Package ‘hdMTD’

April 24, 2025

Type Package

Title Inference for High-Dimensional Mixture Transition Distribution
Models

Version 0.1.0

Description Estimates parameters in Mixture Transition Distribution (MTD) models, a class of high-order Markov chains. The set of relevant pasts (lags) is selected using either the Bayesian Information Criterion or the Forward Stepwise and Cut algorithms. Other model parameters (e.g. transition probabilities and oscillations) can be estimated via maximum likelihood estimation or the Expectation-Maximization algorithm. Additionally, 'hdMTD' includes a perfect sampling algorithm that generates samples of an MTD model from its invariant distribution. For theory, see Ost & Takahashi (2023) <http://jmlr.org/papers/v24/22-0266.html>.

URL https://github.com/MaiaraGripp/hdMTD

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports methods, dplyr, purrr

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Maiara Gripp [aut, cre],
Guilherme Ost [ths],
Giulio Iacobelli [ths]

Maintainer Maiara Gripp <maiara@dme.ufrj.br>

Repository CRAN

Date/Publication 2025-04-24 07:20:02 UTC

Contents

`checkSample` *Checks a sample*

Description

Checks if a sample is a suitable argument for some functions within the package.

Usage

```
checkSample(X)
```

Arguments

`X` A vector, a single-column data frame, a list, or a matrix with a single row or a single column. Must be composed by nonnegative integers.

Value

Returns the sample as a vector or identifies any possible sample problems.

`countsTab` *Counts sequences of length d+1 in a sample*

Description

Creates a tibble containing all unique sequences of length `d+1` found in the sample, along with their absolute frequencies.

Usage

```
countsTab(X, d)
```

Arguments

`X`	A numeric vector, a single-column data frame, or a list with a sample from a Markov chain. The first element must be the most recent observation.
`d`	A positive integer specifying the number of elements in each sequence, which will be `d+1`. Typically, `d` represents the chain order or serves as an upper limit for it.

Details

The function generates a tibble with `d+2` columns. In the first `d+1` columns, each row displays a unique sequence of size `d+1` observed in the sample. The last column, called `Nxa`, contains the number of times each of these sequences appeared in the sample.

The number of rows in the output varies between 1 and $|A|^{d+1}$, where $|A|$ is the number of unique states in `X`, since it depends on the number of unique sequences that appear in the sample.

Value

A tibble with all observed sequences of length `d+1` and their absolute frequencies.

Examples

```
countsTab(c(1,2,2,1,2,1,1,2,1,2), 3)

# Using test data.
countsTab(testChains[, 1], 2)
```

dTV_sample *The total variation distance between distributions*

Description

Calculates the total variation distance between distributions conditioned in a given past sequence.

Usage

```
dTV_sample(S, j, A = NULL, base, lenA = NULL, A_pairs = NULL, x_S)
```

Arguments

S	A numeric vector of positive integers (or NULL) representing a set of past lags. The distributions from which this function will calculate the total variation distance are conditioned on a fixed sequence indexed by S (the user must also input the sequence through the argument x_S).
j	A positive integer representing a lag in the *complement* of S. The symbols indexed by j vary along the state space A, altering the distribution through this single lag, and the size of this change is what this function seeks to measure.
A	A vector of unique positive integers (state space) with at least two elements. A represents the state space. You may leave A=NULL (default) if you provide the function with the arguments lenA and A_pairs (see *Details* below).
base	A data frame with sequences of elements from A and their transition probabilities. base is meant to be an output from function freqTab(), and must be structured as such. The data frame must contain all required transitions conditioned on x_S (i.e. length(A)^2 rows with sequence x_S). See *Details* section for further information.
lenA	An integer >= 2, representing length(A). Required if A is not provided.
A_pairs	A two-column matrix with all unique pairs of elements from A. Required if A is not provided.
x_S	A vector of length length(S) or NULL. If S==NULL, x_S will be set to NULL. x_S represents a sequence of symbols from A indexed by S. This sequence remains constant across the conditional distributions to be compared, representing the fixed configuration of the past.

Details

This function computes the total variation distance between distributions found in base, which is expected to be the output of the function freqTab(). Therefore, base must follow a specific structure (e.g., column names must match, and a column named qax_Sj, containing transition distributions, must be present). For more details on the output structure of freqTab(), refer to its documentation..

If you provide the state space A, the function calculates: lenA <- length(A) and A_pairs <- t(utils::combn(A, 2)). Alternatively, you can input lenA and A_pairs directly and let A <- NULL, which is useful in loops to improve efficiency.

Value

Returns a vector of total variation distances, where each entry corresponds to the distance between a pair of distributions conditioned on the same fixed past `x_S`, differing only in the symbol indexed by `j`, which varies across all distinct pairs of elements in `A`.The output has length equal to the number of unique pairs in `A_pairs`.

