

Package ‘meteorits’

October 13, 2022

Type Package

Title Mixture-of-Experts Modeling for Complex Non-Normal Distributions

Version 0.1.1

Description Provides a unified mixture-of-experts (ME) modeling and estimation framework with several original and flexible ME models to model, cluster and classify heterogeneous data in many complex situations where the data are distributed according to non-normal, possibly skewed distributions, and when they might be corrupted by atypical observations. Mixtures-of-Experts models for complex and non-normal distributions ('meteorits') are originally introduced and written in 'Matlab' by Faicel Chamroukhi. The references are mainly the following ones. The references are mainly the following ones.
Chamroukhi F., Same A., Govaert, G. and Aknin P. (2009) <doi:10.1016/j.neunet.2009.06.040>.
Chamroukhi F. (2010) <<https://chamroukhi.com/FChamroukhi-PhD.pdf>>.
Chamroukhi F. (2015) <[arXiv:1506.06707](https://arxiv.org/abs/1506.06707)>.
Chamroukhi F. (2015) <<https://chamroukhi.com/FChamroukhi-HDR.pdf>>.
Chamroukhi F. (2016) <doi:10.1109/IJCNN.2016.7727580>.
Chamroukhi F. (2016) <doi:10.1016/j.neunet.2016.03.002>.
Chamroukhi F. (2017) <doi:10.1016/j.neucom.2017.05.044>.

URL <https://github.com/fchamroukhi/MEeteorits>

BugReports <https://github.com/fchamroukhi/MEeteorits/issues>

License GPL (>= 3)

Depends R (>= 2.10)

Imports pracma, methods, stats, MASS, Rcpp

Suggests knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo

Collate meteorits-package.R RcppExports.R logsumexp.R utils.R
sampleUnivNMoE.R sampleUnivSNMoE.R sampleUnivStMoE.R
sampleUnivTMoE.R ParamSNMoE.R ParamStMoE.R ParamTMoE.R
ParamNMoE.R StatSNMoE.R StatStMoE.R StatTMoE.R StatNMoE.R
ModelSNMoE.R ModelStMoE.R ModelTMoE.R ModelNMoE.R emSNMoE.R
emStMoE.R emTMoE.R emNMoE.R data-tempanomalies.R

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation yes

Author Faicel Chamroukhi [aut] (<<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 2020-01-10 16:00:02 UTC

R topics documented:

meteorits-package	3
emNMoE	4
emSNMoE	5
emStMoE	7
emTMoE	8
ModelNMoE-class	9
ModelSNMoE-class	10
ModelStMoE-class	12
ModelTMoE-class	13
ParamNMoE-class	14
ParamSNMoE-class	15
ParamStMoE-class	16
ParamTMoE-class	17
sampleUnivNMoE	18
sampleUnivSNMoE	19
sampleUnivStMoE	20
sampleUnivTMoE	22
StatNMoE-class	23
StatSNMoE-class	24
StatStMoE-class	25
StatTMoE-class	27
tempanomalies	28

Index

29

meteorits-package	<i>MEteorits: Mixtures-of-Experts modeling for cOmplex and non-noRmal dIsTributions</i>
-------------------	---

Description

meteorits is a package containing several original and flexible mixtures-of-experts models to model, cluster and classify heterogeneous data in many complex situations where the data are distributed according to non-normal and possibly skewed distributions, and when they might be corrupted by atypical observations. The toolbox also contains sparse mixture-of-experts models for high-dimensional data.

meteorits contains the following Mixture-of-Experts models:

- NMoE (Normal Mixtures-of-Experts) provides a flexible framework for heterogeneous data with Normal expert regressors network;
- SNMoE (Skew-Normal Mixtures-of-Experts) provides a flexible modeling framework for heterogeneous data with possibly skewed distributions to generalize the standard Normal mixture of expert model;
- tMoE (t Mixtures-of-Experts) provides a flexible and robust modeling framework for heterogeneous data with possibly heavy-tailed distributions and corrupted by atypical observations;
- StMoE (Skew t Mixtures-of-Experts) provides a flexible and robust modeling framework for heterogeneous data with possibly skewed, heavy-tailed distributions and corrupted by atypical observations.

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

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

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, F. 2017. *Skew-T Mixture of Experts*. Neurocomputing - Elsevier 266: 390–408. <https://chamroukhi.com/papers/STMoE.pdf>.

Chamroukhi, F. 2016a. *Robust Mixture of Experts Modeling Using the T-Distribution*. Neural Networks - Elsevier 79: 20–36. <https://chamroukhi.com/papers/TMoE.pdf>.

Chamroukhi, F. 2016b. *Skew-Normal Mixture of Experts*. In The International Joint Conference on Neural Networks (IJCNN). Vancouver, Canada. <https://chamroukhi.com/papers/Chamroukhi-SNMoE-IJCNN2016.pdf>.

- Chamroukhi, F. 2015a. *Non-Normal Mixtures of Experts*. <http://arxiv.org/pdf/1506.06707.pdf>.
- Chamroukhi, F. 2015b. *Statistical Learning of Latent Data Models for Complex Data Analysis*. Habilitation Thesis (HDR), Universite de Toulon. <https://chamroukhi.com/FChamroukhi-HDR.pdf>.
- Chamroukhi, F. 2010. *Hidden Process Regression for Curve Modeling, Classification and Tracking*. Ph.D. Thesis, Universite de Technologie de Compiegne. <https://chamroukhi.com/FChamroukhi-PhD.pdf>.
- Chamroukhi, F., A. Same, G. Govaert, and P. Aknin. 2009. *Time Series Modeling by a Regression Approach Based on a Latent Process*. *Neural Networks* 22 (5-6): 593–602. https://chamroukhi.com/papers/Chamroukhi_Neural_Networks_2009.pdf.

See Also

Useful links:

- <https://github.com/fchamroukhi/MEteorits>
- Report bugs at <https://github.com/fchamroukhi/MEteorits/issues>

emNMoE	<i>emNMoE implements the EM algorithm to fit a Normal Mixture of Experts (NMoE).</i>
--------	--

Description

emNMoE implements the maximum-likelihood parameter estimation of a Normal Mixture of Experts (NMoE) model by the Expectation-Maximization (EM) algorithm.

