Package ‘flamingos’

July 22, 2025

Type Package

Title Functional Latent Data Models for Clustering Heterogeneous Curves ('FLaMingos')

Version 0.1.0

Description Provides a variety of original and flexible user-friendly statistical latent variable models for the simultaneous clustering and segmentation of heterogeneous functional data (i.e time series, or more generally longitudinal data, fitted by unsupervised algorithms, including EM algorithms. Functional Latent Data Models for Clustering heterogeneous curves ('FLaMingos') are originally introduced and written in 'Matlab' by Faicel Chamroukhi <`https://github.com/fchamroukhi?utf8=?&tab=repositories&q=mix&type=public&language=matlab`>.
The references are mainly the following ones.
Chamroukhi F. (2010) <`https://chamroukhi.com/FChamroukhi-PhD.pdf`>.
Chamroukhi F., Same A., Govaert, G. and Aknin P. (2010) <doi:10.1016/j.neucom.2009.12.023>.
Chamroukhi F., Same A., Aknin P. and Govaert G. (2011). <doi:10.1109/IJCNN.2011.6033590>.
Same A., Chamroukhi F., Govaert G. and Aknin, P. (2011) <doi:10.1007/s11634-011-0096-5>.
Chamroukhi F., and Glotin H. (2012) <doi:10.1109/IJCNN.2012.6252818>.
Chamroukhi F., Glotin H. and Same A. (2013) <doi:10.1016/j.neucom.2012.10.030>.
Chamroukhi F. (2015) <`https://chamroukhi.com/FChamroukhi-HDR.pdf`>.
Chamroukhi F. and Nguyen H-D. (2019) <doi:10.1002/widm.1298>.

URL `https://github.com/fchamroukhi/FLaMingos`

BugReports `https://github.com/fchamroukhi/FLaMingos/issues`

License GPL (>= 3)

Depends R (>= 2.10)

Imports methods, stats, Rcpp

Suggests knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo

Collate flamingos-package.R RcppExports.R utils.R kmeans.R mkStochastic.R FData.R ParamMixHMM.R ParamMixHMMR.R ParamMixRHLP.R StatMixHMM.R StatMixHMMR.R StatMixRHLP.R

ModelMixHMMR.R ModelMixHMM.R ModelMixRHLP.R emMixHMM.R emMixHMMR.R emMixRHLP.R cemMixRHLP.R data-toydataset.R

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation yes

Author Faicel Chamroukhi [aut] (ORCID:
<`https://orcid.org/0000-0002-5894-3103`>),
Florian Lecocq [aut, trl, cre] (R port),
Marius Bartcus [aut, trl] (R port)

Maintainer Florian Lecocq <`florian.lecocq@outlook.com`>

Repository CRAN

Date/Publication 2019-08-06 09:30:02 UTC

Contents

flamingos-package 3
cemMixRHLP 4
emMixHMM 6
emMixHMMR 7
emMixRHLP 9
FData-class 11
mkStochastic 11
ModelMixHMM-class 12
ModelMixHMMR-class 13
ModelMixRHLP-class 14
ParamMixHMM-class 15
ParamMixHMMR-class 16
ParamMixRHLP-class 17
StatMixHMM-class 18
StatMixHMMR-class 19
StatMixRHLP-class 21
toydataset 22

Index **24**

`flamingos-package` *FLaMingos: Functional Latent datA Models for clusterING heterogeneOus curveS*

Description

`flamingos` is an open-source toolbox for the simultaneous clustering (or classification) and segmentation of heterogeneous functional data (i.e time-series ore more generally longitudinal data), with original and flexible functional latent variable models, fitted by unsupervised algorithms, including EM algorithms.

`flamingos` contains the following time series clustering and segmentation models:

- mixRHLP;
- mixHMM;
- mixHMMR.

For the advantages/differences of each of them, the user is referred to our mentioned paper references.

To learn more about `flamingos`, start with the vignettes: `browseVignettes(package = "flamingos")`

Author(s)

Maintainer: Florian Lecocq <`florian.lecocq@outlook.com`> (R port) [translator]

Authors:

- Faicel Chamroukhi <`faicel.chamroukhi@unicaen.fr`> (0000-0002-5894-3103)
- Marius Bartcus <`marius.bartcus@gmail.com`> (R port) [translator]

References

Chamroukhi, Faicel, and Hien D. Nguyen. 2019. *Model-Based Clustering and Classification of Functional Data.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. `https://chamroukhi.com/papers/MBCC-FDA.pdf`.

Chamroukhi, F. 2016. *Unsupervised Learning of Regression Mixture Models with Unknown Number of Components.* Journal of Statistical Computation and Simulation 86 (November): 2308–34. `https://chamroukhi.com/papers/Chamroukhi-JSCS-2015.pdf`.

Chamroukhi, Faicel. 2016. *Piecewise Regression Mixture for Simultaneous Functional Data Clustering and Optimal Segmentation.* Journal of Classification 33 (3): 374–411. `https://chamroukhi.com/papers/Chamroukhi-PWRM-JournalClassif-2016.pdf`.

Chamroukhi, F. 2015. *Statistical Learning of Latent Data Models for Complex Data Analysis.* Habilitation Thesis (HDR), Universite de Toulon. `https://chamroukhi.com/Dossier/FChamroukhi-Habilitation.pdf`.

Chamroukhi, F., H. Glotin, and A. Same. 2013. *Model-Based Functional Mixture Discriminant Analysis with Hidden Process Regression for Curve Classification.* Neurocomputing 112: 153–63. `https://chamroukhi.com/papers/chamroukhi_et_al_neucomp2013a.pdf`.

