3aug2020_v1

Zach Wang

8/3/2020

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

5. Getting Harmonics curves based on Eigenfunctions

Note: I am having problem getting the math right. Below, I calculated each Harmonics based on the formula:

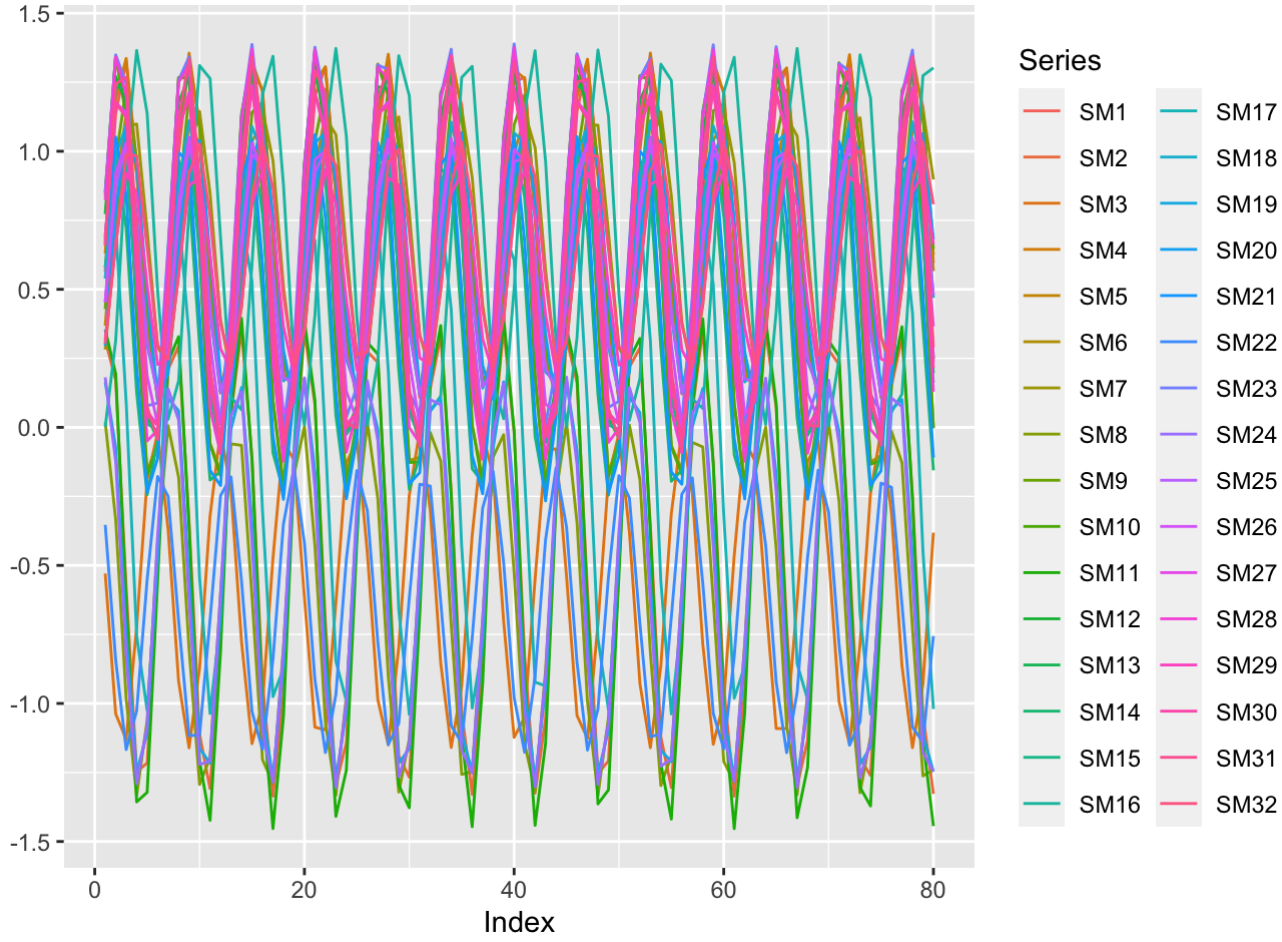
$$Eigenfunction(t) = \sum_{k=1}^{K} c_k \Phi_k(t)$$

and

$$\Phi_k(t) = c_1 + c_2 sin(t) + c_3 cos(t) + c_4 sin(t) + c_5 cos(t) + \ldots = c_1 + (c_2 + c_4 + \ldots) sin(t) + (c_3 + c_5 + \ldots) cos(t)$$

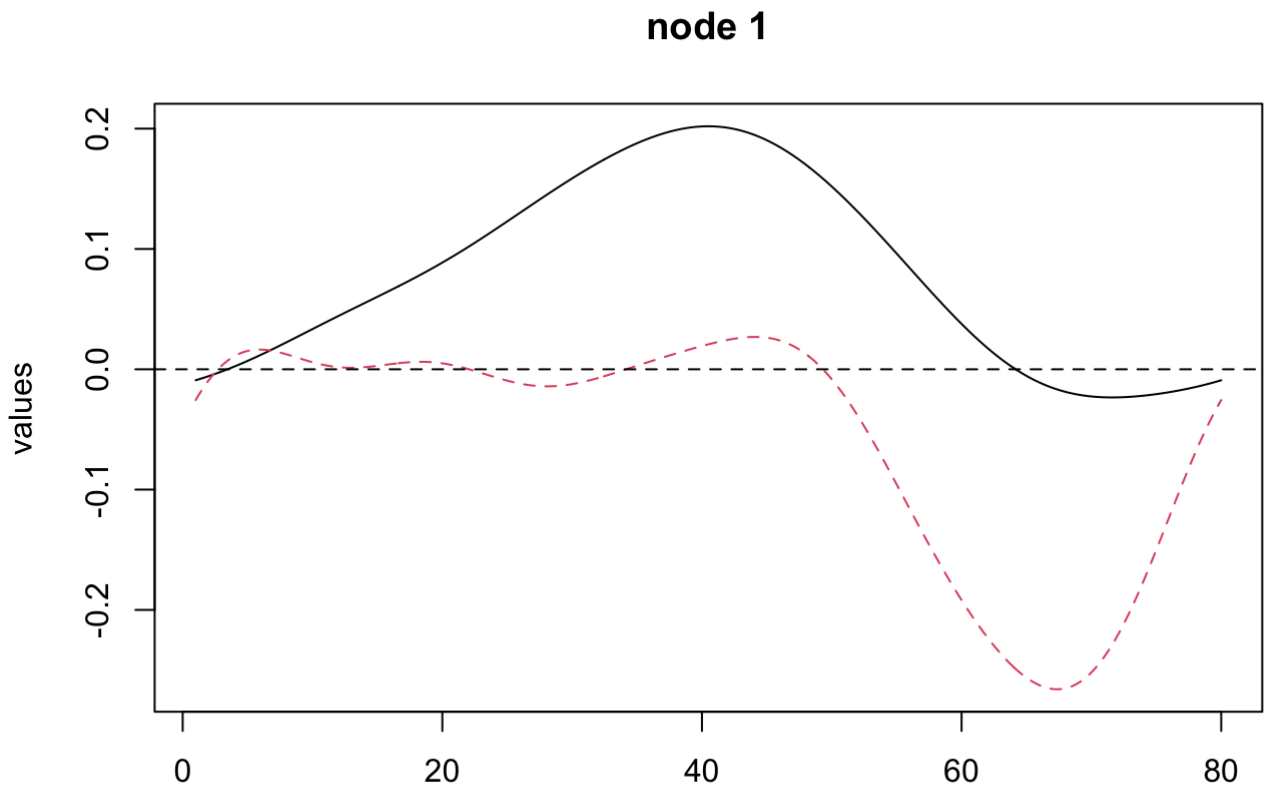
```
# find y value based on PC coef and sin/cos functions
PC1_df = data.frame(matrix(nrow=80))
PC2_df = data.frame(matrix(nrow=80))
for(node in 1:32){
    result_obj <- f_fourier_smooth(time_subset=c(1:600), data_mat, node_subset=c(node),
 k=32)
    smoothed_curve = eval.fd(c(1:600),result_obj$fd)
    transformed_node = transform.Cycle(smoothed_curve, register=1)
    df_tmp = data.frame(cycle=integer(), time=integer(), y_value=integer())
    for(i in 1:length(unique(transformed_node$cycle))){
      tmp=subset(transformed_node, cycle==i)
      tmp$time=(tmp$time)/max(tmp$time)
      df_tmp=rbind(df_tmp,tmp)
    }
    df_new = data.frame(matrix(nrow=80))
    for(i in 1:length(unique(df_tmp$cycle))){
      xx=seq(0,1,length.out=80)
      tmp=subset(df_tmp, cycle==i)
      s=smooth.spline(x=tmp$time, y=tmp$y_value,  df = 10)
      df_new[,ncol(df_new)+1]=predict(s,xx)$y
    }
    df_new=df_new[,2:(length(unique(df_tmp$cycle))+1)]
    ## FPCA now
    fPCA_subset <- function(data_mat, k, nharm, plt){
      basis <- create.fourier.basis(c(1,nrow(data_mat)), k)
      smoothfd <- smooth.basis(1:nrow(data_mat), data_mat, basis)$fd
      pcalist = pca.fd(smoothfd, nharm, harmfdPar=fdPar(smoothfd))
      rotpcalist = varmx.pca.fd(pcalist)
      par(mfrow=c(nharm,1))
      if(plt==1){
        plot.pca.fd(rotpcalist)
      }
      return(rotpcalist)
    }
    df_new <- as.matrix(df_new)
    rotpcalist = fPCA_subset(df_new, k=11, nharm=2, plt=0)
    PCA_df <- data.frame(matrix(nrow=rotpcalist$harmonics$basis$rangeval[2]))
    # Calculation for Harmonics/eigenvectors
    for(x in (rotpcalist$harmonics$basis$rangeval[1]:rotpcalist$harmonics$basis$rangeval
[2])){
      PC1_coef = rotpcalist$harmonics$coefs[,1]
      PC2_coef = rotpcalist$harmonics$coefs[,2]
      flag = ifelse(index(PC1_coef)%%2==0, 1, 0)
      even_idx = (1:length(PC1_coef))[flag==1]
      odd_idx = (1:length(PC1_coef))[flag==0][-1]
      PCA_df[x, 1] = PC1_coef[1] + sin(x)*sum(PC1_coef[even_idx]) + cos(x)*sum(PC1_coef
[odd_idx]) # fixme
      PCA_df[x, 2] = PC2_coef[1] + sin(x)*sum(PC2_coef[even_idx]) + cos(x)*sum(PC2_coef
[odd_idx]) # fixme

        ## calculation trial 2
      #PCA_df[x, 1] = PC1_coef[1]
      #PCA_df[x, 2] = PC2_coef[1]
```

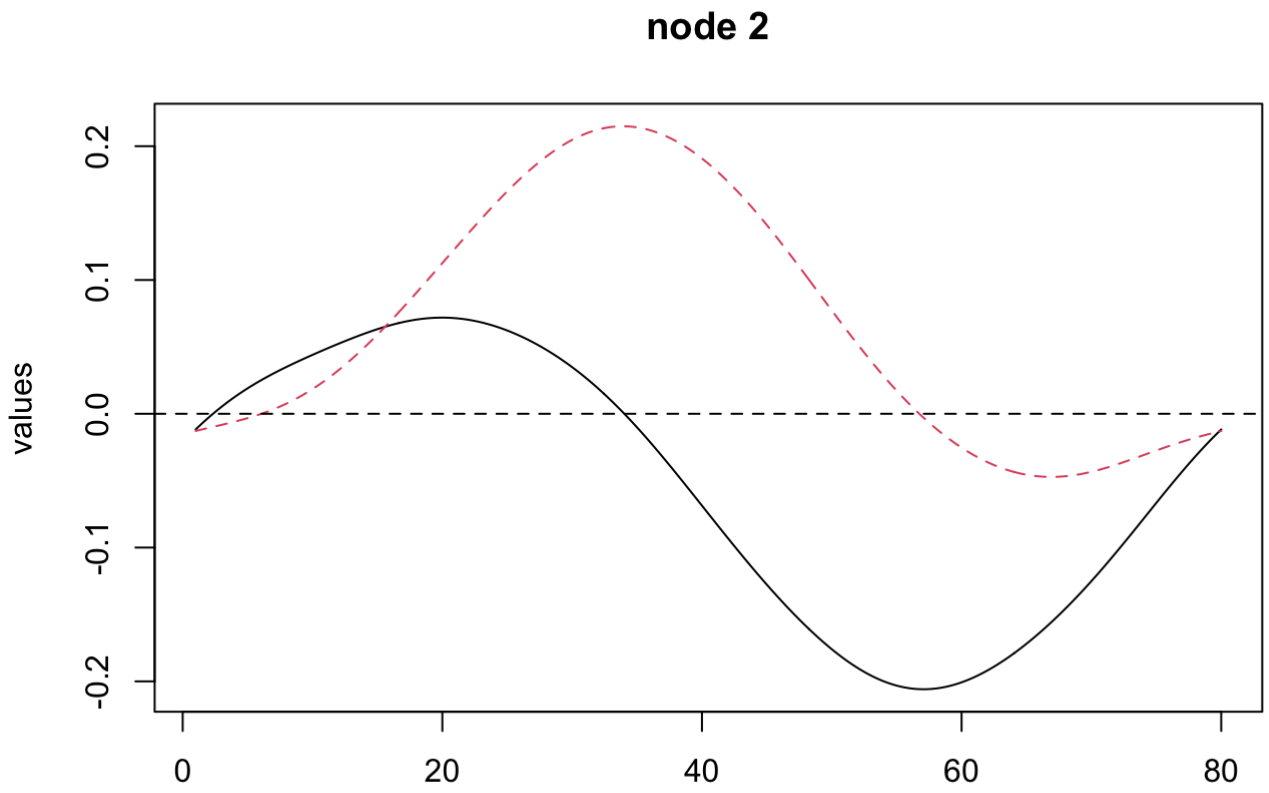
```
      #t=1
      #for(i in 1:length((even_idx))){
      #  PCA_df[x, 1]=PCA_df[x, 1]+sin(t*x*PC1_coef[even_idx[i]])
      #  PCA_df[x, 2]=PCA_df[x, 2]+sin(t*x*PC2_coef[even_idx[i]])
      #  t=t+1
      #}
      #t=1
      #for(j in 1:length((odd_idx))){
      #  PCA_df[x, 1]=PCA_df[x, 1]+sin(t*x*PC1_coef[odd_idx[j]])
      #  PCA_df[x, 2]=PCA_df[x, 2]+sin(t*x*PC2_coef[odd_idx[j]])
      #  t=t+1
      #}

    }
PC1_df[,ncol(PC1_df)+1]=PCA_df[,1]
PC2_df[,ncol(PC2_df)+1]=PCA_df[,2]
}

PC1_df = data.frame(PC1_df[,2:(ncol(PC1_df))])
names(PC1_df)=colnames(data_mat)

PC2_df = data.frame(PC2_df[,2:(ncol(PC2_df))])
names(PC2_df)=colnames(data_mat)

z_1 = read.zoo(PC1_df, index='index')
z_2 = read.zoo(PC2_df, index='index')
autoplot(z_1, facet = NULL)
```

