

# Functional data analysis of eye pupil dilation

*Paolo Girardi, Sara Scrimin and others*

06/12/2019

## Abstract

In general, methods for correlating pupillary response to the cognitive activity of a subject undergoing an evaluation of cognitive activity are based on average values or analysis of peak activity.

However, in this type of analysis, we lose the temporal form of the data which is a valuable feature because of temporal scale (high temporal scale 250 Hz) or specific trend pattern. Eye tracking data contains several features:

- Eye point of gaze;
- Fixation (and relative Area Of Interest -AOI-) and Saccade time periods.
- **Eye pupil dilation**

Differently by the classical approach based on differences on “averaged” values over a certain period, we propose a functional data clustering approach that takes into account the entire behaviour of a eye dilation time series.

## 1 - Functional data clustering... a gentle introduction

The statistical problem is to cluster subjects with common temporal behaviour regarding their eye dilation. Let  $\mathbf{Y}_i = (Y_i(t_{i,1}), \dots, Y_i(t_{i,m_i}))'$  a time series of  $m_i$  values collected for the subject  $s_i$ ,  $i = 1, \dots, n$ . The value  $m_i$  corresponds to the number of observations included in the  $i$ -th time series. We suppose to observe  $n$  time series and we want to cluster them in  $C$  clusters.

### 1.1 - Model-based Functional data clustering

In a mixture-model based approach to clustering the cluster membership of the  $i$ -th time series is represented by a latent random vector  $\mathbf{Z}_i = (Z_{i,1}, \dots, Z_{i,C})'$ , where  $Z_{i,c} = 1$  if the time series belongs to the cluster  $c$ ,  $Z_{i,c} = 0$  otherwise. Then a model for the complete data  $(\mathbf{Y}_i, \mathbf{Z}_i)$ ,  $i = 1, \dots, n$  is specified in a hierarchical way. Suppose that the  $i$ -th time series belongs to the cluster  $c$ , i.e.  $Z_{i,c} = 1$  and  $Z_{i,c'} = 0$  for  $c' \neq c$ .

Given the cluster membership  $\mathbf{Z}_i$ , we suppose that  $\mathbf{Y}_i$  is a discrete and noised measurement of a smooth time-varying curve  $f_c(t)$ , namely

$$Y_i(t_{i,j}) = f_c(t_{i,j}) + \varepsilon_i(t_{i,j}), \quad j = 1, \dots, m_i.$$

The smooth function  $f_c(t)$  is described by a linear combination of  $K$  B-spline basis functions  $B_k(t)$ ,  $k = 1, \dots, K$ , evaluated at  $K - 1$  equally spaced in knots, namely

$$f_c(t) = \sum_{k=1}^K \psi_{c,k} B_k(t) = \boldsymbol{\gamma}^\top \mathbf{B}(t)$$

where  $\boldsymbol{\psi}_c = (\psi_{c,1}, \dots, \psi_{c,K})'$  is a vector of unknown parameters.

In view of our application, it does not seem too restrictive to assume that  $\varepsilon_i(t_{i,j})$  are temporally independent Gaussian random errors with zero mean and cluster specific variance  $\sigma_c^2$ . It turns out that time series  $\mathbf{Y}_1, \dots, \mathbf{Y}_n$  are conditionally independent on  $\mathbf{Z}_1, \dots, \mathbf{Z}_n$  and

$$Y_i(t_{i,j}) | \mathbf{Z}_i \sim \mathcal{N} \left( \sum_{k=1}^K \psi_{c,k} B_k(t_{i,j}), \sigma_c^2 \right), \quad j = 1 \dots, m_i. \quad (1)$$

The clustering involves the membership  $\mathbf{Z}$  by means of iterative algorithm in order to get the more classification that maximized a given likelihood.

## 1.2 - Two steps Functional data clustering

A second different approach is called “2 steps - clustering”. The two steps are:

- estimation (functional data and dimensional reduction);
- classification.

In this attempt, we performed an unsupervised classification based on measures of the distance between the 2 curves.

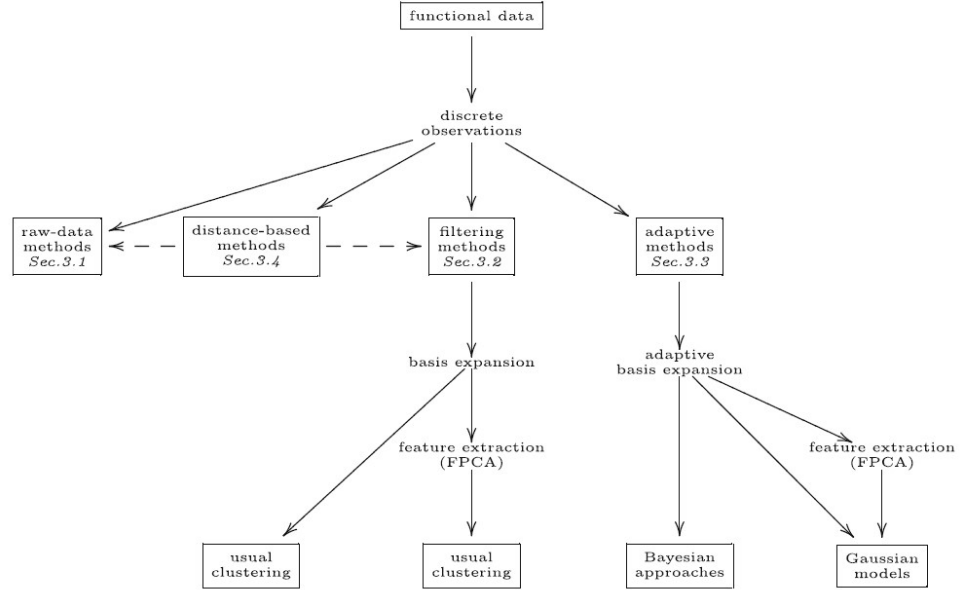
We calculated a distance matrix, based on the L2 norm between two estimated curves  $i$  and  $j$   $f_i(t) = \sum_{k=1}^K \hat{\psi}_{i,k} B_k(t)$  and  $f_j(t) = \sum_{k=1}^K \hat{\psi}_{j,k} B_k(t)$  ( $\hat{\psi}$  are the estimated coefficients) is calculated as following:

$$d_{i,j} = \sqrt{\left( \sum_{k=1}^K \hat{\psi}_{i,k} B_k(t) - \sum_{k=1}^K \hat{\psi}_{j,k} B_k(t) \right)^2} \approx \sqrt{\sum_{k=1}^K (\hat{\psi}_{i,k} - \hat{\psi}_{j,k})^2} \quad \forall i \neq j, \quad (2)$$

where  $d_{i,j}$  is an element of distance matrix  $D$  based on the euclidean distance.

We applied to the distance matrix an algorithm in order to cluster together similar curves.

A summary of the statistical methods proposed for functional data clustering is reported in a review of (Jacques and Preda 2014); in the following figure the presented summary.

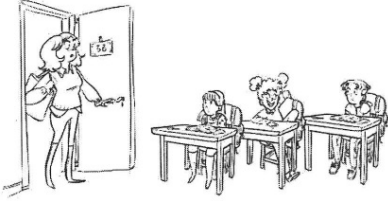


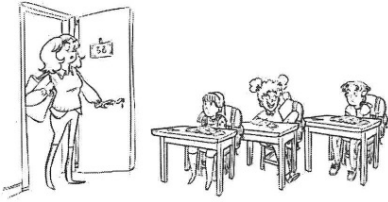




**Fig. 2** Segmentation of the different clustering methods for functional data

## 2 - Eye tracking dilation dataset

We consider a dataset formed by pupil eye dilation records of 25 female childs involved in two trials consisting in the visualization of 3 images for a total of 18 seconds differing between them only for the ending scene:

- Trial 1: Ambiguous situation - Teacher with adverse reaction - **Child with adverse reaction;**
- Trial 2: Ambiguous situation - Teacher with adverse reaction - **Child with sadden reaction.**

Image 1: 5 secs	Duration of the visualizzation Image 1: 3 secs Trial 1	Image 1: 10 secs
		
