fMRI Data Analysis and Curve Smoothing

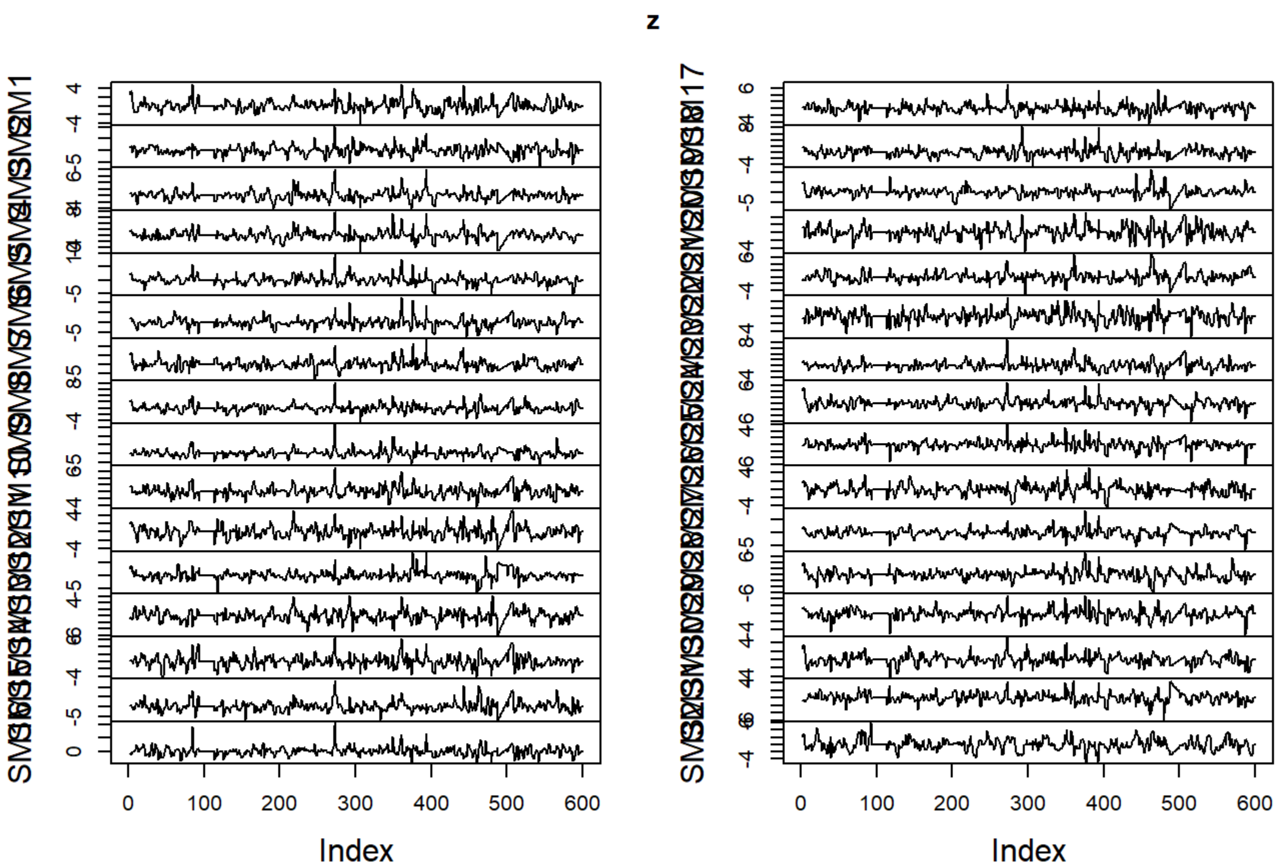
Zach Wang

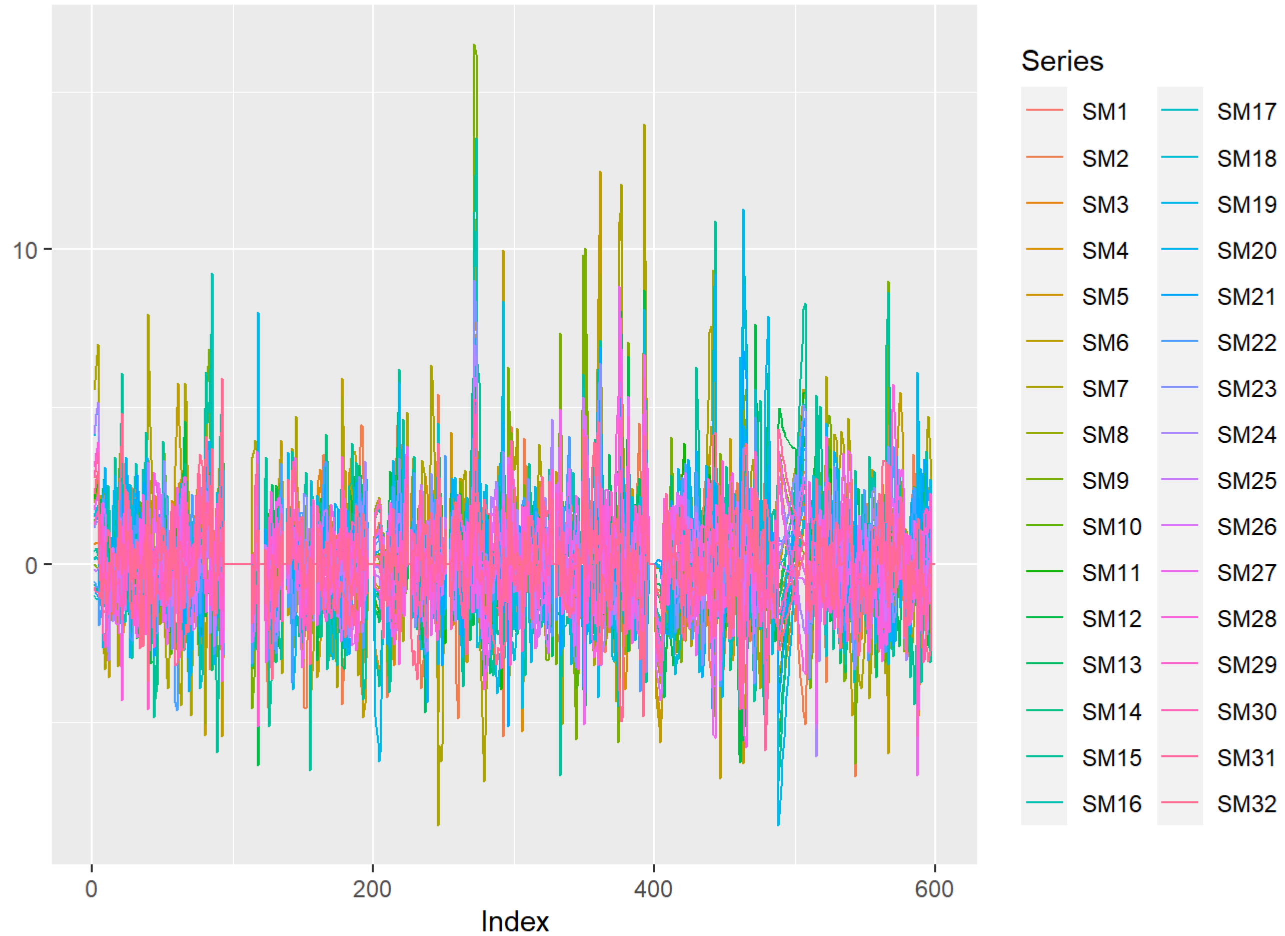
6/7/2020

Introduction

1. Plot entire time Series

The dataframe is converted to 'zoo' series to plot using package *zoo*. I first plot the 32 time series individually and then plot them all on one plot to compare. However, other than seeing some peakness possibly occur around the same time, it is hard to provide any further analysis for this high dimensional time series data.