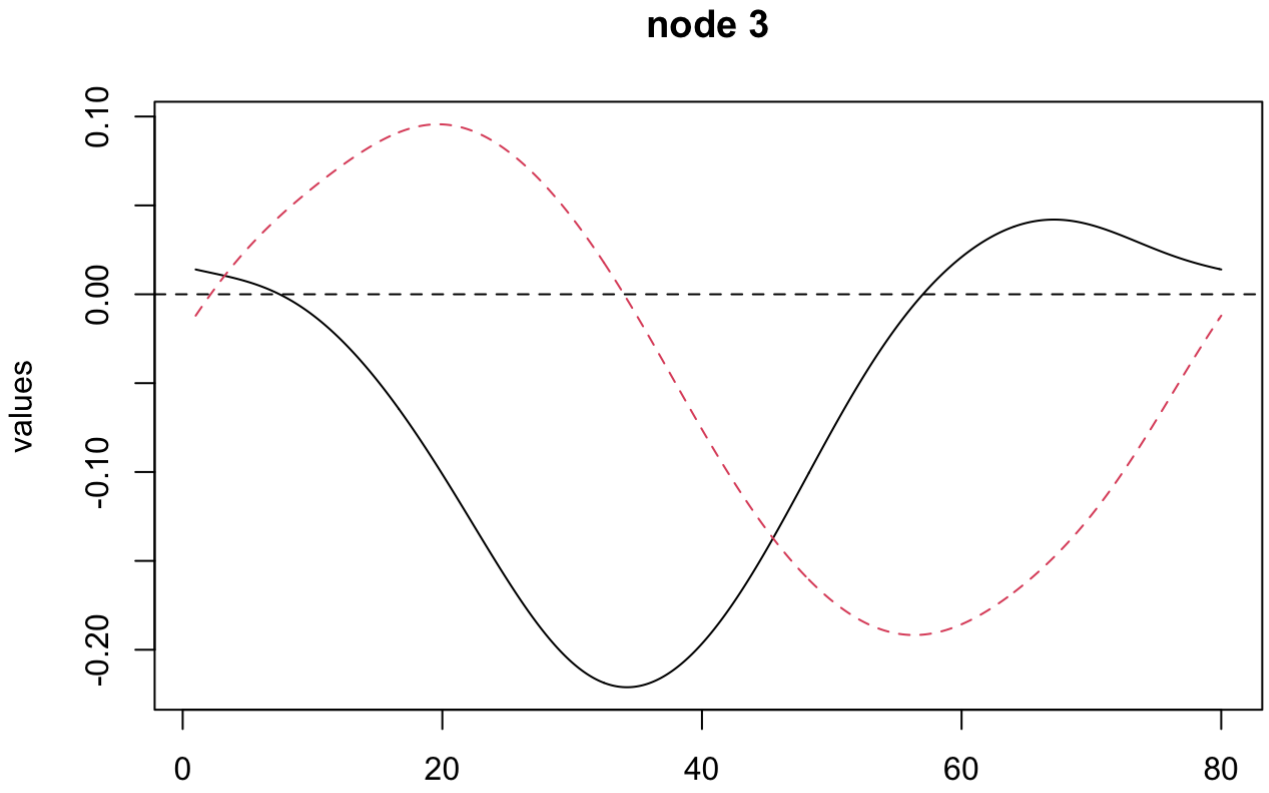


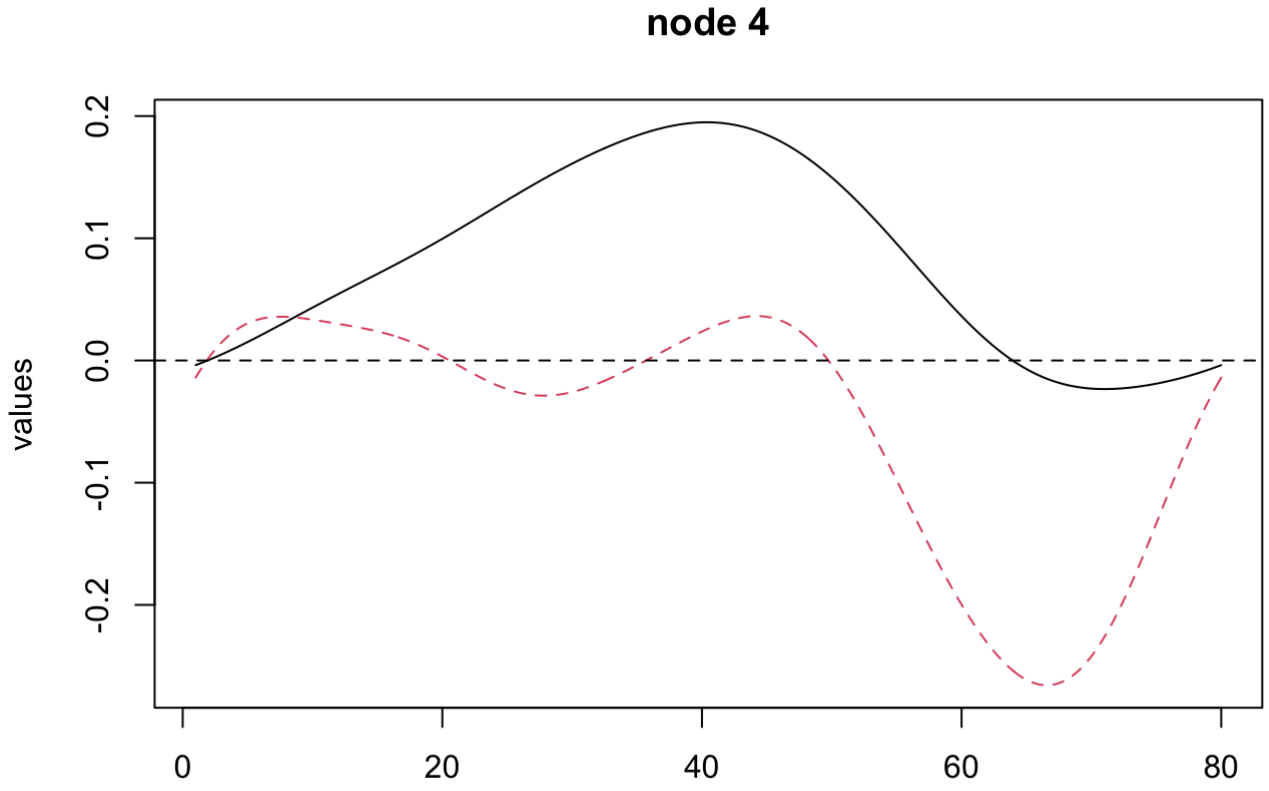
6. Plot the 32 Harmonics plots directly instead

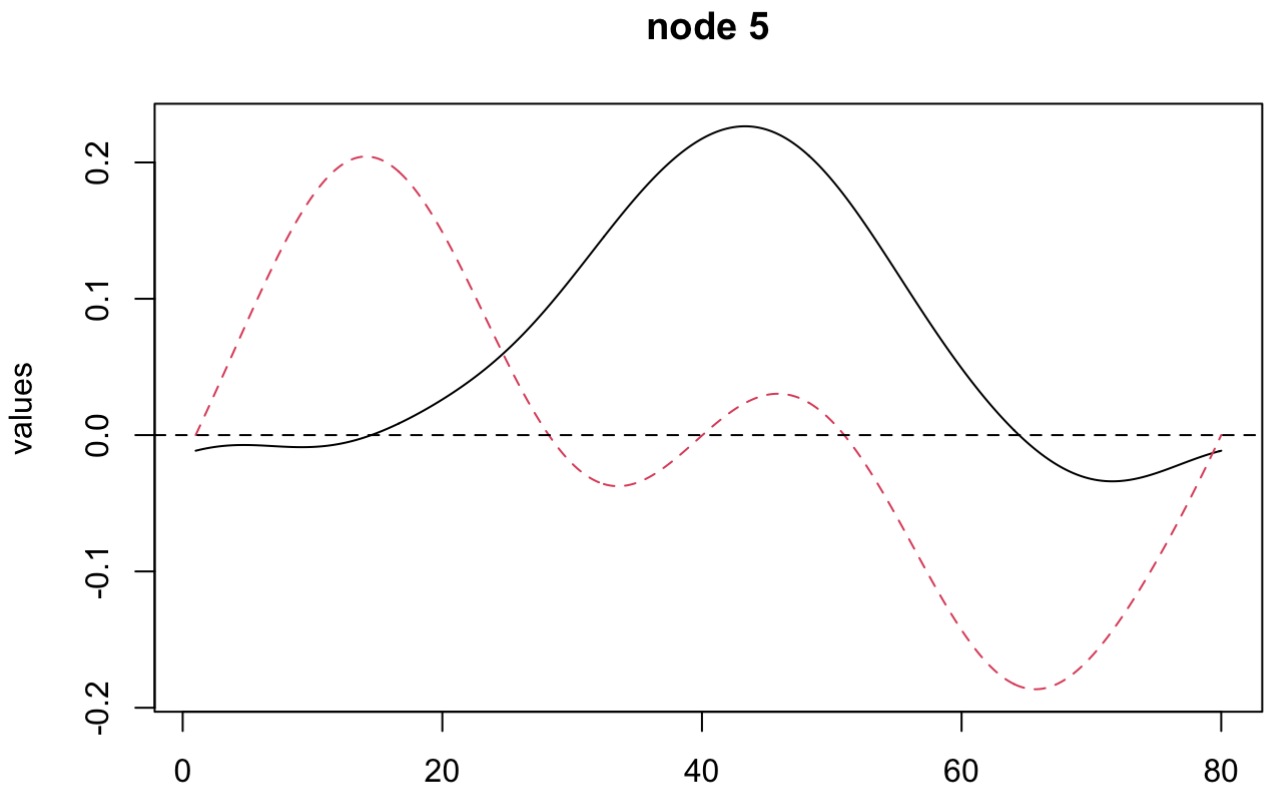
```
# plot 32 plots instead
for(node in 1:32){
    result_obj <- f_fourier_smooth(time_subset=c(1:600), data_mat, node_subset=c(node),
 k=32)
    smoothed_curve = eval.fd(c(1:600),result_obj$fd)
    transformed_node = transform.Cycle(smoothed_curve, register=1)
    df_tmp = data.frame(cycle=integer(), time=integer(), y_value=integer())
    for(i in 1:length(unique(transformed_node$cycle))){
      tmp=subset(transformed_node, cycle==i)
      tmp$time=(tmp$time)/max(tmp$time)
      df_tmp=rbind(df_tmp,tmp)
    }
    df_new = data.frame(matrix(nrow=80))
    for(i in 1:length(unique(df_tmp$cycle))){
      xx=seq(0,1,length.out=80)
      tmp=subset(df_tmp, cycle==i)
      s=smooth.spline(x=tmp$time, y=tmp$y_value,  df = 10)
      df_new[,ncol(df_new)+1]=predict(s,xx)$y
    }
    df_new=df_new[,2:(length(unique(df_tmp$cycle))+1)]
    ## FPCA now
    fPCA_subset <- function(data_mat, k, nharm, plt){
      basis <- create.fourier.basis(c(1,nrow(data_mat)), k)
      smoothfd <- smooth.basis(1:nrow(data_mat), data_mat, basis)$fd
      pcalist = pca.fd(smoothfd, nharm, harmfdPar=fdPar(smoothfd))
      rotpcalist = varmx.pca.fd(pcalist)
      par(mfrow=c(nharm,1))
      if(plt==1){
        plot.pca.fd(rotpcalist)
      }
      return(rotpcalist)
    }
    df_new <- as.matrix(df_new)
    rotpcalist = fPCA_subset(df_new, k=11, nharm=2, plt=0)
    par(mfrow=c(1,1))
    plot(rotpcalist$harmonics)
    title(paste("node",node))
}
```