Chamroukhi, F., and H. Glotin. 2012. *Mixture Model-Based Functional Discriminant Analysis for Curve Classification.* In Proceedings of the International Joint Conference on Neural Networks (IJCNN), IEEE, 1–8. Brisbane, Australia. https://chamroukhi.com/papers/Chamroukhi-ijcnn-2012.pdf.

Same, A., F. Chamroukhi, Gerard Govaert, and P. Aknin. 2011. *Model-Based Clustering and Segmentation of Time Series with Changes in Regime.* Advances in Data Analysis and Classification 5 (4): 301–21. https://chamroukhi.com/papers/adac-2011.pdf.

Chamroukhi, F., A. Same, P. Aknin, and G. Govaert. 2011. *Model-Based Clustering with Hidden Markov Model Regression for Time Series with Regime Changes.* In Proceedings of the International Joint Conference on Neural Networks (IJCNN), IEEE, 2814–21. https://chamroukhi.com/papers/Chamroukhi-ijcnn-2011.pdf.

Chamroukhi, F., A. Same, G. Govaert, and P. Aknin. 2010. *A Hidden Process Regression Model for Functional Data Description. Application to Curve Discrimination.* Neurocomputing 73 (7-9): 1210–21. https://chamroukhi.com/papers/chamroukhi_neucomp_2010.pdf.

Chamroukhi, F. 2010. *Hidden Process Regression for Curve Modeling, Classification and Tracking.* Ph.D. Thesis, Universite de Technologie de Compiegne. https://chamroukhi.com/papers/FChamroukhi-Thesis.pdf.

See Also

Useful links:

- https://github.com/fchamroukhi/FLaMingos
- Report bugs at https://github.com/fchamroukhi/FLaMingos/issues

---

`cemMixRHLP` *cemMixRHLP implements the CEM algorithm to fit a MixRHLP model.*

Description

cemMixRHLP implements the maximum complete likelihood parameter estimation of mixture of RHLP models by the Classification Expectation-Maximization algorithm (CEM algorithm).

Usage

```
cemMixRHLP(X, Y, K, R, p = 3, q = 1,
  variance_type = c("heteroskedastic", "homoskedastic"),
  init_kmeans = TRUE, n_tries = 1, max_iter = 100,
  threshold = 1e-05, verbose = FALSE, verbose_IRLS = FALSE)
```

Arguments

`X`	Numeric vector of length *m* representing the covariates/inputs $x_1, \ldots, x_m$.
`Y`	Matrix of size (n, m) representing the observed responses/outputs. `Y` consists of *n* functions of `X` observed at points $1, \ldots, m$.
`K`	The number of clusters (Number of RHLP models).
`R`	The number of regimes (RHLP components) for each cluster.
`p`	Optional. The order of the polynomial regression. By default, `p` is set at 3.
`q`	Optional. The dimension of the logistic regression. For the purpose of segmentation, it must be set to 1 (which is the default value).
`variance_type`	Optional character indicating if the model is "homoskedastic" or "heteroskedastic". By default the model is "heteroskedastic".
`init_kmeans`	Optional. A logical indicating whether or not the curve partition should be initialized by the K-means algorithm. Otherwise the curve partition is initialized randomly.
`n_tries`	Optional. Number of runs of the EM algorithm. The solution providing the highest log-likelihood will be returned. If `n_tries` > 1, then for the first run, parameters are initialized by uniformly segmenting the data into R segments, and for the next runs, parameters are initialized by randomly segmenting the data into R contiguous segments.
`max_iter`	Optional. The maximum number of iterations for the EM algorithm.
`threshold`	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the EM as stopping criteria.
`verbose`	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during EM iterations.
`verbose_IRLS`	Optional. A logical value indicating whether or not values of the criterion optimized by IRLS should be printed at each step of the EM algorithm.

Details

cemMixRHLP function implements the CEM algorithm. This function starts with an initialization of the parameters done by the method `initParam` of the class ParamMixRHLP, then it alternates between the E-Step, the C-Step (methods of the class StatMixRHLP), and the CM-Step (method of the class ParamMixRHLP) until convergence (until the relative variation of log-likelihood between two steps of the EM algorithm is less than the `threshold` parameter).

Value

EM returns an object of class ModelMixRHLP.

See Also

ModelMixRHLP, ParamMixRHLP, StatMixRHLP

Examples

```
data(toydataset)

#' # Let's fit a mixRHLP model on a dataset containing 2 clusters:
data <- toydataset[1:190,1:21]
x <- data$x
Y <- t(data[,2:ncol(data)])

mixrhlp <- cemMixRHLP(X = x, Y = Y, K = 2, R = 2, p = 1, verbose = TRUE)

mixrhlp$summary()

mixrhlp$plot()
```

`emMixHMM` *emMixHMM implemens the EM (Baum-Welch) algorithm to fit a mixture of HMM models.*

Description

emMixHMM implements the maximum-likelihood parameter estimation of a mixture of HMM models by the Expectation-Maximization (EM) algorithm, known as Baum-Welch algorithm in the context of mixHMM.

Usage

```
emMixHMM(Y, K, R, variance_type = c("heteroskedastic", "homoskedastic"),
  order_constraint = TRUE, init_kmeans = TRUE, n_tries = 1,
  max_iter = 1000, threshold = 1e-06, verbose = FALSE)
```

Arguments

`Y`	Matrix of size (n, m) representing the observed responses/outputs. `Y` consists of *n* functions of `X` observed at points $1, \dots, m$.
`K`	The number of clusters (Number of HMM models).
`R`	The number of regimes (HMM components) for each cluster.
`variance_type`	Optional character indicating if the model is "homoskedastic" or "heteroskedastic". By default the model is "heteroskedastic".
`order_constraint`	Optional. A logical indicating whether or not a mask of order one should be applied to the transition matrix of the Markov chain to provide ordered states. For the purpose of segmentation, it must be set to `TRUE` (which is the default value).
`init_kmeans`	Optional. A logical indicating whether or not the curve partition should be initialized by the K-means algorithm. Otherwise the curve partition is initialized randomly.