Usage

```
emNMoE(X, Y, K, p = 3, q = 1, n_tries = 1, max_iter = 1500,
        threshold = 1e-06, verbose = FALSE, verbose_IRLS = FALSE)
```

Arguments

X	Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
Y	Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
K	The number of experts.
p	Optional. The order of the polynomial regression for the experts.
q	Optional. The order of the logistic regression for the gating network.
n_tries	Optional. Number of runs of the EM algorithm. The solution providing the highest log-likelihood will be returned.
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

emNMoe function implements the EM algorithm for the NMoe model. This function starts with an initialization of the parameters done by the method `initParam` of the class [ParamNMoe](#), then it alternates between the E-Step (method of the class [StatNMoe](#)) and the M-Step (method of the class [ParamNMoe](#)) 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 [ModelNMoe](#).

See Also

[ModelNMoe](#), [ParamNMoe](#), [StatNMoe](#)

Examples

```
data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

nmoe <- emNMoe(X = x, Y = y, K = 2, p = 1, verbose = TRUE)

nmoe$summary()

nmoe$plot()
```

emSNMoE	<i>emSNMoE implements the ECM algorithm to fit a Skew-Normal Mixture of Experts (SNMoE).</i>
---------	--

Description

emSNMoE implements the maximum-likelihood parameter estimation of a Skew-Normal Mixture of Experts (SNMoE) model by the Expectation Conditional Maximization (ECM) algorithm.

Usage

```
emSNMoE(X, Y, K, p = 3, q = 1, n_tries = 1, max_iter = 1500,
        threshold = 1e-06, verbose = FALSE, verbose_IRLS = FALSE)
```

Arguments

X	Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
Y	Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
K	The number of experts.
p	Optional. The order of the polynomial regression for the experts.
q	Optional. The order of the logistic regression for the gating network.
n_tries	Optional. Number of runs of the ECM algorithm. The solution providing the highest log-likelihood will be returned.
max_iter	Optional. The maximum number of iterations for the ECM algorithm.
threshold	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the ECM as stopping criteria.
verbose	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during ECM 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 ECM algorithm.

Details

emSNMoE function implements the ECM algorithm for the SNMoE model. This function starts with an initialization of the parameters done by the method `initParam` of the class [ParamSNMoE](#), then it alternates between the E-Step (method of the class [StatSNMoE](#)) and the M-Step (method of the class [ParamSNMoE](#)) until convergence (until the relative variation of log-likelihood between two steps of the ECM algorithm is less than the `threshold` parameter).

Value

ECM returns an object of class [ModelSNMoE](#).

See Also

[ModelSNMoE](#), [ParamSNMoE](#), [StatSNMoE](#)

Examples

```
data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

snmoe <- emSNMoE(X = x, Y = y, K = 2, p = 1, verbose = TRUE)

snmoe$summary()

snmoe$plot()
```

emStMoE	<i>emStMoE implements the ECM algorithm to fit a Skew-t Mixture of Experts (StMoE).</i>
---------	---

Description

emStMoE implements the maximum-likelihood parameter estimation of a Skew-t Mixture of Experts (StMoE) model by the Expectation Conditional Maximization (ECM) algorithm.

Usage

```
emStMoE(X, Y, K, p = 3, q = 1, n_tries = 1, max_iter = 1500,
        threshold = 1e-06, verbose = FALSE, verbose_IRLS = FALSE)
```

Arguments

X	Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
Y	Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
K	The number of experts.
p	Optional. The order of the polynomial regression for the experts.
q	Optional. The order of the logistic regression for the gating network.
n_tries	Optional. Number of runs of the ECM algorithm. The solution providing the highest log-likelihood will be returned.
max_iter	Optional. The maximum number of iterations for the ECM algorithm.
threshold	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the ECM as stopping criteria.
verbose	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during ECM 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 ECM algorithm.

Details

emStMoE function implements the ECM algorithm for the StMoE model. This function starts with an initialization of the parameters done by the method `initParam` of the class [ParamStMoE](#), then it alternates between the E-Step (method of the class [StatStMoE](#)) and the M-Step (method of the class [ParamStMoE](#)) until convergence (until the relative variation of log-likelihood between two steps of the ECM algorithm is less than the threshold parameter).

Value

ECM returns an object of class [ModelStMoE](#).

See Also

[ModelStMoE](#), [ParamStMoE](#), [StatStMoE](#)

Examples

```

data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

stmoe <- emStMoE(X = x, Y = y, K = 2, p = 1, threshold = 1e-4, verbose = TRUE)

stmoe$summary()

stmoe$plot()

```

emTMOE	<i>emTMOE implements the ECM algorithm to fit a t Mixture of Experts (TMOE).</i>
--------	--

Description

emTMOE implements the maximum-likelihood parameter estimation of a Student Mixture of Experts (TMOE) model by the Conditional Expectation Maximization (ECM) algorithm.

Usage

```

emTMOE(X, Y, K, p = 3, q = 1, n_tries = 1, max_iter = 1500,
        threshold = 1e-06, verbose = FALSE, verbose_IRLS = FALSE)

```

Arguments

X	Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
Y	Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
K	The number of experts.
p	Optional. The order of the polynomial regression for the experts.
q	Optional. The order of the logistic regression for the gating network.
n_tries	Optional. Number of runs of the ECM algorithm. The solution providing the highest log-likelihood will be returned.
max_iter	Optional. The maximum number of iterations for the ECM algorithm.
threshold	Optional. A numeric value specifying the threshold for the relative difference of log-likelihood between two steps of the ECM as stopping criteria.
verbose	Optional. A logical value indicating whether or not values of the log-likelihood should be printed during ECM 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 ECM algorithm.

Details

emTMOE function implements the ECM algorithm for the TMOE model. This function starts with an initialization of the parameters done by the method `initParam` of the class [ParamTMOE](#), then it alternates between the E-Step (method of the class [StatTMOE](#)) and the M-Step (method of the class [ParamTMOE](#)) until convergence (until the relative variation of log-likelihood between two steps of the ECM algorithm is less than the threshold parameter).

