An Introduction to `connector`

Simone Pernice, Roberta Sirovich and Francesca Cordero

february 11, 2020

Contents

Introduction

Connector is built on the model-based approach for clustering functional data presented in (James and Sugar 2003). Such method is particularly effective when the observations are sparse and irregularly spaced, as growth curves usually are. A review on the method and on the tools developed by the authors to suitably set the free parameters, please refer to [our NM paper].

[qui magari due parole sulla rilevanza biologica?]

The method models individual curves g_i using basis functions

$$g_i(t) = \mathbf{s}(t)^T \boldsymbol{\eta}_i, \tag{1}$$

1

where $\mathbf{s}(t)$ is a $p-$dimensional spline basis vector and $\boldsymbol{\eta}_i$ is a vector of spline coefficients. The $\boldsymbol{\eta}_i$'s are treated with a random-effects model rather than considering them as parameters and fitting a separate spline curve for each individual. Moreover, a suitable parametrization of the mean curves allows a lower-dimensional representation of the curves with means in a restricted subspace. The model is fitted through an EM procedure.

Particular attention will be devoted to model selection. In particular, one must choose the number of clusters to fit, the dimension of the spline basis and the reduced dimension of the mean space. `connector` provides a complete toolkit to lead the user through such decisions, one of the most difficult problems in cluster analysis.

Installation

The `connector` package is hosted on the GitHub platform. The simplest way to obtain `connector` is to install it using `devtools`. Type the following commands in R console:

```r
# Install
install.packages("devtools",repos = "http://cran.us.r-project.org")
library(devtools)
install_github("qBioTurin/connector", ref="master",dependencies=TRUE)
# Load
library(connector)
```

Users may change the `repos` options depending on their locations and preferences. For more details, see `help(install.packages)`.

Quick Start

A demo is available to provide the list of commands to perform the analysis of one of the datasets considered in [ref: connector paper]. To run the example just type:

```r
demo("MainCommandsList", package = "connector")
```

Use case: tumor growth dataset

We illustrate the functionalities and the usage of the `connector` package on a tumor growth curves dataset. It is included in the package's data and it is organized as any observed curves are expected by the package's functions.

Observations should be saved into two distinct files:

1. an excel file reporting the discretely sampled curve data. As functional data are longitudinal, we respect the convention of parametrizing the models in terms of time t and y-values y. Each sample is represented as two columns of the excel file, the first one named *time* includes the lags list and the second one named with the *sample name* (each sample has different ID name, for example 475_P1b is the ID name of the first sample in the following figure) includes the list of observed y values. Hence, if we record 24 tumor growth curves, the file have 48 columns. See Figure 1.

2. a csv file containing the annotations associated to the sampled curves. It is composed by a number of rows equal to the number of samples. The first row gives the columns names as follows: the first twos must be a numerical ID and the corresponding ID name *sample name*. The remaing ones codify for the annotated features. Notice that the column *sample name* should contain the same ID names which appear in the sampled curve excel file. See Figure 2.

	A	B	C	D	E	F	G	H	I	J	K	L
1	time	475_P1b	time	475_P1bP2a	time	475_P1bP2b	time	475_P1bP2aP3a	time	475_P1bP2aP3b	time	475_P1bP2aP3bP4a
2	9	63.500364	6	0	6	0	5	163.24589	5	155.6897385	8	166.184865
3	16	0	13	95.05932	13	86.046797	12	201.7201105	12	143.31273	16	98.039808
4	23	78.082284	20	161.872992	20	83.6289155	15	152.6497715	15	158.468778	23	211.310442
5	31	137.9122875	27	117.22128	27	98.1059625	19	180.1715625	19	147.841551	29	367.4509825
6	36	188.4384	34	221.493368	34	197.3022925	22	202.069848	22	182.780862	36	692.8898625
7	44	206.87733	40	318.922848	40	302.830785	25	254.380926	25	289.27054	42	883.658919
8	51	227.2856	47	302.19948	47	304.706709	29	234.353944	29	373.100092	52	1618.681856
9	57	297.041256	60	505.1016	60	210.039786	34	317.25834	34	407.379968		
10	64	356.4912915	68	545.456296	68	147.4506	42	348.9675	42	589.055558		
11	72	428.757616	76	692.499456	76	164.2974	46	388.8712125	46	613.992704		
12	79	337.8964256	83	633.245184	83	306.1658655	50	500.5334375	50	770.646725		
13	87	354.6382	89	822.15675	89	394.5477205	54	471.919608	54	791.796568		
14	92	361.942784	96	605.362048	96	256.130749	57	618.094184	57	849.535074		
15	99	654.0703275	103	759.5610715	103	388.4685805	62	548.856	62	945.561428		
16	107	514.2261105	106	828.655	106	460.291624						
17												
18												

Figure 1: First lines of the excel file of the sampled curve data.

```
ID,SampleName,Progeny,Source,Real.Progeny
1,475_P1b,P1,P0,P3
2,475_P1bP2a,P2,475_P1b,P4
3,475_P1bP2b,P2,475_P1b,P4
4,475_P1bP2aP3a,P3,475_P1bP2a,P5
5,475_P1bP2aP3b,P3,475_P1bP2a,P5
6,475_P1bP2aP3bP4a,P4,475_P1bP2aP3b,P6
7,475_P1bP2aP3bP4b,P4,475_P1bP2aP3b,P6
8,475_P1bP2aP3bP4d,P4,475_P1bP2aP3b,P6
9,475_P1bP2aP3bP4c,P4,475_P1bP2aP3b,P6
10,475_P1bP2aP3bP4e,P4,475_P1bP2aP3b,P6
11,475_P1bP2aP3bP4aP5a,P5,475_P1bP2aP3bP4a,P7
12,475_P1bP2aP3bP4aP5b,P5,475_P1bP2aP3bP4a,P7
13,475_P1bP2aP3bP4aP5c,P5,475_P1bP2aP3bP4a,P7
14,475_P1bP2aP3bP4aP5d,P5,475_P1bP2aP3bP4a,P7
15,475_P1bP2aP3bP4aP5e,P5,475_P1bP2aP3bP4a,P7
16,475_P1bP2aP3bP4cP5f,P5,475_P1bP2aP3bP4c,P7
17,475_P1bP2aP3bP4dP5g,P5,475_P1bP2aP3bP4d,P7
18,475_P1bP2aP3bP4dP5h,P5,475_P1bP2aP3bP4d,P7
19,475_P1bP2aP3bP4dP5i,P5,475_P1bP2aP3bP4d,P7
20,475_P1bP2aP3bP4eP5j,P5,475_P1bP2aP3bP4e,P7
21,475_P1bP2aP3bP4eP5k,P5,475_P1bP2aP3bP4e,P7
```

Figure 2: First lines of the csv file of the annotated features.

