

Package: HMMscpg (via r-universe)

October 28, 2026

Type Package

Title Hidden Markov Models and Shifted Compound Poisson-Gamma Renewal Process

Version 1.0.0

Imports Rdpack, methods, stats, utils, Bessel, Rcpp

LinkingTo Rcpp, RcppArmadillo

Description Provides functions to define, simulate and fit hidden Markov models, hidden semi-Markov models, and their time-varying versions, to incompletely observed point process records. Such data arise when some events of an underlying point process are not recorded or are otherwise missing. The models treat the number of missing events between each pair of consecutively observed events of a point process record as a hidden process. The number of missing events may be fixed or modelled as a random variable whose distribution can vary over time. When the number of missing events is fixed, the observed event times can be modelled as a compound gamma renewal process. When it is random, the observed process is represented by a shifted compound Poisson-gamma renewal process, which may also depend on time, leading to a time-dependent shifted compound Poisson-gamma renewal process. These modelling frameworks are described in detail in Shahzadi, Wang, Parry and Bebbington (2023) <[doi:10.1007/s10463-022-00843-5](https://doi.org/10.1007/s10463-022-00843-5)> and Shahzadi, Wang, Parry and Bebbington (2026) <[doi:10.1007/s11634-025-00632-x](https://doi.org/10.1007/s11634-025-00632-x)>. The model parameters are estimated through direct nonlinear numerical optimization of the corresponding likelihood function, and each fit reports whether the change points of its time-dependent components are identified by the data. The package also provides methods for assessing model adequacy through residual analysis. In addition, it calculates and plots the most probable sequences of hidden states, allowing users to examine changes in the underlying missing-event process over time.

RdMacros Rdpack

License GPL-3
Encoding UTF-8
Suggests knitr, rmarkdown, Rmpfr
Config/roxygen2/version 8.1.0
NeedsCompilation yes
Author Amina Shahzadi [aut, cre], Ting Wang [aut]
Maintainer Amina Shahzadi <aminashahzadi@gcu.edu.pk>
Repository <https://cran.r-universe.dev>
Date/Publication 2026-10-28 10:40:15 UTC
RemoteUrl <https://github.com/cran/HMMscpg>
RemoteRef HEAD
RemoteSha 1766f3c5247daf275174439eadeea551d41a130c

Contents

HMMscpg-package	3
dscpg	5
fit.hmm.scpg	8
fit.hsmm.scpg	13
fit.ihmm.scpg	20
fit.ihsmm.scpg	28
fit.scpg	37
hmm.scpg	41
hsmm.scpg	45
ihmm.scpg	49
ihsmm.scpg	54
pscpg	60
qscpg	62
rscpg	65
sim.hmm.scpg	67
sim.hsmm.scpg	69
sim.ihmm.scpg	73
sim.ihsmm.scpg	77
sp.hmm.scpg	82
sp.hsmm.scpg	86
sp.ihmm.scpg	95
sp.ihsmm.scpg	103
vip.hmm.scpg	112
vip.hsmm.scpg	116
vip.ihmm.scpg	122
vip.ihsmm.scpg	129
WBsum	136

Description

Tools for modelling incomplete geological point process records (such as volcanic eruptions or earthquakes) using hidden Markov models (HMMs) whose state-dependent observation distributions are gamma, shifted compound Poisson-gamma (SCPG), or time-dependent SCPG (TSCPG) renewal processes. The package implements four families of models: standard HMMs, inhomogeneous HMMs (IHMMs), hidden semi-Markov models (HSMMs), and inhomogeneous hidden semi-Markov models (IHSMMs), following Shahzadi et al. (2023, 2026).

Details

Consider a point process with event times

$$H_0 < H_1 < \dots < H_t < \dots$$

at jump points $0, 1, 2, \dots, t, \dots$. A renewal process is defined by interevent times $X_t = H_t - H_{t-1}$, $t = 1, 2, \dots$, which are independently and identically distributed, with $H_0 = 0$. Geological records of random events can be modelled using these processes (Wang et al., 2022). Missing events are common in geophysical point process records, and incompleteness varies over time. Given the unknown nature of missing events, Shahzadi et al. (2023, 2026) proposed several types of HMMs in which the unknown number of missing events between each pair of consecutively observed events is characterized by the distribution of the observed process, the underlying Markov chain, or both.

The package employs four types of HMMs, distinguished by the underlying discrete-time Markov chain (MC), which is described by an initial distribution, a transition probability matrix (TPM), and implicit or explicit state durations. The four types HMMs based on underlying hidden Markov chain are:

HMM has a homogeneous Markov chain (MC) with a time-independent transition probability matrix.

IHMM has an inhomogeneous Markov chain (IMC) with a time-dependent transition probability matrix.

HSMM has a semi-Markov chain (SMC) with a zero-diagonal transition probability matrix and explicit state durations, following the Markov property at transition times only.

IHSMM has an inhomogeneous semi-Markov chain (ISMC) with a zero-diagonal transition probability matrix and time-dependent explicit state durations, following the Markov property at transition times only.

For an observation sequence $\{X_t\}$ and hidden state sequence $\{S_t\}$, $t = 1, \dots, T$, with m states, states are defined as:

State 1 denotes completeness: no missing events between each pair of consecutively observed events.

States $2, \dots, m$ represent incompleteness: the number of missing events between each consecutive observed pair.

Each HMM type is further categorized by its observed process as follows:

State 1 In State 1 the observed process follows a gamma renewal process;

In States $2, \dots, m$ the observed process may follow

- (a) a sum of independent gamma renewal processes when N is fixed, or
- (b) an SCPG renewal process when N is shifted-Poisson distributed, or
- (c) a TSCPG renewal process when N is time-dependent.

The table below lists all types of HMMs considered in this package, according to the underlying Markov chain (MC) and the renewal process (RP) as the observed process (OP). In all models, State 1 represents no missing event between each consecutively observed pair; States $2, \dots, m$ represent incompleteness, with the number of missing events N between each consecutive observed pair as given in the last column.

Type of HMM	Type of MC	OP	N
HMM with gamma RP	MC	gamma RP	Fixed
HMM with SCPG RP	MC	SCPG RP	Shifted Poisson distributed
HMM with TSCPG RP	MC	TSCPG RP	Time-dependent shifted Poisson distributed
IHMM with gamma RP	IMC	gamma RP	Fixed
IHMM with SCPG RP	IMC	SCPG RP	Shifted Poisson distributed
IHMM with TSCPG RP	IMC	TSCPG RP	Time-dependent shifted Poisson distributed
HSMM with gamma RP	SMC	gamma RP	Fixed
HSMM with SCPG RP	SMC	SCPG RP	Shifted Poisson distributed
HSMM with TSCPG RP	SMC	TSCPG RP	Time-dependent shifted Poisson distributed
IHSMM with gamma RP	ISMC	gamma RP	Fixed
IHSMM with SCPG RP	ISMC	SCPG RP	Shifted Poisson distributed
IHSMM with TSCPG RP	ISMC	TSCPG RP	Time-dependent shifted Poisson distributed

Here, N denotes the number of missing events between each consecutive observed pair in a point process record.

Author(s)

Maintainer: Amina Shahzadi <aminashahzadi@gmail.com>

Authors:

- Amina Shahzadi <aminashahzadi@gmail.com>
- Ting Wang <ting.wang@otago.ac.nz>

References

Shahzadi A, Wang T, Parry M, Bebbington M (2023). “Inhomogeneous hidden semi-Markov models for incompletely observed point processes.” *Annals of Institute Statistical Mathematics*, **75**, 253-280. doi:[10.1007/s10463022008435](https://doi.org/10.1007/s10463022008435).

Shahzadi A, Wang T, Parry M, Bebbington M (2026). “Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models.” *Advances in Data Analysis and Classification*, **20**, 663-691. doi:10.1007/s1163402500632x.

Wang T, Bebbington M, Cronin S, Carman J (2022). “Forecasting Eruptions at Poorly Known Volcanoes Using Analogs and Multivariate Renewal Processes.” *Geophysical Research Letters*, **49**(12), e2021GL096715. doi:10.1029/2021GL096715.

See Also

[dscpg](#), [pscpg](#), [fit.scpg](#), [WBsum](#)

dscpg

Density of the SCPG and TSCPG distributions

Description

Evaluates the density of the shifted compound Poisson-gamma (SCPG) distribution, of its time-dependent version (TSCPG), or of a gamma distribution with a fixed number of missing events.

Usage

```
dscpg(
  x,
  rate,
  shape,
  scale,
  rate.fix = FALSE,
  shape.fix = FALSE,
  time.scpg = FALSE,
  log = FALSE,
  WBsum.method = "direct.evaluation",
  WBsum.tol = 1e-12,
  max.direct.terms = 2000L
)
```

Arguments

x	A vector of positive, finite interevent times. For the TSCPG (time.scpg = TRUE) they must be in time order.
rate	The number of missing events: for the SCPG, the Poisson mean $\lambda > 0$; for the TSCPG, the vector $c(K, M, R, P)$ of $\lambda(t)$ (see dscpg); with rate.fix = TRUE, the fixed number $n \geq 0$ of missing events.
shape	The gamma shape parameter $\alpha > 0$.
scale	The gamma scale parameter $\beta > 0$.

rate.fix	Logical. If TRUE, exactly rate events are missing between consecutive observed events, and X_t is gamma distributed with shape $(n + 1)\alpha$ and scale β . Default is FALSE.
shape.fix	Logical; used only by <code>fit.scpg</code> , where TRUE holds the shape at the supplied value instead of estimating it. Default is FALSE.
time.scpg	Logical. If TRUE, the TSCPG distribution is used, with $\text{rate} = c(K, M, R, P)$, and <code>rate.fix</code> must be FALSE. Default is FALSE (SCPG).
log	Logical; if TRUE, log densities are returned.
WBsum.method	The method used by <code>WBsum</code> for the series in the density: "direct.evaluation" (the default), "direct.asymptotic" or "asymptotic.expansion". See the paragraph on choosing the method in Details.
WBsum.tol	The tolerance tol passed to <code>WBsum</code> . Default is 10^{-12} .
max.direct.terms	Passed to <code>WBsum</code> ; used only by <code>WBsum.method = "direct.asymptotic"</code> . Default is 2000.

Details

The model. In a complete record the interevent times Y_k are independent gamma variables with shape α and scale β . If N events are missing between two observed events, the observed interevent time is the sum of $N + 1$ of them,

$$X_t = Y_1 + \cdots + Y_{N+1}.$$

Three cases are available.

1. *Fixed number of missing events* (`rate.fix = TRUE`): $N = n$, so X_t is gamma with shape $(n + 1)\alpha$ and scale β ; $n = 0$ means that no event is missing.
2. *SCPG* (the default): at least one event is missing and N is shifted Poisson, $\Pr(N = n) = e^{-\lambda} \lambda^{n-1} / (n - 1)!$ for $n = 1, 2, \dots$, so on average $1 + \lambda$ events are missing. The density is

$$f(x_t) = \exp\left(-\lambda - \frac{x_t}{\beta}\right) \frac{x_t^{2\alpha-1}}{\beta^{2\alpha}} W(z_t; \alpha), \quad z_t = \lambda \left(\frac{x_t}{\beta}\right)^\alpha,$$

where $W(z; \alpha) = \sum_{k \geq 0} z^k / [k! \Gamma\{\alpha(k + 2)\}]$ is the series computed by `WBsum`. The SCPG is the continuous part of the compound Poisson-gamma distribution of the Tweedie family (Tweedie, 1984; Smyth and Jorgensen, 2002), shifted so that at least one event is missing.

3. *TSCPG* (`time.scpg = TRUE`): the Poisson mean changes along the record as a logistic function of the last occurrence time $h_{t-1} = x_1 + \cdots + x_{t-1}$ ($h_0 = 0$),

$$\lambda(t) = K + \frac{M}{1 + \exp(R h_{t-1} - P)}, \quad K, M, R > 0.$$

It decreases from $K + M/(1 + e^{-P})$ at the start of the record (about $K + M$ when P is large) towards K . It is halfway, $\lambda(t) = K + M/2$, at the inflection time $h_{t-1} = P/R$, and R sets the speed of the change: 80 per cent of it takes place within $\log(9)/R \approx 2.2/R$ of P/R . The density of x_t is the SCPG density with $\lambda(t)$ in place of λ , given the preceding observations, so the order of x matters.

Choosing WBSum.method. The cost of $W(z_t; \alpha)$ depends on $n^* = (z_t/\alpha^\alpha)^{1/(1+\alpha)}$, which is about the most probable number of missing events in the interval; the three methods are described in [WBSum](#). The default "direct.evaluation" is accurate for every x_t and fast for the values met in practice: n^* is at most a few hundred in typical records, and 6.5 at the estimates of the [fit.scp](#) example. Use "direct.asymptotic" when dscpg stops with the error "The direct Wright-series evaluation is refused for arguments this large", which happens only when $n^* > 10^{12}$, or to save time when n^* , computed from the largest x_t , exceeds about 10^8 . Such values arise when x_t is very long compared with β , or when λ is very large. They also arise during [fit.scp](#), whose optimizer may try extreme trial values of the shape and scale even though n^* is small at the estimates. A trial value with any $n^* > 10^{12}$ is refused at once and treated as infeasible, but one with many values of n^* between about 10^8 and 10^{12} takes up to seconds per observation. If a fit is slow, use "direct.asymptotic": it gives the same estimates. "asymptotic.expansion" alone is useful only for checking, since it fails for small z_t .

The script `verify_dscpg.R` in the `examples` directory of the package reproduces the numbers stated here; run it in the console with

```
f <- system.file("examples", "verify_dscpg.R",
                 package = "HMMscpg")
source(f, echo = TRUE)
```

Value

A numeric vector of densities, or log densities when `log = TRUE`, with the same length and names as `x`.