`n_tries`	Optional. Number of runs of the EM algorithm. The solution providing the highest log-likelihood will be returned. If `n_tries` > 1, then for the first run, parameters are initialized by uniformly segmenting the data into K segments, and for the next runs, parameters are initialized by randomly segmenting the data into K contiguous segments.
`max_iter`	Optional. The maximum number of iterations for the EM algorithm.
`threshold`	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the EM as stopping criteria.
`verbose`	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during EM iterations.

Details

emMixHMM function implements the EM algorithm. This function starts with an initialization of the parameters done by the method `initParam` of the class ParamMixHMM, then it alternates between the E-Step (method of the class StatMixHMM) and the M-Step (method of the class ParamMixHMM) until convergence (until the relative variation of log-likelihood between two steps of the EM algorithm is less than the `threshold` parameter).

Value

EM returns an object of class ModelMixHMM.

See Also

ModelMixHMM, ParamMixHMM, StatMixHMM

Examples

```
data(toydataset)
Y <- t(toydataset[,2:ncol(toydataset)])

mixhmm <- emMixHMM(Y = Y, K = 3, R = 3, verbose = TRUE)

mixhmm$summary()

mixhmm$plot()
```

`emMixHMMR` *emMixHMMR implements the EM algorithm to fit a mixture if HMMR models.*

Description

emMixHMMR implements the maximum-likelihood parameter estimation of a mixture of HMMR models by the Expectation-Maximization (EM) algorithm.

Usage

```
emMixHMMR(X, Y, K, R, p = 3, variance_type = c("heteroskedastic",
  "homoskedastic"), order_constraint = TRUE, init_kmeans = TRUE,
  n_tries = 1, max_iter = 1000, threshold = 1e-06, verbose = FALSE)
```

Arguments

`X`	Numeric vector of length *m* representing the covariates/inputs $x_1, \ldots, x_m$.
`Y`	Matrix of size (n, m) representing the observed responses/outputs. `Y` consists of *n* functions of `X` observed at points $1, \ldots, m$.
`K`	The number of clusters (Number of HMMR models).
`R`	The number of regimes (HMMR components) for each cluster.
`p`	Optional. The order of the polynomial regression. By default, `p` is set at 3.
`variance_type`	Optional. character indicating if the model is "homoskedastic" or "heteroskedastic". By default the model is "heteroskedastic".
`order_constraint`	Optional. A logical indicating whether or not a mask of order one should be applied to the transition matrix of the Markov chain to provide ordered states. For the purpose of segmentation, it must be set to `TRUE` (which is the default value).
`init_kmeans`	Optional. A logical indicating whether or not the curve partition should be initialized by the K-means algorithm. Otherwise the curve partition is initialized randomly.
`n_tries`	Optional. Number of runs of the EM algorithm. The solution providing the highest log-likelihood will be returned. If `n_tries` > 1, then for the first run, parameters are initialized by uniformly segmenting the data into K segments, and for the next runs, parameters are initialized by randomly segmenting the data into K contiguous segments.
`max_iter`	Optional. The maximum number of iterations for the EM algorithm.
`threshold`	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the EM as stopping criteria.
`verbose`	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during EM iterations.

Details

emMixHMMR function implements the EM algorithm. This function starts with an initialization of the parameters done by the method `initParam` of the class ParamMixHMMR, then it alternates between the E-Step (method of the class StatMixHMMR) and the M-Step (method of the class ParamMixHMMR) until convergence (until the relative variation of log-likelihood between two steps of the EM algorithm is less than the `threshold` parameter).

Value

EM returns an object of class ModelMixHMMR.

See Also

ModelMixHMMR, ParamMixHMMR, StatMixHMMR

Examples

```
data(toydataset)
x <- toydataset$x
Y <- t(toydataset[,2:ncol(toydataset)])

mixhmmr <- emMixHMMR(X = x, Y = Y, K = 3, R = 3, p = 1, verbose = TRUE)

mixhmmr$summary()

mixhmmr$plot()
```

`emMixRHLP` *emMixRHLP implements the EM algorithm to fit a mixture of RHLP models.*

Description

emMixRHLP implements the maximum-likelihood parameter estimation of a mixture of RHLP models by the Expectation-Maximization (EM) algorithm.

Usage

```
emMixRHLP(X, Y, K, R, p = 3, q = 1,
  variance_type = c("heteroskedastic", "homoskedastic"),
  init_kmeans = TRUE, n_tries = 1, max_iter = 1000,
  threshold = 1e-05, verbose = FALSE, verbose_IRLS = FALSE)
```

Arguments

`X`	Numeric vector of length *m* representing the covariates/inputs $x_1, \dots, x_m$.
`Y`	Matrix of size (n, m) representing the observed responses/outputs. `Y` consists of *n* functions of `X` observed at points $1, \dots, m$.
`K`	The number of clusters (Number of RHLP models).
`R`	The number of regimes (RHLP components) for each cluster.
`p`	Optional. The order of the polynomial regression. By default, `p` is set at 3.
`q`	Optional. The dimension of the logistic regression. For the purpose of segmentation, it must be set to 1 (which is the default value).
`variance_type`	Optional character indicating if the model is "homoskedastic" or "heteroskedastic". By default the model is "heteroskedastic".
`init_kmeans`	Optional. A logical indicating whether or not the curve partition should be initialized by the K-means algorithm. Otherwise the curve partition is initialized randomly.