		

```
rm(list=ls())
setwd("C:/Users/utente/Desktop/Eye_tracker/")
### data are preprocessed (time series with 0 mean and unit standard deviation)
db_tf1<-read.csv("trial1_f.csv")
db_tf2<-read.csv("trial2_f.csv")
#imported variables
names(db_tf1)

## [1] "X"      "sex"    "id"     "test"   "img"    "time"   "id_f"
## [8] "id_s"   "type"   "durata" "p_sx"   "p_dx"   "v_sx"   "v_dx"

# p_sx and d_sx are Right and Left eye dilations
# time is the temporal scale
# id is the subject ID
table(db_tf1$id)

##
## F1301 F1318 F1322 F1326 F1340 F213 F215 F216 F243 F245 F248 F249
## 1083 1084 1082 1082 1082 1083 1084 1080 1083 1083 1083 1083
## F252 F257 F260 F266 F273 F282 F286 F288 F290 F292 F295 F296
## 1084 1085 1084 1082 1084 1082 1084 1083 1082 1082 1085 1084
## F299
## 1082
```

```

###more than 1000 time points for each time series (18/1082= 0.017s time scale)
### but different time points
length(table(db_tf1$time))

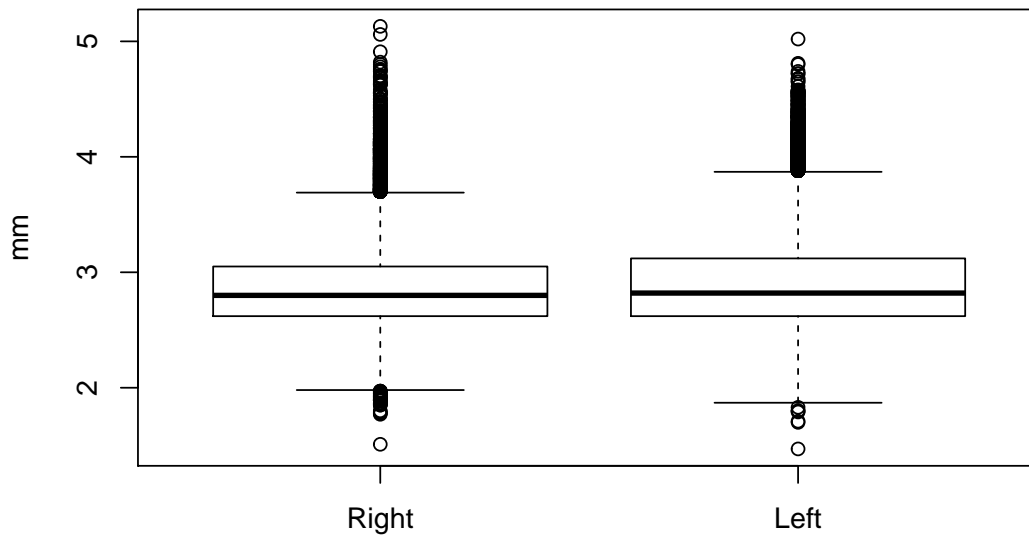
```

```
## [1] 6948
```

```

## Marginal distribution of Right and Left eye dilations
boxplot(db_tf1$p_dx,db_tf1$p_sx,names=c("Right","Left"),ylab="mm")

```



```

## Right and Left eye dilations are highly correlated
cor.test(db_tf1$p_dx,db_tf1$p_sx)

```

```

##
## Pearson's product-moment correlation
##
## data: db_tf1$p_dx and db_tf1$p_sx
## t = 385.96, df = 24860, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.9239338 0.9274899
## sample estimates:
## cor
## 0.9257323

```

```

plot(db_tf1$p_dx,db_tf1$p_sx,xlab="Right pupil",ylab="Left pupil")
abline(a=0,b=1,col="red",lty=2)
### create the average dilation between two eye
db_tf1$p<-apply(cbind(db_tf1$p_dx,db_tf1$p_sx),1,mean,na.rm=T)
db_tf2$p<-apply(cbind(db_tf2$p_dx,db_tf2$p_sx),1,mean,na.rm=T)

```

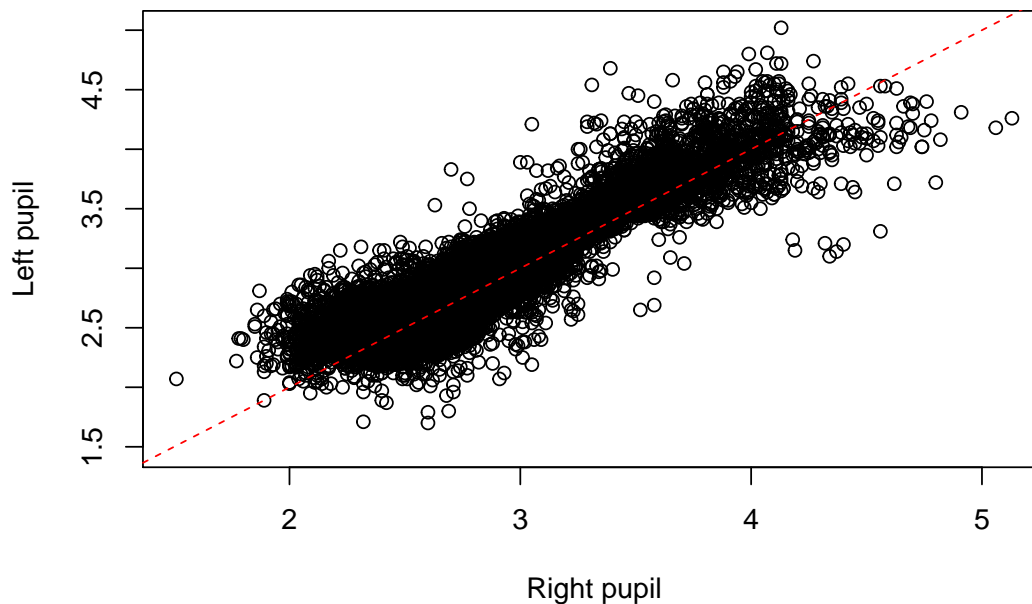
```
#some R function to manage plots and data
source("C:/Users/utente/Desktop/Eye_tracker/functions.R")
```

```
## Registered S3 method overwritten by 'xts':
##   method      from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method      from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method      from
##   fitted.fracdiff fracdiff
##   residuals.fracdiff fracdiff

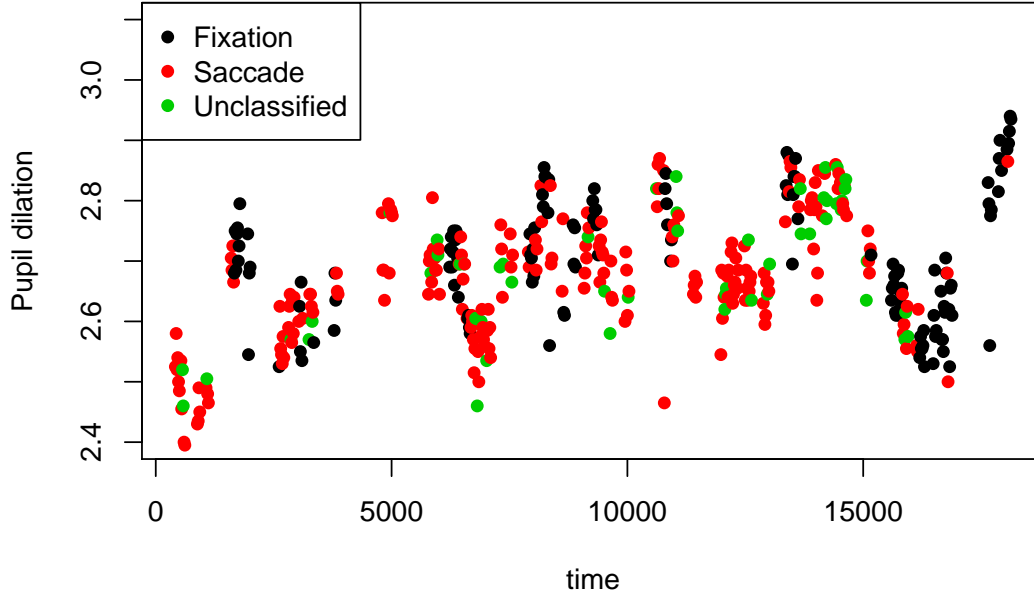
## Loading required package: splines
## Loading required package: Matrix
##
## Attaching package: 'fda'
## The following object is masked from 'package:graphics':
##
##   matplot
```



