Package 'drimmR'

May 8, 2026

Type Package

Title Estimation, Simulation and Reliability of Drifting Markov Models

Version 1.0.3

Description Performs the drifting Markov models (DMM) which are non-homogeneous Markov models designed for modeling the heterogeneities of sequences in a more flexible way than homogeneous Markov chains or even hidden Markov models. In this context, we developed an R package dedicated to the estimation, simulation and the exact computation of associated reliability of drifting Markov models. The implemented methods are described in Vergne, N. (2008), <doi:10.2202/1544-6115.1326> and Barbu, V.S., Vergne, N. (2019) <doi:10.1007/s11009-018-9682-8> .

License GPL-3

Depends R (>= 2.10)

Encoding UTF-8

LazyData true

Suggests utils, knitr, rmarkdown

Imports seqinr, ggplot2, parallel, parallelly, doParallel, foreach, dplyr, Rdpack, reshape2

RdMacros Rdpack

RoxygenNote 7.3.3

NeedsCompilation no

Author Nicolas Vergne [aut, cre] (<nicolas.vergne@univ-rouen.fr>), Corentin Lothodé [aut] (<corentin.lothode@univ-angers.fr>), Alexandre Seiller [aut], Victor Mataigne [ctb], Arnaud Lefebvre [ctb], Annthomy Gilles [ctb], Vlad Stefan Barbu [aut] (<vladstefan.barbu@univ-rouen.fr>)

Maintainer Nicolas Vergne <nicolas.vergne@univ-rouen.fr>

Repository CRAN

Date/Publication 2026-02-12 22:00:02 UTC

Contents

drimmR-package . . . 2
aic . . . 3
aic.dmm . . . 3
availability . . . 5
bic . . . 6
bic.dmm . . . 7
distributions . . . 8
failureRate . . . 9
fitdmm . . . 11
getDistribution . . . 13
getDistribution.dmm . . . 14
getStationaryLaw . . . 16
getStationaryLaw.dmm . . . 17
getTransitionMatrix . . . 18
getTransitionMatrix.dmm . . . 19
lambda . . . 20
lengthWord_probabilities . . . 20
loglik . . . 21
loglik.dmm . . . 22
maintainability . . . 23
reliability . . . 24
simulate . . . 26
simulate.dmm . . . 27
stationary_distributions . . . 28
words_probabilities . . . 29
word_probabilities . . . 30
word_probability . . . 31

Index **33**

`drimmR-package` *drimmR-package*

Description

This package performs the estimation, simulation and the exact computation of associated reliability measures of drifting Markov models (DMMs). These are particular non-homogeneous Markov chains for which the Markov transition matrix is a linear (polynomial) function of two (several) Markov transition matrices. Several statistical frameworks are taken into account (one or several samples, complete or incomplete samples, models of the same length). Computation of probabilities of appearance of a word along a sequence under a given model are also considered.

aic *Akaike Information Criterion (AIC)*

Description

Generic function computing the Akaike Information Criterion of the model `x`, with the list of sequences `sequences`.

Usage

```
aic(x, sequences, ncpu = 2)
```

Arguments

`x`	An object for which the log-likelihood of the DMM can be computed.
`sequences`	A vector of characters or a list of vector of characters representing the sequences for which the AIC will be computed based on `x`.
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A list of AIC (numeric)

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

aic.dmm *Evaluate the AIC of a drifting Markov Model*

Description

Computation of the Akaike Information Criterion.

Usage

```
## S3 method for class 'dmm'
aic(x, sequences, ncpu = 2)
```

Arguments

x	An object of class dmm
sequences	A character vector or a list of character vector representing the sequences for which the AIC will be computed based on x.
ncpu	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A list of AIC (numeric)

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, loglik, aic

Examples

```
data(lambda, package = "drimmR")
sequence <- c("a","g","g","t","c","g","a","t","a","a","a")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
aic(dmm,sequence)
```

availability *Availability function*

Description

The pointwise (or instantaneous) availability of a system S_{ystem} at time $k \in N$ is the probability that the system is operational at time k (independently of the fact that the system has failed or not in $[0; k)$).

