20july2020

Zach Wang

7/20/2020

1. progress summary

This week, as we discussed from last meeting, I have: - 1. selected the 1st complete sin curve of every node and align them to register at the origin. - 2. Scale the the time frame of each node to [-1,1] while time=0 remain unchanged. I also make sure that each node have the same number of rows so that fPCA will be able to apply. - 3. Apply fPCA to the new dataset of curves (1st cycle of each node), and the result is that the first two Principle Component functions covers 91.5% of total variance. - 4. repeat the process above for 2nd cycle of each node, and first two Principle component functions covered in total 93.7% of total variance.

2. read data and attach packages

3. Defined fourier smoothing functions

To study a single brain node response, specify the node number in the *node_subset* list.

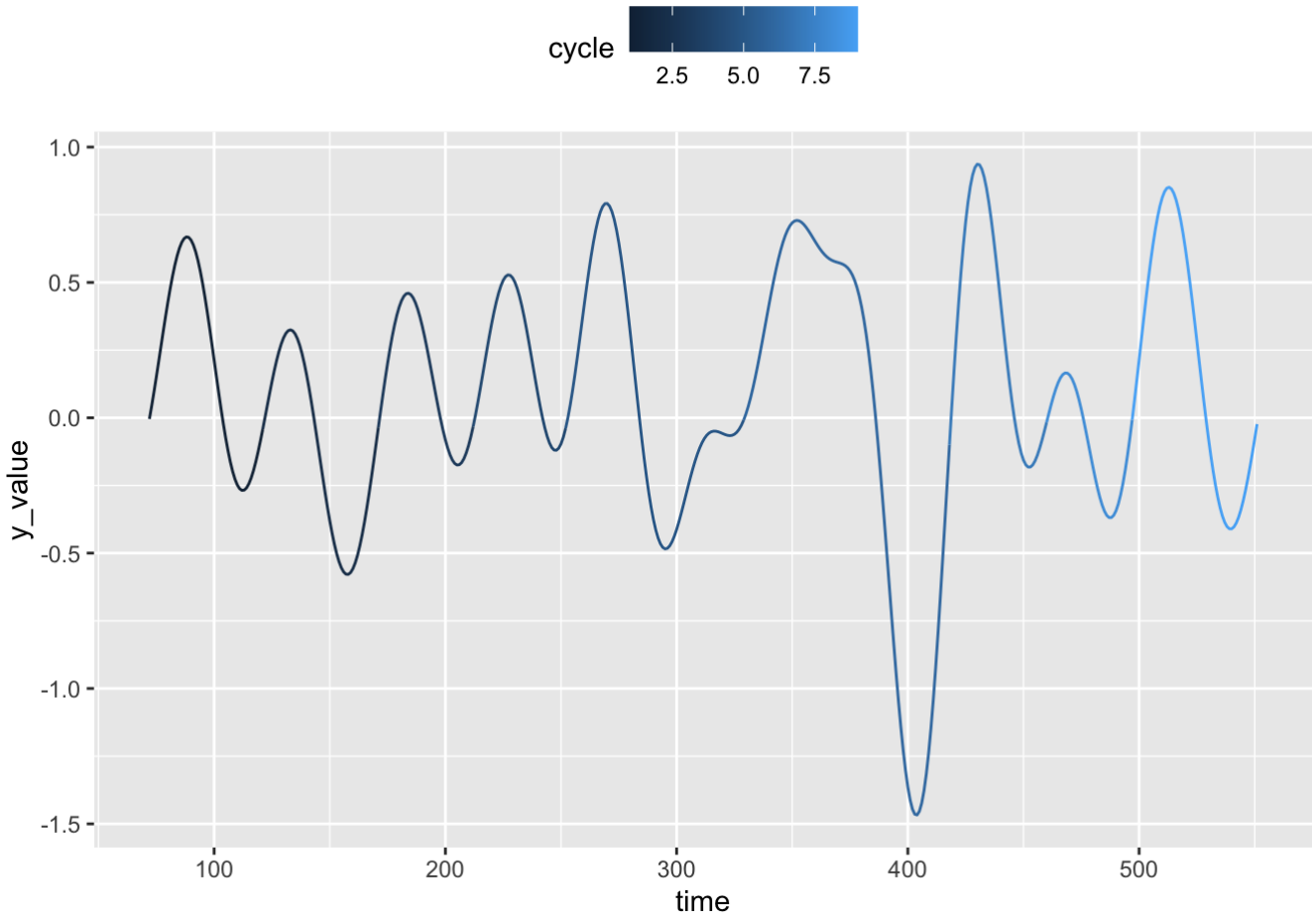
```
f_fourier_smooth <- function(time_subset, data_mat, node_subset, k){
  basis <- create.fourier.basis(c(time_subset[1],time_subset[length(time_subset)]), k)
  fd_obj <- smooth.basis(time_subset, data_mat[time_subset,node_subset], basis)
  smoothfd <- fd_obj$fd
  #plot(smoothfd)
  #title(main=paste("Fourier Basis Smoothing of node:", node_subset, ", Basis_number:",k
))
  return(fd_obj)
}
```