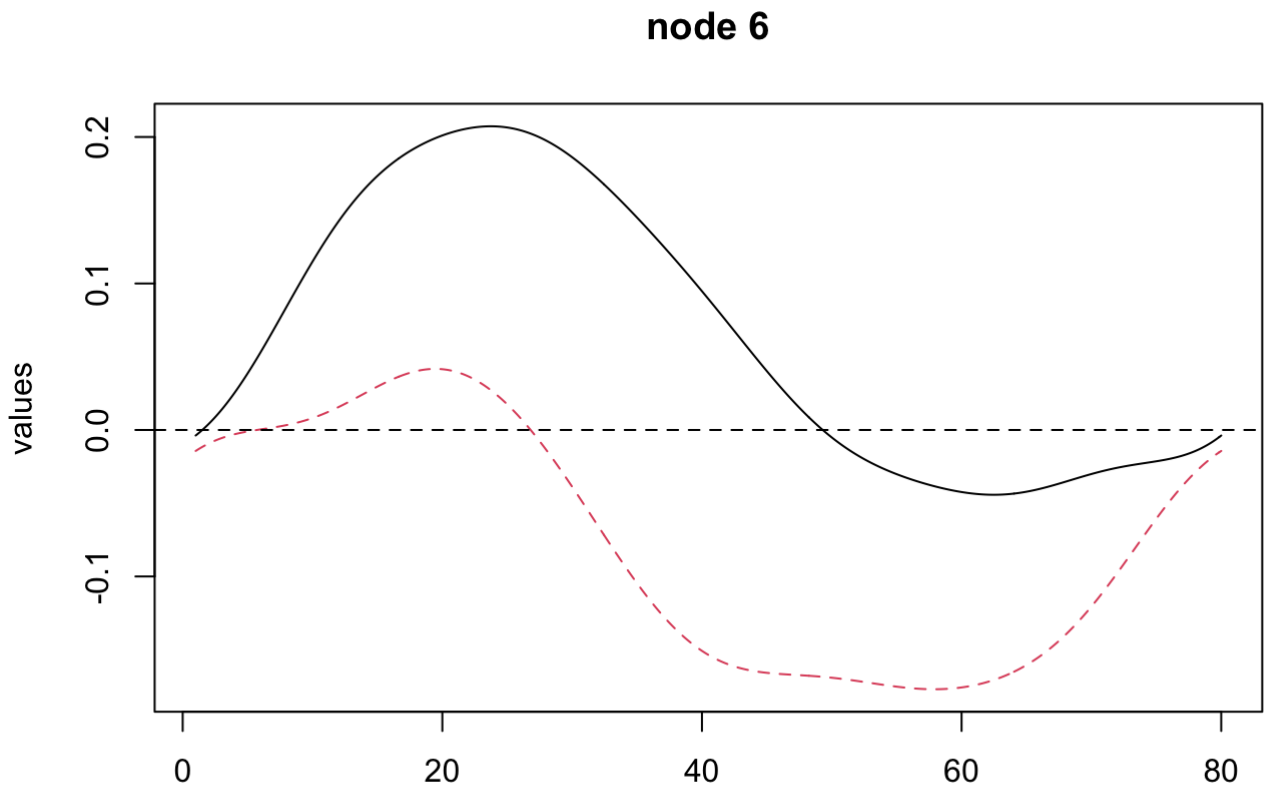

node 1
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

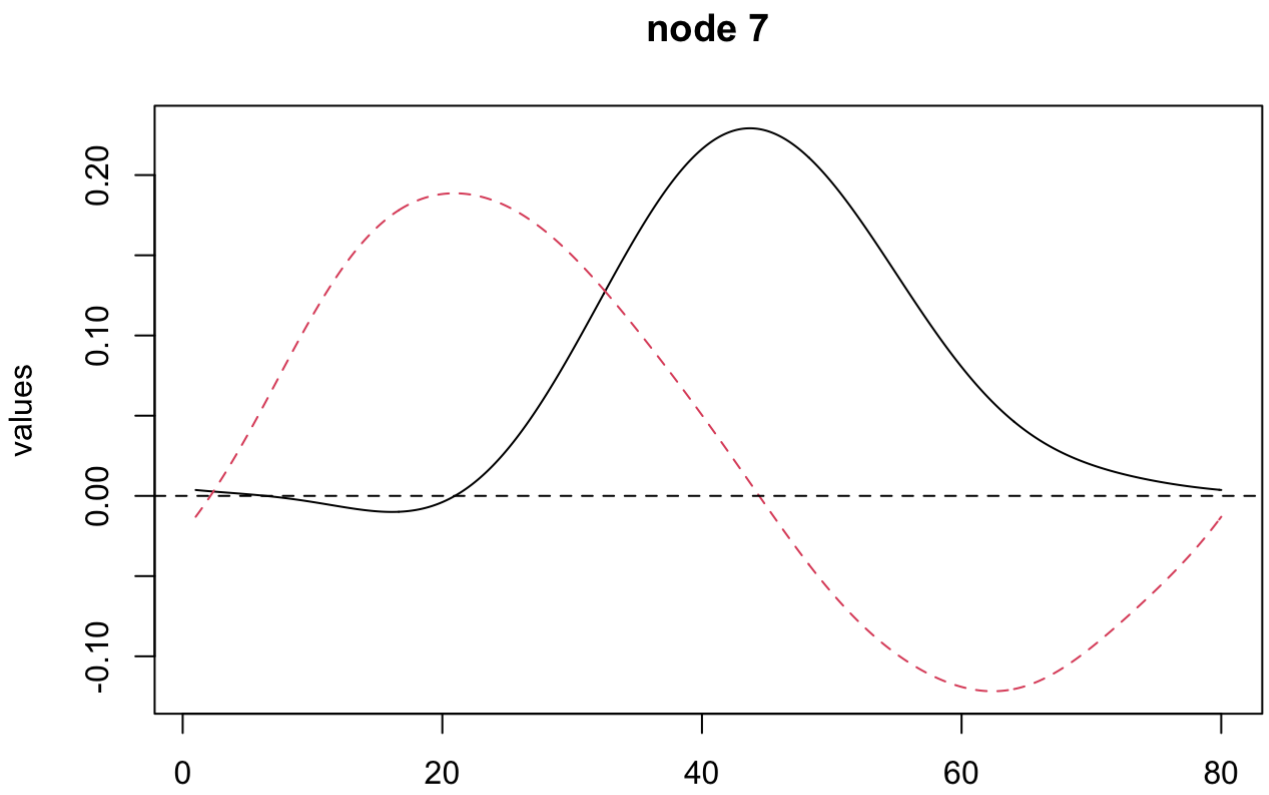

node 2
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

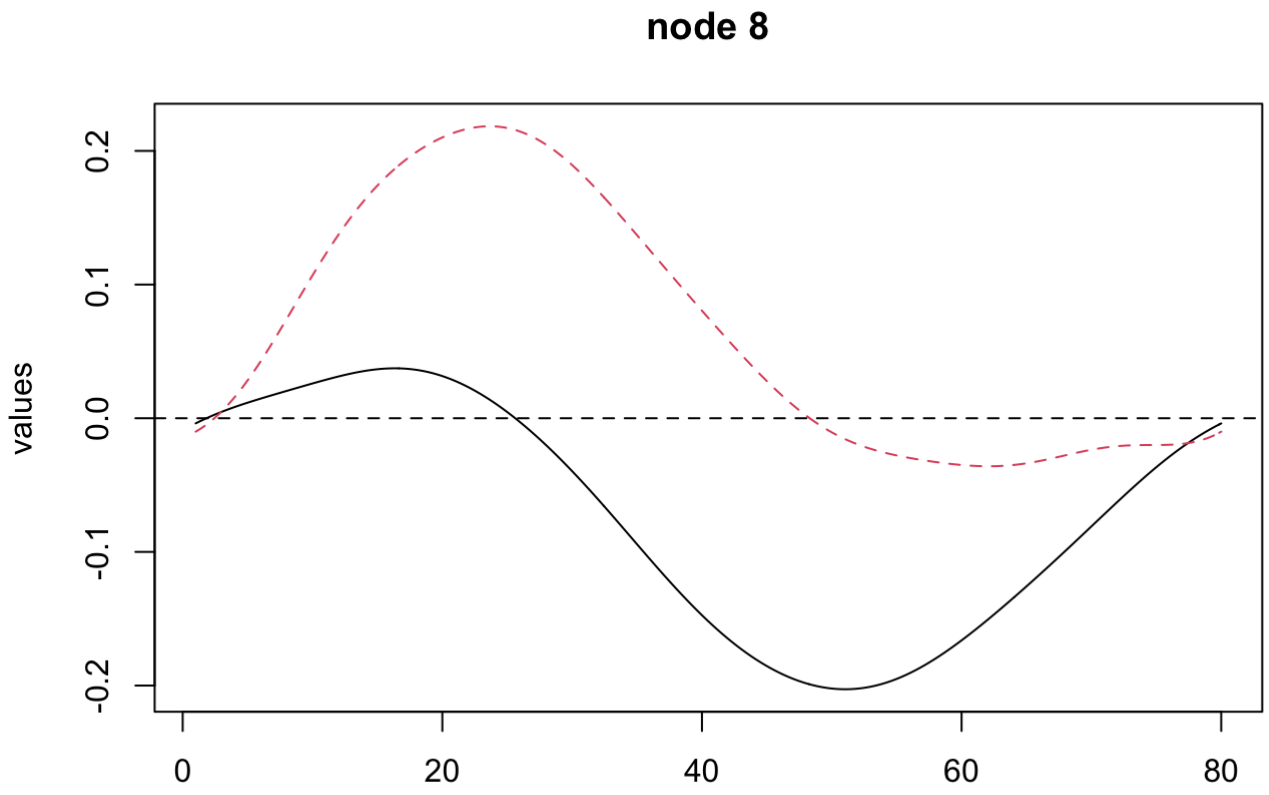

node 3
values
0.10
0.00
-0.10
-0.20
0
20
40
60
80

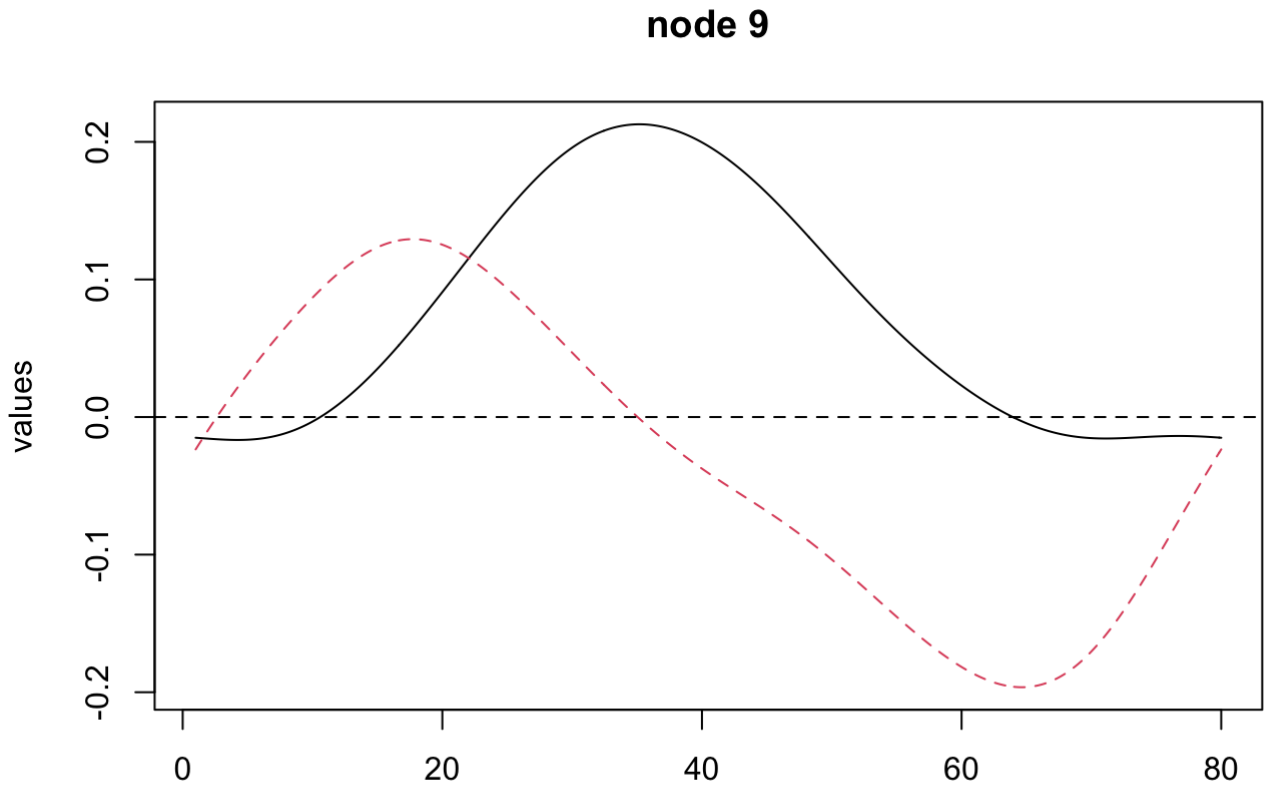

node 4
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

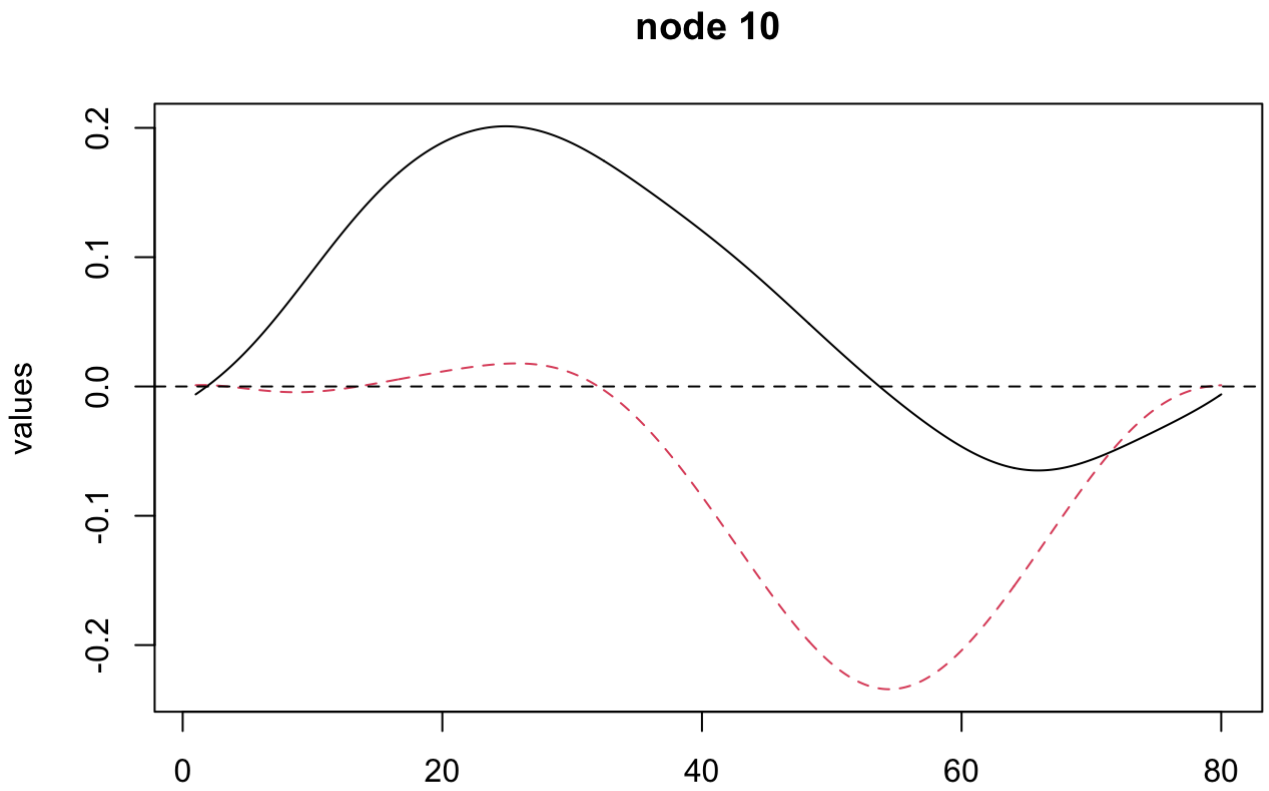

node 5
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