Usage

```
availability(
  x,
  k1 = 0L,
  k2,
  upstates,
  output_file = NULL,
  plot = FALSE,
  ncpu = 2
)
```

Arguments

x	An object of class dmm
k1	Start position (default value=0): a positive integer giving the start position along the sequence from which the availabilities of the DMM should be computed, such that k1<k2
k2	End position : a positive integer giving the end position along the sequence until which the availabilities of the DMM should be computed, such that k2>k1
upstates	Character vector giving the subset of operational states U.
output_file	(Optional) A file containing matrix of availability probabilities (e.g, "C:/.../AVAL.txt")
plot	FALSE (default); TRUE (display a figure plot of availability probabilities by position)
ncpu	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

Consider a system (or a component) System whose possible states during its evolution in time are $E = \{1 \dots s\}$. Denote by $U = \{1 \dots s_1\}$ the subset of operational states of the system (the upstates) and by $D = \{s_1 + 1 \dots s\}$ the subset of failure states (the down states), with 0 < s1 < s(obviously, $E = U \cup DandU \cap D = \emptyset, U \neq \emptyset, D \neq \emptyset$). One can think of the states of U as different operating modes or performance levels of the system, whereas the states of D can be seen as failures of the systems with different modes.

Value

A vector of length k+1 giving the values of the availability for the period $[0 \dots k]$

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828.

See Also

fitdmm, getTransitionMatrix, reliability, maintainability

Examples

```
data(lambda, package = "drimmR")
length(lambda) <- 1000
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq",
 fit.method="sum")
k1 <- 1
k2 <- 200
upstates <- c("c","t")  # vector of working states
getA <- availability(dmm,k1,k2,upstates,plot=TRUE)
```

---

bic *Bayesian Information Criterion (BIC)*

Description

Generic function computing the Bayesian Information Criterion of the model `x`, with the list of sequences `sequences`.

Usage

```
bic(x, sequences, ncpu = 2)
```

Arguments

`x`	An object for which the log-likelihood of the DMM can be computed.
`sequences`	A list of vectors representing the sequences for which the BIC will be computed based on `x`.
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A list of BIC (numeric)

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

`bic.dmm` *Evaluate the BIC of a drifting Markov Model*

Description

Computation of the Bayesian Information Criterion.

Usage

```
## S3 method for class 'dmm'
bic(x, sequences, ncpu = 2)
```

Arguments

`x`	An object of class `dmm`
`sequences`	A character vector or a list of character vector representing the sequences for which the BIC will be computed based on `x`.
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A list of BIC (numeric).

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, loglik, bic

Examples

```
data(lambda, package = "drimmR")
sequence <- c("a","g","g","t","c","g","a","t","a","a","a")
dmm<- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
bic(dmm,sequence)
```

`distributions` *Distributions for a range of positions between <start> and <end>*

Description

Distributions for a range of positions between <start> and <end>

Usage

```
distributions(
  x,
  start = 1,
  end = NULL,
  step = NULL,
  output_file = NULL,
  plot = FALSE,
  ncpu = 2
)
```

Arguments

`x`	An object of class `dmm`
`start`	Start position : a positive integer giving the start position along the sequence from which the distributions of the DMM should be computed
`end`	End position : a positive integer giving the end position along the sequence until which the distributions of the DMM should be computed
`step`	A step (integer)
`output_file`	(Optional) A file containing matrix of distributions (e.g, "C:/.../DIST.txt")

plot	FALSE (default); TRUE (display a figure plot of distribution probabilities by position)
ncpu	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A matrix with positions and distributions of states

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getDistribution, getStationaryLaw