Examples

```
#creating base argument through freqTab function.
pbase <- freqTab(S=c(1,4),j=2,A=c(1,2,3),countsTab = countsTab(testChains[,2],d=5))
dTV_sample(S=c(1,2),j=4,A=c(1,2,3),base=pbase,x_S=c(2,3))
pbase <- freqTab(S=NULL,j=1,A=c(1,2,3),countsTab = countsTab(testChains[,2],d=5))
dTV_sample(S=NULL,j=1,A=c(1,2,3),base=pbase)
```

`freqTab` *A tibble containing sample sequence frequencies and estimated probabilities*

Description

This function returns a tibble containing the sample sequences, their frequencies and the estimated transition probabilities.

Usage

```
freqTab(S, j = NULL, A, countsTab, complete = TRUE)
```

Arguments

`S` A numeric vector of positive integers or `NULL`. Represents a set of past lags that must be present within the columns of the `countsTab` argument and are to be considered while estimating the transition probabilities. Both `S` and `j` cannot be `NULL` at the same time.

`j` An integer or `NULL`. Typically represents a lag `j` in the *complement* of `S`. Both `S` and `j` cannot be `NULL` at the same time. Both `S` and `j` cannot be `NULL` at the same time. See *Details* for further information.

`A` A vector with nonnegative integers. Must have at least two different entries. `A` represents the state space.

`countsTab` A tibble or a data frame with all sequences of length d+1 that appear in the sample, and their absolute frequency. This tibble is typically generated by the function `countsTab()`. If using a custom data frame not generated by `countsTab()`, make sure its format and column names match the expected structure; otherwise, errors may occur in `freqTab()`.

`complete` Logical. If `TRUE` all sequences that did not appear in the sample will be included in the output with frequency equal to 0 .

Details

The parameters S and j determine which columns of countsTab are retained in the output. Specifying a lag j is optional. All lags can be specified via S, while leaving j = NULL (default). The output remains the same as when specifying S and j separately. The inclusion of j as a parameter improves clarity within the package's algorithms. Note that j cannot be an element of S.

Value

A tibble where each row represents a sequence of elements from A. The initial columns display each sequence symbol separated into columns corresponding to their time indexes. The remaining columns show the sample frequencies of the sequences and the MLE (Maximum Likelihood Estimator) of the transition probabilities.

Examples

```
freqTab(S=c(1,4),j=2,A=c(1,2,3),countsTab = countsTab(testChains[,2],d=5))
#Equivalent to freqTab(S=c(1,2,4),j=NULL,A=c(1,2,3),countsTab = countsTab(testChains[,2],d=5))
```

---

hdMTD *Inference in MTD models*

Description

This function estimates the relevant lag set in a Mixture Transition Distribution (MTD) model using one of the available methods. By default, it applies the Forward Stepwise ("FS") method, which is particularly useful in high-dimensional settings. The available methods are:

- "FS" (Forward Stepwise): selects the lags by a criteria that depends on their oscillations.
- "CUT": a method that selects the relevant lag set based on a predefined threshold.
- "FSC" (Forward Stepwise and Cut): applies the "FS" method followed by the "CUT" method.
- "BIC": selects the lag set using the Bayesian Information Criterion.

Usage

```
hdMTD(X, d, method = "FS", ...)
```

Arguments

X	A vector or single-column data frame containing a chain sample.
d	A positive integer representing an upper bound for the chain order.
method	A character string indicating the method for estimating the relevant lag set. The available methods are: "FS" (default), "FSC", "CUT", and "BIC". See the *Details* section and the documentation of the corresponding method functions for more information.
...	Additional arguments for the selected method. If not specified, default values will be used (see *Details*).

Details

The function dynamically calls the corresponding method function (e.g., `hdMTD_FSC()` for `method = "FSC"`). Additional parameters specific to each method can be provided via `...`, and default values are used for unspecified parameters.

#' This function serves as a wrapper for the method-specific functions:

- `hdMTD_FS()`, for `method` = `"FS"`
- `hdMTD_FSC()`, for `method` = `"FSC"`
- `hdMTD_CUT()`, for `method` = `"CUT"`
- `hdMTD_BIC()`, for `method` = `"BIC"`

Any additional parameters (`...`) must match those accepted by the corresponding method function. If a parameter value is not explicitly provided, a default value is used. The main default parameters are:

- `S` = `seq_len(d)`: Used in "BIC" or "CUT" methods.
- `l` = d. Required in "FS" or "FSC" methods.
- `alpha` = `0.05`, `mu` = `1`. Used in "CUT" or "FSC" methods.
- `xi` = `0.5`. Used in "CUT", "FSC" or "BIC" methods.
- `minl` = `1`, `maxl` = `length(S)`, `byl` = `FALSE`. Used in "BIC" method. All default values are specified in the documentation of the method-specific functions.

