Function documentations & Examples

Zach Wang

8/19/2020

1. Function Implementations

1.1 function 1: *ReadFile* and attach packages

1.2. function 2: *Wave*

1.3. function 3: *fourier_smooth*

1.4. function 4: *fPCA.nodes*

1.5. function 5: *node.scaler*

1.6. function 6: *row.check*

main script

2. Examples of using each function

2.1 Read data

```
file_loc = '/Users/hanwang/desktop/Git_desktop/Functional_Data_Analysis/data_15may2020.csv'
data_15may2020 <- ReadFile(file_loc, time_subset=c(1:600), node_subset=c(1:32))
```

read in data from the specified file location

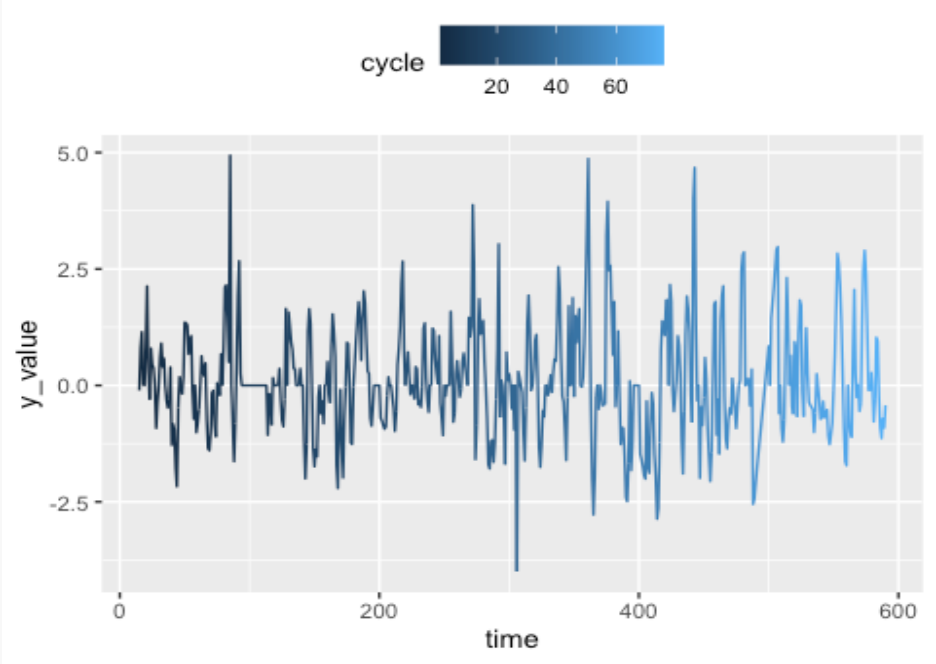
2.2 *Wave* function

Wave function takes the a single node data as input, and transforms the data to N complete sinusoidal cycles. Below are some examples of using Wave function on data of a single node.