## Moving Average Filter

An example of a time series on pupil dilatation is reported in the following figure.

```
plot(db_tf1$time[db_tf1$id=="F1301" & db_tf1$test=="t1" & db_tf1$type=="Fixation"],db_tf1$p[db_tf1$id=="F1301"
legend("topleft",c("Fixation","Saccade","Unclassified"),col=1:3,pch=16)
```



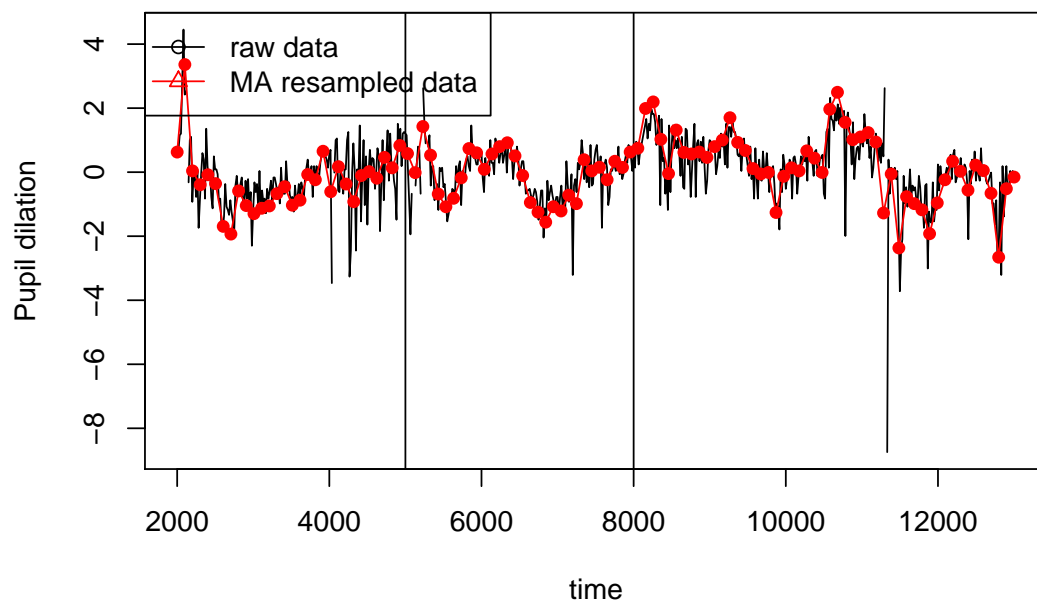
There is a certain grade of variability. The presence of saccade and unclassified moments lead to a very noised time series. To overcome this and to get standardized time series, we need to resample the time series. We apply a Moving Average Filter in order to get the same time points for each time series as follow:

$$Y_{filtered_i}(t_{i,j}) = \frac{\sum_{i \in W_z} Y_i(t_{i,j})}{\#W_z}$$

where  $W_z$  is a time windows and  $\bigcup_{z=1}^{\infty} W_i$  is the entire time window (0-18 seconds). In addition, we scale each time series (mean=0 and var=1) because we are interested to compare only their temporal behaviour.

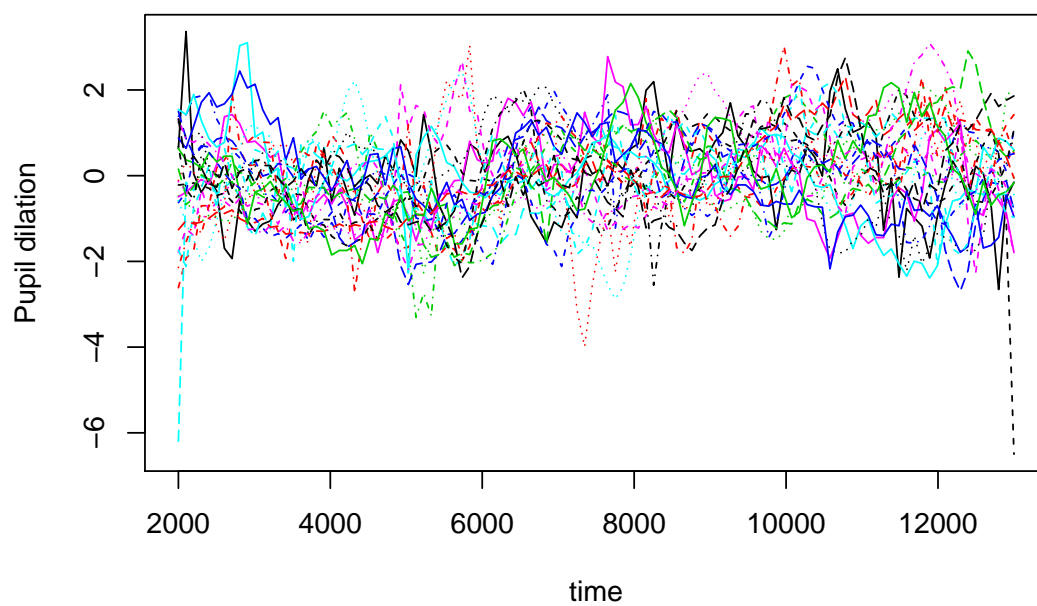
```
#set dropping first 2 seconds and endings 5 seconds
min<-2000
max<-13000
#sampling each time series at 100 ms
step<- round((max-min)/100,0)
# (data vector, subject vector, time vector, time scale, window=(min,max),scale=T)
data_sr1<-sampling_regular(db_tf1$p,db_tf1$id,db_tf1$time,step,window=c(min,max),scale=TRUE)
data_sr2<-sampling_regular(db_tf2$p,db_tf2$id,db_tf2$time,step,window=c(min,max),scale=TRUE)
# in output a list formed by a matrix of observation and a regularized time vector

# MA resampled data for the first child
plot(db_tf1$time[db_tf1$id=="F1301" & db_tf1$test=="t1" & (db_tf1$time> min & db_tf1$time< max)],scale(db_tf1$
abline(v=c(5000,8000))
points(data_sr1$t,data_sr1$x_mat[1,],pch=16,col="red")
points(data_sr1$t,data_sr1$x_mat[1,],col="red",type="l")
legend("topleft",c("raw data","MA resampled data"),col=c(1,2),pch=c(1,2),lty=c(1,1))
```

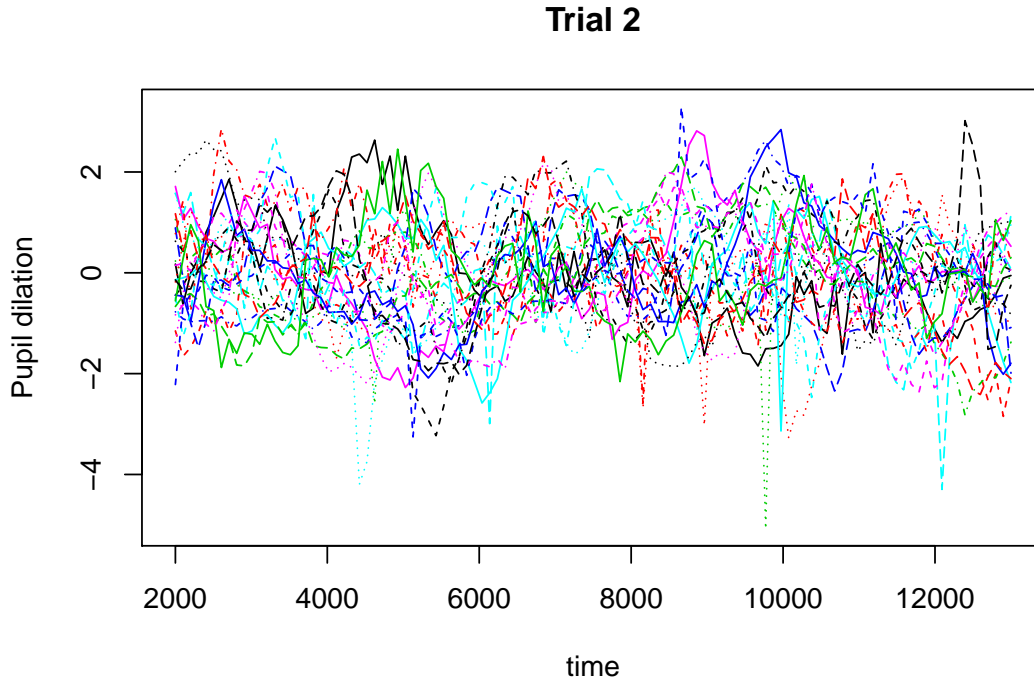


```
#our time series
matplotlib(data_sr1$t,t(data_sr1$x_mat),type="l",main="Trial 1",ylab="Pupil dilation",xlab="time")
```

### Trial 1



```
matplot(data_sr2$t,t(data_sr2$x_mat),type="l",main="Trial 2",ylab="Pupil dilation",xlab="time")
```



Lines are overlapping. It is impossible to observed different time patterns. It seems hard to discover similar curves. Next step: Smoothing!