Value

ECM returns an object of class [ModelTMOE](#).

See Also

[ModelTMOE](#), [ParamTMOE](#), [StatTMOE](#)

Examples

```
data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

tmoe <- emTMOE(X = x, Y = y, K = 2, p = 1, verbose = TRUE)

tmoe$summary()

tmoe$plot()
```

ModelNMoE-class

A Reference Class which represents a fitted NMoE model.

Description

ModelNMoE represents an estimated NMoE model.

Fields

`param` A [ParamNMoE](#) object. It contains the estimated values of the parameters.

`stat` A [StatNMoE](#) object. It contains all the statistics associated to the NMoE model.

Methods

`plot(what = c("meancurve", "confregions", "clusters", "loglikelihood"), ...)` Plot method.

`what` The type of graph requested:

- "meancurve" = Estimated mean and estimated experts means given the input X (fields `Ey` and `Ey_k` of class [StatNMoE](#)).
- "confregions" = Estimated mean and confidence regions. Confidence regions are computed as plus and minus twice the estimated standard deviation (the square root of the field `Vary` of class [StatNMoE](#)).

- "clusters" = Estimated experts means (field Ey_k) and hard partition (field klas of class [StatNMoE](#)).
- "loglikelihood" = Value of the log-likelihood for each iteration (field stored_loglik of class [StatNMoE](#)).

... Other graphics parameters.

By default, all the graphs mentioned above are produced.

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

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

See Also

[ParamNMoE](#), [StatNMoE](#)

Examples

```
data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

nmoe <- emNMoE(X = x, Y = y, K = 2, p = 1, verbose = TRUE)

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

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

# Log-likelihood:
nmoe$stat$loglik

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

ModelSNMoE-class

A Reference Class which represents a fitted SNMoE model.

Description

ModelSNMoE represents an estimated SNMoE model.

Fields

`param` A [ParamSNMoE](#) object. It contains the estimated values of the parameters.

`stat` A [StatSNMoE](#) object. It contains all the statistics associated to the SNMoE model.

Methods

`plot(what = c("meancurve", "confregions", "clusters", "loglikelihood"), ...)` Plot method.

`what` The type of graph requested:

- "meancurve" = Estimated mean and estimated experts means given the input X (fields Ey and Ey_k of class [StatSNMoE](#)).
- "confregions" = Estimated mean and confidence regions. Confidence regions are computed as plus and minus twice the estimated standard deviation (the square root of the field Vary of class [StatSNMoE](#)).
- "clusters" = Estimated experts means (field Ey_k) and hard partition (field klas of class [StatSNMoE](#)).
- "loglikelihood" = Value of the log-likelihood for each iteration (field stored_loglik of class [StatSNMoE](#)).

... Other graphics parameters.

By default, all the graphs mentioned above are produced.

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

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

See Also

[ParamSNMoE](#), [StatSNMoE](#)

Examples

```
data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

snmoe <- emSNMoE(X = x, Y = y, K = 2, p = 1, verbose = TRUE)

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

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

# Log-likelihood:
snmoe$stat$loglik

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

ModelStMoE-class *A Reference Class which represents a fitted StMoE model.*

Description

ModelStMoE represents an estimated StMoE model.

Fields

param A [ParamStMoE](#) object. It contains the estimated values of the parameters.

stat A [StatStMoE](#) object. It contains all the statistics associated to the StMoE model.

Methods

plot(what = c("meancurve", "confregions", "clusters", "loglikelihood"), ...) Plot method.

what The type of graph requested:

- "meancurve" = Estimated mean and estimated experts means given the input X (fields Ey and Ey_k of class [StatStMoE](#)).
- "confregions" = Estimated mean and confidence regions. Confidence regions are computed as plus and minus twice the estimated standard deviation (the square root of the field Vary of class [StatStMoE](#)).
- "clusters" = Estimated experts means (field Ey_k) and hard partition (field klas of class [StatStMoE](#)).
- "loglikelihood" = Value of the log-likelihood for each iteration (field stored_loglik of class [StatStMoE](#)).

... Other graphics parameters.

By default, all the graphs mentioned above are produced.

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

digits The number of significant digits to use when printing.

See Also

[ParamStMoE](#), [StatStMoE](#)

Examples

```
data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

stmoe <- emStMoE(X = x, Y = y, K = 2, p = 1, threshold = 1e-4, verbose = TRUE)

# stmoe is a ModelSTMoE object. It contains some methods such as 'summary' and 'plot'
stmoe$summary()
stmoe$plot()
```

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

# Log-likelihood:
stmoe$stat$loglik

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

ModelTMOE-class	<i>A Reference Class which represents a fitted TMOE model.</i>
-----------------	--

Description

ModelTMOE represents an estimated TMOE model.

Fields

param A [ParamTMOE](#) object. It contains the estimated values of the parameters.

stat A [StatTMOE](#) object. It contains all the statistics associated to the TMOE model.

Methods

plot(what = c("meancurve", "confregions", "clusters", "loglikelihood"), ...) Plot method.

what The type of graph requested:

- "meancurve" = Estimated mean and estimated experts means given the input X (fields Ey and Ey_k of class [StatTMOE](#)).
- "confregions" = Estimated mean and confidence regions. Confidence regions are computed as plus and minus twice the estimated standard deviation (the square root of the field Vary of class [StatTMOE](#)).
- "clusters" = Estimated experts means (field Ey_k) and hard partition (field klas of class [StatTMOE](#)).
- "loglikelihood" = Value of the log-likelihood for each iteration (field stored_loglik of class [StatTMOE](#)).

... Other graphics parameters.

By default, all the graphs mentioned above are produced.

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

digits The number of significant digits to use when printing.

See Also

[ParamTMOE](#), [StatTMOE](#)

Examples

```

data(tempanomalies)
x <- tempanomalies$Year
y <- tempanomalies$AnnualAnomaly

tmoe <- emTMOE(X = x, Y = y, K = 2, p = 1, verbose = TRUE)

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

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

# Log-likelihood:
tmoe$stat$loglik

# Parameters of the polynomial regressions:
tmoe$param$beta

```

ParamNMoE-class

A Reference Class which contains parameters of a NMoE model.