4. define the function to extract periodic cycle of a single node response

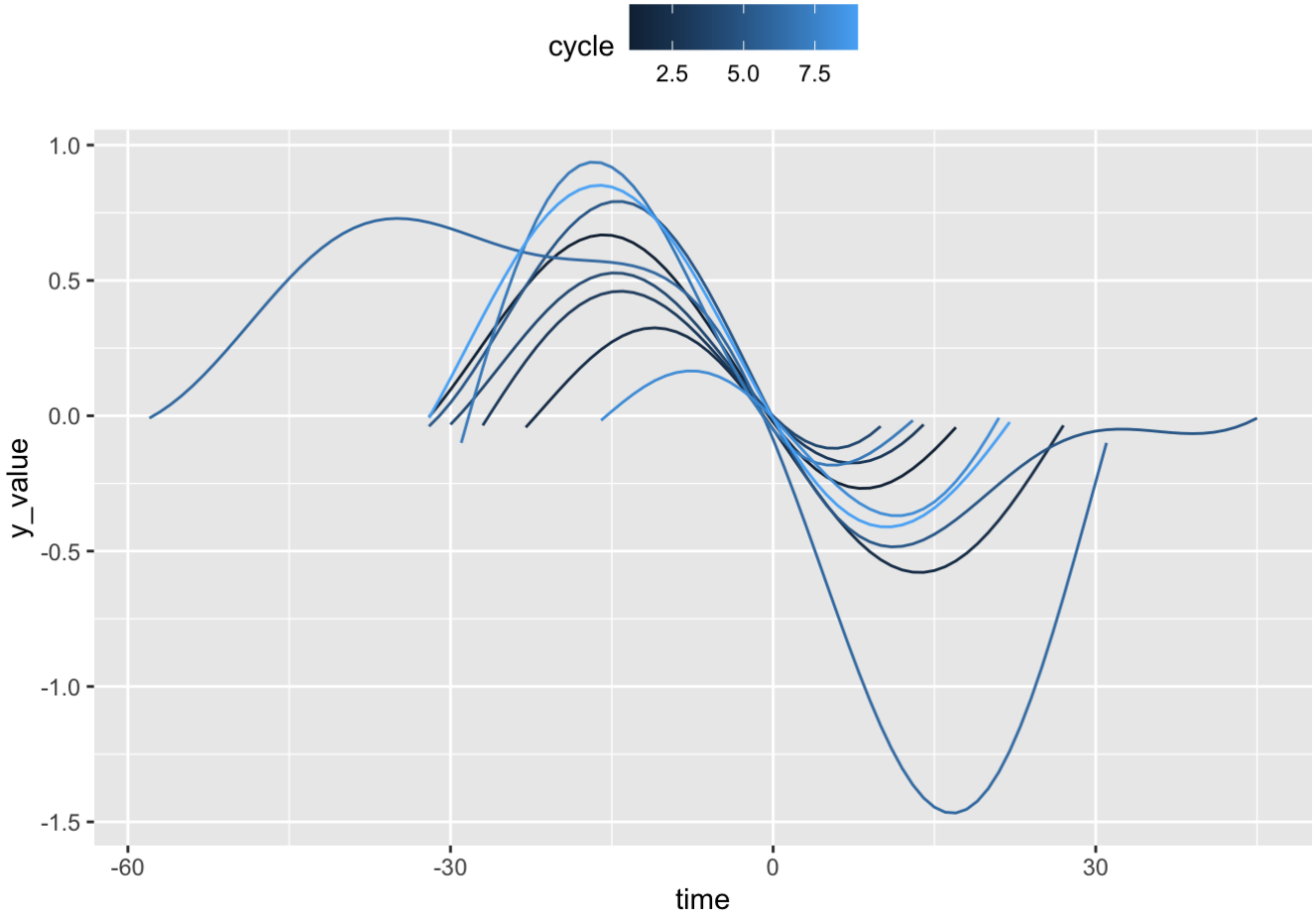
```
transform.Cycle = function(data, register){
  # obtain index at which curve crosses 0
  #crossed 0---> -1: pos to neg,    1: neg to pos
  #returns: location index where curve crosses X-axis
  x=diff(ifelse(data>0,1,0))
  z_idx=(1:599)[x!=0]

  # skip first crossing if it is from positive to negative
  if (x[z_idx[1]]==-1){
    z_idx=z_idx[-1]
  }

  #put every complete cycle in a Dataframe
  i=1
  cl=1
  result=data.frame(cycle=integer(), time=integer(), y_value=integer())
  while (i+2<=length(z_idx)){

      if(register==0){
        tmp=data.frame(cycle=cl, time=seq(z_idx[i],z_idx[i+2]), y_value=smoothed_curve[z
_idx[i]:z_idx[i+2]])
      }
      else{
        tmp=data.frame(cycle=cl, time=seq(1,length(seq(z_idx[i],z_idx[i+2]))), y_value=s
moothed_curve[z_idx[i]:z_idx[i+2]])
        tmp[,2]=tmp[,2]-length(seq(z_idx[i],z_idx[i+1]))-1
      }
      result=rbind(result,tmp)
      i=i+2
      cl=cl+1
    }
  return(result)
}
```

5. Oversmooth with basis functions

```
# OverSmoothed, k=32

## register: 0 --> plot data on the original timeline
##           1 --> register every complete sinusoidal curve starting at 0;


result_obj <- f_fourier_smooth(time_subset=c(1:600), data_mat, node_subset=c(1), k=32)
smoothed_curve = eval.fd(c(1:600),result_obj$fd)

transformed_node1 = transform.Cycle(smoothed_curve, register=0)
ggplot(transformed_node1, aes(time, y_value,group=cycle, colour=cycle)) + geom_line() +
 theme(legend.position="top")
```

