library(mixsmsn)
```

```
## Caricamento del pacchetto richiesto: mvtnorm
```

```
##
## Caricamento pacchetto: 'mvtnorm'
```

```
## Il seguente oggetto è mascherato da 'package:mclust':
##
##     dmvnorm
```

```
library(fda)
```

```
## Warning: il pacchetto 'fda' è stato creato con R versione 4.2.2
```

```
## Caricamento del pacchetto richiesto: splines
```

```
## Caricamento del pacchetto richiesto: fds
```

```
## Warning: il pacchetto 'fds' è stato creato con R versione 4.2.2
```

```
## Caricamento del pacchetto richiesto: rainbow
```

```
## Warning: il pacchetto 'rainbow' è stato creato con R versione 4.2.2
```

```
## Caricamento del pacchetto richiesto: pcaPP
```

```
## Warning: il pacchetto 'pcaPP' è stato creato con R versione 4.2.2
```

```
## Caricamento del pacchetto richiesto: RCurl
```

```
## Caricamento del pacchetto richiesto: deSolve
```

```
## Warning: il pacchetto 'deSolve' è stato creato con R versione 4.2.2
```

```
##
## Caricamento pacchetto: 'fda'
```

```
## Il seguente oggetto è mascherato da 'package:graphics':
##
##     matplot
```

```
library(flexmix)
```

```
## Caricamento del pacchetto richiesto: lattice
```

```
##
## Caricamento pacchetto: 'lattice'

## Il seguente oggetto è mascherato da 'package:fda':
##
##     melanoma
```

```
library(poLCA)
```

```
## Warning: il pacchetto 'poLCA' è stato creato con R versione 4.2.2

## Caricamento del pacchetto richiesto: scatterplot3d
```

```
library(scatterplot3d)

data(election)

election12<-election[,1:12] #extract categorical variables to be clustered

electionNA<-election12
for (i in 1:12){
  levels(electionNA[,i]) <- c(levels(election12[,i]),"NA")
  electionNA[is.na(election12[,i]),i] <- "NA"
}
#This dataset contains the NA values
```

(a) Compute a latent class clustering with 3 clusters using poLCA.

```
f<-cbind(MORALG,CARESG,KNOWG,LEADG,DISHONG,INTELG,MORALB,CARESB,KNOWB,LEADB,DISHONB,INTELB)~1

clas1<-poLCA(f, electionNA, nclass=3)
```

```
## Conditional item response (column) probabilities,
##  by outcome variable, for each class (row)
##
## $MORALG
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.5979       0.3449         0.0170            0.0232 0.0170
## class 2:            0.1343       0.3689         0.2732            0.1892 0.0344
## class 3:            0.0848       0.6045         0.1651            0.0174 0.1282
##
## $CARESG
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.4728       0.4255         0.0496            0.0440 0.0081
## class 2:            0.0437       0.2588         0.3976            0.2695 0.0304
## class 3:            0.0379       0.4933         0.2874            0.0757 0.1057
##
## $KNOWG
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.6984       0.2756         0.0000            0.0243 0.0017
## class 2:            0.1596       0.5335         0.2229            0.0690 0.0151
```

```
## class 3:            0.0518       0.7609         0.1135            0.0104 0.0633
##
## $LEADG
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.4702       0.4440         0.0494            0.0251 0.0113
## class 2:            0.0370       0.2288         0.4594            0.2527 0.0220
## class 3:            0.0189       0.5253         0.3180            0.0371 0.1007
##
## $DISHONG
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.0376       0.0477         0.2665            0.6111 0.0371
## class 2:            0.1728       0.3046         0.3027            0.1748 0.0450
## class 3:            0.0210       0.1547         0.4469            0.2269 0.1505
##
## $INTELG
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.7014       0.2679         0.0090            0.0217 0.0000
## class 2:            0.2048       0.5328         0.1816            0.0672 0.0136
## class 3:            0.0592       0.7637         0.0993            0.0217 0.0561
##
## $MORALB
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.1219       0.3621         0.2990            0.1339 0.0831
## class 2:            0.4467       0.4920         0.0412            0.0112 0.0089
## class 3:            0.0332       0.5251         0.2242            0.0383 0.1792
##
## $CARESB
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.0232       0.1272         0.3868            0.4361 0.0266
## class 2:            0.2419       0.6159         0.1138            0.0204 0.0081
## class 3:            0.0059       0.2850         0.4266            0.1685 0.1141
##
## $KNOWB
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.1120       0.3431         0.3214            0.2024 0.0212
## class 2:            0.3501       0.5946         0.0553            0.0000 0.0000
## class 3:            0.0256       0.5821         0.2655            0.0506 0.0762
##
## $LEADB
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.0461       0.2734         0.3863            0.2456 0.0486
## class 2:            0.3974       0.5626         0.0304            0.0042 0.0054
## class 3:            0.0200       0.5282         0.2813            0.0643 0.1061
##
## $DISHONB
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.0850       0.2690         0.3239            0.2161 0.1059
## class 2:            0.0226       0.0719         0.2730            0.6068 0.0257
## class 3:            0.0227       0.1622         0.4659            0.1652 0.1840
##
## $INTELB
##           1 Extremely well 2 Quite well 3 Not too well 4 Not well at all     NA
## class 1:            0.1600       0.3786         0.2728            0.1710 0.0177
## class 2:            0.4031       0.5643         0.0297            0.0000 0.0029
```

```
## class 3:            0.0278       0.6293         0.2176            0.0395 0.0858
##
## Estimated class population shares
##  0.2654 0.3235 0.4111
##
## Predicted class memberships (by modal posterior prob.)
##  0.2611 0.3182 0.4207
##
## =========================================================
## Fit for 3 latent classes:
## =========================================================
## number of observations: 1785
## number of estimated parameters: 146
## residual degrees of freedom: 1639
## maximum log-likelihood: -25891.5
##
## AIC(3): 52075
## BIC(3): 52876.13
## G^2(3): 25474.41 (Likelihood ratio/deviance statistic)
## X^2(3): 424598787823 (Chi-square goodness of fit)
##
```

log-likelihood: -25990.17 df= 1639 BIC: 53073.46

(b) Compute a latent class clustering with 3 clusters using flexmixedruns.

```
set.seed(1234)
clas2<-flexmixedruns(electionNA, continuous = 0, discrete = 12, n.cluster = 3)
```

```
## k=  3  new best fit found in run  1
## Nonoptimal or repeated fit found in run  2
## k=  3  new best fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## k=  3  new best fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  3  BIC=  52864.71
```

Cluster sizes: 1 2 3 683 845 257

log-likelihood: -25885.28 df=146 BIC: 52863.69

(c) Compute a distance-based clustering of your choice with 3 clusters based on the simple matching distance.

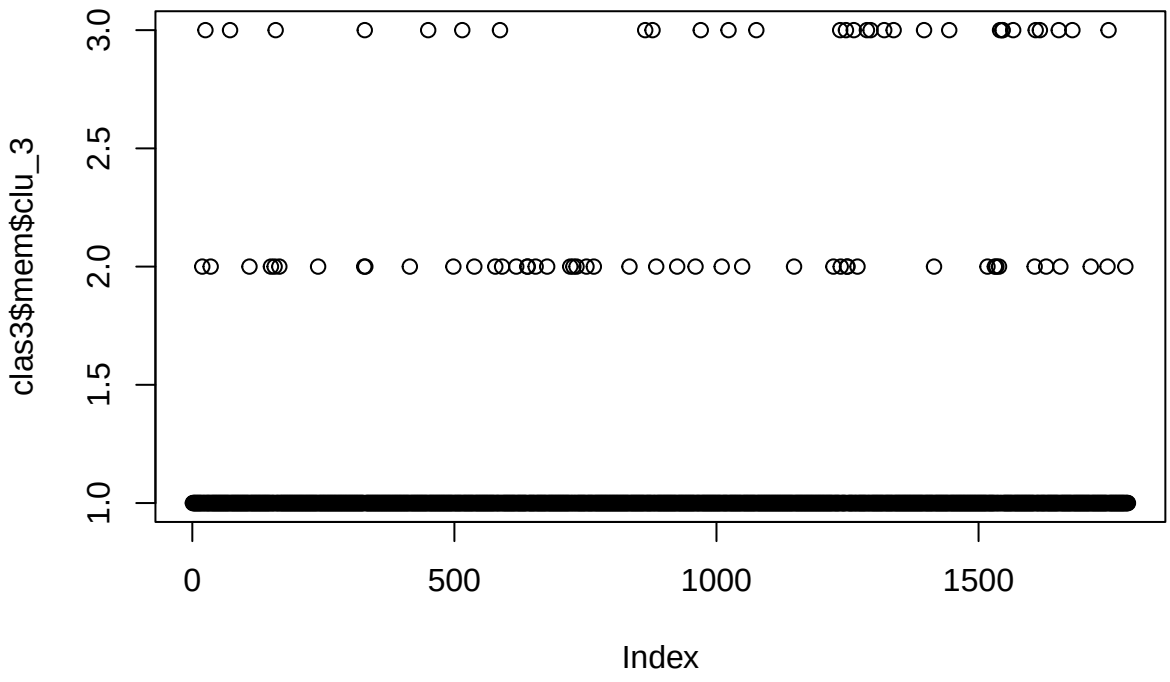
```
library(nomclust)
```

```
## Warning: il pacchetto 'nomclust' è stato creato con R versione 4.2.2
```

```
?nomclust
```

```
## avvio in corso del server httpd per la guida ... fatto
```

```
clas3<-nomclust(electionNA, measure = "sm", method = "average", clu.high = 3)

plot(clas3$mem$clu_3)
```




```
summary(clas3$mem$clu_3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00    1.06    1.00    3.00
```

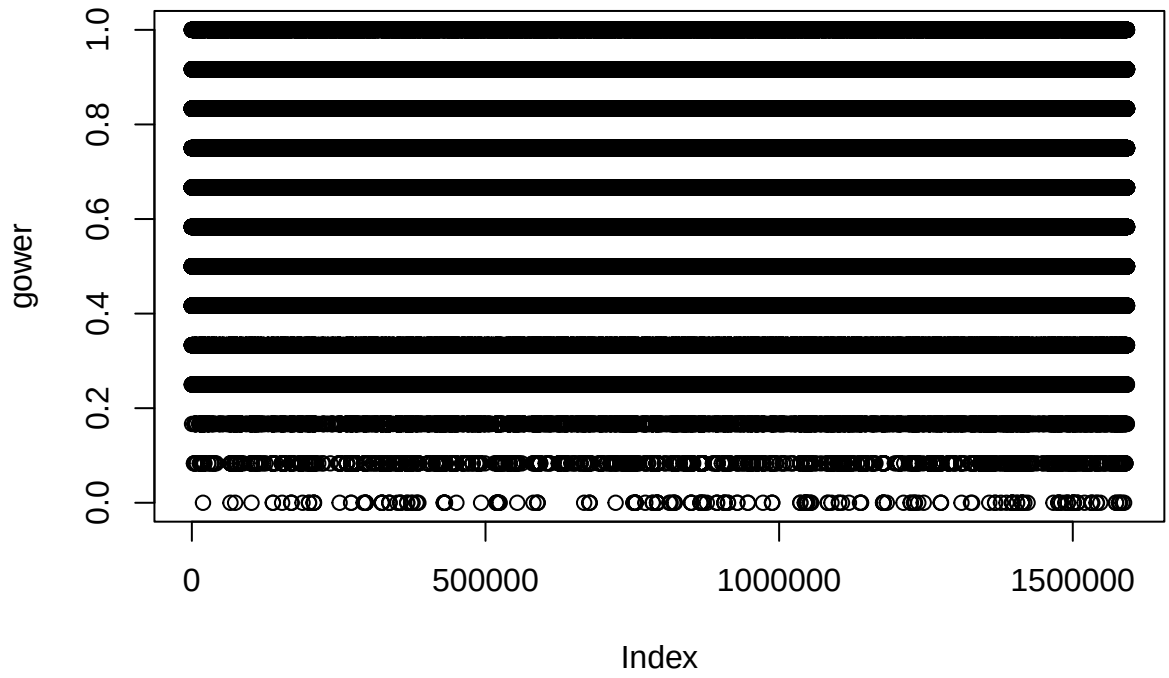
```
table(clas3$mem$clu_3)
```

