Package: LaMa (via r-universe)

November 2, 2024

Type Package

Title Fast Numerical Maximum Likelihood Estimation for Latent Markov Models

Version 1.0.0

Description The class of latent Markov models, including hidden Markov models, hidden semi-Markov models, state space models, and point processes, is a very popular and powerful framework for inference of time series driven by latent processes. Furthermore, all these models can be fitted using direct numerical maximum likelihood estimation using the so-called forward algorithm as discussed in Zucchini et al. (2016) <doi:10.1201/b20790>. However, due to their great flexibility, researchers using these models in applied work often need to build highly customized models for which standard software implementation is lacking, or the construction of such models in said software is as complicated as writing fully tailored 'R' code. While providing greater flexibility and control, the latter suffers from slow estimation speeds that make custom solutions inconvenient. We address the above issues in two ways. First, standard blocks of code, common to all these model classes, are implemented as simple-to-use functions that can be added like Lego blocks to an otherwise fully custom likelihood function, making writing custom code much easier. Second, under the hood, these functions are written in 'C++', allowing for 10-20 times faster evaluation time, and thus drastically speeding up model estimation. To aid in building fully custom likelihood functions, several vignettes are included that show how to simulate data from and estimate all the above model classes.

URL `https://janoleko.github.io/software/`, `https://github.com/janoleko/LaMa`

License GPL-3

Encoding UTF-8

Imports Rcpp, mgcv

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.5.0)

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0), PHSMM

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Jan-Ole Koslik [aut, cre]
(<https://orcid.org/0009-0004-1556-9053>)

Maintainer Jan-Ole Koslik <jan-ole.koslik@uni-bielefeld.de>

Date/Publication 2024-06-04 09:47:46 UTC

Repository https://janoleko.r-universe.dev

RemoteUrl https://github.com/cran/LaMa

RemoteRef HEAD

RemoteSha 80ddd682d7d7d8df027e8032099f0184fcf93557

Contents

`calc_trackInd` *Calculate the index of the first observation of each track based on an ID variable*

Description

Function to conveniently calculate the trackInd variable that is needed when fitting a model to longitudinal data with multiple tracks.

Usage

```
calc_trackInd(ID)
```

Arguments

ID	ID variable of track IDs that is of the same length as the data to be analyzed

Details

Preferably, this function should not be used inside the likelihood function, as it may slow down the computation speed. Instead, it can be called once and the result can then be passed as an argument to the likelihood function.

Value

A vector of indices of the first observation of each track which can be passed to the forward and forward_g to sum likelihood contributions of each track

Examples

```
uniqueID = c("Animal1", "Animal2", "Animal3")
ID = rep(uniqueID, c(100, 200, 300))
trackInd = calc_trackInd(ID)
```

`forward` R*hrefhttps://www.taylorfrancis.com/books/mono/10.1201/b20790/hidden-markov-models-time-series-walter-zucchini-iain-macdonald-roland-langrockForward algorithm with homogeneous transition probability matrix*

Description

Forward algorithm with homogeneous transition probability matrix

Usage

```
forward(delta, Gamma, allprobs, trackInd = NULL)
```

Arguments

`delta`	Initial or stationary distribution of length N, or matrix of dimension c(k,N) for k independent tracks, if trackInd is provided
`Gamma`	Transition probability matrix of dimension c(N,N), or array of k transition probability matrices of dimension c(N,N,k), if trackInd is provided.
`allprobs`	Matrix of state-dependent probabilities/ density values of dimension c(n, N)
`trackInd`	Optional vector of length k containing the indices that correspond to the beginning of separate tracks. If provided, the total log-likelihood will be the sum of each track's likelihood contribution. In this case, Gamma can be a matrix, leading to the same transition probabilities for each track, or an array of dimension c(N,N,k), with one (homogeneous) transition probability matrix for each track. Furthermore, instead of a single vector delta corresponding to the initial distribution, a delta matrix of initial distributions, of dimension c(k,N), can be provided, such that each track starts with it's own initial distribution.

Value

Log-likelihood for given data and parameters

Examples

```
## generating data from homogeneous 2-state HMM
mu = c(0, 6)
sigma = c(2, 4)
Gamma = matrix(c(0.5, 0.05, 0.15, 0.85), nrow = 2, byrow = TRUE)
delta = c(0.5, 0.5)
# simulation
s = x = rep(NA, 500)
s[1] = sample(1:2, 1, prob = delta)
x[1] = rnorm(1, mu[s[1]], sigma[s[1]])
for(t in 2:500){
  s[t] = sample(1:2, 1, prob = Gamma[s[t-1],])
  x[t] = rnorm(1, mu[s[t]], sigma[s[t]])
}

## negative log likelihood function
mllk = function(theta.star, x){
  # parameter transformations for unconstraint optimization
  Gamma = tpm(theta.star[1:2])
  delta = stationary(Gamma) # stationary HMM
  mu = theta.star[3:4]
  sigma = exp(theta.star[5:6])
  # calculate all state-dependent probabilities
  allprobs = matrix(1, length(x), 2)
  for(j in 1:2){ allprobs[,j] = dnorm(x, mu[j], sigma[j]) }
  # return negative for minimization
  -forward(delta, Gamma, allprobs)
}

## fitting an HMM to the data
```

```
theta.star = c(-2,-2,0,5,log(2),log(3))
mod = stats::nlm(mllk, theta.star, x = x)
```

---

forward_g *General* R*hrefhttps://www.taylorfrancis.com/books/mono/10.1201/b20790/hidden-markov-models-time-series-walter-zucchini-iain-macdonald-roland-langrockforward algorithm with time-varying transition probability matrix*

Description

General forward algorithm with time-varying transition probability matrix

Usage

```
forward_g(delta, Gamma, allprobs, trackInd = NULL)
```

Arguments

`delta` Initial distribution of length N, or matrix of dimension c(k,N) for k independent tracks, if trackInd is provided.

