fMRI Data Analysis and fPCA version 1.1

Zach Wang

6/15/2020

Progress summary

In previous work, we used a set of fourier-series basis function to smooth the entire 32 nodes, and then applied fPCA to analyze the variance of the smoothed functions.

Now I am going to try to smooth only a subset of nodes, hopefully will be able to get more insights for the patterns.

I first showed some experiment results by selecting a subset of nodes to perform fPCA on. Then I created a visualization based on fPCA column-scores (a scalar metrics) that I used to guide my choice of node subsets.

1. fPCA with different choices of nodes subset

Below is a short list of different choices of nodes subset.

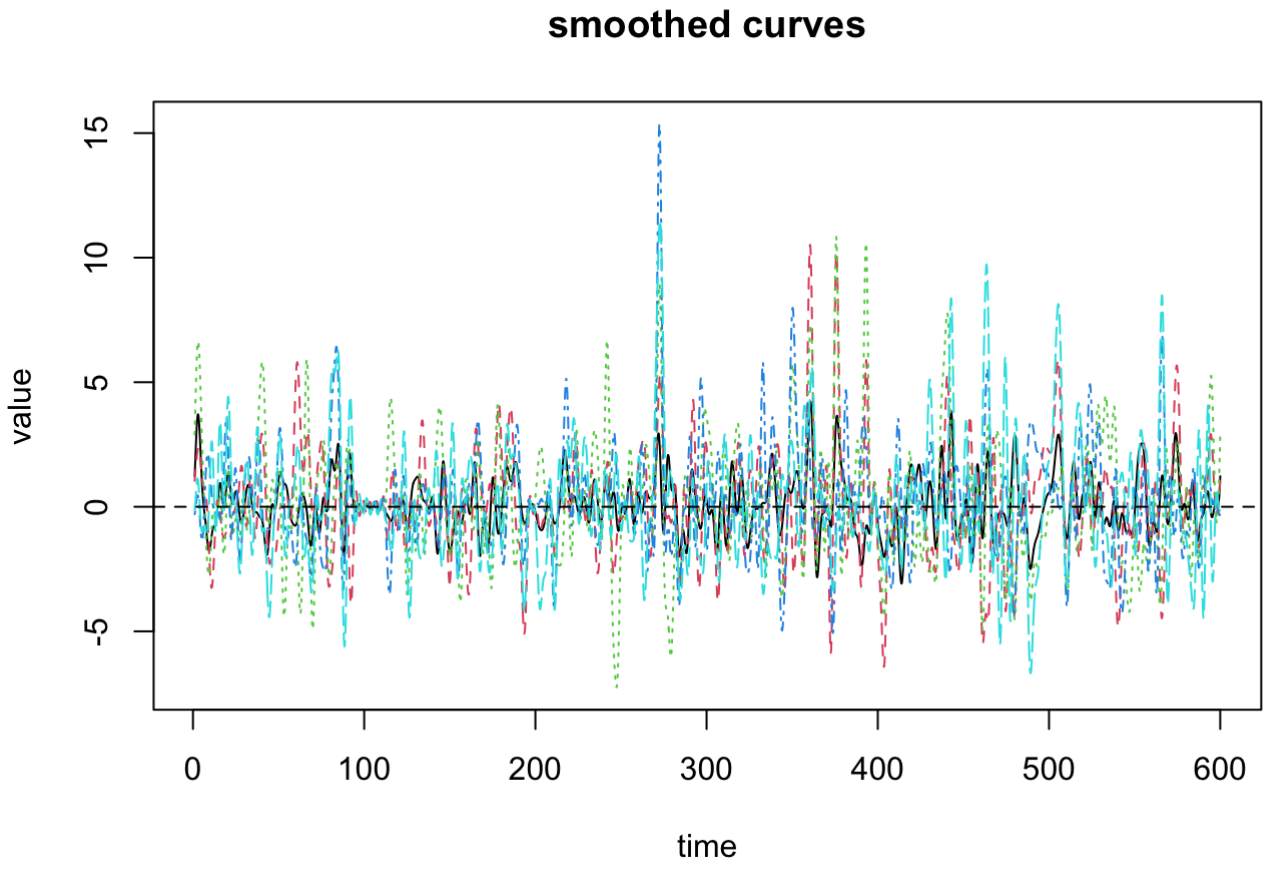
For nodes in Part 1.1, I chose them because those nodes have large value above or below 1.5 interquartile ranges (outliers). For nodes in Part 1.2, I simply chose first 5 nodes as a comparison to nodes in part 1.1. For nodes in Part 1.3, I chose them according to the visualization tool I created in part 3 based on the column-scores (a scalar evaluation provided by fPCA function) – SM2, SM9, SM15 and SM19 are located far from the majority of the node clusters. For nodes in part 1.4, I removed SM1 from the subset in part 1.3.

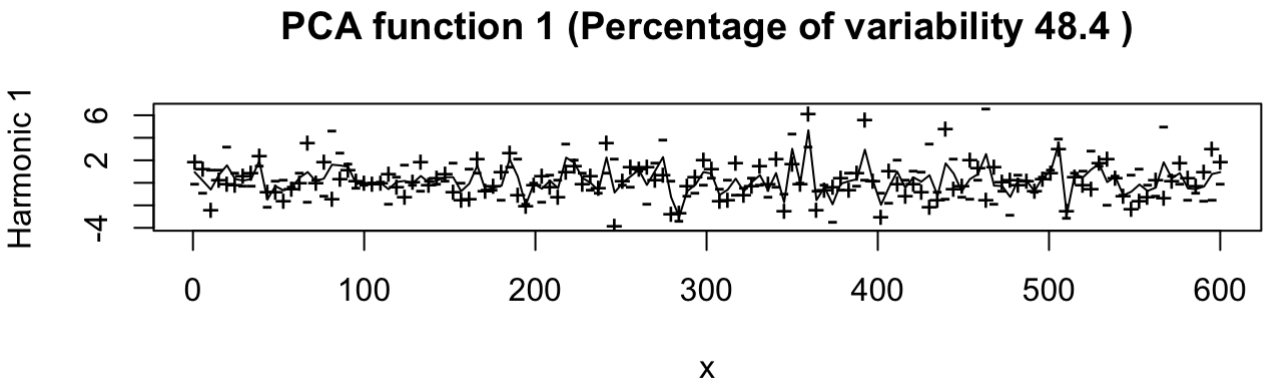
1.1 fPCA with 5 nodes: SM1, SM6, SM7, SM9, SM15

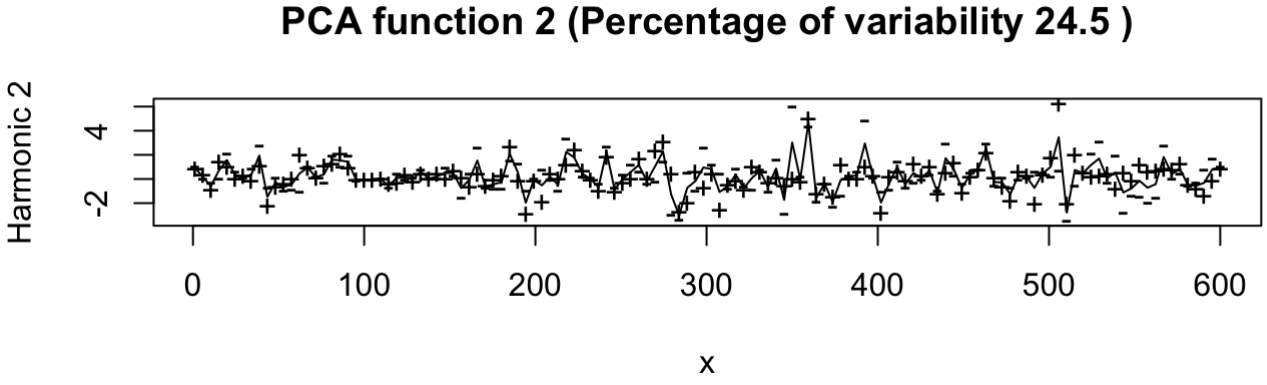
Includes 5 nodes and fPCA with 2 eigenfunctions is applied: total Percentage of variability 72.9%

```
fPCA_subset <- function(time_subset, data_mat, node_subset, k, nharm){
  basis <- create.fourier.basis(c(time_subset[1],time_subset[length(time_subset)]), k)
  smoothfd <- smooth.basis(time_subset, data_mat[time_subset,node_subset], basis)$fd
  plot(smoothfd)
  title(main="smoothed curves")
  pcalist = pca.fd(smoothfd, nharm, harmfdPar=fdPar(smoothfd))
  rotpcalist = varmx.pca.fd(pcalist)
  par(mfrow=c(nharm,1))
  plot.pca.fd(rotpcalist)
  return(rotpcalist)
}

rotpcalist = fPCA_subset(time_subset=c(1:600), data_mat, node_subset = c(1,6,7,9,15)
                         , k=280, nharm=2)
```