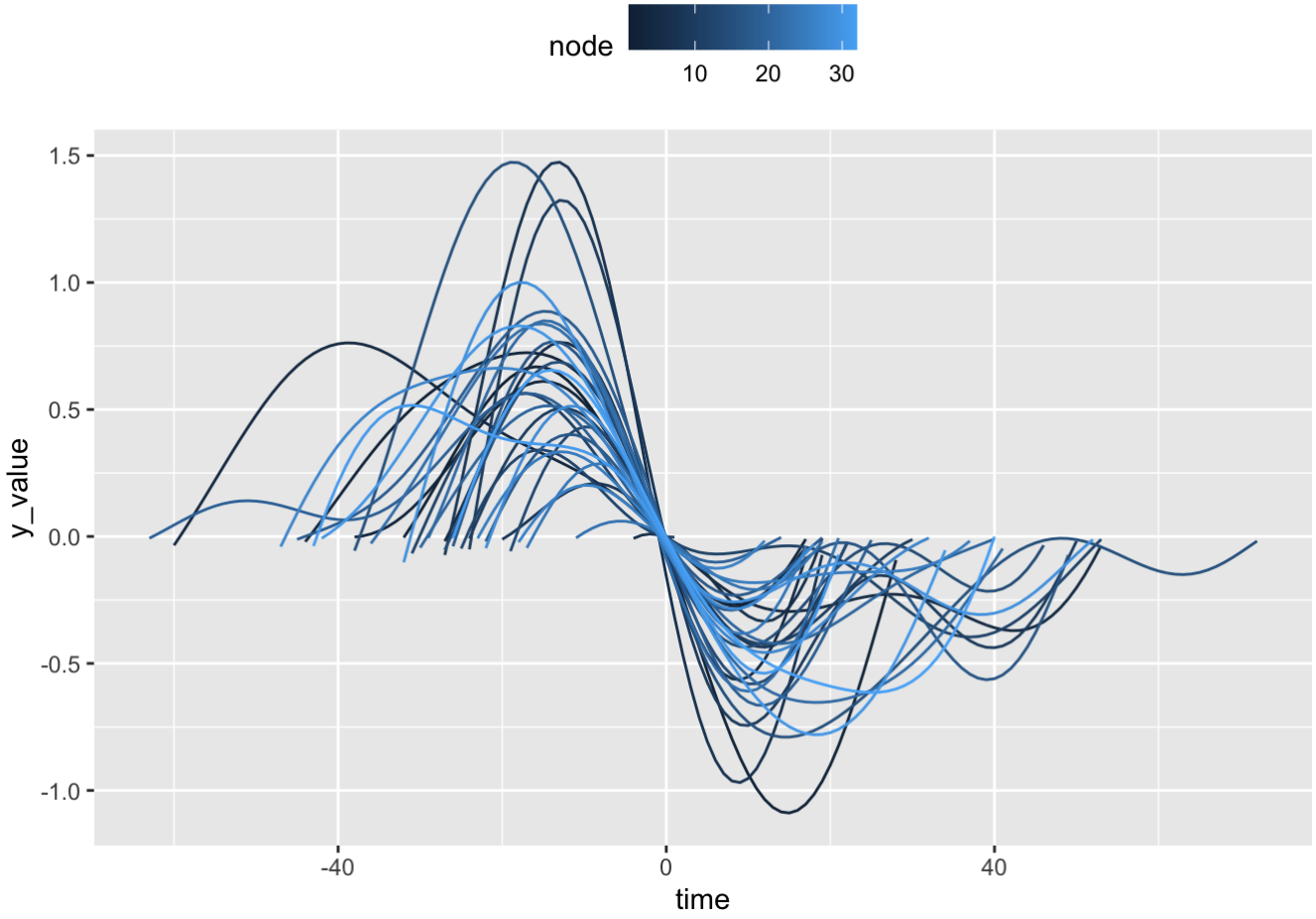



```
transformed_node1 = transform.Cycle(smoothed_curve, register=1)
ggplot(transformed_node1, aes(time, y_value,group=cycle, colour=cycle)) + geom_line() +
 theme(legend.position="top")
```