Data Importing

Once data have been prepared in the two files above described, they are imported by the `DataImport` function. Two arguments, with the file names respectively, should be specified. In this example the full path names have been saved in the `GrwoDatFile` and `AnnotationFile` strings.

```
# find the full path names of the example files
GrowDataFile<-system.file("data", "475dataset.xlsx", package = "connector")
AnnotationFile <-system.file("data", "475info.txt", package = "connector")
# import the samples
CONNECTORList<-DataImport(GrowDataFile = GrowDataFile,
                          AnnotationFile = AnnotationFile)
## ###############################
## ######## Summary ##############
##
##   Number of curves: 21 ;
##   Min curve length:  7 ; Max curve length:  18 .
## ###############################
```

A list of four object is created:

```
# show the CONNECTORList structure
str(CONNECTORList)
## List of 4
##  $ Dataset :'data.frame':    265 obs. of  3 variables:
##   ..$ ID  : int [1:265] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Vol : num [1:265] 63.5 0 78.1 137.9 188.4 ...
##   ..$ Time: num [1:265] 9 16 23 31 36 44 51 57 64 72 ...
##  $ LenCurv : int [1:21] 15 15 15 14 14 7 7 11 7 11 ...
##  $ LabCurv :'data.frame':    21 obs. of  5 variables:
##   ..$ ID          : int [1:21] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ SampleName  : Factor w/ 21 levels "475_P1b","475_P1bP2a",..: 1 2 21 3 4 5 11 14 12 18 ...
##   ..$ Progeny     : Factor w/ 5 levels "P1","P2","P3",..: 1 2 2 3 3 4 4 4 4 4 ...
##   ..$ Source      : Factor w/ 8 levels "475_P1b","475_P1bP2a",..: 8 1 1 2 2 3 3 3 3 3 ...
##   ..$ Real.Progeny: Factor w/ 5 levels "P3","P4","P5",..: 1 2 2 3 3 4 4 4 4 4 ...
##  $ TimeGrid: num [1:66] 3 5 6 8 9 10 12 13 14 15 ...
```

The components of the `CONNECTORList` are:

1. `$Dataset`, a data frame with three variables: `ID` of the curve, `Vol` the y values and `Time` the time lags;

2. `$LenCurv`, the vector of the number of observations per sample;

3. `$LabCurv`, a data frame matching the sample with the corresponding annotated features. Hence the variables are extracted and named from the `AnnotationFile`;

4. `$TimeGrid`, the vector of the complete time grid points.

Data Visualization

The `GrowthCurve` function provides a plot of the sampled curves coloured by a user selected feature out of the ones given in the `AnnotationFile`. In Figure 3 an example is illustrated. It has been produced by the following code:

```
GrowPlot<-GrowthCurve(data = CONNECTORList,
                      feature = "Progeny")
GrowPlot
```
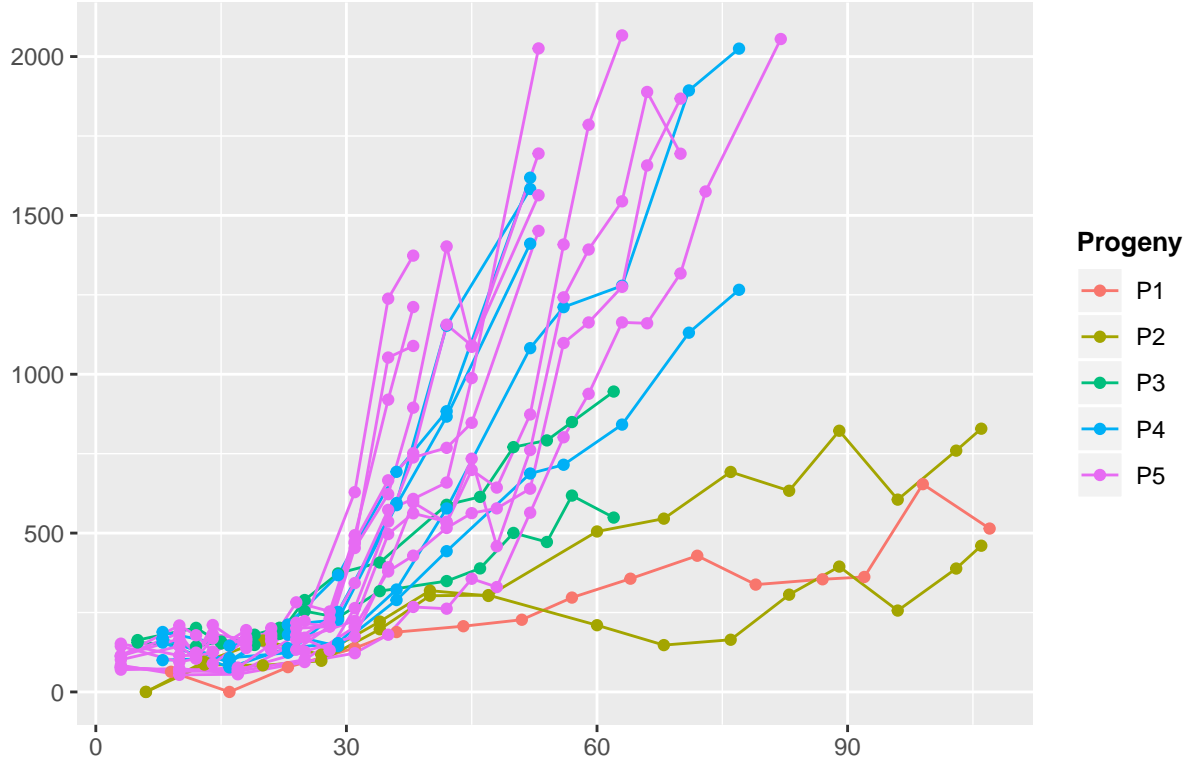
Figure 3: Sampled curves, coloured by progeny feature.

The `DataVisualization` function plots the sampled curves and the density of the time grid. This latter plot may be used to verify the density of the collected observations and eventually decide for a truncation.

```
# Growth curves and time grid visualization
Datavisual<-DataVisualization(data = CONNECTORList,
                              feature = "Progeny",
                              labels = c("Time","Volume","Tumor Growth"))
Datavisual
```