`Gamma` Array of transition probability matrices of dimension c(N,N,n-1), as in a time series of length n, there are only n-1 transitions. If you provide an array of dimension c(N,N,n), the first slice will be ignored.

If the elements of $\Gamma^{(t)}$ depend on covariate values at t or covariates t+1 is your choice in the calculation of the array, prior to using this function. When conducting the calculation by using tpm_g(), the choice comes down to including the covariate matrix Z[-1,] oder Z[-n,].

If trackInd is provided, Gamma needs to be an array of dimension c(N,N,n), matching the number of rows of allprobs. For each track, the transition matrix at the beginning will be ignored. If the parameters for Gamma are pooled across tracks or not, depends on your calculation of Gamma. If pooled, you can use tpm_g(Z, beta) to calculate the entire array of transition matrices when Z is of dimension c(n,p).

This function can also be used to fit continuous-time HMMs, where each array entry is the Markov semigroup $\Gamma(\Delta t) = \exp(Q \Delta t)$ and Q is the generator of the continuous-time Markov chain.

`allprobs` Matrix of state-dependent probabilities/ density values of dimension c(n, N)

`trackInd` Optional vector of length k containing the indices that correspond to the beginning of separate tracks. If provided, the total log-likelihood will be the sum of each track's likelihood contribution. In this case, Gamma needs to be an array of dimension c(N,N,n), matching the number of rows of allprobs. For each track, the transition matrix at the beginning of the track will be ignored (as there is no transition between tracks). Furthermore, instead of a single vector delta corresponding to the initial distribution, a delta matrix of initial distributions, of dimension c(k,N), can be provided, such that each track starts with it's own initial distribution.

Value

Log-likelihood for given data and parameters

Examples

```
## generating data from inhomogeneous 2-state HMM
mu = c(0, 6)
sigma = c(2, 4)
beta = matrix(c(-2,-2,0.5,-0.5),nrow=2)
delta = c(0.5, 0.5)
# simulation
n = 2000
s = x = rep(NA, n)
z = rnorm(n, 0, 2)
s[1] = sample(1:2, 1, prob = delta)
x[1] = rnorm(1, mu[s[1]], sigma[s[1]])
for(t in 2:n){
  Gamma = diag(2)
  Gamma[!Gamma] = exp(beta[,1]+beta[,2]*z[t])
  Gamma = Gamma / rowSums(Gamma)
  s[t] = sample(1:2, 1, prob = Gamma[s[t-1],])
  x[t] = rnorm(1, mu[s[t]], sigma[s[t]])
}

## negative log likelihood function
mllk = function(theta.star, x, z){
  # parameter transformations for unconstraint optimization
  beta = matrix(theta.star[1:4], 2, 2)
  Gamma = tpm_g(Z = z, beta = beta)
  delta = c(plogis(theta.star[5]), 1-plogis(theta.star[5]))
  mu = theta.star[6:7]
  sigma = exp(theta.star[8:9])
  # calculate all state-dependent probabilities
  allprobs = matrix(1, length(x), 2)
  for(j in 1:2){ allprobs[,j] = dnorm(x, mu[j], sigma[j]) }
  # return negative for minimization
  -forward_g(delta, Gamma, allprobs)
}

## fitting an HMM to the data
theta.star = c(-2,-2,1,-1,0,0,5,log(2),log(3))
mod = nlm(mllk, theta.star, x = x, z = z)
```

forward_p — *Rhrefhttps://www.taylorfrancis.com/books/mono/10.1201/b20790/hidden-markov-models-time-series-walter-zucchini-iain-macdonald-roland-langrockForward algorithm with (only) periodically varying transition probability matrix*

Description

When the transition probability matrix only varies periodically (e.g. as a function of time of day), there are only L unique matrices if L is the period length (e.g. $L = 24$ for hourly data and time-of-day variation). Thus, it is much more efficient to only calculate these L matrices and index them by a time variable (e.g. time of day or day of year) instead of calculating such a matrix for each index in the data set (which would be redundant). This function allows for that, by only expecting a transition probability matrix for each time point in a period, and an integer valued $(1, \dots, L)$ time variable that maps the data index to the according time.

Usage

```
forward_p(delta, Gamma, allprobs, tod)
```

Arguments

`delta`	Initial or periodically stationary distribution of length N
`Gamma`	Array of transition probability matrices of dimension c(N,N,L). Here we use the definition $\Pr(S_t = j \mid S_{t-1} = i) = \gamma_{ij}^{(t)}$ such that the transition probabilities between time point $t-1$ and t are an element of $\Gamma^{(t)}$.
`allprobs`	Matrix of state-dependent probabilities/ density values of dimension c(n, N)
`tod`	(Integer valued) time variable in 1, ..., L, mapping the data index to a generalized time of day (length n). For half-hourly data L = 48. It could, however, also be day of year for daily data and L = 365.