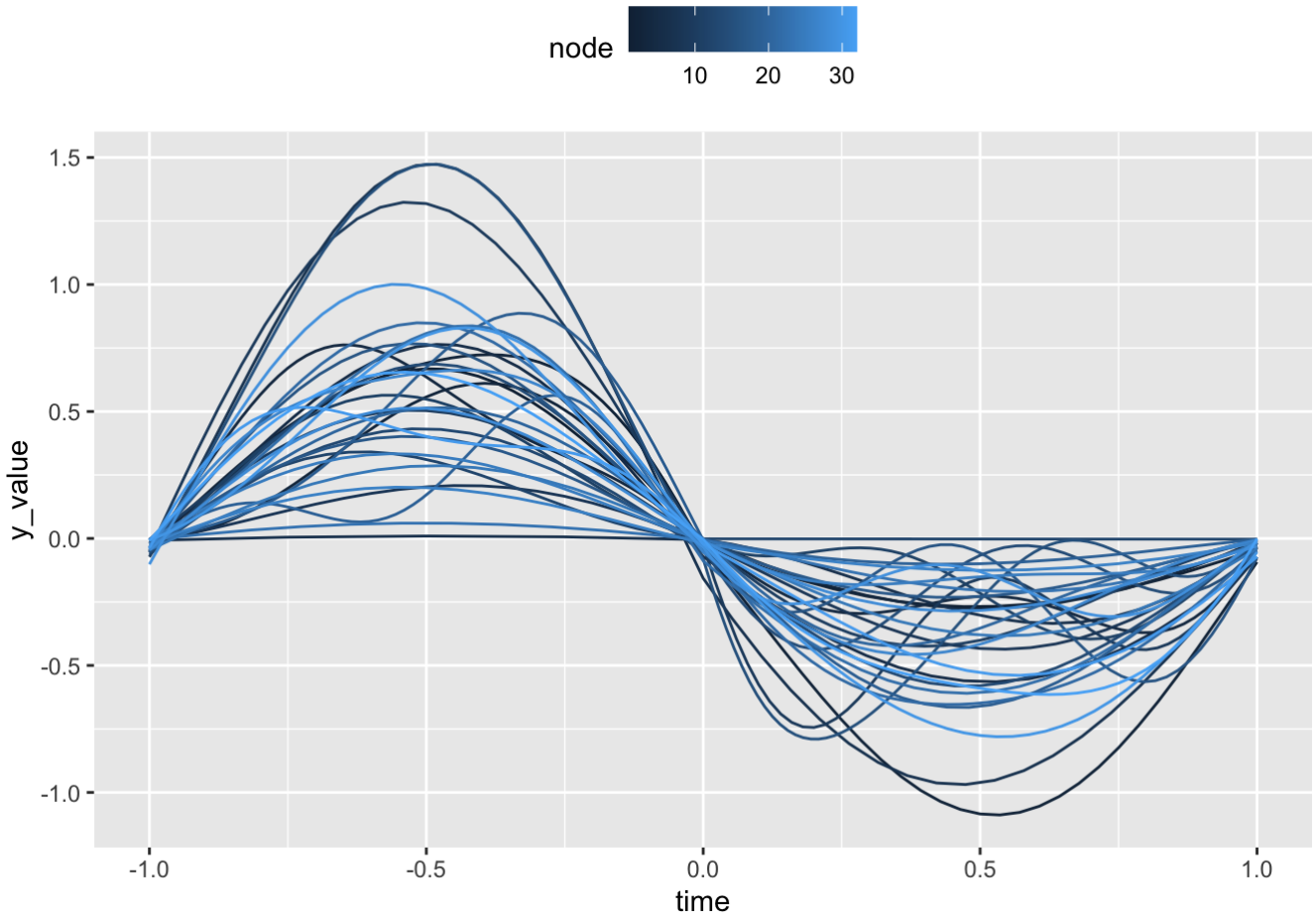
5.1 *1st* cycle across different node

```
df = data.frame(node=integer(),cycle=integer(), time=integer(), y_value=integer())
for(i in 1:32){
  result_obj <- f_fourier_smooth(time_subset=c(1:600), data_mat, node_subset=c(i), k=32)
  smoothed_curve = eval.fd(c(1:600),result_obj$fd)
  transformed_node = transform.Cycle(smoothed_curve, register=1)
  tmp = subset(transformed_node, cycle==1)
  tmp$node=i
  df=rbind(df,tmp)
}
ggplot(df, aes(time, y_value,group=node, colour=node)) + geom_line() + theme(legend.posi
tion="top")
```



5.2 Scale the timeframe of each node

```
df_tmp = data.frame(node=integer(),cycle=integer(), time=integer(), y_value=integer())
for(i in 1:32){
  tmp=subset(df, node==i)
  tmp_1=subset(tmp, time<=0)
  tmp_2=subset(tmp, time>0)
  tmp_1$time=(tmp_1$time) / (max(abs(tmp_1$time)))
  tmp_2$time=(tmp_2$time) / (max(abs(tmp_2$time)))
  tmp=rbind(tmp_1,tmp_2)
  df_tmp=rbind(df_tmp,tmp)
}
ggplot(df_tmp, aes(time, y_value,group=node, colour=node)) + geom_line() + theme(legend.
position="top")
```