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
distributions(dmm,start=1,end=1000,step=100, plot=TRUE)
```

---

failureRate *Failure rates function*

Description

Computation of two different definition of the failure rate : the BMP-failure rate and RG-failure rate.

As for BMP-failure rate, consider a system S starting to work at time $k = 0$. The BMP-failure rate at time $k \in N$ is the conditional probability that the failure of the system occurs at time k, given that the system has worked until time $k - 1$. The BMP-failure rate denoted by $\lambda(k), k \in N$ is usually considered for continuous time systems.

The RG-failure rate is a discrete-time adapted failure-rate proposed by D. Roy and R. Gupta. Classification of discrete lives. *Microelectronics Reliability*, 32(10):1459–1473, 1992. The RG-failure rate is denoted by $r(k), k \in N$.

Usage

```
failureRate(
  x,
  k1 = 0L,
  k2,
  upstates,
  failure.rate = c("BMP", "RG"),
  output_file = NULL,
  plot = FALSE
)
```

Arguments

x	An object of class dmm
k1	Start position (default value=0) : a positive integer giving the start position along the sequence from which the failure rates of the DMM should be computed, such that k1<k2
k2	End position : a positive integer giving the end position along the sequence until which the failure rates of the DMM should be computed, such that k2>k1
upstates	Character vector of the subspace working states among the state space vector such that upstates < s
failure.rate	Default="BMP", then BMP-failure-rate is the method used to compute the failure rate. If failure.rate= "RG", then RG-failure rate is the method used to compute the failure rate.
output_file	(Optional) A file containing matrix of failure rates at each position (e.g, "C:/.../ER.txt")
plot	FALSE (default); TRUE (display a figure plot of failure rates by position)

Details

Consider a system (or a component) System whose possible states during its evolution in time are $E = \{1 \dots s\}$. Denote by $U = \{1 \dots s_1\}$ the subset of operational states of the system (the upstates) and by $D = \{s_1 + 1 \dots s\}$ the subset of failure states (the down states), with 0 < s1 < s(obviously, $E = U \cup DandU \cap D = \emptyset, U \neq \emptyset, D \neq \emptyset$). One can think of the states of U as different operating modes or performance levels of the system, whereas the states of D can be seen as failures of the systems with different modes.

Value

A vector of length k + 1 giving the values of the BMP (or RG) -failure rate for the period $[0 \dots k]$

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Roy D, Gupta R (1992). "Classification of discrete lives. Microelectronics Reliability." *Microelectronics Reliability*, 1459–1473. doi:10.1016/00262714(92)90015D.

See Also

fitdmm, getTransitionMatrix, reliability

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq",
 fit.method="sum")
k1 <- 1
k2 <- 200
upstates <- c("c","t")  # vector of working states
failureRate(dmm,k1,k2,upstates,failure.rate="BMP",plot=TRUE)
```

`fitdmm` *Point by point estimates of a k-th order drifting Markov Model*

Description

Estimation of d+1 points of support transition matrices and $|E|^k$ initial law of a k-th order drifting Markov Model starting from one or several sequences.

Usage

```
fitdmm(
  sequences,
  order,
  degree,
  states,
  init.estim = c("mle", "freq", "prod", "stationary", "unif"),
  fit.method = c("sum"),
  ncpu = 2
)
```

Arguments

`sequences`	A list of character vector(s) representing one (several) sequence(s)
`order`	Order of the Markov chain
`degree`	Degree of the polynomials (e.g., linear drifting if `degree=1`, etc.)
`states`	Vector of states space of length s > 1

`init.estim` Default="mle". Method used to estimate the initial law. If `init.estim` = "mle", then the classical Maximum Likelihood Estimator is used, if `init.estim` = "freq", then, the initial distribution `init.estim` is estimated by taking the frequences of the words of length k for all sequences. If `init.estim` = "prod", then, `init.estim` is estimated by using the product of the frequences of each letter (for all the sequences) in the word of length k. If `init.estim` = "stationary", then `init.estim` is estimated by using the stationary law of the point of support transition matrices of each letter. If `init.estim` = "unif", then, `init.estim` of each letter is estimated by using $\frac{1}{s}$. Or 'init.estim'= customisable vector of length $|E|^k$. See Details for the formulas.