Value

Log-likelihood for given data and parameters

Examples

```
## generating data from periodic 2-state HMM
mu = c(0, 6)
sigma = c(2, 4)
beta = matrix(c(-2,-2,1,-1, 1, -1),nrow=2)
delta = c(0.5, 0.5)
# simulation
n = 2000
s = x = rep(NA, n)
tod = rep(1:24, ceiling(2000/24))
```

```
s[1] = sample(1:2, 1, prob = delta)
x[1] = rnorm(1, mu[s[1]], sigma[s[1]])
# 24 unique t.p.m.s
Gamma = array(dim = c(2,2,24))
for(t in 1:24){
  G = diag(2)
  G[!G] = exp(beta[,1]+beta[,2]*sin(2*pi*t/24)+
    beta[,3]*cos(2*pi*t/24)) # trigonometric link
  Gamma[,,t] = G / rowSums(G)
}
for(t in 2:n){
  s[t] = sample(1:2, 1, prob = Gamma[s[t-1],,tod[t]])
  x[t] = rnorm(1, mu[s[t]], sigma[s[t]])
}
# we can also use function from LaMa to make building periodic tpms much easier
Gamma = tpm_p(1:24, 24, beta, degree = 1)

## negative log likelihood function
mllk = function(theta.star, x, tod){
  # parameter transformations for unconstraint optimization
  beta = matrix(theta.star[1:6], 2, 3)
  Gamma = tpm_p(tod=tod, L=24, beta=beta)
  delta = stationary_p(Gamma, t=tod[1])
  mu = theta.star[8:9]
  sigma = exp(theta.star[10:11])
  # calculate all state-dependent probabilities
  allprobs = matrix(1, length(x), 2)
  for(j in 1:2){ allprobs[,j] = dnorm(x, mu[j], sigma[j]) }
  # return negative for minimization
  -forward_p(delta, Gamma, allprobs, tod)
}

## fitting an HMM to the data
theta.star = c(-2,-2,1,-1,1,-1,0,0,5,log(2),log(3))
mod = nlm(mllk, theta.star, x = x, tod = tod)
```

---

forward_s   R*hrefhttps://www.taylorfrancis.com/books/mono/10.1201/b20790/hidden-markov-models-time-series-walter-zucchini-iain-macdonald-roland-langrockForward algorithm for hidden semi-Markov models with homogeneous transition probability matrix*

Description

Hidden semi-Markov models (HSMMs) are a flexible extension of HMMs. For direct numerical maximum likelihood estimation, HSMMs can be represented as HMMs on an enlarged state space (of size M) and with structured transition probabilities.

Usage

```
forward_s(delta, Gamma, allprobs, sizes)
```

Arguments

delta	Initial or stationary distribution of length M (where M is the number of approximating states)
Gamma	Transition probability matrix of dimension c(M,M)
allprobs	Matrix of state-dependent probabilities/ density values of dimension c(n, N), where N is the number of semi-Markovian states. This will automatically be converted to the appropriate dimension.
sizes	State aggregate sizes that are used for the approximation of the semi-Markov chain.

Value

Log-likelihood for given data and parameters

Examples

```
## generating data from homogeneous 2-state HSMM
mu = c(0, 6)
lambda = c(6, 12)
omega = matrix(c(0,1,1,0), nrow = 2, byrow = TRUE)
# simulation
# for a 2-state HSMM the embedded chain always alternates between 1 and 2
s = rep(1:2, 100)
C = x = numeric(0)
for(t in 1:100){
  dt = rpois(1, lambda[s[t]])+1 # shifted Poisson
  C = c(C, rep(s[t], dt))
  x = c(x, rnorm(dt, mu[s[t]], 1.5)) # fixed sd 2 for both states
}

## negative log likelihood function
mllk = function(theta.star, x, sizes){
  # parameter transformations for unconstraint optimization
  omega = matrix(c(0,1,1,0), nrow = 2, byrow = TRUE) # omega fixed (2-states)
  lambda = exp(theta.star[1:2]) # dwell time means
  dm = list(dpois(1:sizes[1]-1, lambda[1]), dpois(1:sizes[2]-1, lambda[2]))
  Gamma = tpm_hsmm(omega, dm)
  delta = stationary(Gamma) # stationary
  mu = theta.star[3:4]
  sigma = exp(theta.star[5:6])
  # calculate all state-dependent probabilities
  allprobs = matrix(1, length(x), 2)
  for(j in 1:2){ allprobs[,j] = dnorm(x, mu[j], sigma[j]) }
  # return negative for minimization
  -forward_s(delta, Gamma, allprobs, sizes)
}
```

```
## fitting an HSMM to the data
theta.star = c(log(5), log(10), 1, 4, log(2), log(2))
mod = nlm(mllk, theta.star, x = x, sizes = c(20, 30), stepmax = 5)
```

`forward_sp` *Rhrefhttps://www.taylorfrancis.com/books/mono/10.1201/b20790/hidden-markov-models-time-series-walter-zucchini-iain-macdonald-roland-langrockForward algorithm for hidden semi-Markov models with periodically varying transition probability matrices*

Description

Hidden semi-Markov models (HSMMs) are a flexible extension of HMMs. For direct numerical maximum likelihood estimation, HSMMs can be represented as HMMs on an enlarged state space (of size M) and with structured transition probabilities such that approximate inference is possible. Recently, this inference procedure has been generalized to allow either the dwell-time distributions or the conditional transition probabilities to depend on external covariates such as the time of day. This special case is implemented here. This function allows for that, by expecting a transition probability matrix for each time point in a period, and an integer valued $(1, \dots, L)$ time variable that maps the data index to the according time.