Value

A vector containing the estimated relevant lag set.

Examples

```
X <- testChains[,1]
hdMTD(X = X, d = 5, method = "FS", l = 2)
hdMTD(X = X, d = 5, method = "BIC", xi = 1, minl = 3, maxl = 3)
```

`hdMTD_BIC` *The Bayesian Information Criterion (BIC) method for inference in MTD models*

Description

A function for estimating the relevant lag set Λ of a Markov chain using Bayesian Information Criterion (BIC). This means that this method selects the set of lags that minimizes a penalized log likelihood for a given sample, see *References* below for details on the method.

Usage

```
hdMTD_BIC(
  X,
  d,
  S = seq_len(d),
  minl = 1,
  maxl = length(S),
  xi = 1/2,
  A = NULL,
  byl = FALSE,
  BICvalue = FALSE,
  single_matrix = FALSE,
  indep_part = TRUE,
  zeta = maxl,
  warning = FALSE,
  ...
)
```

Arguments

`X` A vector or single-column data frame containing a chain sample (`X[1]` is the most recent).

`d` A positive integer representing an upper bound for the chain order.

`S` A numeric vector of positive integers from which this function will select a set of relevant lags. Typically, `S` is a subset of `1:d`. If `S` is not provided, by default `S=1:d`.

`minl` A positive integer. `minl` represents the smallest length of any relevant lag set this function might return. If `minl` == `maxl`, this function will return the subset of `S` of length `minl` with the lowest BIC. If `minl` < `maxl`, the function will consider subsets ranging from length `minl` to length `maxl` when searching for the subset of `S` with the smallest BIC.

`maxl` A positive integer equal to or greater than `minl` but less than the number of elements in `S` (`maxl` = `length(S)` is accepted but in this case the output will always be `S`). `maxl` represents the largest length of any relevant lag set this function might return.

`xi` The BIC penalization term constant. Defaulted to 1/2. A smaller `xi (near 0)` reduces the impact of overparameterization.

`A` A vector with positive integers representing the state space. If not informed, this function will set `A=sort(unique(X))`.

`byl` Logical. If `TRUE`, the function will look for the set with smallest BIC by each length (from `minl` to `maxl`), and return the set with smallest BIC for each length. If `minl==maxl` setting `byl=TRUE` or `FALSE` makes no difference, since the function will only calculate the BIC for sets with `maxl` elements in the relevant lag set.

`BICvalue` Logical. If `TRUE`, the function will also return the calculated values of the BIC for the estimated relevant lag sets.

single_matrix Logical. If TRUE, the chain sample is thought to come from an MTD model where the stochastic matrices p_j are constant across all lags $j \in \Lambda$. In practice, this means the user believes the stochastic matrices for every lag in S are the same, which reduces the number of parameters in the penalization term.

indep_part Logical. If FALSE there is no independent distribution and $\lambda_0 = 0$ which reduces the number of parameters in the penalization term.

zeta A positive integer representing the number of distinct matrices p_j in the MTD, which affects the number of parameters in the penalization term. Defaulted to maxl. See more in *Details*.

warning Logical. If TRUE, the function warns the user when A is set automatically.

... Additional arguments (not used in this function, but maintained for compatibility with hdMTD().

Details

Note that the upper bound for the order of the chain (d) affects the estimation of the transition probabilities. If we run the function with a certain order parameter d, only the sequences of length d that appeared in the sample will be counted. Therefore, all transition probabilities, and hence all BIC values, will be calculated with respect to that d. If we use another value for d to run the function, even if the output agrees with that of the previous run, its BIC value might change a little.

The parameter zeta indicates the the number of distinct matrices pj in the MTD. If zeta = 1, all matrices p_j are identical; if zeta = 2 there exists two groups of distinct matrices and so on. The largest value for zeta is maxl since this is the largest number of matrices p_j. When minl<maxl, for each minl $\leq$ l $\leq$ maxl, zeta = min(zeta,l). If single_matrix = TRUE then zeta is set to 1.

Value

Returns a vector with the estimated relevant lag set using BIC. It might return more than one set if minl < maxl and byl = TRUE. Additionally, it can return the value of the penalized likelihood for the outputted lag sets if BICvalue = TRUE.