Curves may be (and it is often the case) irregularly and sparsely sampled. As a preliminary step, it may be useful to check whether the assembled pairs (t_{i_j}, t_{i_k}), for alla subjects $i = 1, \ldots, n$ and lags $i_1, \ldots, i_{n_i}$, are sufficiently dense in the domain plane. Low-density subregions may be excluded from the analysis and hence a truncation time for the observations imposed. The function `DataTruncation` have been developed to this aim. According to Figure 4, we decide to truncate the observations at 70 days.

```
# data truncation
trCONNECTORList<-DataTruncation(data = CONNECTORList,
                                feature="Progeny",
                                truncTime = 70,
                                labels = c("Time","Volume","Tumor Growth"))
## #############################################################
## ######## Summary of the trunc. data ############
##
##   Number of curves: 21 ;
##   Min curve length:  7 ; Max curve length:  16 ;
##
```
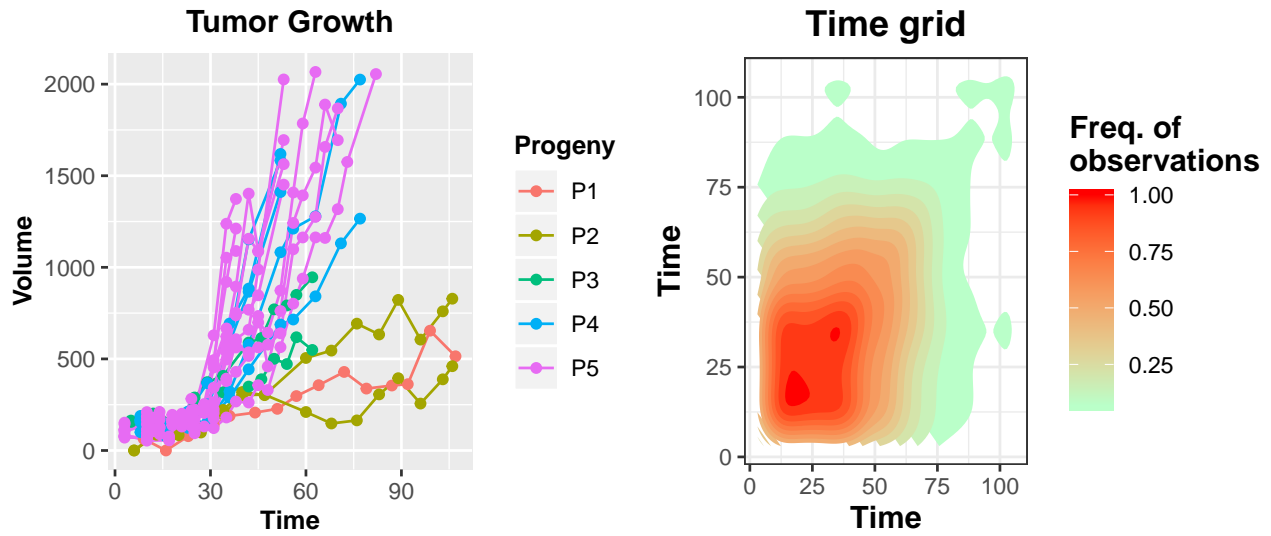
Figure 4: Sampled curves, coloured by progeny feature (left panel) and time grid density (right panel).

```
##  Number of truncated curves: 6 ;
##  Min points deleted:  2 ; Max points deleted:  6 ;
## ############################################################
# the trCONNECTORList structure
str(trCONNECTORList, max.level = 1)
## List of 6
##  $ Dataset         :'data.frame':    241 obs. of  3 variables:
##  $ LenCurv         : num [1:21] 9 9 9 14 14 7 7 9 7 9 ...
##  $ LabCurv         :'data.frame':    21 obs. of  5 variables:
##  $ TimeGrid        : num [1:50] 3 5 6 8 9 10 12 13 14 15 ...
##  $ ColFeature      : chr [1:5] "#FF0000FF" "#CCFF00FF" "#00FF66FF" "#0066FFFF" ...
##  $ GrowthCurve_plot:List of 9
##   ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
```

The output of the function `DataTruncation` is a list of six objects. The firt four of which are the truncated versions of the `DataImport` function. The plot stored in `$GrowthCurve_plot` shows the sampled curves plot with a vertical line at the truncation time, see Figure 5.

```
# plot
trCONNECTORList$GrowthCurve_plot
```

Model Selection Tools

Several model selection questions need a tool set to be addressed. In particular the free parameters to be set are:

1. the spline basis dimension, p;

2. the the mean space dimension, h;
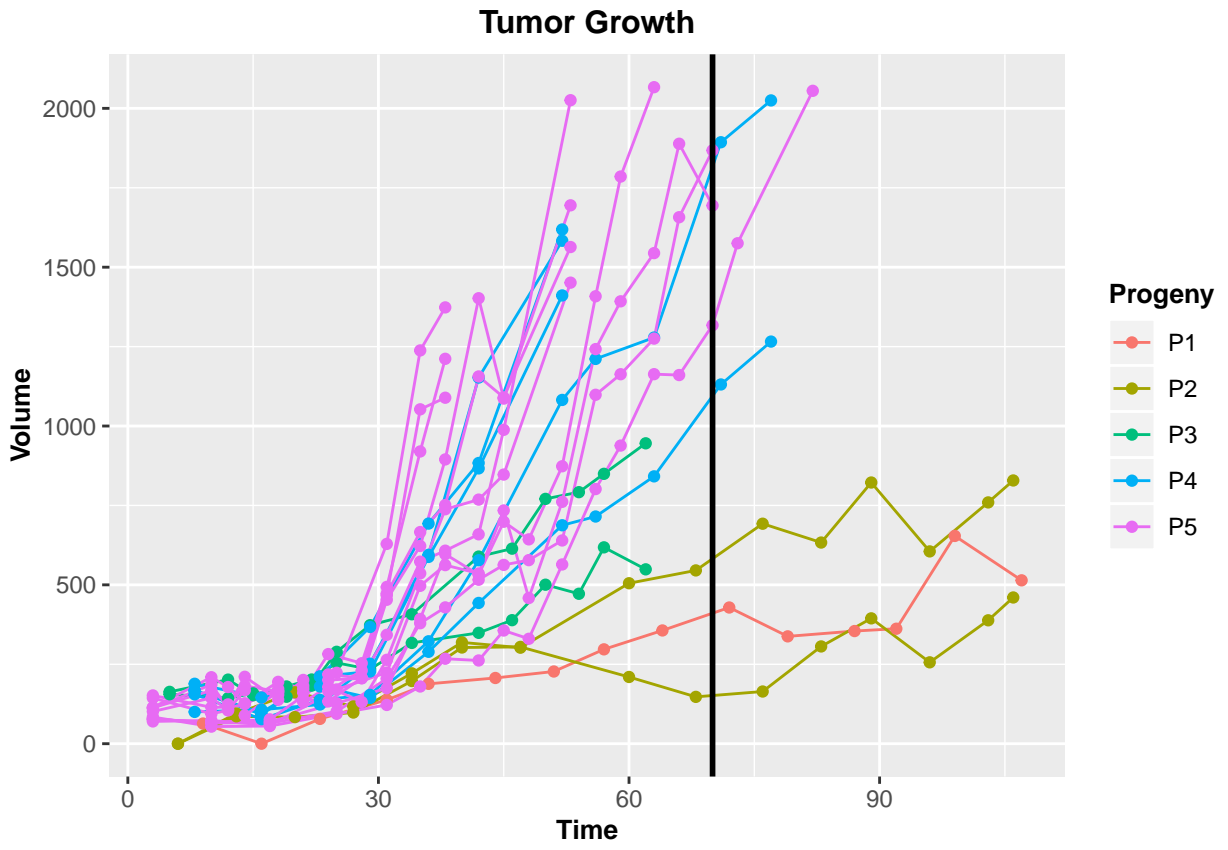
3. the number of clusters, G.

Figure 5: Sampled curves, coloured by progeny feature and truncation lag (vertical solid black line).

We developed a tool to help the user define each of the above parameters. Let us stress out that rough choices for the free parameters may compromise the full analysis.

