$CONNECTOR^n$ User Manual

Sirovich Roberta, Pernice Simone, Frattarola Marco

2025

Abstract

Contents

1 Data importing by files

The analysis starts from two distinct files:

- A file reporting the discretely sampled curve data. The longitudinal data must be reported in terms of time t and y-values y. Each sample is described by four columns: the first column, named subjID, contains a list of subject IDs. The second one, named value, contains a list of y values. The third one, named time, includes the lags. The last one, named measureID, contains a list of the corresponding measure names. Order is not mandatory. Accepted formats are: Excel, CSV, or tbl_df. See Figure 1.

- A file containing the annotations associated with the sampled curves. The first column reports the subjID, while the other columns report the features relevant for the analysis, one per column. Note that the subjID sample must contain the same ID names that appear in the file of the sampled curves. Accepted formats: Excel, CSV, or tbl_df. See Figure 2.

time	value	measureID	subjID
-10.0	207.416776	Parabola	13
-10.0	274.225908	Hyperbola	26
-10.0	28.585194	Sine	3
-10.0	-28.585194	Sine	10
-10.0	28.585194	Sine	15
-10.0	28.585194	Sine	25
-10.0	-28.585194	Sine	26
-10.0	-98.167400	Cosine	13
-10.0	98.167400	Cosine	14
-9.9	-72.514435	Parabola	18
-9.9	249.594127	Hyperbola	16
-9.9	224.939112	Hyperbola	28
-9.9	-23.661854	Sine	16

Figure 1: Example for TimeSeriesFile

subjID	gender	age	treatment_group	baseline_weight	height	comorbidity
1	M	23	Treatment A	74.1	184.1	Diabetes
2	F	77	Treatment B	54.3	169.0	None
3	M	38	Control	81.0	150.1	None
4	M	50	Control	69.0	161.3	None
5	F	47	Treatment A	57.9	183.1	Obesity
6	M	71	Treatment B	56.4	191.9	None
7	F	58	Treatment B	51.4	166.9	Diabetes
8	M	65	Treatment A	99.4	181.5	None
9	M	44	Control	94.4	187.8	Hypertension
10	F	65	Control	90.5	161.5	None

Figure 2: Example for AnnotationFile

2 Import

The two files are imported by the **DataImport** function, whose arguments are the file names. In this example, *TimeSeriesFile* and *AnnotationFile*.

```
Data<-DataImport(TimeSeries, Annotations)
###############################
Data loaded...
Number of curves:120
Number of distinct measures:4
Average length:14.91667
Measure with highest length:# A tibble: 1 x 2
  curvesID     nTimePoints
  <chr>              <int>
1 26_Hyperbola          24

Measure with lowest length:# A tibble: 2 x 2
  curvesID     nTimePoints
  <chr>              <int>
1 23_Hyperbola           9
2 9_Cosine               9
```