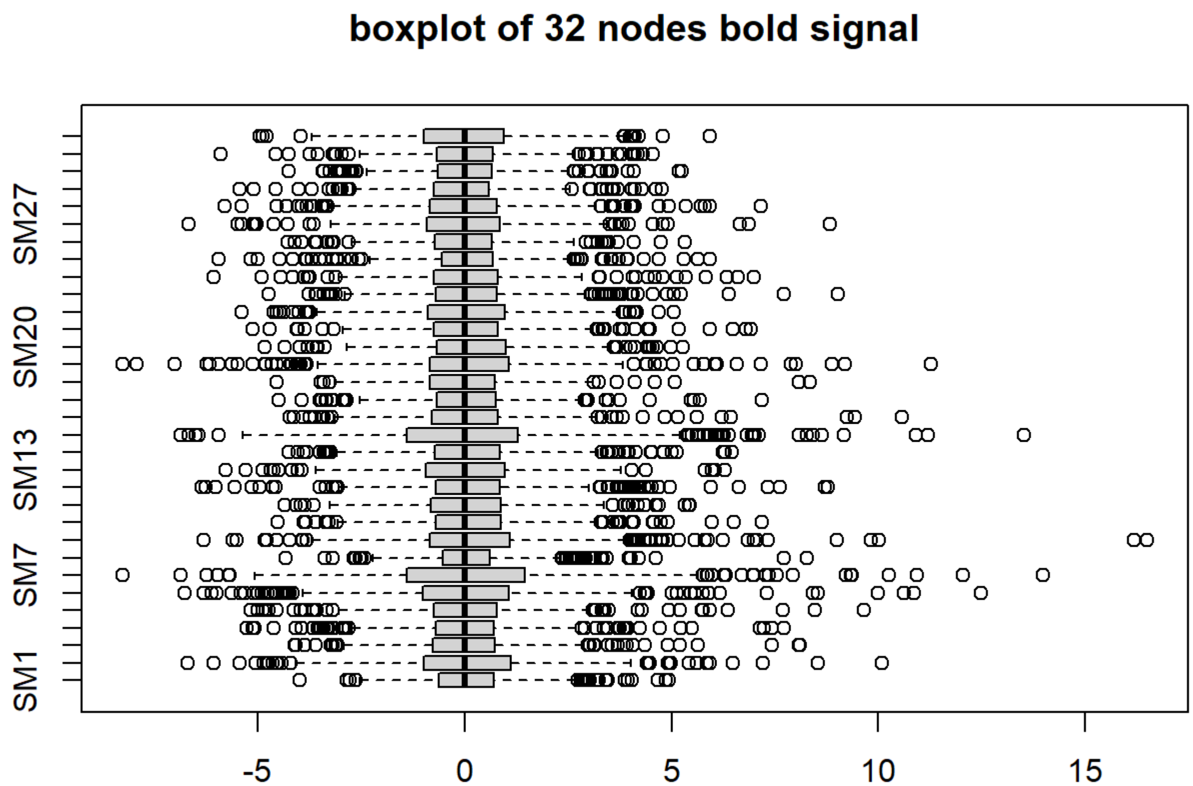


2. Summary Statistics

In the *boxplot*, I found several outliers with large value. Later I may try separate these outliers and put them on a different plot.

SM6, SM7, SM9 and SM15 have maximum value greater than 11. I will plot the paired lineplots of them in part 3.

```
# boxplot
boxplot(data_15may2020, horizontal = TRUE)
title(main="boxplot of 32 nodes bold signal")
```




```
# Summary
summary(data_15may2020)
```

```
##       SM1                SM2                 SM3                 SM4          
##  Min.   :-3.9946   Min.   :-6.69790   Min.   :-4.10330   Min.   :-5.28630  
##  1st Qu.:-0.6169   1st Qu.:-0.99755   1st Qu.:-0.76570   1st Qu.:-0.69760  
##  Median : 0.0000   Median : 0.00000   Median : 0.00000   Median : 0.00000  
##  Mean   : 0.1001   Mean   : 0.01257   Mean   : 0.05552   Mean   : 0.03187  
##  3rd Qu.: 0.6971   3rd Qu.: 1.12265   3rd Qu.: 0.71915   3rd Qu.: 0.69632  
##  Max.   : 4.9380   Max.   :10.08440   Max.   : 8.10310   Max.   : 7.72790  
##       SM5                 SM6                SM7                SM8          
##  Min.   :-5.17430   Min.   :-6.7834   Min.   :-8.2743   Min.   :-4.32390  
##  1st Qu.:-0.74713   1st Qu.:-1.0041   1st Qu.:-1.3890   1st Qu.:-0.52715  
##  Median : 0.00000   Median : 0.0000   Median : 0.0000   Median : 0.00000  
##  Mean   : 0.09668   Mean   : 0.1105   Mean   : 0.1739   Mean   : 0.09949  
##  3rd Qu.: 0.77695   3rd Qu.: 1.0542   3rd Qu.: 1.4444   3rd Qu.: 0.61080  
##  Max.   : 9.65990   Max.   :12.4982   Max.   :13.9896   Max.   : 8.27250  
##       SM9                SM10               SM11                SM12        
##  Min.   :-6.3056   Min.   :-4.5342   Min.   :-4.34890   Min.   :-6.3714  
##  1st Qu.:-0.8451   1st Qu.:-0.7050   1st Qu.:-0.81078   1st Qu.:-0.7069  
##  Median : 0.0000   Median : 0.0000   Median : 0.00000   Median : 0.0000  
##  Mean   : 0.2287   Mean   : 0.1419   Mean   : 0.08206   Mean   : 0.1797  
##  3rd Qu.: 1.0902   3rd Qu.: 0.8735   3rd Qu.: 0.87585   3rd Qu.: 0.8516  
##  Max.   :16.5071   Max.   : 7.1838   Max.   : 5.44480   Max.   : 8.7851  
##       SM13                SM14                SM15               SM16          
##  Min.   :-5.78050   Min.   :-4.26320   Min.   :-6.8753   Min.   :-4.23960  
##  1st Qu.:-0.93907   1st Qu.:-0.72110   1st Qu.:-1.3974   1st Qu.:-0.79673  
##  Median : 0.00000   Median : 0.00000   Median : 0.0000   Median : 0.00000  
##  Mean   : 0.02108   Mean   : 0.08923   Mean   : 0.1794   Mean   : 0.08468  
##  3rd Qu.: 0.97935   3rd Qu.: 0.85777   3rd Qu.: 1.2862   3rd Qu.: 0.80065  
##  Max.   : 6.28430   Max.   : 6.45730   Max.   :13.5269   Max.   :10.58690  
##       SM17                SM18                SM19                SM20        
##  Min.   :-4.49020   Min.   :-4.55940   Min.   :-8.27950   Min.   :-4.8312  
##  1st Qu.:-0.66610   1st Qu.:-0.85395   1st Qu.:-0.83808   1st Qu.:-0.6844  
##  Median : 0.00000   Median : 0.00000   Median : 0.00000   Median : 0.0000  
##  Mean   : 0.06334   Mean   :-0.03859   Mean   : 0.08926   Mean   : 0.1546  
##  3rd Qu.: 0.74257   3rd Qu.: 0.72903   3rd Qu.: 1.06445   3rd Qu.: 0.9878  
##  Max.   : 7.19390   Max.   : 8.36300   Max.   :11.27570   Max.   : 5.2758  
##       SM21               SM22                SM23               SM24          
##  Min.   :-5.1418   Min.   :-5.395400   Min.   :-4.7342   Min.   :-6.06530  
##  1st Qu.:-0.7453   1st Qu.:-0.879075   1st Qu.:-0.6957   1st Qu.:-0.74760  
##  Median : 0.0000   Median : 0.000000   Median : 0.0000   Median : 0.00000  
##  Mean   : 0.1015   Mean   : 0.005579   Mean   : 0.1109   Mean   : 0.05064  
##  3rd Qu.: 0.8068   3rd Qu.: 0.973850   3rd Qu.: 0.7637   3rd Qu.: 0.80470  
##  Max.   : 6.9130   Max.   : 5.064400   Max.   : 9.0250   Max.   : 6.98910  
##       SM25               SM26                SM27                SM28          
##  Min.   :-5.9426   Min.   :-4.28790   Min.   :-6.67120   Min.   :-5.80050  
##  1st Qu.:-0.5648   1st Qu.:-0.71095   1st Qu.:-0.90535   1st Qu.:-0.84965  
##  Median : 0.0000   Median : 0.00000   Median : 0.00000   Median : 0.00000  
##  Mean   : 0.0305   Mean   :-0.01985   Mean   : 0.01604   Mean   :-0.02022  
##  3rd Qu.: 0.6879   3rd Qu.: 0.65495   3rd Qu.: 0.83993   3rd Qu.: 0.76935  
##  Max.   : 5.9432   Max.   : 5.33110   Max.   : 8.82470   Max.   : 7.15750  
##       SM29                SM30               SM31                SM32          
##  Min.   :-5.43320   Min.   :-4.2593   Min.   :-5.89830   Min.   :-4.97280  
##  1st Qu.:-0.73288   1st Qu.:-0.6446   1st Qu.:-0.66585   1st Qu.:-0.97962  
##  Median : 0.00000   Median : 0.0000   Median : 0.00000   Median : 0.00000  
```