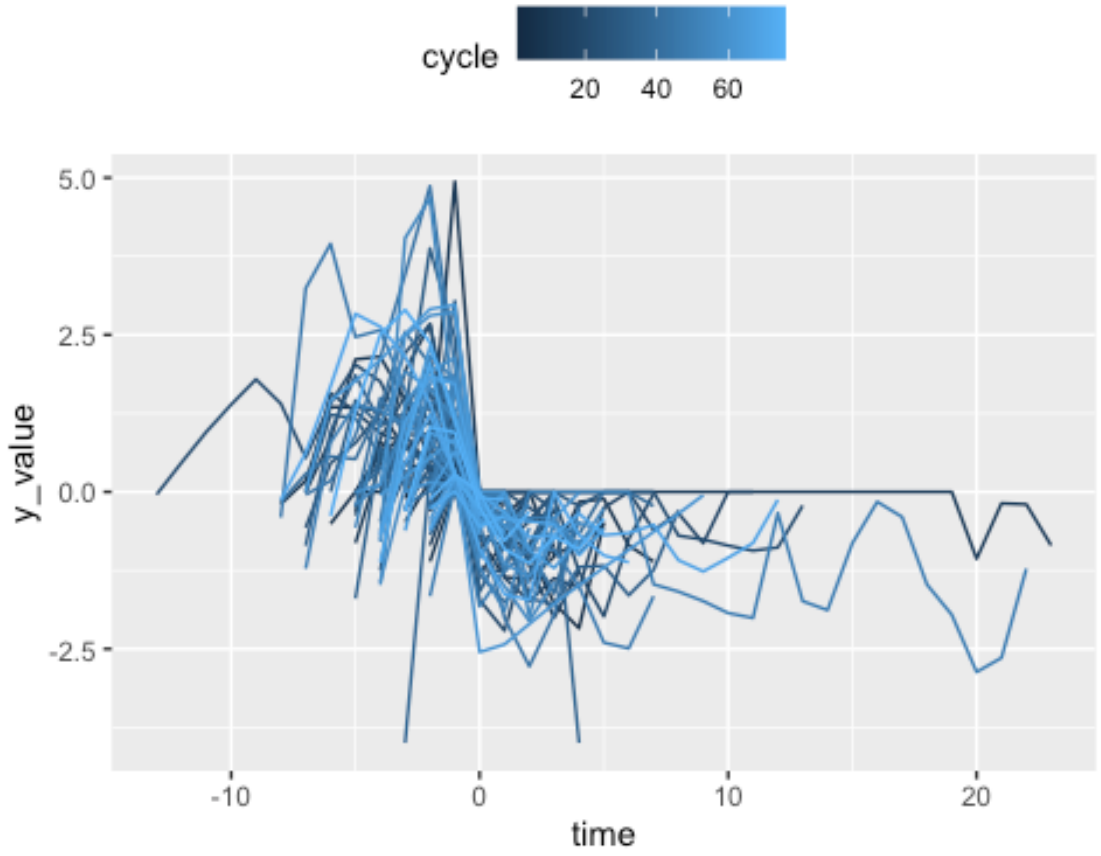
- **Notes:**
  - *Wave* has two arguements:
    - node_data: input takes one column at a time (here is a 600x1 matrix)
    - register: set to 0 if want the result to be the original curves; set to 1 if want the result curves to center around 0

The following 2 plots are generated by the *Wave* function applied on the raw data of node 1 (before smoothing).

```
# extract periodic curves from a single node (before smoothing)
node1 = Wave(data_15may2020[,1], register=0)
ggplot(node1, aes(time, y_value,group=cycle, colour=cycle)) + geom_line() + theme(legend.position="top")
```



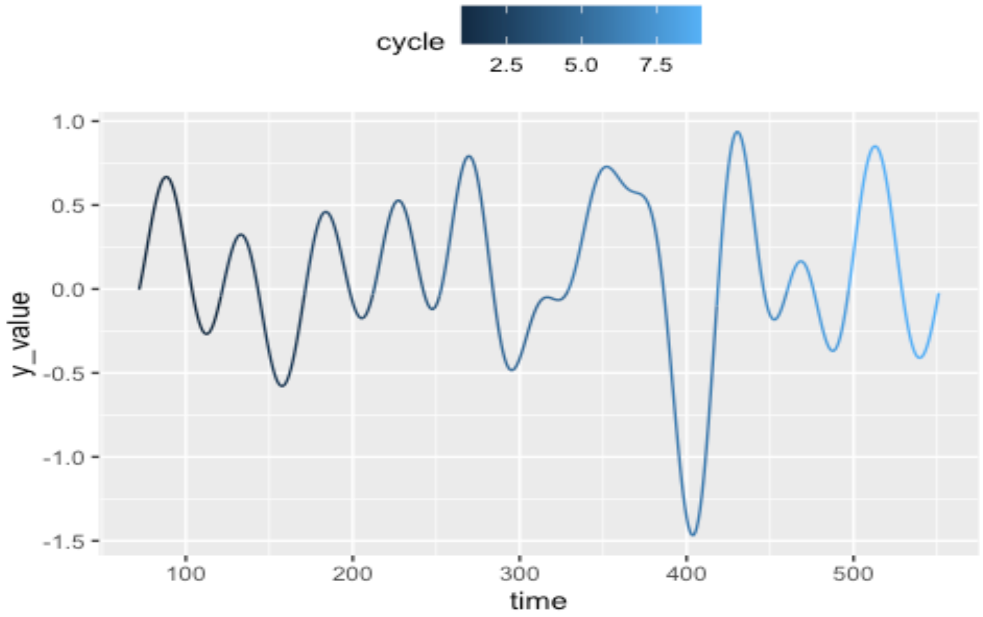
```
node1 = Wave(data_15may2020[,1], register=1)
ggplot(node1, aes(time, y_value,group=cycle, colour=cycle)) + geom_line() + t
heme(legend.position="top")
```