References

Imre Csiszár, Paul C. Shields. The consistency of the BIC Markov order estimator. *The Annals of Statistics*, *28*(6), 1601-1619. doi:10.1214/aos/1015957472

Examples

```
X <- testChains[, 1]
hdMTD_BIC (X, d = 6, minl = 1, maxl = 1)
hdMTD_BIC (X,d = 3,minl = 1, maxl = 2, BICvalue = TRUE)
```

hdMTD_CUT *The CUT method for inference in MTD models*

Description

A function that estimates the set of relevant lags of an MTD model using the CUT method.

Usage

```
hdMTD_CUT(
  X,
  d,
  S = 1:d,
  alpha = 0.05,
  mu = 1,
  xi = 0.5,
  A = NULL,
  warning = FALSE,
  ...
)
```

Arguments

X	A vector or single-column data frame containing a chain sample (X[1] is the most recent).
d	A positive integer representing an upper bound for the chain order.
S	A numeric vector of distinct positive integers from which this function will select a set of relevant lags. Should be a subset of 1:d. Default is 1:d.
alpha	A positive real number used in the CUT threshold (which determines if two distributions can be considered different). The larger the alpha, the greater the distance required to consider that there is a difference between a set of distributions.
mu	A positive real number such that mu $> (e^{\text{mu}} - 1)/2$. mu is also a component of the same threshold as alpha.
xi	A positive real number, xi is also a component of the same threshold as alpha.
A	A vector with positive integers representing the state space. If not informed, this function will set A <- sort(unique(X)).
warning	Logical. If TRUE, the function warns the user when A is set automatically.
...	Additional arguments (not used in this function, but maintained for compatibility with hdMTD().

Details

The "Forward Stepwise and Cut" (FSC) is an algorithm for inference in Mixture Transition Distribution (MTD) models. It consists in the application of the "Forward Stepwise" (FS) step followed by the CUT algorithm. This method and its steps where developed by Ost and Takahashi and are specially useful for inference in high-order MTD Markov chains. This specific function will only apply the CUT step of the algorithm and return an estimated relevant lag set.

Value

Returns a set of relevant lags estimated using the CUT algorithm.

References

Ost, G. & Takahashi, D. Y. (2023). Sparse Markov models for high-dimensional inference. *Journal of Machine Learning Research*, *24*(279), 1-54. `http://jmlr.org/papers/v24/22-0266.html`

Examples

```
X <- testChains[,3]
hdMTD_CUT(X,4,alpha=0.02,mu=1,xi=0.4)
hdMTD_CUT(X,d=6,S=c(1,4,6),alpha=0.0065)
```

`hdMTD_FS` *The Forward Stepwise (FS) method for inference in MTD models*

Description

A function that estimates the set of relevant lags of an MTD model using the FS method.

Usage

```
hdMTD_FS(X, d, l, A = NULL, elbowTest = FALSE, warning = FALSE, ...)
```

Arguments

`X`	A vector or single-column data frame containing a chain sample (`X[1]` is the most recent).
`d`	A positive integer representing an upper bound for the chain order.
`l`	A positive integer specifying the number of lags to be selected as relevant.
`A`	A vector with positive integers representing the state space. If not informed, this function will set `A <- sort(unique(X))`.
`elbowTest`	Logical. If TRUE, the function applies an alternative stopping criterion to determine the length of the set of relevant lags. See *Details* for more information.
`warning`	Logical. If `TRUE`, the function warns the user when `A` is set automatically.
`...`	Additional arguments (not used in this function, but maintained for compatibility with `hdMTD()`.

Details

The "Forward Stepwise" (FS) algorithm is the first step of the "Forward Stepwise and Cut" (FSC) algorithm for inference in Mixture Transition Distribution (MTD) models. This method was developed by Ost and Takahashi This specific function will only apply the FS step of the algorithm and return an estimated relevant lag set of length `l`.

This method iteratively selects the most relevant lags based on a certain quantity ν. In the first step, the lag in `1:d` with the greatest ν is deemed important. This lag is included in the output, and using this knowledge, the function proceeds to seek the next important lag (the one with the highest ν among the remaining ones). The process stops when the output vector reaches length `l` if `elbowTest=FALSE`.

If `elbowTest = TRUE`, the function will store these maximum ν values at each iteration, and output only the lags that appear before the one with smallest ν among them.

Value

A numeric vector containing the estimated relevant lag set using FS algorithm.

References

Ost, G. & Takahashi, D. Y. (2023). Sparse Markov models for high-dimensional inference. *Journal of Machine Learning Research*, *24*(279), 1-54. `http://jmlr.org/papers/v24/22-0266.html`

Examples

```
X <- testChains[,1]
hdMTD_FS(X,d=5,l=2)
hdMTD_FS(X,d=4,l=3,elbowTest = TRUE)
```

hdMTD_FSC *Forward Stepwise and Cut method for inference in MTD models*

Description

A function for inference in MTD Markov chains with FSC method. This function estimates the relevant lag set Λ of an MTD model through the FSC algorithm.