Usage

```
forward_sp(delta, Gamma, allprobs, sizes, tod)
```

Arguments

`delta`	Initial or periodically stationary distribution of length M (where M is the number of approximating states)
`Gamma`	Array of transition probability matrices of dimension c(M,M,L). Here we use the definition $\Pr(S_t = j \mid S_{t-1} = i) = \gamma_{ij}^{(t)}$ such that the transition probabilities between time point $t-1$ and t are an element of $\Gamma^{(t)}$.
`allprobs`	Matrix of state-dependent probabilities/ density values of dimension c(n, N), where N is the number of semi-Markovian states. This will automatically be converted to the appropriate dimension.
`sizes`	State aggregate sizes that are used for the approximation of the semi-Markov chain.
`tod`	(Integer valued) time variable in 1, ..., L, mapping the data index to a generalized time of day (length n). For half-hourly data L = 48. It could, however, also be day of year for daily data and L = 365.

Value

Log-likelihood for given data and parameters

Examples

```
## generating data from homogeneous 2-state HSMM
mu = c(0, 6)
beta = matrix(c(log(4),log(6),-0.2,0.2,-0.1,0.4), nrow=2)
# time varying mean dwell time
Lambda = exp(cbind(1, trigBasisExp(1:24, 24, 1))%*%t(beta))
omega = matrix(c(0,1,1,0), nrow = 2, byrow = TRUE)
# simulation
# for a 2-state HSMM the embedded chain always alternates between 1 and 2
s = rep(1:2, 100)
C = x = numeric(0)
tod = rep(1:24, 50) # time of day variable
time = 1
for(t in 1:100){
  dt = rpois(1, Lambda[tod[time], s[t]])+1 # dwell time depending on time of day
  time = time + dt
  C = c(C, rep(s[t], dt))
  x = c(x, rnorm(dt, mu[s[t]], 1.5)) # fixed sd 2 for both states
}
tod = tod[1:length(x)]

## negative log likelihood function
mllk = function(theta.star, x, sizes, tod){
  # parameter transformations for unconstraint optimization
  omega = matrix(c(0,1,1,0), nrow = 2, byrow = TRUE) # omega fixed (2-states)
  mu = theta.star[1:2]
  sigma = exp(theta.star[3:4])
  beta = matrix(theta.star[5:10], nrow=2)
  # time varying mean dwell time
  Lambda = exp(cbind(1, trigBasisExp(1:24, 24, 1))%*%t(beta))
  dm = list()
  for(j in 1:2){
    dm[[j]] = sapply(1:sizes[j]-1, dpois, lambda = Lambda[,j])
  }
  Gamma = tpm_phsmm(omega, dm)
  delta = stationary_p(Gamma, tod[1])
  # calculate all state-dependent probabilities
  allprobs = matrix(1, length(x), 2)
  for(j in 1:2){ allprobs[,j] = dnorm(x, mu[j], sigma[j]) }
  # return negative for minimization
  -forward_sp(delta, Gamma, allprobs, sizes, tod)
}

## fitting an HSMM to the data
theta.star = c(1, 4, log(2), log(2), # state-dependent parameters
                 log(4), log(6), rep(0,4)) # state process parameters dm
mod = nlm(mllk, theta.star, x = x, sizes = c(10, 15), tod = tod, stepmax = 5)
```

`stateprobs` *Calculate conditional local state probabilities for homogeneous HMMs*

Description

Computes

$\Pr(S_t = j \mid X_1, ..., X_T)$

Usage

```
stateprobs(delta, Gamma, allprobs)
```

Arguments

`delta`	Initial or stationary distribution of length N
`Gamma`	Transition probability matrix of dimension c(N,N)
`allprobs`	Matrix of state-dependent probabilities/ density values of dimension c(n, N)

Value

Matrix of conditional state probabilities of dimension c(n,N)

Examples

```
Gamma = tpm(c(-1,-2))
delta = stationary(Gamma)
allprobs = matrix(runif(200), nrow = 100, ncol = 2)

probs = stateprobs(delta, Gamma, allprobs)
```

`stateprobs_g` *Calculate conditional local state probabilities for inhomogeneous HMMs*

Description

Computes

$\Pr(S_t = j \mid X_1, ..., X_T)$

Usage

```
stateprobs_g(delta, Gamma, allprobs)
```

Arguments

- `delta` Initial distribution of length N
- `Gamma` Array of transition probability matrices of dimension c(N,N,n-1), as in a time series of length n, there are only n-1 transitions.
- `allprobs` Matrix of state-dependent probabilities/ density values of dimension c(n, N)

Value

Matrix of conditional state probabilities of dimension c(n,N)

Examples

```
Gamma = tpm_g(runif(99), matrix(c(-1,-1,1,-2), nrow = 2, byrow = TRUE))
delta = c(0.5, 0.5)
allprobs = matrix(runif(200), nrow = 100, ncol = 2)

probs = stateprobs_g(delta, Gamma, allprobs)
```

`stateprobs_p` *Calculate conditional local state probabilities for periodically inhomogeneous HMMs*

Description

Computes

$\Pr(S_t = j \mid X_1, ..., X_T)$

Usage

```
stateprobs_p(delta, Gamma, allprobs, tod)
```

Arguments

- `delta` Initial or periodically stationary distribution of length N
- `Gamma` Array of transition probability matrices of dimension c(N,N,L) where L is the cycle length.