smoothed curves
value
time
15
10
5
0
-5
0
100
200
300
400
500
600

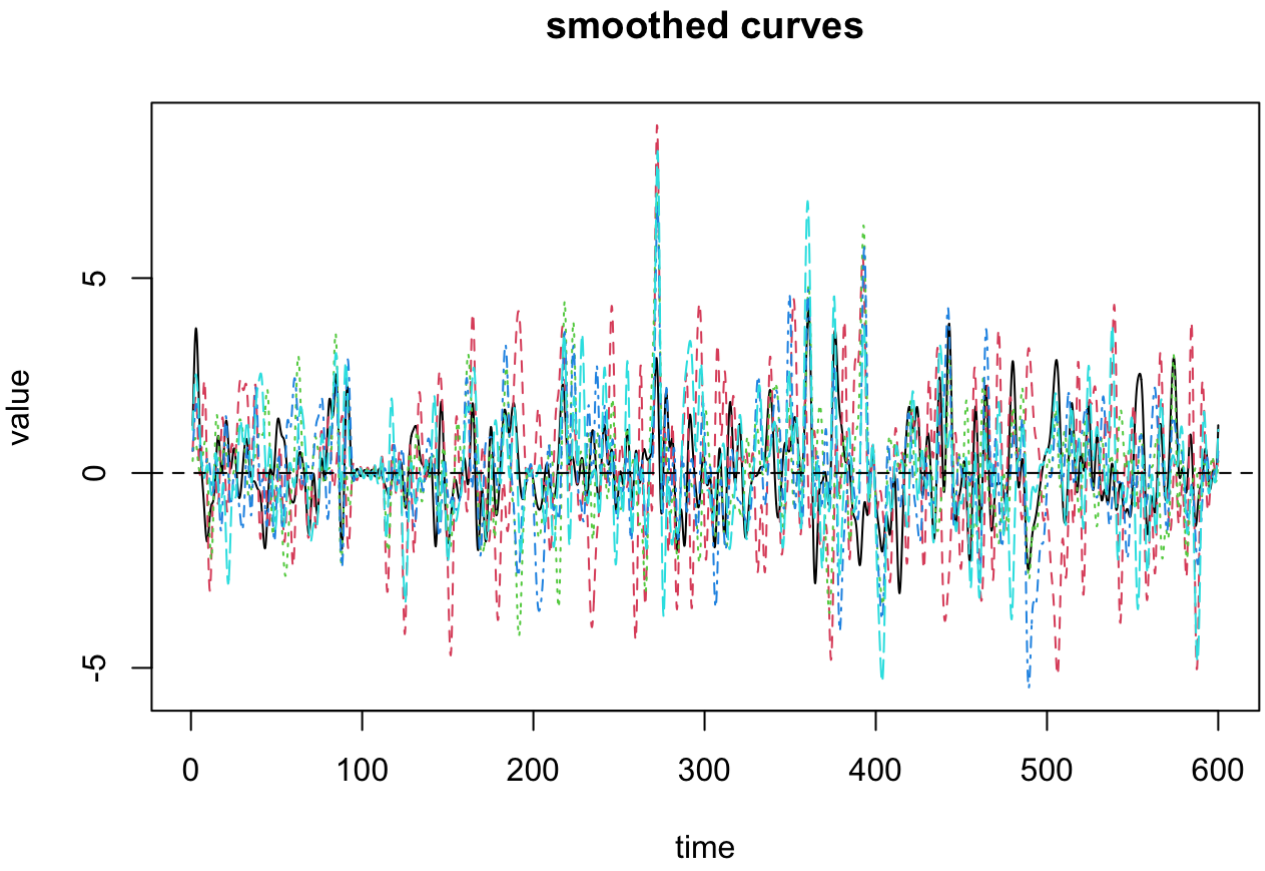

PCA function 1 (Percentage of variability 48.4)
Harmonic 1
x
6
2
-4
0
100
200
300
400
500
600

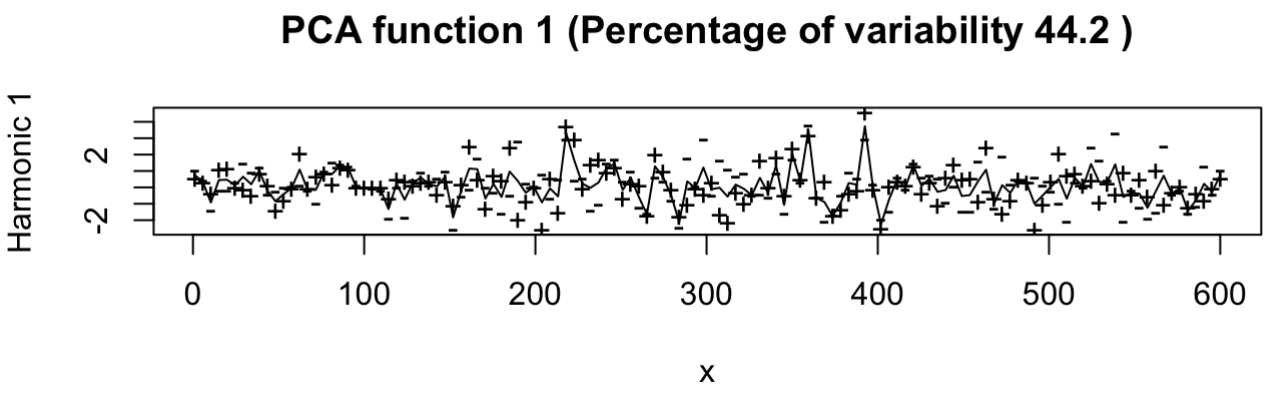

PCA function 2 (Percentage of variability 24.5)
Harmonic 2
x
4
-2
0
100
200
300
400
500
600

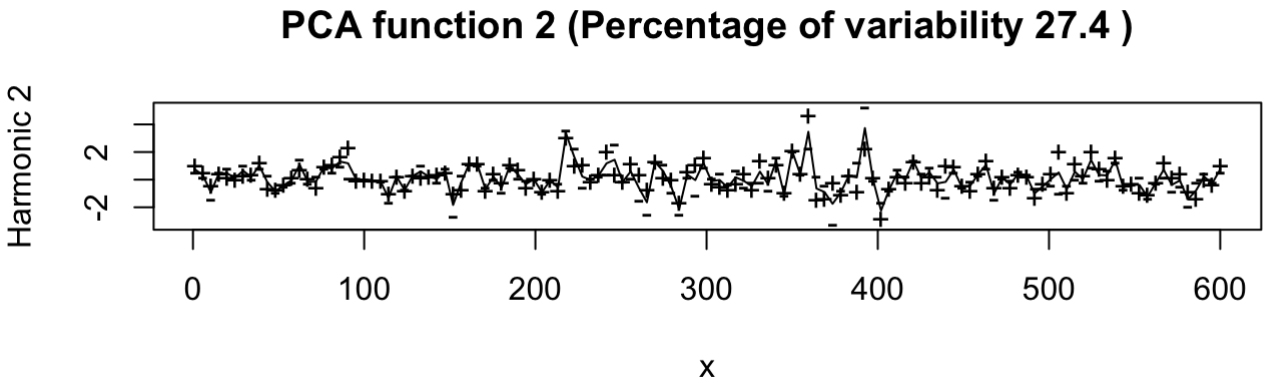
1.2 fPCA with 5 node SM1,SM2, SM3, SM4, SM5

Includes 5 nodes and fPCA with 2 eigenfunctions is applied: total Percentage of variability 71.6%

```
rotpcalist = fPCA_subset(time_subset=c(1:600), data_mat, node_subset = c(1,2,3,4,5)
                         , k=280, nharm=2)
```