The spline basis dimension

As proposed in (James, Hastie, and Sugar 2000), the dimension of the spline basis p can be taken to corresponding to the largest cross-validated likelihood. The function `BasisDimension.Choice` performs a ten-fold crossvalidation for each of the values of `p` decided by the user. The `$CrossLogLikePlot` element of the output gives a visual representation of the results, see Figure 6. Each gray dashed line corresponds to the crossloglkelihood values obtained on different test/learning sets and the solid black line is their mean value. The user should choose the the smallest value of `p` that ensures larger values of the mean crossloglikelihood function. It may be useful to keep in mind that working with sparse data encourage to spare parameters and hence smaller values of p are preferable.

```
# ten−fold crossvalidation
CrossLogLike<-BasisDimension.Choice(data = trCONNECTORList,
                                    p = 2:6 )
CrossLogLike$CrossLogLikePlot
```
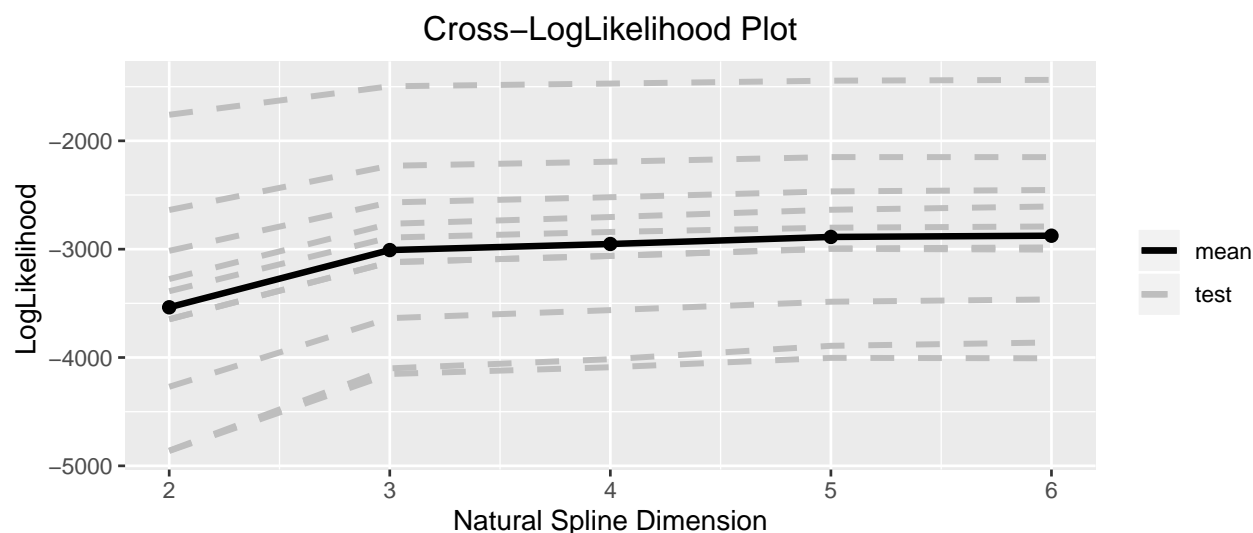

Figure 6: Cross-validated loglikelihood functions.

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

Hence, in the current example, we set $p = 3$, as it is the smallest value that ensures a large loglikelihood.

The mean space dimension

The dimension of the mean space is guided through a principal component analysis (PCA). The `PCA.Analysis` function applies PCA to the spline coefficients of each curve in order to capture the variance in the set of variables. In `$plot`, it is stored the bar plot of the percentage of variability in each component, see Fig. 7.

```
# Calculation of h
pca <- PCA.Analysis(data = trCONNECTORList,
                    p = p)
pca$plot
```
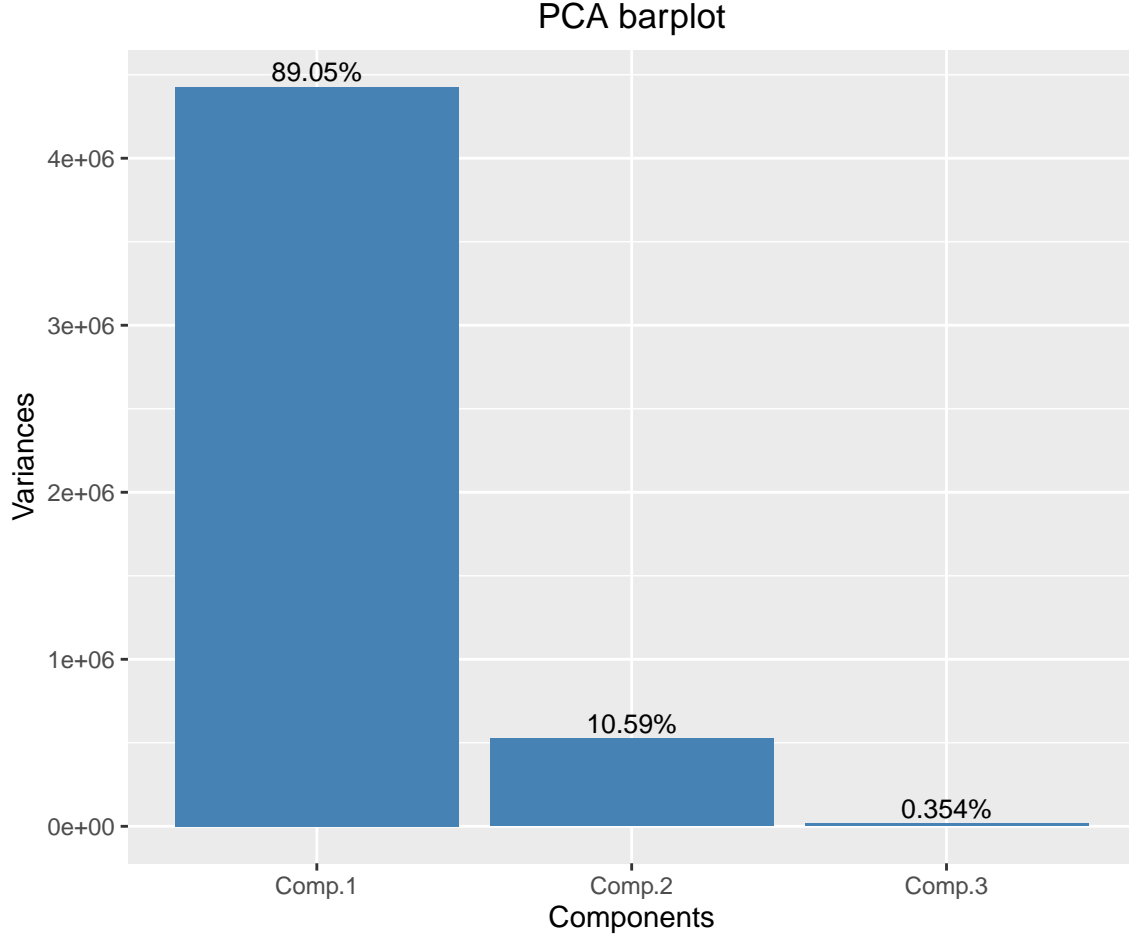

Figure 7: Bar plot of the variability explained by each component of the model.

```
# set h
h <- 1
```

In this example we set h equal to 1 in order to cover up to 89% of the total variability.

The number of clusters

Setting properly the number of clusters to fit is a well known problem in cluster analysis. `connector` provides two different plots to guide the user.

In [qui rif al nostro paper] we introduced two measures of proximity, the *total tightness* T and the *functional Davied-Bouldin index* fDB. Both measures rely on on the family of semi-metrics between curves defined as

$$D_q(f, g) = \sqrt{\int \left| f^{(q)}(s) - g^{(q)}(s) \right|^2 ds}, \qquad d = 0, 1, 2, \tag{2}$$

where f and g are two curves and $f^{(q)}$ and $g^{(q)}$ are their qth derivatives.