References

- Tweedie MCK (1984). "An index which distinguishes between some important exponential families." In *Statistics: Applications and New Directions. Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, 579–604.
- Smyth GK, Jørgensen B (2002). "Fitting Tweedie's compound Poisson model to insurance claims data: dispersion modelling." *ASTIN Bulletin*, **32**(1), 143–157.
- Smyth GK, Jorgensen B (2002). "Fitting Tweedie's compound Poisson model to insurance claims data: Dispersion modelling." *ASTIN Bulletin*, **32**(1), 143–157. doi:10.2143/AST.32.1.1020.
- Wright EM (1935). "The asymptotic expansion of the generalized hypergeometric function." *Journal of the London Mathematical Society*, **10**(4), 286–293.
- Shahzadi A, Wang T, Parry M, Bebbington M (2026). "Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models." *Advances in Data Analysis and Classification*, **20**, 663–691. doi:10.1007/s1163402500632x.

Examples

```
# =====
# Example of an SCPG distribution
# =====
rate <- 3
shape <- 2
scale <- 0.5
```

```

rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from an SCPG distribution

x_scpg <- rscpg(nsim=200, rate, shape, scale, seed=900)
print(x_scpg)
# PDF of an SCPG distribution

pdf_scpg <- dscpg(x_scpg, rate, shape, scale, time.scpg=FALSE,
  WBSum.method="direct.evaluation")
hist(x_scpg, probability = TRUE)
# sorted, so that the density is drawn as a curve
o <- order(x_scpg)
lines(x_scpg[o], pdf_scpg[o], col="blue")
# =====
# Example of a TSCPG distribution
# =====

# rate[1]=K, rate[2]=M, rate[3]=R, rate[4]=P: lambda(t) falls from about K+M
# to K,
# with inflection time P/R = 300
t_rate <- c(1, 2, 0.02, 6)
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from a TSCPG distribution

x_tscpg <- rscpg(nsim=200, t_rate, shape, scale, time.scpg=TRUE, seed=700)
print(x_tscpg)
# PDF of a TSCPG distribution

pdf_tscpg <- dscpg(x_tscpg, t_rate, shape, scale, time.scpg=TRUE,
  WBSum.method="direct.evaluation")
# Each x_t has its own density, at lambda(t) of its occurrence time, so the
# densities are shown as points rather than as one curve
hist(x_tscpg, probability = TRUE)
points(x_tscpg, pdf_tscpg, col="blue", pch=20)

```

fit.hmm.scpg

Fitting an m-state HMM with gamma, SCPG or TSCPG renewal processes

Description

Estimates the parameters of an HMM created by [hmm.scpg](#), of class `hmm.gamma`, `hmm.scpg` or `hmm.tscpg`, by maximum likelihood.

Usage

```
fit.hmm.scpg(
  hmmobj,
  method = "BFGS",
  maxit = 1000,
  reltol = sqrt(.Machine$double.eps),
  negative = TRUE
)
```

Arguments

<code>hmmobj</code>	An object created by <code>hmm.scpg</code> , with the data in <code>hmmobj\$x</code> ; its parameters are the starting values.
<code>method</code>	Optimization method passed to <code>optim</code> . Default is "BFGS".
<code>maxit</code>	Maximum number of iterations passed to the control argument of <code>optim</code> . Default is 1000.
<code>reltol</code>	Relative convergence tolerance passed to the control argument of <code>optim</code> . Default is <code>sqrt(.Machine\$double.eps)</code> .
<code>negative</code>	Logical. If TRUE (default), <code>optim</code> minimizes the negative log-likelihood, otherwise it maximizes the log-likelihood. The estimates are the same either way.

Details

Likelihood. The likelihood sums over all hidden state sequences,

$$L = \sum_{s_1, \dots, s_T} \pi_{s_1} f_{s_1}(x_1) \prod_{t=2}^T a_{s_{t-1}s_t} f_{s_t}(x_t),$$

and is computed by the forward algorithm (Baum et al., 1970),

$$\alpha_1(j) = \pi_j f_j(x_1), \quad \alpha_t(j) = \left\{ \sum_i \alpha_{t-1}(i) a_{ij} \right\} f_j(x_t), \quad L = \sum_j \alpha_T(j),$$

in $O(Tm^2)$ operations. The forward variables are rescaled to sum to one at each step, on the log scale, and $\log L$ is the sum of the logarithms of the scale factors (Zucchini et al., 2016).

Estimation. $\log L$ is maximized with `optim` rather than by the EM algorithm (Dempster et al., 1977), since the SCPG densities give no closed-form M-step. The parameters are mapped to unconstrained working parameters: the initial distribution by logits relative to State m , each row of the transition matrix by logits relative to its diagonal element a_{ii} , and the observation parameters as in `fit.scpg`, with the same bounds on the shape, the Poisson means and, for the TSCPG, $R_j h_{T-1}$, where $h_{T-1} = x_1 + \dots + x_{T-1}$. With `stationary = TRUE` the initial distribution is computed from the transition matrix and not estimated. The optimizer may stop at a local maximum, so several starting values should be tried and the convergence code checked. The densities are computed with `scpg.par$WBsum.method`; the default suits almost all records, and if a fit is slow "direct.asymptotic" gives the same estimates faster (see `dscpg`).

Inflection diagnostics (TSCPG). In State j the Poisson mean $\lambda_j(t)$ falls from about $K_j + M_j$ to K_j , and the central 80 per cent of this change takes place within $2.2/R_j$ of the inflection time P_j/R_j .

The record observes $\lambda_j(t)$ only at $h_0 = 0, h_1, \dots, h_{T-1}$, and only through the observations that the model attributes to State j . The data can therefore locate P_j/R_j only if the change falls inside $[0, h_{T-1}]$ and enough observations of State j fall within it; otherwise the likelihood is nearly flat in P_j/R_j . The fitted object contains the component inflection (see Value), and a warning is given when, for some State j ,

- less than half of the change of $\lambda_j(t)$ takes place on $[0, h_{T-1}]$ (change below 0.5). This covers every P_j/R_j outside $[0, h_{T-1}]$, and a $\lambda_j(t)$ that is almost constant over the record; or
- the expected number of observations of State j within the central 80 per cent of the change is below 10.

These limits come from simulations of the TSCPG with known states (see [fit.scpkg](#)); since the states are hidden here, a fit that meets them may still locate P_j/R_j poorly. After a warning, report P_j/R_j as not identified and compare the model with SCPG states (`time.scpg = FALSE`) by AIC. The script `verify_fit_hmm_scpkg.R` in the examples directory of the package reproduces the statements of this help page; run it in the console with

```
f <- system.file("examples", "verify_fit_hmm_scpkg.R",
                 package = "HMMscpg")
source(f, echo = TRUE)
```

Model selection. The likelihood increases with each state added, so the number of states and the observation process are chosen by $AIC = -2 \log L + 2 \text{ nparam}$ (Akaike, 1974), together with the decoded path ([vip.hmm.scpkg](#)) and the residual process ([sp.hmm.scpkg](#)), as recommended by Pohle et al. (2017).

Value

An object of the same class as `hmmobj`: a list with

`x` NULL; the data are not kept, and must be attached as `x` before [vip.hmm.scpkg](#) or [sp.hmm.scpkg](#) is used.

`mc.par`, `scpg.par` the estimated parameters, in the form of [hmm.scpkg](#), with the settings unchanged.
`logLik` the maximized log-likelihood.

`nparam` the number of estimated parameters.

`convergence`, `message`, `counts` as returned by [optim](#); `convergence = 0` indicates successful convergence.

`objective.error` NULL, or the last reason why the log-likelihood could not be evaluated at a trial value.

`inflection` present only when `time.scpg = TRUE`: a data frame with the inflection diagnostics of Details, one row per State $j = 2, \dots, m$:

`component`, `label` "lambda" and "state j".

`rate`, `intercept` the estimates of R_j and P_j .

`inflection` the inflection time P_j/R_j .

`first`, `last` P_j and $P_j - R_j h_{T-1}$: the position of P_j/R_j relative to 0 and to h_{T-1} , in units of $1/R_j$. It lies inside $[0, h_{T-1}]$ when `first` $\geq 0 \geq$ `last`.

change the fraction, between 0 and 1, of the change of $\lambda_j(t)$ that takes place on $[0, h_{T-1}]$, $|\sigma(P_j) - \sigma(P_j - R_j h_{T-1})|$ with $\sigma(u) = 1/(1 + e^{-u})$; the first limit fails when it is below 0.5.

n the expected number of observations in State j , $\sum_t \Pr(S_t = j \mid x_1, \dots, x_T)$.

in.band the part of n from the x_t whose h_{t-1} lies within the central 80 per cent of the change, $|P_j - R_j h_{t-1}| < \log 9$.

rare always NA here.

flag the limits that are not met, or "".

References

- Baum LE, Petrie T, Soules G, Weiss N (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." *The Annals of Mathematical Statistics*, **41**(1), 164-171. doi:10.1214/aoms/1177697196.
- Zucchini W, MacDonald IL, Langrock R (2016). *Hidden Markov Models for Time Series: An Introduction Using R*, 2 edition. CRC Press, Boca Raton.
- Dempster AP, Laird NM, Rubin DB (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society, Series B*, **39**(1), 1-22. doi:10.1111/j.2517-6161.1977.tb01600.x.
- Akaike H (1974). "A new look at the statistical model identification." *IEEE Transactions on Automatic Control*, **19**(6), 716-723. doi:10.1109/TAC.1974.1100705.
- Pohle J, Langrock R, van Beest FM, Schmidt NM (2017). "Selecting the number of states in hidden Markov models: pragmatic solutions illustrated using animal movement." *Journal of Agricultural, Biological and Environmental Statistics*, **22**(3), 270-293. doi:10.1007/s1325301702838.

Examples

```
# =====
# An example of a 2-state HMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter lambda_2(t), where
#   lambda_2(t) = K + M / (1 + exp(R * h_{t-1} - P))
# falls from about 5.3 at the start of the record towards K = 0.5, around
# the inflection time P/R = 300 of the last occurrence time h_{t-1}. The
# hidden chain is homogeneous, so only lambda_2(t) changes along the record.

m <- 2
tpm <- matrix(c(0.9, 0.1,
                0.2, 0.8), nrow = 2, byrow = TRUE)
mc.par <- list(initd = NULL, tpm = tpm, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5      # K
rate[2, ] <- 5        # M
rate[3, ] <- 0.01     # R
rate[4, ] <- 3        # P: inflection time P/R = 300
shape <- 1
```

```

scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object for 2-state HMM with TSCPG distribution

obj_hmm2 <- hmm.scp(x = NULL, mc.par, scpg.par)
class(obj_hmm2)
# Simulating a 2-state HMM with TSCPG distribution

sim_hmm2 <- sim.hmm.scp(obj_hmm2, nsim = 200, seed = 1)
str(sim_hmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hmm2$x[-200]), h_T = sum(sim_hmm2$x))
# Fitting a 2-state HMM with TSCPG distributions
# (the parameters of obj_hmm2 are the starting values)

fit_hmm2 <- fit.hmm.scp(sim_hmm2)
# transition matrix (true rows 0.9, 0.1 and 0.2, 0.8)
fit_hmm2$mc.par$tpm
fit_hmm2$scpg.par$rate[, 1] # K, M, R, P of lambda_2(t)
fit_hmm2$scpg.par$scale      # true 1
fit_hmm2$logLik
# Inflection diagnostics of the fitted TSCPG rate lambda_2(t), whose true
# inflection time is 300. The change lies inside the record (first > 0 >
# last), almost all of it takes place on [0, h_{T-1}] (change >= 0.5), and
# about 35 expected observations of State 2 lie within its central 80 per
# cent (in.band >= 10), so no warning is given and the inflection time P/R,
# estimated at about 300, is identified.
fit_hmm2$inflection
# =====
# An example of a 3-state HMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 30 seconds on a laptop (the
# 2-state model above takes about 4 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(1/3, 1/3, 1/3)
tpm <- matrix(c(0.2, 0.65, 0.15, 0.50, 0.30, 0.20, 0.25, 0.50, 0.25),
              nrow = 3, byrow = TRUE)
mc.par <- list(initd=NULL, tpm=tpm, stationary=TRUE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(1, 1) # K
rate[2, ] <- c(1.5, 2.5) # M
rate[3, ] <- c(0.02, 0.015) # R
rate[4, ] <- c(3, 3) # P: inflection times P/R = 150 and 200
shape <- 1
scale <- 1.7
scpg.par <- list(rate=rate, shape=shape, scale=scale,

```

```

rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
WBSum.method='direct.evaluation')

# Defining an object for 3-state HMM with TSCPG distribution
obj_hmm2_m3 <- hmm.scpg(x = NULL, mc.par, scpg.par)
# Simulating a 3-state HMM with TSCPG distribution

sim_hmm2_m3 <- sim.hmm.scpg(obj_hmm2_m3, nsim=100, seed=9)
# Fitting a 3-state HMM with TSCPG distributions
# The fit takes about 30 seconds, so it is wrapped in donttest.

fit_hmm2_m3 <- fit.hmm.scpg(sim_hmm2_m3)
print(fit_hmm2_m3)

# Inflection diagnostics of the fitted TSCPG rates lambda_2(t) and
# lambda_3(t) (true inflection times 150 and 200). With 100 observations
# both are flagged here, and the fit warns: lambda_2(t) is fitted as almost
# constant on the record (its inflection time, 413, lies just after
# h_{T-1} = 402, and only 0.2 per cent of its change takes place on
# [0, h_{T-1}]), and lambda_3(t) as an abrupt step at about 99 with no
# expected observation of State 3 within it. Neither inflection time is
# identified; the homogeneous model (time.scpg = FALSE) may be compared by
# AIC.

fit_hmm2_m3$inflection

```

fit.hsmm.scpg

Fitting of an m-state HSMM with gamma or SCPG or TSCPG renewal processes

Description

Estimates the parameters of an HSMM with class `hsmm.gamma`, `hsmm.scpg` and `hsmm.tscpg` as typically returned by [hsmm.scpg](#) using direct nonlinear optimization.