```
##
##    1    2    3
## 1708   47   30
```

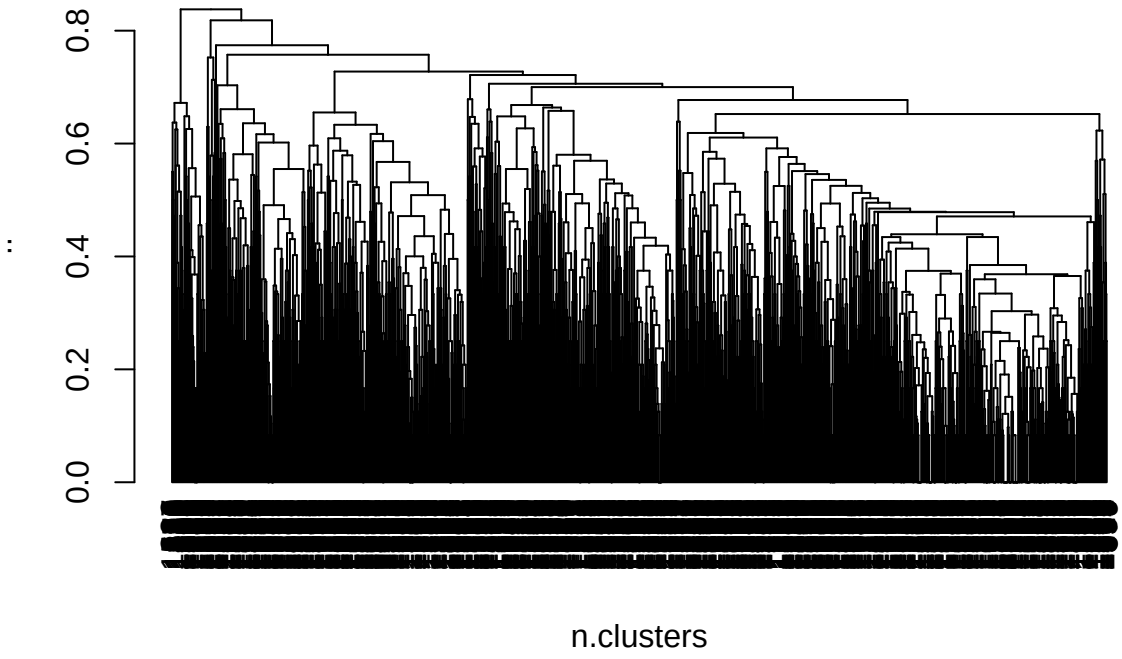
1 2 3 1708 47 30

(d) Compute a dissimilarity-based clustering of your choice with 3 clusters using this dissimilarity on the election12 data with missing values.

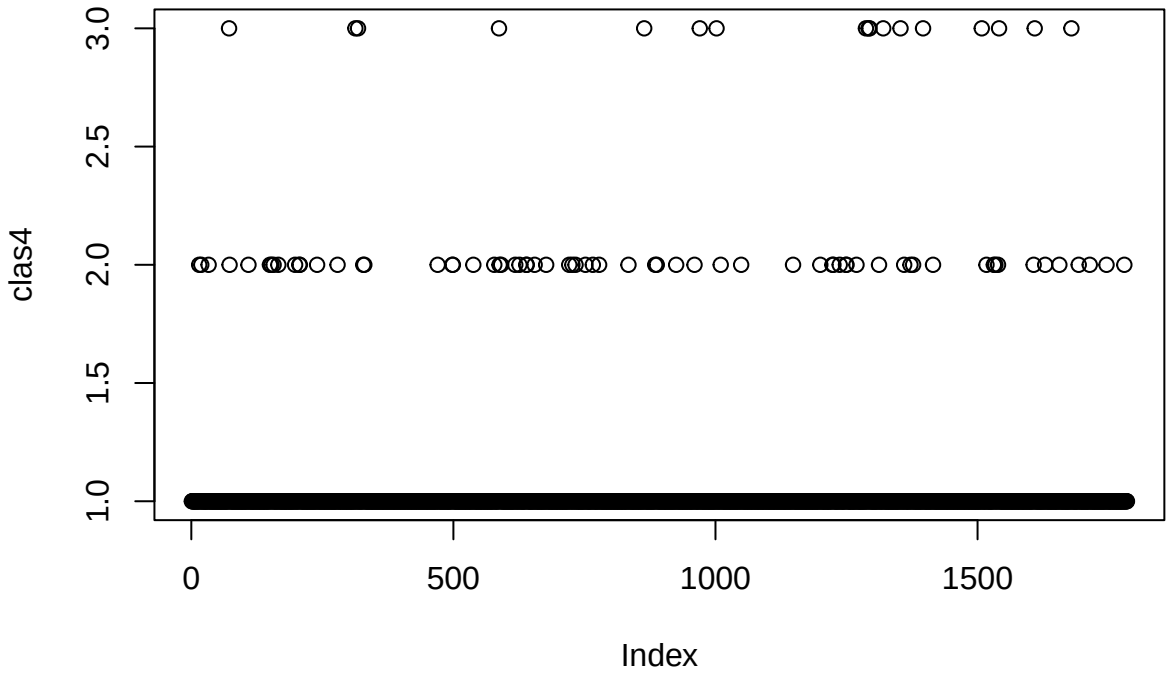
```
gower<-daisy(electionNA, metric="gower")
plot(gower)
```




```
#election12 doesn't work
elec.gower <- hclust(gower,method="average")
plot(as.dendrogram(elec.gower), main="Average Linkage-Gower distance", xlab = "n.clusters", ylab="..")
```




```
clas4<-cutree(elec.gower, k=3)
plot(clas4)
```




```
table(clas4)
```

```
## clas4
##    1    2    3
## 1702   66   17
```

(e) Compute a latent class clustering using flexmixedruns with estimated number of clusters.

```
set.seed(1234)
clas5<-flexmixedruns(electionNA, continuous = 0, discrete = 12, n.cluster = 1:10)
```

```
## k=  1  new best fit found in run  1
## k=  1  BIC=  57445.61
## k=  2  new best fit found in run  1
## Nonoptimal or repeated fit found in run  2
## k=  2  new best fit found in run  3
## k=  2  new best fit found in run  4
## Nonoptimal or repeated fit found in run  5
## k=  2  new best fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
```

```
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  2  BIC=  54295.5
## k=  3  new best fit found in run  1
## k=  3  new best fit found in run  2
## k=  3  new best fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## k=  3  new best fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## k=  3  new best fit found in run  20
## k=  3  BIC=  52863.68
## k=  4  new best fit found in run  1
## k=  4  new best fit found in run  2
## Nonoptimal or repeated fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  4  BIC=  51682.63
## k=  5  new best fit found in run  1
## k=  5  new best fit found in run  2
```

```
## Nonoptimal or repeated fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  5  BIC=  51158.93
## k=  6  new best fit found in run  1
## k=  6  new best fit found in run  2
## Nonoptimal or repeated fit found in run  3
## k=  6  new best fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  6  BIC=  50858.5
## k=  7  new best fit found in run  1
## k=  7  new best fit found in run  2
## k=  7  new best fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
```

```
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## k=  7  new best fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  7  BIC=  50760.26
## k=  8  new best fit found in run  1
## Nonoptimal or repeated fit found in run  2
## k=  8  new best fit found in run  3
## k=  8  new best fit found in run  4
## Nonoptimal or repeated fit found in run  5
## k=  8  new best fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## Nonoptimal or repeated fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  8  BIC=  50746.53
## k=  9  new best fit found in run  1
## Nonoptimal or repeated fit found in run  2
## Nonoptimal or repeated fit found in run  3
## Nonoptimal or repeated fit found in run  4
## Nonoptimal or repeated fit found in run  5
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## k=  9  new best fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  9  BIC=  50810.59
## k=  10  new best fit found in run  1
## k=  10  new best fit found in run  2
## Nonoptimal or repeated fit found in run  3
## k=  10  new best fit found in run  4
## k=  10  new best fit found in run  5
```

```
## Nonoptimal or repeated fit found in run  6
## Nonoptimal or repeated fit found in run  7
## Nonoptimal or repeated fit found in run  8
## Nonoptimal or repeated fit found in run  9
## k=  10  new best fit found in run  10
## Nonoptimal or repeated fit found in run  11
## Nonoptimal or repeated fit found in run  12
## Nonoptimal or repeated fit found in run  13
## Nonoptimal or repeated fit found in run  14
## Nonoptimal or repeated fit found in run  15
## Nonoptimal or repeated fit found in run  16
## Nonoptimal or repeated fit found in run  17
## Nonoptimal or repeated fit found in run  18
## Nonoptimal or repeated fit found in run  19
## Nonoptimal or repeated fit found in run  20
## k=  10  BIC=  50903.72
```