We defined  $k=25$  equally distributed knots for the basis of splines. A number 25 knots seems sufficient (the strategy is to have a great number of coefficients penalizing them with L2norm of applied to the second derivatives).

$$\widehat{\psi_{c,k}} = \underset{\psi}{\operatorname{argmin}} \sum_{t=1}^T \left( Y_i t - \sum_{k=1}^K \psi_{c,k} B_k(t) \right)^2 + \lambda \int \frac{\partial^2 \psi_{c,k} B_k(t)}{\partial t^2} \quad (3)$$

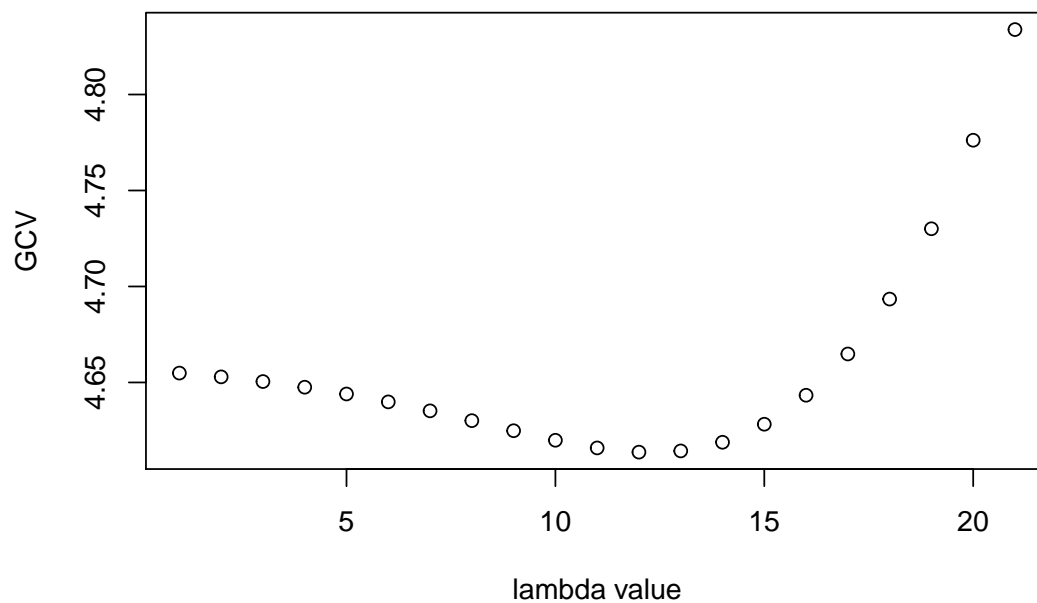
where  $\lambda$  is the smoothing parameter that penalized roughness curves. The number of smoothing  $\lambda$  is chosen by means of the GCV (Generalized Cross validation) criterion.

```
# we adopt a smoothing strategy, high number of knots (15)
# penalizing with L2 norm on second derivatives of coefficients
# GCV criterion to select the best lambda

l<-exp(seq(10,15,0.25))

gcv1<-NULL
for(i in 1:length(l)){
  gcv1<-c(gcv1,fun_register(data_sr1,lambda=l[i], knots=25,plot=FALSE,register=FALSE)$gcv)
}
graphics::plot(gcv1,ylab="GCV",xlab="lambda value")
```

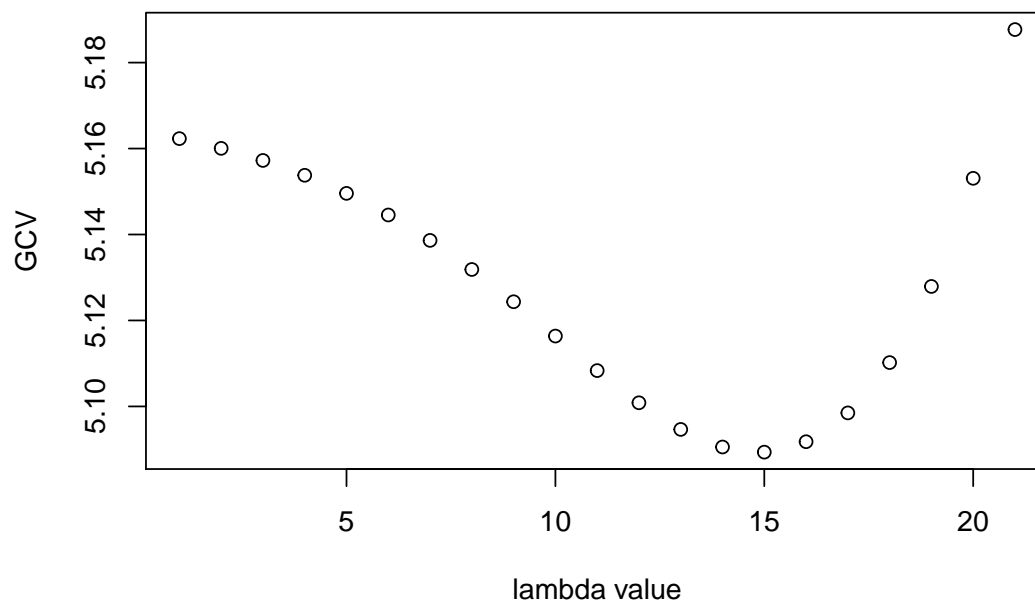




```
which.min(gcv1)
```

```
## [1] 12
```

```
gcv2<-NULL
for(i in 1:length(l)){
  gcv2<-c(gcv2,fun_register(data_sr2,lambda=l[i], knots=25,plot=FALSE,register=FALSE)$gcv)
}
graphics::plot(gcv2,ylab="GCV",xlab="lambda value")
```



```
which.min(gcv2)
```

```
## [1] 15
```

```
# coefficient estimation
```

```
data_reg1<-fun_register(data_sr1,lambda=l[which.min(gcv1)], knots=25,plot=FALSE,register=FALSE)$data_reg
```

```
data_reg2<-fun_register(data_sr2,lambda=l[which.min(gcv2)], knots=25,plot=FALSE,register=FALSE)$data_reg
```

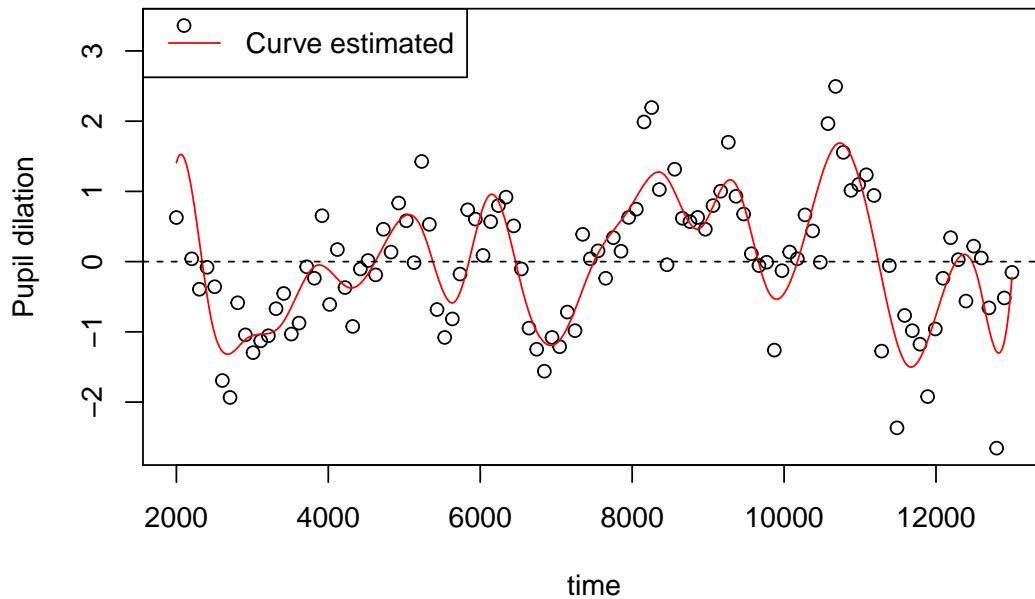
```
#example of estimated curve
```

```
plot(data_sr1$t,data_sr1$x_mat[1,],ylab="Pupil dilation",xlab="time")
```

```
plot(data_reg1[1],add=T,col=2)
```