Usage

```

fit.hsmm.scpg(
  hsmmobj,
  method = "BFGS",
  maxit = 1000,
  reltol = sqrt(.Machine$double.eps),
  negative = TRUE
)

```

Arguments

`hsmmobj` is a model object with class `hsmm.gamma`, `hsmm.scpg` and `hsmm.tscpg` as typically returned by [hsmm.scpg](#).

method	Optimization method passed to <code>optim</code> . Default is "BFGS".
maxit	Maximum number of iterations passed to the control argument of <code>optim</code> . Default is 1000.
reltol	Relative convergence tolerance passed to the control argument of <code>optim</code> . Default is <code>sqrt(.Machine\$double.eps)</code> .
negative	is a logical argument for negative log-likelihood. Default is TRUE.

Details

Likelihood. The likelihood of a right-censored HSMM is evaluated by the forward algorithm of Guedon (2003). Let $u_t(j, d) = \prod_{l=t-d+1}^t f_j(x_l)$ denote the joint state-dependent observation probability of a visit to State j of length d ending at time t , and let

$$\alpha_{t+1}^*(j) = \Pr(X_1, \dots, X_t, S_{t+1} = j, S_t \neq j \mid \theta) = \sum_{i \neq j} \alpha_t(i) a_{ij}$$

denote the probability that a visit to State j commences at time $t+1$, with $\alpha_1^*(j) = \pi_j$. The forward variable $\alpha_t(j) = \Pr(X_1, \dots, X_t, S_t = j, S_{t+1} \neq j \mid \theta)$, the t th observation being the last of a visit to State j , satisfies

$$\alpha_t(j) = \sum_{d=1}^t \alpha_{t-d+1}^*(j) p_j(d) u_t(j, d), \quad t = 1, \dots, T-1.$$

The sojourn time in the last visited state is right-censored at T , only its minimum being known, so the sojourn mass function is replaced by the survival function $S_j(d) = \sum_{k \geq d} p_j(k)$,

$$\alpha_T(j) = \sum_{d=1}^T \alpha_{T-d+1}^*(j) S_j(d) u_T(j, d),$$

and $L(\theta) = \sum_{j=1}^m \alpha_T(j)$. Treating the final sojourn as complete would bias the estimated expected sojourn times downwards.

Because the forward variable is indexed by state and elapsed sojourn jointly, the recursion is of order $O(T^2 m^2)$ rather than the $O(T m^2)$ of an HMM, a visit being admissible at any length up to t . The observation products are accumulated by the recursion $u_t(j, d) = u_{t-1}(j, d-1) f_j(x_t)$ of Mitchell and Jamieson (1993). Scaling is embedded in the recursion at each time step and the logarithms of the scale factors accumulated.

Estimation. The likelihood is maximized by direct numerical minimization of the negative log-likelihood using `optim` over working parameters defined on the real line. Each row of the zero-diagonal transition probability matrix is represented by multinomial logits over its $m-1$ transition probabilities, which for $m=2$ leaves the embedded chain fully determined and contributing no parameters; the sojourn parameters are represented by logarithms, or by scaled logits where `dur.lb` and `dur.ub` are supplied; and the observation parameters are transformed as in `fit.scpg`. The forward recursion is evaluated by compiled code. The fitted object is returned on the natural scale and is suitable for `vip.hsmm.scpg` and `sp.hsmm.scpg`.

Inflection diagnostics (TSCPG). These apply when the observation process is a TSCPG (`time.scpg = TRUE`); otherwise the model has no time-dependent component and inflection is NULL. In each state $j = 2, \dots, m$ the rate $\lambda_j(t) = K_j + M_j / \{1 + \exp(R_j h_{t-1} - P_j)\}$ changes around the inflection

time $h_{t-1} = P_j/R_j$. Over the record the last occurrence time h_{t-1} runs from $h_0 = 0$ ($t = 1$) to $h_{T-1} = x_1 + \dots + x_{T-1}$ ($t = T$), so $\lambda_j(t)$ is observed only on $[0, h_{T-1}]$, and P_j/R_j is identified only when this interval covers the change and enough observations of state j fall within it. Writing $u_t = P_j - R_j h_{t-1}$, the sensitivity of $\lambda_j(t)$ to P_j/R_j is at most $R_j e^{-|u_t|}$, so observations more than a few multiples of $1/R_j$ from the inflection time carry almost no information about it; when P_j/R_j lies δ multiples of $1/R_j$ outside $[0, h_{T-1}]$, the profile information decreases like $e^{-4\delta}$, by a factor of about 45 per multiple when the states are known. The state being hidden, only the observations attributed to state j inform the curve, and less than they would if the states were known.

The fitted object contains the component inflection (see Value). Its limits are that at least half of the change of $\lambda_j(t)$ takes place on $[0, h_{T-1}]$ (change ≥ 0.5), which places P_j/R_j inside this interval, and that in.band is at least 10. They were derived from profile likelihoods of P/R in simulations of 250 TSCPG observations (see [fit.scpg](#)). With hidden states they are necessary rather than sufficient.

A warning is issued when a limit is not met. The inflection time should then be reported as not identified rather than by its estimate, and the model with SCPG states (time.scpg = FALSE) compared by AIC. When a fitted $\lambda_j(t)$ is nearly constant on $[0, h_{T-1}]$, P_j/R_j is not identified under the hypothesis of no change and the chi-squared calibration of likelihood-ratio tests and profile intervals fails.

The script `verify_fit_hsmm_scpg.R` in the examples directory of the package reproduces the numbers stated on this help page; run it in the console with

```
f <- system.file("examples", "verify_fit_hsmm_scpg.R",
                 package = "HMMscpg")
source(f, echo = TRUE)
```

Value

The returned object provides estimated HSMM with additional information about the function evaluation and gradient counts at the estimated minimum of negative log-likelihood function, convergence code and minimized likelihood function value as returned by `optim` and nparam is the number of parameters to be estimated. With TSCPG renewal processes it also contains inflection, a data frame with the inflection diagnostics of Details, one row per state $j = 2, \dots, m$:

component, label "lambda" and "state j".

rate, intercept R_j and P_j .

inflection P_j/R_j .

first, last $u_1 = P_j$ and $u_T = P_j - R_j h_{T-1}$, the values of u_t for the first and last interevent times, at $h_0 = 0$ and h_{T-1} . The inflection time lies in $(0, h_{T-1})$ when `first` $> 0 >$ `last`; otherwise `-first` is its distance before 0, or `last` its distance after h_{T-1} , in units of $1/R_j$.

change the part of the change of $\lambda_j(t)$, between 0 and 1, that takes place on $[0, h_{T-1}]$, $|\sigma(u_1) - \sigma(u_T)|$ with $\sigma(u) = 1/(1 + e^{-u})$.

n the expected number of observations in state j , $\sum_{t=1}^T \Pr(S_t = j \mid x)$, from the smoothed state probabilities.

in.band the part of n from the observations x_t whose h_{t-1} lies in the central 10 to 90 per cent of the change, where $|u_t| < \log 9$.

rare NA; used by [fit.ihmm.scpg](#).

flag the limits of Details that are not met, or "".

References

Guédon Y (2003). “Estimating hidden semi-Markov chains from discrete sequences.” *Journal of Computational and Graphical Statistics*, **12**(3), 604-639. doi:10.1198/1061860032030.

Mitchell CD, Jamieson LH (1993). “Modeling duration in a hidden Markov model with the exponential family.” In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 331-334. doi:10.1109/ICASSP.1993.319304.

Examples

```
# =====
# An example of a 2-state HSMM with an SCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times with shape
# alpha); in the incomplete State 2 the number N of events missing between
# two observed events is shifted Poisson with parameter lambda_2, so the
# observed events are N + 1 such gamma interevent times apart. The sojourns
# are shifted Poisson with
# parameters dur (expected sojourns 1 + dur = 5 and 4 observations). With
# m = 2 the zero-diagonal transition matrix is fixed: the chain alternates
# between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
               rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
               WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with SCPG distribution

obj_hsmm1 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm1)
# Simulating a 2-state HSMM with SCPG distribution

sim_hsmm1 <- sim.hsmm.scpg(obj_hsmm1, nsim = 150, seed = 1)
str(sim_hsmm1[c("x", "states")])

# Number of visits to each state in the simulated path
table(rle(sim_hsmm1$states)$values)
# Fitting a 2-state HSMM with SCPG distributions
# (the parameters of obj_hsmm1 are the starting values)

fit_hsmm1 <- fit.hsmm.scpg(sim_hsmm1)
fit_hsmm1$smc.par$dur # sojourn parameters mu_1, mu_2 (true 4, 3)
fit_hsmm1$scpg.par$rate # lambda_2 (true 3)
```

```

fit_hsmm1$scpg.par$shape # alpha (true 1.5)
fit_hsmm1$scpg.par$scale # beta (true 1)
fit_hsmm1$logLik
# =====
# An example of a 3-state HSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 15 seconds on a laptop (the
# 2-state model above takes about 1 second), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.3, 0.7, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd=NULL, tpm=tpm, dur=c(3, 5, 2),
               dur.lb=NA, dur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
               rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
               WBSum.method='direct.evaluation')

# Defining an object of 3-state HSMM with SCPG distribution

obj_hsmm1_m3 <- hsmm.scp(x = NULL, smc.par, scpg.par)
# Simulating a 3-state HSMM with SCPG distribution

sim_hsmm1_m3 <- sim.hsmm.scp(obj_hsmm1_m3, nsim = 80, seed = 1)
print(sim_hsmm1_m3)
# Fitting a 3-state HSMM with SCPG distributions
# The fit takes about 15 seconds, so it is wrapped in donttest.

fit_hsmm1_m3 <- fit.hsmm.scp(sim_hsmm1_m3)
print(fit_hsmm1_m3)

# =====
# An example of a 2-state HSMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter  $\lambda_2(t)$ , where
#  $\lambda_2(t) = K + M / (1 + \exp(R * h_{t-1} - P))$ 
# falls from about 5.8 at the start of the record towards  $K = 0.5$ , around
# the inflection time  $P/R = 400$  of the last occurrence time  $h_{t-1}$ . The
# sojourns are shifted Poisson with parameters dur (expected sojourns
#  $1 + dur = 5$  and 4 observations). With  $m = 2$  the zero-diagonal transition
# matrix is fixed: the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,

```

```

      1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5      # K
rate[2, ] <- 6        # M
rate[3, ] <- 0.005    # R
rate[4, ] <- 2        # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with TSCPG distribution

obj_hsmm2 <- hsmm.scp(x = NULL, smc.par, scpg.par)
class(obj_hsmm2)
# Simulating a 2-state HSMM with a TSCPG distribution

sim_hsmm2 <- sim.hsmm.scp(obj_hsmm2, nsim = 300, seed = 3)
str(sim_hsmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2$x[-300]), h_T = sum(sim_hsmm2$x))
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state HSMM with a TSCPG distribution
# (the parameters of obj_hsmm2 are the starting values)

fit_hsmm2 <- fit.hsmm.scp(sim_hsmm2)
fit_hsmm2$smc.par$dur      # sojourn parameters mu_1, mu_2 (true 4, 3)
fit_hsmm2$scpg.par$rate    # K, M, R, P of lambda_2(t)
fit_hsmm2$scpg.par$scale   # true 1
fit_hsmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# Inflection diagnostics of the fitted TSCPG rate lambda_2(t), whose true
# inflection time is 400. The change lies inside the record (first > 0 >
# last), most of it takes place on [0, h_{T-1}] (change >= 0.5), and the
# expected number of observations of State 2 within its central 80 per cent
# (in.band, from the smoothed state probabilities) is well above 10, so no
# warning is given and the inflection time P/R, estimated at about 412, is
# identified.
fit_hsmm2$inflection

# =====
# An example of a 3-state HSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 7 seconds), so the examples that fit

```

```

# it are wrapped in donttest.

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete States 2 and 3 the number of missing events N between two
# observed events is shifted Poisson with parameter  $\lambda_j(t)$ , where
#  $\lambda_j(t) = K_j + M_j / (1 + \exp(R_j * h_{t-1} - P_j))$ 
# falls around the inflection time  $P_j/R_j$  of the last occurrence time
#  $h_{t-1}$ :  $\lambda_2(t)$  from about 3.4 towards 0.5 around 600, and
#  $\lambda_3(t)$  from about 10 towards 4 around 900. The sojourns are shifted
# Poisson with parameters dur (expected sojourns 1 + dur = 5, 4 and 4
# observations), and the zero-diagonal transition matrix gives the next
# state of each visit. The shape 2 makes the interevent times of the three
# states less overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3.6, 5.4)    # P: inflection times P/R = 600 and 900
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBsum.method = 'direct.evaluation')

# Defining an object of 3-state HSMM with TSCPG distribution

obj_hsmm2_m3 <- hsmm.scp(x = NULL, smc.par, scpg.par)
class(obj_hsmm2_m3)
# Simulating a 3-state HSMM with a TSCPG distribution

sim_hsmm2_m3 <- sim.hsmm.scp(obj_hsmm2_m3, nsim = 400, seed = 3)
str(sim_hsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rates are evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2_m3$x[-400]), h_T = sum(sim_hsmm2_m3$x))

# Number of visits to each state in the simulated path
table(rle(sim_hsmm2_m3$states)$values)
# Fitting a 3-state HSMM with a TSCPG distribution
# (the parameters of obj_hsmm2_m3 are the starting values). The fit
# takes about 2 minutes, so it is wrapped in donttest.

fit_hsmm2_m3 <- fit.hsmm.scp(sim_hsmm2_m3)
fit_hsmm2_m3$smc.par$tpm      # transition matrix of the embedded chain
# sojourn parameters mu_1, mu_2, mu_3 (true 4, 3, 3)
fit_hsmm2_m3$smc.par$dur

```

```

fit_hsmm2_m3$scpg.par$rate      # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_hsmm2_m3$scpg.par$scale    # true 0.5
fit_hsmm2_m3$logLik

# Inflection diagnostics of the fitted TSCPG rates lambda_2(t) and
# lambda_3(t), whose true inflection times are 600 and 900. For each rate
# the change lies inside the record (first > 0 > last), most of it takes
# place on [0, h_{T-1}] (change >= 0.5), and the expected number of
# observations of the state within its central 80 per cent (in.band, from
# the smoothed state probabilities) is above 10, so no warning is given and
# both inflection times P_j/R_j are identified.

fit_hsmm2_m3$inflection

```

fit.ihmm.scpg	<i>Fitting of an m-state IHMM with gamma or SCPG or TSCPG renewal processes</i>
---------------	---

Description

Estimates the parameters of an IHMM with class `ihmm.gamma`, `ihmm.scpg` and `ihmm.tscpg` as typically returned by [ihmm.scpg](#) using direct nonlinear optimization.