`fit.method` If `sequences` is a list of several character vectors of the same length, the usual LSE over the sample paths is proposed when `fit.method`="sum" (a list of a single character vector is its special case).

`ncpu` Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

The fitdmm function creates a drifting Markov model object `dmm`.

Let $E = 1, \ldots, s$, s < ∞ be random system with finite state space, with a time evolution governed by discrete-time stochastic process of values in E. A sequence $X_0, X_1, \ldots, X_n$ with state space $E = 1, 2, \ldots, s$ is said to be a linear drifting Markov chain (of order 1) of length n between the Markov transition matrices Π_0 and Π_1 if the distribution of X_t, $t = 1, \ldots, n$, is defined by $P(X_t = v \mid X_{t-1} = u, X_{t-2}, \ldots) = \Pi_{\frac{t}{n}}(u, v), ; u, v \in E$, where $\Pi_{\frac{t}{n}}(u, v) = (1 - \frac{t}{n})\Pi_0(u, v) + \frac{t}{n}\Pi_1(u, v),\ u, v \in E$. The linear drifting Markov model of order 1 can be generalized to polynomial drifting Markov model of order k and degree d.Let $\Pi_{\frac{i}{d}} = (\Pi_{\frac{i}{d}}(u_1, \ldots, u_k, v))_{u_1,\ldots,u_k,v \in E}$ be d Markov transition matrices (of order k) over a state space E.

The estimation of DMMs is carried out for 4 different types of data :

One can observe one sample path : It is denoted by $H(m, n) := (X_0, X_1, \ldots, X_m)$, where m denotes the length of the sample path and n the length of the drifting Markov chain. Two cases can be considered:

1. m=n (a complete sample path),
2. m < n (an incomplete sample path).

One can also observe H i.i.d. sample paths : It is denoted by $H_i(m_i, n_i), i = 1, \ldots, H$. Two cases cases are considered :

1. $m_i = n_i = n \forall i = 1, \ldots, H$ (complete sample paths of drifting Markov chains of the same length),
2. $n_i = n \forall i = 1, \ldots, H$ (incomplete sample paths of drifting Markov chains of the same length). In this case, an usual LSE over the sample paths is used.

The initial distribution of a k-th order drifting Markov Model is defined as $\mu_i = P(X_1 = i)$. The initial distribution of the k first letters is freely customisable by the user, but five methods are proposed for the estimation of the latter :

Estimation based on the Maximum Likelihood Estimator: The Maximum Likelihood Estimator for the initial distribution. The formula is: $\widehat{\mu_i} = \frac{Nstart_i}{L}$, where $Nstart_i$ is the number of

occurences of the word i (of length k) at the beginning of each sequence and L is the number of sequences. This estimator is reliable when the number of sequences L is high.

Estimation based on the frequency: The initial distribution is estimated by taking the frequences of the words of length k for all sequences. The formula is $\widehat{\mu_i} = \frac{N_i}{N}$, where N_i is the number of occurences of the word i (of length k) in the sequences and N is the sum of the lengths of the sequences.

Estimation based on the product of the frequences of each state: The initial distribution is estimated by using the product of the frequences of each state (for all the sequences) in the word of length k.

Estimation based on the stationary law of point of support transition matrix for a word of length k : The initial distribution is estimated using $\mu(\Pi_{\frac{k-1}{n}})$

Estimation based on the uniform law : $\frac{1}{s}$

Value

An object of class `dmm`

Author(s)

Geoffray Brelurut, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

Examples

```
data(lambda, package = "drimmR")
states <- c("a","c","g","t")
order <- 1
degree <- 1
fitdmm(lambda,order,degree,states, init.estim = "freq",fit.method="sum")
```

`getDistribution` *Distributions of the drifting Markov Model*

Description

Generic function evaluating the distribution of a model `x` at a given position `pos` or at every position `all.pos`

Usage

```
getDistribution(x, pos, all.pos = FALSE, internal = FALSE, ncpu = 2)
```