Description

ParamNMoE contains all the parameters of a NMoE model.

Fields

X Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .

Y Numeric vector of length n representing the observed response/output y_1, \dots, y_n .

n Numeric. Length of the response/output vector Y.

K The number of experts.

p The order of the polynomial regression for the experts.

q The order of the logistic regression for the gating network.

alpha Parameters of the gating network. $\alpha = (\alpha_1, \dots, \alpha_{K-1})$ is a matrix of dimension $(q + 1, K - 1)$, with q the order of the logistic regression for the gating network. q is fixed to 1 by default.

beta Polynomial regressions coefficients for each expert. $\beta = (\beta_1, \dots, \beta_K)$ is a matrix of dimension $(p + 1, K)$, with p the order of the polynomial regression. p is fixed to 3 by default.

sigma2 The variances for the K mixture components (matrix of size $(1, K)$).

df The degree of freedom of the NMoE model representing the complexity of the model.

Methods

`initParam(segmental = FALSE)` Method to initialize parameters alpha, beta and sigma2.
 If `segmental = TRUE` then alpha, beta and sigma2 are initialized by clustering the response Y uniformly into K contiguous segments. Otherwise, alpha, beta and sigma2 are initialized by clustering randomly the response Y into K segments.

ParamSNMoE-class *A Reference Class which contains parameters of a SNMoE model.*

Description

ParamSNMoE contains all the parameters of a SNMoE model.

Fields

`X` Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
`Y` Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
`n` Numeric. Length of the response/output vector Y .
`K` The number of experts.
`p` The order of the polynomial regression for the experts.
`q` The order of the logistic regression for the gating network.
`alpha` Parameters of the gating network. $\alpha = (\alpha_1, \dots, \alpha_{K-1})$ is a matrix of dimension $(q + 1, K - 1)$, with q the order of the logistic regression for the gating network. q is fixed to 1 by default.
`beta` Polynomial regressions coefficients for each expert. $\beta = (\beta_1, \dots, \beta_K)$ is a matrix of dimension $(p + 1, K)$, with p the order of the polynomial regression. p is fixed to 3 by default.
`sigma2` The variances for the K mixture components (matrix of size $(1, K)$).
`lambda` The skewness parameters for each experts (matrix of size $(1, K)$).
`delta` delta is equal to $\delta = \frac{\lambda}{\sqrt{1+\lambda^2}}$.
`df` The degree of freedom of the SNMoE model representing the complexity of the model.

Methods

`initParam(segmental = FALSE)` Method to initialize parameters alpha, beta and sigma2.
 If `segmental = TRUE` then alpha, beta and sigma2 are initialized by clustering the response Y uniformly into K contiguous segments. Otherwise, alpha, beta and sigma2 are initialized by clustering randomly the response Y into K segments.

`MStep(statSNMoE, verbose_IRLS)` Method which implements the M-step of the EM algorithm to learn the parameters of the SNMoE model based on statistics provided by the object `statSNMoE` of class [StatSNMoE](#) (which contains the E-step).

ParamStMoE-class *A Reference Class which contains parameters of a StMoE model.*

Description

ParamStMoE contains all the parameters of a StMoE model.

Fields

- X Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
- Y Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
- n Numeric. Length of the response/output vector Y.
- K The number of experts.
- p The order of the polynomial regression for the experts.
- q The order of the logistic regression for the gating network.
- alpha Parameters of the gating network. $\alpha = (\alpha_1, \dots, \alpha_{K-1})$ is a matrix of dimension $(q + 1, K - 1)$, with q the order of the logistic regression for the gating network. q is fixed to 1 by default.
- beta Polynomial regressions coefficients for each expert. $\beta = (\beta_1, \dots, \beta_K)$ is a matrix of dimension $(p + 1, K)$, with p the order of the polynomial regression. p is fixed to 3 by default.
- sigma2 The variances for the K mixture components (matrix of size $(1, K)$).
- lambda The skewness parameters for each experts (matrix of size $(1, K)$).
- delta delta is equal to $\delta = \frac{\lambda}{\sqrt{1+\lambda^2}}$.
- nu The degree of freedom for the Student distribution for each experts (matrix of size $(1, K)$).
- df The degree of freedom of the StMoE model representing the complexity of the model.

Methods

- initParam(segmental = FALSE) Method to initialize parameters alpha, beta and sigma2.
If segmental = TRUE then alpha, beta and sigma2 are initialized by clustering the response Y uniformly into K contiguous segments. Otherwise, alpha, beta and sigma2 are initialized by clustering randomly the response Y into K segments.
- MStep(statStMoE, calcAlpha = FALSE, calcBeta = FALSE, calcSigma2 = FALSE, calcLambda = FALSE, calcNu = FALSE) Method which implements the M-step of the EM algorithm to learn the parameters of the StMoE model based on statistics provided by the object statStMoE of class [StatStMoE](#) (which contains the E-step).

ParamTMoE-class	<i>A Reference Class which contains parameters of a TMoE model.</i>
-----------------	---

Description

ParamTMoE contains all the parameters of a TMoE model.

Fields

- X Numeric vector of length n representing the covariates/inputs x_1, \dots, x_n .
- Y Numeric vector of length n representing the observed response/output y_1, \dots, y_n .
- n Numeric. Length of the response/output vector Y .
- K The number of experts.
- p The order of the polynomial regression for the experts.
- q The order of the logistic regression for the gating network.
- alpha Parameters of the gating network. $\alpha = (\alpha_1, \dots, \alpha_{K-1})$ is a matrix of dimension $(q + 1, K - 1)$, with q the order of the logistic regression for the gating network. q is fixed to 1 by default.
- beta Polynomial regressions coefficients for each expert. $\beta = (\beta_1, \dots, \beta_K)$ is a matrix of dimension $(p + 1, K)$, with p the order of the polynomial regression. p is fixed to 3 by default.
- sigma2 The variances for the K mixture components (matrix of size $(1, K)$).
- nu The degree of freedom for the Student distribution for each experts (matrix of size $(1, K)$).
- df The degree of freedom of the TMoE model representing the complexity of the model.