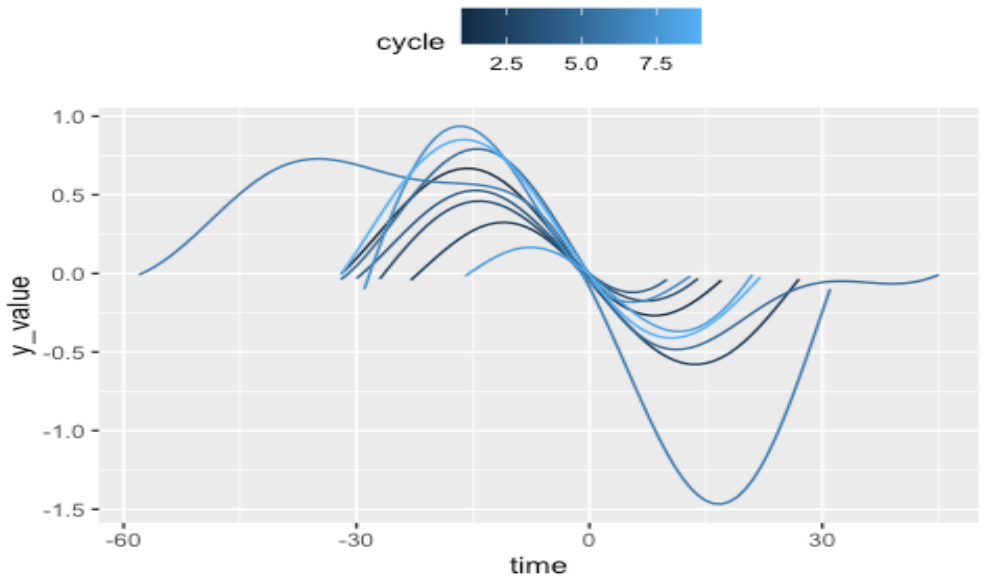
The following 2 plots are generated by *Wave* functions on the transformed data of node 1 (after smoothing). In the examples below, another function, **fourier_smooth** is called first to smooth the raw data using fourier basis functions.

- **Notes:**
  - *fourier_smooth* has four arguments:
    - data_mat: Input data matrix, which should contains the entire data set, but we can use other arguments to select only *"subsets"* or *"Region of interest"* of the data for analysis
    - time_subset: specify a subset of time or all for analysis
    - node_subset: specify a subset of node or all for analysis
    - k: specify the *"nummber of fourier basis functions"* to be used in the smoothing process

```r
# extract periodic curves from a single node (after smoothing)
result_obj <- fourier_smooth(data_mat=data_15may2020, time_subset=c(1:600), n
ode_subset=c(1), k=32)
node1_smoothed = eval.fd(c(1:600),result_obj$fd)
node1_smoothed_extraced = Wave(node1_smoothed, register=0)
ggplot(node1_smoothed_extraced, aes(time, y_value,group=cycle, colour=cycle))
+ geom_line() + theme(legend.position="top")
```




```r
node1_smoothed_extraced = Wave(node1_smoothed, register=1)
ggplot(node1_smoothed_extraced, aes(time, y_value,group=cycle, colour=cycle))
+ geom_line() + theme(legend.position="top")
```



