Package 'sasfunclust'

April 2, 2021

Type Package

Title Sparse and Smooth Functional Clustering

Version 1.0.0

Description Implements the sparse and smooth functional clustering (SaS-Funclust) method (Centofanti et al. (2021) <arXiv:2103.15224>) that aims to classify a sample of curves into homogeneous groups while jointly detecting the most informative portions of domain.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

LinkingTo Rcpp,RcppArmadillo

Imports Rcpp, fda, mclust, matrixcalc, MASS, Matrix

URL https://github.com/unina-sfere/sasfunclust

BugReports https://github.com/unina-sfere/sasfunclust/issues

Suggests knitr, rmarkdown, testthat

SystemRequirements GNU make

NeedsCompilation yes

Author Fabio Centofanti [cre, aut],
Antonio Lepore [aut],
Biagio Palumbo [aut]

Maintainer Fabio Centofanti <fabio.centofanti@unina.it>

Repository CRAN

Date/Publication 2021-04-02 11:30:02 UTC

R topics documented:

`sasfunclust-package` *Sparse and smooth functional data clustering*

Description

Implements the sparse and smooth functional clustering (SaS-Funclust) method (Centofanti et al. (2021) <arXiv:2103.15224>) that aims to classify a sample of curves into homogeneous groups while jointly detecting the most informative portions of domain.

Details

Package:	sasfunclust
Type:	Package
Version:	1.0.0
Date:	2021-04-01
License:	GPL-3

Author(s)

Fabio Centofanti, Antonio Lepore, Biagio Palumbo

References

Centofanti, F., Lepore, A., & Palumbo, B. (2021). Sparse and Smooth Functional Data Clustering. *arXiv preprint arXiv:2103.15224*.

See Also

`sasfclust`, `sasfclust_cv`

Examples

```
## Not run:

n_i=20
train<-simulate_data("Scenario I",n_i=n_i,sd = 1,sd2_basis = 0.5^2)
lambda_s_seq=10^seq(-4,-3)
lambda_l_seq=10^seq(-1,0)
G_seq=2
mod_cv<-sasfclust_cv(X=train$X,grid=train$grid,G_seq=G_seq,
lambda_l_seq = lambda_l_seq,lambda_s_seq =lambda_s_seq,maxit = 5,K_fold = 2,q=10)
plot(mod_cv)
```

```
mod<-sasfclust(X=train$X,grid=train$grid,G_seq=mod_cv$G_opt,
lambda_l = mod_cv$lambda_l_opt,lambda_s_seq =mod_cv$lambda_s_opt,maxit = 5,q=10)

print(aa$clus$classes)
plot(mod)

## End(Not run)
```

`plot.sasfclust` *Plot the results of the Sas-funclust method*

Description

This function provides plots of the estimated cluster mean functions and of the classified curves when applied to the output of `sasfclust`, whereas provides the cross-validation plots when applied to the output of `sasfclust_cv`. In the latter case the first plot displays the CV values as a function of `G`, `lambda_s` and `lambda_l`; the second plot displays the CV values as a function of `lambda_s` and `lambda_l` for `G` fixed at its optimal value; the third plot displays the CV values as a function of `lambda_l` for `G` and `lambda_s` fixed at their optimal value.

Usage

```
## S3 method for class 'sasfclust_cv'
plot(x, ...)

## S3 method for class 'sasfclust'
plot(x, ...)
```

Arguments

`x`	The output of either `sasfclust` or `sasfclust_cv`.
`...`	No additional parameters, called for side effects.

Value

No return value, called for side effects.

Examples

```
library(sasfunclust)
train<-simulate_data("Scenario I",n_i=20,var_e = 1,var_b = 0.5^2)
lambda_s_seq=10^seq(-4,-3)
lambda_l_seq=10^seq(-1,0)
G_seq=2
mod_cv<-sasfclust_cv(X=train$X,grid=train$grid,G_seq=G_seq,
lambda_l_seq = lambda_l_seq,lambda_s_seq =lambda_s_seq,maxit = 20,K_fold = 2,q=10)
plot(mod_cv)
mod<-sasfclust(X=train$X,grid=train$grid,lambda_s = 10^-6,lambda_l =10,G = 2,maxit = 20,q=10)
```

```
plot(mod)
```

`sasfclust` *Sparse and Smooth Functional Data Clustering*

Description

Sparse and smooth functional clustering (SaS-Funclust) allows to cluster a sample of curves into homogeneous groups while jointly detecting the most informative portion of domain. (Centofanti et al., 2021).