`n_tries`	Optional. Number of runs of the EM algorithm. The solution providing the highest log-likelihood will be returned. If `n_tries` > 1, then for the first run, parameters are initialized by uniformly segmenting the data into R segments, and for the next runs, parameters are initialized by randomly segmenting the data into R contiguous segments.
`max_iter`	Optional. The maximum number of iterations for the EM algorithm.
`threshold`	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the EM as stopping criteria.
`verbose`	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during EM iterations.
`verbose_IRLS`	Optional. A logical value indicating whether or not values of the criterion optimized by IRLS should be printed at each step of the EM algorithm.

Details

emMixRHLP function implements the EM algorithm. This function starts with an initialization of the parameters done by the method `initParam` of the class ParamMixRHLP, then it alternates between the E-Step (method of the class StatMixRHLP) and the M-Step (method of the class ParamMixRHLP) until convergence (until the relative variation of log-likelihood between two steps of the EM algorithm is less than the `threshold` parameter).

Value

EM returns an object of class ModelMixRHLP.

See Also

ModelMixRHLP, ParamMixRHLP, StatMixRHLP

Examples

```
data(toydataset)

# Let's fit a mixRHLP model on a dataset containing 2 clusters:
data <- toydataset[1:190,1:21]
x <- data$x
Y <- t(data[,2:ncol(data)])

mixrhlp <- emMixRHLP(X = x, Y = Y, K = 2, R = 2, p = 1, verbose = TRUE)

mixrhlp$summary()

mixrhlp$plot()
```

`FData-class` *A Reference Class which represents functional data.*

Description

FData is a reference class which represents general independent and identically distributed (i.i.d.) functional objects. The data can be ordered by time (functional time series). In the last case, the field `X` represents the time.

Fields

`X` Numeric vector of length *m* representing the covariates/inputs.

`Y` Matrix of size (n, m) representing the observed responses/outputs. `Y` consists of *n* functions of `X` observed at points $1, \dots, m$.

`mkStochastic` *mkStochastic ensures that it is a stochastic vector, matrix or array.*

Description

mkStochastic ensures that it is a stochastic vector, matrix or array.

Usage

```
mkStochastic(M)
```

Arguments

`M`	A vector, matrix or array to transform.

Details

mkStochastic ensures that the giving argument is a stochastic vector, matrix or array, i.e., that the sum over the last dimension is 1.

Value

A vector, matrix or array for which the sum over the last dimension is 1.

`ModelMixHMM-class` *A Reference Class which represents a fitted Mixture of HMM model.*

Description

ModelMixHMM represents an estimated mixture of HMM model.

Fields

`param` A ParamMixHMM object. It contains the estimated values of the parameters.

`stat` A StatMixHMM object. It contains all the statistics associated to the MixHMM model.

Methods

`plot(what = c("clustered", "smoothed", "loglikelihood"), ...)` Plot method

- `what` The type of graph requested:
    - `"clustered" =` Clustered curves (field `klas` of class StatMixHMM).
    - `"smoothed" =` Smoothed signal (field `smoothed` of class StatMixHMM).
    - `"loglikelihood" =` Value of the log-likelihood for each iteration (field `stored_loglik` of class StatMixHMM).
- `...` Other graphics parameters.

`summary(digits = getOption("digits"))` Summary method.

- `digits` The number of significant digits to use when printing.

See Also

ParamMixHMM, StatMixHMM

Examples

```
data(toydataset)
Y <- t(toydataset[,2:ncol(toydataset)])

mixhmm <- emMixHMM(Y = Y, K = 3, R = 3, verbose = TRUE)

# mixhmm is a ModelMixHMM object. It contains some methods such as 'summary' and 'plot'
mixhmm$summary()
mixhmm$plot()

# mixhmm has also two fields, stat and param which are reference classes as well

# Log-likelihood:
mixhmm$stat$loglik

# Means
mixhmm$param$mu
```

`ModelMixHMMR-class` *A Reference Class which represents a fitted mixture of HMMR model.*

Description

ModelMixHMMR represents an estimated mixture of HMMR model.

Fields

`param` A ParamMixHMMR object. It contains the estimated values of the parameters.

`stat` A StatMixHMMR object. It contains all the statistics associated to the MixHMMR model.

Methods

`plot(what = c("clustered", "smoothed", "loglikelihood"), ...)` Plot method

  `what` The type of graph requested:
  - `"clustered" =` Clustered curves (field `klas` of class StatMixHMMR).
  - `"smoothed" =` Smoothed signal (field `smoothed` of class StatMixHMMR).
  - `"loglikelihood" =` Value of the log-likelihood for each iteration (field `stored_loglik` of class StatMixHMMR).

  `...` Other graphics parameters.

`summary(digits = getOption("digits"))` Summary method.

  `digits` The number of significant digits to use when printing.

See Also

ParamMixHMMR, StatMixHMMR

Examples

```
data(toydataset)
x <- toydataset$x
Y <- t(toydataset[,2:ncol(toydataset)])

mixhmmr <- emMixHMMR(X = x, Y = Y, K = 3, R = 3, p = 1, verbose = TRUE)

# mixhmmr is a ModelMixHMMR object. It contains some methods such as 'summary' and 'plot'
mixhmmr$summary()
mixhmmr$plot()

# mixhmmr has also two fields, stat and param which are reference classes as well

# Log-likelihood:
mixhmmr$stat$loglik

# Parameters of the polynomial regressions:
mixhmmr$param$beta
```

`ModelMixRHLP-class` *A Reference Class which represents a fitted mixture of RHLP model.*

Description

ModelMixRHLP represents an estimated mixture of RHLP model.