A CONNECTORData object is created:

```
    > str(Data)
Formal class 'CONNECTORData' [package ".GlobalEnv"] with 4 slots
  ..@ curves     : tibble [1,790 x 5] (S3: tbl_df/tbl/data.frame)
  .. ..$ subjID   : Factor w/ 30 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ measureID: chr [1:1790] "Parabola" "Parabola" "Parabola" "Parabola" ...
  .. ..$ time     : num [1:1790] -6.7 -5.8 -5 -3.6 -1.3 ...
  .. ..$ curvesID : chr [1:1790] "1_Parabola" "1_Parabola" "1_Parabola" "1_Parabola" ...
  .. ..$ value    : num [1:1790] 160.4 130.2 66.1 41.9 30.4 ...
  ..@ dimension  : tibble [120 x 2] (S3: tbl_df/tbl/data.frame)
  .. ..$ curvesID   : chr [1:120] "10_Cosine" "10_Hyperbola" "10_Parabola" "10_Sine" ...
  .. ..$ nTimePoints: int [1:120] 13 14 10 14 14 17 17 13 11 10 ...
  ..@ annotations: tibble [30 x 7] (S3: tbl_df/tbl/data.frame)
  .. ..$ subjID          : Factor w/ 30 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ gender          : chr [1:30] "M" "F" "M" "M" ...
  .. ..$ age             : num [1:30] 23 77 38 50 47 71 58 65 44 65 ...
  .. ..$ treatment_group: chr [1:30] "Treatment A" "Treatment B" "Control" "Control" ...
```

```
  .. ..$ baseline_weight: num [1:30] 74.1 54.3 81 69 57.9 56.4 51.4
    99.4 94.4 90.5 ...
  .. ..$ height         : num [1:30] 184 169 150 161 183 ...
  .. ..$ comorbidity    : chr [1:30] "Diabetes" "None" "None" "None
  " ...
  ..@ TimeGrids  :List of 4
  .. ..$ Cosine   : num [1:179] -10 -9.9 -9.6 -9.5 -9.4 -9.2 -9.1
  -9 -8.9 -8.8 ...
  .. ..$ Hyperbola: num [1:190] -10 -9.9 -9.8 -9.7 -9.6 -9.5 -9.4
  -9.3 -9.2 -9.1 ...
  .. ..$ Parabola : num [1:181] -10 -9.9 -9.8 -9.7 -9.5 -9.4 -9.3
  -9.2 -9.1 -9 ...
  .. ..$ Sine     : num [1:176] -10 -9.9 -9.8 -9.7 -9.6 -9.5 -9.4
  -9.3 -9.2 -9.1 ...
```

The components of CONNECTORData are:

- curves: a tibble similar to TimeSeries, but with the addition of curvesID, which is the concatenation of SubjID and measureID.
- dimensions: a tibble that contains the number of points for each curve.
- annotations: a tibble that contains data imported from AnnotationFile.
- TimeGrids: a list containing the time grids for each measure.

3 Data visualization

The PlotTimeSeries function generates the plot of the sampled curves, coloured by the selected feature from the AnnotationFile. Figure reports the line plot generated by the following code:

```
PlotTimeSeries(Data, feature="treatment_group")
```

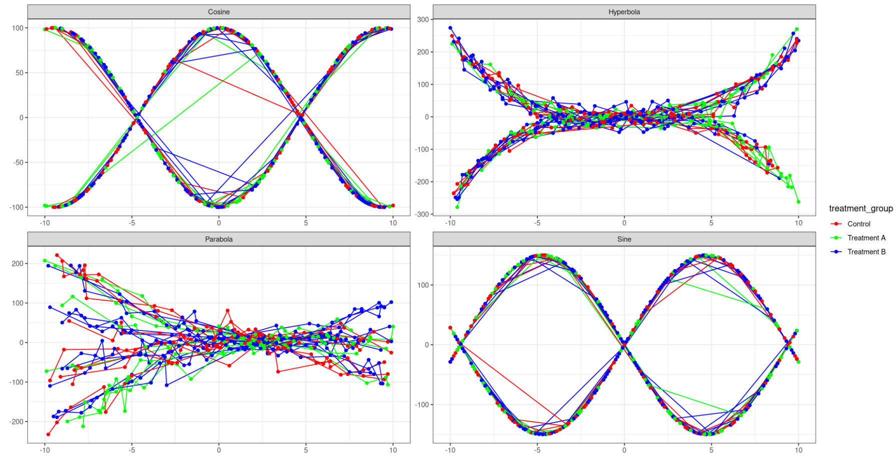

Figure 3: Example for PlotTimeSeries

The DataVisualization function plots the time distribution helping in the inspection of the sparsity of the time points.

```
DataVisualization(Data)
```