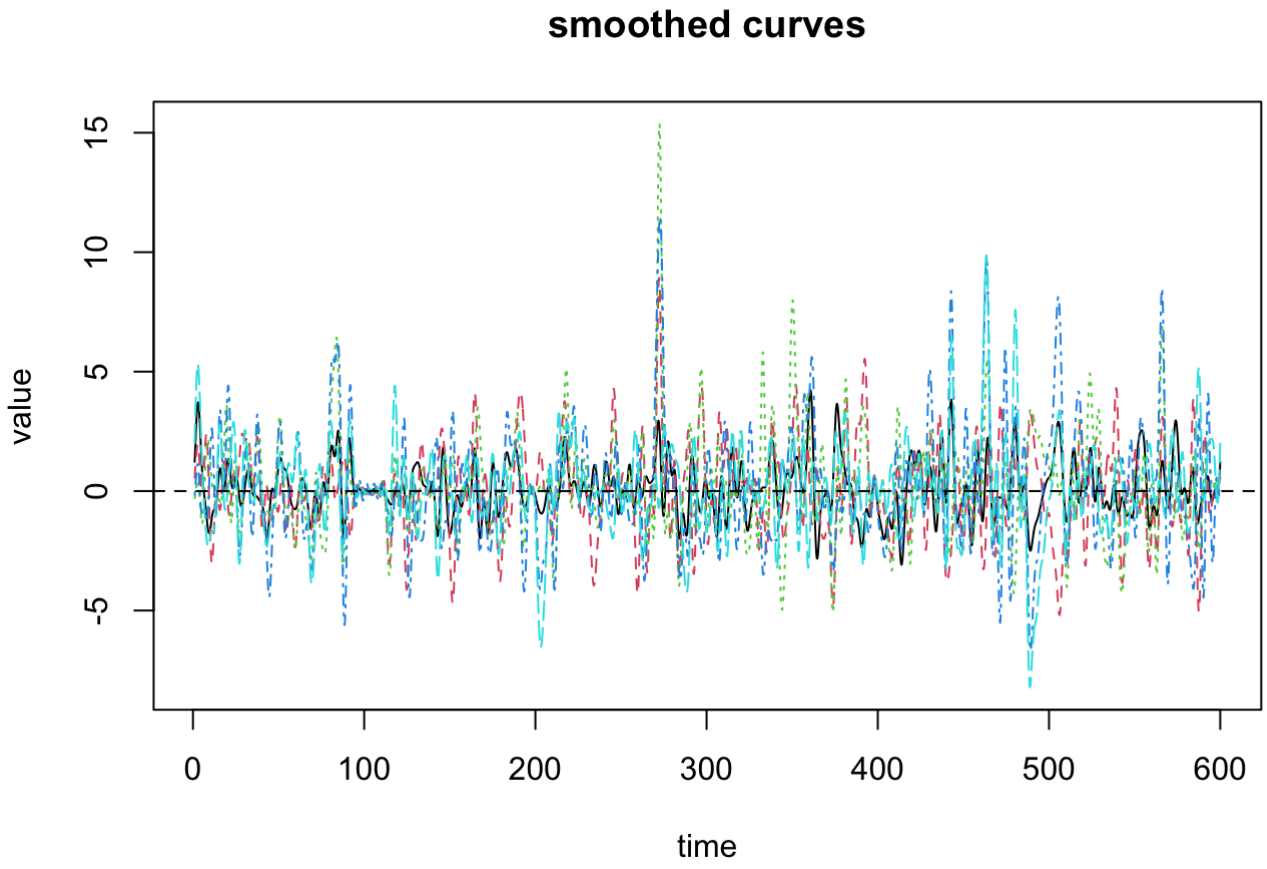


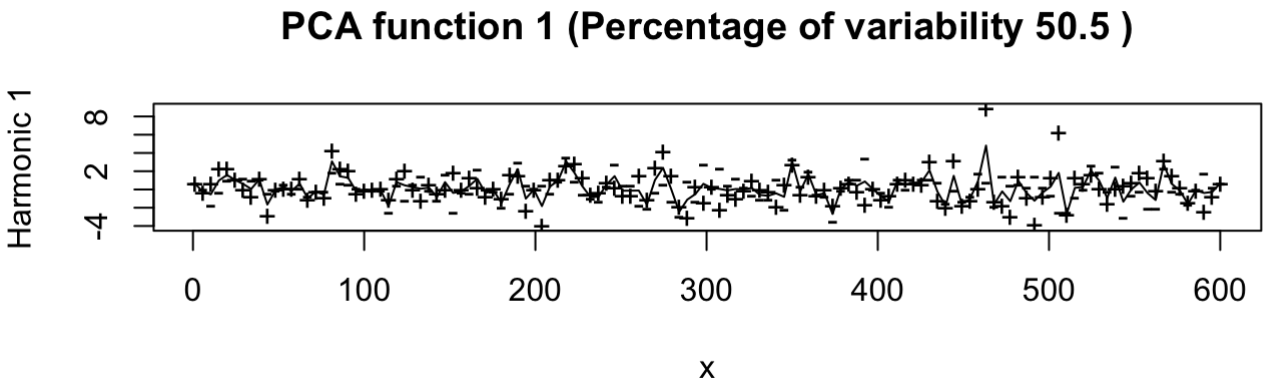


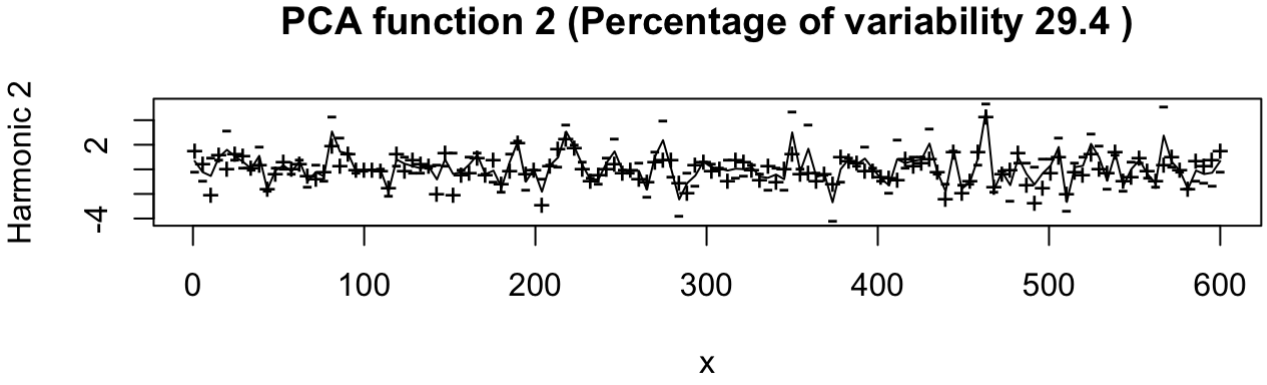
1.3 fPCA with 5 node SM1,SM2, SM9, SM15, SM19

Includes 5 nodes and fPCA with 2 eigenfunctions is applied: total Percentage of variability 79.9%

```
rotpcalist = fPCA_subset(time_subset=c(1:600), data_mat, node_subset = c(1,2,9,15,19)
                         , k=280, nharm=2)
```


smoothed curves
value
15
10
5
0
-5
0
100
200
300
400
500
600
time

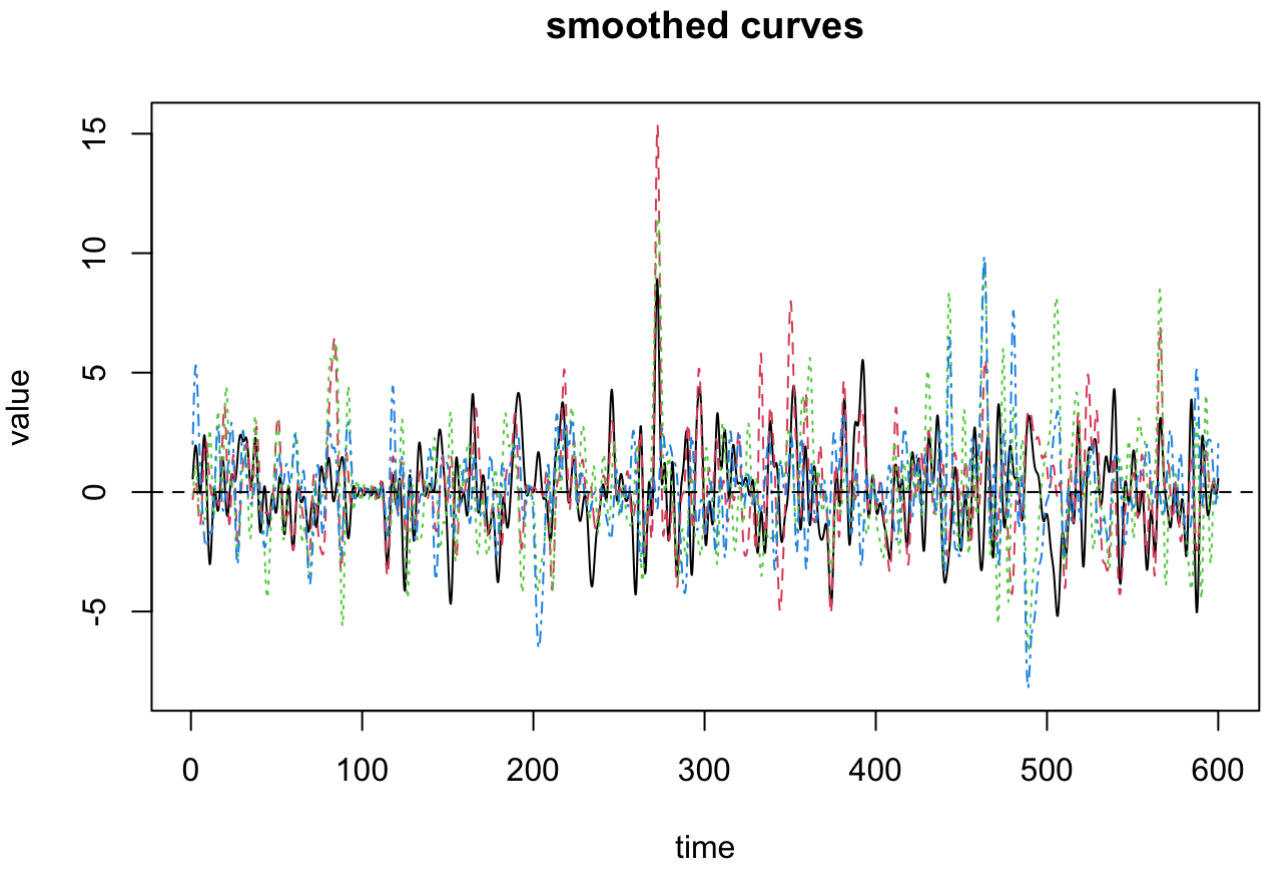

PCA function 1 (Percentage of variability 50.5)
Harmonic 1
8
2
-4
0
100
200
300
400
500
600
x