Usage

```
hdMTD_FSC(X, d, l, alpha = 0.05, mu = 1, xi = 0.5, A = NULL, ...)
```

Arguments

`X` A vector or single-column data frame containing a chain sample (`X[1]` is the most recent).

`d` A positive integer representing an upper bound for the chain order.

`l` A positive integer that sets the number of elements in the output vector.

`alpha` A positive real number used in the CUT threshold (which determines if two distributions can be considered different). The larger the `alpha`, the greater the distance required to consider that there is a difference between a set of distributions. Defaulted to 0.05.

`mu` A positive real number such that $\texttt{mu} > (e^{\texttt{mu}} - 1)/2$. `mu` is also a component of the same threshold as `alpha`.

`xi` A positive real number, `xi` is also a component of the same threshold as `alpha`.

`A` A vector with positive integers representing the state space. If not informed, this function will set `A <- sort(unique(X))`.

`...` Additional arguments (not used in this function, but maintained for compatibility with `hdMTD()`.

Details

The "Forward Stepwise and Cut" (FSC) is an algorithm for inference in Mixture Transition Distribution (MTD) models. It consists in the application of the "Forward Stepwise" (FS) step followed by the CUT algorithm. This method and its steps where developed by Ost and Takahashi and are specially useful for inference in high-order MTD Markov chains.

Value

Returns a vector with the estimated relevant lag set using FSC algorithm.

References

Ost, G. & Takahashi, D. Y. (2023). Sparse Markov models for high-dimensional inference. *Journal of Machine Learning Research*, *24*(279), 1-54. `http://jmlr.org/papers/v24/22-0266.html`

Examples

```
X <- testChains[,1]
hdMTD_FSC(X,4,3,alpha=0.02)
hdMTD_FSC(X,4,2,alpha=0.001)
```

`MTDest` *EM estimation of MTD parameters*

Description

Estimation of MTD parameters through the Expectation Maximization (EM) algorithm.

Usage

```
MTDest(
  X,
  S,
  M = 0.01,
  init,
  iter = FALSE,
  nIter = 100,
  A = NULL,
  oscillations = FALSE
)
```

Arguments

`X` A vector or single-column data frame containing an MTD chain sample (`X[1]` is the most recent).

`S` A numeric vector of positive integers. Typically, `S` represents a set of relevant lags.

`M` A stopping point for the EM algorithm. If `M=NULL` the algorithm will run for a total of `nIter` iteractions.

`init` A list with initial parameters: `p0` (optional), `lambdas` (required), `pj` (required). The entries in `lambdas` are weights for the distribution `p0` and the distributions present in the list `pj`. Therefore, the order in which the elements appear in the vector `lambdas` is important for correct assignment. Please refer to the *Details* section for more information.

`iter` Logical. If `TRUE`, returns the number of iterations of the algorithm, that is, the number of times the initial parameters were updated.

`nIter` An integer positive number with the maximum number of iterations.

`A` A vector with positive integers representing the state space. If not informed, this function will set `A=unique(X)`.

`oscillations` Logical. If `TRUE`, the function will return the estimated oscillations for the updated model along with the estimated parameters.

Details

Regarding the M parameter: it functions as a stopping criterion within the EM algorithm. When the difference between the log-likelihood computed with the newly estimated parameters and that computed with the previous parameters falls below M, the algorithm halts. Nevertheless, if the value of nIter (which represents the maximum number of iterations) is smaller than the number of iterations required to meet the M criterion, the algorithm will conclude its execution when nIter is reached. To ensure that the M criterion is effectively utilized, we recommend using a higher value for nIter, which is set to a default of 100.

Concerning the init parameter, it is expected to be a list comprising either 2 or 3 entries. These entries consist of: an optional vector named p0, representing an independent distribution (the probability in the first entry of p0 must be that of the smallest element in A and so on), a required list of matrices pj, containing a stochastic matrix for each element of S (the first matrix must refer to the smallest element of S and so on), and a vector named lambdas representing the weights, the first entry must be the weight for p0, and then one entry for each element in pj list. If your MTD model does not have an independent distribution p0, set init$lambda[1]=0.

Value

A list with the estimated parameters of the MTD model.