Fields

`param` A ParamMixRHLP object. It contains the estimated values of the parameters.

`stat` A StatMixRHLP object. It contains all the statistics associated to the MixRHLP model.

Methods

`plot(what = c("estimatedsignal", "regressors", "loglikelihood"), ...)` Plot method.

`what` The type of graph requested:

- `"estimatedsignal" =` Estimated signal (field `Ey` of class StatMixRHLP).
- `"regressors" =` Polynomial regression components (fields `polynomials` and `pi_jkr` of class StatMixRHLP).
- `"loglikelihood" =` Value of the log-likelihood for each iteration (field `stored_loglik` of class StatMixRHLP).

`...` Other graphics parameters.

By default, all the above graphs are produced.

`summary(digits = getOption("digits"))` Summary method.

`digits` The number of significant digits to use when printing.

See Also

ParamMixRHLP, StatMixRHLP

Examples

```
data(toydataset)

# Let's fit a mixRHLP model on a dataset containing 2 clusters:
data <- toydataset[1:190,1:21]
x <- data$x
Y <- t(data[,2:ncol(data)])

mixrhlp <- cemMixRHLP(X = x, Y = Y, K = 2, R = 2, p = 1, verbose = TRUE)

# mixrhlp is a ModelMixRHLP object. It contains some methods such as 'summary' and 'plot'
mixrhlp$summary()
mixrhlp$plot()

# mixrhlp has also two fields, stat and param which are reference classes as well
```

```
# Log-likelihood:
mixrhlp$stat$loglik

# Parameters of the polynomial regressions:
mixrhlp$param$beta
```

`ParamMixHMM-class` *A Reference Class which contains parameters of a mixture of HMM models.*

Description

ParamMixHMM contains all the parameters of a mixture of HMM models.

Fields

`fData` FData object representing the sample (covariates/inputs `X` and observed responses/outputs `Y`).

`K` The number of clusters (Number of HMM models).

`R` The number of regimes (HMM components) for each cluster.

`variance_type` Character indicating if the model is homoskedastic (`variance_type = "homoskedastic"`) or heteroskedastic (`variance_type = "heteroskedastic"`). By default the model is heteroskedastic.

`order_constraint` A logical indicating whether or not a mask of order one should be applied to the transition matrix of the Markov chain to provide ordered states. For the purpose of segmentation, it must be set to `TRUE` (which is the default value).

`alpha` Cluster weights. Matrix of dimension $(K, 1)$.

`prior` The prior probabilities of the Markov chains. `prior` is a matrix of dimension (R, K). The k-th column represents the prior distribution of the Markov chain asociated to the cluster k.

`trans_mat` The transition matrices of the Markov chains. `trans_mat` is an array of dimension (R, R, K).

`mask` Mask applied to the transition matrices `trans_mat`. By default, a mask of order one is applied.

`mu` Means. Matrix of dimension (R, K). The k-th column gives represents the k-th cluster and gives the means for the `R` regimes.

`sigma2` The variances for the `K` clusters. If MixHMM model is heteroskedastic (`variance_type = "heteroskedastic"`) then `sigma2` is a matrix of size (R, K) (otherwise MixHMM model is homoskedastic (`variance_type = "homoskedastic"`) and `sigma2` is a matrix of size $(1, K)$).

`nu` The degrees of freedom of the MixHMM model representing the complexity of the model.

Methods

`initGaussParamHmm(Y, k, R, variance_type, try_algo)` Initialize the means `mu` and `sigma2` for the cluster `k`.

`initParam(init_kmeans = TRUE, try_algo = 1)` Method to initialize parameters `alpha`, `prior`, `trans_mat`, `mu` and `sigma2`.

If `init_kmeans = TRUE` then the curve partition is initialized by the K-means algorithm. Otherwise the curve partition is initialized randomly.

If `try_algo = 1` then `mu` and `sigma2` are initialized by segmenting the time series `Y` uniformly into `R` contiguous segments. Otherwise, `mu` and `sigma2` are initialized by segmenting randomly the time series `Y` into `R` segments.

`MStep(statMixHMM)` Method which implements the M-step of the EM algorithm to learn the parameters of the MixHMM model based on statistics provided by the object `statMixHMM` of class StatMixHMM (which contains the E-step).

`ParamMixHMMR-class` *A Reference Class which contains parameters of a mixture of HMMR models.*

Description

ParamMixHMMR contains all the parameters of a mixture of HMMR models.

Fields

`fData` FData object representing the sample (covariates/inputs `X` and observed responses/outputs `Y`).

`K` The number of clusters (Number of HMMR models).

`R` The number of regimes (HMMR components) for each cluster.

`p` The order of the polynomial regression.

`variance_type` Character indicating if the model is homoskedastic (`variance_type = "homoskedastic"`) or heteroskedastic (`variance_type = "heteroskedastic"`). By default the model is heteroskedastic.

`order_constraint` A logical indicating whether or not a mask of order one should be applied to the transition matrix of the Markov chain to provide ordered states. For the purpose of segmentation, it must be set to `TRUE` (which is the default value).

`alpha` Cluster weights. Matrix of dimension $(K, 1)$.

`prior` The prior probabilities of the Markov chains. `prior` is a matrix of dimension (R, K). The k-th column represents the prior distribution of the Markov chain asociated to the cluster k.

`trans_mat` The transition matrices of the Markov chains. `trans_mat` is an array of dimension (R, R, K).

`mask` Mask applied to the transition matrices `trans_mat`. By default, a mask of order one is applied.