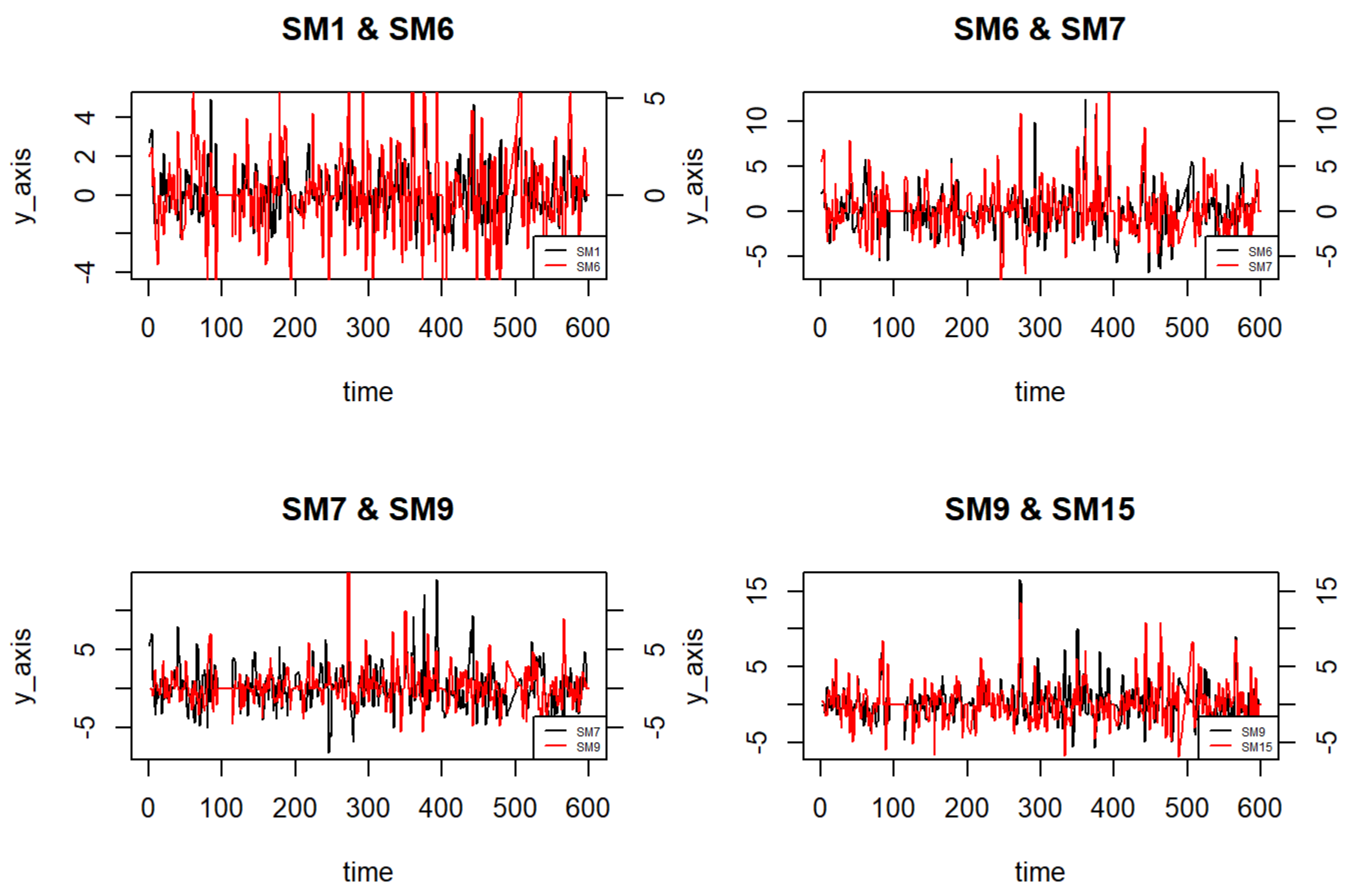
```
##  Mean   :-0.03549   Mean   : 0.0323   Mean   : 0.04033   Mean   : 0.01196
##  3rd Qu.: 0.58165   3rd Qu.: 0.6589   3rd Qu.: 0.68780   3rd Qu.: 0.95103
##  Max.   : 4.77690   Max.   : 5.2447   Max.   : 4.55700   Max.   : 5.93040
```

3. Paired lineplots

Below are the paired lineplots of SM6, SM7, SM9, SM15.

While SM1 represents the a “normal behaving” node, SM6, SM7, SM9 and SM15 are those with the larget value around the peak. One thing to notice is that SM7 and SM9 tend to move to the opposite direction around the peak time.

```
par(mfrow=c(2,2))
colList1 <- c(1, 6, 7, 9)
colList2 <- c(6, 7, 9, 15)
nameList <- names(data_15may2020)

for(i in 1:4) {
        plot(data_15may2020[,colList1[i]],type = "l",
             xlab = "time", ylab = "y_axis")
        lines(data_15may2020[,colList2[i]], col = "red")
        axis(side = 4, at = pretty(data_15may2020[,colList2[i]]),
             ylab=names(data_15may2020)[colList2[i]])
        legend(x = "bottomright",
               legend = c(nameList[colList1[i]], nameList[colList2[i]]),
               col = c("black", "red"),
               cex=0.45,
               lty = c(1, 1))
        title(main=paste(nameList[colList1[i]], "&", nameList[colList2[i]]))
}
```