References

Lebre, Sophie and Bourguignon, Pierre-Yves. (2008). An EM algorithm for estimation in the Mixture Transition Distribution model. *Journal of Statistical Computation and Simulation*, *78*(1), 1-15. doi:10.1080/00949650701266666

Examples

```
# Simulating data.
# Model:
MTD <- MTDmodel(Lambda=c(1,10),A=c(0,1),lam0=0.01)
# Sampling a chain:
X <- hdMTD::perfectSample(MTD,N=2000)

# Initial Parameters:
init <- list('p0'=c(0.4,0.6),'lambdas'=c(0.05,0.45,0.5),
  'pj'=list(matrix(c(0.2,0.8,0.45,0.55),byrow = TRUE,ncol=2),
   matrix(c(0.25,0.75,0.3,0.7),byrow = TRUE,ncol=2)))

# MTDest() ------------------------------------
MTDest(X,S=c(1,10),M=1,init)
MTDest(X,S=c(1,10),init=init,iter = TRUE)
MTDest(X,S=c(1,10),init=init,iter = TRUE,nIter=5)
MTDest(X,S=c(1,10),init=init,oscillations = TRUE)
```

MTDmodel *Creates a Mixture Transition Distribution (MTD) Model*

Description

Generates an MTD model as an object of class MTD given a set of parameters.

Usage

```
MTDmodel(
  Lambda,
  A,
  lam0 = NULL,
  lamj = NULL,
  pj = NULL,
  p0 = NULL,
  single_matrix = FALSE,
  indep_part = TRUE
)
```

Arguments

Lambda	A numeric vector of positive integers representing the relevant lag set. The elements will be sorted from smallest to greatest. The smallest number represents the latest (most recent) time in the past, and the largest number represents the earliest time in the past.
A	A vector with nonnegative integers representing the state space.
lam0	A numeric value in [0,1), representing the weight of the independent distribution.
lamj	A numeric vector of weights for the transition probability matrices in pj. Values must be in the range [0, 1), and their sum with lam0 must be equal to 1. The first element in lamj must be the weight for the first element in Lambda and so on.
pj	A list with length(Lambda) stochastic matrices, each of size length(A) x length(A). The first matrix in pj must refer to the first element in Lambda and so on.
p0	A probability vector for the independent component of the MTD model. If NULL and indep_part=TRUE, the distribution will be sampled from a uniform distribution. If indep_part=FALSE, then there is no independent distribution and p0 entries will be set to zero. If you enter p0=0, indep_part is set to FALSE.
single_matrix	Logical. If TRUE, all matrices in list pj are identical.
indep_part	Logical. If FALSE, the model does not include an independent distribution and p0 is set to zero.

Details

The resulting MTD object can be used by functions such as oscillation(), which retrieves the model's oscillation, and perfectSample(), which will sample an MTD Markov chain from its invariant distribution.

Value

A list of class MTD containing:

P The transition probability matrix of the MTD model.

lambdas A vector with MTD weights (lam0 and lamj).

pj A list of stochastic matrices defining conditional transition probabilities.

p0 The independent probability distribution.

Lambda The vector of relevant lags.

A The state space.

Examples

```
MTDmodel(Lambda=c(1,3),A=c(4,8,12))

MTDmodel(Lambda=c(2,4,9),A=c(0,1),lam0=0.05,lamj=c(0.35,0.2,0.4),
pj=list(matrix(c(0.5,0.7,0.5,0.3),ncol=2)),p0=c(0.2,0.8),single_matrix=TRUE)

MTDmodel(Lambda=c(2,4,9),A=c(0,1),lam0=0.05,
pj=list(matrix(c(0.5,0.7,0.5,0.3),ncol=2)),single_matrix=TRUE,indep_part=FALSE)
```

---

oscillation *Oscillations of an MTD Markov chain*

Description

Calculates the oscillations of an MTD model object or estimates the oscillations of a chain sample.

Usage

```
oscillation(x, ...)
```

Arguments

x	Must be an MTD object or a chain sample.
...	Additional parameters that might be required. Such as: S: If x is a chain sample the user should provide a set of lags for which he wishes to estimate the oscillations. It must be labeled as S, an in this scenario the function takes an upper bound for the order as d=max{S}. A: If x is a chain sample, and there may be elements in A that did not appear in x, the state space should be specified, and it must be labeled as A.

Details

The oscillation for a certain lag j of an MTD model ($\{\delta_j : j \in \Lambda\}$), is the product of the weight λ_j multiplied by the maximum of the total variation distance between the distributions in a stochastic matrix p_j.

$$\delta_j = \lambda_j \max_{b,c \in \mathcal{A}} d_{TV}(p_j(\cdot|b), p_j(\cdot|c)).$$

So, if `x` is an MTD object, the parameters Λ, $\mathcal{A}$, λ_j, and p_j are inputted through, respectively, the entries `Lambda`, `A`, `lambdas` and the list `pj` of stochastic matrices. Hence, an oscillation δ_j may be calculated for all $j \in \Lambda$.