`beta` Parameters of the polynomial regressions. `beta` is an array of dimension $(p+1, R, K)$, with `p` the order of the polynomial regression. `p` is fixed to 3 by default.

`sigma2` The variances for the `K` clusters. If MixHMMR model is heteroskedastic (`variance_type = "heteroskedastic"`) then `sigma2` is a matrix of size (R, K) (otherwise MixHMMR model is homoskedastic (`variance_type = "homoskedastic"`) and `sigma2` is a matrix of size

`nu` The degree of freedom of the MixHMMR model representing the complexity of the model.

`phi` A list giving the regression design matrix for the polynomial regressions.

Methods

`initParam(init_kmeans = TRUE, try_algo = 1)` Method to initialize parameters `alpha`, `prior`, `trans_mat`, `beta` and `sigma2`.

If `init_kmeans = TRUE` then the curve partition is initialized by the K-means algorithm. Otherwise the curve partition is initialized randomly.

If `try_algo = 1` then `beta` and `sigma2` are initialized by segmenting the time series `Y` uniformly into `R` contiguous segments. Otherwise, `beta` and `sigma2` are initialized by segmenting randomly the time series `Y` into `R` segments.

`initRegressionParam(Y, k, R, phi, variance_type, try_algo)` Initialize `beta` and `sigma2` for the cluster `k`.

`MStep(statMixHMMR)` Method which implements the M-step of the EM algorithm to learn the parameters of the MixHMMR model based on statistics provided by the object `statMixHMMR` of class StatMixHMMR (which contains the E-step).

`ParamMixRHLP-class` *A Reference Class which contains parameters of a mixture of RHLP models.*

Description

ParamMixRHLP contains all the parameters of a mixture of RHLP models.

Fields

`fData` FData object representing the sample (covariates/inputs `X` and observed responses/outputs `Y`).

`K` The number of clusters (Number of RHLP models).

`R` The number of regimes (RHLP components) for each cluster.

`p` The order of the polynomial regression.

`q` The dimension of the logistic regression. For the purpose of segmentation, it must be set to 1.

`variance_type` Character indicating if the model is homoskedastic (`variance_type = "homoskedastic"`) or heteroskedastic (`variance_type = "heteroskedastic"`). By default the model is heteroskedastic.

`alpha` Cluster weights. Matrix of dimension $(1, K)$.

`W` Parameters of the logistic process. $\boldsymbol{W} = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_K)$ is an array of dimension $(q + 1, R - 1, K)$, with $\boldsymbol{w}_k = (\boldsymbol{w}_{k,1}, \dots, \boldsymbol{w}_{k,R-1})$, $k = 1, \dots, K$, and `q` the order of the logistic regression. `q` is fixed to 1 by default.

`beta` Parameters of the polynomial regressions. $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K)$ is an array of dimension $(p + 1, R, K)$, with $\boldsymbol{\beta}_k = (\boldsymbol{\beta}_{k,1}, \dots, \boldsymbol{\beta}_{k,R})$, $k = 1, \dots, K$, `p` the order of the polynomial regression. `p` is fixed to 3 by default.

`sigma2` The variances for the `K` clusters. If MixRHLP model is heteroskedastic (`variance_type = "heteroskedastic"`) then `sigma2` is a matrix of size (R, K) (otherwise MixRHLP model is homoskedastic (`variance_type = "homoskedastic"`) and `sigma2` is a matrix of size $(K, 1)$).

`nu` The degree of freedom of the MixRHLP model representing the complexity of the model.

`phi` A list giving the regression design matrices for the polynomial and the logistic regressions.

Methods

`CMStep(statMixRHLP, verbose_IRLS = FALSE)` Method which implements the M-step of the CEM algorithm to learn the parameters of the MixRHLP model based on statistics provided by the object `statMixRHLP` of class StatMixRHLP (which contains the E-step and the C-step).

`initParam(init_kmeans = TRUE, try_algo = 1)` Method to initialize parameters `alpha`, `W`, `beta` and `sigma2`.

If `init_kmeans = TRUE` then the curve partition is initialized by the R-means algorithm. Otherwise the curve partition is initialized randomly.

If `try_algo = 1` then `beta` and `sigma2` are initialized by segmenting the time series `Y` uniformly into `R` contiguous segments. Otherwise, `W`, `beta` and `sigma2` are initialized by segmenting randomly the time series `Y` into `R` segments.

`initRegressionParam(Yk, k, try_algo = 1)` Initialize the matrix of polynomial regression coefficients beta_k for the cluster `k`.

`MStep(statMixRHLP, verbose_IRLS = FALSE)` Method which implements the M-step of the EM algorithm to learn the parameters of the MixRHLP model based on statistics provided by the object `statMixRHLP` of class StatMixRHLP (which contains the E-step).

`StatMixHMM-class` *A Reference Class which contains statistics of a mixture of HMM model.*

Description

StatMixHMM contains all the statistics associated to a MixHMM model, in particular the E-Step of the EM algorithm.

Fields

`tau_ik` Matrix of size (n, K) giving the posterior probabilities that the curve $\boldsymbol{y}_i$ originates from the k-th HMM model.

`gamma_ikjr` Array of size (nm, R, K) giving the posterior probabilities that the observation $\boldsymbol{y}_{ij}$ originates from the r-th regime of the k-th HMM model.

`loglik` Numeric. Log-likelihood of the MixHMM model.

`stored_loglik` Numeric vector. Stored values of the log-likelihood at each iteration of the EM algorithm.

`klas` Row matrix of the labels issued from `tau_ik`. Its elements are $klas[i] = z\_i$, $i = 1, \dots, n$.