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

```
legend("topleft",c("Curve estimated"),col=c(2),lty=1)
```



## Number of clusters

The choice of the number of clusters is a crucial point: we select the number of cluster by means of the Gap Statistics (Tibshirani, Walther, and Hastie 2001), using PAM algorithm applying the euclidean distance (L2-norm) to the B-splines coefficients (the L2 norm applied to the curves or to the Bspline coefficient is the same).

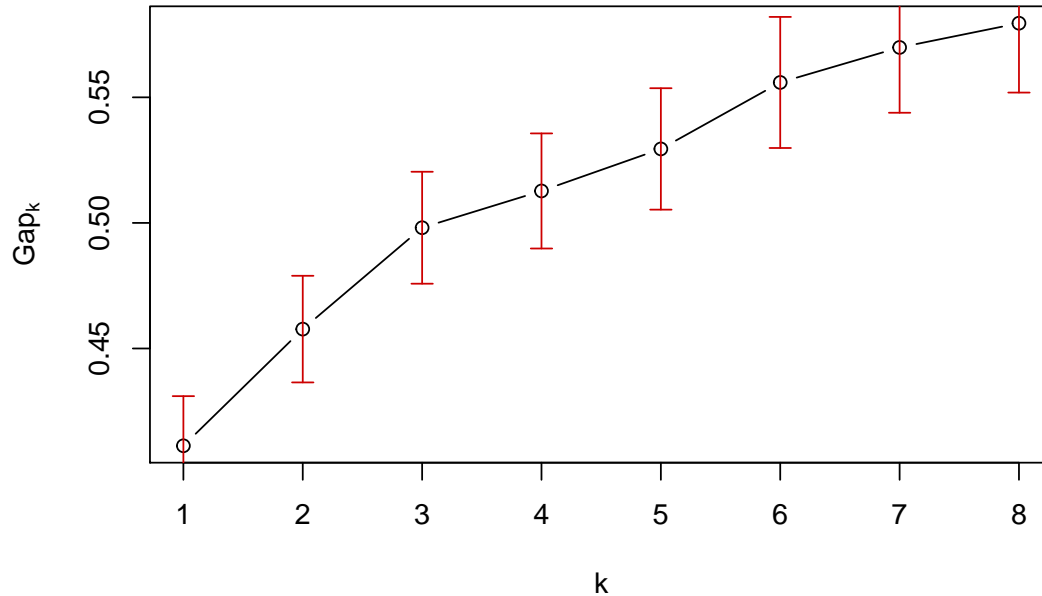
```
library(cluster)
#euclidean distance matrix
dist_mat1<-dist(t(data_reg1$coefs))
dist_mat2<-dist(t(data_reg2$coefs))

pam1 <- function(x,k) list(cluster = pam(x,k, cluster.only=TRUE,diss=T))
mod_sel<-clusGap(as.matrix(dist_mat1), FUN = pam1, K.max = 8, B =500,spaceH0 = "original")
#useTibs2001SEmax method and SE.factor=1
print(mod_sel, method="Tibs2001SEmax",SE.factor=1)

## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = as.matrix(dist_mat1), FUNcluster = pam1, K.max = 8,      B = 500, spaceH0 = "original")
## B=500 simulated reference sets, k = 1..8; spaceH0="original"
## --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 3
##      logW      E.logW      gap      SE.sim
## [1,] 4.538289 4.949550 0.4112606 0.01975891
## [2,] 4.443564 4.901279 0.4577149 0.02122957
## [3,] 4.356796 4.854878 0.4980820 0.02228950
## [4,] 4.294081 4.806803 0.5127216 0.02291485
## [5,] 4.227533 4.756992 0.5294593 0.02417791
## [6,] 4.151893 4.707820 0.5559265 0.02609529
## [7,] 4.084612 4.654481 0.5698686 0.02601035
## [8,] 4.018902 4.598417 0.5795143 0.02762039
```

```
plot(mod_sel)
```

**clusGap(x = as.matrix(dist\_mat1), FUNcluster = pam1, K.max = 8, B = 500, spaceH0 = "original")**

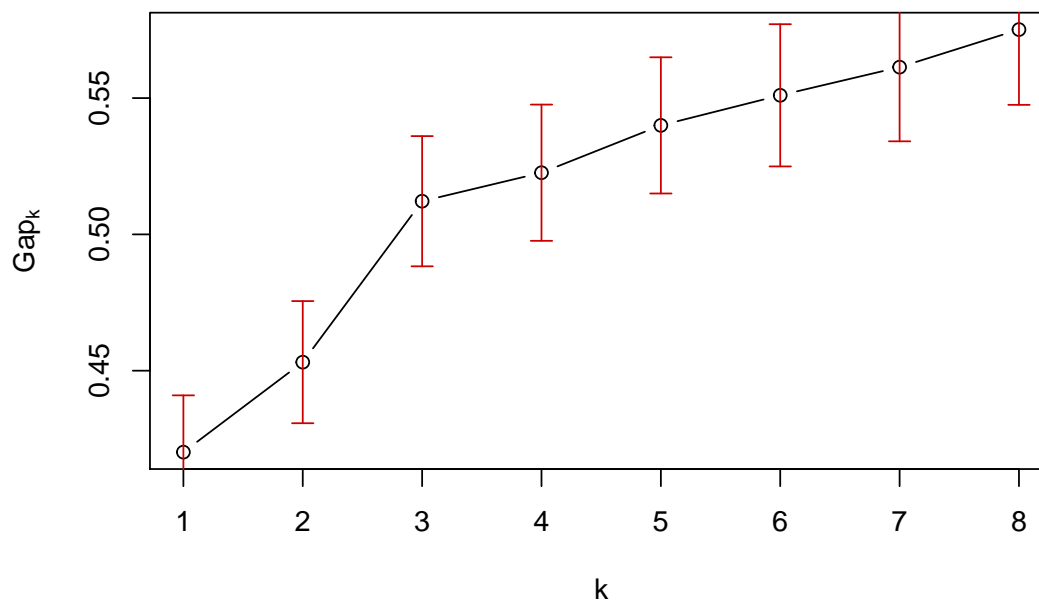


```
mod_sel<-clusGap(as.matrix(dist_mat2), FUN = pam1, K.max = 8, B=500,spaceH0 = "original")
#uso FirstSEmax method
print(mod_sel, method="Tibs2001SEmax",SE.factor=1)
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = as.matrix(dist_mat2), FUNcluster = pam1, K.max = 8,      B = 500, spaceH0 = "original")
## B=500 simulated reference sets, k = 1..8; spaceH0="original"
## --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 3
##      logW  E.logW    gap    SE.sim
## [1,] 4.508519 4.928631 0.4201123 0.02082982
## [2,] 4.427345 4.880466 0.4531208 0.02240202
## [3,] 4.321256 4.833423 0.5121666 0.02388096
## [4,] 4.264298 4.786917 0.5226189 0.02499415
## [5,] 4.198616 4.738601 0.5399846 0.02498348
## [6,] 4.136206 4.687224 0.5510173 0.02609085
## [7,] 4.072363 4.633710 0.5613475 0.02720553
## [8,] 4.002999 4.578126 0.5751276 0.02761226
```

```
plot(mod_sel)
```

```
clusGap(x = as.matrix(dist_mat2), FUNcluster = pam1, K.max
= 8, B = 500, spaceH0 = "original")
```