  Here we use the definition $\Pr(S_t = j \mid S_{t-1} = i) = \gamma_{ij}^{(t)}$ such that the transition probabilities between time point $t-1$ and t are an element of $\Gamma^{(t)}$.
- `allprobs` Matrix of state-dependent probabilities/ density values of dimension c(n, N)
- `tod` (Integer valued) time variable in 1, ..., L, mapping the data index to a generalized time of day (length n). For half-hourly data L = 48. It could, however, also be day of year for daily data and L = 365.

Value

Matrix of conditional state probabilities of dimension c(n,N)

Examples

```
L = 24
beta = matrix(c(-1, 2, -1, -2, 1, -1), nrow = 2, byrow = TRUE)
Gamma = tpm_p(1:L, L, beta, degree = 1)
delta = stationary_p(Gamma, 1)
allprobs = matrix(runif(200), nrow = 100, ncol = 2)
tod = rep(1:24, 5)[1:100]

probs = stateprobs_p(delta, Gamma, allprobs, tod)
```

---

stationary — *Compute the stationary distribution of a homogeneous Markov chain*

Description

A homogeneous, finite state Markov chain that is irreducible and aperiodic converges to a unique stationary distribution, here called δ. As it is stationary, this distribution satisfies

$\delta\Gamma = \delta$, subject to $\sum_{j=1}^{N} \delta_j = 1$,

where Γ is the transition probability matrix. This function solves the linear system of equations above.

Usage

```
stationary(Gamma, tol = .Machine$double.eps)
```

Arguments

Gamma	Transition probability matrix of dimension c(N,N)
tol	The tolerance for detecting linear dependencies in the columns of Gamma. The default is .Machine$double.eps.

Value

Stationary distribution of the Markov chain with the given transition probability matrix

Examples

```
Gamma = tpm(c(rep(-2,3), rep(-3,3)))
delta = stationary(Gamma)
```

`stationary_p` *Compute the periodically stationary distribution of a periodically inhomogeneous Markov chain*

Description

If the transition probability matrix of an inhomogeneous Markov chain varies only periodically (with period length L), it converges to a so-called periodically stationary distribution. This happens, because the thinned Markov chain, which has a full cycle as each time step, has homogeneous transition probability matrix

$\Gamma_t = \Gamma^{(t)}\Gamma^{(t+1)}\dots\Gamma^{(t+L-1)}$ for all $t = 1, \dots, L$.

The stationary distribution for time t satifies $\delta^{(t)}\Gamma_t = \delta^{(t)}$.
This function calculates the periodically stationary distribution.

Usage

```
stationary_p(Gamma, t = NULL, tol = .Machine$double.eps)
```

Arguments

`Gamma`	Array of transition probability matrices of dimension c(N,N,L).
`t`	Integer index of the time point in the cycle, for which to calculate the stationary distribution If t is not provided, the function calculates all stationary distributions for each time point in the cycle.
`tol`	The tolerance for detecting linear dependencies in the columns of the thinned transition matrix. The default is .Machine$double.eps.

Value

Either the periodically stationary distribution at time t or all periodically stationary distributions.

Examples

```
L = 24
beta = matrix(c(-1, 2, -1, -2, 1, -1), nrow = 2, byrow = TRUE)
Gamma = tpm_p(1:L, L, beta, degree = 1)
# Periodically stationary distribution for specific time point
delta = stationary_p(Gamma, 4)

# All periodically stationary distributions
Delta = stationary_p(Gamma)
```

tpm	*Build the transition probability matrix from unconstraint parameter vector*

Description

This function builds the transition probability matrix from an unconstraint parameter vector. For each row of the matrix, the inverse multinomial logistic link is applied.

Usage

```
tpm(param, byrow = FALSE)
```

Arguments

param	Unconstraint parameter vector of length N*(N-1) where N is the number of states of the Markov chain
byrow	Logical that indicates if the transition probability matrix should be filled by row. Defaults to FALSE, but should be set to TRUE if one wants to work with a matrix of beta parameters returned by popular HMM packages like moveHMM, momentuHMM, or hmmTMB.

Value

Transition probability matrix of dimension c(N,N)

Examples

```
# 2 states: 2 free off-diagonal elements
param1 = rep(-1, 2)
Gamma1 = tpm(param1)

# 3 states: 6 free off-diagonal elements
param2 = rep(-2, 6)
Gamma2 = tpm(param2)
```

tpm_cont	*Calculation of continuous time transition probabilities*

Description

A continuous-time Markov chain is described by an infinitesimal generator matrix Q. When observing data at time points $t_1, \ldots, t_n$ the transition probabilites between t_i and t_{i+1} are caluclated as

$$\Gamma(\Delta t_i) = \exp(Q \Delta t_i),$$

where $\exp()$ is the matrix exponential. The mapping $\Gamma(\Delta t)$ is also called the Markov semigroup. This function calculates all transition matrices based on a given generator and time differences.