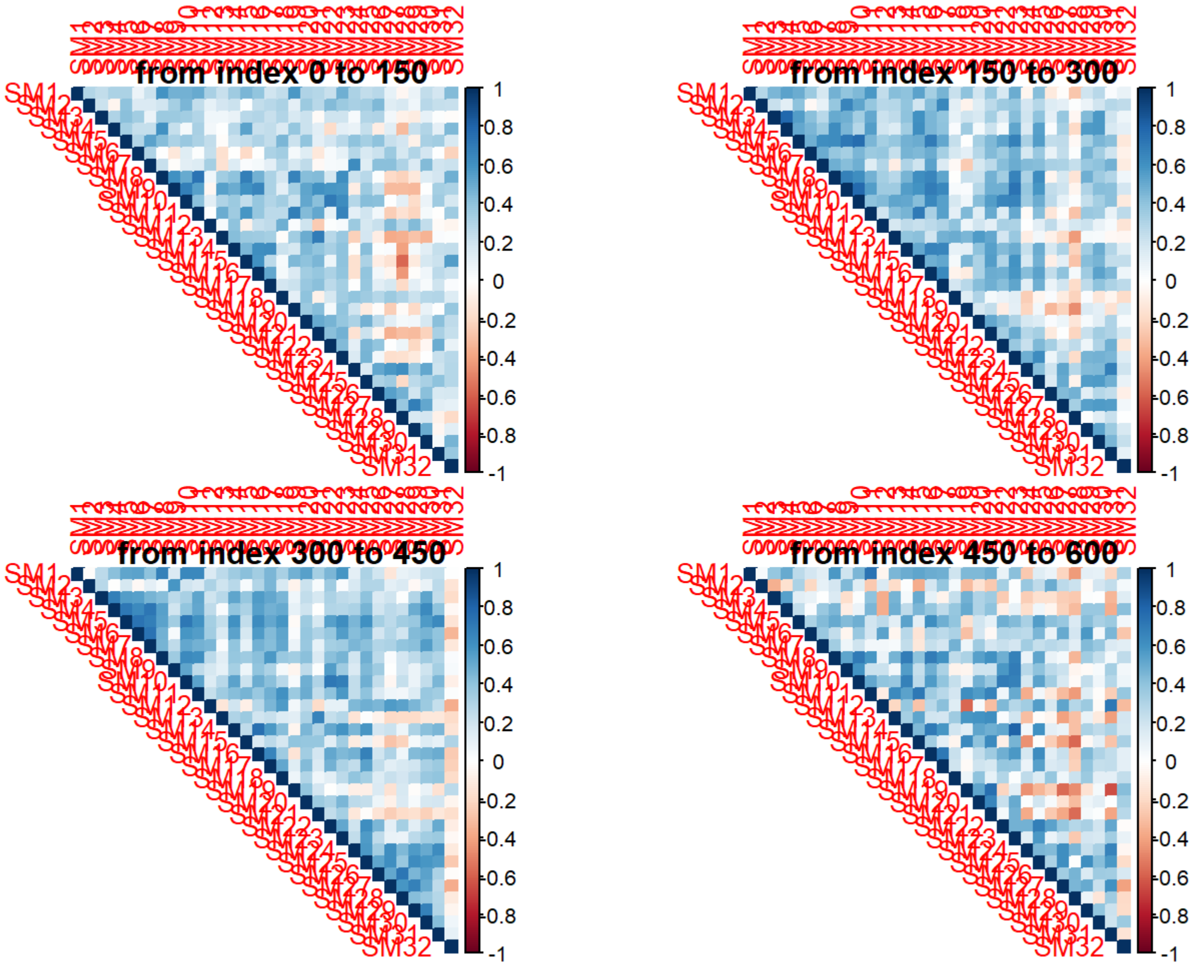


4. Correlation of Raw data

This correlation heatmap is not very useful, because correlation on high dimensional space is not only hard to interpret but also missing a lot of information. I kept it here just for future references.

Below are the 4 correlation heatmaps of the time-series data of the 32 nodes divided into 4 time range. We can see the correlation between each pair of nodes are changing from time to time. It again, showing us the correlation analysis on high dimensional time series is not very useful.

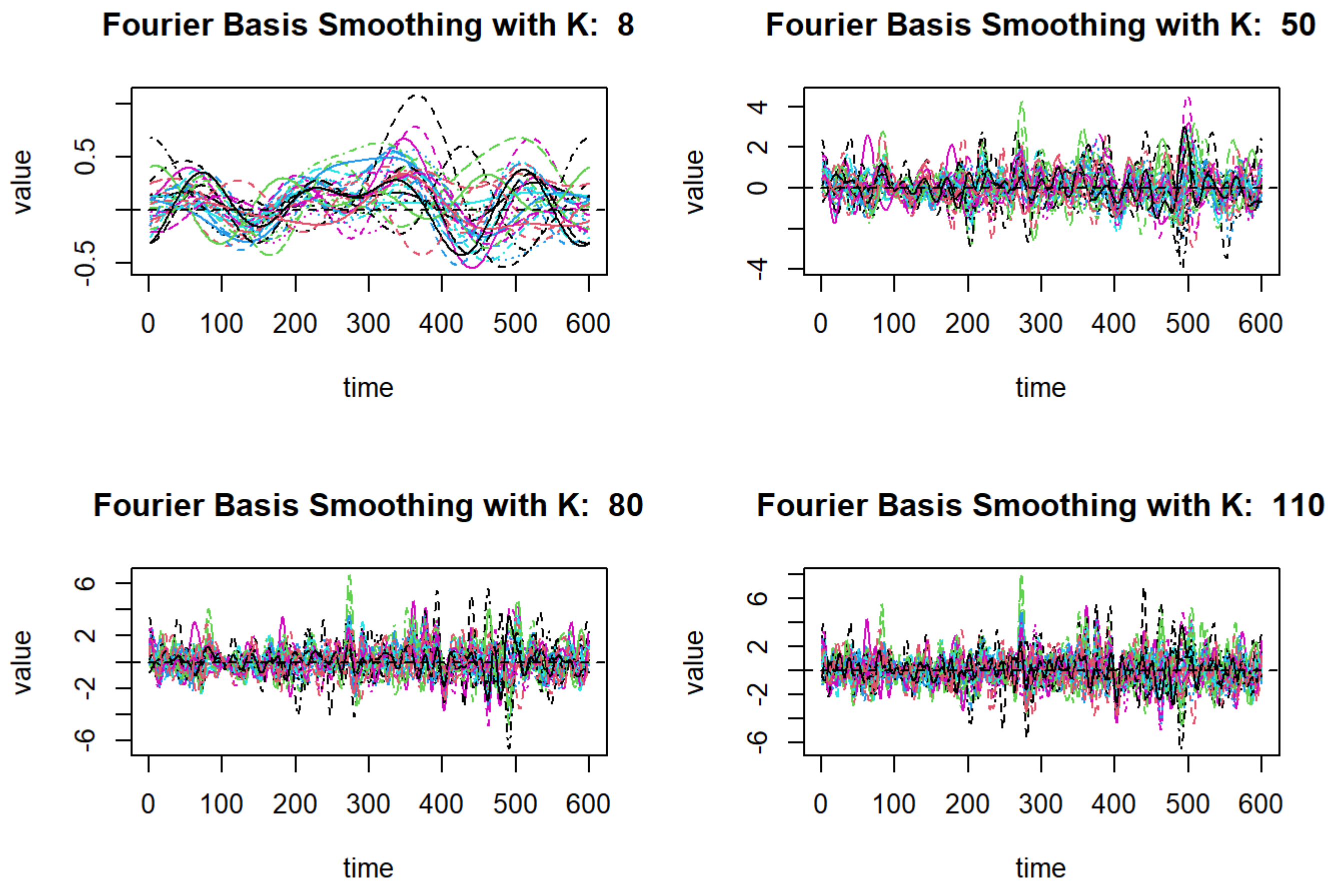
```
par(mfrow=c(2,2))
for (i in 0:3){
        cor_mat <- cor(z[(i*150):(i*150+150),])
        #corrplot(cor_mat, method = "color", type = "upper", order = "hclust")
        corrplot(cor_mat, method = "color", type = "upper")
        title(main = paste("from index", i*150, "to", i*150+150))
}
```



5. Fourier Smoothing of fMRI data

I applied *Fourier basis function* to smooth the data of a 600x32 time-series matrix, because according to the visualization of the data I assume the data is periodic since they are all oscilating around zero. Below I created four smoothing curve plots, each with the respective K value 8, 50, 80,110 indicating how many basis functions were used in the *Fourier Basis system*.

```
time <- (1:600)
data_mat <- as.matrix(data_15may2020)
kList <- c(8, 50, 80, 110)
par(mfrow=c(2,2))

for(i in 1:4) {
   basis <- create.fourier.basis(c(1,600), kList[i])
   smoothfd <- smooth.basis(time, data_mat, basis)$fd
   plot(smoothfd)
   title(main=paste("Fourier Basis Smoothing with K: ", kList[i]))
}
```