For both trials, Gap statistic suggests 3 clusters. At the end we cluster time series and make appropriate graphs.

```
#two clusterings
clust1<-pam(as.matrix(dist_mat1),3,diss=T)$clust
clust2<-pam(as.matrix(dist_mat2),3,diss=T)$clust
table(clust1)
```

```
## clust1
## 1 2 3
## 16 6 3
```

```
table(clust2)
```

```
## clust2
## 1 2 3
## 10 8 7
```

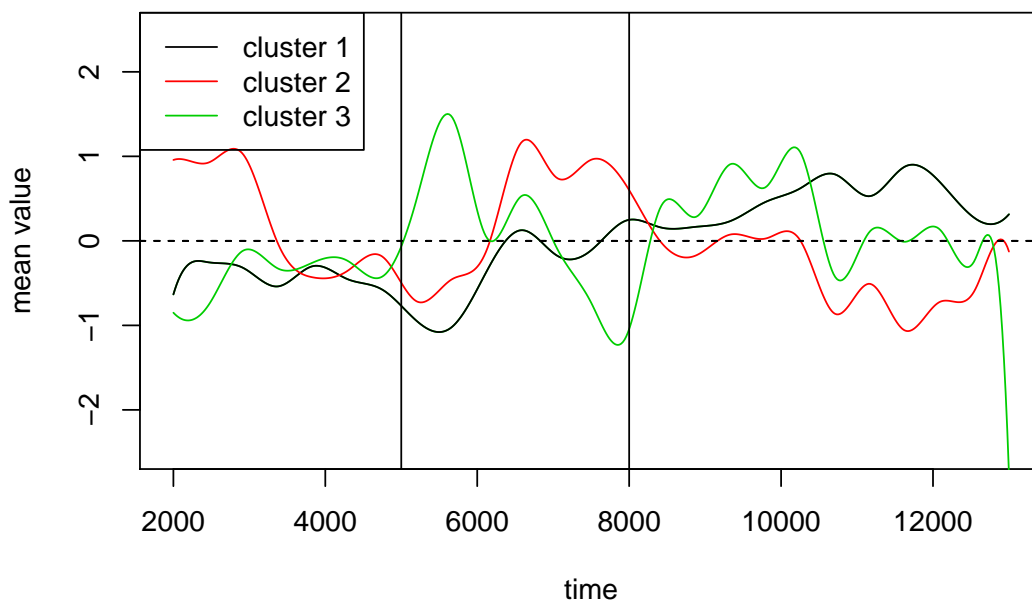
```
table(clust1,clust2)
```

```
##      clust2
## clust1 1 2 3
##      1 7 7 2
##      2 2 1 3
##      3 1 0 2
```

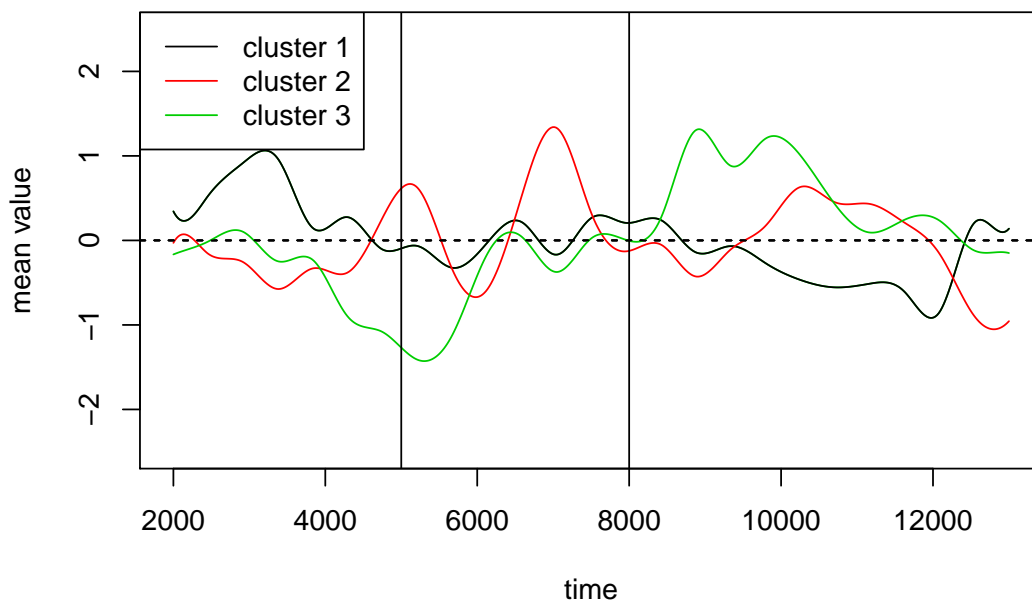
## Results

In the first trials we classified 16 time series in the first group, 6 in the second one and 3 in the last group. In the second trials we classified 10, 8 and 7 curves in the group 1, 2 and 3 respectively. We can comment in function of the different behaviours on temporal profile in each group by trial membership.

```
plot_cluster_curves2(data_reg1,clust1)
legend("topleft",c("cluster 1","cluster 2","cluster 3"),lty=1, col=1:3)
abline(v=c(5000,8000))
```



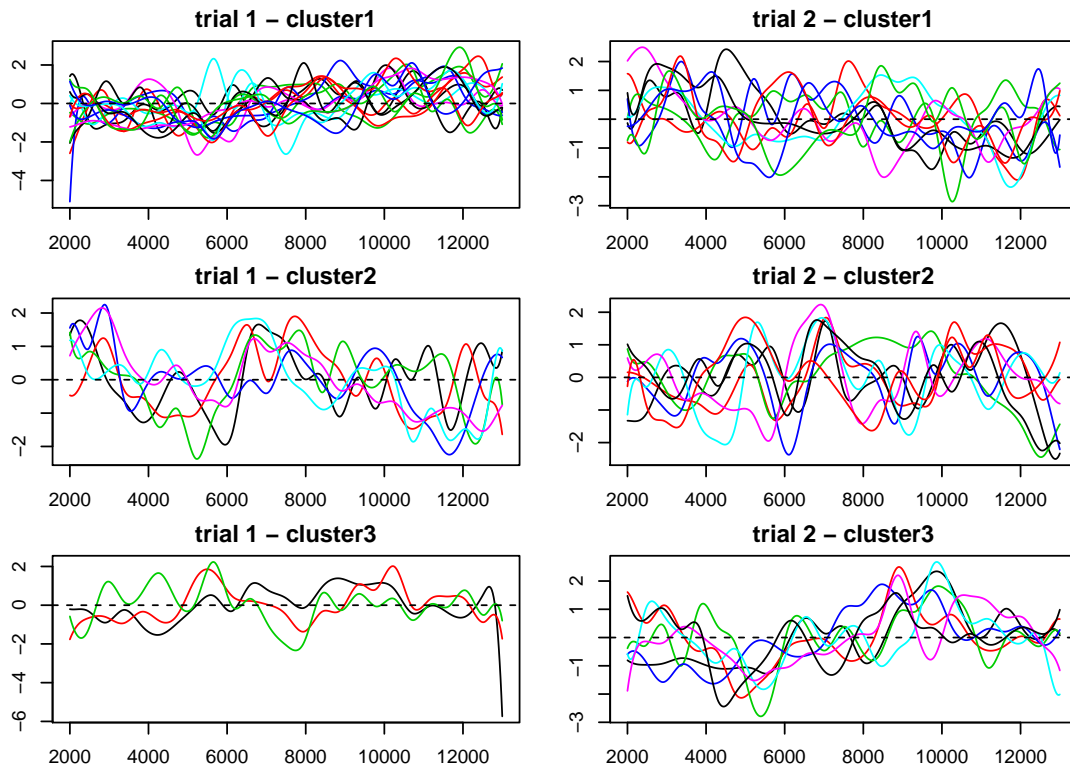
```
plot_cluster_curves2(data_reg2, clust2)
legend("topleft", c("cluster 1", "cluster 2", "cluster 3"), lty=1, col=1:3)
abline(v=c(5000, 8000))
```



```

par(mfrow=c(3,2),mar = c(2, 2, 2, 2),mgp=c(5,1,0))
for (i in 1:3){
plot(data_reg1[clust1==i],lty=1)
title(paste("trial 1 - cluster",i, sep=""))
plot(data_reg2[clust2==i],lty=1)
title(paste("trial 2 - cluster",i, sep=""))
}