Usage

```
tpm_cont(Q, timediff)
```

Arguments

Q	Infinitesimal generator matrix of the continuous-time Markov chain of dimension c(N,N)
timediff	Time differences between observations of length n-1 when based on n observations

Value

An array of transition matrices of dimension c(N,N,n-1)

Examples

```
# building a Q matrix for a 3-state cont.-time Markov chain
Q = diag(3)
Q[!Q] = rexp(6)
diag(Q) = 0
diag(Q) = - rowSums(Q)

# draw time differences
timediff = rexp(1000, 10)

Gamma = tpm_cont(Q, timediff)
```

---

tpm_g — *Build all transition probability matrices of an inhomogeneous HMM*

Description

In an HMM, we can model the influence of covariates on the state process, by linking them to the transition probabiltiy matrix. Most commonly, this is done by specifying a linear predictor

$$\eta_{ij}^{(t)} = \beta_0^{(ij)} + \beta_1^{(ij)} z_{t1} + \dots + \beta_p^{(ij)} z_{tp}$$

for each off-diagonal element ($i \neq j$) and then applying the inverse multinomial logistic link to each row. This function efficiently calculates all transition probabilty matrices for a given design matrix Z and parameter matrix beta.

Usage

```
tpm_g(Z, beta, byrow = FALSE)
```

Arguments

`Z`	Covariate design matrix (excluding intercept column) of dimension c(n, p), where p can also be one (i.e. Z can be a vector).
`beta`	Matrix of coefficients for the off-diagonal elements of the transition probability matrix. Needs to be of dimension c(N*(N-1), p+1), where the first column contains the intercepts.
`byrow`	Logical that indicates if each transition probability matrix should be filled by row. Defaults to FALSE, but should be set to TRUE if one wants to work with a matrix of beta parameters returned by popular HMM packages like moveHMM, momentuHMM, or hmmTMB.

Value

Array of transition probability matrices of dimension c(N,N,n)

Examples

```
n = 1000
Z = matrix(runif(n*2), ncol = 2)
beta = matrix(c(-1, 1, 2, -2, 1, -2), nrow = 2, byrow = TRUE)
Gamma = tpm_g(Z, beta)
```

`tpm_hsmm` *Build the transition probability matrix of an HSMM-approximating HMM*

Description

Hidden semi-Markov models (HSMMs) are a flexible extension of HMMs. For direct numerical maximum likelihood estimation, HSMMs can be represented as HMMs on an enlarged state space (of size M) and with structured transition probabilities. This function computes the transition matrix of an HSMM.

Usage

```
tpm_hsmm(omega, dm, eps = 1e-10)
```

Arguments

`omega`	Embedded transition probability matrix of dimension c(N,N)
`dm`	State dwell-time distributions arranged in a list of length(N). Each list element needs to be a vector of length N_i, where N_i is the state aggregate size.
`eps`	Rounding value: If an entry of the transition probabily matrix is smaller, than it is rounded to zero.

Value

The extended-state-space transition probability matrix of the approximating HMM

Examples

```
# building the t.p.m. of the embedded Markov chain
omega = matrix(c(0,1,1,0), nrow = 2, byrow = TRUE)
# defining state aggregate sizes
sizes = c(20, 30)
# defining state dwell-time distributions
lambda = c(5, 11)
dm = list(dpois(1:sizes[1]-1, lambda[1]), dpois(1:sizes[2]-1, lambda[2]))
# calculating extended-state-space t.p.m.
Gamma = tpm_hsmm(omega, dm)
```

`tpm_p` *Build all transition probability matrices of a periodically inhomogeneous HMM*

Description

Given a periodically varying variable such as time of day or day of year and the associated cycle length, this function calculates the transition probability matrices by applying the inverse multinomial logistic link to linear predictors of the form

$\eta_{ij}^{(t)} = \beta_0^{(ij)} + \sum_{k=1}^K \big(\beta_{1k}^{(ij)} \sin(\frac{2\pi kt}{L}) + \beta_{2k}^{(ij)} \cos(\frac{2\pi kt}{L})\big)$

for the off-diagonal elements ($i \neq j$). This is relevant for modeling e.g. diurnal variation and the flexibility can be increased by adding smaller frequencies (i.e. increasing K).

Usage

```
tpm_p(tod = 1:24, L = 24, beta, degree = 1, Z = NULL, byrow = FALSE)
```

Arguments

`tod` Equidistant (generalized) time of day sequence, denoting the time point in a cycle. For time of day and e.g. half-hourly data, this could be 1, ..., L and L = 48, or 0.5, 1, 1.5, ..., 24 and L = 24.

`L` Length of one full cycle, on the scale of tod

`beta` Matrix of coefficients for the off-diagonal elements of the transition probability matrix. Needs to be of dimension c(N*(N-1), 2*degree+1), where the first column contains the intercepts.