Usage

```
sasfclust(
  X = NULL,
  timeindex = NULL,
  curve = NULL,
  grid = NULL,
  q = 30,
  lambda_l = 10,
  lambda_s = 10,
  G = 2,
  tol = 10^-7,
  maxit = 50,
  par_LQA = list(eps_diff = 1e-06, MAX_iter_LQA = 200, eps_LQA = 1e-05),
  plot = F,
  trace = F,
  init = "kmeans",
  varcon = "diagonal",
  lambda_s_ini = NULL
)
```

Arguments

`X`	For functional data observed over a regular grid: a matrix of where the rows must correspond to argument values and columns to replications. For functional data observed over an irregular grid: a vector of length $\sum_{i=1}^{N} n_i$, with N the number of curves, where the entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ are elements representing the observations for curve k.
`timeindex`	A vector of length $\sum_{i=1}^{N} n_i$. The entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ provide the locations on `grid` of curve k. So for example, if the kth curve is observed at time points t_l, t_m of the `grid` then the the entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ would be l, m, being $n_k = 2$. If X is a matrix, timeindex is ignored.
`curve`	A vector of length $\sum_{i=1}^{N} n_i$. The entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ are equal to k. If X is a matrix, curve is ignored.

grid	The vector of time points where the curves are sampled. For Functional data observed over an irregular grid, timeindex and grid provide the time points for each curve.
q	The dimension of the set of B-spline functions.
lambda_l	Tuning parameter of the functional adaptive pairwise fusion penalty (FAPFP).
lambda_s	Tuning parameter of the smoothness penalty.
G	The number of clusters.
tol	The tolerance for the stopping condition of the expectation conditional maximization (ECM) algorithms. The algorithm stops when the log-likelihood difference between two consecutive iterations is less or equal than tol.
maxit	The maximum number of iterations allowed in the ECM algorithm.
par_LQA	A list of parameters for the local quadratic approximation (LQA) in the ECM algorithm. eps_diff is the lower bound for the coefficient mean differences, values below eps_diff are set to zero. MAX_iter_LQA is the maximum number of iterations allowed in the LQA. eps_LQA is the tolerance for the stopping condition of LQA.
plot	If TRUE, the estimated cluster means are plotted at each iteration of the ECM algorithm. Default is FALSE.
trace	If TRUE, information are shown at each iteration of the ECM algorithm. Default is FALSE.
init	It is the way to initialize the ECM algorithm. There are three ways of initialization: "kmeans", "model-based", and "hierarchical", that provide initialization through the k-means algorithm, model-based clustering based on parameterized finite Gaussian mixture model, and hierarchical clustering, respectively. Default is "kmeans".
varcon	A vector of character strings indicating the type of coefficient covariance matrix. Three values are allowed: "full", "diagonal", and "equal". "full" means unrestricted cluster coefficient covariance matrices allowed to be different among clusters. "diagonal" means diagonal cluster coefficient covariance matrices that are equal among clusters. "equal" means diagonal cluster coefficient covariance matrices, with equal diagonal entries, that are equal among clusters. Default is "diagonal".
lambda_s_ini	The tuning parameter used to obtain the functional data through smoothing B-splines before applying the initialization algorithm. If NULL a Generalized cross validation procedure is used as described in Ramsay (2005). Default is NULL.

Value

A list containing the following arguments: mod that is a list composed by

- data: A list containing the vectorized form of X, timeindex, and curve. For functional data observed over a regular grid timeindex and curve are trivially obtained.
- parameters: A list containing all the estimated parameters.
- vars: A list containing results from the Expectation step of the ECM algorithm.