```



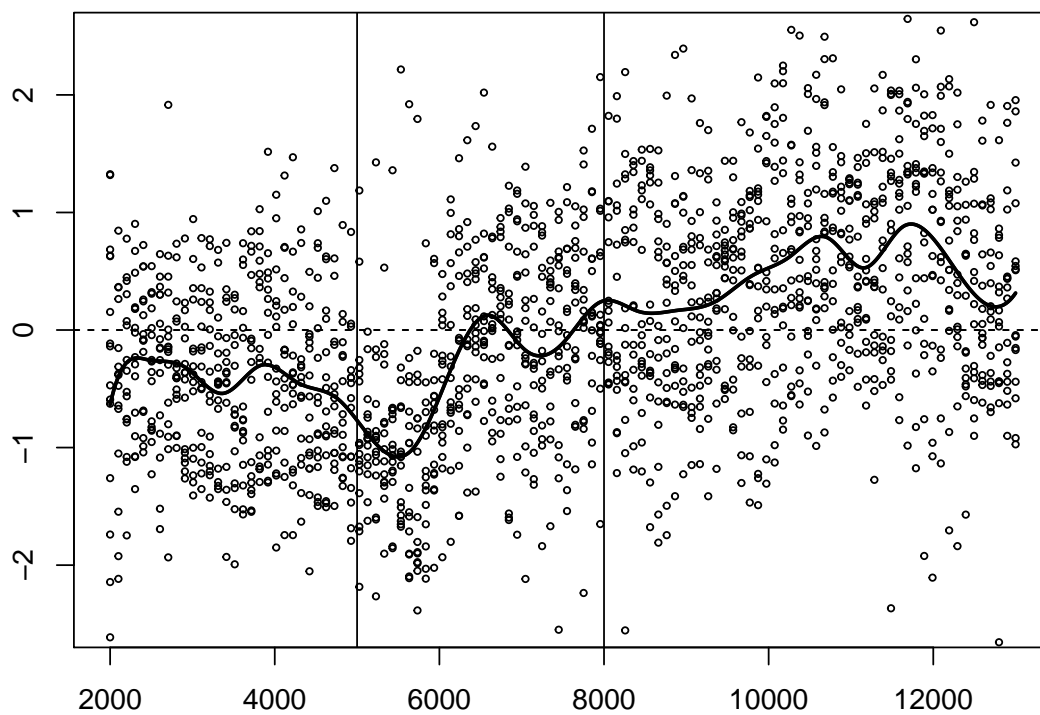
```

par(mfrow=c(1,1))

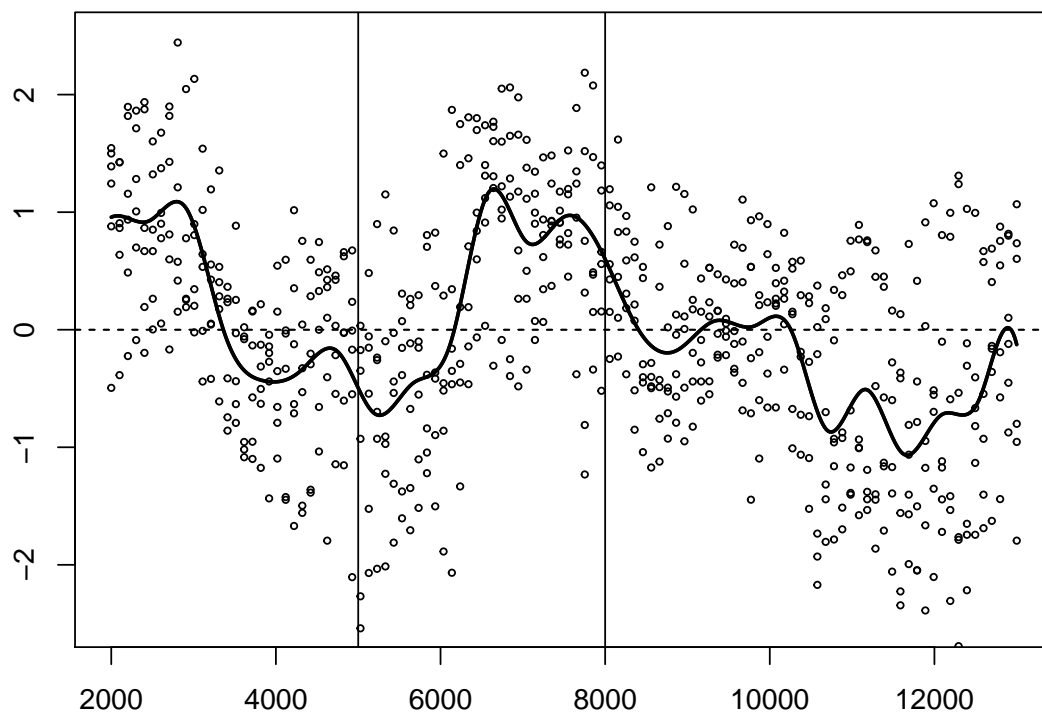
for (i in 1:max(clust1)){
plot_cluster_curves2(data_reg1[i %in% clust1],clust1==i,lwd=2)
abline(v=c(5000,8000))
title(paste("trial 1 cluster",i))
for(j in 1:table(clust1)[i]) points(data_sr1$t,data_sr1$x_mat[clust1==i,][j,],cex=0.5)
}