`degree` Degree of the trigonometric link function. For each additional degree, one sine and one cosine frequency are added.

Z
: Pre-calculated design matrix (excluding intercept column). Defaults to NULL if trigonometric link should be calculated. From an efficiency perspective, Z should be pre-calculated within the likelhood function, as the basis expansion should not be redundantly calculated. This can be done by using trigBasisExpansion().

  Furthermore, Z can also be a pre-calculated design matrix from mgcv::cSplineDes() (with p columns), when one wants to use cyclic P-splines, or it can be any other basis expansion of the cyclic variable. In that case, the dimension of beta needs to be c(N*(N-1), p+1) and a penalty term should be added at the end of the negative log-likelihood.

byrow
: Logical that indicates if each transition probability matrix should be filled by row. Defaults to FALSE, but should be set to TRUE if one wants to work with a matrix of beta parameters returned by popular HMM packages like moveHMM, momentuHMM, or hmmTMB.

Value

Array of transition probability matrices of dimension c(N,N,length(tod))

Examples

```
# hourly data
tod = seq(1, 24, by = 1)
L = 24
beta = matrix(c(-1, 2, -1, -2, 1, -1), nrow = 2, byrow = TRUE)
Gamma = tpm_p(tod, L, beta, degree = 1)

# half-hourly data
## integer tod sequence
tod = seq(1, 48, by = 1)
L = 48
beta = matrix(c(-1, 2, -1, -2, 1, -1), nrow = 2, byrow = TRUE)
Gamma1 = tpm_p(tod, L, beta, degree = 1)

## equivalent specification
tod = seq(0.5, 24, by = 0.5)
L = 24
beta = matrix(c(-1, 2, -1, -2, 1, -1), nrow = 2, byrow = TRUE)
Gamma2 = tpm_p(tod, L, beta, degree = 1)

Gamma1-Gamma2 # same result

# cubic P-splines
set.seed(123)
nk = 8 # number of basis functions
tod = seq(0.5, 24, by = 0.5)
L = 24
k = L * 0:nk / nk # equidistant knots
Z = mgcv::cSplineDes(tod, k) ## cyclic spline design matrix
beta = matrix(c(-1, runif(8, -2, 2), # 9 parameters per off-diagonal element
```

```
                    -2, runif(8, -2, 2)), nrow = 2, byrow = TRUE)
Gamma = tpm_p(tod, L, beta, Z = Z)
```

---

tpm_phsmm *Build all transition probability matrices of an periodic-HSMM-approximating HMM*

Description

Hidden semi-Markov models (HSMMs) are a flexible extension of HMMs. For direct numerical maximum likelihood estimation, HSMMs can be represented as HMMs on an enlarged state space (of size M) and with structured transition probabilities. This function computes the transition matrices of a periodically inhomogeneos HSMMs.

Usage

```
tpm_phsmm(omega, dm, eps = 1e-10)
```

Arguments

omega	Embedded transition probability matrix. Either a matrix of dimension c(N,N) for homogeneous conditional transition probabilities, or an array of dimension c(N,N,L) for inhomogeneous conditional transition probabilities.
dm	State dwell-time distributions arranged in a list of length(N). Each list element needs to be a matrix of dimension c(L, N_i), where each row t is the (approximate) probability mass function of state i at time t.
eps	Rounding value: If an entry of the transition probabily matrix is smaller, than it is rounded to zero.

Value

An array of dimension c(N,N,L), containing the extended-state-space transition probability matrices of the approximating HMM for each time point of the cycle.

Examples

```
N = 3
L = 24
# time-varying mean dwell times
Lambda = exp(matrix(rnorm(L*N, 2, 0.5), nrow = L))
sizes = c(25, 25, 25) # approximating chain with 75 states
# state dwell-time distributions
dm = list()
for(i in 1:3){
  dmi = matrix(nrow = L, ncol = sizes[i])
  for(t in 1:L){
    dmi[t,] = dpois(1:sizes[i]-1, Lambda[t,i])
  }
```

```
  dm[[i]] = dmi
}

## homogeneous conditional transition probabilites
# diagonal elements are zero, rowsums are one
omega = matrix(c(0,0.5,0.5,0.2,0,0.8,0.7,0.3,0), nrow = N, byrow = TRUE)

# calculating extended-state-space t.p.m.s
Gamma = tpm_phsmm(omega, dm)

## inhomogeneous conditional transition probabilites
# omega can be an array
omega = array(rep(omega,L), dim = c(N,N,L))
omega[1,,4] = c(0, 0.2, 0.8) # small change for inhomogeneity

# calculating extended-state-space t.p.m.s
Gamma = tpm_phsmm(omega, dm)
```

`tpm_thinned` *Compute the transition probability matrix of a thinned periodically inhomogeneous Markov chain.*

Description

If the transition probability matrix of an inhomogeneous Markov chain varies only periodically (with period length L), it converges to a so-called periodically stationary distribution. This happens, because the thinned Markov chain, which has a full cycle as each time step, has homogeneous transition probability matrix

$\Gamma_t = \Gamma^{(t)}\Gamma^{(t+1)} \dots \Gamma^{(t+L-1)}$ for all $t = 1, \dots, L$.

This function calculates the matrix above efficiently as a preliminery step to calculating the periodically stationary distribution.