If we wish to estimate the oscillations from a sample, then `x` must be a chain, and `S`, a vector representing a set of lags, must be informed. This way the transition probabilities can be estimated. Let $\hat{p}(\cdot|x_S)$ symbolize an estimated distribution in $\mathcal{A}$ given a certain past x_S (which is a sequence of elements of $\mathcal{A}$ where each element occurred at a lag in `S`), and $\hat{p}(\cdot|b_j x_S)$ an estimated distribution given past x_S and that the symbol $b \in \mathcal{A}$ occurred at lag j. If N is the sample size, $d =$`max(S)` and $N(x_S)$ is the number of times the sequence x_S appeared in the sample, then

$$\delta_j = \max_{c_j, b_j \in \mathcal{A}} \frac{1}{N-d} \sum_{x_S \in \mathcal{A}^S} N(x_S) d_{TV}(\hat{p}(.|b_j x_S), \hat{p}(.|c_j x_S))$$

is the estimated oscillation for a lag $j \in \{1, \ldots, d\}\backslash$`S`. Note that $\mathcal{A}^S$ is the space of sequences of $\mathcal{A}$ indexed by `S`.

Value

If the `x` parameter is an MTD object, it will provide the oscillations for each element in `Lambda`. In case `x` is a chain sample, it estimates the oscillations for a user-inputted set `S` of lags.

Examples

```
oscillation( MTDmodel(Lambda=c(1,4),A=c(2,3) ) )
oscillation(MTDmodel(Lambda=c(1,4),A=c(2,3),lam0=0.01,lamj=c(0.49,0.5),
pj=list(matrix(c(0.1,0.9,0.9,0.1),ncol=2)), single_matrix=TRUE))
```

`perfectSample` *Perfectly samples an MTD Markov chain*

Description

Samples an MTD Markov Chain from the stationary distribution.

Usage

```
perfectSample(MTD, N = NULL)
```

Arguments

`MTD`	An MTD object, see `MTDmodel()` for properly generating a MTD object.
`N`	The sample size. If `NULL` sample size will be set to 1000.

Details

This perfect sample algorithm requires that the MTD model has an independent distribution (p0) with a positive weight (i.e., `MTD$lambdas["lam0"]>0` which means $\lambda_0 > 0$).

Value

Returns a sample from an MTD model (the first element is the most recent).

Examples

```
perfectSample(MTDmodel(Lambda=c(1,4), A = c(0,2)), N = 200 )
perfectSample(MTDmodel(Lambda=c(2,5), A = c(1,2,3)), N = 1000 )
```

`probs` *Estimated transition probabilities*

Description

Computes the Maximum Likelihood estimators (MLE) for an MTD Markov chain with relevant lag set `S`.

Usage

```
probs(X, S, matrixform = FALSE, A = NULL, warning = FALSE)
```

Arguments

`X`	A vector or single-column data frame containing a sample of a Markov chain (`X[1]` is the most recent).
`S`	A numeric vector of unique positive integers. Typically, `S` represents a set of relevant lags.
`matrixform`	Logical. If `TRUE`, the output is formatted as a stochastic transition matrix.
`A`	A numeric vector of distinct integers representing the state space. If not provided, this function will set `A <- sort(unique(X))`.
`warning`	Logical. If `TRUE`, the function warns the user when the state space is automatically set as `A <- sort(unique(X))`.

Details

The probabilities are estimated as:

$$\hat{p}(a|x_S) = \frac{N(x_S a)}{N(x_S)}$$

where $N(x_S a)$ is the number of times the sequence x_S appeared in the sample followed by a, and $N(x_S)$ is the number of times x_S appeared (followed by any state). If $N(x_S) = 0$, the probability is set to $1/|A|$ (assuming a uniform distribution over `A`).

Value

A data frame or a matrix containing estimated transition probabilities:

- If `matrixform = FALSE`, the function returns a data frame with three columns:
    - The past sequence x_S (a concatenation of past states).
    - The current state a.
    - The estimated probability $\hat{p}(a|x_S)$.
- If `matrixform = TRUE`, the function returns a stochastic transition matrix, where rows correspond to past sequences x_S and columns correspond to states in `A`.

Examples

```
X <- testChains[, 3]
probs(X, S = c(1, 30))
probs(X, S = c(1, 15, 30))
```

raindata *Rain data set for the city of Canberra, Australia*

Description

A data frame with the rainfall history in the city of Canberra, Australia. The data spans from 01/11/2007 to 25/06/2017.

Usage

```
raindata
```

Format

A data frame with 3525 rows and 2 columns. Each row corresponds to a day specified in column 1 ("Date"). The value in column 2 ("RainToday") is 0 if no rain was recorded in the city of Canberra that day, and 1 otherwise.