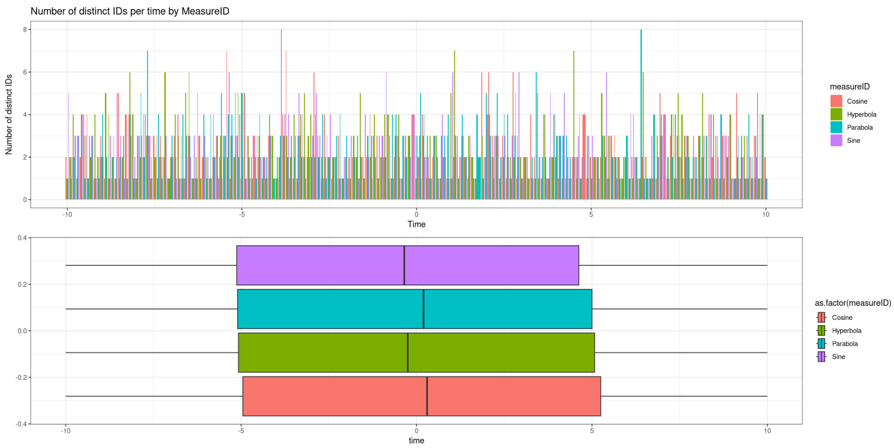

Figure 4: Example for DataVisualization

Setting large as true we can also see the time heat map

```
DataVisualization(Data, large=T)
```

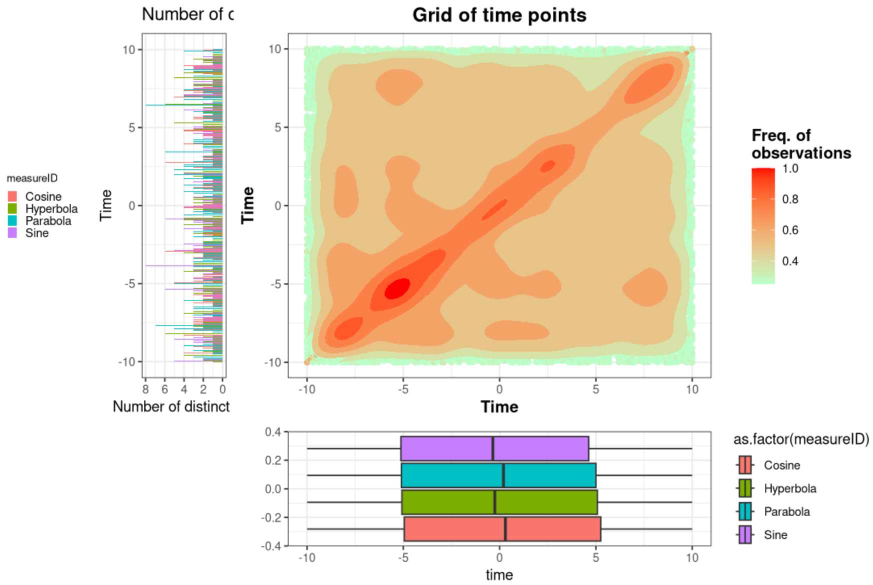

Figure 5: Example for DataVisualization large=T

The DataTruncation function have been developed to truncate the time series at specific time value.

```
DataTruncation(Data, feature = "gender",measure = "Parabola",
  truncTime = 5)
```

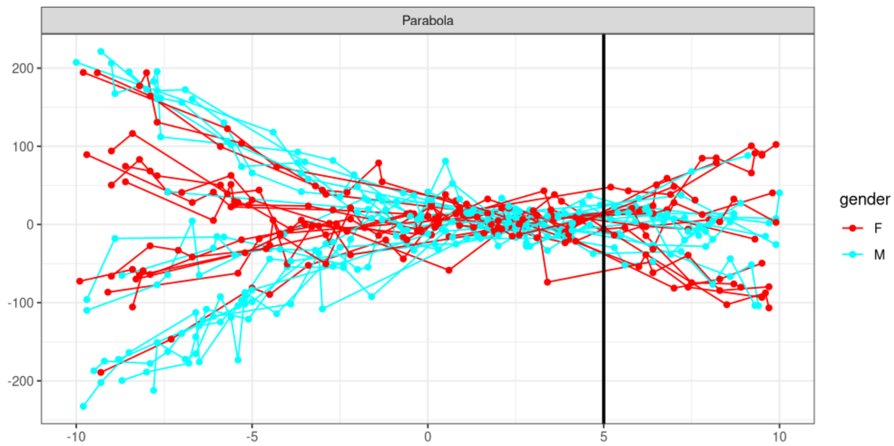

Figure 6: Example for DataTruncation

4 Model Selection Tools

Before running the fitting and clustering approach, we have to properly chose the two free parameters:

- the spline basis dimension, p;
- the number of clusters, G.