```

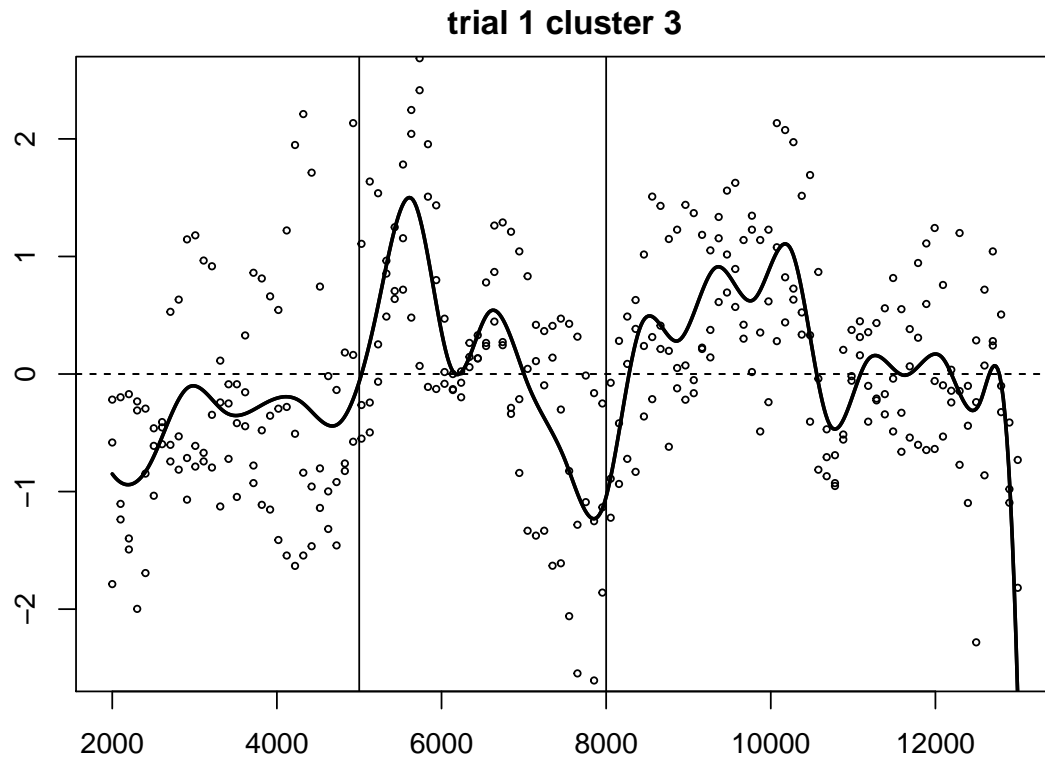
trial 1 cluster 1



trial 1 cluster 2

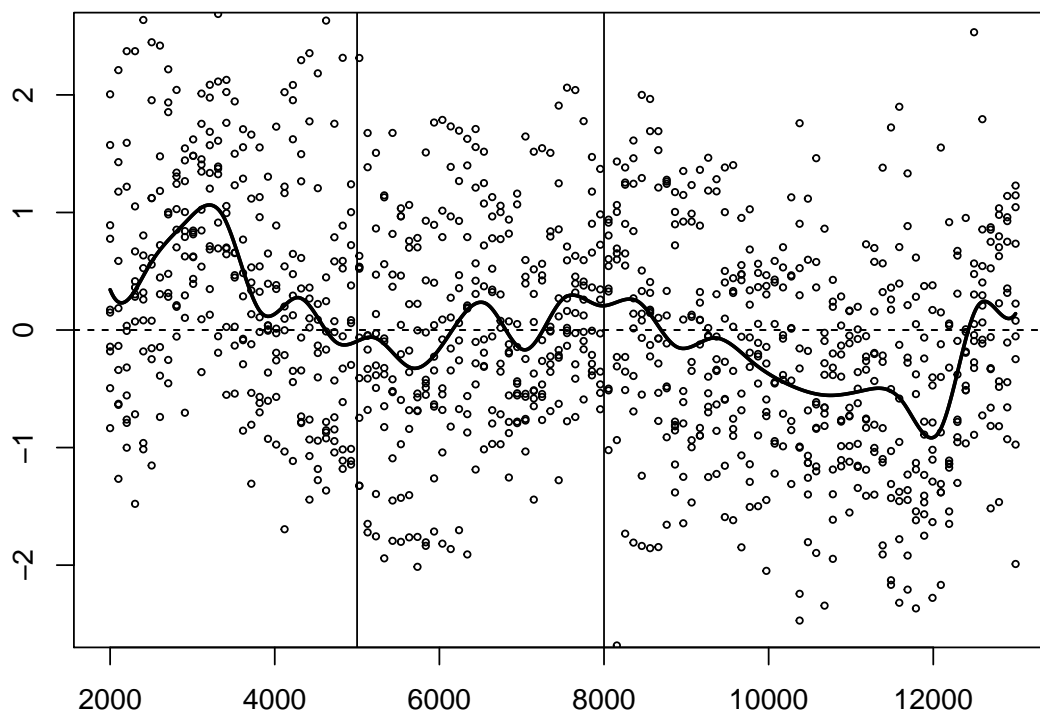




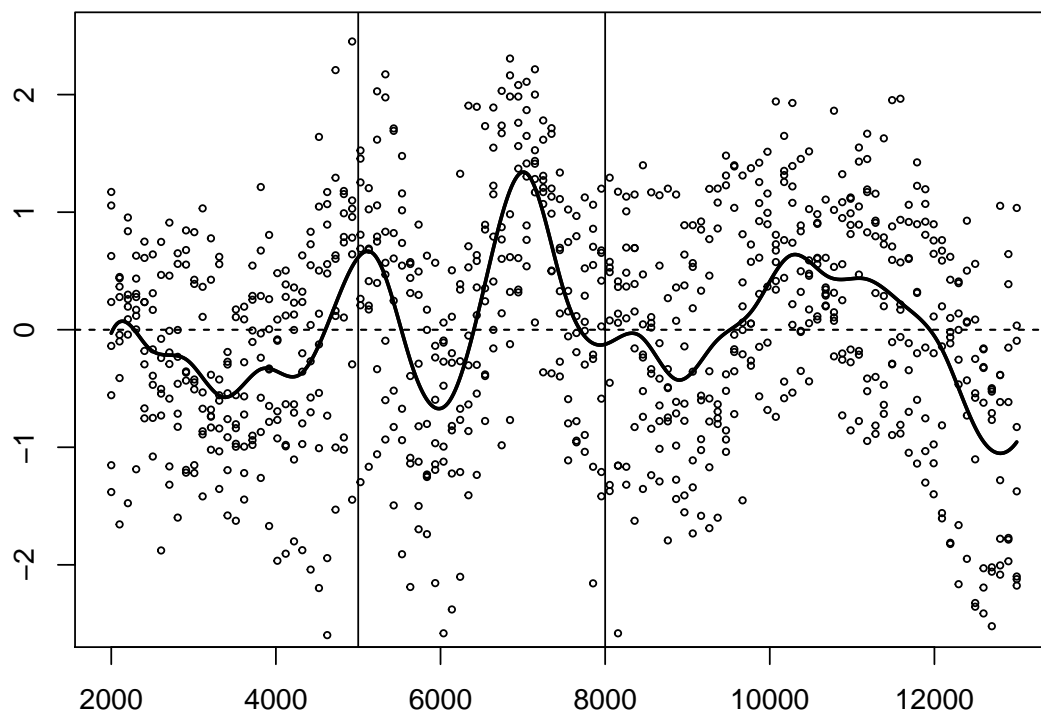


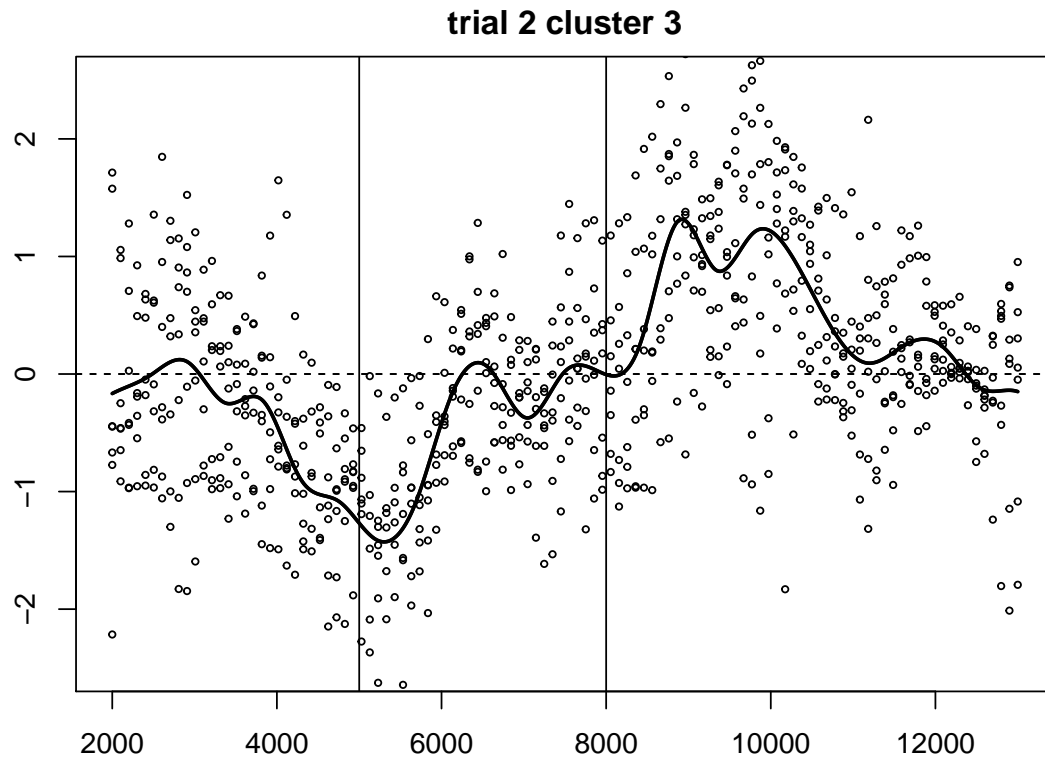
```
for (i in 1:max(clust2)){
  plot_cluster_curves2(data_reg2[i %in% clust2],clust2==i,lwd=2)
  abline(v=c(5000,8000))
  title(paste("trial 2 cluster",i))
  for(j in 1:table(clust2)[i]) points(data_sr2$t,data_sr2$x_mat[clust2==i,][j,],cex=0.5)
}
```

trial 2 cluster 1



trial 2 cluster 2





## Conclusions

To be continued. . .

## Other

R-packages for EyeTrackerData  
<http://www.eyetracking-r.com/>  
 #References

Jacques, Julien, and Cristian Preda. 2014. "Functional Data Clustering: A Survey." *Advances in Data Analysis and Classification* 8 (3): 231–55.

Tibshirani, Robert, Guenther Walther, and Trevor Hastie. 2001. "Estimating the Number of Clusters in a Data Set via the Gap Statistic." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2): 411–23.