Methods

- initParam(segmental = FALSE) Method to initialize parameters alpha, beta and sigma2.
If segmental = TRUE then alpha, beta and sigma2 are initialized by clustering the response Y uniformly into K contiguous segments. Otherwise, alpha, beta and sigma2 are initialized by clustering randomly the response Y into K segments.
- MStep(statTMoE, verbose_IRLS) Method which implements the M-step of the EM algorithm to learn the parameters of the TMoE model based on statistics provided by the object statTMoE of class [StatTMoE](#) (which contains the E-step).

 sampleUnivNMoE

Draw a sample from a normal mixture of linear experts model.

Description

Draw a sample from a normal mixture of linear experts model.

Usage

```
sampleUnivNMoE(alphak, betak, sigmak, x)
```

Arguments

alphak	The parameters of the gating network. alphak is a matrix of size $(q + 1, K - 1)$, with $K - 1$, the number of regressors (experts) and q the order of the logistic regression
betak	Matrix of size $(p + 1, K)$ representing the regression coefficients of the experts network.
sigmak	Vector of length K giving the standard deviations of the experts network.
x	A vector of length n representing the inputs (predictors).

Value

A list with the output variable y and statistics.

- y Vector of length n giving the output variable.
- z_i A vector of size n giving the hidden label of the expert component generating the i -th observation. Its elements are $z_i[i] = k$, if the i -th observation has been generated by the k -th expert.
- z A matrix of size (n, K) giving the values of the binary latent component indicators Z_{ik} such that $Z_{ik} = 1$ iff $Z_i = k$.
- stats A list whose elements are:
 - Ey_k Matrix of size (n, K) giving the conditional expectation of Y_i the output variable given the value of the hidden label of the expert component generating the i th observation $z_i = k$, and the value of predictor $X = x_i$.
 - Ey Vector of length n giving the conditional expectation of Y_i given the value of predictor $X = x_i$.
 - $Vary_k$ Vector of length k representing the conditional variance of Y_i given $z_i = k$, and $X = x_i$.
 - $Vary$ Vector of length n giving the conditional expectation of Y_i given $X = x_i$.

Examples

```

n <- 500 # Size of the sample
alphak <- matrix(c(0, 8), ncol = 1) # Parameters of the gating network
betak <- matrix(c(0, -2.5, 0, 2.5), ncol = 2) # Regression coefficients of the experts
sigmak <- c(1, 1) # Standard deviations of the experts
x <- seq.int(from = -1, to = 1, length.out = n) # Inputs (predictors)

# Generate sample of size n
sample <- sampleUnivNMoe(alphak = alphak, betak = betak, sigmak = sigmak, x = x)

# Plot points and estimated means
plot(x, sample$y, pch = 4)
lines(x, sample$stats$Ey_k[, 1], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey_k[, 2], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey, col = "red", lwd = 1.5)

```

sampleUnivSNMoE

Draw a sample from a skew-normal mixture of linear experts model.

Description

Draw a sample from a skew-normal mixture of linear experts model.

Usage

```
sampleUnivSNMoE(alphak, betak, sigmak, lambdak, x)
```

Arguments

alphak	The parameters of the gating network. alphak is a matrix of size $(q + 1, K - 1)$, with $K - 1$, the number of regressors (experts) and q the order of the logistic regression
betak	Matrix of size $(p + 1, K)$ representing the regression coefficients of the experts network.
sigmak	Vector of length K giving the standard deviations of the experts network.
lambdak	Vector of length K giving the skewness parameter of each experts.
x	A vector of length n representing the inputs (predictors).

Value

A list with the output variable y and statistics.

- y Vector of length n giving the output variable.
- z_i A vector of size n giving the hidden label of the expert component generating the i -th observation. Its elements are $z_i[i] = k$, if the i -th observation has been generated by the k -th expert.

- z A matrix of size (n, K) giving the values of the binary latent component indicators Z_{ik} such that $Z_{ik} = 1$ iff $Z_i = k$.
- stats A list whose elements are:
 - Ey_k Matrix of size (n, K) giving the conditional expectation of Y_i the output variable given the value of the hidden label of the expert component generating the i th observation $z_i = k$, and the value of predictor $X = x_i$.
 - Ey Vector of length n giving the conditional expectation of Y_i given the value of predictor $X = x_i$.
 - Vary_k Vector of length k representing the conditional variance of Y_i given $z_i = k$, and $X = x_i$.
 - Vary Vector of length n giving the conditional expectation of Y_i given $X = x_i$.

Examples

```
n <- 500 # Size of the sample
alphak <- matrix(c(0, 8), ncol = 1) # Parameters of the gating network
betak <- matrix(c(0, -2.5, 0, 2.5), ncol = 2) # Regression coefficients of the experts
lambdak <- c(3, 5) # Skewness parameters of the experts
sigmak <- c(1, 1) # Standard deviations of the experts
x <- seq.int(from = -1, to = 1, length.out = n) # Inputs (predictors)

# Generate sample of size n
sample <- sampleUnivSNMoE(alphak = alphak, betak = betak, sigmak = sigmak,
                          lambdak = lambdak, x = x)

# Plot points and estimated means
plot(x, sample$y, pch = 4)
lines(x, sample$stats$Ey_k[, 1], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey_k[, 2], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey, col = "red", lwd = 1.5)
```

sampleUnivStMoE

Draw a sample from a univariate skew-t mixture.

Description

Draw a sample from a univariate skew-t mixture.

Usage

```
sampleUnivStMoE(alphak, betak, sigmak, lambdak, nuk, x)
```

Arguments

alphak The parameters of the gating network. alphak is a matrix of size $(q + 1, K - 1)$, with $K - 1$, the number of regressors (experts) and q the order of the logistic regression

betak	Matrix of size $(p + 1, K)$ representing the regression coefficients of the experts network.
sigmak	Vector of length K giving the standard deviations of the experts network.
lambdak	Vector of length K giving the skewness parameter of each experts.
nuk	Vector of length K giving the degrees of freedom of the experts network t densities.
x	A vector of length n representing the inputs (predictors).