We developed several functions to enable the user to properly set the free parameters.

4.1 The spline basis dimension - p

The dimension of the spline basis can be chose by exploiting the *BasisDimensionChoice* function, by taking the $p \in [p_{\min}, p_{\max}]$ value corresponding to the largest cross-validated likelihood, as proposed in (James, Hastie, and Sugar 2000), where the interval of values $[p_{\min}, p_{\max}]$ is given by the user. In particular, a ten-fold cross-validation approach is implemented: firstly the data are split into 10 equal-sized parts, secondly the model is fitted considering 9 parts and the computation of the log-likelihood on the excluded part is performed. A plot is returned: The CrossLogLikePlot return the plot of the mean tested log-likelihoods versus the dimension of the basis, see Figure 7. Each gray dashed line corresponds to the cross-log-likelihood values obtained on different test/learning sets and the solid black line is their mean value. The resulting plot could be used as a guide to choose the largest cross-validated likelihood. Specifically, the optimal value of p is generally the smallest ensuring the larger values of the mean cross-log-likelihood function. CossLogLikePlot is a list containing a plot for each measure

```
CrossLogLikePlot<-BasisDimensionChoice(Data, p=2:10)
```

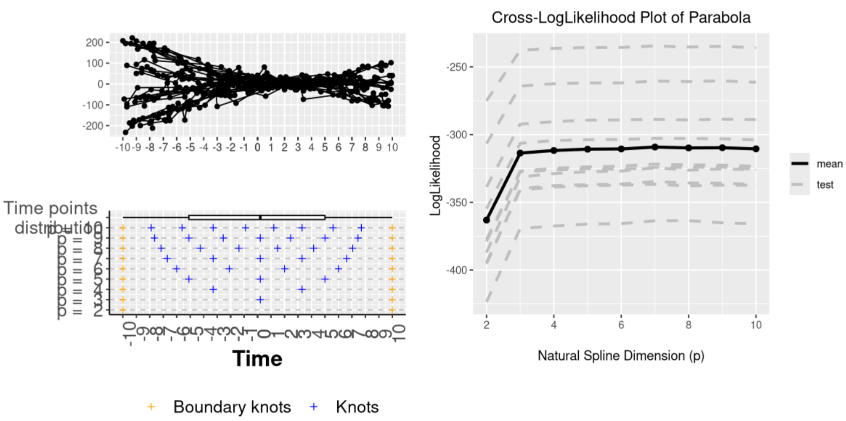

Figure 7: Example for the first element of BasisDimensionChoice

```
#set p
p<-3
```

In our example, the optimal value of p is 3.

4.2 The number of clusters - G

Using ClusterAnalysis we can choose which G is better based on TT, fDB and silhouette score

```
clusters<-ClusterAnalysis(Data, G=2:6, p=c(4,4,6,6), runs=100)
```

The function IndexesPlotExtrapolation

```
IndexPlotExtrapolation(clusters)
```

generate one plot reported in Figure 8. In details, the fDB, silhouette and tightness indexes are plotted for each value of G. Each value of G is associated with a violin plot created by the distribution of the index values collected from the runs performed. The stars indicates maximum frequence, minimum fDB and maximum silhouette score.