Arguments

x	An object for which the distributions of the DMM can be computed.
pos	A positive integer giving the position along the sequence on which the distribution of the DMM should be computed
all.pos	'FALSE' (evaluation at position index) ; 'TRUE' (evaluation for all position indices)
internal	'FALSE' (default) ; 'TRUE' (for internal use of the distributions function)
ncpu	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

Distribution at position l is evaluated by $\mu_l = \mu_0 \prod_{t=k}^{l} \pi_{\frac{t}{n}}, \forall l \geq k, k \in N^*$ order of the DMM

Value

A vector or matrix of distribution probabilities

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

---

getDistribution.dmm *Get the distributions of the DMM*

Description

Evaluate the distribution of the DMM at a given position or at every position

Usage

```
## S3 method for class 'dmm'
getDistribution(x, pos, all.pos = FALSE, internal = FALSE, ncpu = 2)
```

Arguments

x	An object of class dmm
pos	A positive integer giving the position along the sequence on which the distribution of the DMM should be computed
all.pos	'FALSE' (default, evaluation at position index) ; 'TRUE' (evaluation for all position indices)
internal	'FALSE' (default) ; 'TRUE' (for internal use of distributions function)
ncpu	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

Distribution at position l is evaluated by $\mu_l = \mu_0 \prod_{t=k}^{l} \pi_{\frac{t}{n}}, \forall l \geq k, k \in N^*$ order of the DMM

Value

A vector or matrix of distribution probabilities

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, distributions, getStationaryLaw

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
t <- 10
getDistribution(dmm,pos=t)
```

`getStationaryLaw` *Stationary laws of the drifting Markov Model*

Description

Generic function evaluating the stationary law of a model `x` at a given position `pos` or at every position `all.pos`

Usage

```
getStationaryLaw(x, pos, all.pos = FALSE, internal = FALSE, ncpu = 2)
```

Arguments

`x`	An object for which the stationary laws of the DMM can be computed.
`pos`	A positive integer giving the position along the sequence on which the stationary law of the DMM should be computed
`all.pos`	'FALSE' (default, evaluation at position index) ; 'TRUE' (evaluation for all position indices)
`internal`	'FALSE' (default) ; 'TRUE' (for internal use of the initial law computation)
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

Stationary law at position t is evaluated by solving $\mu_t \ \pi_{\frac{t}{n}} = \mu$

Value

A vector or matrix of stationary law probabilities

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

`getStationaryLaw.dmm` *Get the stationary laws of the DMM*

Description

Evaluate the stationary law of the DMM at a given position or at every position

Usage

```
## S3 method for class 'dmm'
getStationaryLaw(x, pos, all.pos = FALSE, internal = FALSE, ncpu = 2)
```

Arguments

`x`	An object of class `dmm`
`pos`	A positive integer giving the position along the sequence on which the stationary law of the DMM should be computed
`all.pos`	'FALSE' (default, evaluation at position index) ; 'TRUE' (evaluation for all position indices)
`internal`	'FALSE' (default) ; 'TRUE' (for internal use of th initial law of fitdmm and word applications)
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

Stationary law at position t is evaluated by solving $\mu_t \, \pi_{\frac{t}{n}} = \mu$

Value

A vector or matrix of stationary law probabilities

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, stationary_distributions, getDistribution

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
t <- 10
getStationaryLaw(dmm,pos=t)
```

`getTransitionMatrix` *Transition matrix of the drifting Markov Model*

Description

Generic function evaluating the transition matrix of a model `x` at a given position `pos`

Usage

```
getTransitionMatrix(x, pos)
```

Arguments

`x`	An object for which the transition matrices of the DMM can be computed.
`pos`	A positive integer giving the position along the sequence on which the transition matrix of the DMM should be computed

Value

A transition matrix at a given position

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

`getTransitionMatrix.dmm` *Get transition matrix of the drifting Markov Model*

Description

Evaluate the transition matrix of the DMM at a given position

Usage

```
## S3 method for class 'dmm'
getTransitionMatrix(x, pos)
```

Arguments

`x`	An object of class `dmm`
`pos`	A positive integer giving the position along the sequence on which the transition matrix of the DMM should be computed