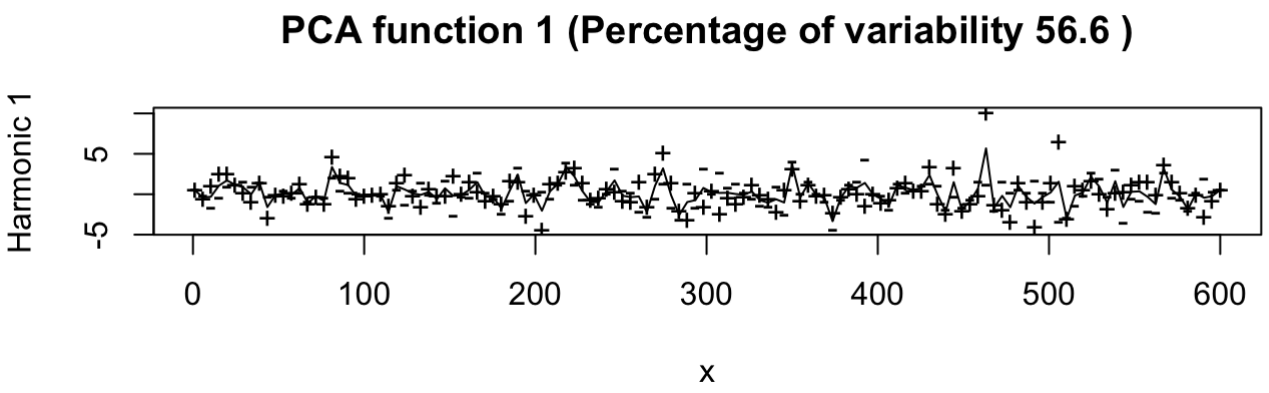

PCA function 2 (Percentage of variability 29.4)
Harmonic 2
2
-4
0
100
200
300
400
500
600
x

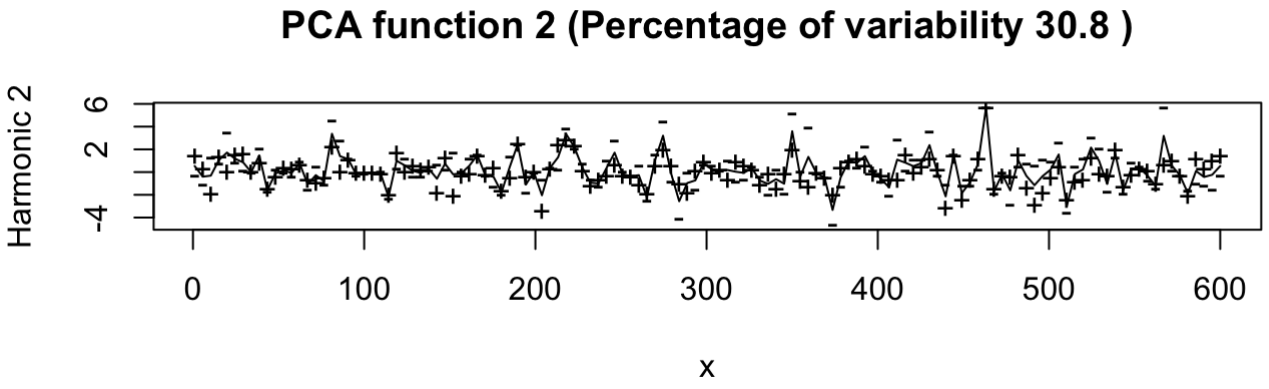
1.4 fPCA with 4 node SM2, SM9, SM15, SM19

Includes 4 nodes and fPCA with 2 eigenfunctions is applied: total Percentage of variability 87.4%

```
rotpcalist = fPCA_subset(time_subset=c(1:600), data_mat, node_subset = c(2,9,15,19)
                         , k=280, nharm=2)
```