The *total tightness* T is the dispersion measure defined as

$$T = \sum_{k=1}^{G} \sum_{i=1}^{n} D_0(\hat{g}_i, \bar{g}^k), \tag{3}$$

where $\hat{g}_i$ is the estimated i–th curve given in eq. (8) and $\bar{g}^k$ is the center of k–th cluster given in eq. (7). As the number of clusters increases, the total tightness decreases to zero, the value which is attained when the number of fitted clusters equals the number of sampled curves. In this limiting case, any kth cluster mean curve coincides with an estimated curve and $D_0(\hat{g}_i, \bar{g}^k) = 0$ for any i and k. A proper number of clusters can be inferred as large enough to let the total tightness drop down to relatively little values but as the smallest over which the total tightness does not decrease substantially. Hence, we look for the location of an "elbow" in the plot of the total tightness against the number of clusters.

In [qui rif al nostro paper] we defined a second index, which is a cluster separation measure, and we called it *functional* DB (fDB). It is defined as follows

$$\text{fDB}_q = \frac{1}{G} \sum_{k=1}^{G} \max_{h \neq k} \left\{ \frac{S_h + S_k}{M_{hk}} \right\}, \tag{4}$$

where, for each cluster k and h

$$S_k = \sqrt{\frac{1}{G_k} \sum_{i=1}^{G_k} D_q^2(\hat{g}_i, \bar{g}^k)} \qquad \text{and} \qquad M_{hk} = D_q(\bar{g}^h, \bar{g}^k),$$

with G_k the number of curves in the kth cluster. The significance of eq. (4) can be understood as the average of the blend measures of each cluster from its most overlapping cluster. The "best" choice of clusters, then, will be that which minimizes this average blend.

To effectively take advantage of those two measures, `connector` supplies the function `StabilityAnalysis` which repeats the clustering procedure a number of times equal to the parameter `runs` and for each of the number of clusters given in the parameter `G`. The output of the function is a list of 3, and by means of the function `BoxPlot.Extrapolation` the two plots represented in Figure 8 can be extracted and plotted.

```
Stability.clustering <-StabilityAnalysis(data = trCONNECTORList,
                                         G = 2:5,
                                         h = h,
                                         p = p,
                                         runs = 100)
```

```
# the output structure
str(Stability.clustering, max.level = 1, vec.len=1)
## List of 3
##  $ ConsensusInfo:List of 1
##  $ BoxPlots     :List of 1
##  $ seed         : int [1:626] 403 624 ...
```

```
BoxPlot.Extrapolation(stability.list = Stability.clustering,
                      h = h)
```
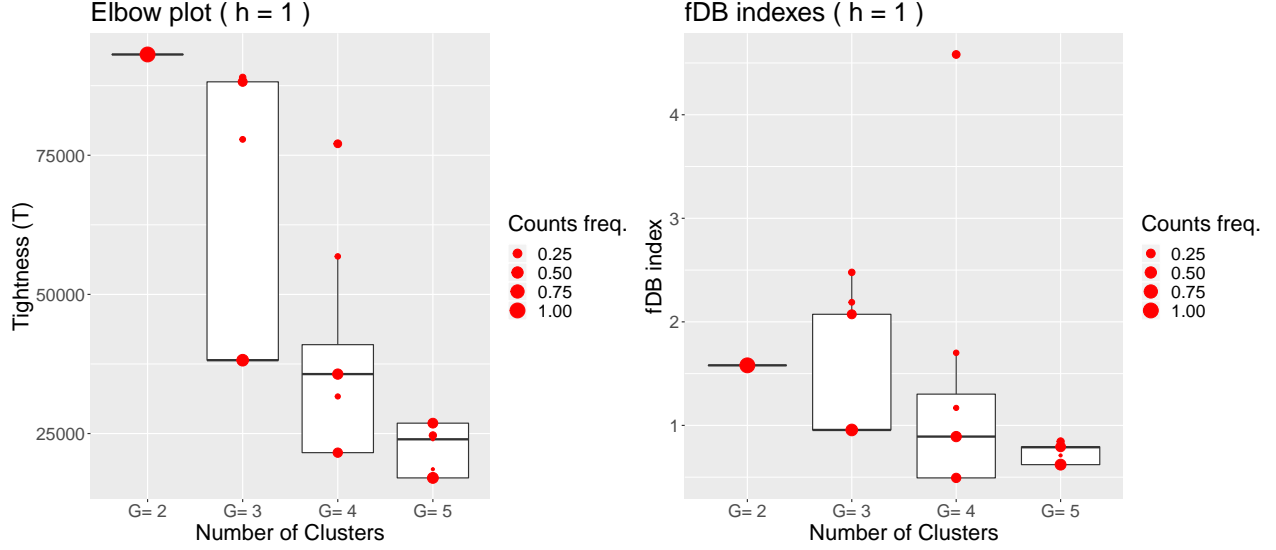
Figure 8: Box Plots of the *total tightness* T calculated on each run and for different number of clusters G (left panel). Box Plots of the *functional DB index* fDB calculated on each run and for different number of clusters G (right panel).

```
G <- 5
```

The left panel shows that $G = 4$ and $G = 5$ may be good choices for the parameter. The fDB indexes Box Plots plotted in the right panel lead the choice to $G = 4$, a number of clusters that explicitly minimizes the fDB index. Hence, for the illustrated example, we are choosing $G = 5$.

The variability of the two measures among runs, exhibited in Figure 8, is related to the random initialization of the k-means algorithm to get initial cluster memberships from points. The stability of the clustering procedure can be visualized through the consensus matrix extrapolated by the function `ConsMatrix.Extrapolation`, as shown in Figure 9

```
ConsMatrix.Extrapolation(stability.list = Stability.clustering,
                         h = h,
                         G = G)
```

Once the free parameters are all set, the function `MostProbableClustering.Extrapolation` can be used to fix the most probable clustering with given dimension of the spline basis p, dimension of the mean space h and number of clusters G and save the result in a dedicated object.

```
CONNECTORList.FCM.opt<-MostProbableClustering.Extrapolation(
                                  stability.list = Stability.clustering,
                                  h = h,
                                  G = G )
```

Results Visualization and Inspection

connector provides the function `ClusterWithMeanCurve` which plots the sampled curves grouped by cluster membership, together with the mean curve for each cluster, see Figure 10. The function prints as well the values for S_h, M_{hk} and fDB given in equation (4).