- `FullS`: The matrix of B-spline computed over `grid`.
- `grid`: The vector of time points where the curves are sampled.
- `W`: The basis roughness penalty matrix containing the inner products of pairs of basis function second derivatives.
- `AW_vec`: Vectorized version of the diagonal matrix used in the approximation of FAPFP.
- `P_tot`: Sparse Matrix used to compute all the pairwise comparisons in the FAPFP.
- `lambda_s`: Tuning parameter of the smoothness penalty.
- `lambda_l`: Tuning parameter of the FAPFP.

A list, named `clus`, containing the following arguments:

- `classes`: The vector of cluster membership.
- `po_pr`: Posterior probabilities of cluster membership.

`mean_fd` The estimated cluster mean functions.

`class` A label for the output type.

References

Centofanti, F., Lepore, A., & Palumbo, B. (2021). Sparse and Smooth Functional Data Clustering. *arXiv preprint arXiv:2103.15224*.

Ramsay, J., Ramsay, J., & Silverman, B. W. (2005). Functional Data Analysis. Springer Science & Business Media.

See Also

`sasfclust_cv`

Examples

```
library(sasfunclust)
train<-simulate_data("Scenario I",n_i=20,var_e = 1,var_b = 0.5^2)
mod<-sasfclust(X=train$X,grid=train$grid,lambda_s = 10^-6,lambda_l =10,G = 2,maxit = 5,q=10)
plot(mod)
```

---

sasfclust_cv *Cross-validation for sasfclust*

Description

K-fold cross-validation procedure to choose the number of clusters and the tuning parameters for the sparse and smooth functional clustering (SaS-Funclust) method (Centofanti et al., 2021).

Usage

```
sasfclust_cv(
  X = NULL,
  timeindex = NULL,
  curve = NULL,
  grid = NULL,
  q = 30,
  lambda_l_seq = 10^seq(-1, 2),
  lambda_s_seq = 10^seq(-5, -3),
  G_seq = 2,
  tol = 10^-7,
  maxit = 50,
  par_LQA = list(eps_diff = 1e-06, MAX_iter_LQA = 200, eps_LQA = 1e-05),
  plot = FALSE,
  trace = FALSE,
  init = "kmeans",
  varcon = "diagonal",
  lambda_s_ini = NULL,
  K_fold = 5,
  X_test = NULL,
  grid_test = NULL,
  m1 = 1,
  m2 = 0,
  m3 = 1,
  ncores = 1
)
```

Arguments

X	For functional data observed over a regular grid: a matrix of where the rows must correspond to argument values and columns to replications. For functional data observed over an irregular grid: a vector of length $\sum_{i=1}^{N} n_i$, with N the number of curves, where the entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ are elements representing the observations for curve k.
timeindex	A vector of length $\sum_{i=1}^{N} n_i$. The entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ provide the locations on grid of curve k. So for example, if the kth curve is observed at time points t_l, t_m of the grid then the the entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ would be l, m, being $n_k = 2$. If X is a matrix, timeindex is ignored.
curve	A vector of length $\sum_{i=1}^{N} n_i$. The entries from $\sum_{i=1}^{k-1}(n_i+1)$ to $\sum_{i=1}^{k} n_i$ are equal to k. If X is a matrix, curve is ignored.
grid	The vector of time points where the curves are sampled. For Functional data observed over an irregular grid, timeindex and grid provide the time points for each curve.
q	The dimension of the set of B-spline functions.
lambda_l_seq	Sequence of tuning parameter of the functional adaptive pairwise fusion penalty (FAPFP).