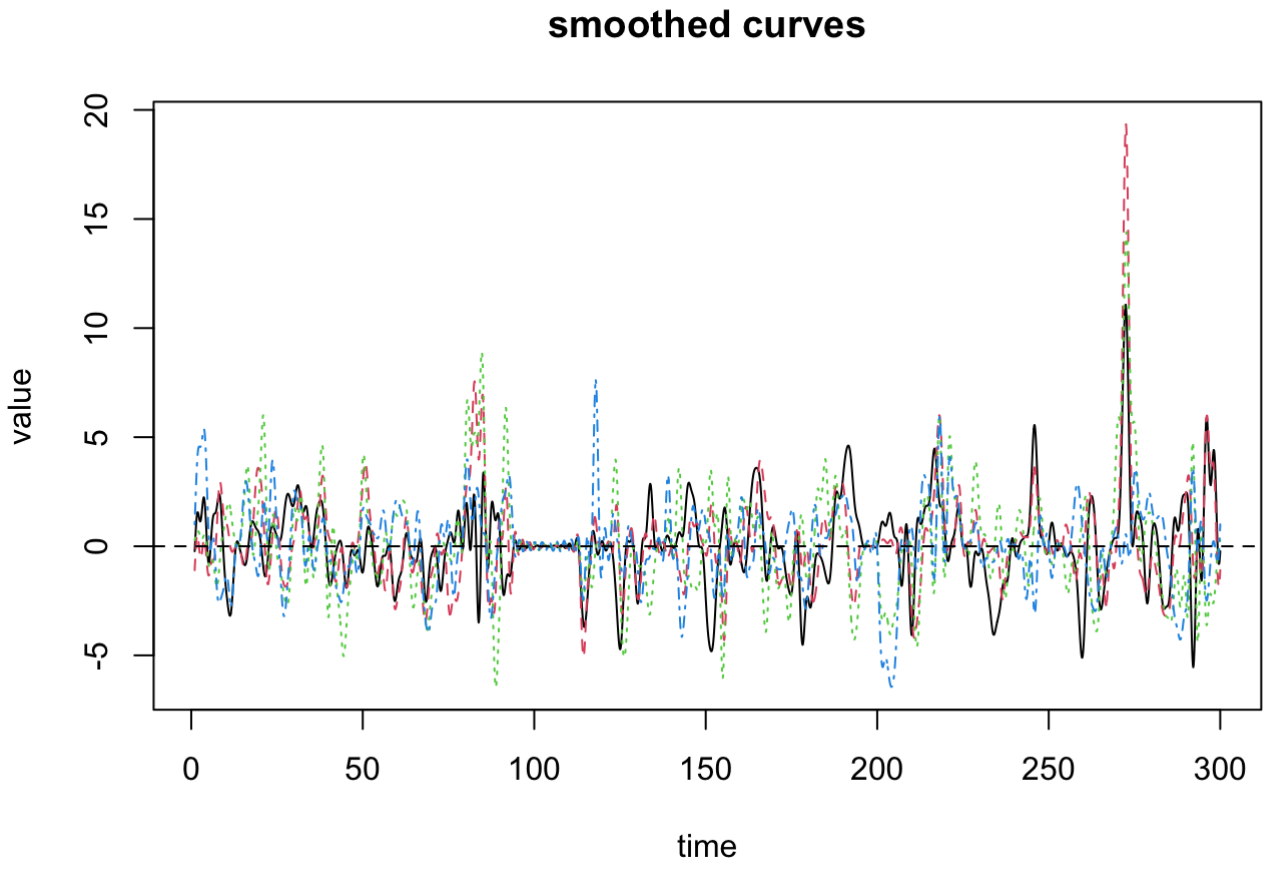


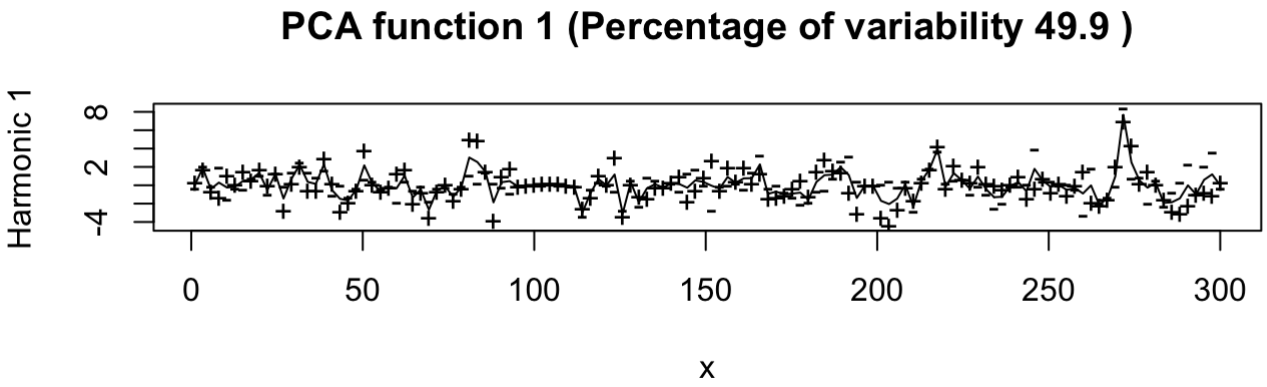


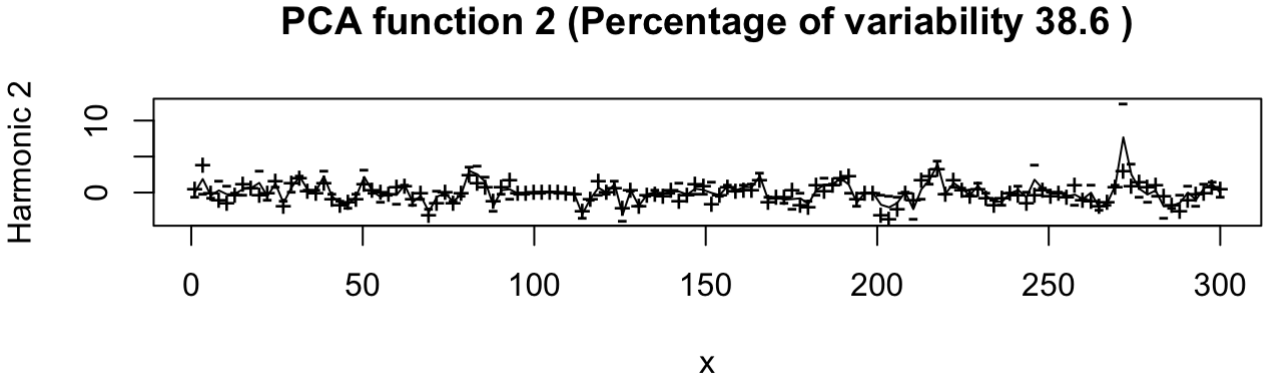
1.5 fPCA with 4 node SM2, SM9, SM15, SM19, but with shorter time period (1 to 300)

Includes 4 nodes and fPCA with 2 eigenfunctions is applied: total Percentage of variability 88.5%

```
rotpcalist = fPCA_subset(time_subset=c(1:300), data_mat, node_subset = c(2,9,15,19)
                         , k=280, nharm=2)
```


smoothed curves
value
time
-5
0
5
10
15
20
0
50
100
150
200
250
300

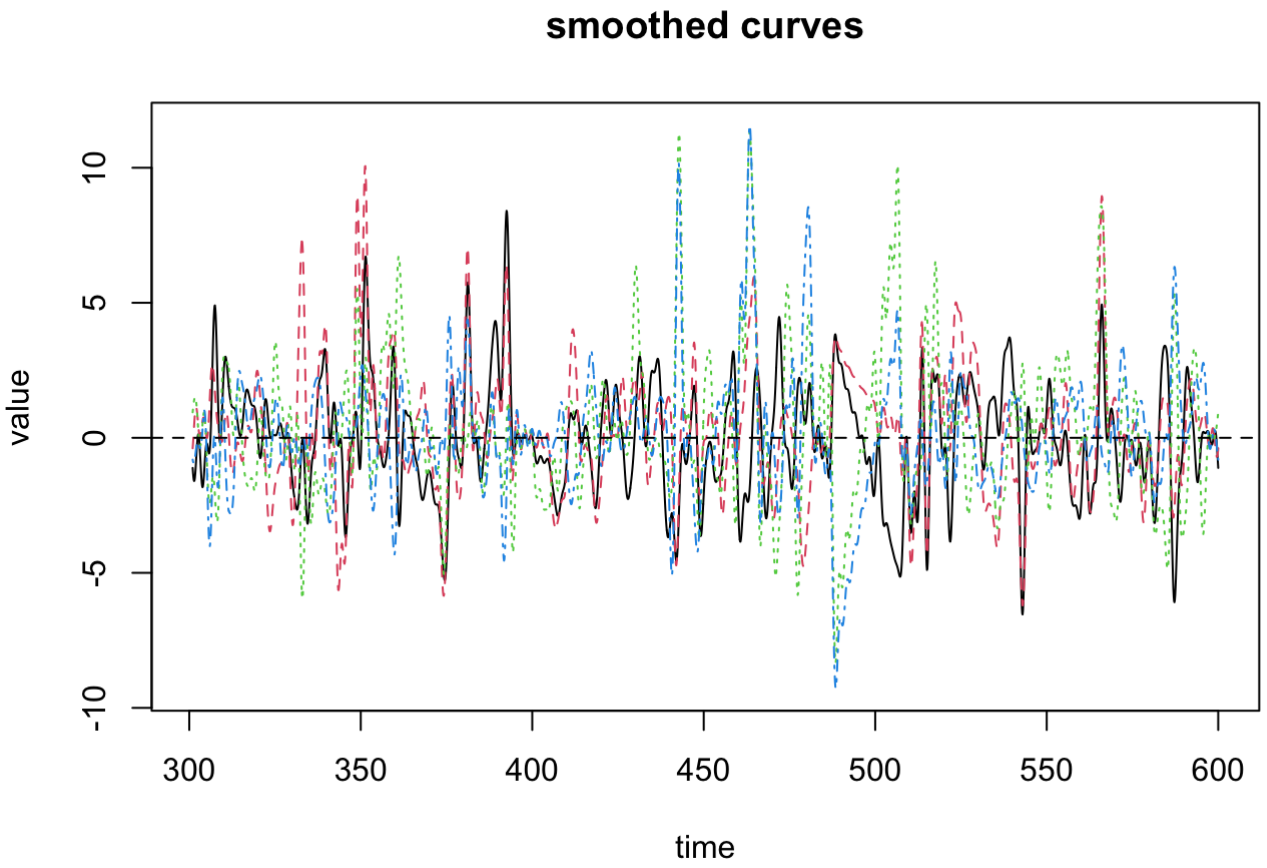

PCA function 1 (Percentage of variability 49.9)
Harmonic 1
x
-4
2
8
0
50
100
150
200
250
300

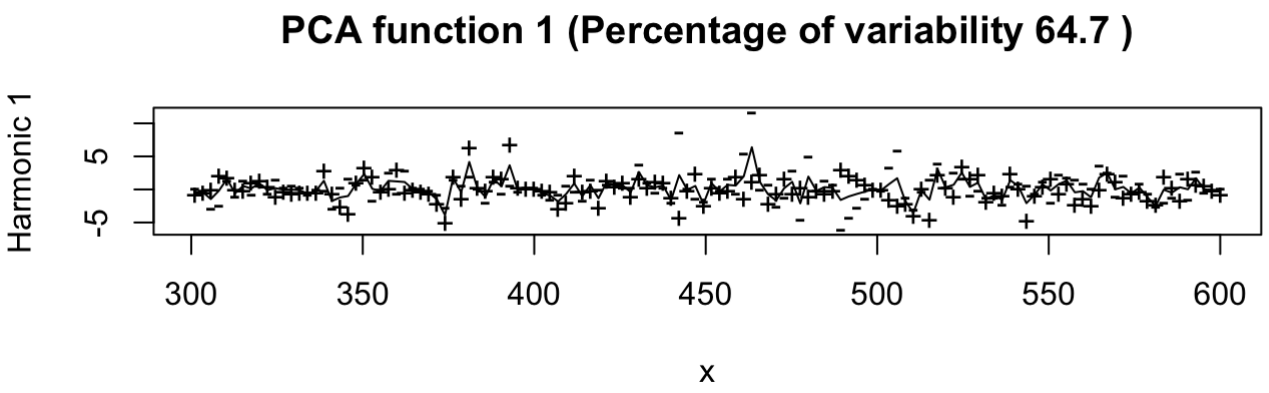

PCA function 2 (Percentage of variability 38.6)
Harmonic 2
x
0
10
0
50
100
150
200
250
300