```
clas5$optimalk
```

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

The optimal number of clusters is 8. Cluster sizes: 1 2 3 4 5 6 7 8 131 202 107 177 362 327 250 229

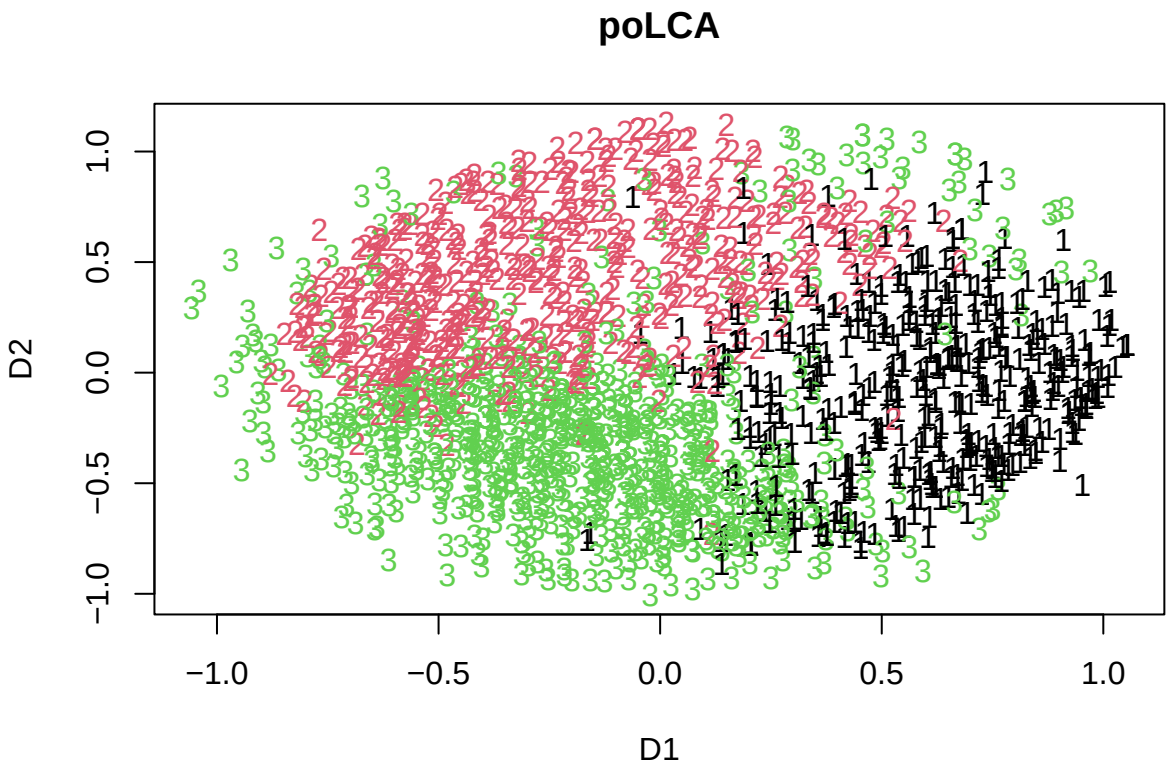
Compute ARIs for every pair of clusterings.

```
ARI12<-adjustedRandIndex(clas1$predclass,clas2$flexout[[3]]@cluster)
ARI13<-adjustedRandIndex(clas1$predclass,clas3$mem$clu_3)
ARI14<-adjustedRandIndex(clas1$predclass,clas4)
ARI15<-adjustedRandIndex(clas1$predclass,clas5$flexout[[8]]@cluster)
ARI23<-adjustedRandIndex(clas2$flexout[[3]]@cluster,clas3$mem$clu_3)
ARI24<-adjustedRandIndex(clas2$flexout[[3]]@cluster,clas4)
ARI25<-adjustedRandIndex(clas2$flexout[[3]]@cluster,clas5$flexout[[8]]@cluster)
ARI34<-adjustedRandIndex(clas3$mem$clu_3,clas4)
ARI35<-adjustedRandIndex(clas3$mem$clu_3,clas5$flexout[[8]]@cluster)
ARI45<-adjustedRandIndex(clas4,clas5$flexout[[8]]@cluster)


vect11<-c(0,ARI12, ARI13,ARI14,ARI15)
vect22<-c(ARI12,0,ARI23,ARI24,ARI25)
vect33<-c(ARI13,ARI23,0,ARI34,ARI35)
vect44<-c(ARI14,ARI24,ARI34,0,ARI45)
vect55<-c(ARI15,ARI25,ARI35,ARI45,0)

ARI<-cbind(vect11,vect22,vect33,vect44, vect55)
ARI
```

```
##            vect11     vect22       vect33       vect44     vect55
## [1,]  0.000000000 0.34874809 -0.006678225 -0.009536389 0.23722218
## [2,]  0.348748093 0.00000000  0.072436338  0.082505828 0.24452615
## [3,] -0.006678225 0.07243634  0.000000000  0.673363509 0.01630295
## [4,] -0.009536389 0.08250583  0.673363509  0.000000000 0.01670301
## [5,]  0.237222178 0.24452615  0.016302952  0.016703010 0.00000000
```

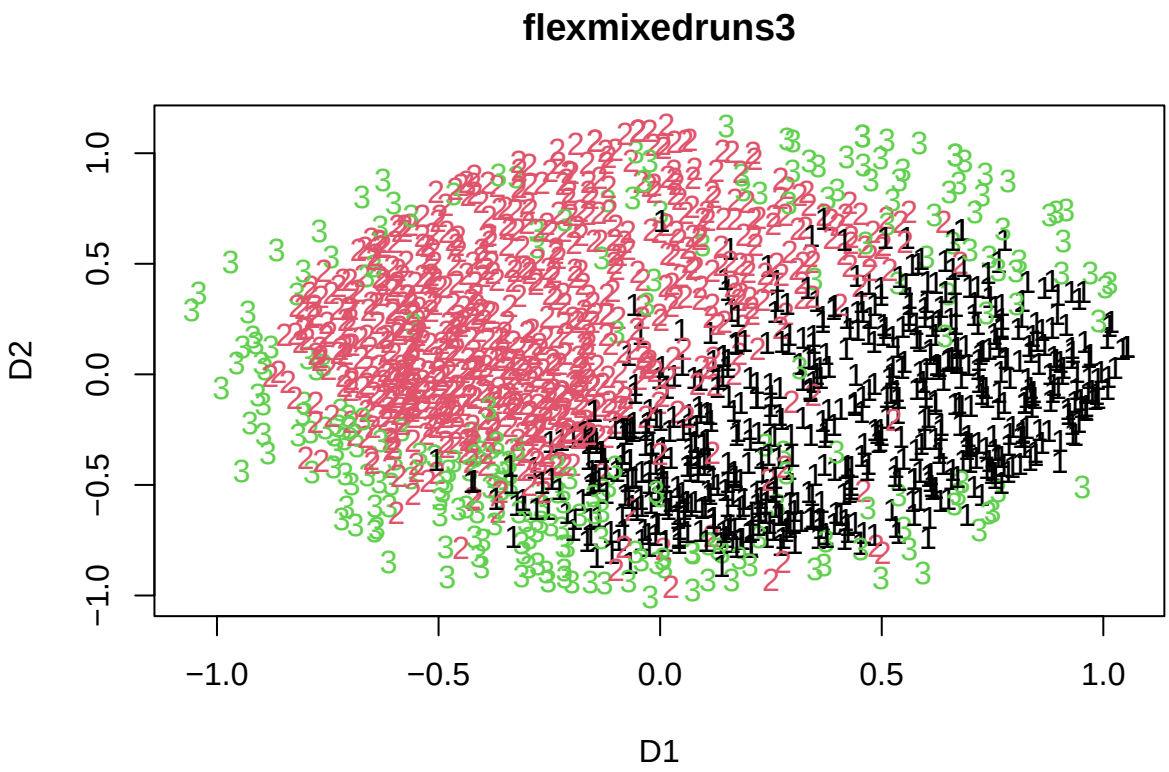
ARI values are very low for every pairs considered, the one that assume an almost good value is the ARI34 that is equal to 0.67336. This means that model3 with the average linkage method and model4 with the gower distance matrix perform a little bit in a similar way in terms of clustering.

```
dist<-sm(electionNA)
mds<-mds(dist, ndim=2)

plot(mds$conf,pch=clusym[clas1$predclass],col=clas1$predclass,main="poLCA")
```



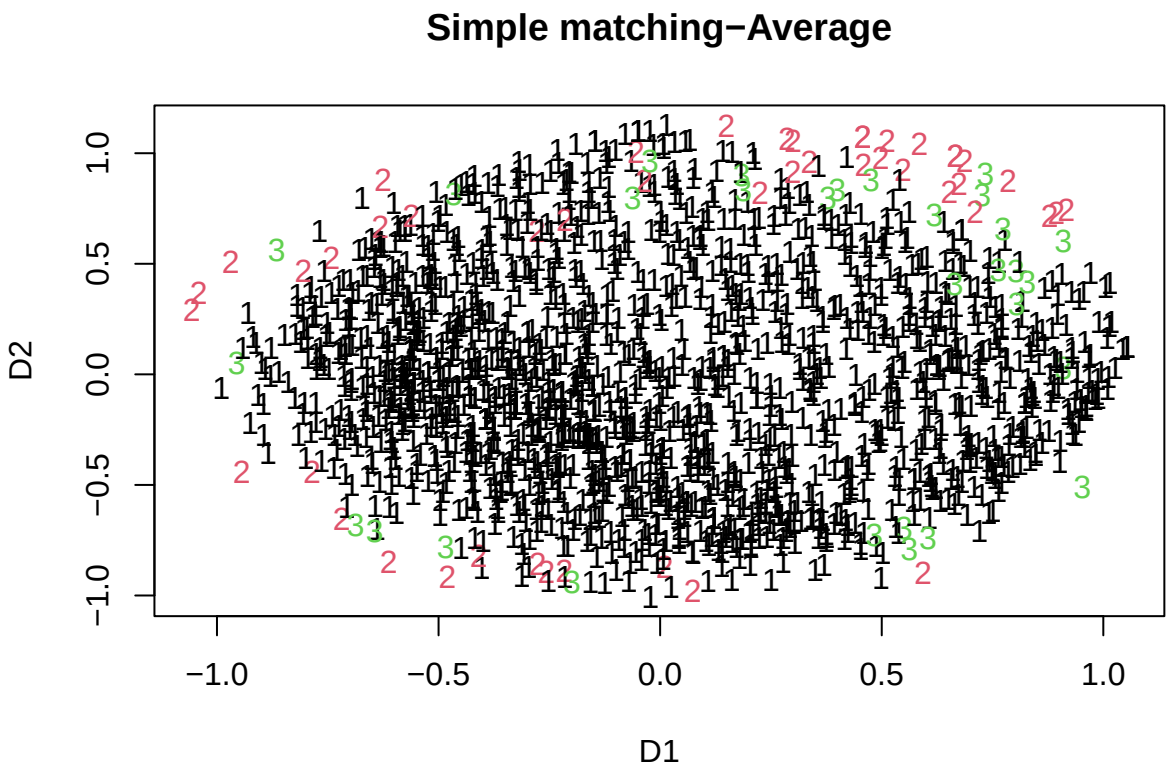
The cluster 2 is the better represented while the third is so confused.