Usage

```

fit.ihmm.scpg(
  ihmmobj,
  method = "BFGS",
  maxit = 1000,
  reltol = sqrt(.Machine$double.eps),
  negative = TRUE
)

```

Arguments

<code>ihmmobj</code>	a model object with class <code>ihmm.gamma</code> , <code>ihmm.scpg</code> and <code>ihmm.tscpg</code> as typically returned by ihmm.scpg .
<code>method</code>	Optimization method passed to optim . Default is "BFGS".
<code>maxit</code>	Maximum number of iterations passed to the control argument of optim . Default is 1000.
<code>reltol</code>	Relative convergence tolerance passed to the control argument of optim . Default is <code>sqrt(.Machine\$double.eps)</code> .
<code>negative</code>	is a logical argument for negative log-likelihood. Default is TRUE.

Details

Likelihood. The likelihood of an IHMM is obtained from that of an HMM by replacing the transition probabilities with their time-varying counterparts (Zucchini et al., 2016),

$$L(\theta) = \Pr(X_{1:T} \mid H_{1:T}; \theta) = \sum_{s_1, \dots, s_T} \pi_{s_1} f_{s_1}(x_1) \prod_{t=2}^T a_{s_{t-1}s_t}(t) f_{s_t}(x_t),$$

where $\theta = \{\pi_j, r_{ij}, w_{ij}, \theta_j\}$. With the forward variable $\alpha_t(j) = \Pr(X_1, \dots, X_t, S_t = j \mid H_{t-1}; \theta)$, the recursion becomes

$$\alpha_1(j) = \pi_j f_j(x_1), \quad \alpha_t(j) = \left\{ \sum_{i=1}^m \alpha_{t-1}(i) a_{ij}(t) \right\} f_j(x_t), \quad t = 2, \dots, T,$$

and $L(\theta) = \sum_{j=1}^m \alpha_T(j)$. The transition probability matrix is reconstructed from its logistic coefficients at each step, using the occurrence time accumulated from the preceding observations; the likelihood is consequently not invariant under permutation of the observations. As in the homogeneous case, the recursion is scaled at each step and the logarithms of the scale factors accumulated.

Estimation. The likelihood is maximized by direct numerical minimization of the negative log-likelihood using `optim` over working parameters defined on the real line. The transition structure is described by $2m(m-1)$ logistic coefficients. Each pair (r_{ij}, w_{ij}) enters the log-odds $w_{ij} - r_{ij}h_{t-1}$, evaluated at the last occurrence time $h_{t-1} = x_1 + \dots + x_{t-1}$, and is represented by $\log(r_{ij}h_{T-1})$ and $w_{ij}/(r_{ij}h_{T-1})$, the time w_{ij}/r_{ij} as a fraction of $h_{T-1} = x_1 + \dots + x_{T-1}$, the value of h_{t-1} for the last interevent time $t = T$. The estimates of $\log r_{ij}$ and w_{ij} are almost perfectly correlated whenever w_{ij}/r_{ij} lies in $(0, h_{T-1})$, whereas these working parameters are much less correlated and do not depend on the unit of time. The initial distribution is represented by multinomial logits and the observation parameters as in `fit.scpg`. The inverse transformations are applied to the minimizer, so that the fitted object is a valid `ihmm.scpg` object.

Inflection diagnostics. The transition probabilities change along the record through the log-odds $u_t = w_{ij} - r_{ij}h_{t-1}$, which is zero at $h_{t-1} = w_{ij}/r_{ij}$: for $m = 2$ the inflection time of $a_{ij}(t)$, and for $m \geq 3$ the time at which $a_{ij}(t) = a_{i1}(t)$. The departure at time t uses $a_{ij}(t)$ at h_{t-1} , so over the record $a_{ij}(t)$ is observed for h_{t-1} between h_1 ($t = 2$) and $h_{T-1} = x_1 + \dots + x_{T-1}$ ($t = T$). The information about (r_{ij}, w_{ij}) comes from the departures from state i , each weighted by $a_{ij}(t)\{1 - a_{ij}(t)\}$: where the transition probability is close to 0 or 1 the observed transitions are almost all of one kind. The likelihood is therefore nearly flat in w_{ij}/r_{ij} when the rarer outcome of a departure from state i (moving to j , or not) is expected fewer than about 10 times, and exactly flat in one direction when it is never observed. Because the log-odds is linear, the profile information about w_{ij}/r_{ij} decreases more slowly than for the four-parameter curves of the TSCPG and the IHSM: with the states known, by a factor of about 3.7 (against about 45) for each multiple δ of $1/r_{ij}$ by which it lies outside $[0, h_{T-1}]$. When the states are TSCPG (`time.scpg = TRUE`), the rates $\lambda_j(t)$ are checked as in `fit.hmm.scpg`; how the two kinds of curve affect each other is described at the end of Details.

The fitted object contains the component `inflection` (see `Value`). The limits for a transition are that at least half of the logistic change takes place on $[0, h_{T-1}]$ (change ≥ 0.5), which places w_{ij}/r_{ij} inside this interval, and that `rare` is at least 10; the limits for $\lambda_j(t)$ are those of `fit.hmm.scpg`. In simulations of a two-state IHMM with 300 observations the 95 per cent profile interval of w_{12}/r_{12} was bounded when the rarer outcome of a departure from State 1 was expected 8.6 times (at the true values) and unbounded when it was expected 5.4 times or fewer, even with 293 departures.

A warning is issued when a limit is not met. The crossing time should then be reported as not identified rather than by its estimate, and a model with fewer time-dependent parameters, such as the homogeneous `fit.hmm.scpG`, compared by AIC. When a fitted transition probability is nearly constant on $[0, h_T - 1]$, its crossing time is not identified under the hypothesis of no change and the chi-squared calibration of likelihood-ratio tests and profile intervals fails.

Two kinds of time dependence (class `ihmm.tscpg`). With `time.scpG = TRUE` the record can become more complete in two ways: the transition probabilities $a_{ij}(t)$ govern how often the record is in an incomplete state, and the Poisson means $\lambda_j(t)$ how many events are missed while it is there. Both change the mean interevent time, $\alpha\beta$ in State 1 and $\alpha\beta\{\lambda_j(t) + 2\}$ in State j , so the data tell them apart only through the shape of the distribution of x_t and the persistence of the states. This affects their inflection times in two ways. First, $\lambda_j(t)$ is informed only by the observations of State j : if the transitions make State j rare where $\lambda_j(t)$ changes, few observations inform it, which `in.band` shows, since it counts the expected observations of State j . Second, as $\lambda_j(t)$ falls, the interevent times of State j (sums of at least two complete ones) come closer to those of State 1, the state probabilities become less certain, and every curve is then less precisely determined, which the counts `in.band` and `rare` do not show. In simulations of a two-state IHMM with 500 observations, whose $a_{12}(t)$ and $a_{22}(t)$ change around 304 and 600 and whose $\lambda_2(t)$ falls between the levels 5.5 and 0.5 around 200, 400 or 600 (30 records each), the estimated inflection time of $\lambda_2(t)$ and the crossing times of $a_{12}(t)$ and $a_{22}(t)$ were almost uncorrelated (Spearman correlations between -0.01 and 0.21), also when $\lambda_2(t)$ and $a_{22}(t)$ changed at the same time. Their precision depended on how long the two states stayed distinct: with the inflection time of $\lambda_2(t)$ at 200 rather than 600, the spread (interquartile range divided by 1.349) of the estimated inflection times was 124 rather than 49 for $\lambda_2(t)$ and 149 rather than 49 for $a_{22}(t)$, and the mean absolute error of the proportion of events recorded, $p_t = \Pr(S_t = 1 | x) + \sum_{j \geq 2} \Pr(S_t = j | x) / \{2 + \lambda_j(t)\}$, averaged over each quarter of the record, was 0.060 rather than 0.030, although `in.band` was about the same (medians 49 and 53). When both kinds are fitted, compare the model by AIC with those that have only one: `fit.hmm.scpG` with `time.scpG = TRUE` (only $\lambda_j(t)$) and `fit.ihmm.scpG` with `time.scpG = FALSE` (only $a_{ij}(t)$); and report p_t , which combines them, alongside the inflection times.

The script `verify_fit_ihmm_scpG.R` in the `examples` directory of the package reproduces the numbers stated on this help page; run it in the console with

```
f <- system.file("examples", "verify_fit_ihmm_scpG.R",
                 package = "HMMscpg")
source(f, echo = TRUE)
```

Value

The returned object provides estimated HMM with additional information about the function evaluation and gradient counts at the estimated minimum of negative log-likelihood function, convergence code and minimized likelihood function value as returned by `optim` and `nparam` is the number of parameters to be estimated. It also contains `inflection`, a data frame with the inflection diagnostics of Details, with one row for each transition $i \rightarrow j$, $j = 2, \dots, m$, and, for TSCPG states, one row for each $\lambda_j(t)$. For the transitions:

`component`, label "a" and "i -> j".

`rate`, intercept r_{ij} and w_{ij} .

`inflection` w_{ij}/r_{ij} .

first, last w_{ij} and $u_T = w_{ij} - r_{ij}h_{T-1}$, the log-odds at $h_{t-1} = 0$ and at h_{T-1} . The crossing lies in $(0, h_{T-1})$ when $\text{first} > 0 > \text{last}$; otherwise $-\text{first}$ is its distance before 0, or last its distance after h_{T-1} , in units of $1/r_{ij}$.

change the part of the logistic change, between 0 and 1, that takes place on $[0, h_{T-1}]$.

n the expected number of departures from state i , $\sum_{t=2}^T \Pr(S_{t-1} = i \mid x)$, from the smoothed state probabilities.

in.band the part of n from departures whose log-odds u_t lies in $(-\log 9, \log 9)$.

rare the expected number of the rarer outcome of a departure from state i , $\sum_{t=2}^T \Pr(S_{t-1} = i \mid x) \min\{a_{ij}(t), 1 - a_{ij}(t)\}$.

flag the limits of Details that are not met, or "".

References

Zucchini W, MacDonald IL, Langrock R (2016). *Hidden Markov Models for Time Series: An Introduction Using R*, 2 edition. CRC Press, Boca Raton.

Examples

```
# =====
# An example of a 2-state IHMM with an SCPG renewal process
# =====

# The transition probabilities a_12(t) and a_22(t) into the incomplete
# State 2 decrease along the record: the log-odds of moving from state i to
# State 2 rather than to State 1 is w_i2 - r_i2 * h_{t-1}, which is zero at
# the crossing time w_i2 / r_i2 (250 for a_12(t) and 500 for a_22(t)). In
# State 2 the number N of events missing between two observed events is
# shifted Poisson with parameter lambda_2. The chain starts mostly in State 2
# (initd = (0.2, 0.8)),
# since the early part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.004, 0.006) # rates of change r_12, r_22
ttpm[, , 2] <- c(1, 3)        # intercepts w_12, w_22
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

# Defining an object of 2-state IHMM with SCPG distribution

obj_ihmm1 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm1)
# Simulating a 2-state IHMM with SCPG distribution
```

```

sim_ihmm1 <- sim.ihmm.scpg(obj_ihmm1, nsim = 200, seed = 1)
str(sim_ihmm1[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm1$states,
      half = ifelse(seq_along(sim_ihmm1$states) <= 100, "first", "second"))
# The fit takes about 6 s, so the code below is wrapped in donttest.

# Fitting a 2-state IHMM with SCPG distributions
# (the parameters of obj_ihmm1 are the starting values)

fit_ihmm1 <- fit.ihmm.scpg(sim_ihmm1)
fit_ihmm1$imc.par$ttpm[, 1, ] # r_i2 (first column) and w_i2 (second)
fit_ihmm1$scpg.par$rate      # lambda_2 (true 3)
fit_ihmm1$scpg.par$shape     # true 1.5
fit_ihmm1$scpg.par$scale     # true 1
fit_ihmm1$logLik

# The fit takes about 6 s, so the code below is wrapped in donttest.

# Inflection diagnostics of the fitted transition probabilities a_12(t) and
# a_22(t) (true crossing times 250 and 500; labelled "i -> j"). Each curve
# is checked against the limits change >= 0.5 and rare >= 10 expected
# departures of the rarer kind (moving to State 2, or not). Here both are
# met (rare about 19 and 14), so no warning is given; the crossing times
# are estimated at about 442 and 486.
fit_ihmm1$inflection

# =====
# An example of a 3-state IHMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 6 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(0.45, 0.25, 0.30)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.0008, 0.0010, 0.0006,
                      0.0006, 0.0008, 0.0010), nrow = m, ncol = m-1)
# intercepts w_ij; a_ij(t) = a_i1(t) at h = w_ij/r_ij, between 2000 and 3000
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(2500, 2000, 3000,
                                      2000, 3000, 2500), nrow = m, ncol = m-1)

stationary <- FALSE
imc.par <- list(initd=initd, ttpm=ttpm, stationary=stationary)
rate <- matrix(c(7, 2), nrow = 1, ncol = m-1)
shape <- 1.7
scale <- 3.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,