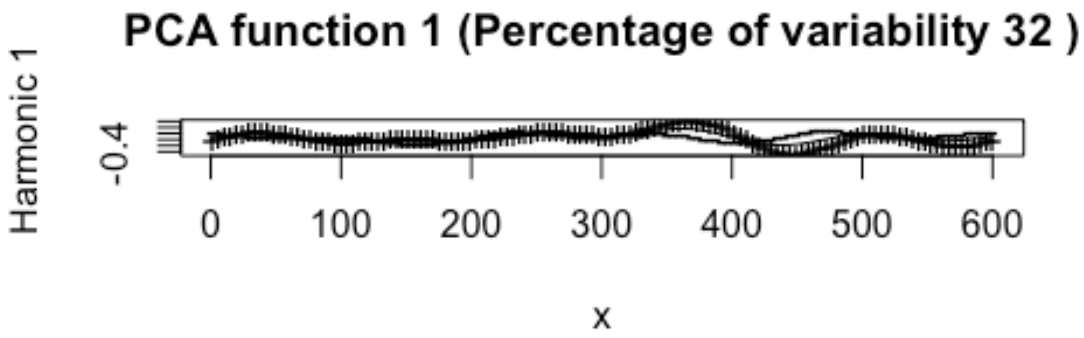
2.3 *fPCA.nodes* function

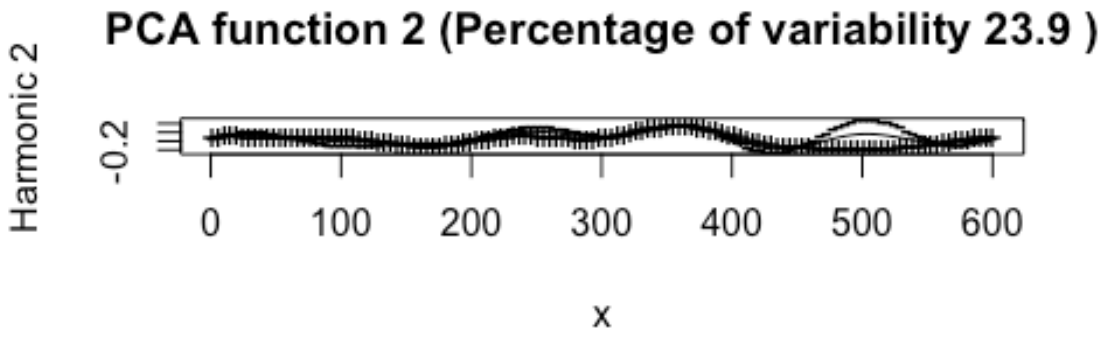
fPCA.nodes function takes input data in matrix form (N rows x M columns), and transform the data to N complete sinusoidal cycles – there has to be more than one column, otherwise PCA won't make sense.

- **Notes:**
  - *fPCA.nodes* has four arguements:
    - data_mat: data in matrix form (N rows x M columns).
    - k: number of fourier basis functions used
    - nharm: number of Principle Components interested
    - plt: set to 1 if want to generate plot; set to 0 if don't want plot

Below are some examples of using **fPCA.nodes** function on the original data.

```
# fPCA on raw data
rotpcalist = fPCA.nodes(data_15may2020, k=11, nharm=2, plt=1)
```





2.4 *node.scaler* & *row.check* functions

node.scaler function is designed to transform a single node curves into T complete sinusoidal cycles (*Wave* function is called). The returned matrix has 3 columns (*cycle*, *time*, *y_value*). The *time* columns will be scaled between range of [-1,1].

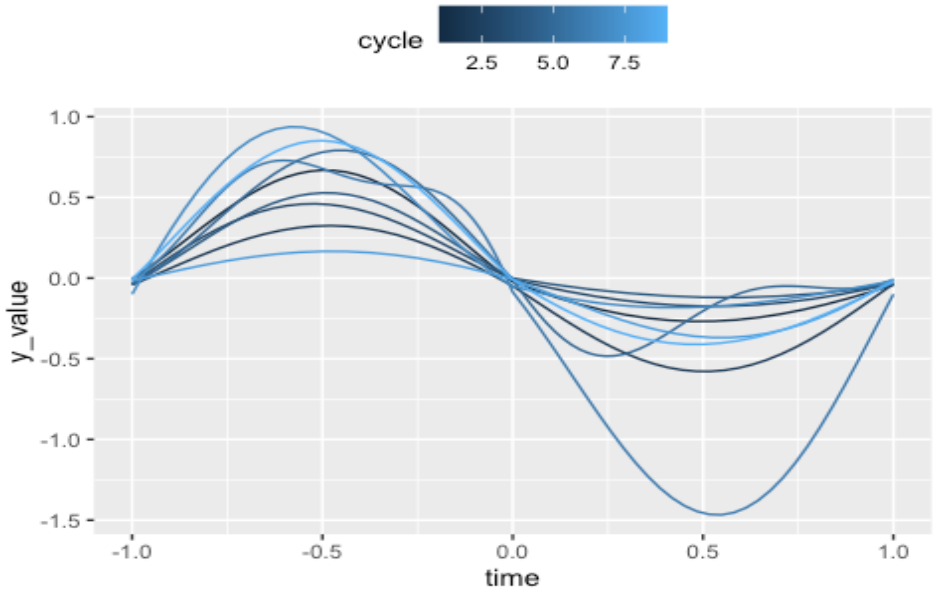
- **Notes:**
  - *node.scaler* has two arguements:
    - data_mat: input data in matrix form (N rows x M columns).
    - node: specify which node will be currently used, since scaler function only look at one node at a time