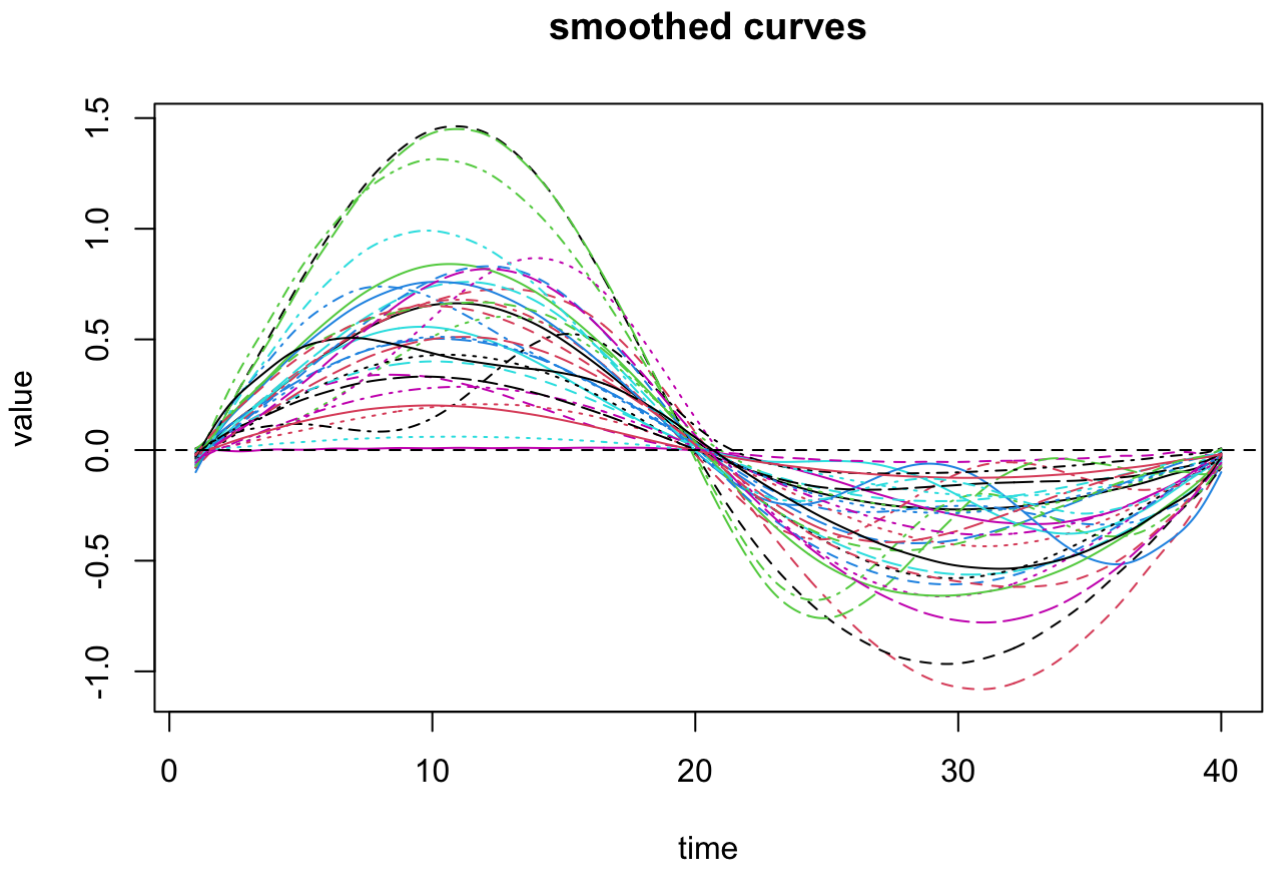


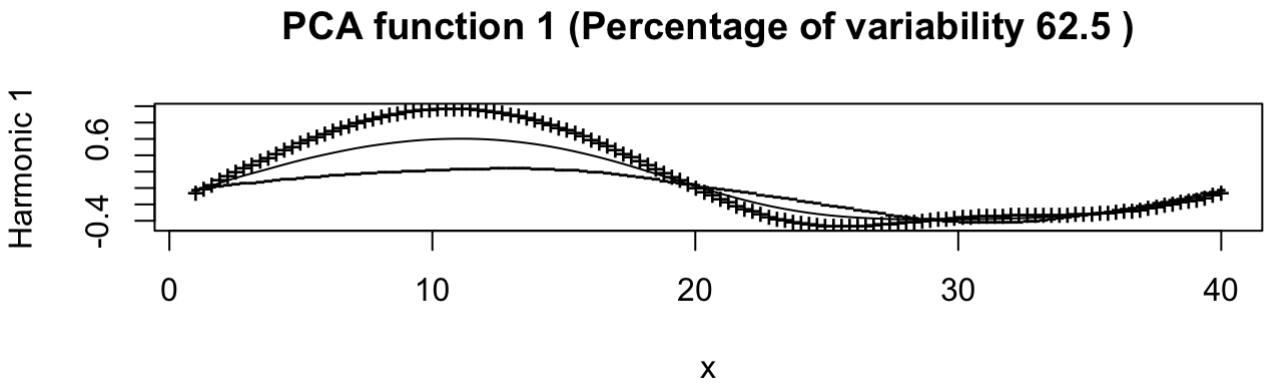
5.3 pivot the new dataframe

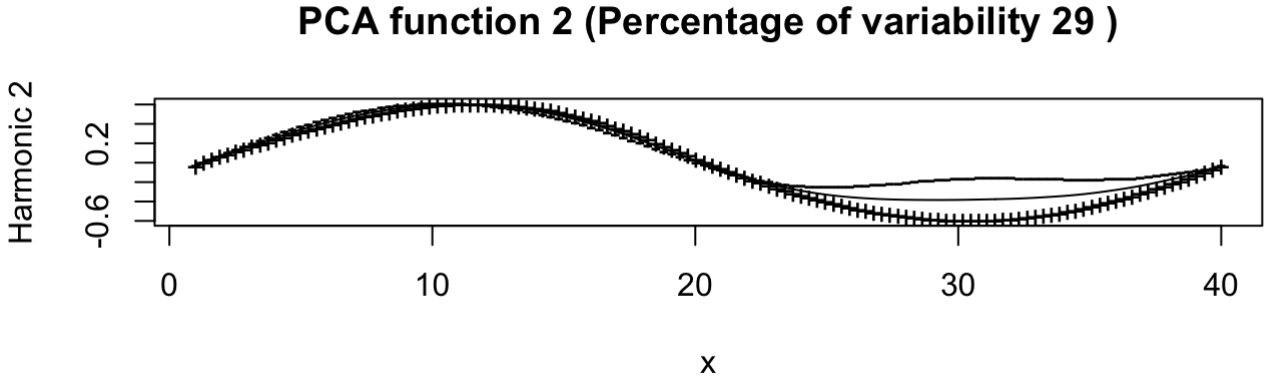
```
df_new = data.frame(matrix(nrow=40))
for(i in 1:32){
  xx=seq(-1,1,length.out=40)
  tmp=subset(df_tmp, node==i)
  s=smooth.spline(x=tmp$time, y=tmp$y_value,  df = 10)
  df_new[,ncol(df_new)+1]=predict(s,xx)$y
}
df_new=df_new[,2:33]
oldnames = colnames(df_new)
newnames = colnames(data_mat)
for(i in 1:32) names(df_new)[names(df_new) == oldnames[i]] = newnames[i]
```

5.4 fPCA on new dataframe

```
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)
}
df_new <- as.matrix(df_new)
rotpcalist = fPCA_subset(time_subset=c(1:40), df_new, node_subset = c(1:32)
                         , k=32, nharm=2)
```