```
plot(mds$conf,pch=clusym[clas2$flexout[[3]]@cluster],col=clas2$flexout[[3]]@cluster,main="flexmixedruns
```



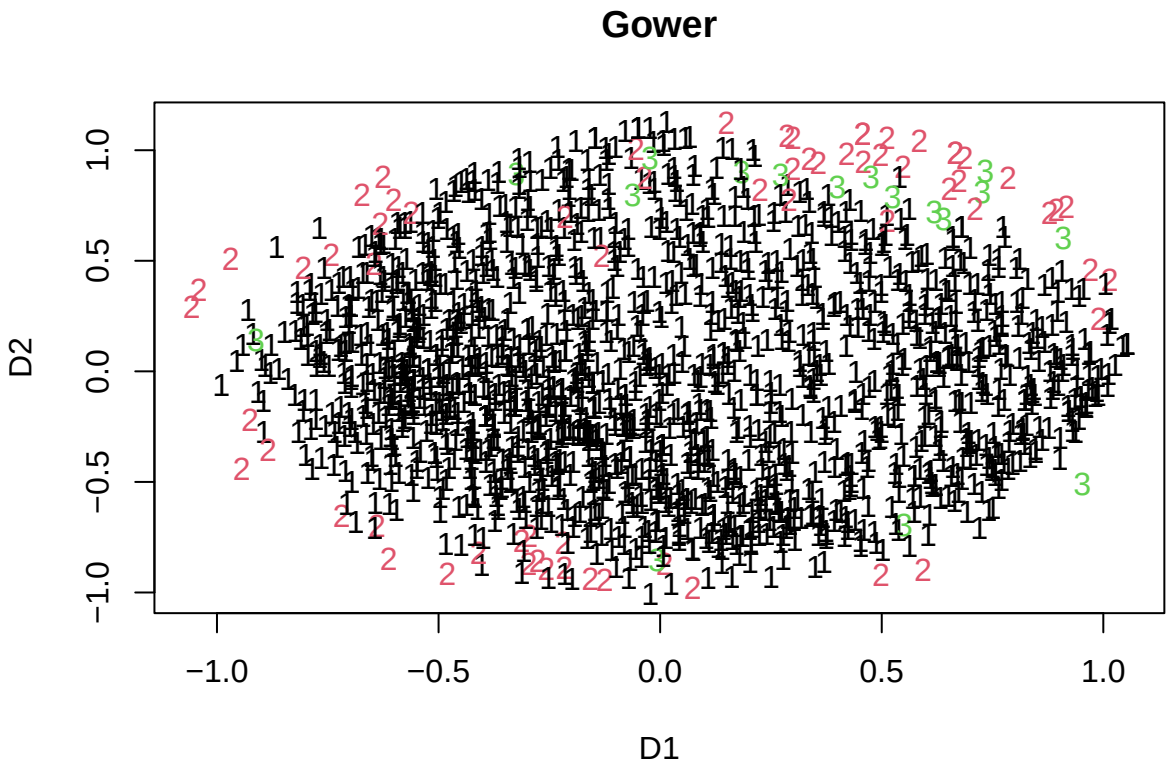
This clustering is similar to the previous one if we look at the third cluster composition, first and second clusters are sufficiently good separated.

```
plot(mds$conf,pch=clusym[clas3$mem$clu_3],col=clas3$mem$clu_3,main="Simple matching-Average")
```



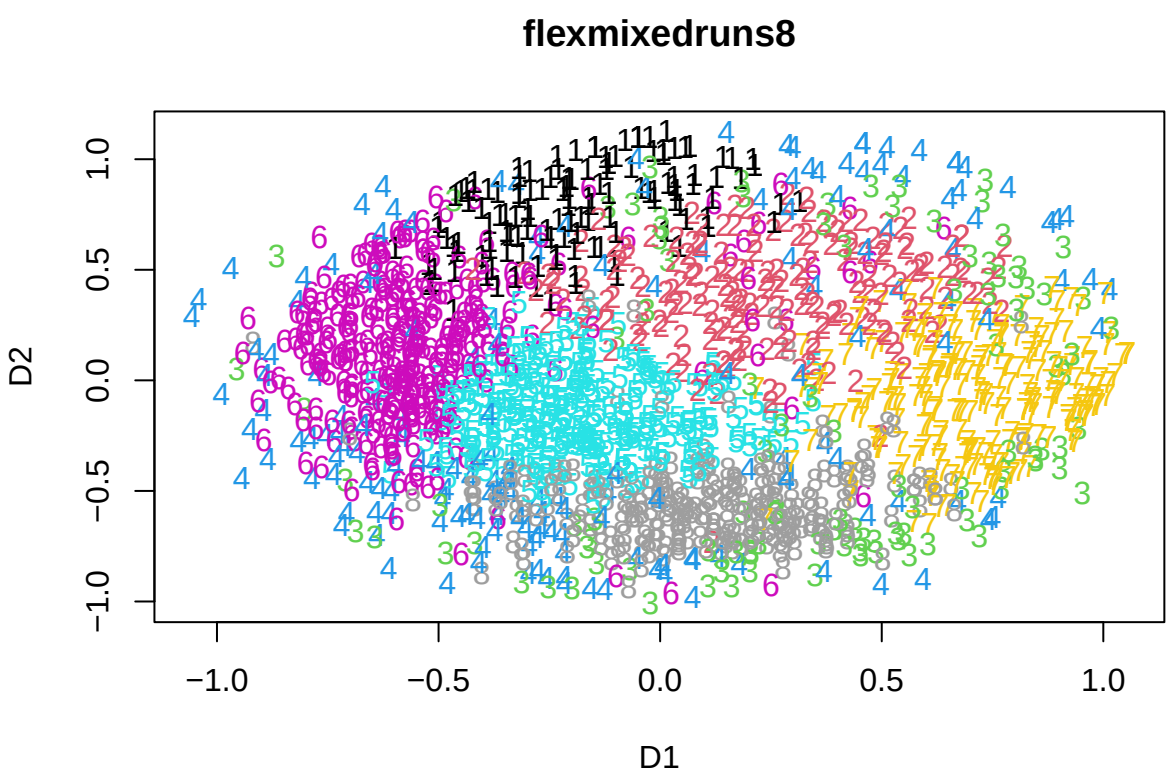
In this plot the biggest majority of the units are classified in the first cluster while the other two clusters are composed by scattered observations.

```
plot(mds$conf,pch=clusym[clas4],col=clas4,main="Gower")
```



As the previous plot the biggest majority of observations belong to cluster 1 while the others are scattered.

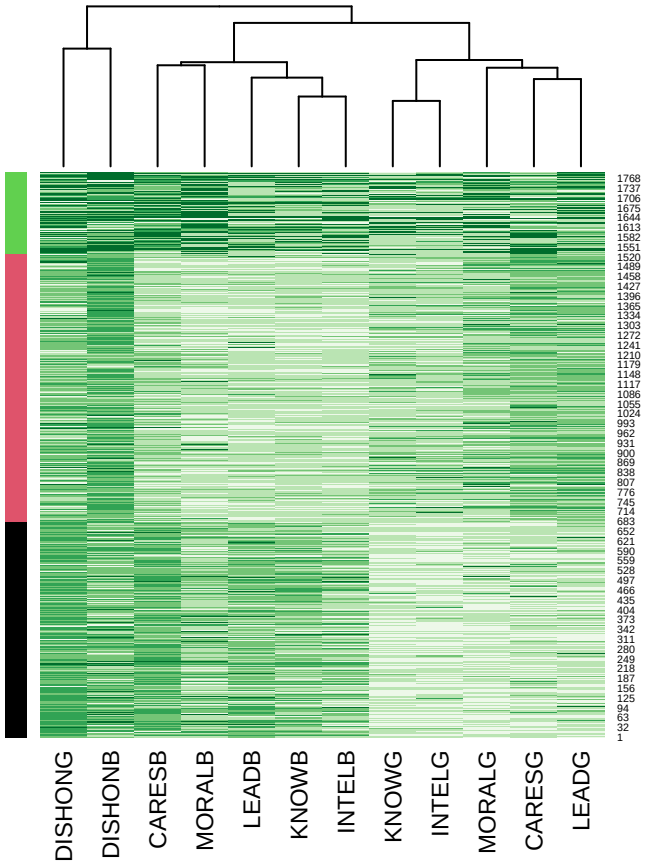
```
plot(mds$conf,pch=clusym[clas5$flexout[[8]]@cluster],col=clas5$flexout[[8]]@cluster,main="flexmixedruns
```



This plot is confused if we want to define clusters 3, 4 and 6, it can not provide a good classification.

2. For two different latent class clusterings computed in question 1 on the electionwithna-data produce heatmaps. Comment on the plots. Do you find the clusters convincing? Why or why not? Is there evidence against local independence?

```
library(RColorBrewer)
heatmap(data.matrix(electionNA)[order(clas2$flexout[[3]]@cluster),],
        Rowv=NA,
        Colv=clas1$predclass,
        RowSideColors=palette()[clas2$flexout[[3]]@cluster]
        [order(clas2$flexout[[3]]@cluster)],
        col=brewer.pal(5, 'Greens'),scale="none")
```

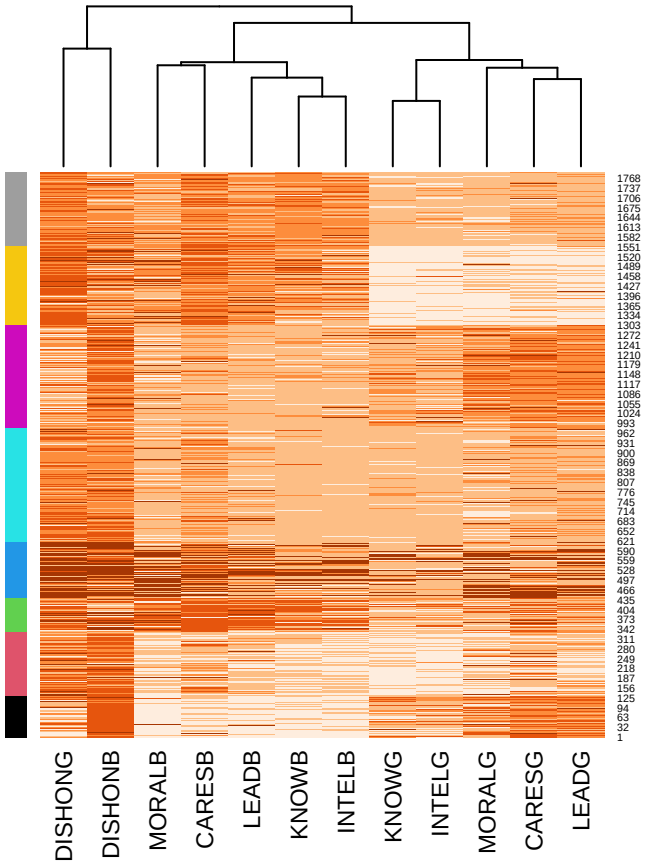



```
#legend(x="bottomright", legend=c("Extremely well","Quite well","Not too well","Not well at all"),fill=
```

The heatmap plot gives information about what values of what variables characterize the clusters. Heat Maps are graphical representations of data that utilize color-coded systems. The clusters in the dendrogram on the top are ordered based on the behaviour of the observations.

Variables 3-7 for the black cluster and last 5 variables for the pink one, assume more positive values for the responses. The green cluster show the majority of 'Not well at all' and 'Not too well' for all variables.