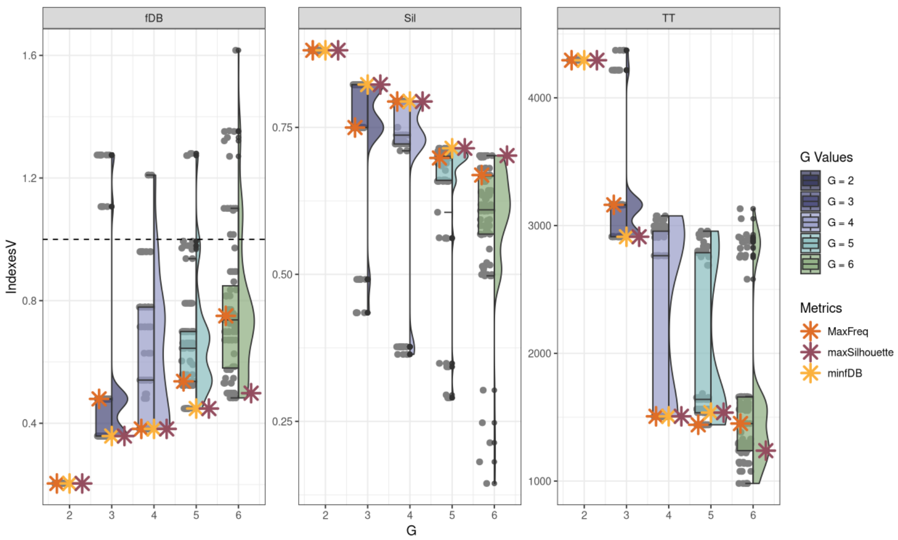

Figure 8: Example for IndexPlotExtrapolation

Then we can set our prefered setup by *ConfigSelection*. In our case the best one is G=2

```
Set<-ConfigSelection(clusters, G=2, "MinfDB")
```

Then we can plot with *IndexPlotExtrapolation2* for each measure all clusters with their mean, coloring each curve by a feature

```
IndexPlotExtrapolation2(Data, ConfigChoosen=Set,KData =
clusters$KData, feature="comorbidity")
```

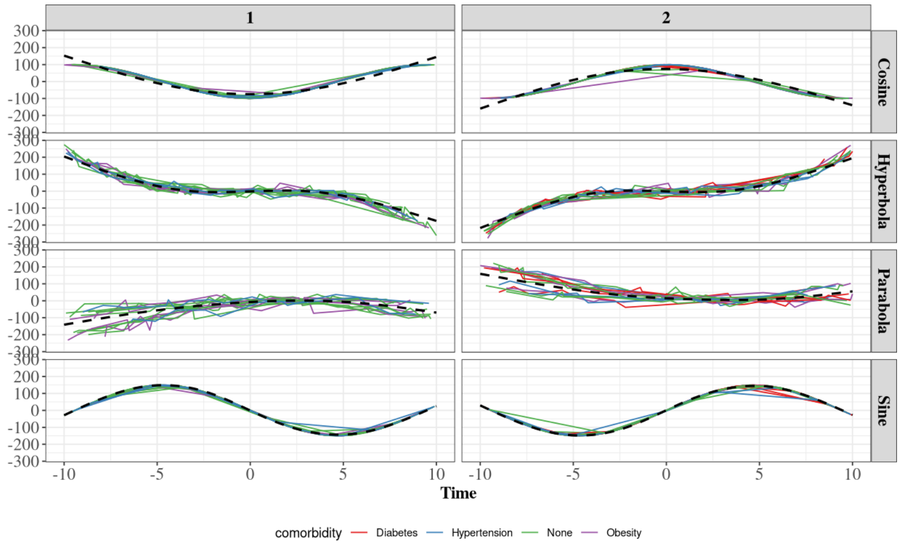

Figure 9: Example for IndexPlotExtrapolation2

You can also plot a Discriminant plot with *DiscriminantPlot*. If the clustering division can be reduced using PCA, the plot dimensions will be automatically adjusted.

```
DiscriminantPlot(Data, ConfigChoosen=Set, KData=clusters$KData,
feature="gender")
```