```

```

WBsum.method='direct.evaluation')

# Defining an object of 3-state IHMM with SCPG distribution

obj_ihmm1_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
# Simulating a 3-state IHMM with SCPG distribution

sim_ihmm1_m3 <- sim.ihmm.scpg(obj_ihmm1_m3, nsim = 200, seed = 2)
print(sim_ihmm1_m3)
# Fitting a 3-state IHMM with SCPG distributions
# The fit takes about 1.5 minutes, so it is wrapped in donttest.

fit_ihmm1_m3 <- fit.ihmm.scpg(sim_ihmm1_m3)
print(fit_ihmm1_m3)

# Inflection diagnostics of the fitted transition probabilities (true
# crossing times between 2000 and 3000). The rates  $r_{ij}$  of this example are
# small (0.0006 to 0.001), so each change is wide compared with the record
# ( $h_{\{T-1\}} = 4733$ ), and with 200 observations only  $a_{12}(t)$  meets the
# limits here: the crossing times of  $a_{13}(t)$ ,  $a_{22}(t)$ ,  $a_{32}(t)$  and  $a_{33}(t)$ 
# lie outside the record, with less than half of the change on
#  $[0, h_{\{T-1\}}]$ , and  $a_{13}(t)$ ,  $a_{22}(t)$  and  $a_{23}(t)$  have fewer than 10
# expected departures of the rarer kind. The fit warns, and the flagged
# crossing times should be reported as not identified.

fit_ihmm1_m3$inflection

# =====
# An example of a 2-state IHMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scpg):
# - the transition probabilities  $a_{12}(t)$  and  $a_{22}(t)$  into the incomplete
#   State 2 decrease: the log-odds of moving from state  $i$  to State 2
#   rather than to State 1 is  $w_{i2} - r_{i2} * h_{\{t-1\}}$ , which is zero at the
#   crossing time  $w_{i2} / r_{i2}$  (about 304 for  $a_{12}(t)$  and 600 for  $a_{22}(t)$ );
# - the Poisson mean  $\lambda_{22}(t)$  of the TSCPG falls from about 5.4 at the
#   start of the record towards  $K = 0.5$ , around the inflection time
#    $P/R = 400$ .
# The chain starts mostly in State 2 ( $initd = (0.2, 0.8)$ ), since the early
# part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.0028, 0.005) # rates of change  $r_{12}$ ,  $r_{22}$ 
ttpm[, , 2] <- c(0.85, 3) # intercepts  $w_{12}$ ,  $w_{22}$ 
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(c(0.5, 5, 0.01, 4), nrow = 4) # K, M, R, P:  $P/R = 400$ 
shape <- 1
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,

```

```

        rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
        WBSum.method='direct.evaluation')

# Defining an object of 2-state IHMM with TSCPG distribution

obj_ihmm2 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm2)
# Simulating a 2-state IHMM with a TSCPG distribution

sim_ihmm2 <- sim.ihmm.scpg(obj_ihmm2, nsim = 200, seed = 2)
str(sim_ihmm2[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm2$states,
      half = ifelse(seq_along(sim_ihmm2$states) <= 100, "first", "second"))
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state IHMM with a TSCPG distribution
# (the parameters of obj_ihmm2 are the starting values)

fit_ihmm2 <- fit.ihmm.scpg(sim_ihmm2)
fit_ihmm2$imc.par$ttpm[, 1, ] # r_i2 (first column) and w_i2 (second)
fit_ihmm2$scpg.par$rate[, 1] # K, M, R, P of lambda_2(t)
fit_ihmm2$scpg.par$scale      # true 1
fit_ihmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# Inflection diagnostics of the three fitted logistic curves: lambda_2(t)
# (true inflection time 400; component "lambda") and a_12(t), a_22(t)
# (true crossing times 304 and 600; component "a", labelled "i -> j").
# Each curve is checked against its own limits: change >= 0.5 for all of
# them, in.band >= 10 expected observations of State 2 for lambda_2(t), and
# rare >= 10 expected departures of the rarer kind for a_ij(t). Here every
# limit is met, so no warning is given. lambda_2(t) is located well (402),
# but a_12(t) changes slowly (1/r_12 is about 360 time units) and its
# crossing time is estimated at 542, while that of a_22(t), at 637, lies
# near the end of the record (h_{T-1} = 691), with only 55 per cent of its
# change on [0, h_{T-1}]: identified is not the same as precisely
# estimated. How the two kinds of curve affect each other is described in
# the Details of ?fit.ihmm.scpg.
fit_ihmm2$inflection

# =====
# An example of a 3-state IHMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2.5 minutes on a laptop (the
# 2-state model above takes about 8 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along the

```

```

# record (see the Details of ?fit.ihmm.scpg):
# - the transition probabilities  $a_{ij}(t)$ : moves into the incomplete
#   States 2 and 3 become less likely as time goes on. The log-odds of
#   moving from state  $i$  to state  $j$  rather than to State 1 is
#    $w_{ij} - r_{ij} * h_{\{t-1\}}$ , which is zero at the crossing time  $w_{ij} / r_{ij}$ ;
# - the Poisson means  $\lambda_j(t)$  of the TSCPG: in the incomplete states
#   fewer events are missed after the inflection times  $P_j / R_j$ .
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1, so that the states are easier to tell apart.

m <- 3
initd <- c(0.4, 0.3, 0.3)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change  $r_{ij}$  (rows  $i = 1, 2, 3$ ; columns  $j = 2, 3$ )
ttpm[, , 1] <- matrix(c(0.006, 0.006, 0.006,
                        0.006, 0.006, 0.006), nrow = m, ncol = m-1)
# intercepts  $w_{ij}$ ; crossing times  $w_{ij}/r_{ij}$  between 400 and 600
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(500, 600, 500,
                                       400, 500, 600), nrow = m, ncol = m-1)
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(0.5, 4)      # K:  $\lambda_2(t)$  falls from about 3.4 towards
rate[2, ] <- c(3, 6)       # M: 0.5,  $\lambda_3(t)$  from about 10 towards 4
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3, 4.2)     # P: inflection times  $P/R = 500$  and  $700$ 
shape <- 2
scale <- 0.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHMM with TSCPG distribution

obj_ihmm2_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm2_m3)
# Simulating a 3-state IHMM with a TSCPG distribution

sim_ihmm2_m3 <- sim.ihmm.scpg(obj_ihmm2_m3, nsim = 500, seed = 1)
str(sim_ihmm2_m3[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into the incomplete states become less likely
table(state = sim_ihmm2_m3$states,
      half = ifelse(seq_along(sim_ihmm2_m3$states) <= 250, "first", "second"))
# Fitting a 3-state IHMM with a TSCPG distribution
# (the parameters of obj_ihmm2_m3 are the starting values). The fit
# takes about 2.5 minutes, so it is wrapped in donttest.

fit_ihmm2_m3 <- fit.ihmm.scpg(sim_ihmm2_m3)
fit_ihmm2_m3$imc.par$ttpm[, , 1] #  $r_{ij}$  (rows  $i$ , columns  $j = 2, 3$ )
fit_ihmm2_m3$imc.par$ttpm[, , 2] #  $w_{ij}$ 
fit_ihmm2_m3$scpg.par$rate      # K, M, R, P of  $\lambda_2(t)$  and  $\lambda_3(t)$ 
fit_ihmm2_m3$logLik

```

```

# Inflection diagnostics of the eight fitted logistic curves: lambda_2(t)
# and lambda_3(t) (true inflection times 500 and 700; component "lambda")
# and the six transition curves (true crossing times 400 to 600; component
# "a", labelled "i -> j"). Each curve is checked against its own limits:
# change >= 0.5 for all of them, in.band >= 10 expected observations of the
# state for lambda_j(t), and rare >= 10 expected departures of the rarer
# kind for a_ij(t).
#
# Here lambda_2(t) and the six transition curves meet their limits, while
# lambda_3(t) is flagged: it is fitted as a step, with no observation of
# State 3 inside its change. This shows how the two kinds of time
# dependence interact: lambda_3(t) is informed only by the observations of
# State 3, and the transitions make State 3 rare later in the record (in
# the simulated path, 31 of the 70 observations before h = 500 but only 16
# of the 374 after h = 700, where lambda_3(t) changes), so the fit can put
# the whole change between two of them (see the Details
# of ?fit.ihsmm.scpG). Its inflection time should be reported
# as not identified; a model with fewer time-dependent parameters may be
# compared by AIC.

fit_ihmm2_m3$inflection

```

fit.ihsmm.scpG

Fitting of an m-state IHSMM with gamma or SCPG or TSCPG renewal processes

Description

Estimates the parameters of an IHSMM with class `ihsmm.gamma`, `ihsmm.scpG` and `ihsmm.tscpg` as typically returned by `ihsmm.scpG` using direct nonlinear optimization.

Usage

```

fit.ihsmm.scpG(
  ihsmmobj,
  method = "BFGS",
  maxit = 1000,
  reltol = sqrt(.Machine$double.eps),
  negative = TRUE
)

```

Arguments

<code>ihsmmobj</code>	a model object with class <code>ihsmm.gamma</code> , <code>ihsmm.scpG</code> and <code>ihsmm.tscpg</code> as typically returned by <code>ihsmm.scpG</code> .
<code>method</code>	Optimization method passed to <code>optim</code> . Default is "BFGS".

maxit	Maximum number of iterations passed to the control argument of <code>optim</code> . Default is 1000.
reltol	Relative convergence tolerance passed to the control argument of <code>optim</code> . Default is <code>sqrt(.Machine\$double.eps)</code> .
negative	is a logical argument for negative log-likelihood. Default is TRUE.

Details

Likelihood. The likelihood of a right-censored IHSMM is evaluated by the extension of the forward algorithm given by Shahzadi et al. (2023, 2026). The structure is that of `fit.hsmm.scp`, with the sojourn distribution evaluated at the time at which each visit commences. With $u_t(j, d) = \prod_{l=t-d+1}^t f_j(x_l)$ and $\alpha_{t+1}^*(j) = \sum_{i \neq j} \alpha_t(i) a_{ij}$, the forward variable is

$$\alpha_t(j) = \sum_{d=1}^t \alpha_{t-d+1}^*(j) p_j(d, t-d+1) u_t(j, d), \quad t = 1, \dots, T-1,$$

where $p_j(d, t)$ is the shifted Poisson sojourn mass function with parameter $\mu_j(t)$ (expected sojourn $1 + \mu_j(t)$). At $t = T$ the sojourn in the last visited state is right-censored and the mass function is replaced by the corresponding survival function $S_j(d, T-d+1)$, giving $L(\theta) = \sum_{j=1}^m \alpha_T(j)$. The expression reduces to the right-censored HSMM likelihood when the sojourn distributions are independent of t .

The time dependence has a computational consequence. In a homogeneous HSMM the sojourn mass and survival functions are tabulated once; here they are evaluated at every admissible starting time, the forward variable being indexed by state, elapsed sojourn and commencement time. Scaling is embedded in the recursion at each step and the logarithms of the scale factors accumulated. The recursion is evaluated by compiled code, which computes the sojourn mass and survival functions as it proceeds. Fitting an IHSMM is the most demanding of the four model families, and the estimates of the corresponding homogeneous HSMM provide suitable starting values.

Estimation. The likelihood is maximized by direct numerical minimization of the negative log-likelihood using `optim` over working parameters defined on the real line: each row of the zero-diagonal transition probability matrix by multinomial logits; A_j and B_j by logarithms; and the pair (C_j, D_j) , which enters $\mu_j(t)$ through $C_j h_{t-1} - D_j$ at the last occurrence time $h_{t-1} = x_1 + \dots + x_{t-1}$ of the commencement t of a visit, by $\log(C_j h_{T-1})$ and $D_j / (C_j h_{T-1})$, the inflection time D_j / C_j as a fraction of $h_{T-1} = x_1 + \dots + x_{T-1}$, the value of h_{t-1} for the last interevent time $t = T$. The estimates of $\log C_j$ and D_j are almost perfectly correlated whenever D_j / C_j lies in $(0, h_{T-1})$, whereas these working parameters are much less correlated and do not depend on the unit of time. Entries for which `tdur.lb` and `tdur.ub` are supplied are represented by scaled logits instead, and a state with bounds for C_j or D_j keeps $\log C_j$ and D_j . The observation parameters are transformed as in `fit.scp`. The fitted object is returned on the natural scale and is suitable for `vip.ihsmm.scp` and `sp.ihsmm.scp`.

Inflection diagnostics. The expected sojourns change along the record through $C_j h_{t-1} - D_j$, with inflection time $h_{t-1} = D_j / C_j$, at which $\mu_j(t) = A_j + B_j / 2$. Two aspects of identifiability should be noted. A visit that commences at time t uses $\mu_j(t)$ at h_{t-1} , which over the record runs from $h_0 = 0$ ($t = 1$) to $h_{T-1} = x_1 + \dots + x_{T-1}$ ($t = T$), so the duration curves are determined only on $[0, h_{T-1}]$: when D_j / C_j lies δ multiples of $1/C_j$ outside this interval, the values of $\mu_j(t)$ on it determine only three combinations of (A_j, B_j, C_j, D_j) , and the profile information about D_j / C_j decreases like $e^{-4\delta}$, by a factor of about 45 per multiple when the states are known. And

the four duration parameters of a state are informed by the number of visits to that state, each visit contributing one sojourn, rather than by the length of the record. When the states are TSCPG (`time.scpg = TRUE`), the rates $\lambda_j(t)$ are checked as in `fit.hmm.scpg`; how the two kinds of curve affect each other is described at the end of Details.