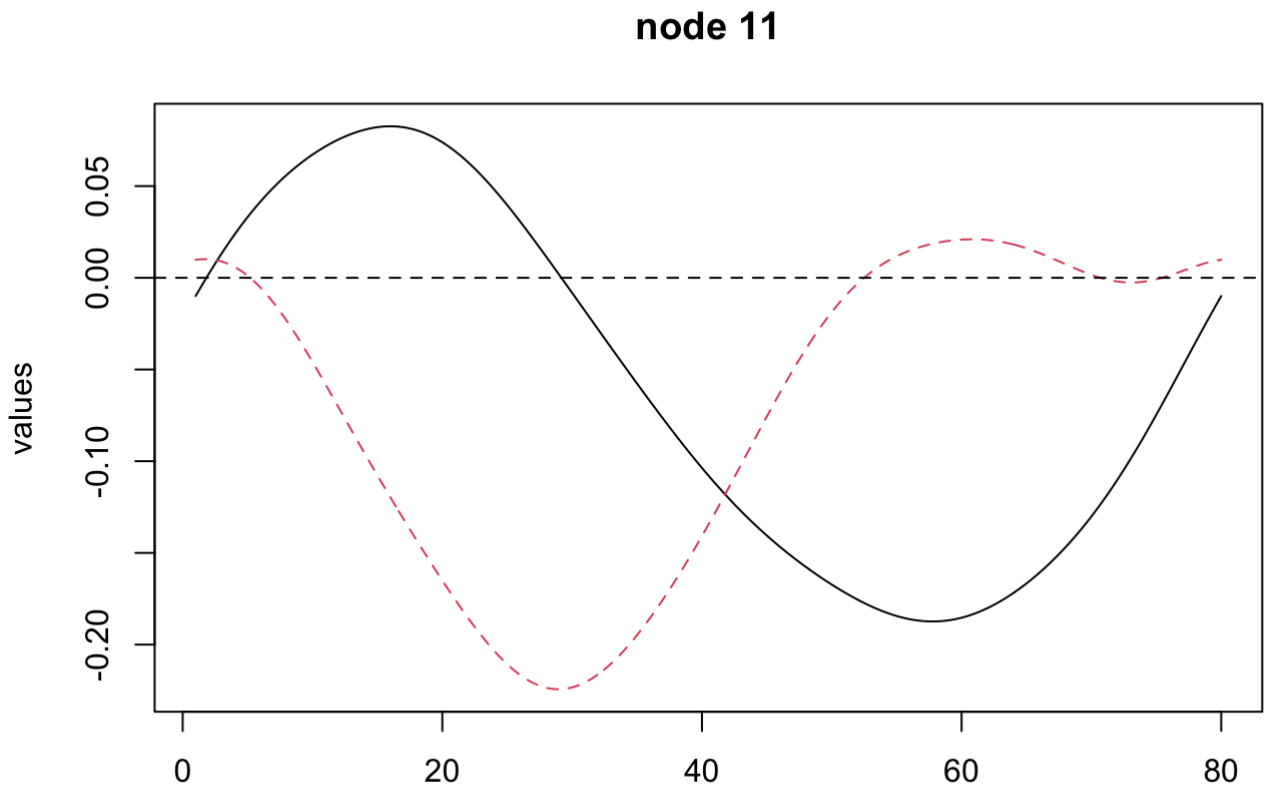

node 6
values
0.2
0.1
0.0
-0.1
0
20
40
60
80

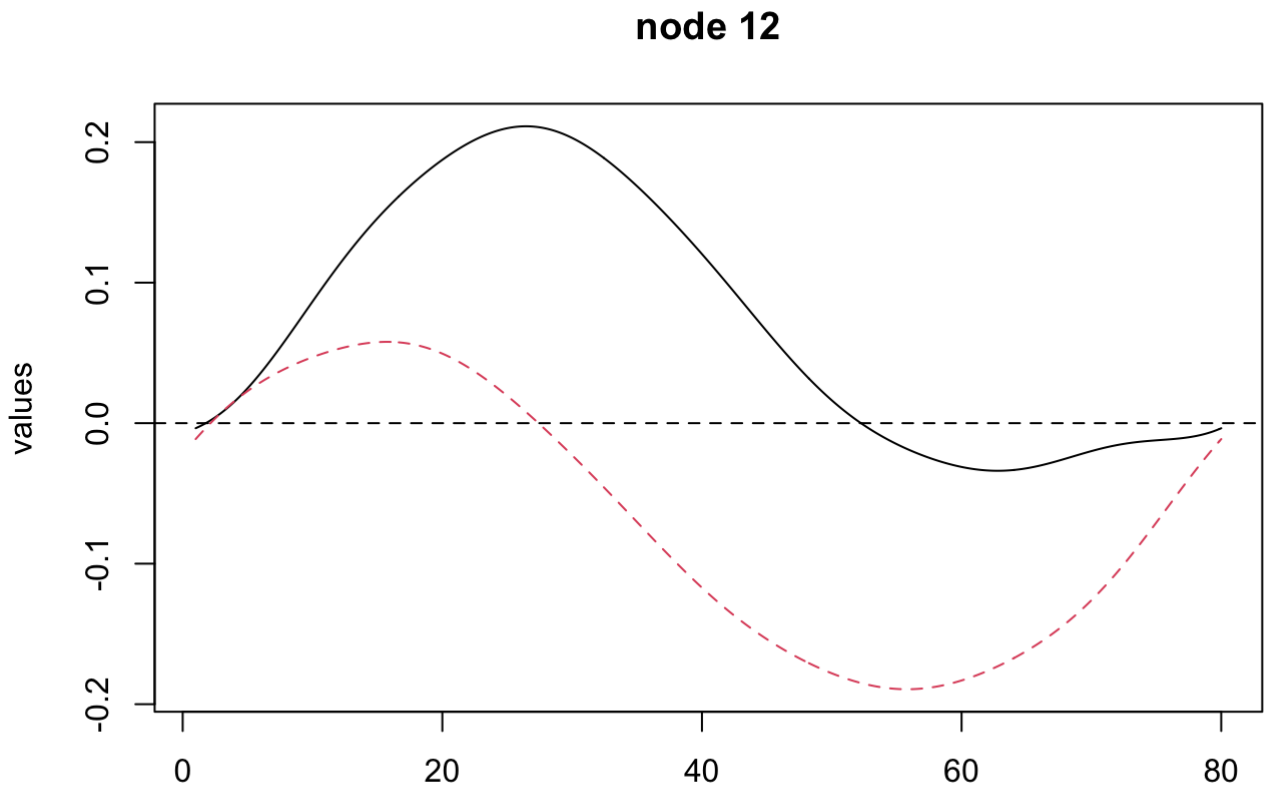

node 7
values
0.20
0.10
0.00
-0.10
0
20
40
60
80

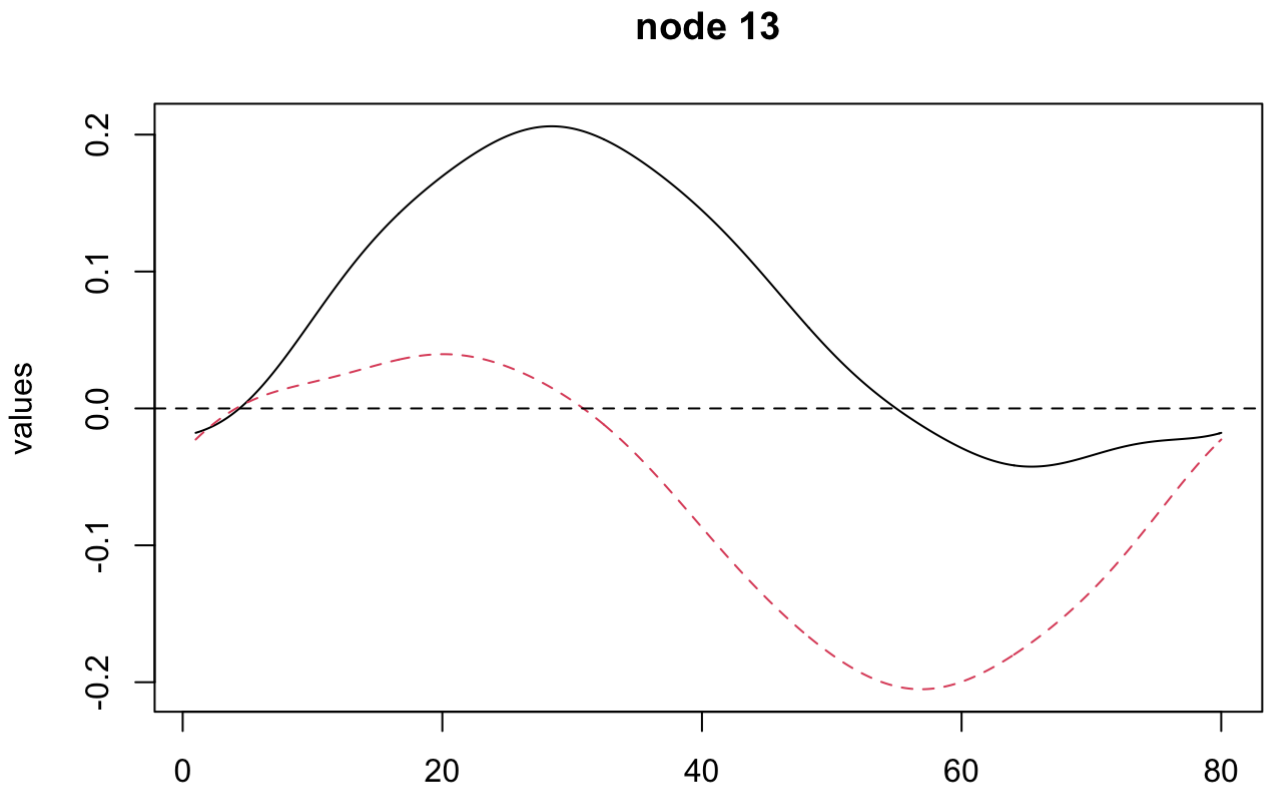

node 8
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

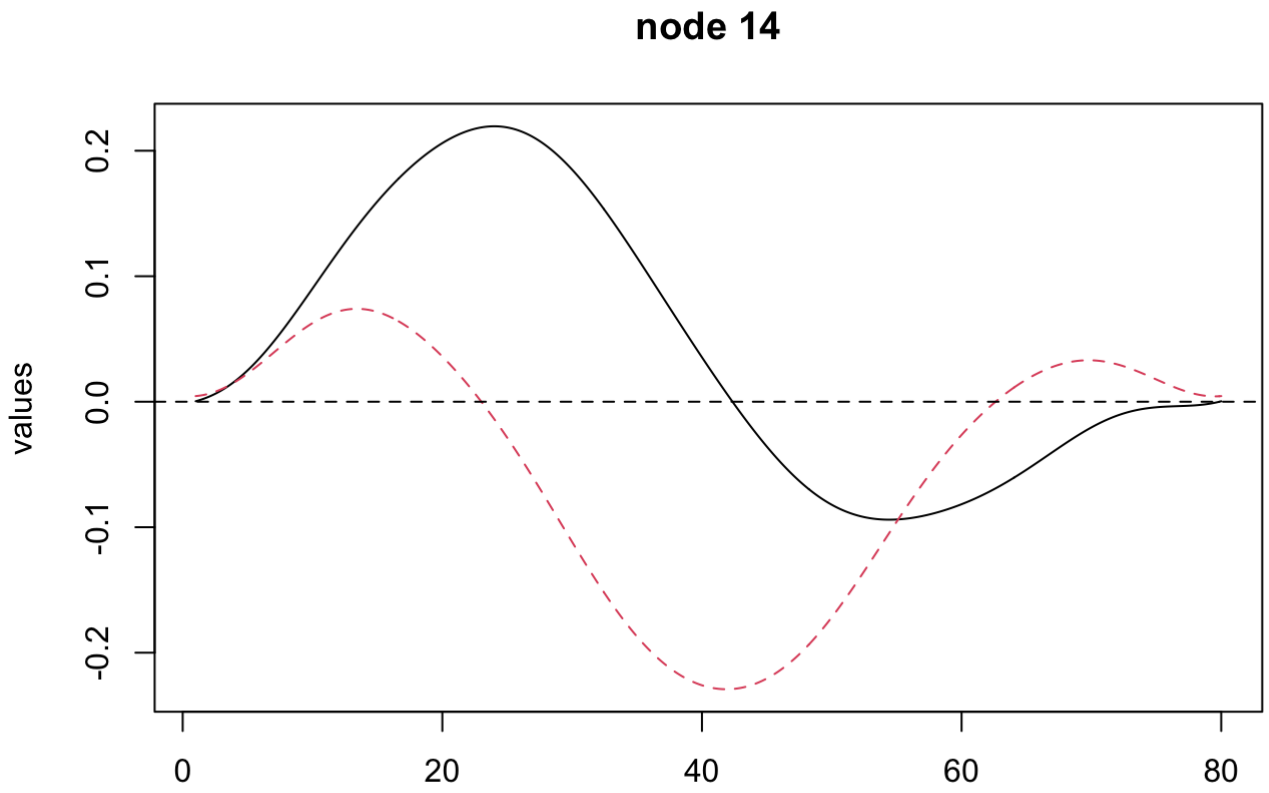

node 9
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

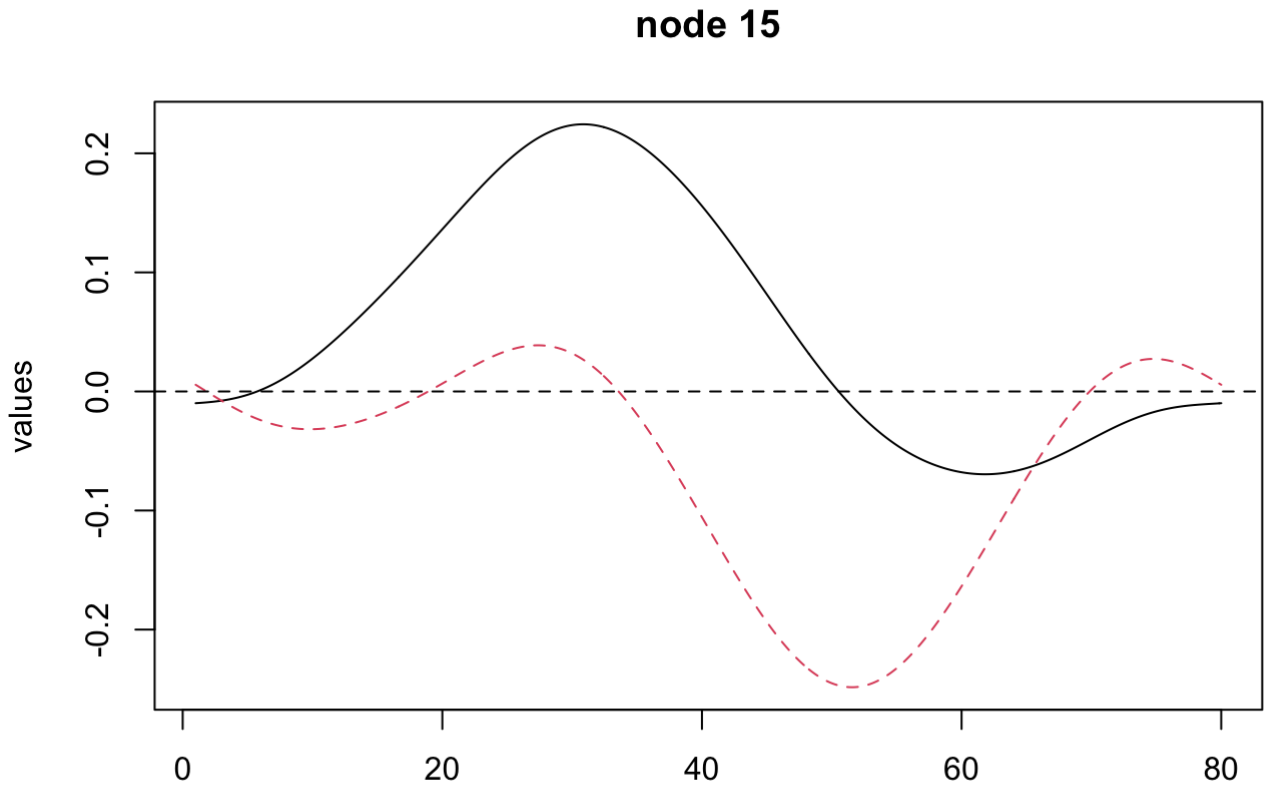

node 10
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