Value

A list with the output variable y and statistics.

- y Vector of length n giving the output variable.
- z_i A vector of size n giving the hidden label of the expert component generating the i -th observation. Its elements are $z_i[i] = k$, if the i -th observation has been generated by the k -th expert.
- z A matrix of size (n, K) giving the values of the binary latent component indicators Z_{ik} such that $Z_{ik} = 1$ iff $Z_i = k$.
- stats A list whose elements are:
 - Ey_k Matrix of size (n, K) giving the conditional expectation of Y_i the output variable given the value of the hidden label of the expert component generating the i th observation $z_i = k$, and the value of predictor $X = x_i$.
 - Ey Vector of length n giving the conditional expectation of Y_i given the value of predictor $X = x_i$.
 - $Vary_k$ Vector of length k representing the conditional variance of Y_i given $z_i = k$, and $X = x_i$.
 - $Vary$ Vector of length n giving the conditional expectation of Y_i given $X = x_i$.

Examples

```
n <- 500 # Size of the sample
alphak <- matrix(c(0, 8), ncol = 1) # Parameters of the gating network
betak <- matrix(c(0, -2.5, 0, 2.5), ncol = 2) # Regression coefficients of the experts
sigmak <- c(0.5, 0.5) # Standard deviations of the experts
lambdak <- c(3, 5) # Skewness parameters of the experts
nuk <- c(5, 7) # Degrees of freedom of the experts network t densities
x <- seq.int(from = -1, to = 1, length.out = n) # Inputs (predictors)

# Generate sample of size n
sample <- sampleUnivStMoE(alphak = alphak, betak = betak, sigmak = sigmak,
                          lambdak = lambdak, nuk = nuk, x = x)

# Plot points and estimated means
plot(x, sample$y, pch = 4)
lines(x, sample$stats$Ey_k[, 1], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey_k[, 2], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey, col = "red", lwd = 1.5)
```

 sampleUnivTMOE

Draw a sample from a univariate t mixture of experts (TMOE).

Description

Draw a sample from a univariate t mixture of experts (TMOE).

Usage

```
sampleUnivTMOE(alphak, betak, sigmak, nuk, x)
```

Arguments

alphak	The parameters of the gating network. alphak is a matrix of size $(q + I, K - I)$, with $K - I$, the number of regressors (experts) and q the order of the logistic regression
betak	Matrix of size $(p + I, K)$ representing the regression coefficients of the experts network.
sigmak	Vector of length K giving the standard deviations of the experts network.
nuk	Vector of length K giving the degrees of freedom of the experts network t densities.
x	A vector of length n representing the inputs (predictors).

Value

A list with the output variable y and statistics.

- y Vector of length n giving the output variable.
- z_i A vector of size n giving the hidden label of the expert component generating the i -th observation. Its elements are $z_i[i] = k$, if the i -th observation has been generated by the k -th expert.
- z A matrix of size (n, K) giving the values of the binary latent component indicators Z_{ik} such that $Z_{ik} = 1$ iff $Z_i = k$.
- stats A list whose elements are:
 - Ey_k Matrix of size (n, K) giving the conditional expectation of Y_i the output variable given the value of the hidden label of the expert component generating the i th observation $z_i = k$, and the value of predictor $X = x_i$.
 - Ey Vector of length n giving the conditional expectation of Y_i given the value of predictor $X = x_i$.
 - $Vary_k$ Vector of length k representing the conditional variance of Y_i given $z_i = k$, and $X = x_i$.
 - $Vary$ Vector of length n giving the conditional expectation of Y_i given $X = x_i$.

Examples

```

n <- 500 # Size of the sample
alphak <- matrix(c(0, 8), ncol = 1) # Parameters of the gating network
betak <- matrix(c(0, -2.5, 0, 2.5), ncol = 2) # Regression coefficients of the experts
sigmak <- c(0.5, 0.5) # Standard deviations of the experts
nuk <- c(5, 7) # Degrees of freedom of the experts network t densities
x <- seq.int(from = -1, to = 1, length.out = n) # Inputs (predictors)

# Generate sample of size n
sample <- sampleUnivTMOE(alphak = alphak, betak = betak, sigmak = sigmak,
                        nuk = nuk, x = x)

# Plot points and estimated means
plot(x, sample$y, pch = 4)
lines(x, sample$stats$Ey_k[, 1], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey_k[, 2], col = "blue", lty = "dotted", lwd = 1.5)
lines(x, sample$stats$Ey, col = "red", lwd = 1.5)

```

StatNMoE-class

A Reference Class which contains statistics of a NMoE model.

Description

StatNMoE contains all the statistics associated to a [NMoE](#) model. It mainly includes the E-Step of the EM algorithm calculating the posterior distribution of the hidden variables, as well as the calculation of the log-likelihood.

Fields

piik Matrix of size (n, K) representing the probabilities $\pi_k(x_i; \Psi) = P(z_i = k | x; \Psi)$ of the latent variable $z_i, i = 1, \dots, n$.

z_ik Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z_{ik} = 1$ if $z_{ik} = \arg \max_s \tau_{is}$; 0 otherwise, $k = 1, \dots, K$.

klas Column matrix of the labels issued from **z_ik**. Its elements are $klas(i) = k, k = 1, \dots, K$.

tik Matrix of size (n, K) giving the posterior probability τ_{ik} that the observation y_i originates from the k -th expert.

Ey_k Matrix of dimension (n, K) giving the estimated means of the experts.

Ey Column matrix of dimension n giving the estimated mean of the NMoE.

Var_yk Column matrix of dimension K giving the estimated means of the experts.

Vary Column matrix of dimension n giving the estimated variance of the response.

loglik Numeric. Observed-data log-likelihood of the NMoE model.

com_loglik Numeric. Complete-data log-likelihood of the NMoE model.

stored_loglik Numeric vector. Stored values of the 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_piik_fik Matrix of size (n, K) giving the values of the logarithm of the joint probability $P(y_i, z_i = k | \mathbf{x}, \Psi)$, $i = 1, \dots, n$.

log_sum_piik_fik Column matrix of size m giving the values of $\log \sum_{k=1}^K P(y_i, z_i = k | \mathbf{x}, \Psi)$, $i = 1, \dots, n$.