6. mean GCV and SSE of Fourier Basis Smoothing with different K

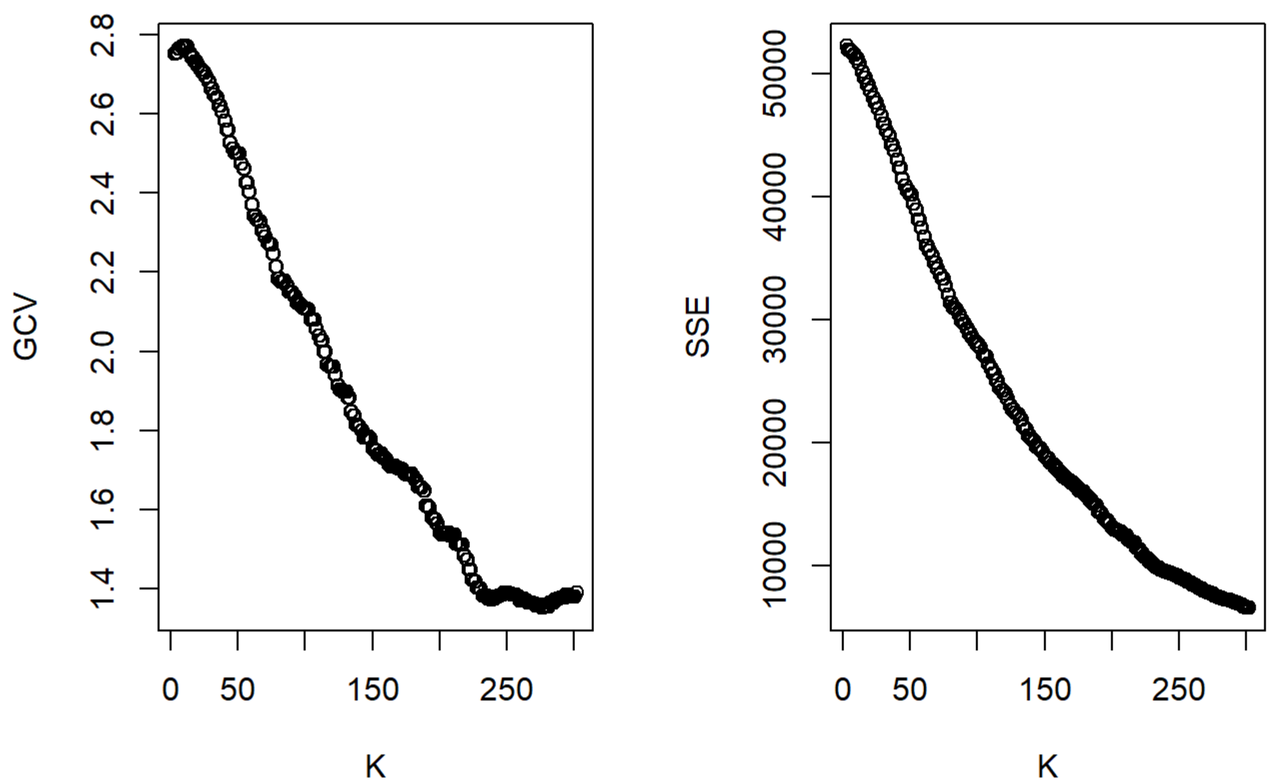
Below the code iterates the *fourier basis function* for different *K* values, and plot the corresponding *Generalized Cross-validation* (*gcv*) and the *Sum of Squared Error* (*SSE*) measure on the y-axis. *gcv* is calculated using the criterion,

$$\frac{n}{n - df(\lambda)} \frac{SSE}{n - df(\lambda)}$$

This metrics is designed to twice-discounted for the degree of freedom in the basis functions. It is useful because we are increasing the number of basis function at each step. The result of *gcv* is a 300 x 32 matrix (because I limited the choice of K value to be no larger than 303, and the K=1,2 is not accepted by the algorithm in this case). I then plotted the mean value of the 32 columns. *SSE* is the frequently used *sum of squared errors* metrics in many statistical analysis,

$$SSE = \sum_{j}^{n} [y_j - x(t_j)]^2$$

```
smoothK.unwrapped = matrix(0, 300, 3)
colnames(smoothK.unwrapped) = c('k', 'gcv', 'sse')
kList <- c(3:303)

for(row in 1:300) {
        basis <- create.fourier.basis(c(1,600), kList[row])
        smoothList <- smooth.basis(time, data_mat, basis)
        smoothK.unwrapped[row, 1] = kList[row]
        smoothK.unwrapped[row, 2] = mean(smoothList$gcv)
        smoothK.unwrapped[row, 3] = smoothList$SSE
}
par(mfrow=c(1,2))
plot(smoothK.unwrapped[,1], smoothK.unwrapped[,2], xlab='K', ylab='GCV')
plot(smoothK.unwrapped[,1], smoothK.unwrapped[,3], xlab='K', ylab='SSE')
```



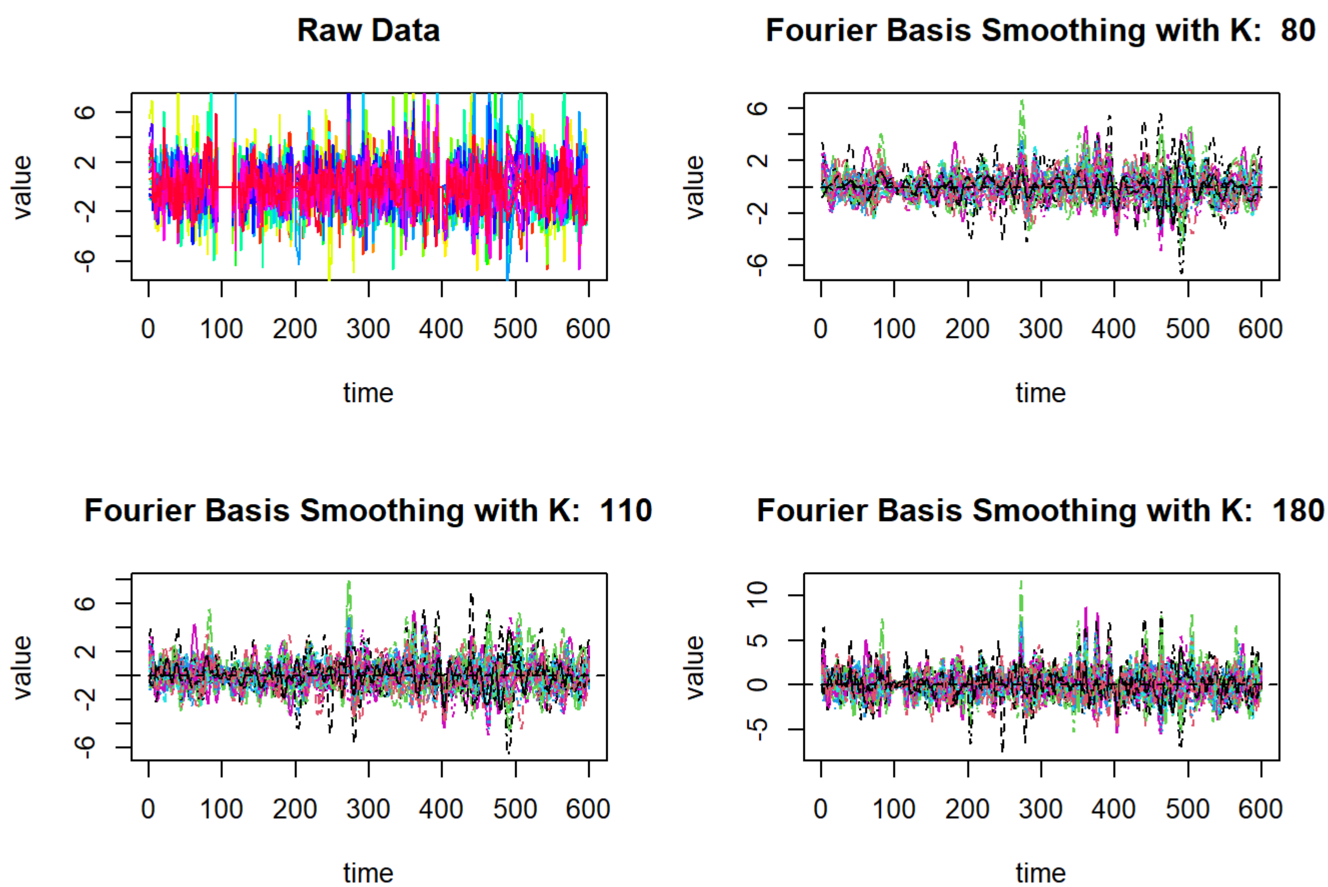
From the two error measure curves, we can see the error measures kept decreasing as K became larger. It is not surprising if we pick the number large enough, the curves will fit the raw data almost perfectly. But we will be going exact the opposite way of what we are trying to acheive. We are hoping to smooth the data, not to model it with more complexity. Thus, what is the optimal value of K still requires more investigations. According to "*Elbow Method*", we might want to consider picking K equals to around 180, but I am not sure at this point.