11

Consensus Matrix
Black line for the most probable clustering

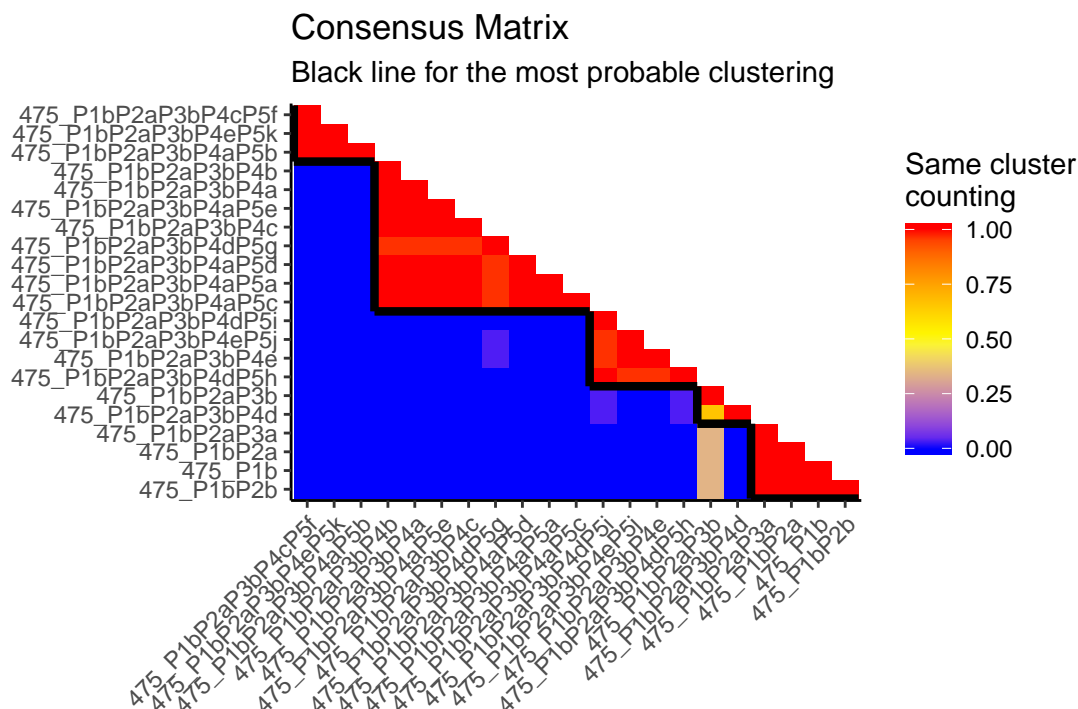Figure 9: Consensus Matrix for G = 4.

```r
FCMplots<- ClusterWithMeanCurve(clusterdata = CONNECTORList.FCM.opt,
                                data = trCONNECTORList,
                                feature = "Progeny",
                                labels = c("Time","Volume"),
                                title = paste("FCM model h=",h))
##
## ######## S indexes #############
##
##
##                    S           S_1          S_2
## ----------    ----------    ---------    ----------
## Cluster D     1307.8531     33.95039     0.2173674
## Cluster E     2518.7531     71.99226     0.0537308
## Cluster A      815.6152     24.59319     0.3182596
## Cluster C     1530.0366     35.92706     0.1615514
## Cluster B      464.0418     28.63248     0.6927886
##
## ##############################################################
## ##############################################################
##
##           ######## M indexes #############
##
##
##              Cluster D    Cluster E    Cluster A    Cluster C    Cluster B
## ----------   ----------   ----------   ----------   ----------   ----------
## Cluster D        0.000      7485.09     7607.603     3433.105     6170.482
## Cluster E     7485.090         0.00    15092.693    10918.195    13655.572
```

```
## Cluster A      7607.603     15092.69         0.000     4174.498     1437.121
## Cluster C      3433.105     10918.20      4174.498        0.000     2737.377
## Cluster B      6170.482     13655.57      1437.121     2737.377        0.000
##
## ###############################################################
## ######## R indexes ############
##
##
##                       R          R_1          R_2
## ----------  ----------  ----------  ----------
## Cluster D   0.8266247   0.4161246   0.0949809
## Cluster E   0.5112305   0.2893658   0.0352023
## Cluster A   0.8904307   0.7571834   0.4530207
## Cluster C   0.8266247   0.4821690   0.2009721
## Cluster B   0.8904307   0.7571834   0.4530207
##
## ###############################################################
## ######## fDB indexes ############
##
##
##        fDB        fDB_1        fDB_2
## ----------  ----------  ----------
##  0.7890683   0.5404052   0.2474394
##
## ###############################################################
```

A detailed visualization of the clusterization of the sample curves can be obtained by means of the
`DiscriminantPlot` function. In (James and Sugar 2003), the authors describe how to obtain low-
dimensional plots of curve datasets, enabling a visual assessment of clustering. They propose to project
the curves into the lower dimensional space of the mean space, so that they can be plotted as points (with
coordinates the functional linear discriminant components), making it much easier to detect the presence
of clusters. Moreover, in case $h = 1$ and hence one single functional linear discriminant component is
calculated, they suggest to plot it versus the standard deviation to indicate the level of accuracy with which
it has been observed. In the case study here described, we get the plots in Figure 11 which is coloured by
cluster membership and in Figure 12 which is coloured by the user selected feature called `"Progeny"`.

```
DiscrPlt<-DiscriminantPlot(clusterdata = CONNECTORList.FCM.opt,
                           data = trCONNECTORList,
                           h = h,
                           feature = "Progeny")

DiscrPlt$ColCluster
```

```
DiscrPlt$ColFeature
```