The fitted object contains the component inflection (see Value). The limits for a sojourn parameter are that at least 65 per cent of the change of $\mu_j(t)$ takes place on $[0, h_{T-1}]$ (change ≥ 0.65 , stricter than the 0.5 of the other curves), which places D_j/C_j inside this interval, and that the state is expected to be visited at least 20 times; the limits for $\lambda_j(t)$ are those of `fit.hmm.scpg`. In simulations of a two-state IHSM with 150 observations and 12 to 14 visits to each state, the 95 per cent profile interval of D_1/C_1 was unbounded wherever the inflection time lay, including 2.75 multiples of $1/C_1$ inside $(0, h_{T-1})$; with the states known it was bounded around the true value only when the inflection time lay inside this interval (with the inflection time two multiples before the start it was bounded but excluded the true value). The limit on change comes from profile likelihoods of D_j/C_j for 900 sojourn curves simulated with the states known (two-state IHSMs with 150, 300 or 600 observations, $C_j h_{T-1}$ between 2 and 32 and the inflection time at 0.3, 0.6 or 0.9 of the record): of the curves meeting the limits change ≥ 0.5 and 20 visits, 374 had a bounded 95 per cent interval and 177 did not; raising the limit to 0.65 flagged 32 of the 177 and none of the 374 (0.75 would have flagged 59 and 2). A limit on `in.band`, the expected number of visits commencing within the central 80 per cent of the change, was also tried and rejected: an abrupt change located between two visits had a bounded interval with `in.band` as low as 0, whereas most unbounded intervals had `in.band` above 15, so any minimum flagged more identified curves than unidentified ones (a minimum of 5 flagged 100 and 29).

A warning is issued when a limit is not met. The inflection time should then be reported as not identified rather than by its estimate, and a model with fewer time-dependent parameters, such as the homogeneous `fit.hsmm.scpg`, compared by AIC. When a fitted $\mu_j(t)$ is nearly constant on $[0, h_{T-1}]$, D_j/C_j is not identified under the hypothesis of no change and the chi-squared calibration of likelihood-ratio tests and profile intervals fails.

Two kinds of time dependence (class ihsmm.tscpg). With `time.scpg = TRUE` the record can become more complete in two ways: the sojourn parameters $\mu_j(t)$ govern how long the chain stays in the complete and the incomplete states, and the Poisson means $\lambda_j(t)$ how many events are missed while it is in State j . Both change the mean interevent time, $\alpha\beta$ in State 1 and $\alpha\beta\{\lambda_j(t) + 2\}$ in State j , so the data tell them apart only through the shape of the distribution of x_t and the lengths of the visits. As for `fit.ihmm.scpg`, $\lambda_j(t)$ is informed only by the observations of State j , which `in.band` counts, and the states become harder to tell apart as $\lambda_j(t)$ falls. In simulations of a two-state IHSM with 300 observations, whose $\mu_1(t)$ (increasing) and $\mu_2(t)$ (decreasing) change around 500 and whose $\lambda_2(t)$ falls between the levels 6.5 and 0.5 around 200, 500 or 800 (20 records each), the inflection times of the sojourn parameters were poorly determined wherever $\lambda_2(t)$ changed (spread, the interquartile range divided by 1.349, between 222 and 273, more than one multiple $1/C_j = 200$). When $\lambda_2(t)$ and $\mu_2(t)$, which both describe State 2, changed at the same time, their estimated inflection times were negatively correlated (Spearman correlation -0.52 : a later fall in the number of missing events was offset by an earlier shortening of the incomplete visits); otherwise the correlations between the two kinds of curve lay between -0.19 and 0.15 . The spread of the inflection time of $\lambda_2(t)$ grew from 127 to 261 as it moved from 200 to 800, where State 2 is seldom visited, which `in.band` shows (medians 90 and 59). The mean absolute error of the proportion of events recorded, $p_t = \Pr(S_t = 1 \mid x) + \sum_{j \geq 2} \Pr(S_t = j \mid x) / \{2 + \lambda_j(t)\}$, averaged over each quarter of the record, lay between 0.023 and 0.047. When both kinds are fitted, compare the model by AIC with those that have only one: `fit.hsmm.scpg` with `time.scpg = TRUE` (only $\lambda_j(t)$) and `fit.ihsmm.scpg` with `time.scpg = FALSE` (only $\mu_j(t)$); judge the inflection times of $\lambda_j(t)$ and

$\mu_j(t)$ of the same state jointly, for example by a parametric bootstrap with `sim.ihsmm.scpg`; and report p_t , which combines them, alongside the inflection times.

The script `verify_fit_ihsmm_scpg.R` in the `examples` directory of the package reproduces the numbers stated on this help page; run it in the console with

```
f <- system.file("examples", "verify_fit_ihsmm_scpg.R",
                 package = "HMMscpg")
source(f, echo = TRUE)
```

Value

The returned object provides estimated IHSM with additional information about the function evaluation and gradient counts at the estimated minimum of negative log-likelihood function, convergence code and minimized likelihood function value as returned by `optim` and `nparam` is the number of parameters to be estimated. It also contains `inflection`, a data frame with the inflection diagnostics of Details, with one row for each $\mu_j(t)$, $j = 1, \dots, m$, and, for TSCPG states, one row for each $\lambda_j(t)$. For the sojourn parameters:

`component`, label "mu" and "state j".

`rate`, intercept C_j and D_j .

`inflection` D_j/C_j .

`first`, `last` $u_1 = D_j$ and $u_T = D_j - C_j h_{T-1}$, the values of $u_t = D_j - C_j h_{t-1}$ for visits commencing at $t = 1$ ($h_0 = 0$) and $t = T$ (h_{T-1}). The inflection time lies in $(0, h_{T-1})$ when `first` $> 0 > \text{last}$; otherwise `-first` is its distance before 0, or `last` its distance after h_{T-1} , in units of $1/C_j$.

`change` the part of the change of $\mu_j(t)$, between 0 and 1, that takes place on $[0, h_{T-1}]$.

`n` the expected number of visits to state j , the sum over $t = 1, \dots, T$ of the probabilities, given the record, that a visit to state j begins at t , obtained from the forward-backward recursion.

`in.band` the part of `n` from visits commencing at t with h_{t-1} in the central 10 to 90 per cent of the change, where $|u_t| < \log 9$.

`rare` NA; used by `fit.ihmm.scpg`.

`flag` the limits of Details that are not met, or "".

References

Shahzadi A, Wang T, Parry M, Bebbington M (2023). "Inhomogeneous hidden semi-Markov models for incompletely observed point processes." *Annals of Institute Statistical Mathematics*, **75**, 253-280. doi:10.1007/s10463022008435.

Shahzadi A, Wang T, Parry M, Bebbington M (2026). "Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models." *Advances in Data Analysis and Classification*, **20**, 663-691. doi:10.1007/s1163402500632x.

Examples

```
# =====
# An example of a 2-state IHSMM with an SCPG renewal process
# =====

# The sojourn parameters  $\mu_j(t)$  depend on the time at which a visit
# commences (rows A, B, C, D of tdur): the expected sojourn  $1 + \mu_1(t)$  in
# the complete State 1 increases, and that in the incomplete State 2
# decreases, between the levels  $1 + A_j = 1.5$  and  $1 + A_j + B_j = 5.5$ 
# observations, around the inflection time  $D_j/C_j = 500$ . In State 2,
# the number N of events missing between two observed events is shifted
# Poisson with parameter  $\lambda_2$ .
# With  $m = 2$  the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5,      # A
  4, 4,          # B
  0.005, 0.005, # C
  2.5, 2.5      # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(3, nrow = 1, ncol = m-1) #  $\lambda_2$ 
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHSMM with SCPG distribution

obj_ihsmm1 <- ihsmm.scpg(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm1)
# Simulating a 2-state IHSMM with SCPG distribution

sim_ihsmm1 <- sim.ihsmm.scpg(obj_ihsmm1, nsim = 150, seed = 3)
str(sim_ihsmm1[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm1$states)$values)
# The fit takes about 30 s, so the code below is wrapped in donttest.

# Fitting a 2-state IHSMM with SCPG distributions
# (the parameters of obj_ihsmm1 are the starting values)

fit_ihsmm1 <- fit.ihsmm.scpg(sim_ihsmm1)
fit_ihsmm1$ismc.par$tdur      # A, B, C, D of  $\mu_1(t)$  and  $\mu_2(t)$ 
fit_ihsmm1$scpg.par$rate     #  $\lambda_2$  (true 3)
```

```

fit_ihsmm1$scpg.par$shape      # true 1.5
fit_ihsmm1$scpg.par$scale      # true 1
fit_ihsmm1$logLik

# The fit takes about 30 s, so the code below is wrapped in donttest.

# Inflection diagnostics of the fitted sojourn parameters mu_1(t) and
# mu_2(t) (true inflection times D_j/C_j = 500). The duration parameters of
# a state are informed by its visits; the limits are change >= 0.65 and at
# least 20 expected visits (n). Here both are met (about 21 visits to each
# state), so no warning is given; the inflection times are estimated at
# about 424 and 452. mu_2(t) is fitted as an abrupt step, with almost no
# visit commencing within its change (in.band close to 0); such a step can
# still be located, which is why in.band is not a limit for mu_j(t) (see
# the Details of ?fit.ihsmm.scp).
fit_ihsmm1$inflection

# =====
# An example of a 3-state IHSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 30 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.5, 0.5, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.3, 0.3, 0.3,      # A
  1.5, 2, 1.5,        # B
  0.005, 0.005, 0.005, # C
  3, 3, 3            # D: inflection times D/C = 600
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHSMM with SCPG distribution

obj_ihsmm1_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
# Simulating a 3-state IHSMM with SCPG distribution

sim_ihsmm1_m3 <- sim.ihsmm.scp(obj_ihsmm1_m3, nsim = 150, seed = 1)
print(sim_ihsmm1_m3)
# Fitting a 3-state IHSMM with SCPG distributions
# The fit takes about 2 minutes, so it is wrapped in donttest.

```

```

fit_ihsmm1_m3 <- fit.ihsmm.scp(sim_ihsmm1_m3)
print(fit_ihsmm1_m3)

# Inflection diagnostics of the fitted sojourn parameters (true inflection
# times  $D_j/C_j = 600$ ). The duration parameters of a state are informed by
# its visits, and with 150 observations all three curves are flagged here,
# and the fit warns: the inflection time of  $\mu_1(t)$  lies after the record
# (4850 against  $h_{\{T-1\}} = 1146$ , with only 2 per cent of its change on
#  $[0, h_{\{T-1\}}]$ ), and States 2 and 3 are expected to be visited only 19.5
# and 6.1 times, fewer than the 20 visits required.

fit_ihsmm1_m3$inflection

# =====
# An example of a 2-state IHSMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur): the expected
#   sojourn  $1 + \mu_1(t)$  in the complete State 1 increases, and that in the
#   incomplete State 2 decreases, between the levels  $1 + A_j = 2$  and
#    $1 + A_j + B_j = 8$  observations, around the inflection time
#    $D_j/C_j = 500$ ;
# - the Poisson mean  $\lambda_2(t)$  of the TSCPG (rows K, M, R, P of rate)
#   falls from about 5.8 at the start of the record towards  $K = 0.5$ ,
#   around the inflection time  $P/R = 400$ .

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  1, 1,          # A
  6, 6,          # B
  0.005, 0.005,  # C
  2.5, 2.5       # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5   # K
rate[2, ] <- 6     # M
rate[3, ] <- 0.005 # R
rate[4, ] <- 2     # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state IHSMM with TSCPG distribution

```

```

obj_ihsmm2 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2)
# Simulating a 2-state IHSMM with a TSCPG distribution

sim_ihsmm2 <- sim.ihsmm.scp(obj_ihsmm2, nsim = 200, seed = 1)
str(sim_ihsmm2[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2$states)$values)
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state IHSMM with a TSCPG distribution
# (the parameters of obj_ihsmm2 are the starting values)

fit_ihsmm2 <- fit.ihsmm.scp(sim_ihsmm2)
fit_ihsmm2$ismc.par$tdur      # A, B, C, D of mu_1(t) and mu_2(t)
fit_ihsmm2$scpg.par$rate     # K, M, R, P of lambda_2(t)
fit_ihsmm2$scpg.par$scale    # true 1
fit_ihsmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# Inflection diagnostics of the three fitted logistic curves: lambda_2(t)
# (true inflection time 400) and mu_1(t), mu_2(t) (true 500). Each curve is
# checked against its own limits: change >= 0.5 and in.band >= 10
# observations of State 2 for lambda_2(t); change >= 0.65 and at least 20
# expected visits (n) for mu_1(t) and mu_2(t). Here every limit is met
# (about 22 expected visits to each state), so no warning is given, and
# the inflection times are estimated at about 399, 530 and 440. The two
# kinds of curve describe the same improvement of the record and are
# estimated jointly; meeting the limits does not show that their
# inflection times are estimated independently of each other (see the
# Details of ?fit.ihsmm.scp).
fit_ihsmm2$inflection

# =====
# An example of a 3-state IHSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 8 minutes on a laptop (the
# 2-state model above takes about 50 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters mu_j(t) (rows A, B, C, D of tdur), with
#   inflection times D_j/C_j = 500: the expected sojourn 1 + mu_1(t) in the
#   complete State 1 increases from about 2.1 towards 6.5 observations,
#   and those in the incomplete States 2 and 3 decrease from about 5.9
#   towards 1.5;
# - the Poisson means lambda_j(t) of the TSCPG (rows K, M, R, P of rate):
#   lambda_2(t) falls from about 3.3 towards 0.5 around 400, and