We can see the speed of error decreasing slows down a bit around K=80, so that's why I have plotted the smoothed curves with K=80 in part 5 above. What I have found is, after K is greater than certain threshold value, keeping increasing the K value will not change the overall shape of smoothed curve too much, but instead, the amplitude of the peak values will be captured as a larger value (see plot in section 5 around t = 300 and K = 80 & 110).

In the next command cell, I again, plotted the the raw data, and compare it with the smoothed data using fourier basis function with K=80 & 110 & 180

7. Plot of raw data and smoothed data using fourier basis function with K=80 & 110 & 180

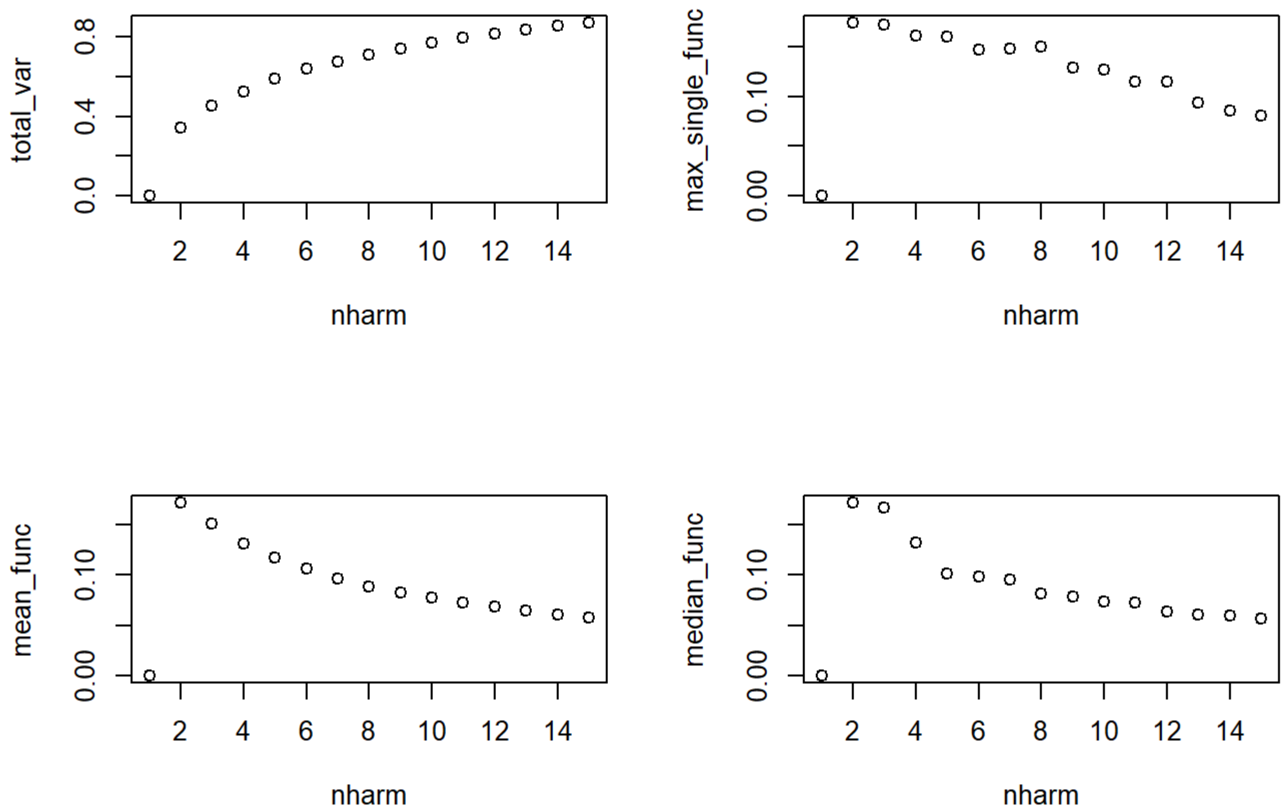
```
par(mfrow=c(2,2))

plot(index(data_15may2020),data_15may2020[,1], type = 'l', ylim = c(-7,7),
     xlab='time', ylab='value')
title(main=paste("Raw Data"))
cl<-rainbow(32)
for (i in 2:32){
        lines(index(data_15may2020),data_15may2020[,i], col=cl[i])
}

for(i in c(80,110,180)) {
        basis <- create.fourier.basis(c(1,600), i)
        smoothfd <- smooth.basis(time, data_mat, basis)$fd
        plot(smoothfd)
        title(main=paste("Fourier Basis Smoothing with K: ", i))
}
```



8. fPCA_1

loop for differnt choice of nharm (# of Priciple Components) of a fixed smoothfd

```
basis <- create.fourier.basis(c(1,600), 280)
time <- (1:600)
data_mat <- as.matrix(data_15may2020)
smoothfd <- smooth.basis(time, data_mat, basis)$fd

N = c(1:15)
pcalist.sumvar = matrix(0, length(N), 4)
colnames(pcalist.sumvar) = c('total_var_explained', 'max_single_func'
                             ,'mean_func', 'median_func')

for(row in 2:length(N)) {
  pcalist = pca.fd(smoothfd, nharm=row, harmfdPar=fdPar(smoothfd))
  rotpcalist = varmx.pca.fd(pcalist)
  pcalist.sumvar[row,1] = sum(rotpcalist$varprop)
  pcalist.sumvar[row,2] = max(rotpcalist$varprop)
  pcalist.sumvar[row,3] = mean(rotpcalist$varprop)
  pcalist.sumvar[row,4] = median(rotpcalist$varprop)
}
par(mfrow=c(2,2))
plot(N, pcalist.sumvar[,1], xlab='nharm', ylab='total_var')
plot(N, pcalist.sumvar[,2], xlab='nharm', ylab='max_single_func')
plot(N, pcalist.sumvar[,3], xlab='nharm', ylab='mean_func')
plot(N, pcalist.sumvar[,4], xlab='nharm', ylab='median_func')
```