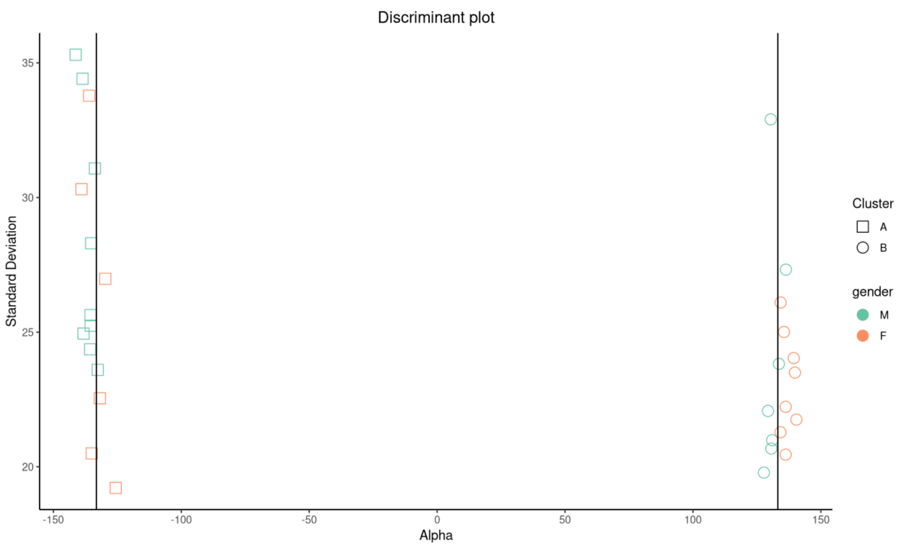

Figure 10: Example for DiscriminantPlot

You can also see entropy and silhouette score for each subject

```
SilEntropy(Set, clusters)
```

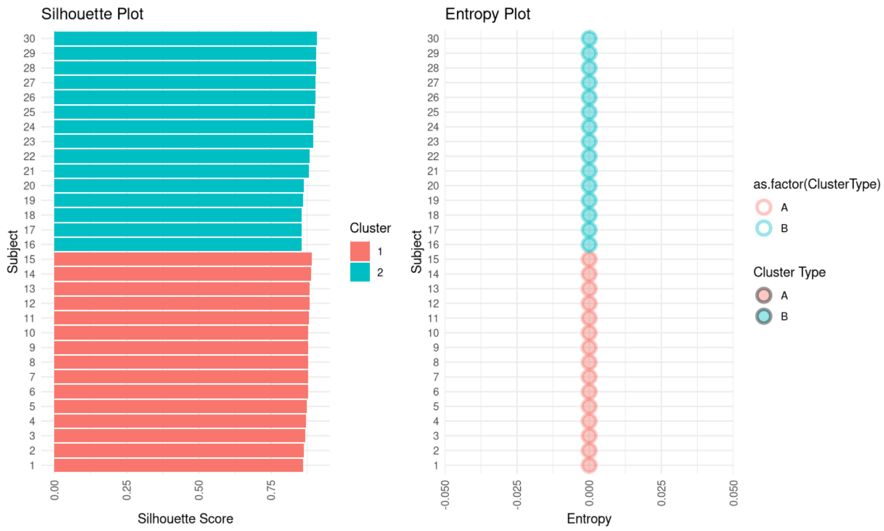

Figure 11: Example for SilEntropy

5 SplinePlot

We can recostruct splines using our data as we can see in figure

```
splinePlot(KData=clusters$KData, ConfigChosen = Set)[1]
```