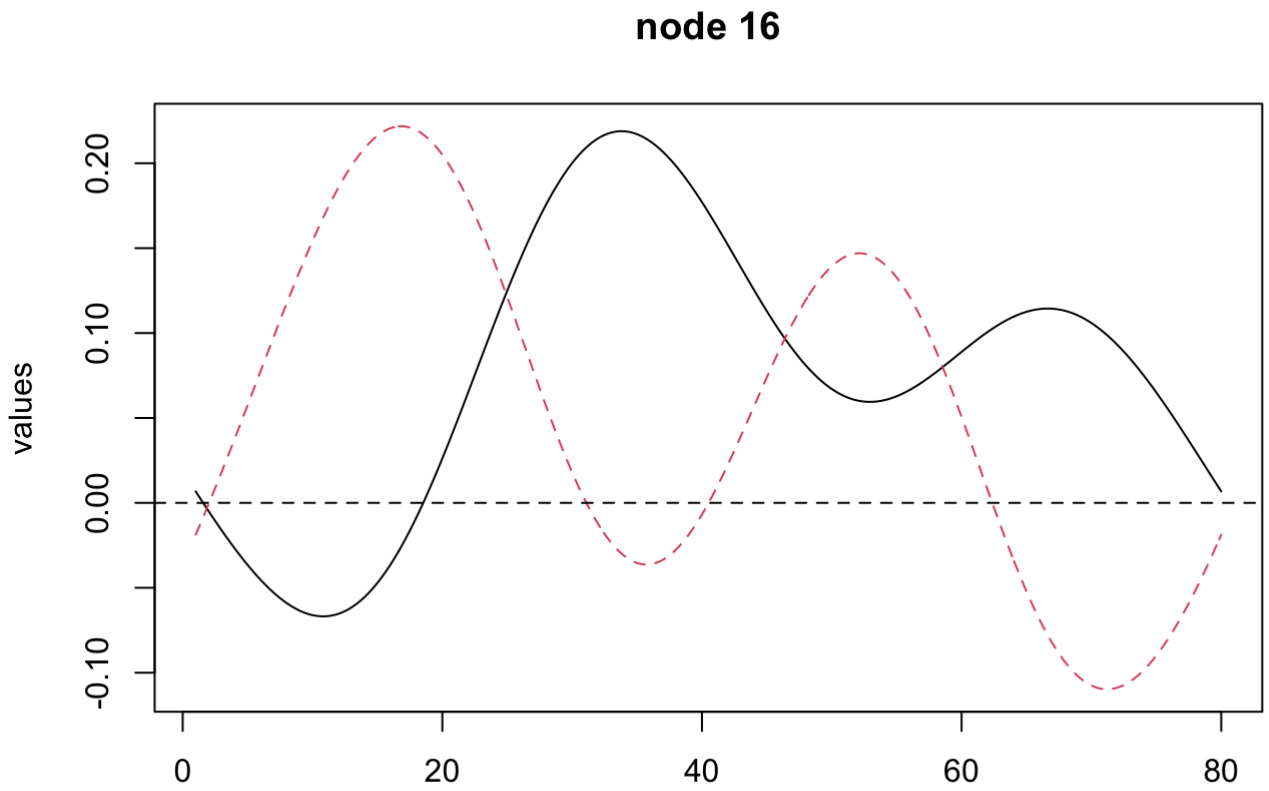

node 11
values
0.05
0.00
-0.10
-0.20
0
20
40
60
80

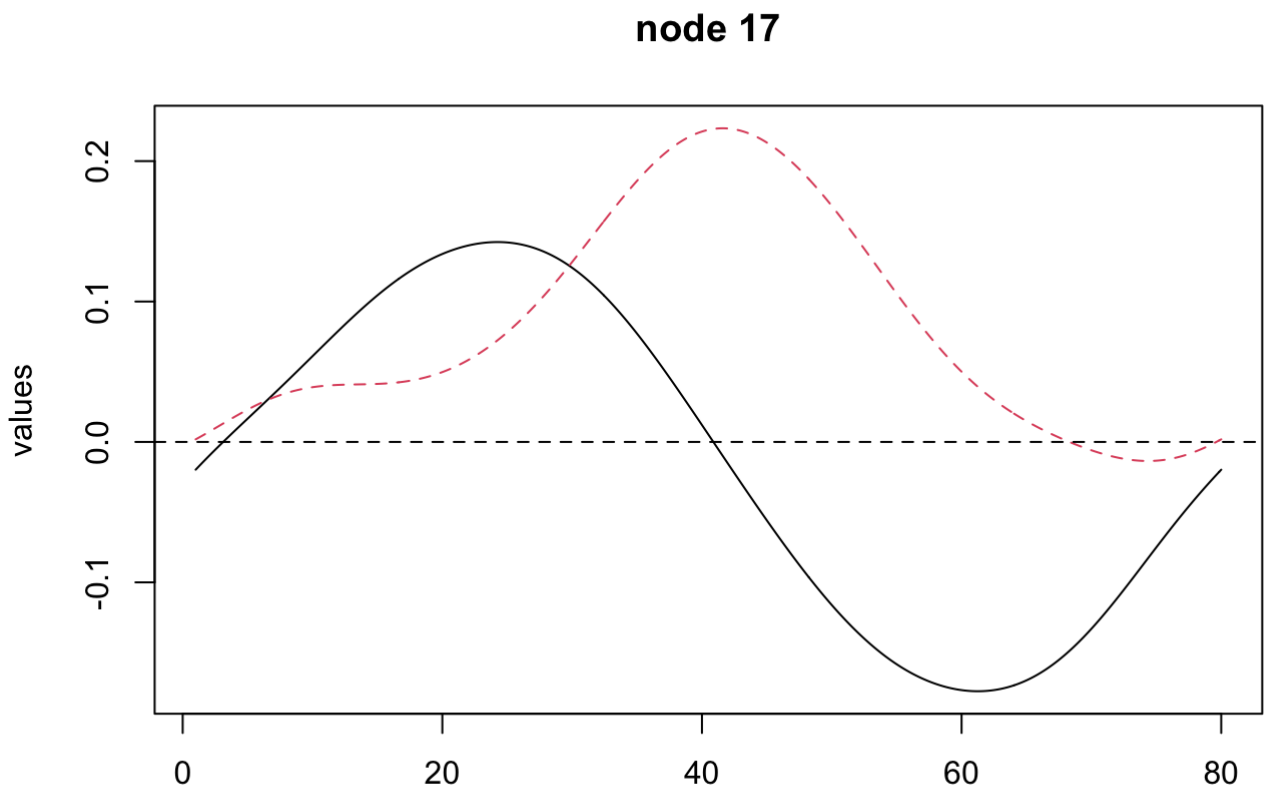

node 12
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

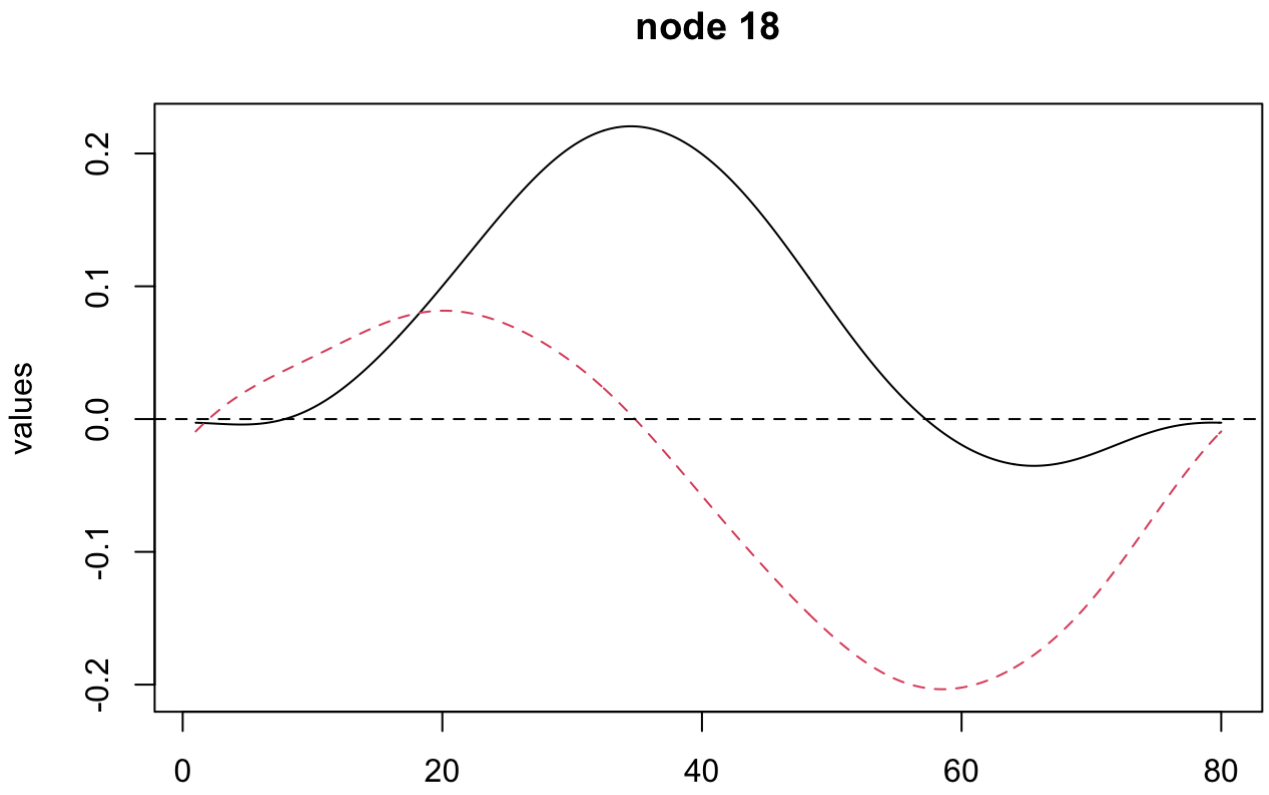

node 13
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

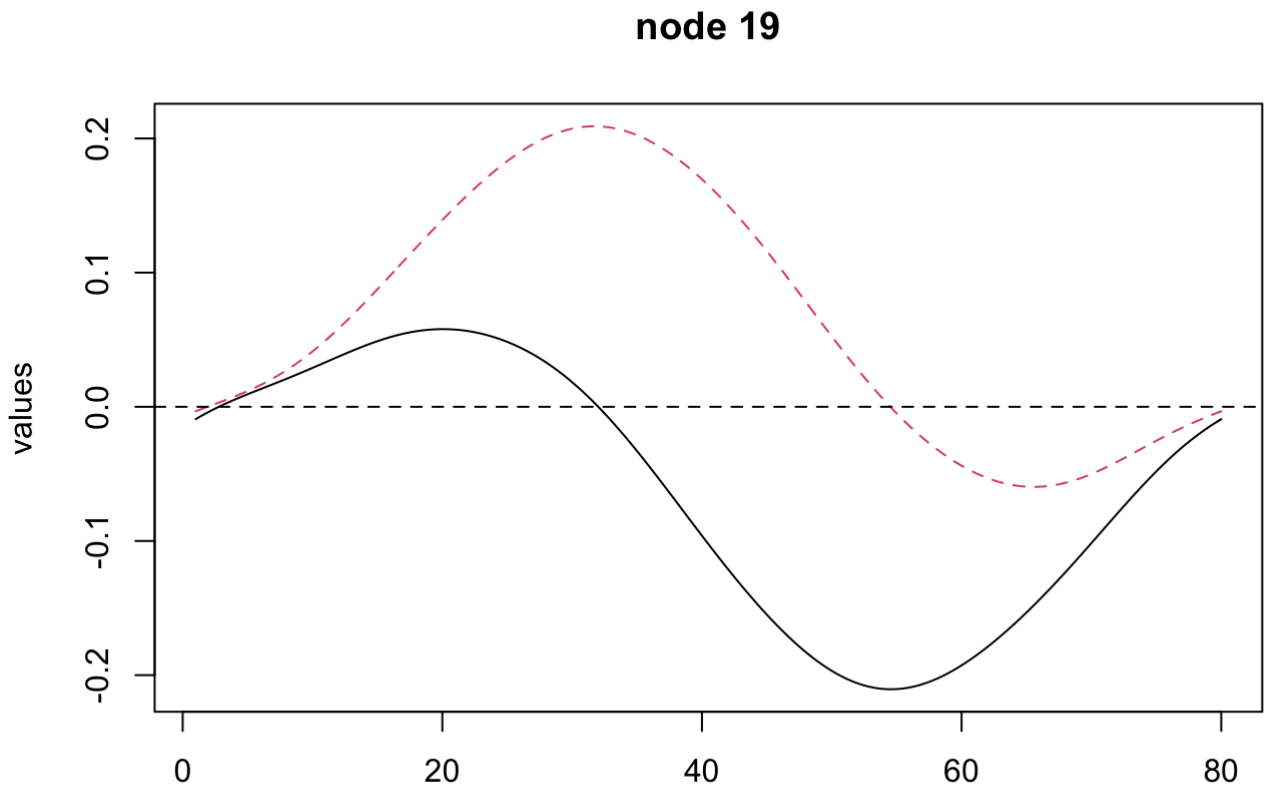

node 14
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

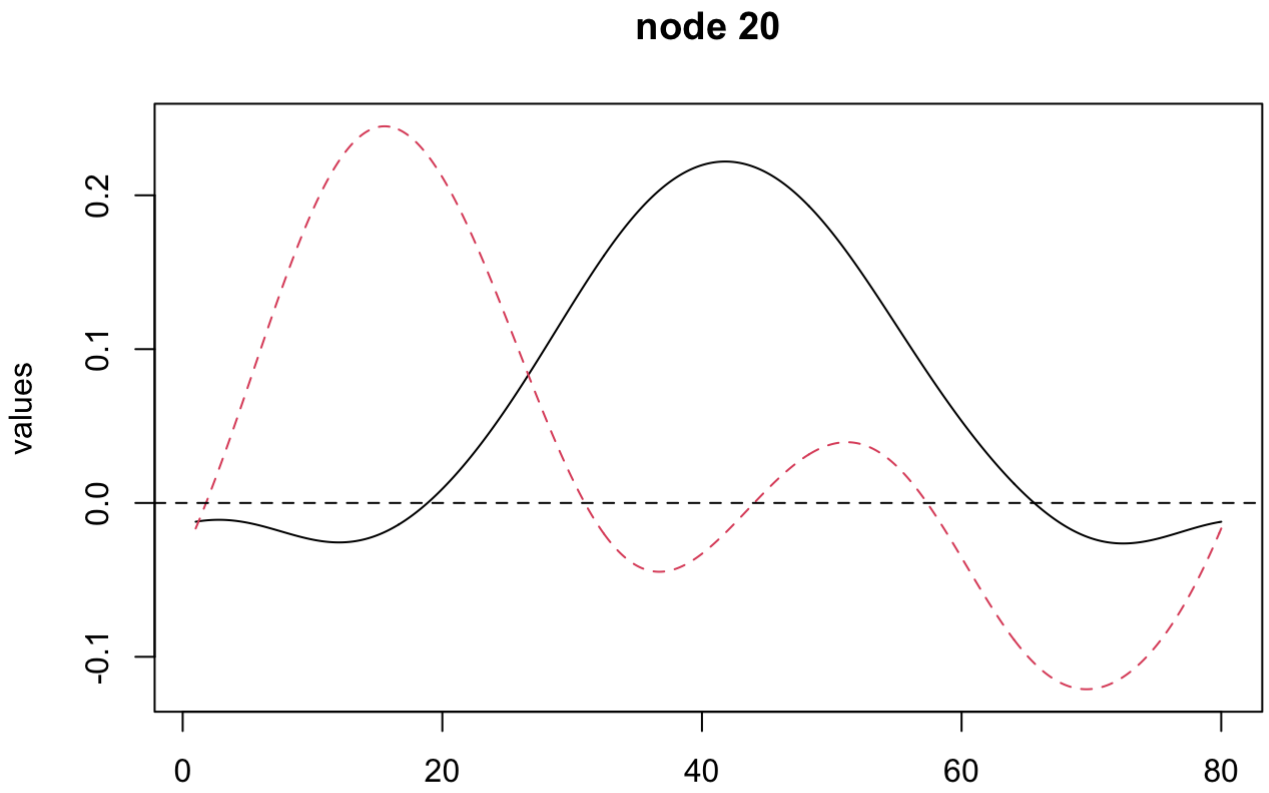

node 15
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