```
heatmap(data.matrix(electionNA)[order(clas5$flexout[[8]]@cluster),],
        Rowv=NA,
        Colv=clas3$mem$clu_3,
        RowSideColors=palette()[clas5$flexout[[8]]@cluster]
        [order(clas5$flexout[[8]]@cluster)],
        col=brewer.pal(5, 'Oranges'),scale="none",)
```




```
#legend(x="bottomright", legend=c("Extremely well","Quite well","Not too well","Not well at all"),fill=
```

Observations on the left are ones which return the worst degree of responses (not well at all/ not too well). The more positive responses are in the grey and yellow clusters for the last 5 variables and i the clusters black and pink for the variables 3-7.

3. Assume a situation with 10 categorical variables. Five variables are binary, three variables have three categories, and two variables have five categories. What is the number of free parameters for

(a) a general categorical model that models all possible probabilities,

```
p=10

(2^5)*(3^3)*(5^2)-1
```

```
## [1] 21599
```

The number of free parameters is the number of possible different observations minus one.

(b) a latent class mixture model with 4 mixture components?

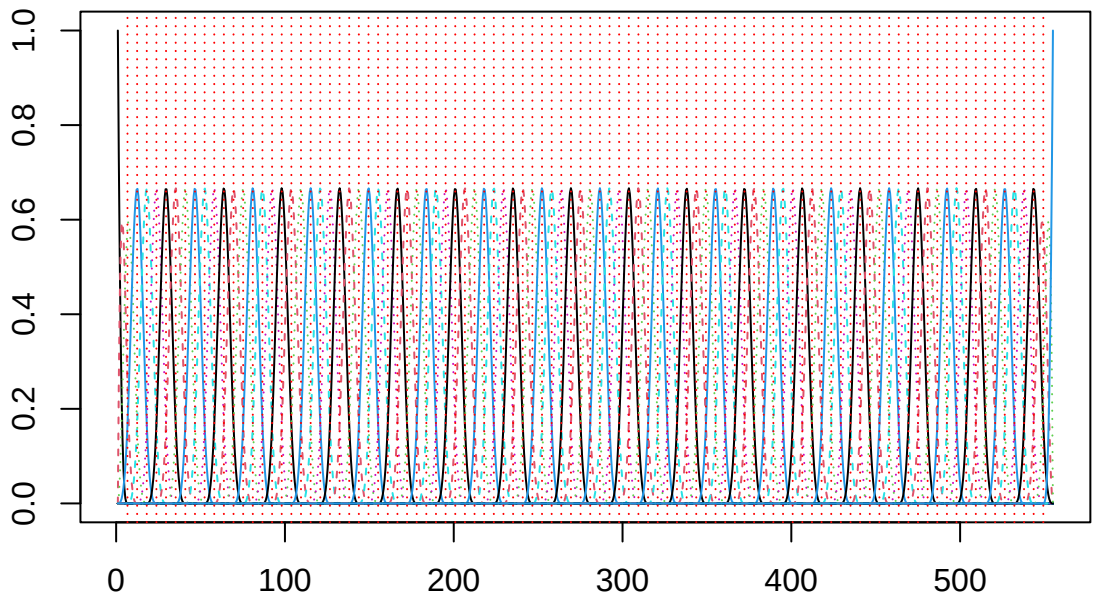
```
3+4*(1+1+1+1+1+2+2+2+4+4)
```

```
## [1] 79
```

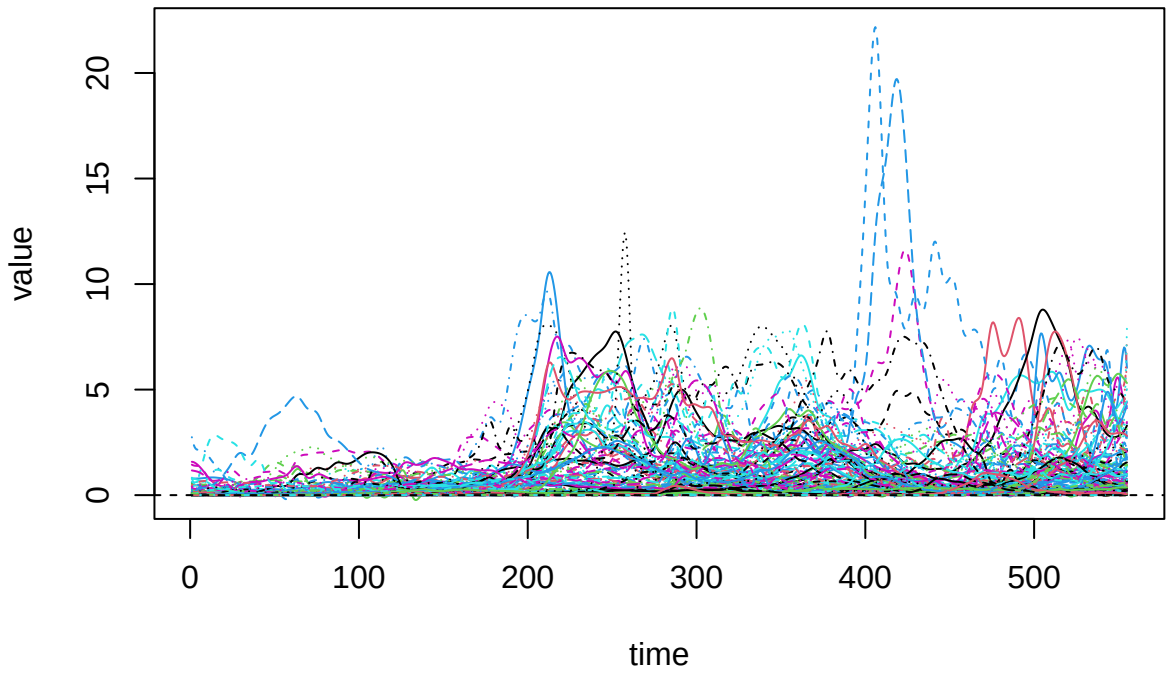
4. Consider the COVID data set analysed in Chapter 7 of the course slides. Consider Italy, Haiti, and the USA (country 79, 69, and 164). Produce residual plots, i.e., plots with the time points on the x-axis and the residuals (difference between data and fit) on the y-axis for these countries for

(a) the fit by a B-spline basis with p = 100, Comment on potential model assumption violations from these plots.

```
covid21 <- read.table("C:/Users/Utente/OneDrive/Desktop/bigData/datasets/covid2021.dat", quote="\"", co
#covid21[,1]
covid21v<-as.matrix(covid21[,5:559])
i=79 #italy
i=69 #haiti
i=164 #usa

basis<-create.bspline.basis(c(1,555), nbasis=100)
fd.cov<-Data2fd(1:555,y=t(as.matrix(covid21v)),basisobj=basis)
plot(basis)
```

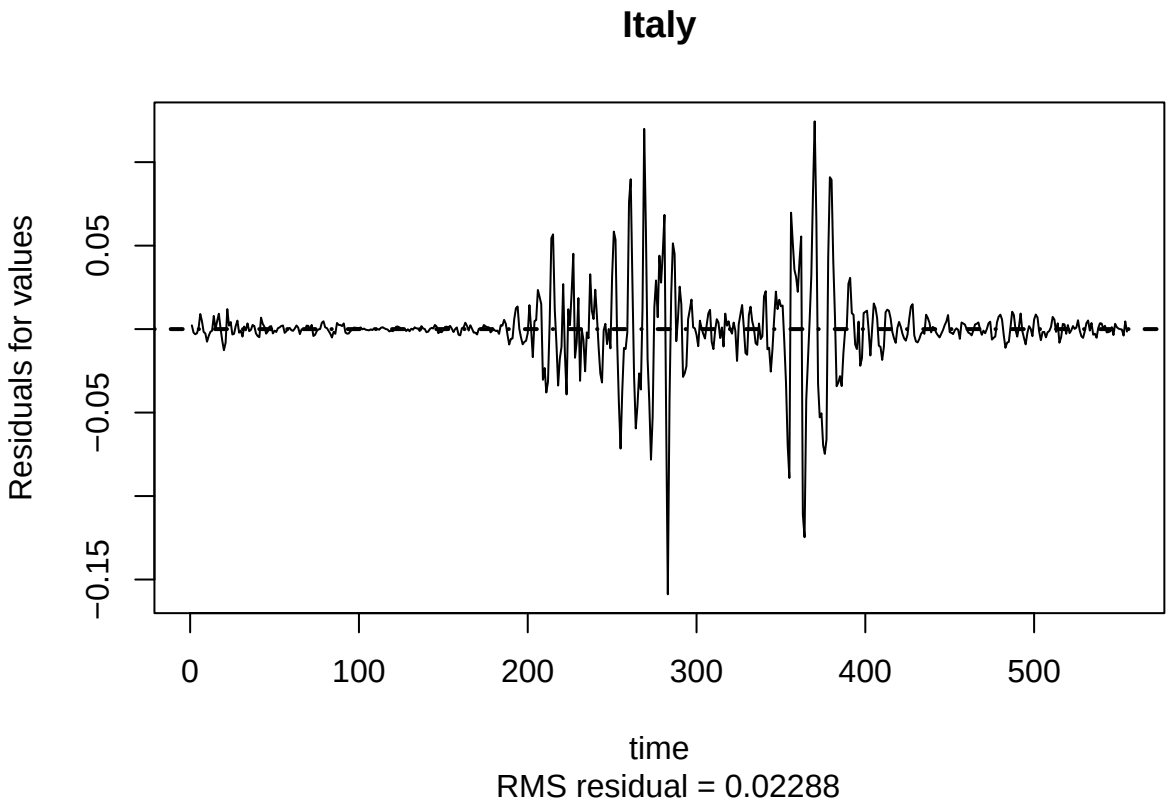