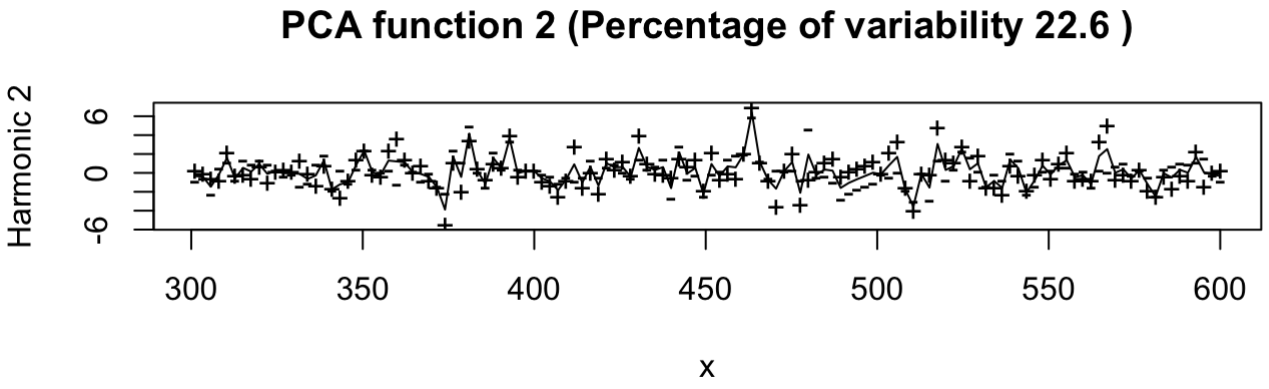
1.6 fPCA with 4 node SM2, SM9, SM15, SM19, but with shorter time period (301 to 600)

Includes 4 nodes and fPCA with 2 eigenfunctions is applied: total Percentage of variability 87.3%

```
rotpcalist = fPCA_subset(time_subset=c(301:600), data_mat, node_subset = c(2,9,15,19)
                         , k=280, nharm=2)
```







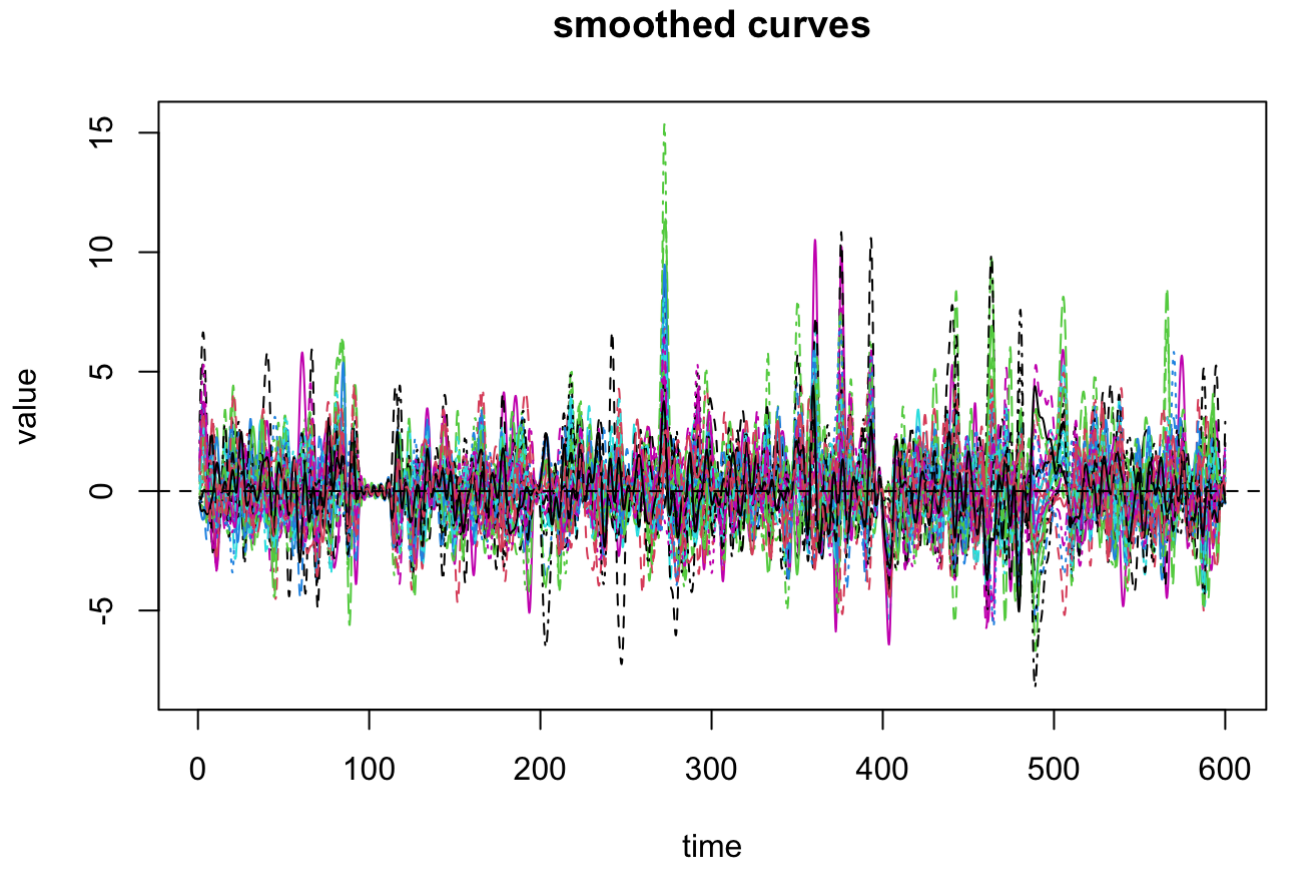
Notes of discoveries:

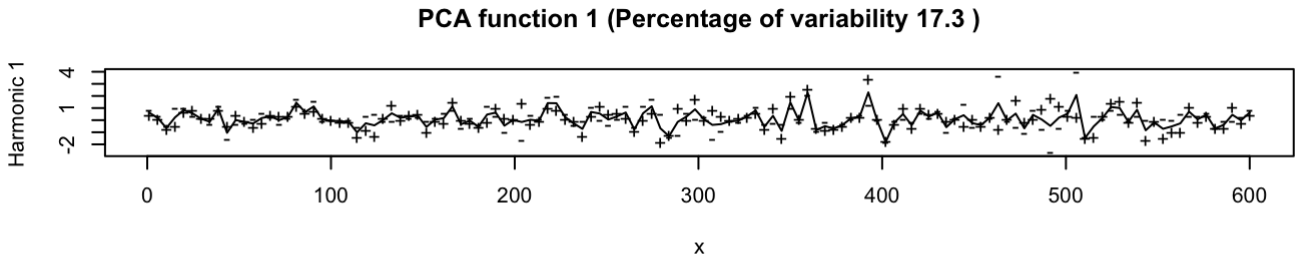
Decreasing the number of nodes will have an effect on fPCA Percentage of Variability, while decreasing the number of observations won't or has only little effect.

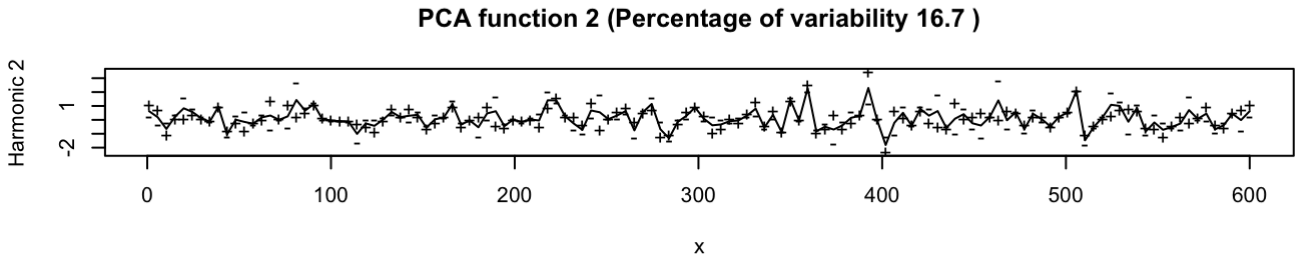
The choices of nodes in part 1.3 has more Percentage of variability explained. However, the validity of this approach needs further investigation. One thing we can tell now, is that the behavior of nodes tend to be different, and different choices of nodes subset will have effects on fPCA performance. We need to figure out ways to cluster nodes together to reveal more patterns.