Methods

computeLikelihood(reg_irls) Method to compute the log-likelihood. reg_irls is the value of the regularization part in the IRLS algorithm.

computeStats(paramNMoe) Method used in the EM algorithm to compute statistics based on parameters provided by the object paramNMoe of class [ParamNMoe](#).

EStep(paramNMoe) Method used in the EM algorithm to update statistics based on parameters provided by the object paramNMoe of class [ParamNMoe](#) (prior and posterior probabilities).

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

$$z_{ik} = 1 \text{ if } k = \arg \max_s \tau_{is}; 0 \text{ otherwise}$$

See Also

[ParamNMoe](#)

StatSNMoE-class *A Reference Class which contains statistics of a SNMoE model.*

Description

StatSNMoE contains all the statistics associated to a [SNMoE](#) model. It mainly includes the E-Step of the ECM algorithm calculating the posterior distribution of the hidden variables, as well as the calculation of the log-likelihood.

Fields

piik Matrix of size (n, K) representing the probabilities $\pi_k(x_i; \Psi) = P(z_i = k | \mathbf{x}; \Psi)$ of the latent variable z_i , $i = 1, \dots, n$.

z_ik Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z_{ik} = 1$ if $z_{ik} = \arg \max_s \tau_{is}$; 0 otherwise, $k = 1, \dots, K$.

klas Column matrix of the labels issued from z_ik. Its elements are $klas(i) = k$, $k = 1, \dots, K$.

tik Matrix of size (n, K) giving the posterior probability τ_{ik} that the observation y_i originates from the k -th expert.

Ey_k Matrix of dimension (n, K) giving the estimated means of the experts.

Ey Column matrix of dimension n giving the estimated mean of the SNMoE.

Var_yk Column matrix of dimension K giving the estimated means of the experts.
 Vary Column matrix of dimension n giving the estimated variance of the response.
 loglik Numeric. Observed-data log-likelihood of the SNMoE model.
 com_loglik Numeric. Complete-data log-likelihood of the SNMoE model.
 stored_loglik Numeric vector. Stored values of the log-likelihood at each ECM 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_piik_fik Matrix of size (n, K) giving the values of the logarithm of the joint probability $P(y_i, z_i = k | \mathbf{x}, \Psi)$, $i = 1, \dots, n$.
 log_sum_piik_fik Column matrix of size m giving the values of $\log \sum_{k=1}^K P(y_i, z_i = k | \mathbf{x}, \Psi)$, $i = 1, \dots, n$.
 E1ik Conditional expectations of U_i (Matrix of size (n, K)).
 E2ik Conditional expectations of U_i^2 (Matrix of size (n, K)).

Methods

computeLikelihood(reg_irls) Method to compute the log-likelihood. reg_irls is the value of the regularization part in the IRLS algorithm.
 computeStats(paramSNMoE) Method used in the ECM algorithm to compute statistics based on parameters provided by the object paramSNMoE of class [ParamSNMoE](#).
 EStep(paramSNMoE) Method used in the ECM algorithm to update statistics based on parameters provided by the object paramSNMoE of class [ParamSNMoE](#) (prior and posterior probabilities).
 MAP() MAP calculates values of the fields z_ik and klas by applying the Maximum A Posteriori Bayes allocation rule.

$$z_{ik} = 1 \text{ if } k = \arg \max_s \tau_{is}; 0 \text{ otherwise}$$

See Also

[ParamSNMoE](#)

StatStMoE-class

A Reference Class which contains statistics of a StMoE model.

Description

StatStMoE contains all the statistics associated to a [StMoE](#) model. It mainly includes the E-Step of the ECM algorithm calculating the posterior distribution of the hidden variables, as well as the calculation of the log-likelihood.

Fields

- piik** Matrix of size (n, K) representing the probabilities $\pi_k(x_i; \Psi) = P(z_i = k | \mathbf{x}; \Psi)$ of the latent variable $z_i, i = 1, \dots, n$.
- z_ik** Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z_{ik} = 1$ if $z_{ik} = \arg \max_s \tau_{is}$; 0 otherwise, $k = 1, \dots, K$.
- klas** Column matrix of the labels issued from **z_ik**. Its elements are $klas(i) = k, k = 1, \dots, K$.
- tik** Matrix of size (n, K) giving the posterior probability τ_{ik} that the observation y_i originates from the k -th expert.
- Ey_k** Matrix of dimension (n, K) giving the estimated means of the experts.
- Ey** Column matrix of dimension n giving the estimated mean of the StMoE.
- Var_yk** Column matrix of dimension K giving the estimated means of the experts.
- Vary** Column matrix of dimension n giving the estimated variance of the response.
- loglik** Numeric. Observed-data log-likelihood of the StMoE model.
- com_loglik** Numeric. Complete-data log-likelihood of the StMoE model.
- stored_loglik** Numeric vector. Stored values of the log-likelihood at each ECM 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_piik_fik** Matrix of size (n, K) giving the values of the logarithm of the joint probability $P(y_i, z_i = k | \mathbf{x}, \Psi), i = 1, \dots, n$.
- log_sum_piik_fik** Column matrix of size m giving the values of $\log \sum_{k=1}^K P(y_i, z_i = k | \mathbf{x}, \Psi), i = 1, \dots, n$.
- dik** It represents the value of d_{ik} .
- wik** Conditional expectations w_{ik} .
- E1ik** Conditional expectations $e_{1,ik}$.
- E2ik** Conditional expectations $e_{2,ik}$.
- E3ik** Conditional expectations $e_{3,ik}$.
- stme_pdf** Skew-t mixture of experts density.