```
plot(fd.cov)
```



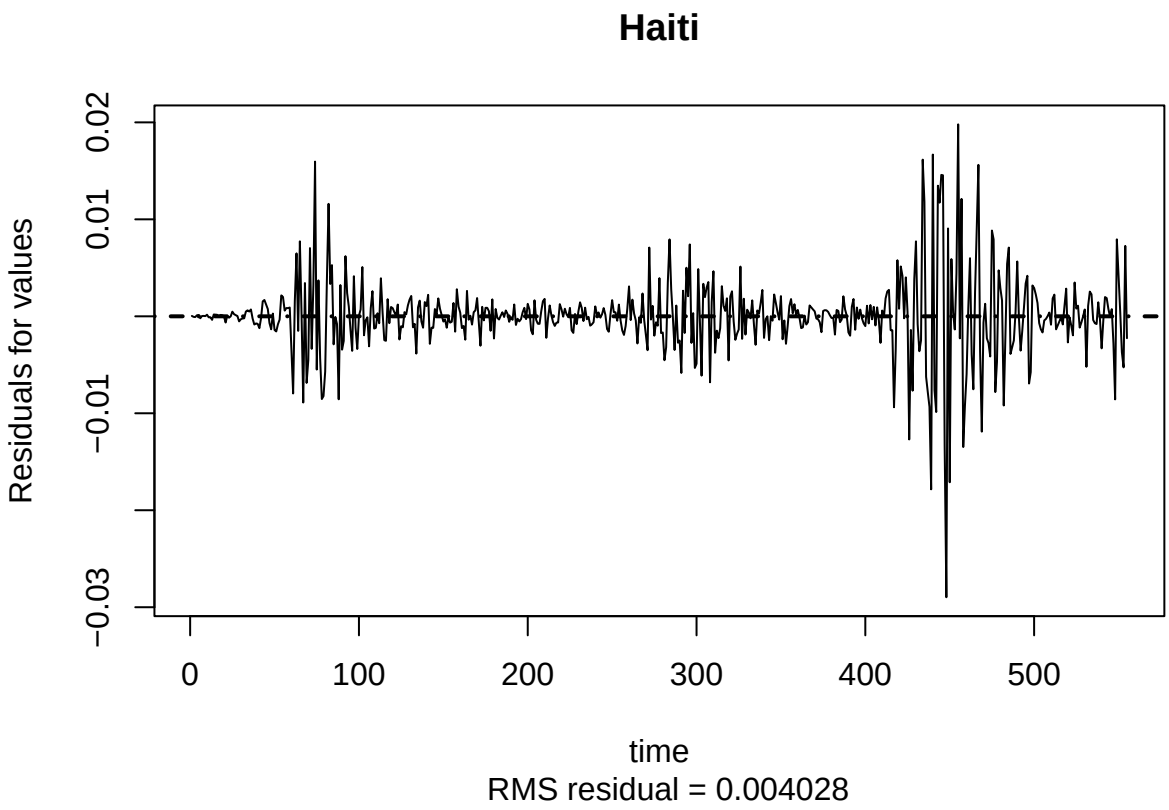

```
## [1] "done"
```

```
?plotfit.fd
plotfit.fd(t(covid21v),1:555,fd.cov,index=79,cex.pch=0.5, residual = TRUE)
```



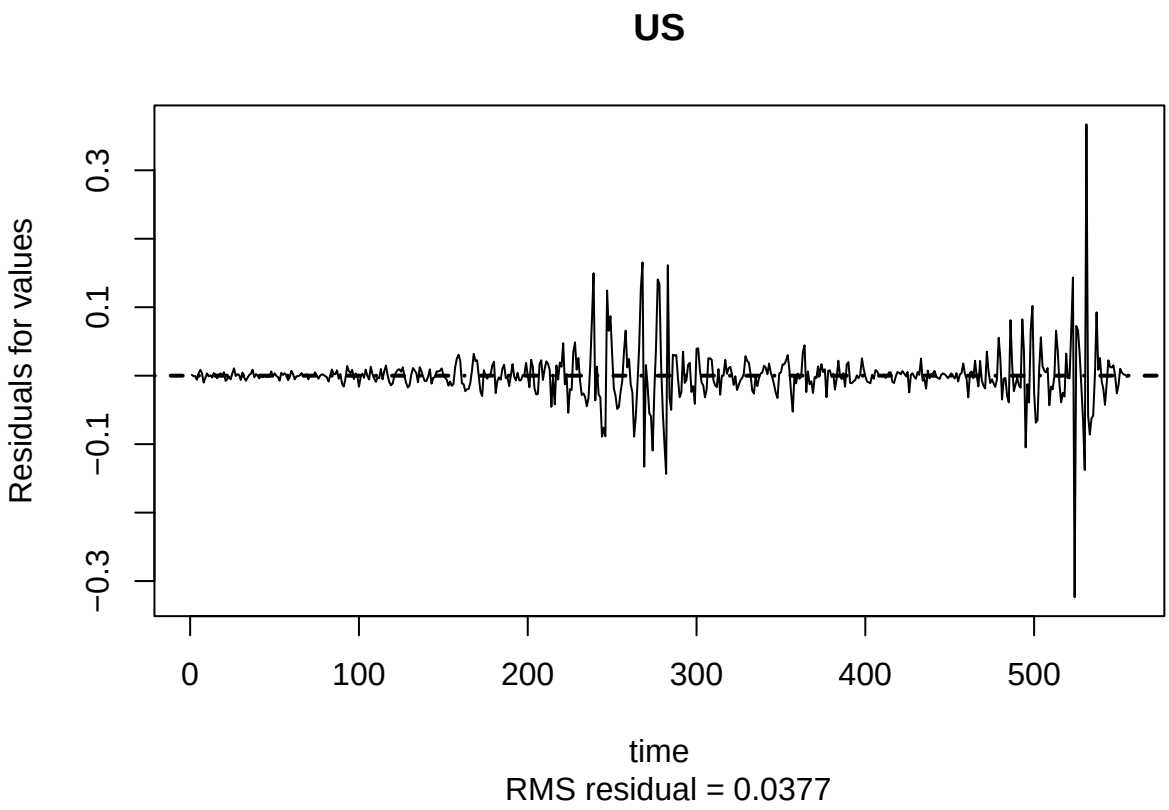
The residuals swings are not well developed around the 0. In correspondence of the periods day200-day300 and day320-day450 we can see a worst fit of the model and the so prediction of data is not good. This is shown by the high variability of the residuals on these days. Linearity assumption seems to be violated.

```
plotfit.fd(t(covid21v),1:555,fd.cov,index=69,cex.pch=0.5, residual = TRUE)
```



Haiti has the lowest RMS residual and so it provides a better fit of the data since it is a measure of how accurately the model predicts the response.

```
plotfit.fd(t(covid21v),1:555,fd.cov,index=164,cex.pch=0.5, residual= TRUE)
```

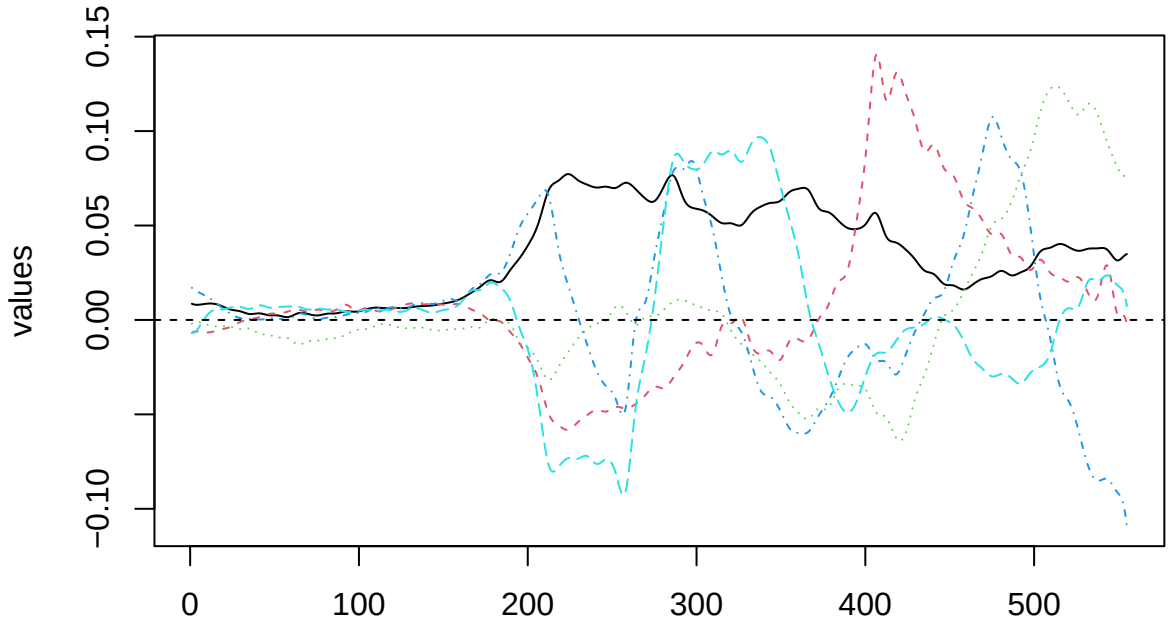


It has the worst RMS residual value so this model fit worst for the data related to USA. But it is lower than 1 so it is very good. the linearity assumption in not violated because the swings have the same height.

The normality assumption is violated in all plots, especially for Haiti. The IID assumption is violated because the variance of the residuals for all countries changes.

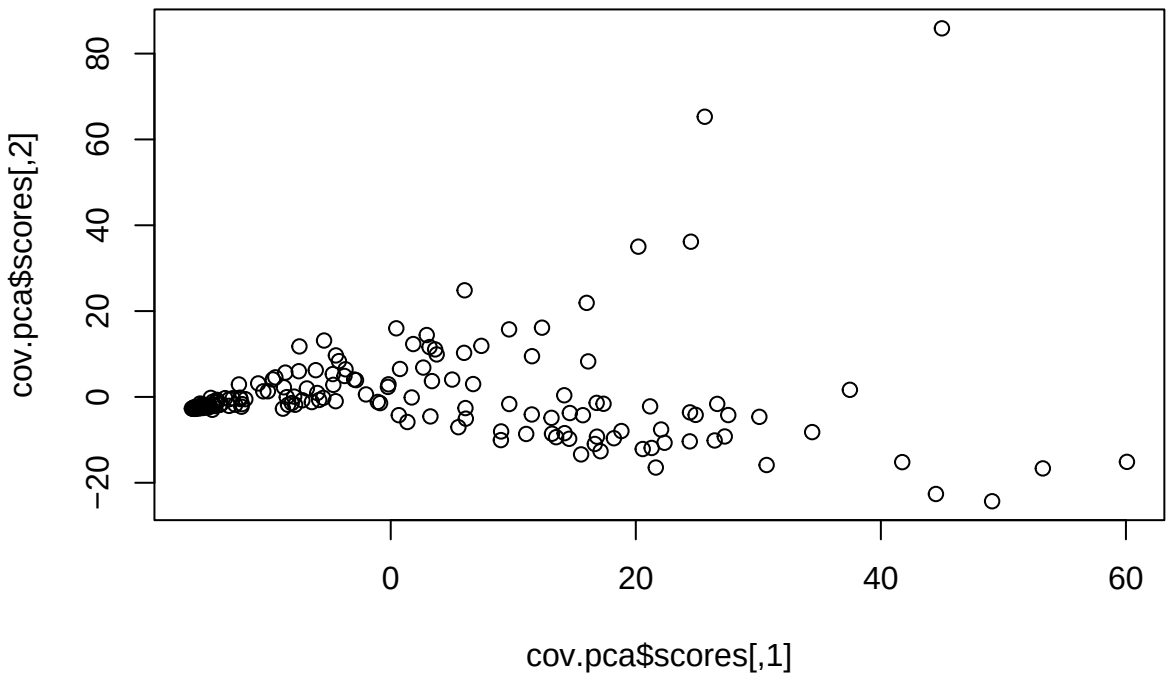
(b) the fit by a 5-dimensional principal components basis.