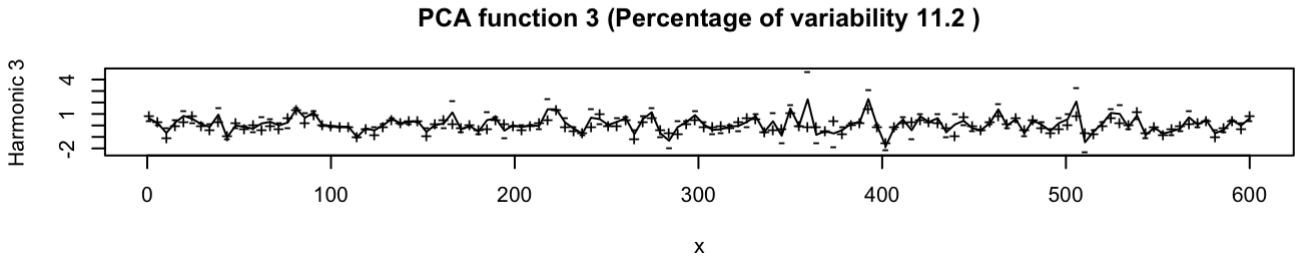
2. scores along eigenfunctions 3Dplot

```
rotpcalist = fPCA_subset(time_subset=c(1:600), data_mat, node_subset = c(1:32)
                         , k=280, nharm=3)
```

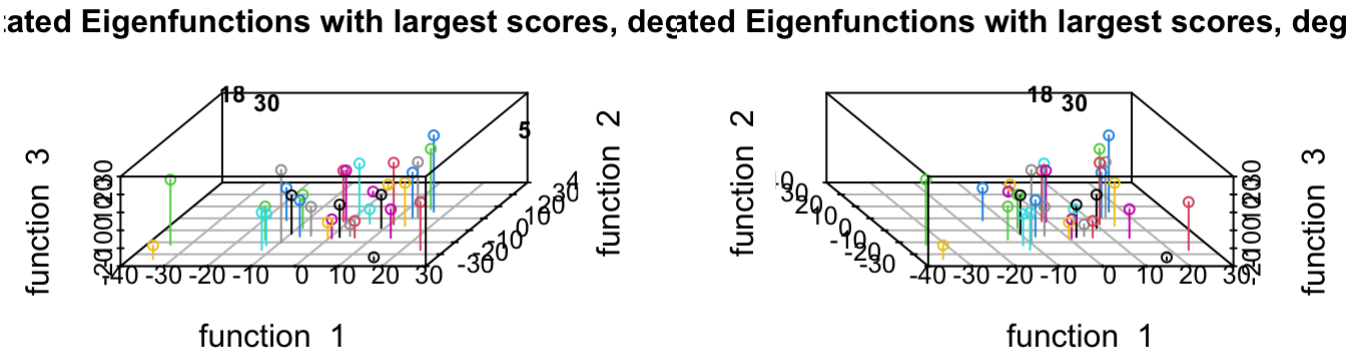

smoothed curves
value
time
15
10
5
0
-5
0
100
200
300
400
500
600

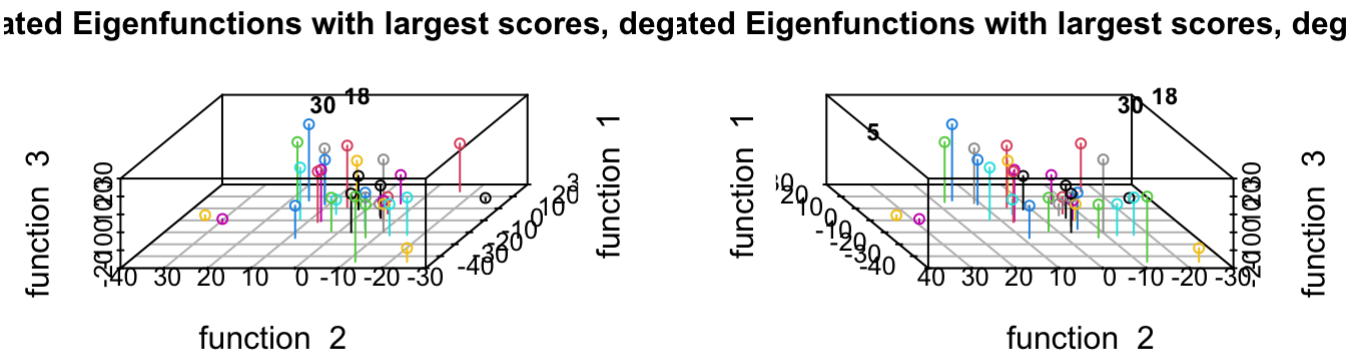

PCA function 1 (Percentage of variability 17.3)
Harmonic 1
x


PCA function 2 (Percentage of variability 16.7)
Harmonic 2
x


PCA function 3 (Percentage of variability 11.2)
Harmonic 3
x

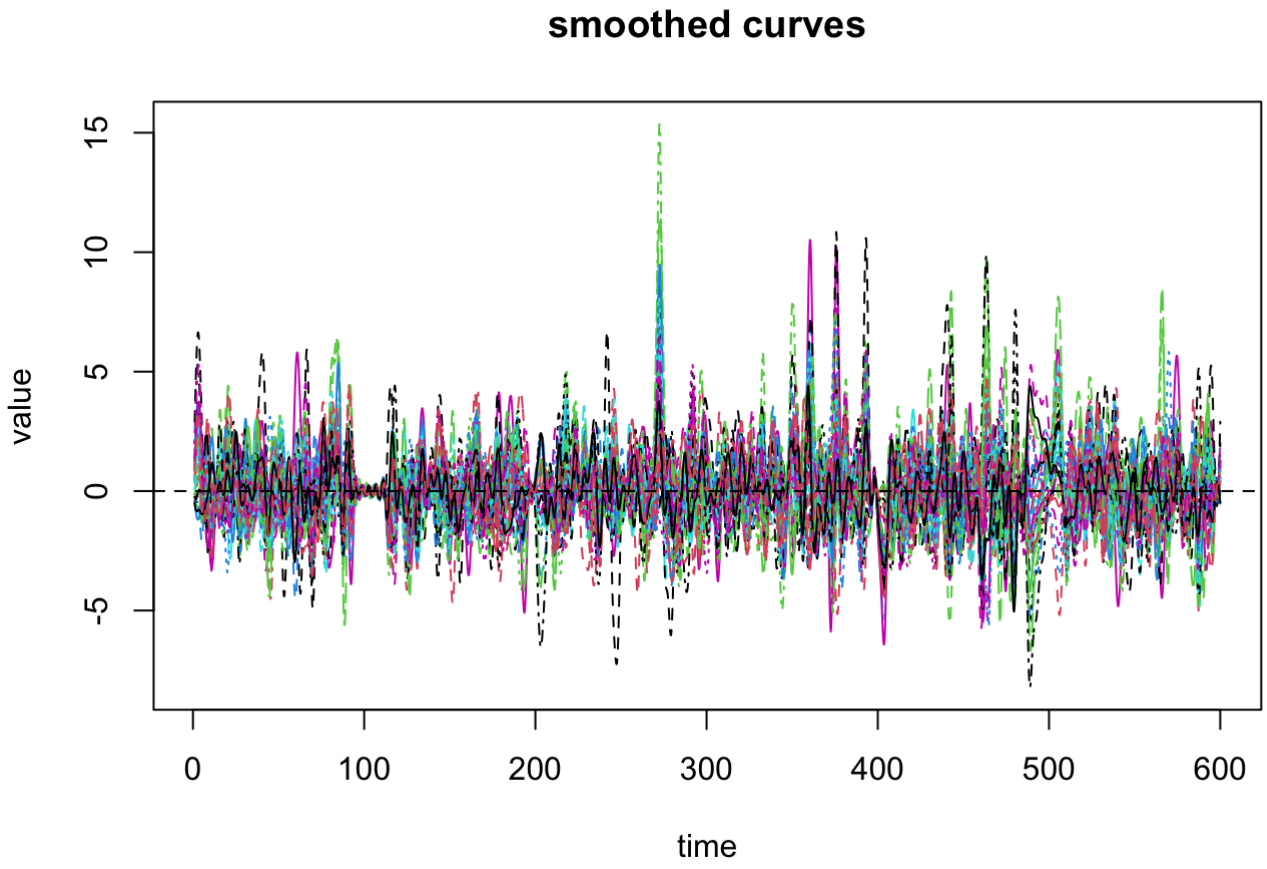
```
colvar <- rotpcalist$varprop
colScores <- rotpcalist$scores
lst <- sort(colvar, index.return=TRUE, decreasing=TRUE)
topidx <- lapply(lst, `[`, lst$x %in% head(unique(lst$x),3))$ix
par(mfrow=c(2,2))
for (deg in c(60,120,240,300)){
  scatterplot3d(colScores[,topidx[1]],colScores[, topidx[2]]
              ,colScores[,topidx[3]]
              ,xlab=paste('function ', topidx[1])
              ,ylab=paste('function ', topidx[2])
              ,zlab=paste('function ', topidx[3]),
              type='h', angle=deg, color =index(colScores)
              ,main = paste("Three rotated Eigenfunctions with largest scores, degree =
 ", deg))
  text(colScores[, topidx[2]]~colScores[,topidx[1]],labels=index(colScores),data=colScor
es, cex=0.9, font=2)
}
```