In the end, to inspect the composition of the clusters, the function `CountingSamples` reports the number
and the name of samples in each cluster according to the feature selected by the user.

```
NumberSamples<-CountingSamples(clusterdata = CONNECTORList.FCM.opt,
                               data = trCONNECTORList,
                               feature = "Progeny")
```
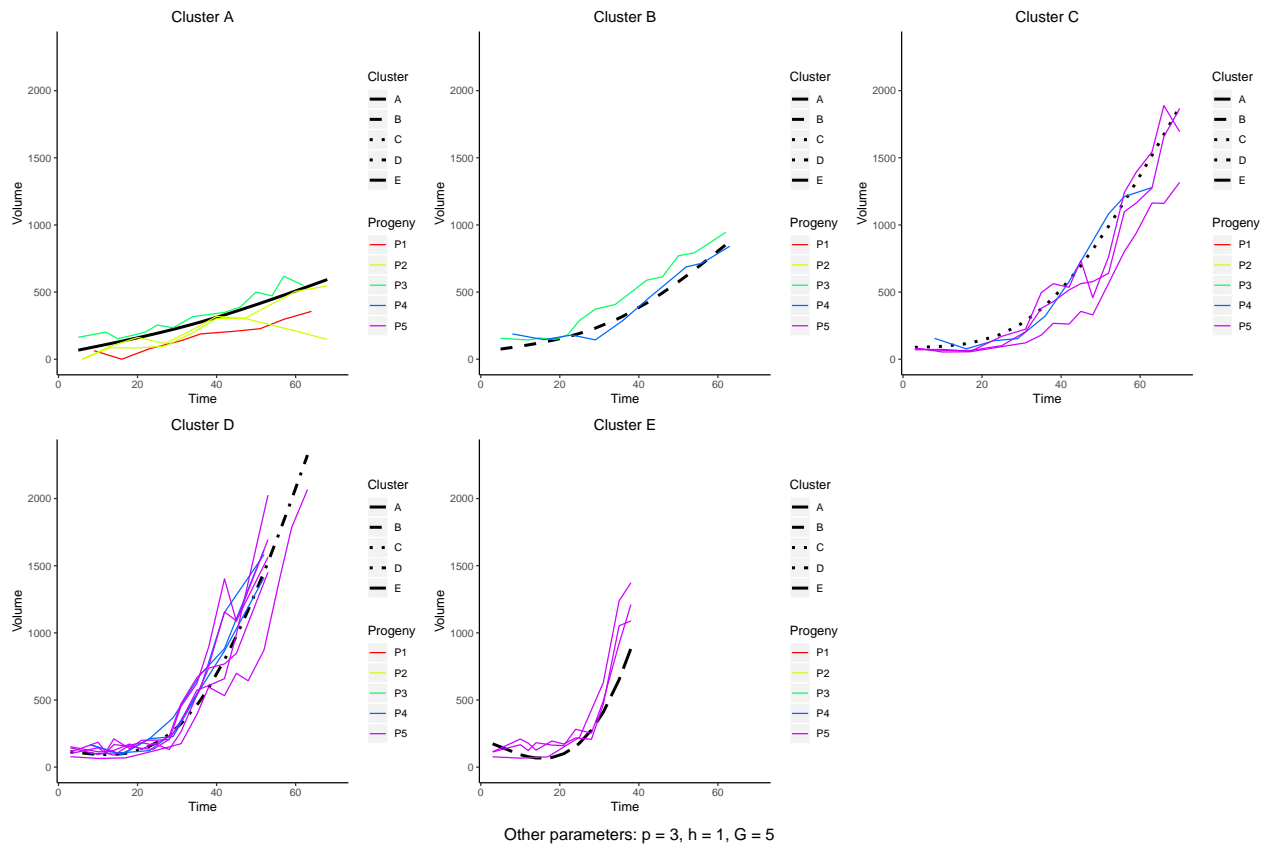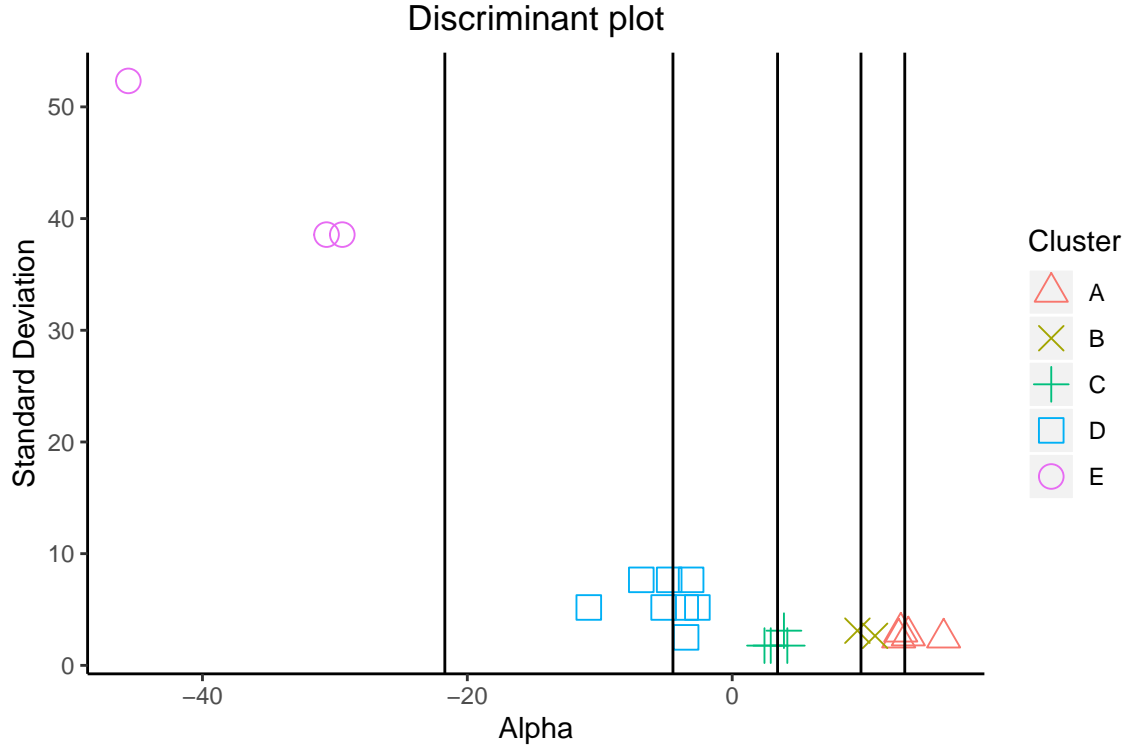
13

Figure 10: Sampled curves grouped by cluster membership.

Figure 11: Curves projected onto the 1-dimensional mean space: the functional linear discriminant α_1 versus the standard deviation. Symbols are coloured by cluster membership.
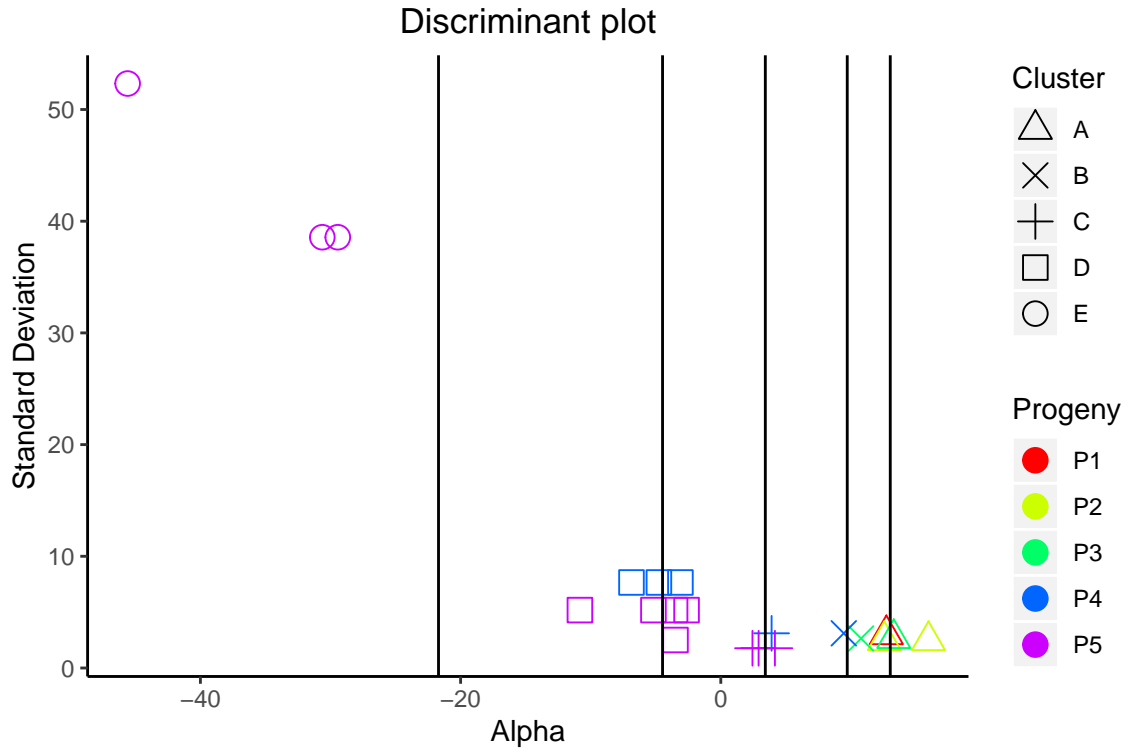

Figure 12: Curves projected onto the 1-dimensional mean space: the functional linear discriminant α_1 versus the standard deviation. Symbols are coloured by progeny.

```
str(NumberSamples, max.level = 2)
## List of 2
##  $ Counting    :'data.frame':    10 obs. of  3 variables:
##   ..$ Cluster: chr [1:10] "A" "A" "A" "B" ...
##   ..$ Progeny: Factor w/ 5 levels "P1","P2","P3",..: 1 2 3 3 4 4 5 4 5 5
##   ..$ freq   : int [1:10] 1 2 1 1 1 1 3 3 5 3
##  $ ClusterNames:'data.frame':    21 obs. of  2 variables:
##   ..$ Cluster   : Factor w/ 5 levels "A","B","C","D",..: 1 1 2 4 2 4 3 4 4 5 ...
##   ..$ SampleName: Factor w/ 21 levels "475_P1b","475_P1bP2a",..: 1 2 21 3 4 5 11 14 12 18 ...
```