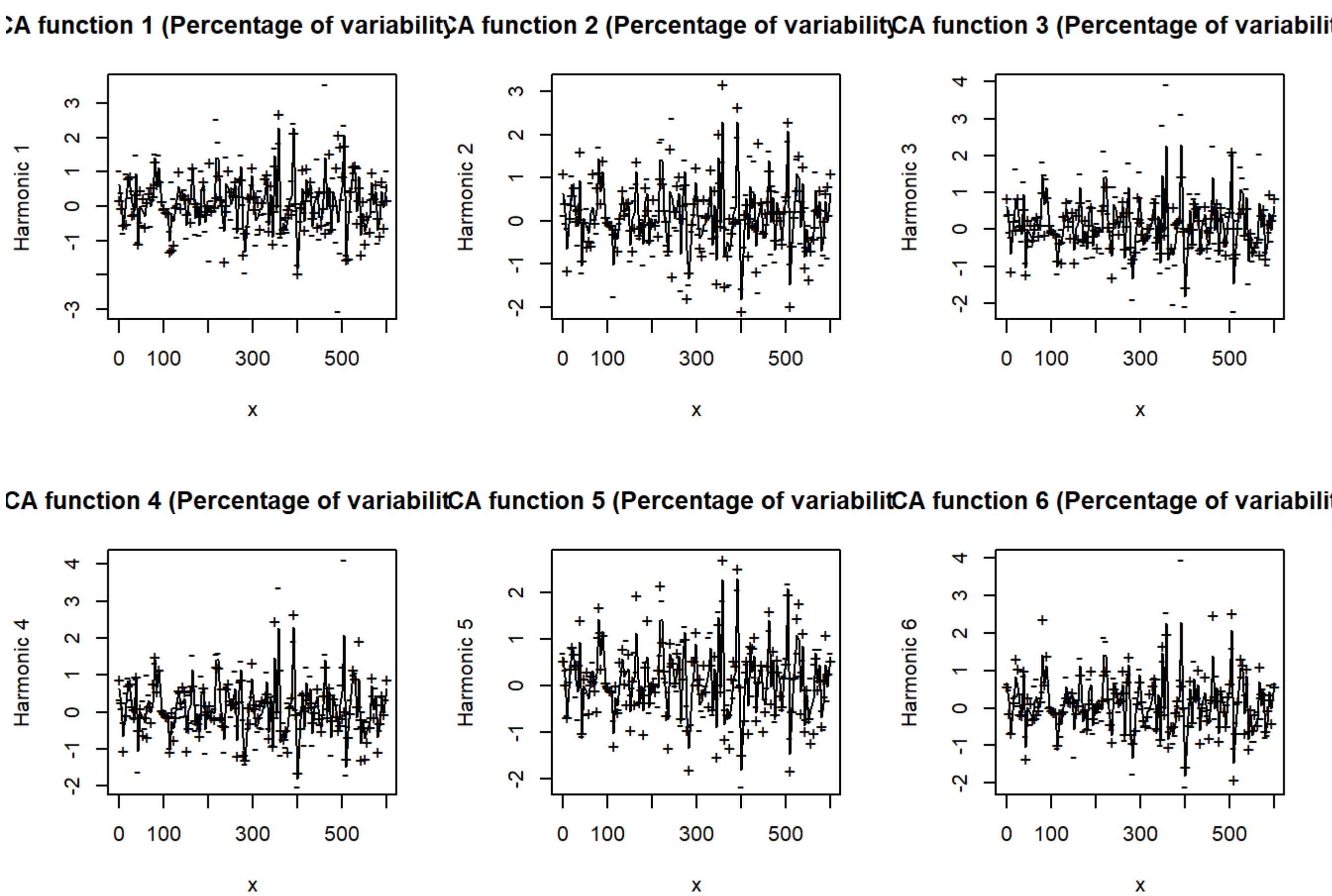


9. fPCA_2

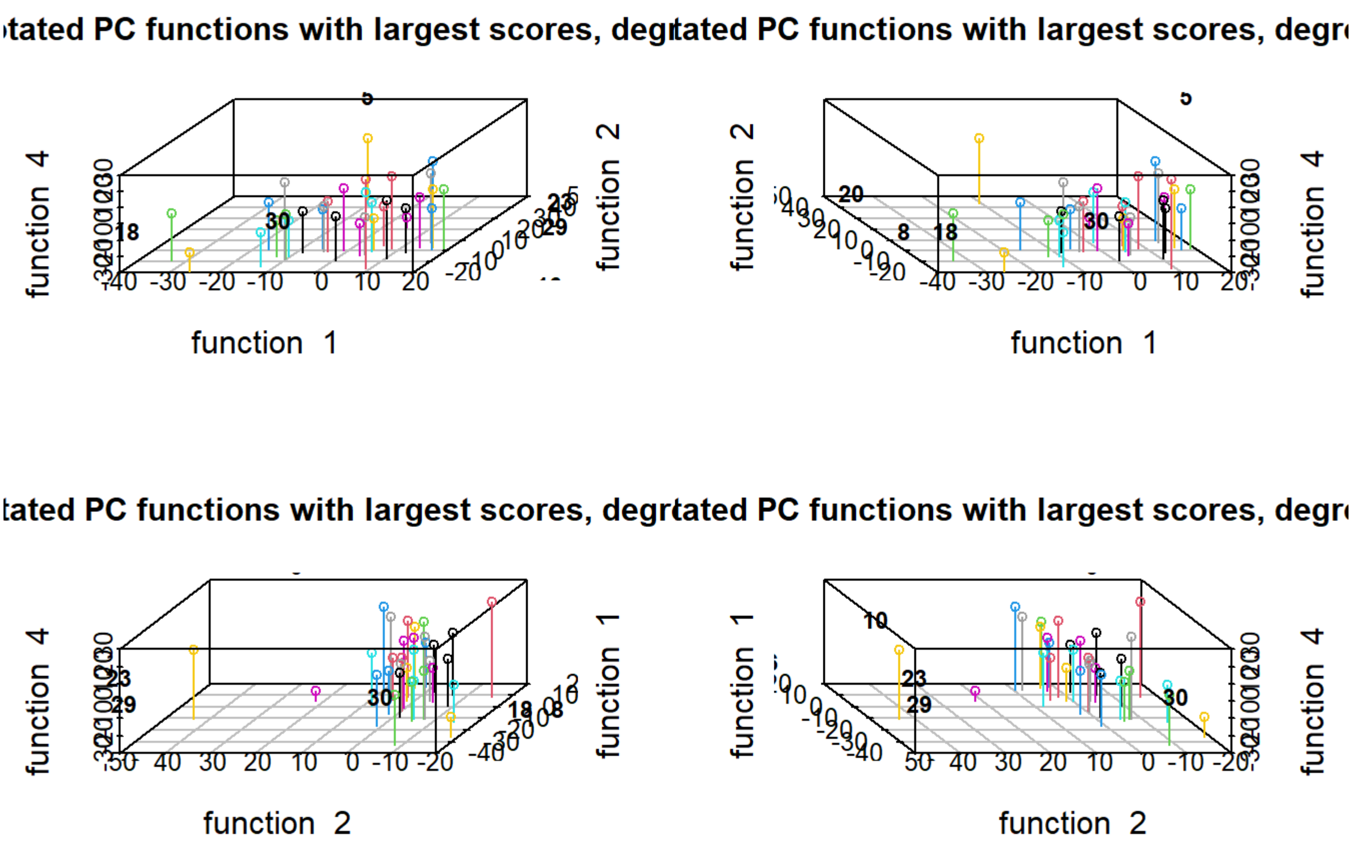
3D plot of top 3 Principle component:

3D scatter plot is created by specifying narhm = 6 with total variance covered about 65%, and picking the top 3 PCs (10.5%, 15.03%, 10.3% respectively) to draw.

```
pcalist = pca.fd(smoothfd, nharm=6, harmfdPar=fdPar(smoothfd))
rotpcalist = varmx.pca.fd(pcalist)
par(mfrow=c(2,3))
plot.pca.fd(rotpcalist)
```