```

```

# lambda_3(t) from about 9.8 towards 4 around 600.
# The zero-diagonal transition matrix gives the next state of each visit.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5, 0.5,      # A
  5, 5, 5,            # B
  0.004, 0.004, 0.004, # C
  2, 2, 2            # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(2.4, 3.6)    # P: inflection times P/R = 400 and 600
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 3-state IHSMM with TSCPG distribution

obj_ihsmm2_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2_m3)
# Simulating a 3-state IHSMM with a TSCPG distribution

sim_ihsmm2_m3 <- sim.ihsmm.scp(obj_ihsmm2_m3, nsim = 250, seed = 2)
str(sim_ihsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the time-dependent components are evaluated
# on [0, h_{T-1}]
c(h_T1 = sum(sim_ihsmm2_m3$x[-250]), h_T = sum(sim_ihsmm2_m3$x))

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2_m3$states)$values)
# Fitting a 3-state IHSMM with a TSCPG distribution
# (the parameters of obj_ihsmm2_m3 are the starting values). The fit
# takes about 8 minutes, so it is wrapped in donttest.

fit_ihsmm2_m3 <- fit.ihsmm.scp(sim_ihsmm2_m3)
fit_ihsmm2_m3$ismc.par$tpm      # transition matrix of the embedded chain
fit_ihsmm2_m3$ismc.par$tdur     # A, B, C, D of mu_1(t), mu_2(t) and mu_3(t)
fit_ihsmm2_m3$scpg.par$rate     # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_ihsmm2_m3$scpg.par$scale    # true 0.5

```

```

fit_ihsmm2_m3$logLik

# Inflection diagnostics of the five fitted logistic curves: lambda_2(t)
# and lambda_3(t) (true inflection times 400 and 600; component "lambda")
# and mu_1(t), mu_2(t) and mu_3(t) (true 500; component "mu"). Each curve
# is checked against its own limits: change >= 0.5 and in.band >= 10
# expected observations of the state for lambda_j(t); change >= 0.65 and
# at least 20 expected visits (n) for mu_j(t).
#
# With 250 observations and about 20 visits to each state, two curves are
# flagged, one by each kind of limit:
# - lambda_2(t): its estimated inflection time, 398, is close to the true
#   one, but the fitted change is narrow (its central 80 per cent lies
#   between about 346 and 451) and only 6.5 expected observations of
#   State 2 fall within it, too few to locate it;
# - mu_3(t): it falls on the record, from about 2.8 to 1.2, but as the
#   tail of a much wider curve whose inflection time D_3/C_3 = -1323 lies
#   before the start, with only 12 per cent of its change on [0, h_{T-1}].
# Both inflection times should be reported as not identified. The other
# three curves meet their limits, which does not make them precise: the
# inflection times of mu_1(t) and mu_2(t) are estimated at 729 and 609
# rather than 500, and lambda_3(t) at 586 but falling from about 14 to 2
# instead of from 9.8 to 4.5. The two kinds of curve describe the same
# improvement of the record and are estimated jointly (see the Details
# of ?fit.ihsmm.scpg).

fit_ihsmm2_m3$inflection

```

fit.scpg

Fitting the SCPG and TSCPG distributions

Description

Estimates the parameters of an SCPG or TSCPG distribution, or of a gamma distribution with a fixed number of missing events, by maximum likelihood; see [dscpg](#) for the models.

Usage

```

fit.scpg(
  x,
  rate,
  shape,
  scale,
  rate.fix = FALSE,
  shape.fix = FALSE,
  time.scpg = FALSE,
  WBsum.method = "direct.evaluation",
  method = "BFGS",

```

```

maxit = 1000,
reltol = sqrt(.Machine$double.eps),
negative = TRUE
)

```

Arguments

<code>x</code>	A vector of positive, finite interevent times. For the TSCPG (<code>time.scpg = TRUE</code>) they must be in time order.
<code>rate, shape, scale</code>	Starting values, as in dscpg . With <code>rate.fix = TRUE</code> , <code>rate</code> is the fixed number of missing events and is not estimated; with <code>shape.fix = TRUE</code> , <code>shape</code> is not estimated.
<code>rate.fix</code>	Logical. If TRUE, exactly rate events are missing between consecutive observed events, and X_t is gamma distributed with shape $(n + 1)\alpha$ and scale β . Default is FALSE.
<code>shape.fix</code>	Logical; used only by fit.scpg , where TRUE holds the shape at the supplied value instead of estimating it. Default is FALSE.
<code>time.scpg</code>	Logical. If TRUE, the TSCPG distribution is used, with <code>rate = c(K, M, R, P)</code> , and <code>rate.fix</code> must be FALSE. Default is FALSE (SCPG).
<code>WBsum.method</code>	The method used by WBsum in the densities: "direct.evaluation" (the default), "direct.asymptotic" or "asymptotic.expansion". If a fit is slow, use "direct.asymptotic", which gives the same estimates; see the Details of dscpg .
<code>method</code>	Optimization method passed to optim . Default is "BFGS".
<code>maxit</code>	Maximum number of iterations passed to the control argument of optim . Default is 1000.
<code>reltol</code>	Relative convergence tolerance passed to the control argument of optim . Default is <code>sqrt(.Machine\$double.eps)</code> .
<code>negative</code>	Logical. If TRUE (default), optim minimizes the negative log-likelihood, otherwise it maximizes the log-likelihood. The estimates are the same either way.

Details

Likelihood. For interevent times $x_1, \dots, x_T$ the log-likelihood is $\log L = \sum_{t=1}^T \log f(x_t)$, with f the density of [dscpg](#); the parameters are (λ, α, β) for the SCPG and $(K, M, R, P, \alpha, \beta)$ for the TSCPG. For the TSCPG, $f(x_t)$ depends on $h_{t-1} = x_1 + \dots + x_{t-1}$, so `x` must be in time order.

Estimation. There is no closed-form estimator, and $\log L$ is maximized with [optim](#) over unconstrained working parameters (Zucchini et al., 2016). The shape is kept in $(0.3, 100)$ and λ , or K and M , in $(0.001, 500)$, through the scaled logit $\log\{(\theta - l)/(u - \theta)\}$ of each interval (l, u) ; the scale is estimated on the log scale. Without these bounds the likelihood has a ridge along which the shape tends to zero with almost no gain, while the series of [WBsum](#) needs ever more terms. For the TSCPG, the pair (R, P) is replaced by $\log(R h_{T-1})$ and $P/(R h_{T-1})$, the inflection time P/R as a fraction of $h_{T-1} = x_1 + \dots + x_{T-1}$, the last occurrence time used by the record; $R h_{T-1}$ is kept in $(0.001, 500)$. On this scale the two working parameters are nearly uncorrelated and do not depend on the unit of time, whereas $\log R$ and P are almost perfectly correlated. In 16 simulated TSCPG

records fitted from perturbed starting values, this choice reached the best optimum in all 16 fits, against 10 for $(\log R, P)$. Starting values must lie inside the bounds. The estimates are returned on the natural scale, and the number of estimated parameters `nparam` allows comparison of models by $AIC = -2 \log L + 2 \text{nparam}$ (Akaike, 1974).

Inflection diagnostics (TSCPG). The rate $\lambda(t)$ falls from about $K + M$ to K , and most of this change (the central 80 per cent) takes place within $2.2/R$ of the inflection time P/R . The record observes $\lambda(t)$ only at $h_0 = 0, h_1, \dots, h_{T-1}$. The data can therefore locate P/R only if the change falls inside $[0, h_{T-1}]$ and enough observations fall within it. Otherwise the record shows only one end, or only the almost straight middle, of the S-shaped curve, which many values of P/R fit almost equally well: the likelihood is nearly flat in P/R , and likelihood-ratio tests and profile intervals for P/R are unreliable. `fit.scpg` therefore returns the component inflection (see `Value`) and gives a warning when

- less than half of the change of $\lambda(t)$ takes place on $[0, h_{T-1}]$ (change below 0.5). This covers every P/R outside $[0, h_{T-1}]$, and a $\lambda(t)$ that is almost constant over the record; or
- fewer than 10 observations lie within the central 80 per cent of the change.

These limits come from profile likelihoods of P/R in 87 simulated records of 250 observations, with the shape held at 1, $\lambda(t)$ falling between the levels 4.5 and 0.5, $R h_{T-1}$ between about 0.5 and 10, and the inflection time at 0.5, 0.9 or 1.15 times h_{T-1} . Every fit with change below 0.5 had a 95 per cent profile interval for P/R reaching 1.5 record lengths before 0 or after h_{T-1} , and every bounded interval came from a fit with change of at least 0.5. The limits are necessary rather than sufficient: 22 of the 52 fits that met them still had unbounded intervals, so a profile likelihood or a parametric bootstrap should confirm an inflection time that is to be interpreted. After a warning, report P/R as not identified rather than quoting its estimate, and compare the fit with the SCPG (`time.scpg = FALSE`) by AIC.

The script `verify_fit_scpg.R` in the examples directory of the package reproduces the numbers stated here; run it in the console with

```
f <- system.file("examples", "verify_fit_scpg.R",
                 package = "HMMscpg")
source(f, echo = TRUE)
```

Value

A list with components

`fun.count`, `grad.count` numbers of function and gradient evaluations used by `optim`.

`convergence` the `optim` convergence code; 0 indicates successful convergence.

`nparam` the number of estimated parameters.

`llk` the maximized log-likelihood.

`rate.est` the estimate of λ (SCPG) or of $c(K, M, R, P)$ (TSCPG), or the fixed rate (`rate.fix = TRUE`).

`shape.est`, `scale.est` the estimates of α (the supplied value when `shape.fix = TRUE`) and β .

`rate.fix`, `shape.fix`, `time.scpg`, `WBsum.method` the settings used.

`objective.error` NULL, or a message when the log densities at the estimates could not be computed or are not finite.

inflection NULL unless time.scpg = TRUE. For the TSCPG, a one-row data frame with the inflection diagnostics of Details:

- component, label "lambda" and "record".
- rate, intercept the estimates of R and P .
- inflection the inflection time P/R .
- first, last P and $P - R h_{T-1}$: the position of P/R relative to 0 and to h_{T-1} , in units of $1/R$. P/R lies inside $[0, h_{T-1}]$ when $\text{first} \geq 0 \geq \text{last}$.
- change the fraction, between 0 and 1, of the change of $\lambda(t)$ that takes place on $[0, h_{T-1}]$, $|\sigma(P) - \sigma(P - R h_{T-1})|$ with $\sigma(u) = 1/(1 + e^{-u})$; the first limit fails when it is below 0.5.
- n the number of observations T .
- in.band the number of observations x_t whose h_{t-1} lies within the central 80 per cent of the change, $|P - R h_{t-1}| < \log 9$.
- rare always NA for fit.scpg.
- flag the limits that are not met, or "".

References

Zucchini W, MacDonald IL, Langrock R (2016). *Hidden Markov Models for Time Series: An Introduction Using R*, 2 edition. CRC Press, Boca Raton.

Akaike H (1974). "A new look at the statistical model identification." *IEEE Transactions on Automatic Control*, **19**(6), 716-723. doi:10.1109/TAC.1974.1100705.

Examples

```
# =====
# Example of an SCPG distribution
# =====
rate <- 3
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from an SCPG distribution

x_scpg <- rscpg(nsim=200, rate, shape, scale, seed=900)
print(x_scpg)
# Fitting an SCPG distribution

fit_scpg <- fit.scpg(x_scpg, rate, shape, scale, rate.fix=FALSE,
  shape.fix = TRUE,
  WBSum.method = 'direct.evaluation')
print(fit_scpg)
# =====
# Example of a TSCPG distribution
# =====

# rate[1]=K, rate[2]=M, rate[3]=R, rate[4]=P: lambda(t) falls from about K+M
```

```

# to K,
# with inflection time P/R = 300
t_rate <- c(1, 2, 0.02, 6)
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from a TSCPG distribution

x_tscpg <- rscpg(nsim=200, t_rate, shape, scale, time.scpg=TRUE, seed=700)
print(x_tscpg)
# Fitting of a TSCPG distribution

fit_tscpg <- fit.scpg(x_tscpg, t_rate, shape, scale, rate.fix=FALSE,
  shape.fix = FALSE,
  time.scpg=TRUE, WBSum.method = 'direct.evaluation')
print(fit_tscpg)

```

hmm.scpg

An m-state HMM with gamma, SCPG or TSCPG renewal processes

Description

Creates an m -state hidden Markov model (HMM) for an incomplete record of interevent times, with class `hmm.gamma`, `hmm.scpg` or `hmm.tscpg`.

Usage

```

hmm.scpg(
  x = NULL,
  mc.par = list(initd = NULL, tpm = NULL, stationary = TRUE),
  scpg.par = list(rate = NULL, shape = NULL, scale = NULL,
    rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
    WBSum.method = "direct.evaluation")
)

```

Arguments

<code>x</code>	A vector of positive interevent times in time order, or <code>NULL</code> when the object is only used to simulate.
<code>mc.par</code>	A named list with the parameters of the hidden Markov chain: <code>initd</code> the initial distribution, a vector of m probabilities summing to one; ignored when <code>stationary = TRUE</code>. <code>tpm</code> the $m \times m$ transition probability matrix, with rows summing to one. <code>stationary</code> logical; if <code>TRUE</code> (default), the initial distribution is the stationary distribution of <code>tpm</code>.