`z_ik` Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z\_ik = 1 \text{ if } z\_i = \arg\max_k P(z_{ik} = 1 | \boldsymbol{y}_i; \boldsymbol{\Psi}) = tau\_tk;\ 0$ otherwise.

`smoothed` Matrix of size (m, K) giving the smoothed time series. The smoothed time series are computed by combining the time series $\boldsymbol{y}_i$ with both the estimated posterior regime probabilities `gamma_ikjr` and the corresponding estimated posterior cluster probability `tau_ik`. The k-th column gives the estimated mean series of cluster k.

`BIC` Numeric. Value of BIC (Bayesian Information Criterion).

`AIC` Numeric. Value of AIC (Akaike Information Criterion).

`ICL1` Numeric. Value of ICL (Integrated Completed Likelihood Criterion).

`log_alpha_k_fyi` Private. Only defined for calculations.

`exp_num_trans` Private. Only defined for calculations.

`exp_num_trans_from_l` Private. Only defined for calculations.

Methods

`computeStats(paramMixHMM)` Method used in the EM algorithm to compute statistics based on parameters provided by the object `paramMixHMM` of class ParamMixHMM.

`EStep(paramMixHMM)` Method used in the EM algorithm to update statistics based on parameters provided by the object `paramMixHMM` of class ParamMixHMM (prior and posterior probabilities).

`MAP()` MAP calculates values of the fields `z_ik` and `klas` by applying the Maximum A Posteriori Bayes allocation rule.

$z\_ik = 1 \text{ if } z\_i = \arg\max_k P(z_{ik} = 1 | \boldsymbol{y}_i; \boldsymbol{\Psi}) = tau\_tk;\ 0$ otherwise.

See Also

ParamMixHMM

`StatMixHMMR-class` *A Reference Class which contains statistics of a mixture of HMMR models.*

Description

StatMixHMMR contains all the statistics associated to a MixHMMR model, in particular the E-Step of the EM algorithm.

Fields

`tau_ik` Matrix of size (n, K) giving the posterior probabilities that the curve $\boldsymbol{y}_i$ originates from the k-th HMMR model.

`gamma_ikjr` Array of size (nm, R, K) giving the posterior probabilities that the observation $\boldsymbol{y}_{ij}$ originates from the r-th regime of the k-th HMM model.

`loglik` Numeric. Log-likelihood of the MixHMMR model.

`stored_loglik` Numeric vector. Stored values of the log-likelihood at each iteration of the EM algorithm.

`klas` Row matrix of the labels issued from `tau_ik`. Its elements are $klas[i] = z\_i$, $i = 1, \dots, n$.

`z_ik` Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z\_ik = 1 \text{ if } z\_i = \text{arg max}_k\ P(z_{ik} = 1 | \boldsymbol{y}_i; \boldsymbol{\Psi}) = tau\_ik;\ 0 \text{ otherwise}.$

`smoothed` Matrix of size (m, K) giving the smoothed time series. The smoothed time series are computed by combining the polynomial regression components with both the estimated posterior regime probabilities `gamma_ikjr` and the corresponding estimated posterior cluster probability `tau_ik`. The k-th column gives the estimated mean series of cluster k.

`BIC` Numeric. Value of BIC (Bayesian Information Criterion).

`AIC` Numeric. Value of AIC (Akaike Information Criterion).

`ICL1` Numeric. Value of ICL (Integrated Completed Likelihood Criterion).

`log_alpha_k_fyi` Private. Only defined for calculations.

`exp_num_trans` Private. Only defined for calculations.

`exp_num_trans_from_l` Private. Only defined for calculations.

Methods

`computeStats(paramMixHMMR)` Method used in the EM algorithm to compute statistics based on parameters provided by the object `paramMixHMMR` of class ParamMixHMMR.

`EStep(paramMixHMMR)` Method used in the EM algorithm to update statistics based on parameters provided by the object `paramMixHMMR` of class ParamMixHMMR (prior and posterior probabilities).

`MAP()` MAP calculates values of the fields `z_ik` and `klas` by applying the Maximum A Posteriori Bayes allocation rule.

$z\_ik = 1 \text{ if } z\_i = \text{arg max}_k\ P(z_{ik} = 1 | \boldsymbol{y}_i; \boldsymbol{\Psi}) = tau\_ik;\ 0 \text{ otherwise}.$

See Also

ParamMixHMMR

`StatMixRHLP-class` *A Reference Class which contains statistics of a mixture of RHLP models.*

Description

StatMixRHLP contains all the statistics associated to a MixRHLP model, in particular the E-Step (and C-Step) of the (C)EM algorithm.

Fields

`pi_jkr` Array of size (nm, R, K) representing the logistic proportion for cluster k.

`tau_ik` Matrix of size (n, K) giving the posterior probabilities (fuzzy segmentation matrix) that the curve $\boldsymbol{y}_i$ originates from the k-th RHLP model.

`z_ik` Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z\_ik = 1$ if $z\_i = \arg\max_k\ tau\_ik;\ 0$ otherwise.

`klas` Column matrix of the labels issued from `z_ik`. Its elements are $klas[i] = z\_i$, $i = 1, \dots, n$.

`gamma_ijkr` Array of size (nm, R, K) giving the posterior probabilities that the observation $\boldsymbol{y}_{ij}$ originates from the r-th regime of the k-th RHLP model.

`polynomials` Array of size (m, R, K) giving the values of the estimated polynomial regression components.

`weighted_polynomials` Array of size (m, R, K) giving the values of the estimated polynomial regression components weighted by the prior probabilities `pi_jkr`.