Advanced Topics

In (James and Sugar 2003) the authors suggest to consider all the information about the traits that distinguish
one cluster from another. In particular, they calculate the optimal weights to apply to each dimension for
determining cluster membership. It turns out that when $h = 1$ such weights are a vector and can be plotted
as in Figure 13.

```
MaxDiscrPlots<-MaximumDiscriminationFunction(clusterdata = CONNECTORList.FCM.opt)

MaxDiscrPlots
## [[1]]
```
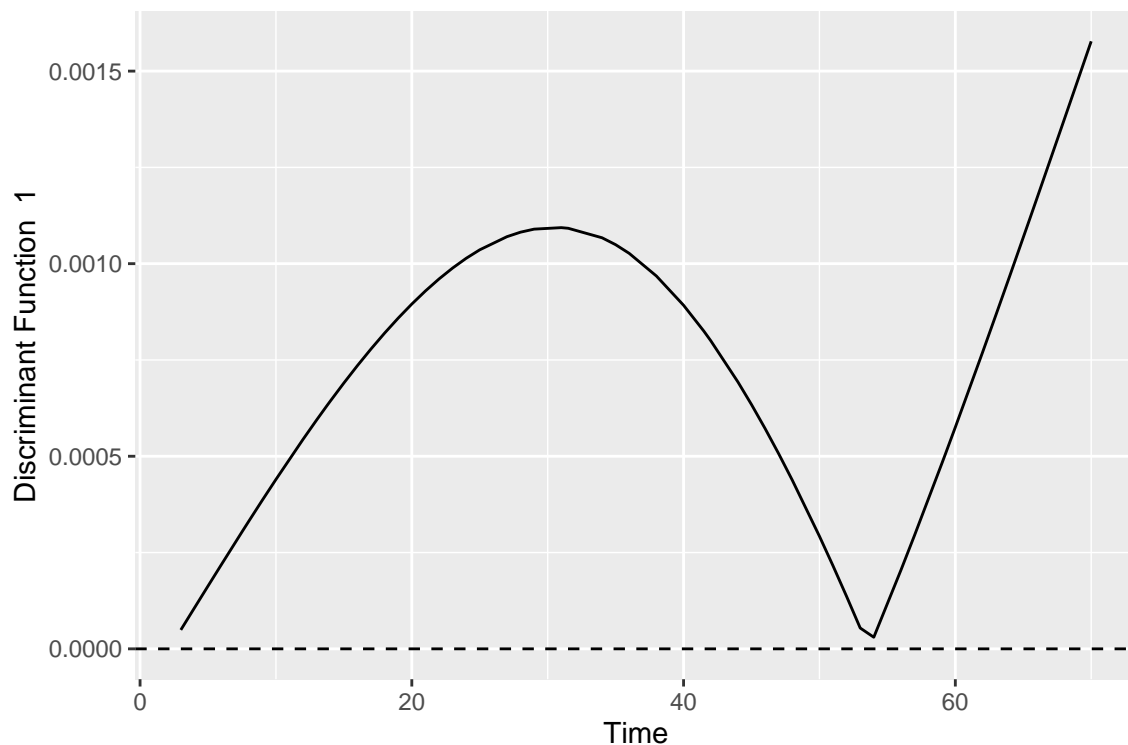

Figure 13: Discriminant curve.

Large absolute values correspond to large weights and hence large discrimination between clusters. Roughly
speaking Figure 13 tells us that earlier measurements are important in determining cluster assignment as
well as latter ones.

Details on the functional clustering model

The curves, $g_i(t)$ for each ith selected individual, are supposed to be observed with measurement errors and only at few discrete time points. Hence the vector $\mathbf{Y}_i$ of observed values at times $t_{i_1}, \ldots, t_{i_{n_i}}$ is given as

$$\mathbf{Y}_i = \mathbf{g}_i + \boldsymbol{\varepsilon}_i,$$

where $\mathbf{g}_i$ and $\boldsymbol{\varepsilon}_i$ are the vectors of true values and measurement errors at time grid, respectively. As there are only finite number of observations, individual curves are modeled using basis functions, in particular cubic splines. Let

$$g_i(t) = \mathbf{s}(t)^T \boldsymbol{\eta}_i, \tag{5}$$

where $\mathbf{s}(t)$ is a $p-$dimensional spline basis vector and $\boldsymbol{\eta}_i$ is a vector of spline coefficients. The $\boldsymbol{\eta}_i$'s are treated with a random-effects model rather than considering them as parameters and fitting a separate spline curve for each individual. Cluster means are furthermore rewritten as

$$\boldsymbol{\mu}_k = \boldsymbol{\lambda}_0 + \Lambda \boldsymbol{\alpha}_k,$$

where $\boldsymbol{\lambda}_0$ and $\boldsymbol{\alpha}_k$ are $p-$ and $h-$ dimensional vectors, Λ is a (p, h) matrix and $h \leq \min(p, G-1)$, where G denote the true number of clusters. This parametrization allows a lower-dimensional representation of the curves with means in a restricted subspace (for $h < G - 1$).

With this formulation, the functional clustering model can be written as

$$\mathbf{Y}_i = S_i \cdot (\boldsymbol{\lambda}_0 + \Lambda \boldsymbol{\alpha}_{\mathbf{z}_i} + \boldsymbol{\gamma}_i) + \boldsymbol{\varepsilon}_i, \quad i = 1, \ldots, n,$$
$$\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, R), \quad \boldsymbol{\gamma}_i \sim \mathcal{N}(\mathbf{0}, \Gamma), \tag{6}$$

where $S_i = (\mathbf{s}(t_{i_1}), \ldots, \mathbf{s}(t_{i_{n_i}}))^T$ is the spline basis matrix for the $i-$th curve.

The model is fitted following (James and Sugar 2003) and all the estimated parameters and the predicted cluster membership are returned. Notice that the kth cluster mean curve can be retrieved as

$$\bar{g}^k(t) = \mathbf{s}(t)^T (\hat{\boldsymbol{\lambda}}_0 + \hat{\Lambda} \hat{\boldsymbol{\alpha}}_k). \tag{7}$$

Moreover, the functional clustering procedure can accurately predict unobserved portions of the curves $g_i(t)$ by means of the natural estimate

$$\hat{g}_i(t) = \mathbf{s}(t)^T \hat{\boldsymbol{\eta}}_i, \tag{8}$$

where $\hat{\boldsymbol{\eta}}_i$ is a prediction for $\boldsymbol{\eta}_i$ which is proven to be optimally computed as $\mathbb{E}(\boldsymbol{\eta}_i \mid \mathbf{Y}_i)$ and explicitly given in (James and Sugar 2003), eq. (17).

References

Benzekry, Clare AND Beheshti, Sébastien AND Lamont. 2014. "Classical Mathematical Models for Description and Prediction of Experimental Tumor Growth." *PLOS Computational Biology* 10 (8): 1–19. https://doi.org/10.1371/journal.pcbi.1003800.

James, Gareth M, Trevor J Hastie, and Catherine A Sugar. 2000. "Principal Component Models for Sparse Functional Data." *Biometrika* 87 (3): 587–602.

James, Gareth M, and Catherine A Sugar. 2003. "Clustering for Sparsely Sampled Functional Data." *Journal of the American Statistical Association* 98 (462): 397–408.