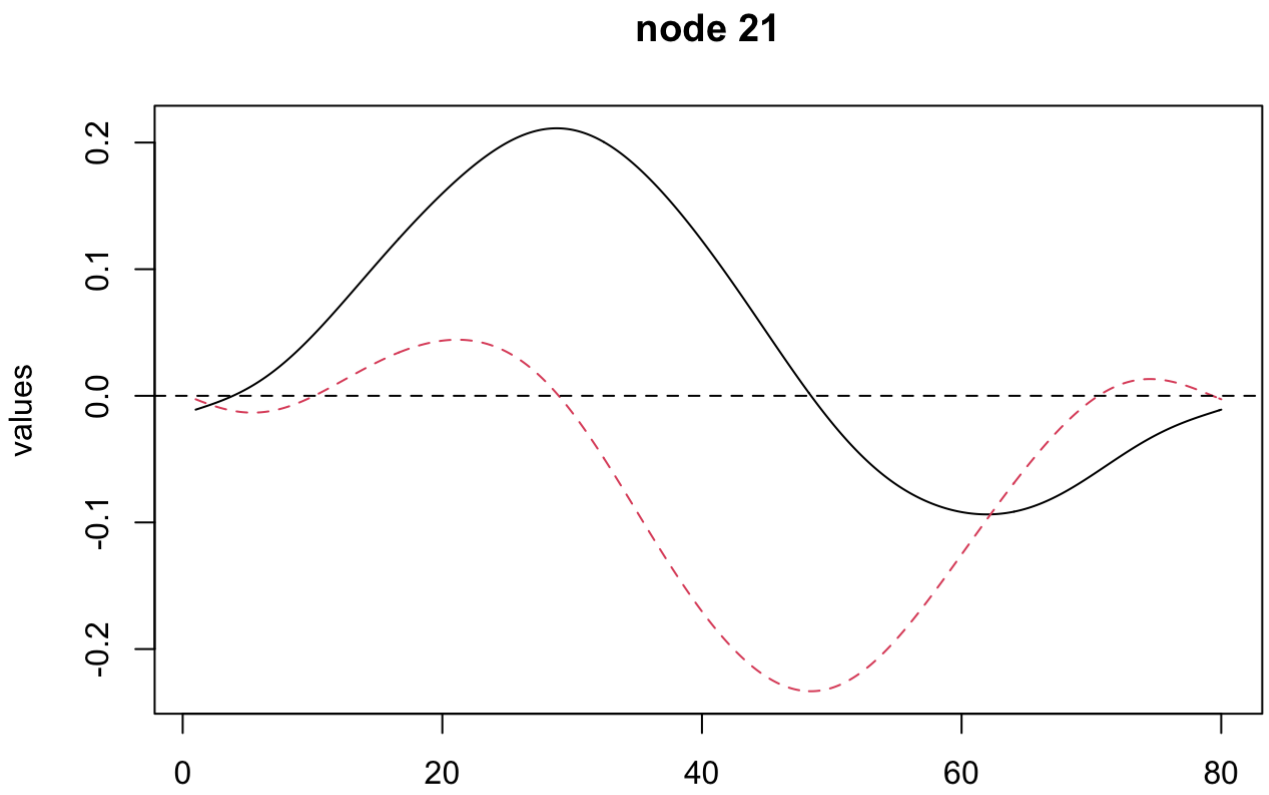

node 16
values
0.20
0.10
0.00
-0.10
0
20
40
60
80

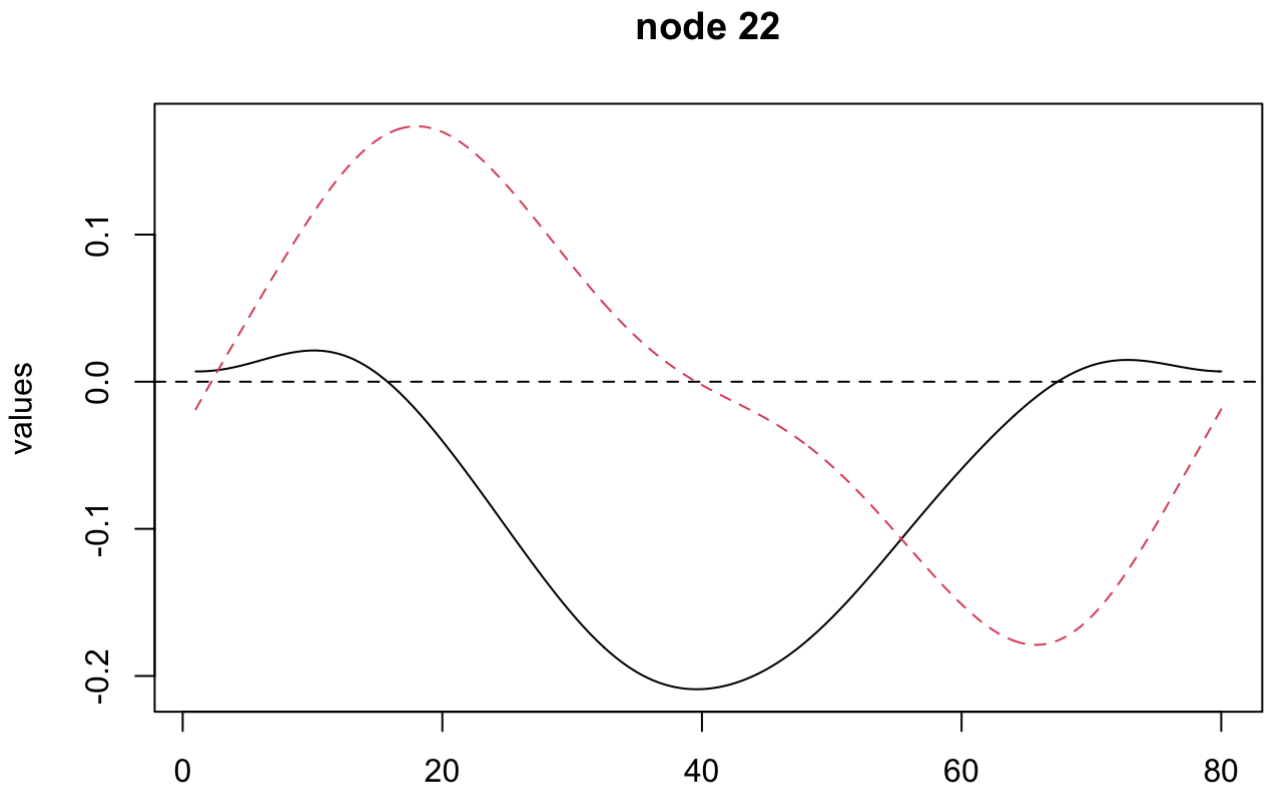

node 17
values
0.2
0.1
0.0
-0.1
0
20
40
60
80

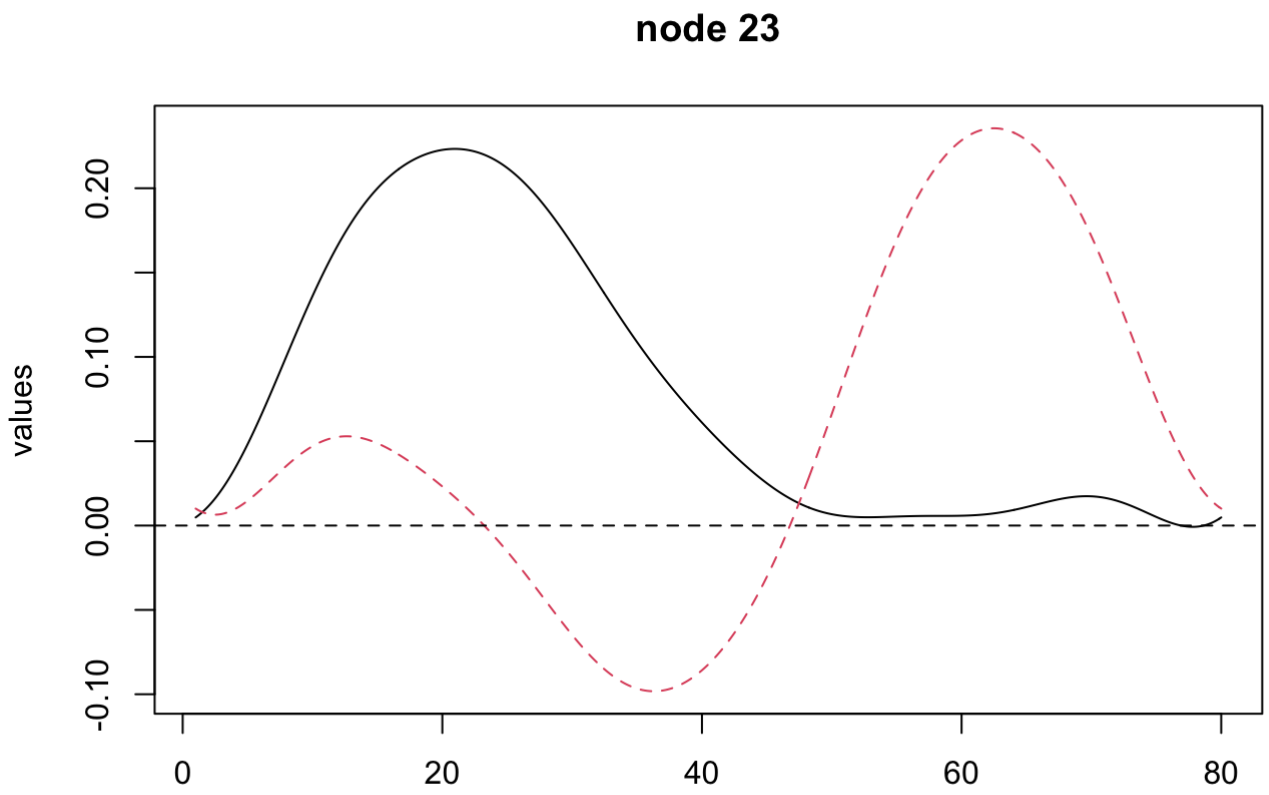

node 18
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

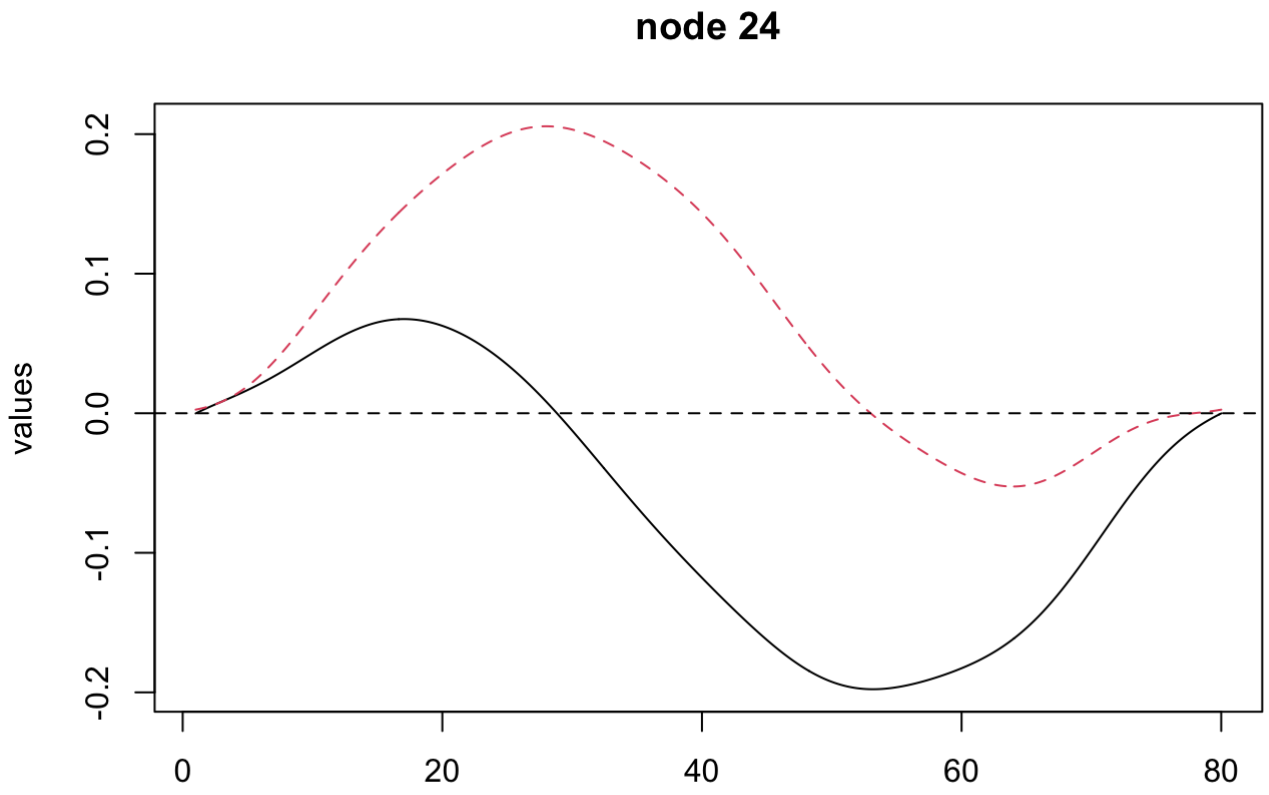

node 19
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

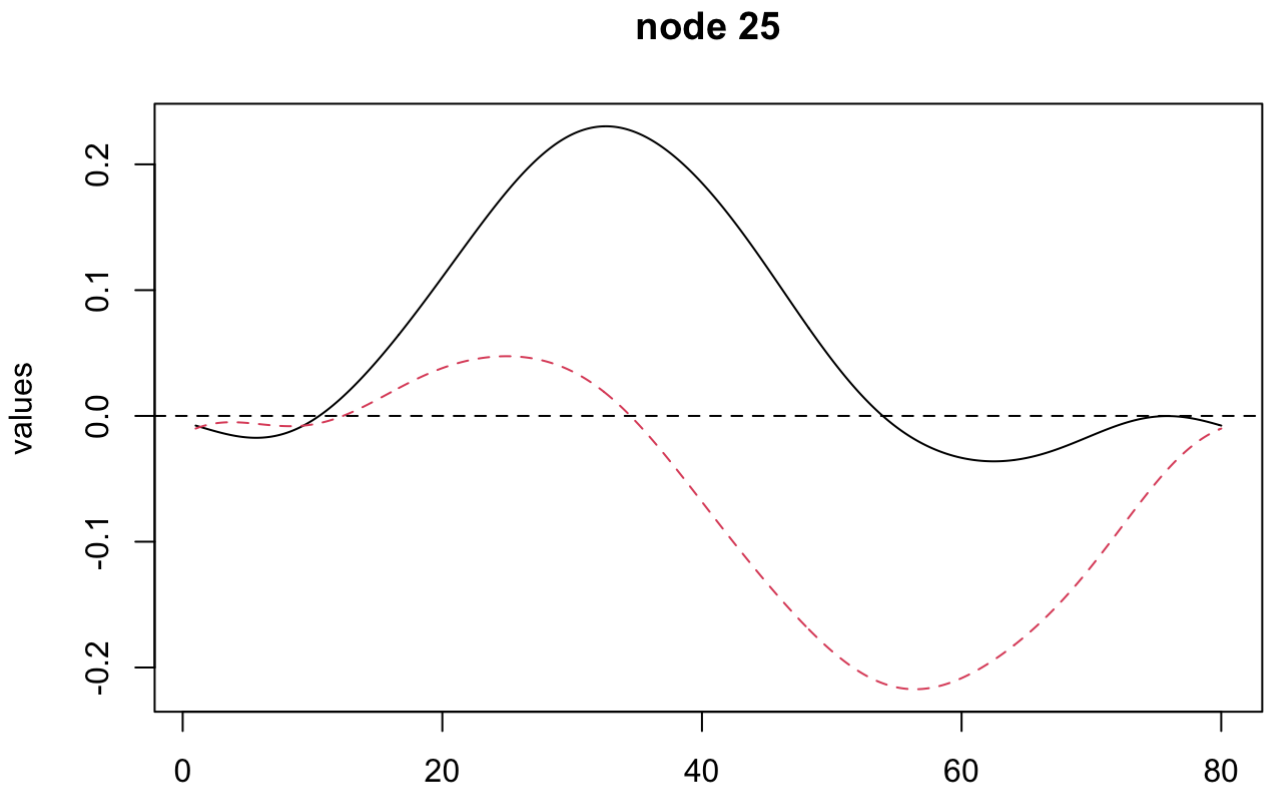

node 20
values
0.2
0.1
0.0
-0.1
0
20
40
60