Methods

- computeLikelihood(reg_irls)** Method to compute the log-likelihood. **reg_irls** is the value of the regularization part in the IRLS algorithm.
- computeStats(paramStMoE)** Method used in the ECM algorithm to compute statistics based on parameters provided by the object **paramStMoE** of class **ParamStMoE**.
- EStep(paramStMoE, calcTau = FALSE, calcE1 = FALSE, calcE2 = FALSE, calcE3 = FALSE)** Method used in the ECM algorithm to update statistics based on parameters provided by the object **paramStMoE** of class **ParamStMoE** (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 $k = \arg \max_s \tau_{is}$; 0 otherwise

See Also[ParamStMoE](#)

StatTMOE-class

*A Reference Class which contains statistics of a TMOE model.***Description**

StatTMOE contains all the statistics associated to a [TMOE](#) model. It mainly includes the E-Step of the ECM algorithm calculating the posterior distribution of the hidden variables, as well as the calculation of the log-likelihood.

Fields

piik Matrix of size (n, K) representing the probabilities $\pi_k(x_i; \Psi) = P(z_i = k | \mathbf{x}; \Psi)$ of the latent variable $z_i, i = 1, \dots, n$.

z_ik Hard segmentation logical matrix of dimension (n, K) obtained by the Maximum a posteriori (MAP) rule: $z_{ik} = 1$ if $z_{ik} = \arg \max_s \tau_{is}$; 0 otherwise, $k = 1, \dots, K$.

klas Column matrix of the labels issued from **z_ik**. Its elements are $klas(i) = k, k = 1, \dots, K$.

tik Matrix of size (n, K) giving the posterior probability τ_{ik} that the observation y_i originates from the k -th expert.

Ey_k Matrix of dimension (n, K) giving the estimated means of the experts.

Ey Column matrix of dimension n giving the estimated mean of the TMOE.

Var_yk Column matrix of dimension K giving the estimated means of the experts.

Vary Column matrix of dimension n giving the estimated variance of the response.

loglik Numeric. Observed-data log-likelihood of the TMOE model.

com_loglik Numeric. Complete-data log-likelihood of the TMOE model.

stored_loglik Numeric vector. Stored values of the log-likelihood at each ECM 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_piik_fik Matrix of size (n, K) giving the values of the logarithm of the joint probability $P(y_i, z_i = k | \mathbf{x}, \Psi), i = 1, \dots, n$.

log_sum_piik_fik Column matrix of size m giving the values of $\log \sum_{k=1}^K P(y_i, z_i = k | \mathbf{x}, \Psi), i = 1, \dots, n$.

Wik Conditional expectations w_{ik} .

Methods

`computeLikelihood(reg_irls)` Method to compute the log-likelihood. `reg_irls` is the value of the regularization part in the IRLS algorithm.

`computeStats(paramTMoE)` Method used in the ECM algorithm to compute statistics based on parameters provided by the object `paramTMoE` of class `ParamTMoE`.

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

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

$$z_{ik} = 1 \text{ if } k = \arg \max_s \tau_{is}; 0 \text{ otherwise}$$

See Also

[ParamTMoE](#)

tempanomalies	<i>Global Annual Temperature Anomalies (Land Meteorological Stations) (1880-2015)</i>
---------------	---

Description

This dataset is from https://cdiac.ess-dive.lbl.gov/ftp/trends/temp/hansen/gl_land.txt.

Usage

```
tempanomalies
```

Format

A data frame with 136 rows and 3 columns:

Year Year of observation.

AnnualAnomaly Value in degrees C of the global annual temperature anomaly.

5-YearMean 5-Year mean of temperature anomalies.

Details

Global annual temperature anomalies (degrees C) computed using data from land meteorological stations, 1880-2015. Anomalies are relative to the 1951-1980 base period means.

Non-computed values are indicated by "-99.99".

Index

* datasets

- tempanomalies, [28](#)

- emNMoE, [4](#)
- emSNMoE, [5](#)
- emStMoE, [7](#)
- emTMoE, [8](#)

- meteorits (meteorits-package), [3](#)
- meteorits-package, [3](#)
- ModelNMoE, [5](#)
- ModelNMoE (ModelNMoE-class), [9](#)
- ModelNMoE-class, [9](#)
- ModelSNMoE, [6](#)
- ModelSNMoE (ModelSNMoE-class), [10](#)
- ModelSNMoE-class, [10](#)
- ModelStMoE, [7](#)
- ModelStMoE (ModelStMoE-class), [12](#)
- ModelStMoE-class, [12](#)
- ModelTMoE, [9](#)
- ModelTMoE (ModelTMoE-class), [13](#)
- ModelTMoE-class, [13](#)

- NMoE, [23](#)

- ParamNMoE, [5](#), [9](#), [10](#), [24](#)
- ParamNMoE (ParamNMoE-class), [14](#)
- ParamNMoE-class, [14](#)
- ParamSNMoE, [6](#), [10](#), [11](#), [25](#)
- ParamSNMoE (ParamSNMoE-class), [15](#)
- ParamSNMoE-class, [15](#)
- ParamStMoE, [7](#), [12](#), [26](#), [27](#)
- ParamStMoE (ParamStMoE-class), [16](#)
- ParamStMoE-class, [16](#)
- ParamTMoE, [9](#), [13](#), [28](#)
- ParamTMoE (ParamTMoE-class), [17](#)
- ParamTMoE-class, [17](#)

- sampleUnivNMoE, [18](#)
- sampleUnivSNMoE, [19](#)
- sampleUnivStMoE, [20](#)

- sampleUnivTMoE, [22](#)
- SNMoE, [24](#)
- StatNMoE, [5](#), [9](#), [10](#)
- StatNMoE (StatNMoE-class), [23](#)
- StatNMoE-class, [23](#)
- StatSNMoE, [6](#), [10](#), [11](#), [15](#)
- StatSNMoE (StatSNMoE-class), [24](#)
- StatSNMoE-class, [24](#)
- StatStMoE, [7](#), [12](#), [16](#)
- StatStMoE (StatStMoE-class), [25](#)
- StatStMoE-class, [25](#)
- StatTMoE, [9](#), [13](#), [17](#)
- StatTMoE (StatTMoE-class), [27](#)
- StatTMoE-class, [27](#)
- StMoE, [25](#)

- tempanomalies, [28](#)
- TMoE, [27](#)