Value

A transition matrix at a given position

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'),init.estim = "freq", fit.method="sum")
t <- 10
getTransitionMatrix(dmm,pos=t)
```

lambda *lambda genome*

Description

Complete data from phage genome [WT71] of length 48502

Usage

```
data(lambda)
```

Format

A vector object of class "Rdata".

Examples

```
data(lambda)
```

lengthWord_probabilities *Probability of occurrence of the observed word of size m in a sequence at several positions*

Description

Probability of occurrence of the observed word of size m in a sequence at several positions

Usage

```
lengthWord_probabilities(m, sequence, pos, x, output_file = NULL, plot = FALSE)
```

Arguments

m	An integer, the length word
sequence	A vector of characters
pos	A vector of integer positions
x	An object of class dmm
output_file	(Optional) A file containing the vector of probabilities (e.g,"C:/.../PROB.txt")
plot	FALSE (default); TRUE (display figure plots of probabilities of occurrence of the observed word of size m by position)

Value

A dataframe of probability by position

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, word_probability

Examples

```
data(lambda, package = "drimmR")
length(lambda) <- 1000
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
m <- 2
lengthWord_probabilities(m, lambda, c(1,length(lambda)-m), dmm, plot=TRUE)
```

---

loglik *Log-likelihood of the drifting Markov Model*

Description

Generic function computing the log-likelihood of the model `x`, with the list of sequences `sequences`.

Usage

```
loglik(x, sequences, ncpu = 2)
```

Arguments

`x`	An object for which the log-likelihood of the DMM can be computed.
`sequences`	A vector of character or list of vectors representing the sequences for which the
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores. log-likelihood of the model must be computed.

Value

A list of loglikelihood (numeric)

Author(s)

Annthomy Gilles, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

`loglik.dmm` *Evaluate the log-likelihood of a drifting Markov Model*

Description

Evaluate the log-likelihood of a drifting Markov Model

Usage

```
## S3 method for class 'dmm'
loglik(x, sequences, ncpu = 2)
```

Arguments

`x`	An object of class `dmm`
`sequences`	A character vector or a list of character vectors representing the sequence
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A list of log-likelihood (numeric)

Author(s)

Annthomy Gilles, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix

Examples

```
data(lambda, package = "drimmR")
sequence <- c("a","g","g","t","c","g","a","t","a","a","a")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
loglik(dmm,sequence)
```

`maintainability` *Maintainability function*

Description

Maintainability of a system at time $k \in N$ is the probability that the system is repaired up to time k, given that is has failed at time $k = 0$.

Usage

```
maintainability(
  x,
  k1 = 0L,
  k2,
  upstates,
  output_file = NULL,
  plot = FALSE,
  ncpu = 2
)
```

Arguments

`x`	An object of class `dmm`
`k1`	Start position (default value=0) : a positive integer giving the start position along the sequence from which the maintainabilities of the DMM should be computed, such that `k1<k2`
`k2`	End position : a positive integer giving the end position along the sequence until which the maintainabilities of the DMM should be computed, such that `k2>k1`
`upstates`	Character vector of the subspace working states among the state space vector such that upstates < s
`output_file`	(Optional) A file containing matrix of maintainability probabilities (e.g, "C:/.../MAIN.txt")
`plot`	`FALSE` (default); `TRUE` (display a figure plot of maintainability probabilities by position)
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

Consider a system (or a component) System whose possible states during its evolution in time are $E = \{1 \dots s\}$. Denote by $U = \{1 \dots s_1\}$ the subset of operational states of the system (the upstates) and by $D = \{s_1 + 1 \dots s\}$ the subset of failure states (the down states), with 0 < s1 < s(obviously, $E = U \cup DandU \cap D = \emptyset, U \neq \emptyset, D \neq \emptyset$). One can think of the states of U as different operating modes or performance levels of the system, whereas the states of D can be seen as failures of the systems with different modes.

Value

A vector of length k + 1 giving the values of the maintainability for the period $[0 \dots k]$

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828.

See Also

fitdmm, getTransitionMatrix

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'),
init.estim = "freq", fit.method="sum")
k1 <- 1
k2 <- 200
upstates <- c("c","t")  # vector of working states
maintainability(dmm,k1,k2,upstates,plot=TRUE)
```