Usage

```
tpm_thinned(Gamma, t)
```

Arguments

Gamma	Array of transition probability matrices of dimension c(N,N,L).
t	Integer index of the time point in the cycle, for which to calculate the thinned transition probility matrix

Value

Thinned transition probabilty matrix of dimension c(N,N)

Examples

```
# setting parameters for trigonometric link
beta = matrix(c(-1, -2, 2, -1, 2, -4), nrow = 2, byrow = TRUE)
# building trigonometric design matrix
Z = cbind(1,trigBasisExp(1:24, 24, 1))
# calculating all 24 linear predictor vectors
Eta = Z%*%t(beta)
# building all 24 t.p.m.s
Gamma = array(dim = c(2,2,24))
for(t in 1:24){
  Gamma[,,t] = tpm(Eta[t,])
}
# calculating
tpm_thinned(Gamma, 4)
```

`trigBasisExp` *Trigonometric Basis Expansion*

Description

Given a periodically varying variable such as time of day or day of year and the associated cycle length, this function performs a basis expansion to efficiently calculate a linear predictor of the form

$\eta^{(t)} = \beta_0 + \sum_{k=1}^{K} \big(\beta_{1k} \sin(\frac{2\pi k t}{L}) + \beta_{2k} \cos(\frac{2\pi k t}{L})\big).$

This is relevant for modeling e.g. diurnal variation and the flexibility can be increased by adding smaller frequencies (i.e. increasing K).

Usage

```
trigBasisExp(tod, L = 24, degree = 1)
```

Arguments

`tod`	Time variable, describing the time point in a cycle. Could for example be time of day (between 0 and 24) or day of year.
`L`	Length of one cycle on the scale of the time variable. For time of day, this would be 24.
`degree`	Degree K of the trigonometric link above. Increasing K increases the flexibility.

Value

A design matrix (without intercept column of ones), ordered as sin1, cos1, sin2, cos2, ...

Examples

```
## hourly data
tod = rep(1:24, 10)
Z = trigBasisExp(tod, L = 24, degree = 2)

## half-hourly data
tod = rep(1:48/2, 10) # in [0,24] -> L = 24
Z1 = trigBasisExp(tod, L = 24, degree = 3)

tod = rep(1:48, 10) # in [1,48] -> L = 48
Z2 = trigBasisExp(tod, L = 48, degree = 3)

Z1 - Z2
# The latter two are equivalent specifications!
```

---

viterbi *Viterbi algorithm for decoding states*

Description

Viterbi algorithm for decoding states

Usage

```
viterbi(delta, Gamma, allprobs)
```

Arguments

delta	Initial or stationary distribution of length N
Gamma	Transition probability matrix of dimension c(N,N)
allprobs	Matrix of state-dependent probabilities/ density values of dimension c(n, N)

Value

Vector of decoded states of length n

Examples

```
delta = c(0.5, 0.5)
Gamma = matrix(c(0.9, 0.1, 0.2, 0.8), nrow = 2, byrow = TRUE)
allprobs = matrix(runif(200), nrow = 100, ncol = 2)
states = viterbi(delta, Gamma, allprobs)
```

viterbi_g — *Viterbi algorithm for decoding states of inhomogeneous HMMs*

Description

Viterbi algorithm for decoding states of inhomogeneous HMMs

Usage

```
viterbi_g(delta, Gamma, allprobs)
```

Arguments

`delta`	Initial distribution of length N
`Gamma`	Array of transition probability matrices of dimension c(N,N,n-1), as in a time series of length n, there are only n-1 transitions. If you provide an array of dimension c(N,N,n), the first slice will be ignored.
`allprobs`	Matrix of state-dependent probabilities/ density values of dimension c(n, N)

Value

Vector of decoded states of length n

Examples

```
delta = c(0.5, 0.5)
Gamma = array(dim = c(2,2,99))
for(t in 1:99){
  gammas = rbeta(2, shape1 = 0.4, shape2 = 1)
  Gamma[,,t] = matrix(c(1-gammas[1], gammas[1],
                        gammas[2], 1-gammas[2]), nrow = 2, byrow = TRUE)
}
allprobs = matrix(runif(200), nrow = 100, ncol = 2)
states = viterbi_g(delta, Gamma, allprobs)
```

viterbi_p — *Viterbi algorithm for decoding states of periodically inhomogeneous HMMs*

Description

Viterbi algorithm for decoding states of periodically inhomogeneous HMMs

Usage

```
viterbi_p(delta, Gamma, allprobs, tod)
```

Arguments

delta	Initial or periodically statioanary distribution of length N
Gamma	Array of transition probability matrices of dimension c(N,N,L), where L is the cycle length.
allprobs	Matrix of state-dependent probabilities/ density values of dimension c(n, N)
tod	Integer valued cyclic variable to index the transition probability matrix.

Value

Vector of decoded states of length n

Examples

```
delta = c(0.5, 0.5)
beta = matrix(c(-2, 1, -1,
                -2, -1, 1), nrow = 2, byrow = TRUE)
Gamma = tpm_p(1:24, 24, beta)

tod = rep(1:24, 10)
n = length(tod)

allprobs = matrix(runif(2*n), nrow = n, ncol = 2)
states = viterbi_p(delta, Gamma, allprobs, tod)
```

Index