```
cov.pca <- pca.fd(fd.cov, nharm = 5 )
plot(cov.pca$harmonics)
```

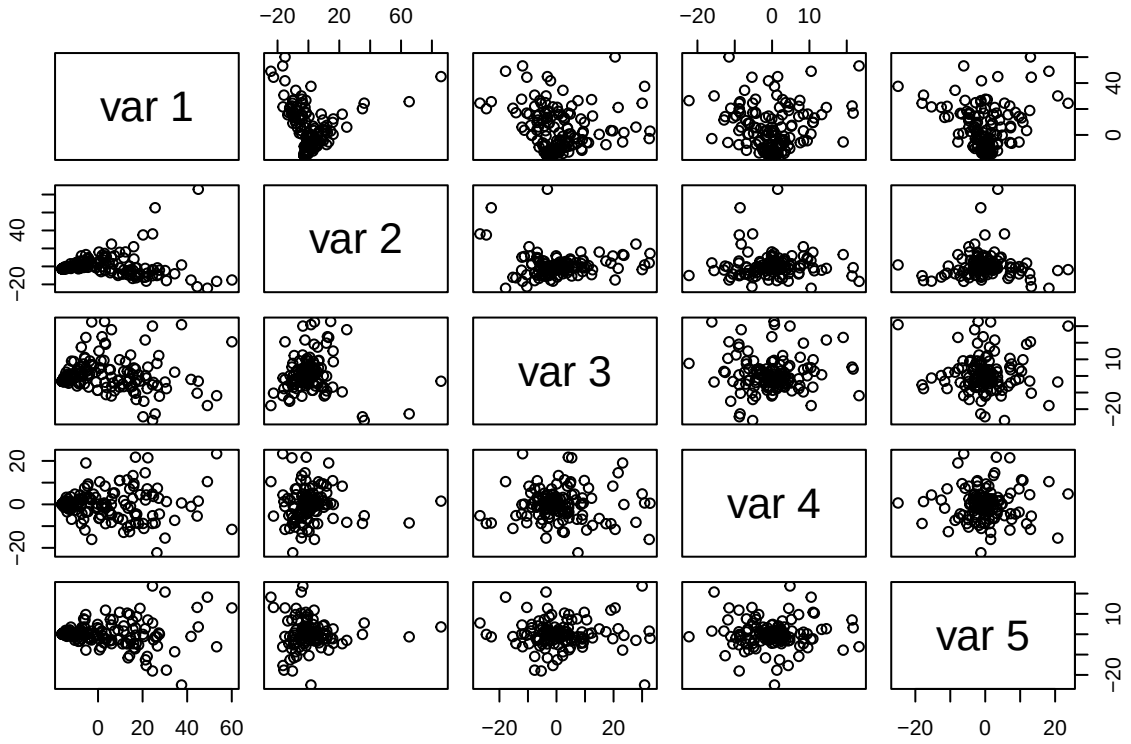



```
## [1] "done"
```

```
plot(cov.pca$scores)
```



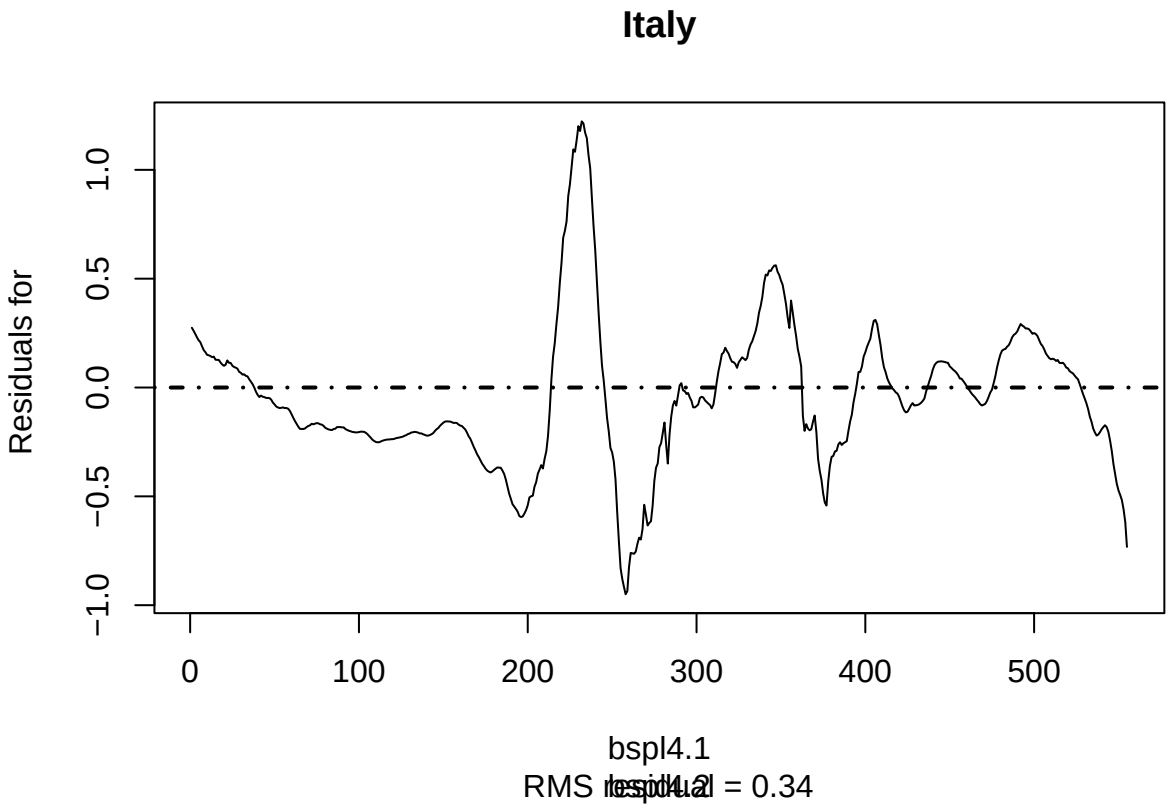

```
pairs(cov.pca$scores)
```




```
cumsum(cov.pca$varprop)
```

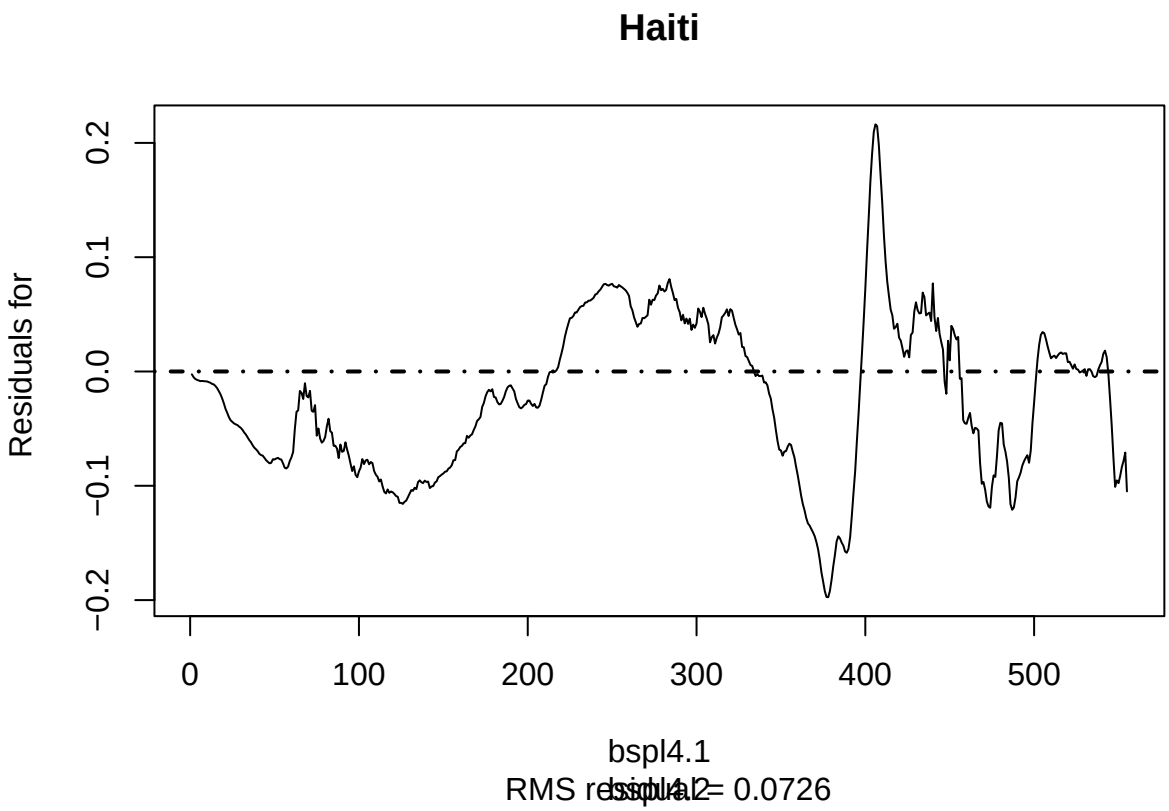
```
## [1] 0.4087192 0.5942757 0.7114464 0.7672579 0.8133276
```