---

reliability *Reliability function*

Description

Reliability or the survival function of a system at time $k \in N$

Usage

```
reliability(
  x,
  k1 = 0L,
  k2,
  upstates,
  output_file = NULL,
  plot = FALSE,
  ncpu = 2
)
```

Arguments

x	An object of class dmm
k1	Start position (default value=0) : a positive integer giving the start position along the sequence from which the reliabilities of the DMM should be computed, such that k1<k2
k2	End position : a positive integer giving the end position along the sequence until which the reliabilities of the DMM should be computed, such that k2>k1
upstates	Character vector of the subspace working states among the state space vector such that upstates < s
output_file	(Optional) A file containing matrix of reliability probabilities (e.g, "C:/.../REL.txt")
plot	FALSE (default); TRUE (display a figure plot of reliability probabilities by position)
ncpu	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Details

The reliability at time $k \in N$ is the probability that the system has functioned without failure in the period $[0, k]$

Value

A vector of length k + 1 giving the values of the reliability for the period $[0 \ldots k]$

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828.

See Also

fitdmm, getTransitionMatrix

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq",
fit.method="sum")
k1 <- 1
k2 <- 200
upstates <- c("c","t")  # vector of working states
reliability(dmm,k1,k2,upstates,plot=TRUE)
```

`simulate` *Simulate a sequence under a drifting Markov model*

Description

Generic function simulating a sequence of length `model_size` under a model `x`

Usage

```
simulate(x, output_file, model_size = 1e+05, ncpu = 2)
```

Arguments

`x`	An object for which simulated sequences of the DMM can be computed.
`output_file`	(Optional) File containing the simulated sequence (e.g, "C:/.../SIM.txt")
`model_size`	Size of the model
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

the vector of simulated sequence

Author(s)

Annthomy Gilles, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

`simulate.dmm` *Simulate a sequence under a drifting Markov model*

Description

Simulate a sequence under a k-th order DMM.

Usage

```
## S3 method for class 'dmm'
simulate(x, output_file = NULL, model_size = NULL, ncpu = 2)
```

Arguments

`x`	An object of class `dmm`
`output_file`	(Optional) File containing the simulated sequence (e.g, "C:/.../SIM.txt")
`model_size`	Size of the model
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

the vector of simulated sequence

Author(s)

Annthomy Gilles, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, getStationaryLaw

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
simulate(dmm, model_size=100)
```

`stationary_distributions` *Stationary laws for a range of positions between <start> and <end>*

Description

Stationary laws for a range of positions between <start> and <end>

Usage

```
stationary_distributions(
  x,
  start = 1,
  end = NULL,
  step = NULL,
  output_file = NULL,
  plot = FALSE
)
```

Arguments

`x`	An object of class `dmm`
`start`	Start position : a positive integer giving the start position along the sequence from which the stationary laws of the DMM should be computed
`end`	End position : a positive integer giving the end position along the sequence until which the stationary laws of the DMM should be computed
`step`	A step (integer)
`output_file`	(Optional) A file containing matrix of stationary laws (e.g, "C:/.../SL.txt")
`plot`	`FALSE` (default); `TRUE` (display a figure plot of stationary distribution probabilities by position)

Value

A matrix with positions and stationary laws of states

Author(s)

Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getStationaryLaw

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq",fit.method="sum")
stationary_distributions(dmm,start=1,end=1000,step=100, plot=TRUE)
```

`words_probabilities` *Probability of appearance of several words at several positions of a DMM*

Description

Probability of appearance of several words at several positions of a DMM

Usage