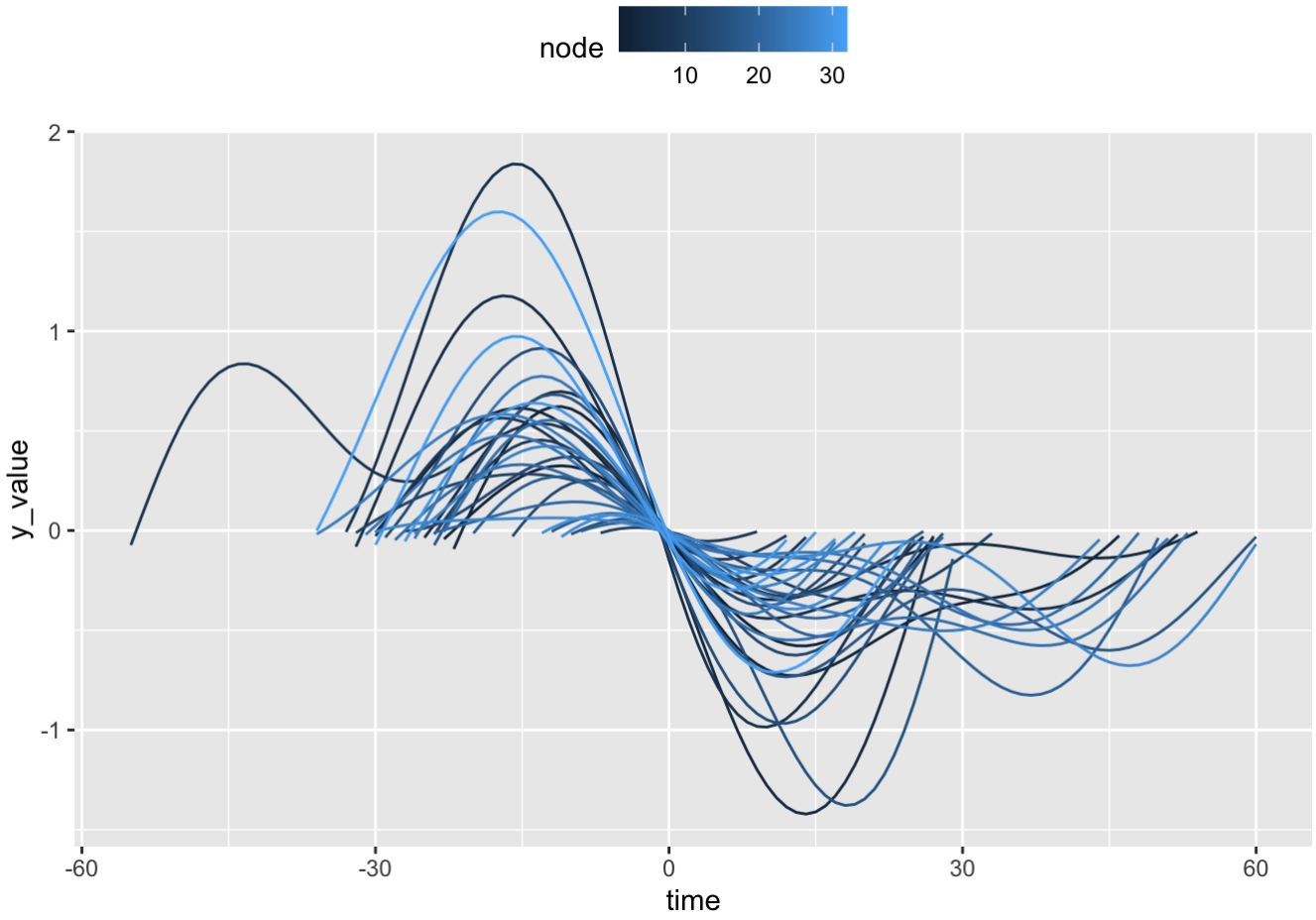

smoothed curves
value
1.5
1.0
0.5
0.0
-0.5
-1.0
0
10
20
30
40
time


PCA function 1 (Percentage of variability 62.5)
Harmonic 1
0.6
-0.4
0
10
20
30
40
x


PCA function 2 (Percentage of variability 29)
Harmonic 2
0.2
-0.6
0
10
20
30
40
x