row.check function is designed to transform the returned dataframe from calling the *node.scaler* function, and most importantly make sure the total number of rows is the same for every column. The result is a 80 x T matrix (the pivot of the matrix from *node.scaler* result), so that each column now represents a different *cycle* and each row stores the *y_value* corresponding to each time point.

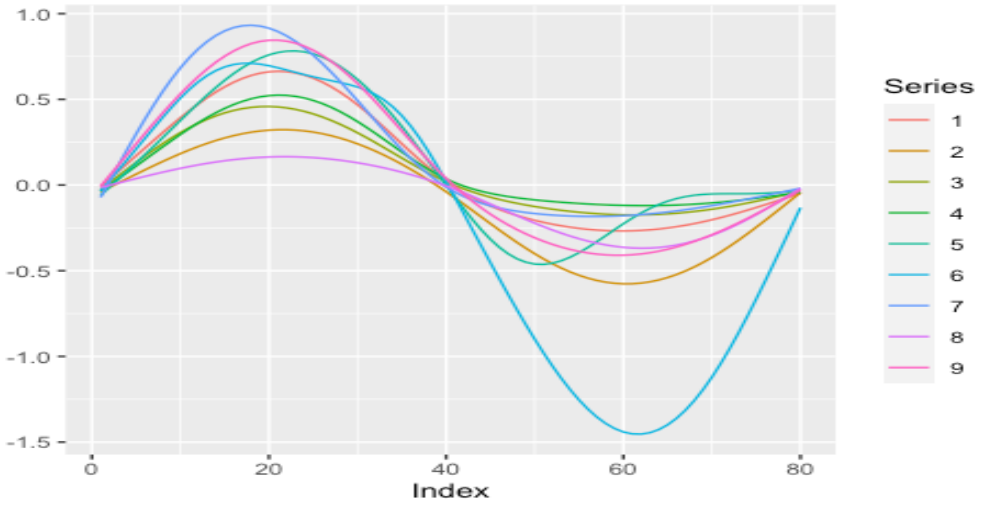
- **Notes:**
  - *row.check* has only one input argument:
    - node_data: the transformed node data from *node.scaler*.

Below are examples of using **node.scaler** together with **row.check** function on a single node. Two plots were generated to show that the result shape of curves after calling the node.scaler and row.check functions are the same.

```
# apply scaler to *single node* and match row numbers
a = node.scaler(data_15may2020, node=c(1))
ggplot(a, aes(time, y_value,group=cycle, colour=cycle)) + geom_line() + theme
(legend.position="top")
```



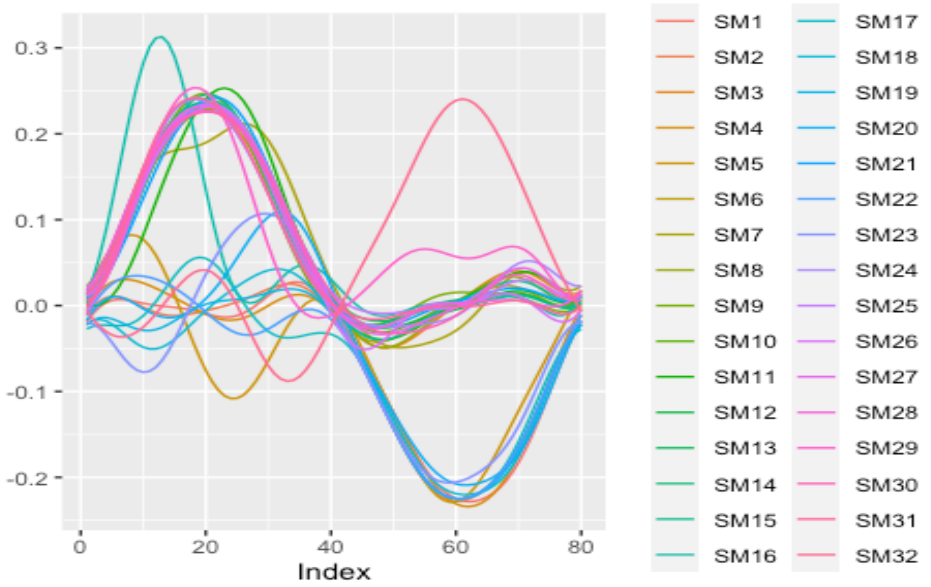
```
b = row.check(node_data=a)
b = read.zoo(b, index='index')
autoplot(b, facet = NULL)
```