```
words_probabilities(words, pos, x, output_file = NULL, plot = FALSE)
```

Arguments

`words`	A vector of characters containing words
`pos`	A vector of integer positions
`x`	An object of class `dmm`
`output_file`	(Optional) A file containing the matrix of probabilities (e.g,"C:/.../PROB.txt")
`plot`	`FALSE` (default); `TRUE` (display figure plots of words' probabilities by position)

Value

A dataframe of word probabilities along the positions of the sequence

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, word_probability, word_probabilities

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq",
fit.method="sum")
words <- c("atcgattc", "taggct", "ggatcgg")
pos <- c(100,300)
words_probabilities(words=words,pos=pos,dmm,plot=TRUE)
```

`word_probabilities` *Probabilities of a word at several positions of a DMM*

Description

Probabilities of a word at several positions of a DMM

Usage

```
word_probabilities(word, pos, x, output_file = NULL, plot = FALSE)
```

Arguments

`word`	A subsequence (string of characters)
`pos`	A vector of integer positions
`x`	An object of class `dmm`
`output_file`	(Optional) A file containing the vector of probabilities (e.g,"C:/.../PROB.txt")
`plot`	`FALSE` (default); `TRUE` (display figure plot of word's probabilities by position)

Value

A numeric vector, probabilities of `word`

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, word_probability, words_probabilities

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'),
init.estim = "freq", fit.method="sum")
word_probabilities("aggctga",c(100,300),dmm, plot=TRUE)
```

`word_probability` *Probability of a word at a position t of a DMM*

Description

Probability of a word at a position t of a DMM

Usage

```
word_probability(word, pos, x, output_file = NULL, internal = FALSE, ncpu = 2)
```

Arguments

`word`	A subsequence (string of characters)
`pos`	A position (numeric)
`x`	An object of class `dmm`
`output_file`	(Optional) A file containing the probability (e.g,"C:/.../PROB.txt")
`internal`	`FALSE` (default) ; `TRUE` (for internal use of word applications)
`ncpu`	Default=2. Represents the number of cores used to parallelized computation. If ncpu=-1, then it uses all available cores.

Value

A numeric, probability of `word`

Author(s)

Victor Mataigne, Alexandre Seiller

References

Barbu VS, Vergne N (2018). "Reliability and survival analysis for drifting Markov models: modelling and estimation." *Methodology and Computing in Applied Probability*, 1–33. doi:10.1007/s1100901896828. Vergne N (2008). "Drifting Markov models with polynomial drift and applications to DNA sequences." *Statistical Applications in Genetics Molecular Biology*, **7(1)**. doi:10.2202/15446115.1326.

See Also

fitdmm, getTransitionMatrix, word_probabilities, words_probabilities

Examples

```
data(lambda, package = "drimmR")
dmm <- fitdmm(lambda, 1, 1, c('a','c','g','t'), init.estim = "freq", fit.method="sum")
word_probability("aggctga",4,dmm)
```

Index

∗ **datasets**
lambda, 20

aic, 3, *4*
aic.dmm, 3
availability, 5

bic, 6, *8*
bic.dmm, 7

distributions, 8, *14, 15*
drimmR *(*drimmR-package*)*, 2
drimmR-package, 2

failureRate, 9
fitdmm, *4*, *6*, *8, 9*, *11*, 11, *12*, *15*, *17*, *19*, *21*, *23, 24*, *26, 27*, *29–31*

getDistribution, *9*, 13, *17*
getDistribution.dmm, 14
getStationaryLaw, *9*, *15*, 16, *27*, *29*
getStationaryLaw.dmm, 17
getTransitionMatrix, *4*, *6*, *8*, *11*, *15*, *17*, 18, *21*, *23, 24*, *26, 27*, *29–31*
getTransitionMatrix.dmm, 19

lambda, 20
lengthWord_probabilities, 20
loglik, *4*, *8*, 21
loglik.dmm, 22

maintainability, *6*, 23

reliability, *6*, *11*, 24

simulate, 26
simulate.dmm, 27
stationary_distributions, *17*, 28

word_probabilities, *29*, 30, *31*
word_probability, *21*, *29, 30*, 31
words_probabilities, 29, *30, 31*