```
mcovid<-mean.fd(fd.cov)

cov.approx<-cov.pca$harmonics
i <- 1
pcacoefi <- cov.pca$harmonics$coefs %*% cov.pca$scores[i,]+mcovid$coefs
cov.approx$coefs <- pcacoefi

for (i in 2:179){
pcacoefi <- cov.pca$harmonics$coefs %*% cov.pca$scores[i,]+mcovid$coefs
cov.approx$coefs <- cbind(cov.approx$coefs, pcacoefi)
}
dimnames(cov.approx$coefs)[[2]] <- covid21[,1]

plotfit.fd(t(covid21v),1:555,cov.approx,index=79,cex.pch=0.5, residual = TRUE)
```



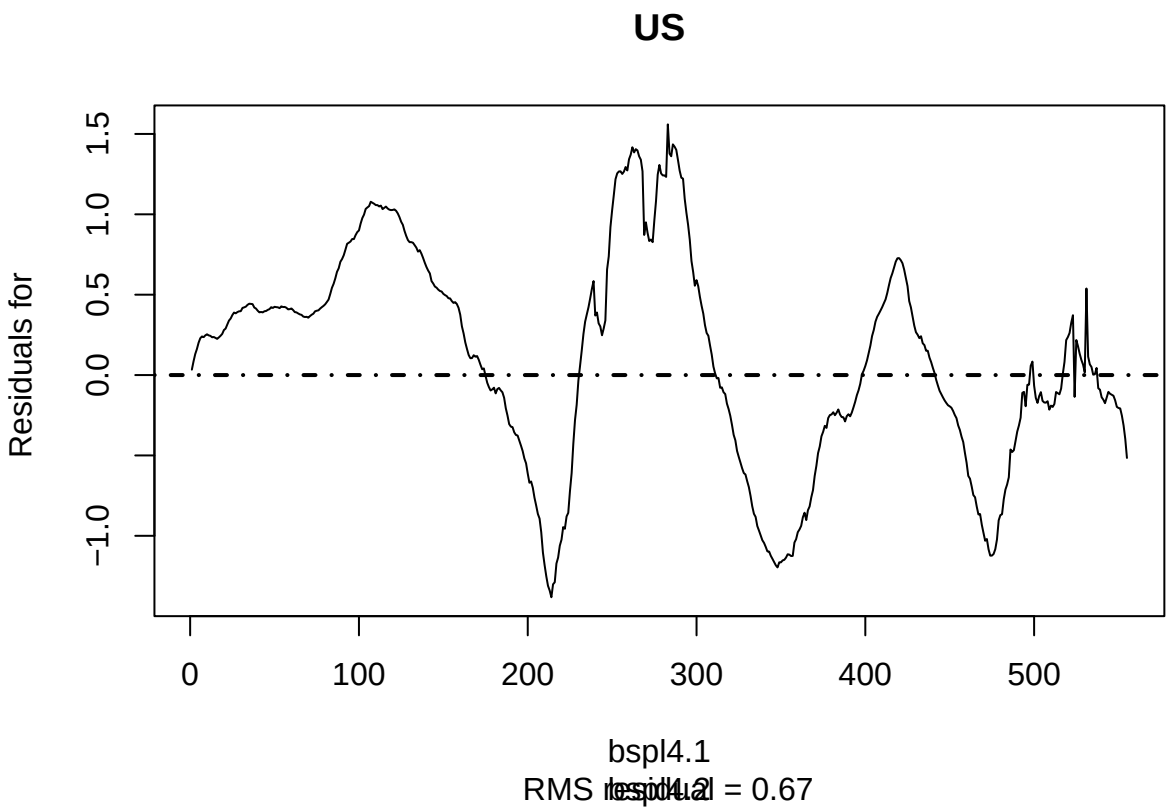
Residuals are eteroschedastic and the plot is not perfect developed around a mean perfect equal to 0.

```
plotfit.fd(t(covid21v),1:555,cov.approx,index=69,cex.pch=0.5, residual=TRUE)
```



The RMS is the lowest also in this case.

```
plotfit.fd(t(covid21v),1:555,cov.approx,index=164,cex.pch=0.5, residual = TRUE)
```



The RMS is the highest and residuals have high variability.

The normality and IID assumptions are all violated for all three countries. Swings are to large and far from 0.

5.Representing all countries in the COVID data set by the first functional principal component scores only, run a one-way analysis of variance to test whether there is evidence that the scores from different continents have different means. Also visualise the scores so that the continents can easily be compared, and interpret plot and result (try to figure out what larger or smaller/positive or negative scores on the first principal component actually mean).

```
cov.pca$scores[,1]
```

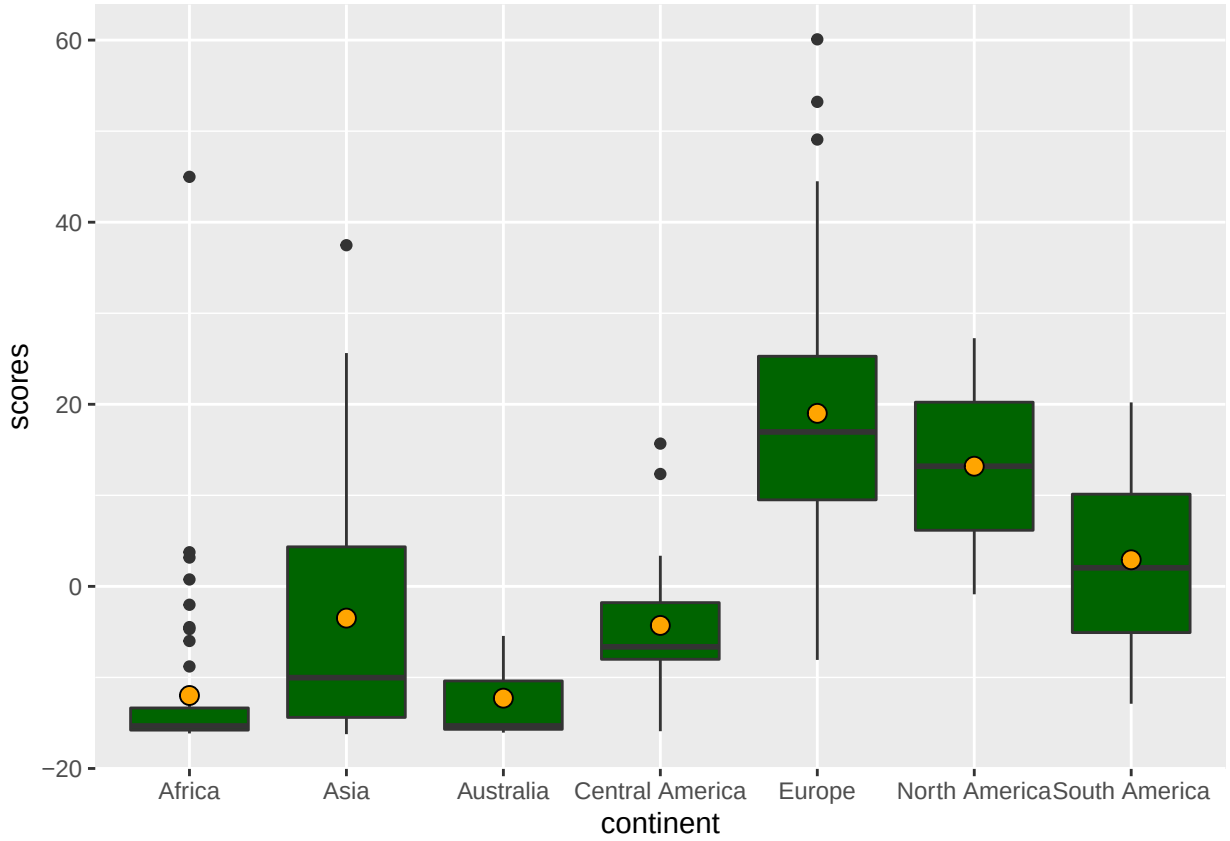
```
##   [1] -15.5655012   5.5203815 -15.0902982  53.2195619 -15.7280046  -5.8424338
##   [7]  15.9880669  13.5010542  16.6561255   0.6592412  -2.9799716  24.4964123
##  [13] -14.2028809  -6.4374641   1.7122388  22.0646273   1.3595146 -15.7137972
##  [19] -15.3663277  -6.1199942  11.0543193   3.1656644  11.5265442 -11.8678389
##  [25]  13.1610783 -15.9877992 -13.9603932 -15.8437147   3.7499972 -14.6880272
##  [31] -15.2425242 -16.1387728   5.9862964   9.6584264 -14.5608617 -15.6158217
##  [37] -16.0880781  12.3457700 -15.6634190  24.4051745   2.9374714  16.1141456
##  [43]  49.0794354   6.0975887 -13.1929285  -2.8340595  -7.8804122  -8.4856072
##  [49] -15.4995386 -12.1393598 -14.5403726 -15.6906948  30.0826345  -5.9964097
##  [55] -15.3567019  -5.4399423  -7.8047472  24.8864411 -12.6865339 -15.4421789
##  [61]  37.4713148   3.2390517 -15.3344666   5.0140524  -8.7213517 -15.7043911
##  [67] -15.6632861  -4.7264199 -15.9059726  -6.8421139  20.5646051  -8.0777442
##  [73]  -9.4166046 -12.3045680   2.6556611  -3.7718816   9.6731369  24.4217789
##  [79]  14.5616768  -7.2101846 -12.2571949  16.8318029  -3.6914135 -14.8606416
```

```
##  [85] -14.4992528  14.6167435   7.3949760 -10.3994328 -15.4153335  17.3659789
##  [91]  18.8253068 -16.0481395  -2.0198640  21.6321255  34.3938215  30.6769860
##  [97] -15.8605501 -15.3735693   1.8357367  25.6268901 -15.9899787  14.1917266
## [103] -14.3228778 -13.1781936  -8.3825521   9.0067814  13.1176357   6.0288653
## [109]  60.0832480  -8.7970351 -14.9718818  -4.4743768  -8.5999327  26.6438294
## [115] -16.0636727 -15.8319565 -16.1531120 -15.9640601  18.2124479  -4.4933170
## [121]  -4.2286007 -14.7536401  15.6810993 -15.4558823   3.6234437  -0.1835888
## [127] -10.0260739  17.1217924  22.3463714  -0.2262291   8.9907841  -1.0550758
## [133] -14.5325961  -7.4992426   3.3643603  -5.5028616  44.4933825 -12.1788498
## [139] -14.1904937 -15.2202122  26.4344858  44.9834111 -16.0907835 -13.8550302
## [145]  15.5364897  41.7430107 -15.8679256  -4.6868148 -15.9186055  16.8047153
## [151]  -9.6221185 -15.9955480   0.4543119  27.5453738  21.2938651 -15.6028743
## [157] -15.9422791 -10.8135154 -12.3893704 -15.3105605  -7.4521840   0.7573122
## [163]  14.1606821  27.2606674 -15.5451550   6.1276182   6.7248062  21.1668842
## [169]  20.2074372 -15.1730481 -12.8926755 -14.2863636  11.5112565 -16.1510039
## [175] -13.5178321 -14.2301447 -15.3299455  -0.8683476 -16.2317445
```

```
#ncontinent <- as.numeric(as.factor(covid21$continent))
#plot(cov.pca$scores[,1],col=ncontinent,pch=clusym[ncontinent], ylim=range(-25:60))

cov.anova <- cbind(covid21, cov.pca$scores[,1])
anova <- aov(cov.pca$scores[,1]~continent, data = cov.anova)
summary.aov(anova)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## continent     6  25062    4177   27.62 <2e-16 ***
## Residuals   172  26008     151
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
covid21["scores"]<-cov.pca$scores[,1]

library(ggplot2)
ggplot(covid21,aes(x=continent,y=scores))+ geom_boxplot(fill='darkgreen')+stat_summary(
  geom = "point",fun = "mean",col = "black",size = 3,shape = 21,fill="orange")
```



The boxplot allows to individuate potential outliers. In Africa we can see more outliers than in other continent while for Australia, North America and south america there are not outliers. Africa has the lowest variability and the lowest median like Australia. Also Australia and Central America has a quite low variance of scores. Asia has the largest variance within scores and the majority of them are negative. For these continents data are asymmetric because the medians are neare the bottom of the boxplot. Europe has the highest median and variance within scores similar to the one for North America and South America. Africa, Australia and Central America show only negative scores. South America ha slightly more than half the values of the scores that are positive.