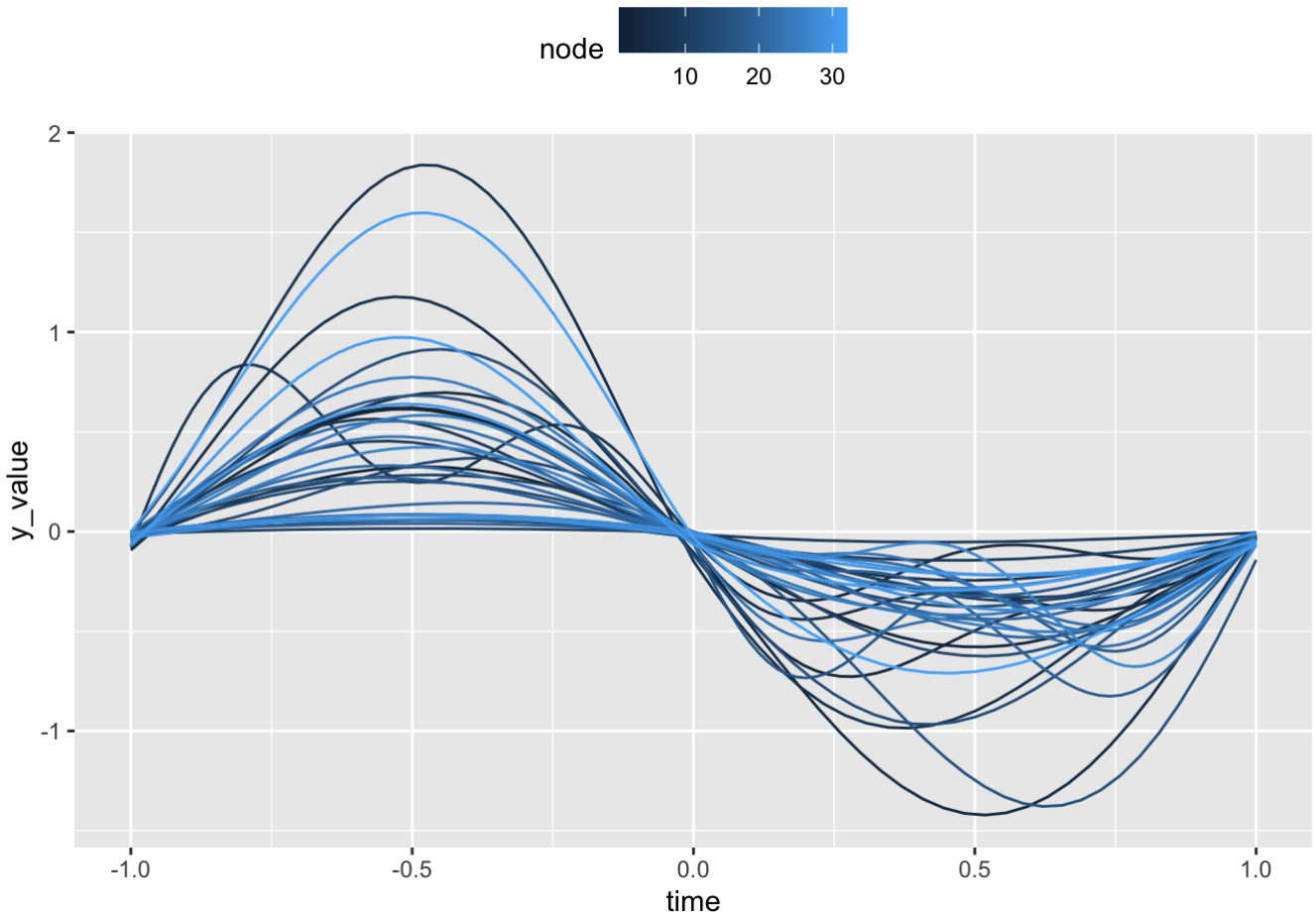
6.1 *2nd* cycle across different node

```
df = data.frame(node=integer(),cycle=integer(), time=integer(), y_value=integer())
for(i in 1:32){
  result_obj <- f_fourier_smooth(time_subset=c(1:600), data_mat, node_subset=c(i), k=32)
  smoothed_curve = eval.fd(c(1:600),result_obj$fd)
  transformed_node = transform.Cycle(smoothed_curve, register=1)
  tmp = subset(transformed_node, cycle==2)
  tmp$node=i
  df=rbind(df,tmp)
}
ggplot(df, aes(time, y_value,group=node, colour=node)) + geom_line() + theme(legend.posi
tion="top")
```



6.2 Scale the timeframe of each node

```
df_tmp = data.frame(node=integer(),cycle=integer(), time=integer(), y_value=integer())
for(i in 1:32){
  tmp=subset(df, node==i)
  tmp_1=subset(tmp, time<=0)
  tmp_2=subset(tmp, time>0)
  tmp_1$time=(tmp_1$time) / (max(abs(tmp_1$time)))
  tmp_2$time=(tmp_2$time) / (max(abs(tmp_2$time)))
  tmp=rbind(tmp_1,tmp_2)
  df_tmp=rbind(df_tmp,tmp)
}
ggplot(df_tmp, aes(time, y_value,group=node, colour=node)) + geom_line() + theme(legend.
position="top")
```