Date Date in YYYY-MM-DD format.

RainToday Binary indicator (0 = no rain, 1 = rain).

Note

The original BOM data is subject to their Terms of Use. For direct access, visit the BOM website manually: https://www.bom.gov.au/climate/data/ (may require browser access).

Source

Original data source: Australian Bureau of Meteorology (BOM). **Accessed via**: Kaggle (https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package).

Examples

```
data(raindata)
```

---

sleepscoring *Data with sleeping patterns*

Description

The dataset contains 136,925 rows and 7 columns, representing the sleeping patterns of 151 patients over the course of one night, with measurements taken at 30-second intervals. It is a collection of 151 whole-night polysomno-graphic (PSG) sleep recordings (85 Male, 66 Female, mean age of 53.9 ± 15.4) collected during 2018 at the Haaglanden Medisch Centrum (HMC, The Netherlands) sleep center.

Usage

```
sleepscoring
```

Format

A tbl_df object with 136,925 rows representing the sleeping patterns of 151 patients.

Patient Identifies the patient.

Date Date when the measurements where made.

Time Time each measurement was made.

Recorging.onset Time, in seconds, since the beginning of the recordings.

Duration The duration of each lag between recordings, in seconds.

SleepStage The annotated sleeping stage. ’W’ refers to wakefulness, ’R’ to REM sleep, and ’N1’, ’N2’, and ’N3’ refer to non-REM stages 1, 2, and 3 respectively.

ConscientLevel A measurement of the level of consciousness during different sleep stages. 0 indicates Wake, 1 represents REM sleep, and 2, 3, and 4 correspond to N1, N2, and N3 stages respectively.

Source

Alvarez-Estevez, D. & Rijsman, R. M. (2022). Haaglanden Medisch Centrum sleep staging database (version 1.1). PhysioNet. doi:10.13026/t79qfr32

Alvarez-Estevez, D. & Rijsman, R. M. (2021). Inter-database validation of a deep learning approach for automatic sleep scoring. *PLOS ONE*, *16*(8), 1-27. doi:10.1371/journal.pone.0256111

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K. & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, *101*(23), e215-e220. doi:10.1161/01.CIR.101.23.e215

Examples

```
data(sleepscoring)
```

tempdata *Maximum temperatures in the city of Brasília, Brazil.*

Description

A data frame with the maximum temperature of the last hour, by each hour, in the city of Brasília, Brazil. The data spans from 01/01/2003 to 31/08/2024.

Usage

```
tempdata
```

Format

A data frame with 189936 rows and 3 columns. Each row corresponds to a time in a day specified in columns 2 ("TIME") and 1 ("DATE") respectively. The value in column 3 ("MAXTEMP") is the maximum temperature measured in the last hour, in Celsius (Cº), in the city of Brasília, the capital of Brazil, located in the central-western part of the country.

DATE The day, from 01/01/2003 to 31/08/2024

TIME The time, form 00:00 to 23:00 each day

MAXTEMP The maximum temperature measured in the last hour in Celsius

Source

Meteorological data provided by INMET (National Institute of Meteorology, Brazil). Data collected from automatic weather station in Brasília (latitude: -15.79°, longitude: -47.93°, altitude: 1159.54 m). Available at: `https://bdmep.inmet.gov.br/`

Examples

```
data(tempdata)
```

`testChains` *MTD samples for tests*

Description

A tibble with chains perfectly sampled from a MTD model. Each chain was sampled from the same MTD model. Hence the differences between the samples are due to randomness within the perfect sample algorithm.

Usage

```
testChains
```

Format

A tibble with 3000 rows and 3 perfectly sampled chains.

testChain1 perfectSample1,N=3000

testChain2 perfectSample2,N=3000

testChain3 perfectSample3,N=3000

Details

The MTD model from which the chains were sampled was created as follows:

set.seed(1)

pj <- list("p-1"=matrix(c(0.1,0.1,0.8,0.4,0.4,0.2,0.5,0.3,0.2), byrow = T,ncol = 3),

"p-30"=matrix(c(0.05,0.2,0.75,0.4,0.4,0.2,0.3,0.3,0.4), byrow = T,ncol = 3))

MTDseed1 <- MTDmodel(Lambda=c(1,30),A=c(1,2,3),lam0=0.05, lamj = c(0.35,0.6),pj=pj)

testChain1 <- perfectSample(MTDseed1,3000)

testChain2 <- perfectSample(MTDseed1,3000)

testChain3 <- perfectSample(MTDseed1,3000)

testChains <- dplyr::as_tibble(cbind(testChain1,testChain2,testChain3))

Source

Created in-house to serve as example.

Examples

```
data(testChains)
```

Index