80

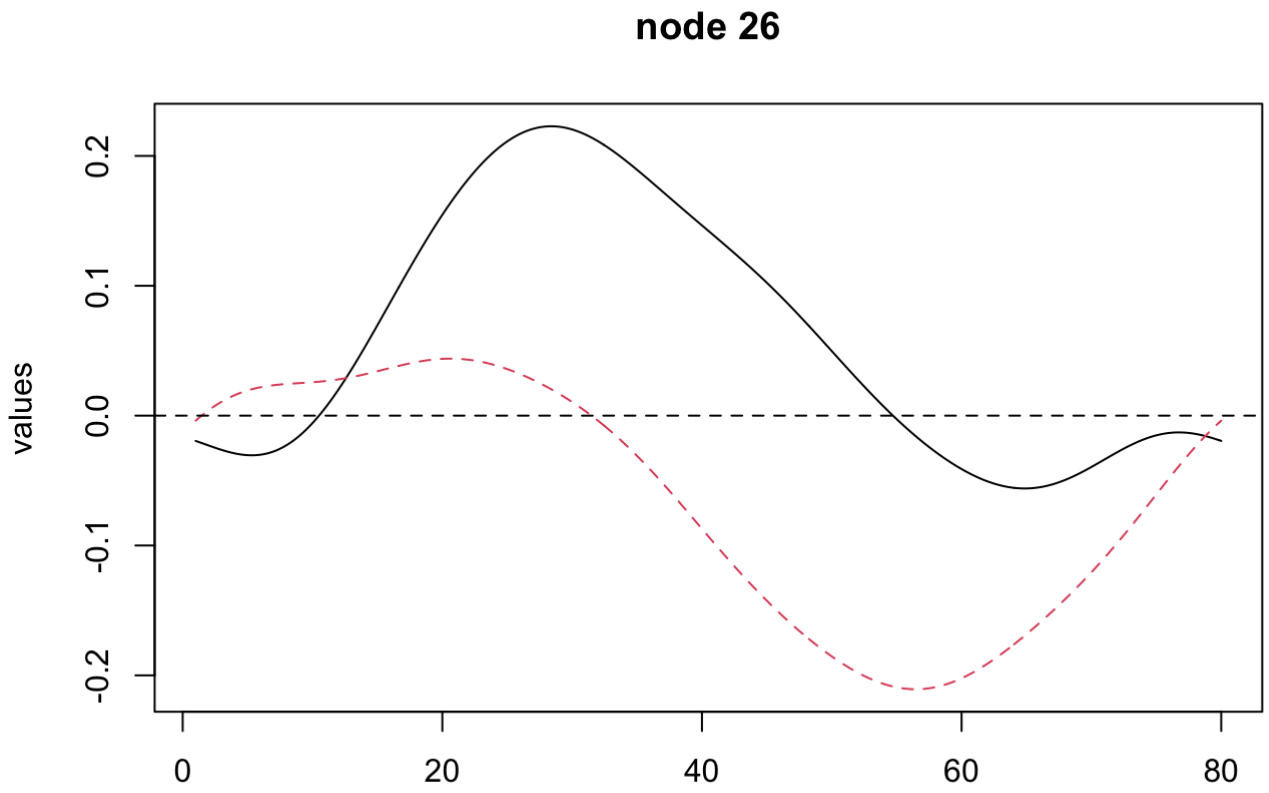

node 21
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

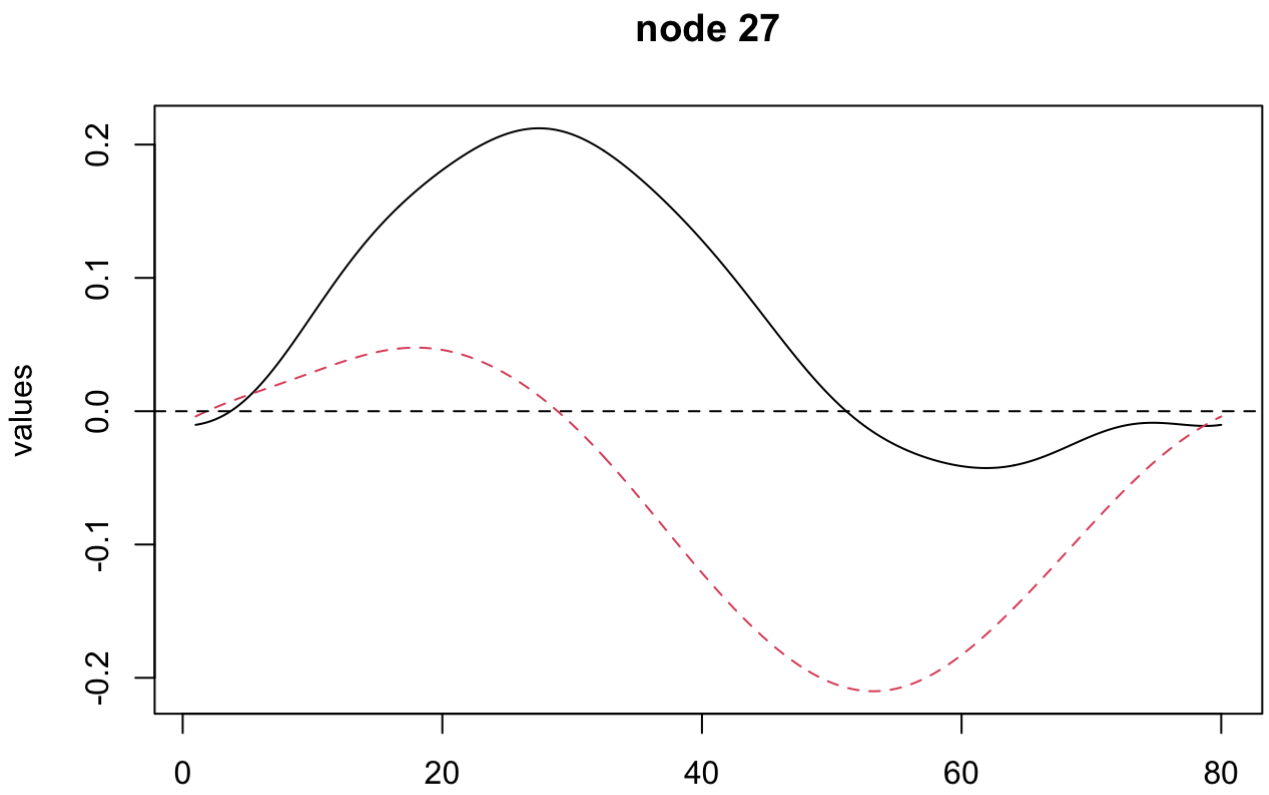

node 22
values
0.1
0.0
-0.1
-0.2
0
20
40
60
80

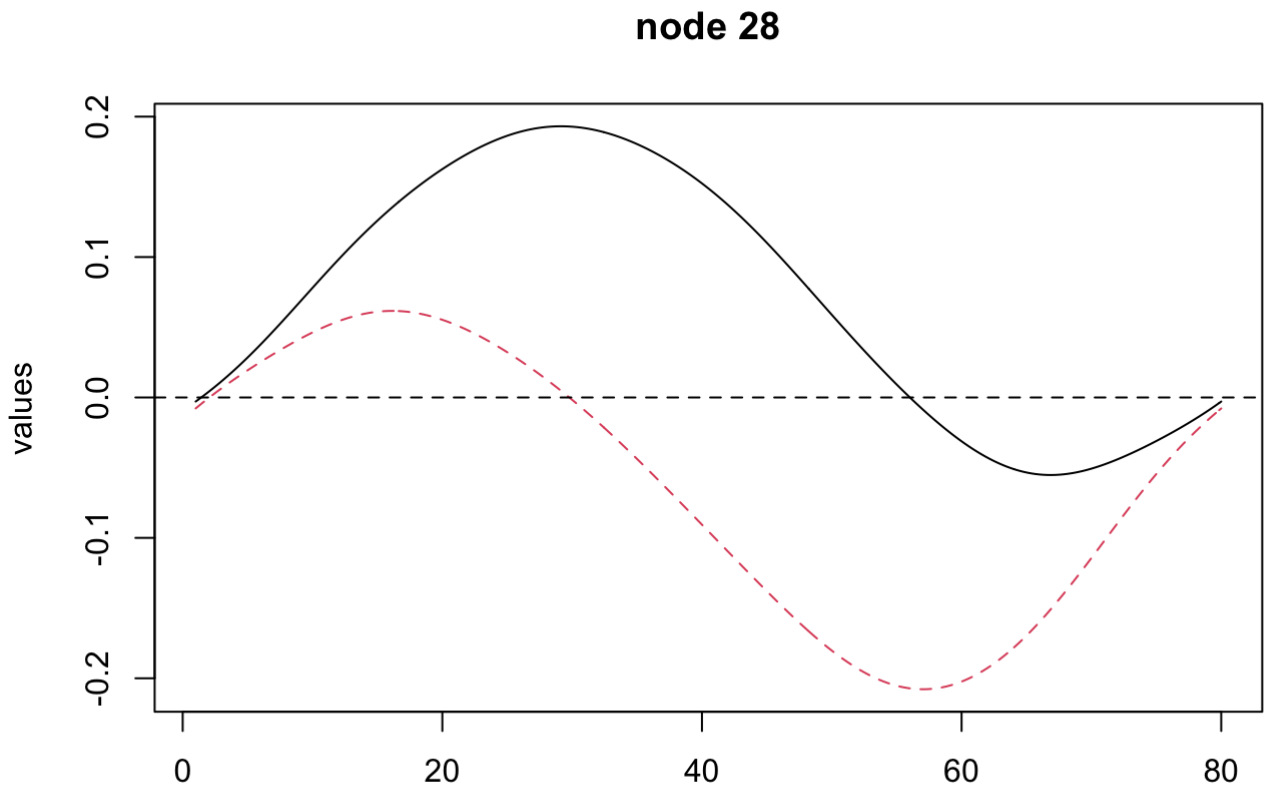

node 23
values
0.20
0.10
0.00
-0.10
0
20
40
60
80

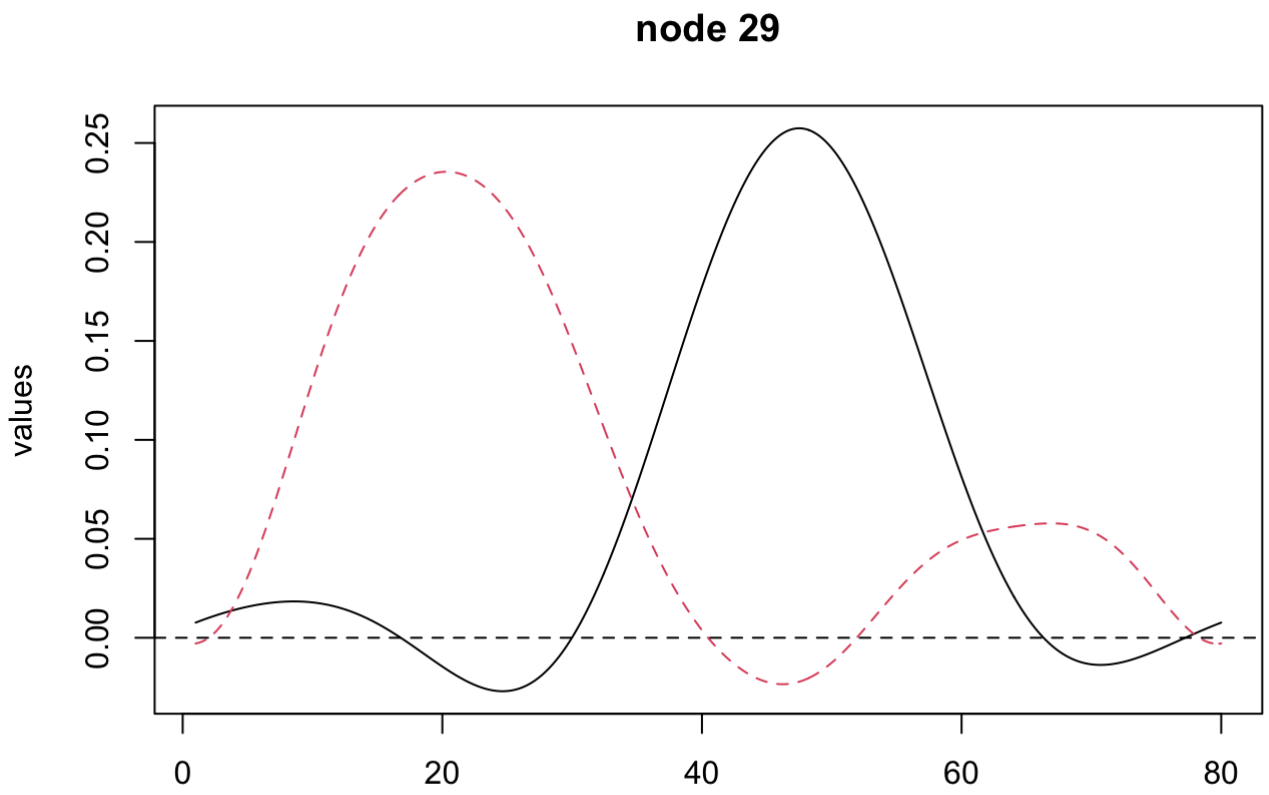

node 24
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

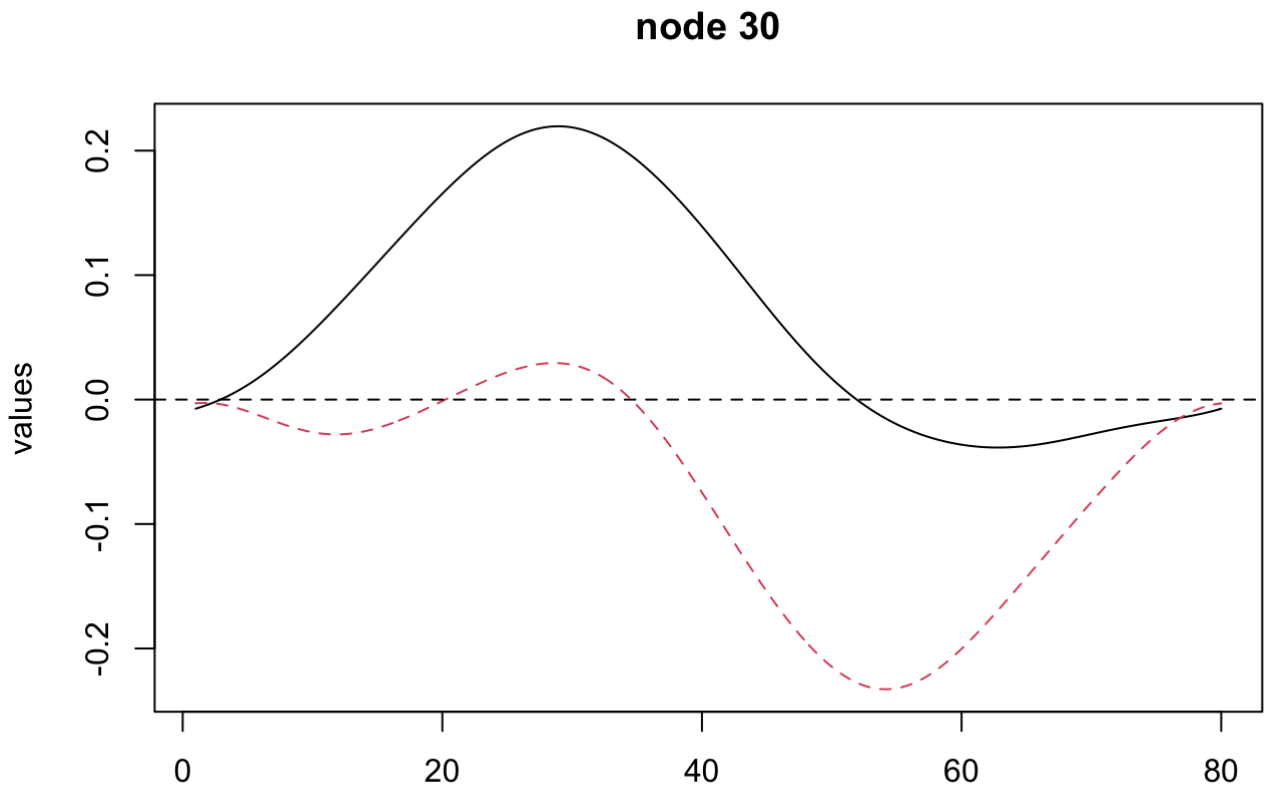

node 25
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