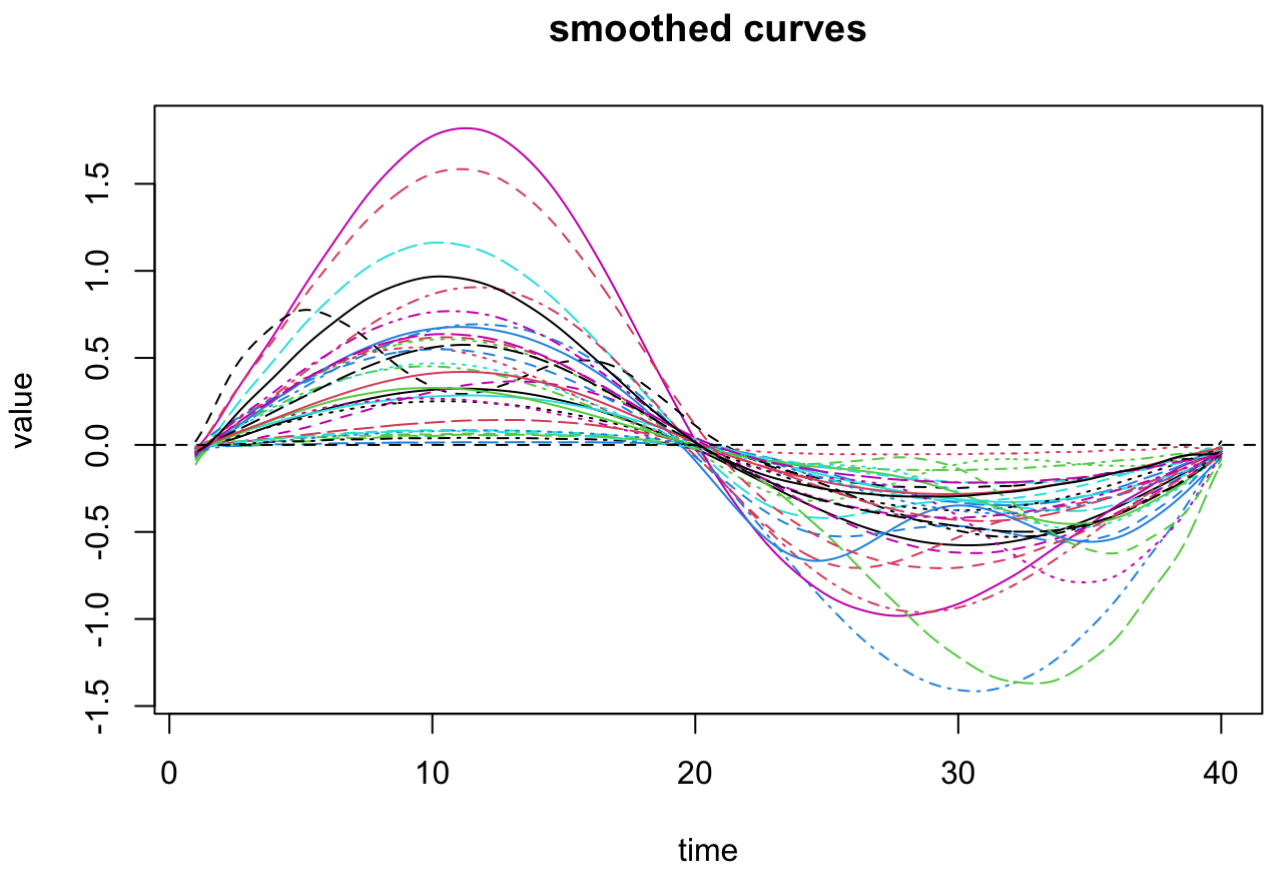


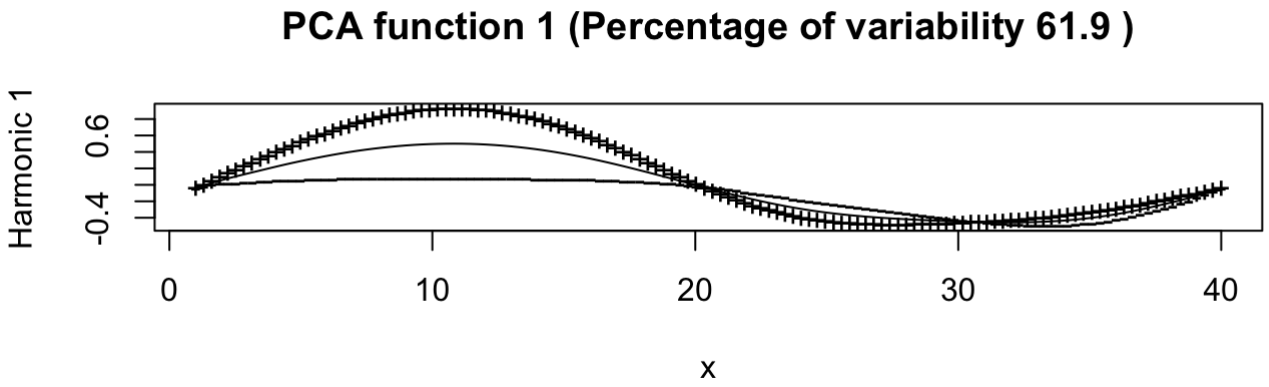
6.3 pivot the new dataframe

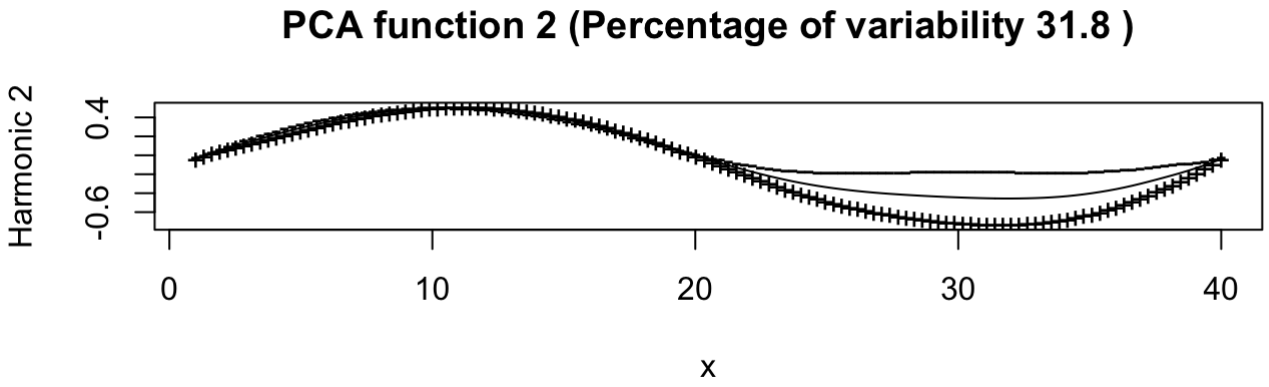
```
df_new = data.frame(matrix(nrow=40))
for(i in 1:32){
  xx=seq(-1,1,length.out=40)
  tmp=subset(df_tmp, node==i)
  s=smooth.spline(x=tmp$time, y=tmp$y_value,  df = 10)
  df_new[,ncol(df_new)+1]=predict(s,xx)$y
}
df_new=df_new[,2:33]
oldnames = colnames(df_new)
newnames = colnames(data_mat)
for(i in 1:32) names(df_new)[names(df_new) == oldnames[i]] = newnames[i]
```

6.4 fPCA on new dataframe

```
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)
}
df_new <- as.matrix(df_new)
rotpcalist = fPCA_subset(time_subset=c(1:40), df_new, node_subset = c(1:32)
                         , k=32, nharm=2)
```


smoothed curves
value
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0
10
20
30
40
time


PCA function 1 (Percentage of variability 61.9)
Harmonic 1
0.6
-0.4
0
10
20
30
40
x


PCA function 2 (Percentage of variability 31.8)
Harmonic 2
0.4
-0.6
0
10
20
30
40
x