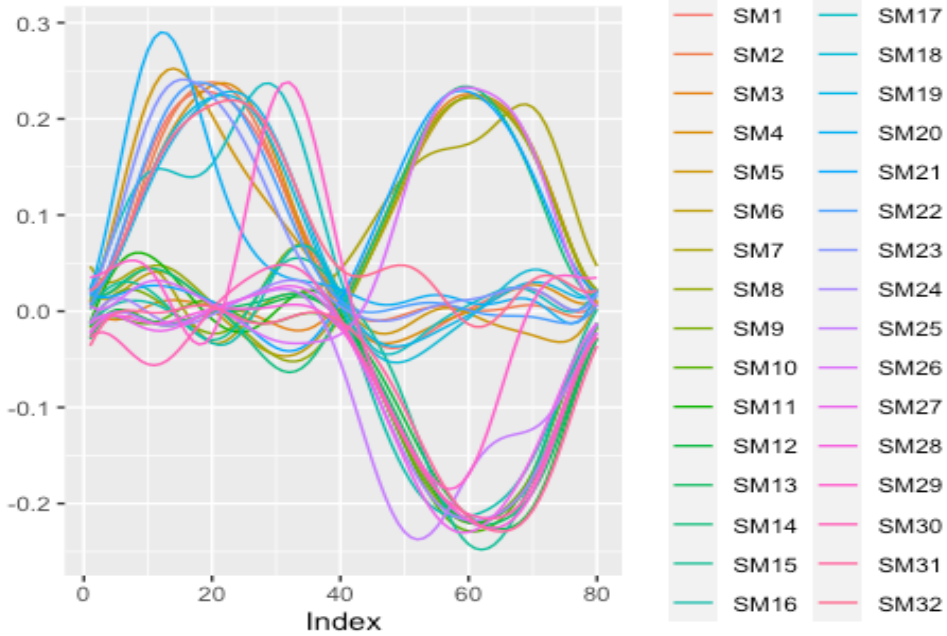
3. Use all the functions defined above to analyze the entire dataset (all nodes, all time points)

```
# fPCA on all nodes
PC1_df = data.frame(matrix(nrow=80))
PC2_df = data.frame(matrix(nrow=80))
mean_df = data.frame(matrix(nrow=80))
for(node in 1:32){
  node.df = node.scaler(data_15may2020, node)
  node.df = as.matrix(row.check(node.df))
  rotpcalist = fPCA.nodes(data_mat=node.df, k=11, nharm=2, plt=0)
  # PC1 & PC2
  harmfd <- rotpcalist[[1]]
  basisfd <- harmfd$basis
  rangex <- basisfd$rangeval
  x <- seq(rangex[1], rangex[2], length = harmfd$basis$rangeval[2])
  fdmat <- eval.fd(x, harmfd)
  meanmat <- eval.fd(x, rotpcalist$meanfd)
  PC1_df[,ncol(PC1_df)+1]=fdmat[,1]
  PC2_df[,ncol(PC2_df)+1]=fdmat[,2]
  mean_df[,ncol(mean_df)+1]=meanmat
}
PC1_df = data.frame(PC1_df[,2:(ncol(PC1_df))])
names(PC1_df)=colnames(data_15may2020)
PC2_df = data.frame(PC2_df[,2:(ncol(PC2_df))])
names(PC2_df)=colnames(data_15may2020)
mean_df = data.frame(mean_df[,2:(ncol(mean_df))])
names(mean_df)=colnames(data_15may2020)
```

```r
## plot
### PC1
z_1 = read.zoo(PC1_df, index='index')
### PC2
z_2 = read.zoo(PC2_df, index='index')
### Mean Functional Curves
z_3 = read.zoo(mean_df, index='index')
autoplot(z_1, facet = NULL)
```




```r
autoplot(z_2, facet = NULL)
```



```
autoplot(z_3, facet = NULL)
```