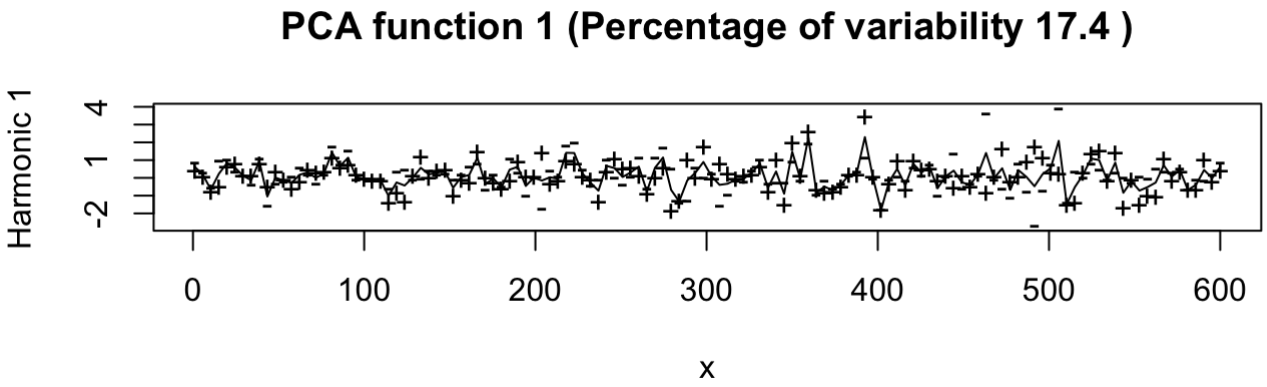


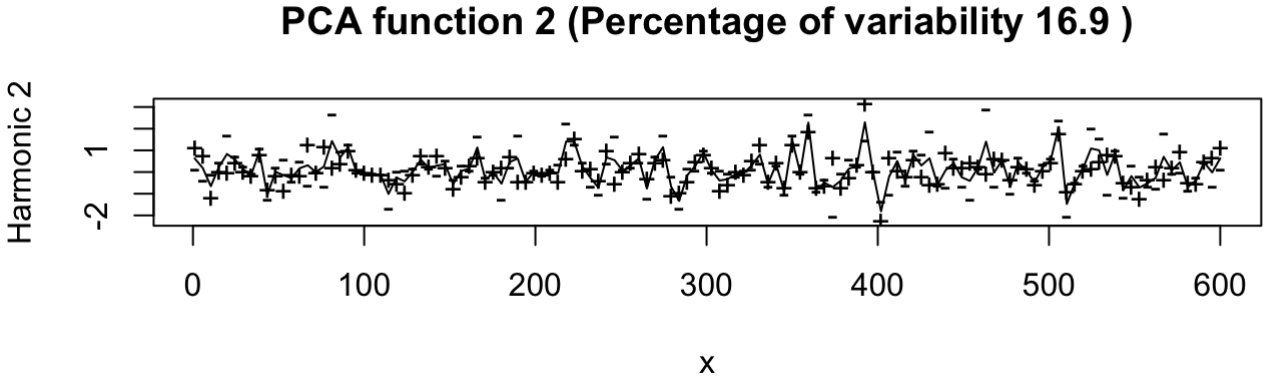


3. scores along eigenfunctions 2Dplot

```
rotpcalist = fPCA_subset(time_subset=c(1:600), data_mat, node_subset = c(1:32)
                         , k=280, nharm=2)
```


smoothed curves
value
time
15
10
5
0
-5
0
100
200
300
400
500
600

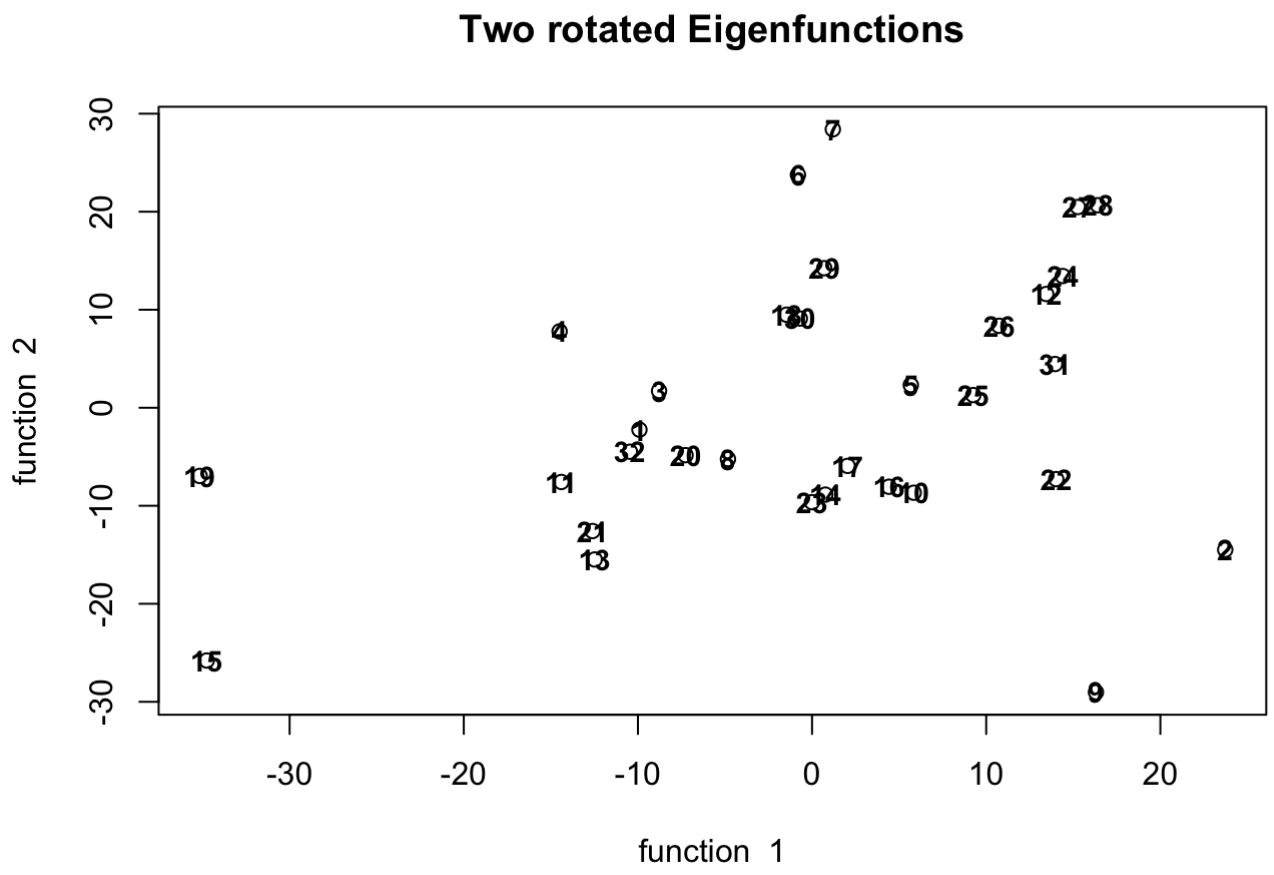

PCA function 1 (Percentage of variability 17.4)
Harmonic 1
x
4
1
-2
0
100
200
300
400
500
600


PCA function 2 (Percentage of variability 16.9)
Harmonic 2
x
1
-2
0
100
200
300
400
500
600

```
colvar <- rotpcalist$varprop
colScores <- rotpcalist$scores
lst <- sort(colvar, index.return=TRUE, decreasing=TRUE)
topidx <- lapply(lst, `[`, lst$x %in% head(unique(lst$x),3))$ix

par(mfrow=c(1,1))

plot(colScores[,topidx[1]],colScores[, topidx[2]]
                ,xlab=paste('function ', topidx[1])
                ,ylab=paste('function ', topidx[2])
                ,color =index(colScores)
        ,main = paste("Two rotated Eigenfunctions"))
text(colScores[, topidx[2]]~colScores[,topidx[1]],labels=index(colScores),data=colScore
s, cex=0.9, font=2)
```



plans for next step research

1. I need to keep reading chapters about registrations in the textbook. Our dataset has already been normalized and shifted to zero mean. I need to read more to understand how registrations influence the Eigenfunction decomposition
2. Since we don't have more direct information about the brain scanning experiments to assist us, in order to improve signal to noise ratio, I am thinking about imposing penalty to the smooth functions (smoothness constraints).

3. However, in the research paper I am reading, it is pointed out that the signals with low-frequency tends to be overrepresented in the first functional principle component (bias towards low-frequency signals). The standard approach is to apply high-frequency filters to the data and investigate signals of interest. I need to design an experiment to test this, and further understand the mechanism.

4. Also, it is not likely that all 32 nodes have the similar pattern, and thus it is not appropriate to apply a single basis function to smooth the entire functional data uniformly. Thus, a program to automatically determine the appropriate smoothing function parameters for each node should be developed.

More to come after discussion with Bill, Andrew and Rohan