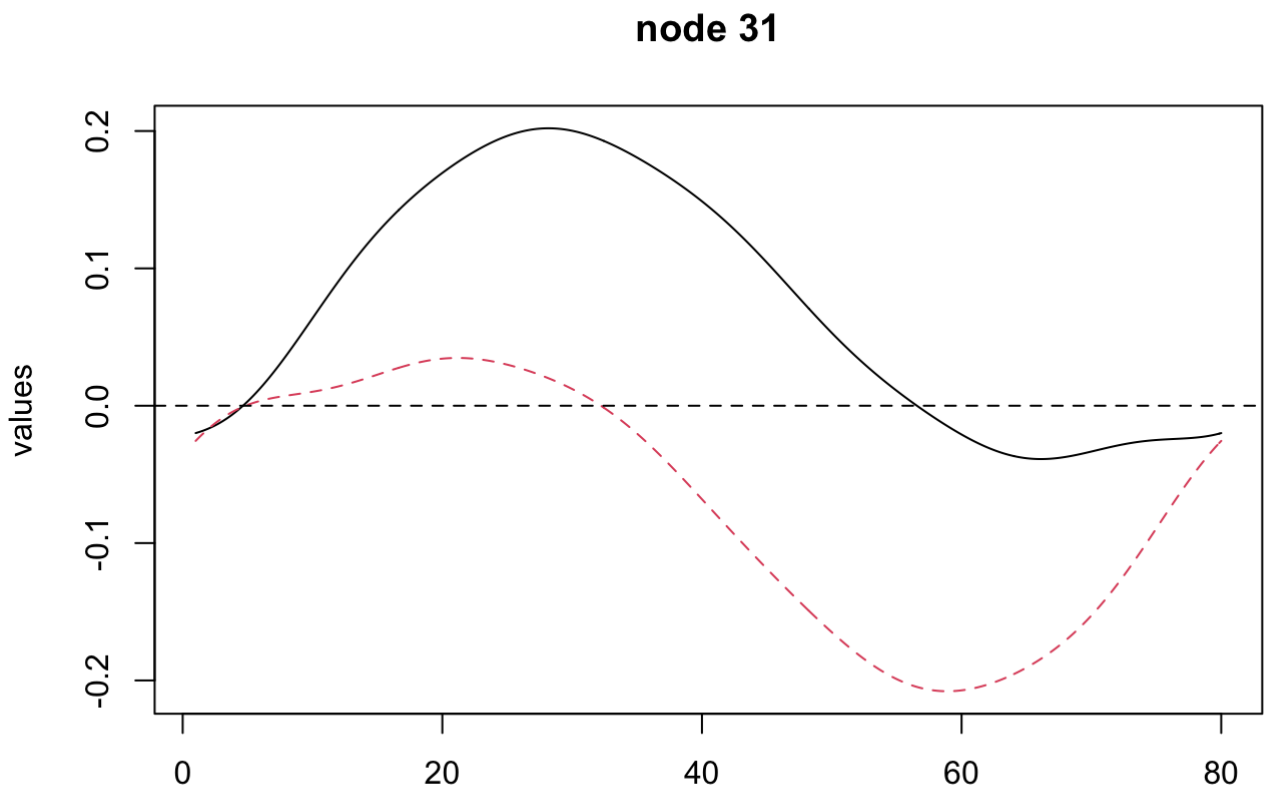

node 26
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80

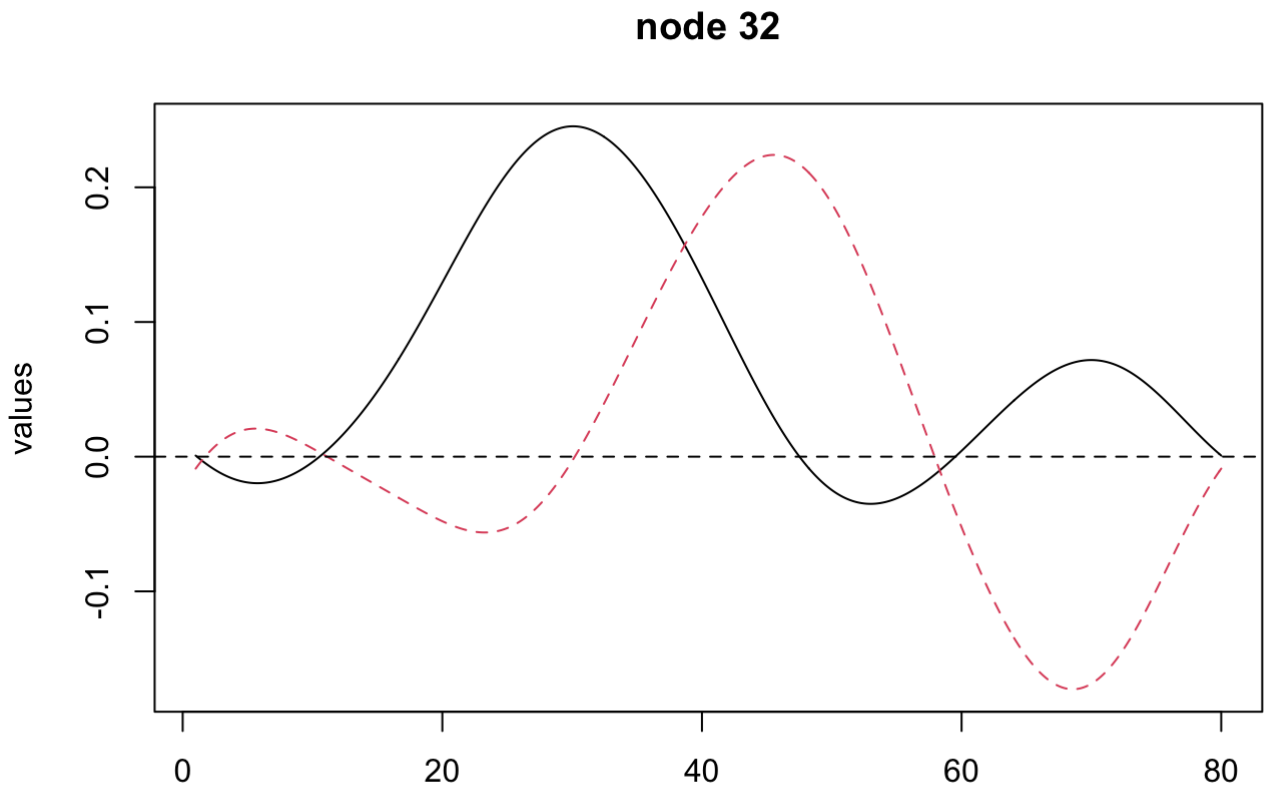

node 27
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80


node 28
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80


node 29
values
0.00
0.05
0.10
0.15
0.20
0.25
0
20
40
60
80


node 30
values
-0.2
-0.1
0.0
0.1
0.2
0
20
40
60
80


node 31
values
0.2
0.1
0.0
-0.1
-0.2
0
20
40
60
80


node 32
values
0.2
0.1
0.0
-0.1
0
20
40
60
80