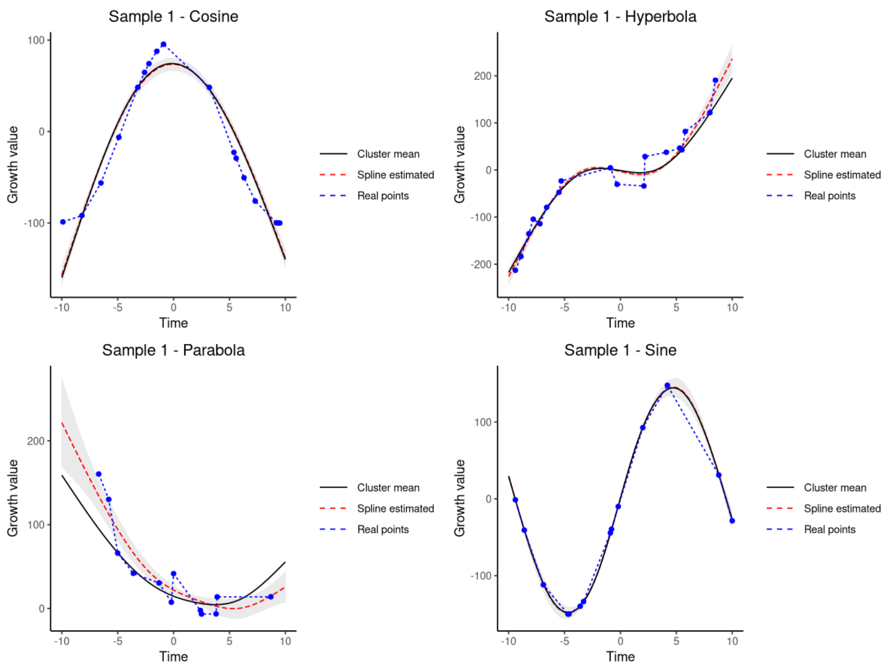

Figure 12: Example for SplinePlot

6 MaximumDiscriminationFuncion

With

```
MaximumDiscriminationFunction(ConfigChoosen = Set, KData =
clusters$KData)
```

we can plot the next figures:

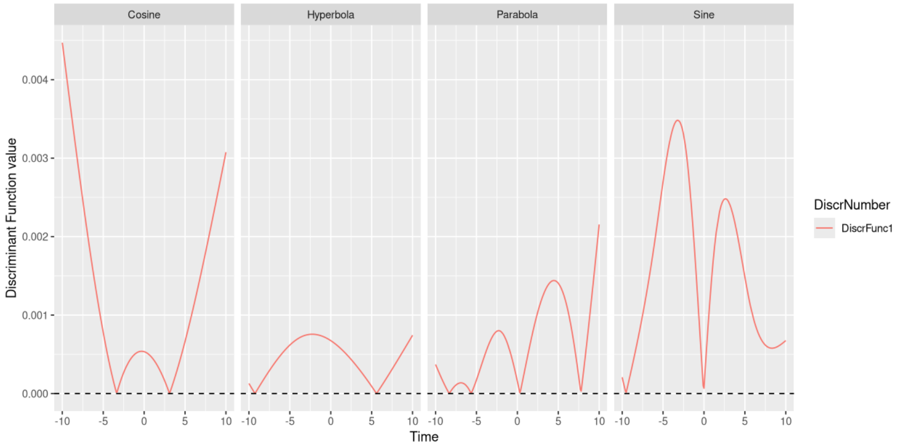

Figure 13: Example for MaximumDiscriminationFuncion[1]

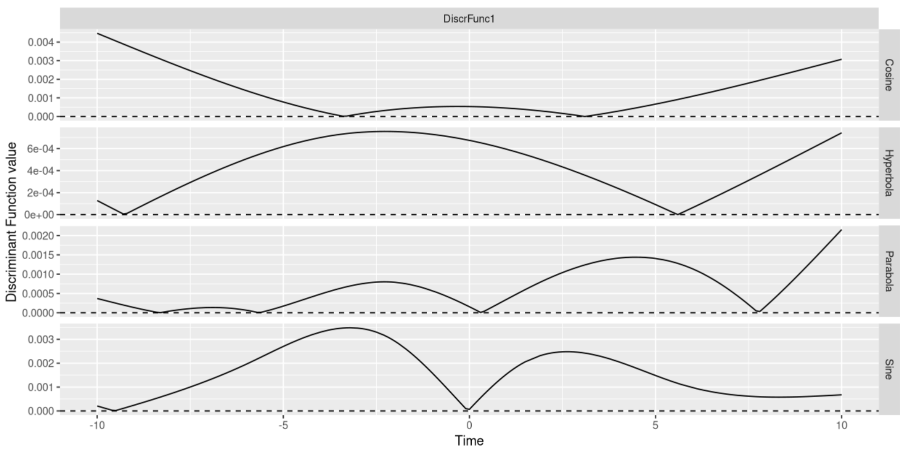

Figure 14: Example for MaximumDiscriminationFuncion[2]