lambda_s_seq	Sequence of tuning parameter of the smoothness penalty.
G_seq	Sequence of number of clusters.
tol	The tolerance for the stopping condition of the expectation conditional maximization (ECM) algorithms. The algorithm stops when the log-likelihood difference between two consecutive iterations is less or equal than tol.
maxit	The maximum number of iterations allowed in the ECM algorithm.
par_LQA	A list of parameters for the local quadratic approximation (LQA) in the ECM algorithm. eps_diff is the lower bound for the coefficient mean differences, values below eps_diff are set to zero. MAX_iter_LQA is the maximum number of iterations allowed in the LQA. eps_LQA is the tolerance for the stopping condition of LQA.
plot	If TRUE, the estimated cluster means are plotted at each iteration of the ECM algorithm. Default is FALSE.
trace	If TRUE, information are shown at each iteration of the ECM algorithm. Default is FALSE.
init	It is the way to initialize the ECM algorithm. There are three ways of initialization: "kmeans", "model-based", and "hierarchical", that provide initialization through the k-means algorithm, model-based clustering based on parameterized finite Gaussian mixture model, and hierarchical clustering, respectively. Default is "kmeans".
varcon	A vector of character strings indicating the type of coefficient covariance matrix. Three values are allowed: "full", "diagonal", and "equal". "full" means unrestricted cluster coefficient covariance matrices allowed to be different among clusters. "diagonal" means diagonal cluster coefficient covariance matrices that are equal among clusters. "equal" means diagonal cluster coefficient covariance matrices, with equal diagonal entries, that are equal among clusters. Default is "diagonal".
lambda_s_ini	The tuning parameter used to obtain the functional data through smoothing B-splines before applying the initialization algorithm. If NULL a Generalized cross validation procedure is used as described in Ramsay (2005). Default is NULL.
K_fold	Number of folds. Default is 5.
X_test	Only for functional data observed over a regular grid, a matrix where the rows must correspond to argument values and columns to replications of the test set. Default in NULL.
grid_test	The vector of time points where the test set curves are sampled. Default is NULL.
m1	The m-standard deviation rule parameter to choose G for each lambda_s and lambda_l.
m2	The m-standard deviation rule parameter to choose lambda_s fixed G for each lambda_l.
m3	The m-standard deviation rule parameter to choose lambda_l fixed G and lambda_s.
ncores	If ncores>1, then parallel computing is used, with ncores cores. Default is 1.

Value

A list containing the following arguments:

`G_opt`: The optimal number of clusters.

`lambda_l_opt`: The optimal tuning parameter of the FAPFP.

`lambda_s_opt`: The optimal tuning parameter of the smoothness penalty.

`comb_list`: The combinations of `G`,`lambda_s` and `lambda_l` explored.

`CV`: The cross-validation values obtained for each combination of `G`,`lambda_s` and `lambda_l`.

`CV_sd`: The standard deviations of the cross-validation values.

`zeros`: Fraction of domain over which the estimated cluster means are fused.

`ms`: The m-standard deviation rule parameters.

`class`: A label for the output type.

References

Centofanti, F., Lepore, A., & Palumbo, B. (2021). Sparse and Smooth Functional Data Clustering. *arXiv preprint arXiv:2103.15224*.

Ramsay, J., Ramsay, J., & Silverman, B. W. (2005). Functional Data Analysis. Springer Science & Business Media.

See Also

`sasfclust`

Examples

```
library(sasfunclust)
train<-simulate_data("Scenario I",n_i=20,var_e = 1,var_b = 0.5^2)
lambda_s_seq=10^seq(-4,-3)
lambda_l_seq=10^seq(-1,0)
G_seq=2
mod_cv<-sasfclust_cv(X=train$X,grid=train$grid,G_seq=G_seq,
lambda_l_seq = lambda_l_seq,lambda_s_seq =lambda_s_seq,maxit = 20,K_fold = 2,q=10)
plot(mod_cv)
```

`simulate_data` *Simulate data for functional clustering*

Description

Generate synthetic data as in the simulation study of Centofanti et al., 2021.

Usage

```
simulate_data(
  scenario,
  n_i = 50,
  nbasis = 30,
  length_tot = 50,
  var_e = 1,
  var_b = 1
)
```

Arguments

scenario	A character strings indicating the scenario considered. It could be "Scenario I", "Scenario II", and "Scenario III".
n_i	Number of curves in each cluster.
nbasis	The dimension of the set of B-spline functions.
length_tot	Number of evaluation points.
var_e	Variance of the measurement error.
var_b	Diagonal entries of the coefficient variance matrix, which is assumed to be diagonal, with equal diagonal entries, and the same among clusters.

Value

A list containing the following arguments:

X: Observation matrix, where the rows correspond to argument values and columns to replications.

X_fd: Functional observations without measurement error.

mu_fd: True cluster mean function.

grid: The vector of time points where the curves are sampled.

clus: True cluster membership vector.

Examples

```
library(sasfunclust)
train<-simulate_data("Scenario I",n_i=20,var_e = 1,var_b = 0.5^2)
```

Index