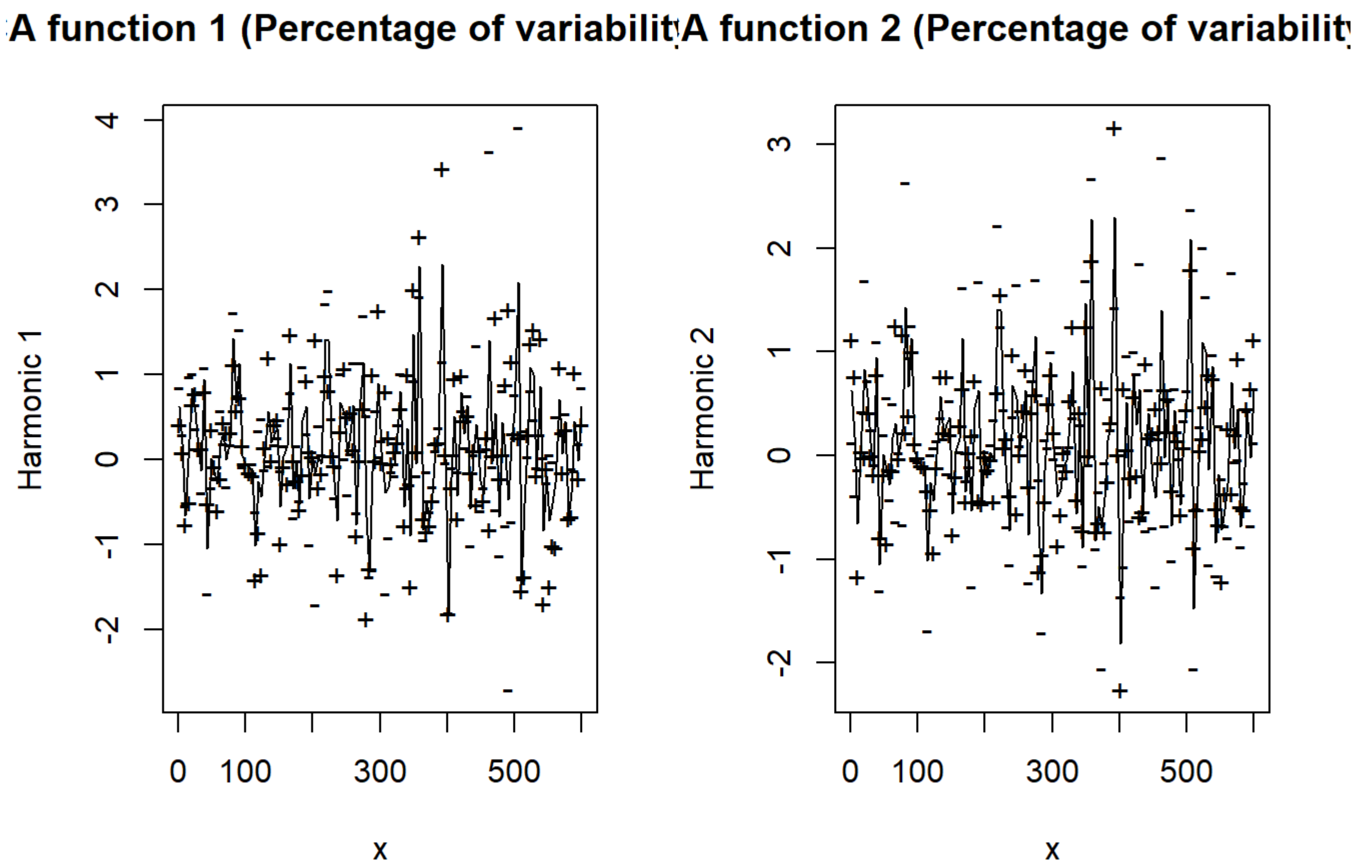

```
colvar <- rotpcalist$varprop
colScores <- rotpcalist$scores
lst <- sort(colvar, index.return=TRUE, decreasing=TRUE)
topidx <- lapply(lst, `[`, lst$x %in% head(unique(lst$x),3))$ix

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 PC functions with largest scores, degree = ", deg))
  text(colScores[, topidx[2]]~colScores[,topidx[1]],labels=index(colScores),data=colScores, cex=
0.9, font=2)
}
```



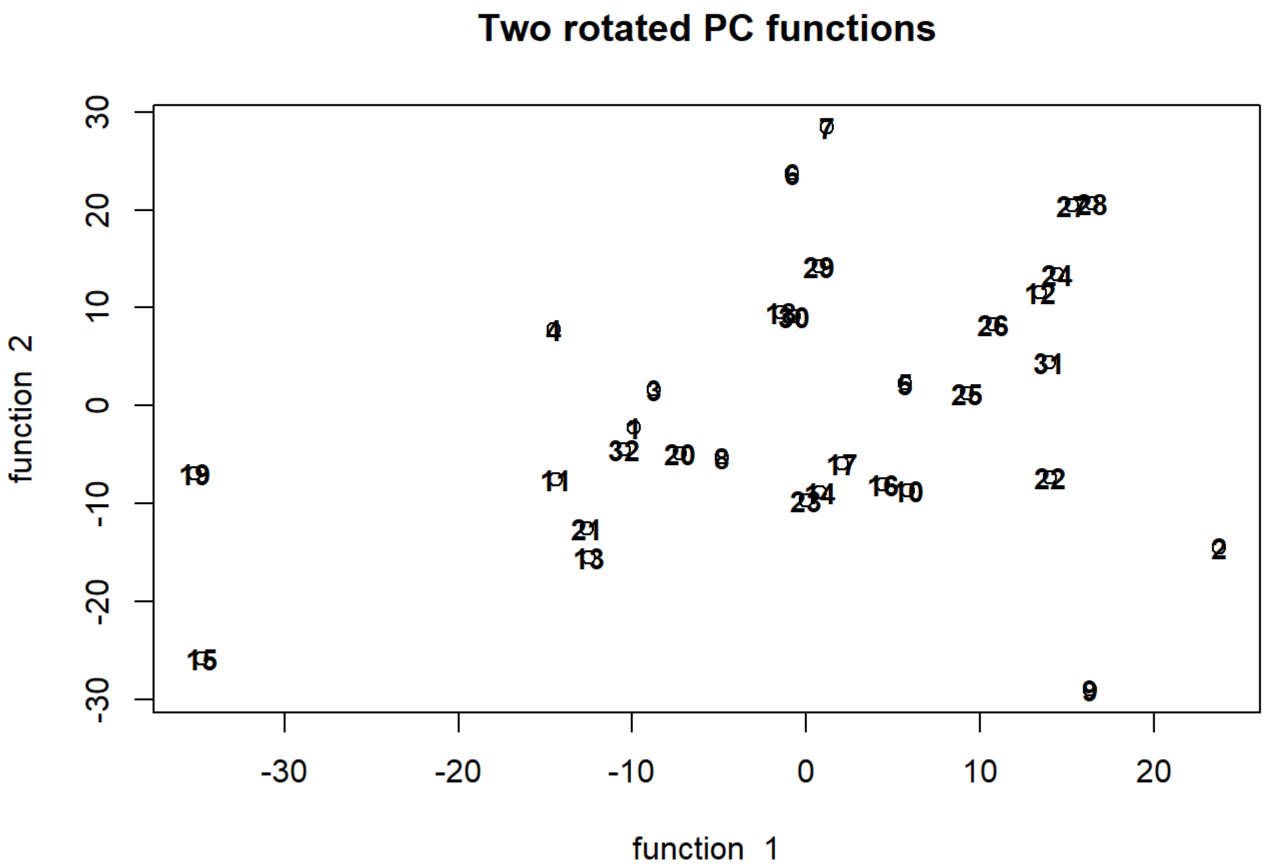
10. fPCA_3

The 2D scatter plot is created by specifying nharm=2 (# of fPCAs).

```
pcalist = pca.fd(smoothfd, nharm=2, harmfdPar=fdPar(smoothfd))
rotpcalist = varmx.pca.fd(pcalist)
par(mfrow=c(1,2))
plot.pca.fd(rotpcalist)
```




```
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 PC functions"))
text(colScores[, topidx[2]]~colScores[,topidx[1]],labels=index(colScores),data=colScores, cex=0.
9, font=2)
```


Two rotated PC functions
function 2
function 1
30
20
10
0
-10
-20
-30
-30
-20
-10
0
10
20