`Ey` Matrix of size *(m, K)*. `Ey` is the curve expectation (estimated signal): sum of the polynomial components weighted by the logistic probabilities `pi_jkr`.

`loglik` Numeric. Observed-data log-likelihood of the MixRHLP model.

`com_loglik` Numeric. Complete-data log-likelihood of the MixRHLP model.

`stored_loglik` Numeric vector. Stored values of the log-likelihood at each EM iteration.

`stored_com_loglik` Numeric vector. Stored values of the Complete log-likelihood at each EM iteration.

`BIC` Numeric. Value of BIC (Bayesian Information Criterion).

`ICL` Numeric. Value of ICL (Integrated Completed Likelihood).

`AIC` Numeric. Value of AIC (Akaike Information Criterion).

`log_fk_yij` Matrix of size (n, K) giving the values of the probability density function $f(\boldsymbol{y}_i|z_i = k, \boldsymbol{x}, \boldsymbol{\Psi})$, $i = 1, \dots, n$.

`log_alphak_fk_yij` Matrix of size (n, K) giving the values of the logarithm of the joint probability density function $f(\boldsymbol{y}_i,\ z_i = k|\boldsymbol{x}, \boldsymbol{\Psi})$, $i = 1, \dots, n$.

`log_gamma_ijkr` Array of size (nm, R, K) giving the logarithm of `gamma_ijkr`.

Methods

`computeStats(paramMixRHLP)` Method used in the EM algorithm to compute statistics based on parameters provided by the object `paramMixRHLP` of class ParamMixRHLP.

`CStep(reg_irls)` Method used in the CEM algorithm to update statistics.

`EStep(paramMixRHLP)` Method used in the EM algorithm to update statistics based on parameters provided by the object `paramMixRHLP` of class ParamMixRHLP (prior and posterior probabilities).

`MAP()` MAP calculates values of the fields `z_ik` and `klas` by applying the Maximum A Posteriori Bayes allocation rule.

$z\_ik = 1$ if $z\_i = \text{arg max}_k\ tau\_ik;\ 0$ otherwise.

See Also

ParamMixRHLP

`toydataset` *A dataset composed of simulated time series with regime changes.*

Description

A dataset composed of 30 simulated time series with regime changes.

Usage

```
toydataset
```

Format

A data frame with 350 rows and 31 variables:

x The covariate variable which is the time in that case.

y1 Times series with a wave form shape and for which a normally distributed random noise has been added.

y2 Same as `y1`.

y3 Same as `y1`.

y4 Same as `y1`.

y5 Same as `y1`.

y6 Same as `y1`.

y7 Same as `y1`.

y8 Same as `y1`.

y9 Same as `y1`.

y10 Same as `y1`.

y11 Time series generated as follows:

  - First regime: 120 values of Normally distributed random numbers with mean 5 and variance 1.
  - Second regime: 70 values of Normally distributed random numbers with mean 7 and variance 1.
  - Third regime: 160 values of Normally distributed random numbers with mean 5 variance 1.

y12 Same as `y11`.

y13 Same as `y11`.

y14 Same as `y11`.

y15 Same as `y11`.

y16 Same as `y11`.

y17 Same as `y11`.

y18 Same as `y11`.

y19 Same as `y11`.

y20 Same as `y11`.

y21 Time series generated as follows:

  - First regime: 80 values of Normally distributed random numbers with mean 7 variance 1.
  - Second regime: 130 values of Normally distributed random numbers with mean 5 variance 1.
  - Third regime: 140 values of Normally distributed random numbers with mean 4 variance 1.

y22 Same as `y21`.

y23 Same as `y21`.

y24 Same as `y21`.

y25 Same as `y21`.

y26 Same as `y21`.

y27 Same as `y21`.

y28 Same as `y21`.

y29 Same as `y21`.

y30 Same as `y21`.

Index

∗ **datasets**
    toydataset, 22

cemMixRHLP, 4

emMixHMM, 6
emMixHMMR, 7
emMixRHLP, 9

FData, *15–17*
FData *(*FData-class*)*, 11
FData-class, 11
flamingos *(*flamingos-package*)*, 3
flamingos-package, 3

MixHMM, *18*
MixHMMR, *19*
MixRHLP, *21*
mkStochastic, 11
ModelMixHMM, *7*
ModelMixHMM *(*ModelMixHMM-class*)*, 12
ModelMixHMM-class, 12
ModelMixHMMR, *8*, *9*
ModelMixHMMR *(*ModelMixHMMR-class*)*, 13
ModelMixHMMR-class, 13
ModelMixRHLP, *5*, *10*
ModelMixRHLP *(*ModelMixRHLP-class*)*, 14
ModelMixRHLP-class, 14

ParamMixHMM, *7*, *12*, *19*
ParamMixHMM *(*ParamMixHMM-class*)*, 15
ParamMixHMM-class, 15
ParamMixHMMR, *8*, *9*, *13*, *20*
ParamMixHMMR *(*ParamMixHMMR-class*)*, 16
ParamMixHMMR-class, 16
ParamMixRHLP, *5*, *10*, *14*, *22*
ParamMixRHLP *(*ParamMixRHLP-class*)*, 17
ParamMixRHLP-class, 17

StatMixHMM, *7*, *12*, *16*
StatMixHMM *(*StatMixHMM-class*)*, 18
StatMixHMM-class, 18
StatMixHMMR, *8*, *9*, *13*, *17*
StatMixHMMR *(*StatMixHMMR-class*)*, 19
StatMixHMMR-class, 19
StatMixRHLP, *5*, *10*, *14*, *18*
StatMixRHLP *(*StatMixRHLP-class*)*, 21
StatMixRHLP-class, 21

toydataset, 22