`scpg.par` A named list with the parameters of the observation process:

- `rate` the numbers of missing events in States $2, \dots, m$, one column per state: a $1 \times (m - 1)$ matrix of Poisson means λ_j (SCPG), a $4 \times (m - 1)$ matrix with columns (K_j, M_j, R_j, P_j) (TSCPG), or a $1 \times (m - 1)$ matrix of fixed numbers n_j (`rate.fix = TRUE`).
- `shape` the gamma shape α : one value common to all states, or one value per state.
- `scale` the gamma scale β , common to all states.
- `rate.fix` logical; if TRUE, exactly n_j events are missing in State j , which is then not estimated.
- `shape.fix` logical; if TRUE, the shape is held at its supplied value when fitting, for example 1, which makes the complete process a Poisson process.
- `time.scpg` logical; if TRUE, States $2, \dots, m$ are TSCPG; requires `rate.fix = FALSE`.
- `WBsum.method` the method used by [WBsum](#) for the SCPG and TSCPG densities, one of
 - "direct.evaluation" (default), which suits almost all records;
 - "direct.asymptotic", for when to use it see [dscpg](#);
 - "asymptotic.expansion".

Details

The model. The hidden state $S_t \in \{1, \dots, m\}$ of the t th interevent time follows a homogeneous Markov chain with initial distribution $\pi_i = \Pr(S_1 = i)$ and transition probabilities $a_{ij} = \Pr(S_t = j \mid S_{t-1} = i)$. State j therefore lasts a geometric number of observations, $a_{jj}^{d-1}(1 - a_{jj})$, $d = 1, 2, \dots$. Given the state, x_t has density $f_j(x_t)$:

- State 1 is the state of completeness: no event is missing and x_t is gamma with shape α_1 and scale β .
- States $2, \dots, m$ are states of incompleteness, in which x_t is the sum of $N + 1$ complete interevent times, N being the number of missing events, with the density of [dscpg](#): gamma with $N = n_j$ fixed (class `hmm.gamma`), SCPG with N shifted Poisson with parameter λ_j (`hmm.scpg`), or TSCPG with parameter $\lambda_j(t) = K_j + M_j / \{1 + \exp(R_j h_{t-1} - P_j)\}$, which changes around the inflection time P_j / R_j of the last occurrence time $h_{t-1} = x_1 + \dots + x_{t-1}$ (`hmm.tscpg`).

The scale β is common to all states because the sum of gamma variables is gamma only when they share the scale; the shape may be common or differ between states. The parameters are therefore π_i , a_{ij} and θ_j , with $\theta_1 = (\alpha_1, \beta)$ and, for $j \geq 2$, $\theta_j = (n_j, \alpha_j, \beta)$ with n_j fixed, $(\lambda_j, \alpha_j, \beta)$ or $(K_j, M_j, R_j, P_j, \alpha_j, \beta)$.

The object. `hmm.scpg` checks the parameters and returns them with `x`. With `stationary = TRUE` the initial distribution is computed as the solution of $\pi = \pi A$ and adds no parameters. The object is the model and starting values for `fit.hmm.scpg`, the generating model for `sim.hmm.scpg`, and, once fitted and with the data attached, the input of `vip.hmm.scpg` and `sp.hmm.scpg`.

Value

A list with components `x`, `mc.par` and `scpg.par` as above (`initd` computed when `stationary = TRUE`), of class `hmm.gamma`, `hmm.scpg` or `hmm.tscpg` according to `rate.fix` and `time.scpg`.

Examples

```
# =====
# An example of a 2-state HMM with an SCPG renewal process
# =====

# State 1 is the state of completeness: its interevent times are gamma with
# shape alpha and scale beta. In the incomplete State 2 the number of
# missing events N between two observed events is shifted Poisson with
# parameter lambda_2, so on average 1 + lambda_2 = 4 events are missing,
# and the observed interevent time is the sum of N + 1 such gamma
# interevent times. The shape is common to both states and, with
# shape.fix = FALSE, estimated; a separate shape for each state (a vector
# of shapes) is also possible but is less well determined.

m <- 2
tpm <- matrix(c(0.9, 0.1,
                0.2, 0.8), nrow = 2, byrow = TRUE)
mc.par <- list(initd = NULL, tpm = tpm, stationary = TRUE)
rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
                 WBSum.method = 'direct.evaluation')

# Defining an object for 2-state HMM with SCPG distribution

obj_hmm1 <- hmm.scpg(x = NULL, mc.par, scpg.par)
print(obj_hmm1)
# =====
# An example of a 3-state HMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 2 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(1/3, 1/3, 1/3)
tpm <- matrix(c(0.2, 0.65, 0.15, 0.60, 0.30, 0.10, 0.45, 0.30, 0.25),
              nrow = 3, byrow = TRUE)
mc.par <- list(initd=NULL, tpm=tpm, stationary=TRUE)
rate <- matrix(c(7, 3), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object for 3-state HMM with SCPG distribution

obj_hmm1_m3 <- hmm.scpg(x = NULL, mc.par, scpg.par)
```

```

print(obj_hmm1_m3)
# =====
# An example of a 2-state HMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter lambda_2(t), where
#   lambda_2(t) = K + M / (1 + exp(R * h_{t-1} - P))
# falls from about 5.3 at the start of the record towards K = 0.5, around
# the inflection time P/R = 300 of the last occurrence time h_{t-1}. The
# hidden chain is homogeneous, so only lambda_2(t) changes along the record.

m <- 2
tpm <- matrix(c(0.9, 0.1,
                0.2, 0.8), nrow = 2, byrow = TRUE)
mc.par <- list(initd = NULL, tpm = tpm, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5   # K
rate[2, ] <- 5     # M
rate[3, ] <- 0.01  # R
rate[4, ] <- 3     # P: inflection time P/R = 300
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpG = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object for 2-state HMM with TSCPG distribution

obj_hmm2 <- hmm.scpG(x = NULL, mc.par, scpg.par)
class(obj_hmm2)
# =====
# An example of a 3-state HMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 30 seconds on a laptop (the
# 2-state model above takes about 4 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(1/3, 1/3, 1/3)
tpm <- matrix(c(0.2, 0.65, 0.15, 0.50, 0.30, 0.20, 0.25, 0.50, 0.25),
              nrow = 3, byrow = TRUE)
mc.par <- list(initd=initd, tpm=tpm, stationary=TRUE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(1, 1) # K
rate[2, ] <- c(1.5, 2.5) # M
rate[3, ] <- c(0.02, 0.015) # R
rate[4, ] <- c(3, 3) # P: inflection times P/R = 150 and 200
shape <- 1
scale <- 1.7
scpg.par <- list(rate=rate, shape=shape, scale=scale,

```

```

rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
WBSum.method='direct.evaluation')

# Defining an object for 3-state HMM with TSCPG distribution
obj_hmm2_m3 <- hmm.scpg(x = NULL, mc.par, scpg.par)

```

hsmm.scpg

An m -state HSMM with gamma or SCPG or TSCPG renewal processes

Description

Creates a discrete time HSMM object with class `hsmm.gamma`, `hsmm.scpg`, or `hsmm.tscpg`.

Usage

```

hsmm.scpg(
  x = NULL,
  smc.par = list(initd = NULL, tpm = NULL, dur = NULL,
    dur.lb = NA, dur.ub = NA, stationary = TRUE),
  scpg.par = list(rate = NULL, shape = NULL, scale = NULL,
    rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
    WBSum.method = "direct.evaluation")
)

```

Arguments

- | | |
|-----------------------|---|
| <code>x</code> | A vector of positive interevent times in time order, or <code>NULL</code> when the object is only used to simulate. |
| <code>smc.par</code> | <p>named list containing semi-Markov chain parameters:</p> <p><code>initd</code> A numeric vector of length m for initial probability distribution. Sum of all probabilities must be one.</p> <p><code>tpm</code> is an $m \times m$ zero-diagonal numerical row stochastic matrix with each row summing to 1.</p> <p><code>dur</code> is a vector of length m containing parameters of shifted Poisson distributions for state durations in each of m hidden states.</p> <p><code>dur.lb</code> is a vector of length m to specify the reasonable lower bound for the shifted Poisson duration parameters in each of m states. Default is <code>NA</code>.</p> <p><code>dur.ub</code> is a vector of length m to specify the reasonable upper bound for the shifted Poisson duration parameters in each of m states. Default is <code>NA</code>.</p> <p><code>stationary</code> A logical argument. If <code>TRUE</code>, initial distribution is computed as stationary distribution implied by <code>tpm</code>.</p> |
| <code>scpg.par</code> | <p>A named list with the parameters of the observation process:</p> <p><code>rate</code> the numbers of missing events in States $2, \dots, m$, one column per state: a $1 \times (m - 1)$ matrix of Poisson means λ_j (SCPG), a $4 \times (m - 1)$ matrix with columns (K_j, M_j, R_j, P_j) (TSCPG), or a $1 \times (m - 1)$ matrix of fixed numbers n_j (<code>rate.fix = TRUE</code>).</p> |

shape the gamma shape α : one value common to all states, or one value per state.

scale the gamma scale β , common to all states.

rate.fix logical; if TRUE, exactly n_j events are missing in State j , which is then not estimated.

shape.fix logical; if TRUE, the shape is held at its supplied value when fitting, for example 1, which makes the complete process a Poisson process.

time.scpg logical; if TRUE, States 2, ..., m are TSCPG; requires rate.fix = FALSE.

WBsum.method the method used by [WBsum](#) for the SCPG and TSCPG densities, one of

- "direct.evaluation" (default), which suits almost all records;
- "direct.asymptotic", for when to use it see [dscpg](#);
- "asymptotic.expansion".

Details

A semi-Markov chain (SMC) does not follow implicit geometric state durations; it is Markovian only at transition times, and the sojourn time in each state is at least 1. An HMM with an underlying SMC is an HSMM. The state sequence becomes two-dimensional, $S_{1:T} = \{(j_1, d_1), \dots, (j_n, d_n)\}$, where $j_r \in \{1, \dots, m\}$ is the state at the r th visit with $j_r \neq j_{r+1}$, d_r its sojourn time, and transition times $\tau_r = \sum_{l=1}^r d_l + 1$. The components are:

1. *Initial distribution*: the process starts in state j_1 with probability π_{j_1} .
2. *TPM with no self-transitions*: $a_{j_r j_{r+1}} = \Pr(S_{\tau_r+1} = j_{r+1} \mid S_{\tau_r} = j_r)$, $j_r \neq j_{r+1}$. The transition probability matrix `tpm` is required to have a zero diagonal. In a semi-Markov chain the embedded chain governs only the destination of a transition, the time spent in a state being governed by the sojourn time distribution; a non-zero self-transition probability would render the sojourn time unidentified (Guedon, 2003). For $m = 2$ the embedded chain is thereby determined completely and contributes no parameters to the estimation.
3. *Explicit state durations*: Sojourn times follow a shifted Poisson distribution with parameter μ_{j_r} ,

$$p_{j_r}(d_r) = \frac{\mu_{j_r}^{d_r-1} \exp(-\mu_{j_r})}{(d_r - 1)!}.$$

Since the last visited state j_n is right-censored, its duration distribution is replaced by the survival function $S_{j_n}(d_n) = \sum_{d_n \geq 1} p_{j_n}(d_n)$. The distribution being shifted, every visit occupies at least one observation.

4. *Joint state-dependent observation distribution*: the observations of the r th visit are independent given the state,

$$f(x_{(\tau_{r-1}+1):\tau_r} \mid S = j_r, \theta_j) = \prod_{l=\tau_{r-1}+1}^{\tau_r} f_{j_r}(x_l \mid \theta_j).$$

The gamma, SCPG, and TSCPG variants use SMC parameters $\{\pi_j, \{a_{ij}\}_{i \neq j}, \mu_j\}$ together with the corresponding observation parameters θ_j , see [hmm.scpg](#).

hsmm.scpg constructs and validates a specification of the m -state HSMM defined above. The vector `dur` contains the m sojourn parameters μ_j of the shifted Poisson sojourn distribution $p_j(d) = \mu_j^{d-1} \exp(-\mu_j)/(d-1)!, d = 1, 2, \dots$, whose expected sojourn time is $1 + \mu_j$ observations.

The arguments `dur.lb` and `dur.ub` form no part of the model. They delimit the interval within which each expected sojourn time is sought by `fit.hsmm.scpg`, an expected sojourn time otherwise being free to exceed the length of the record. Entries left as NA are estimated on the logarithmic scale subject only to positivity.

The observation conventions are those of `hmm.scpg`: State 1 denotes completeness and carries the gamma renewal density, `rate` describes the $m - 1$ incompleteness states alone, and a single scale parameter is shared across states. Observations occurring within one visit are conditionally independent draws from the observation distribution of the state occupied.

Value

returns a `list` object with class `hsmm.scpg`, `hsmm.tscpg`, or `hsmm.gamma` with the above arguments as named components.

References

Guédon Y (2003). “Estimating hidden semi-Markov chains from discrete sequences.” *Journal of Computational and Graphical Statistics*, **12**(3), 604-639. doi:[10.1198/1061860032030](https://doi.org/10.1198/1061860032030).

Examples

```
# =====
# An example of a 2-state HSMM with an SCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times with shape
# alpha); in the incomplete State 2 the number N of events missing between
# two observed events is shifted Poisson with parameter lambda_2, so the
# observed events are N + 1 such gamma interevent times apart. The sojourns
# are shifted Poisson with
# parameters dur (expected sojourns 1 + dur = 5 and 4 observations). With
# m = 2 the zero-diagonal transition matrix is fixed: the chain alternates
# between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with SCPG distribution
```

```

obj_hsmm1 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm1)
# =====
# An example of a 3-state HSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 15 seconds on a laptop (the
# 2-state model above takes about 1 second), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.3, 0.7, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd=NULL, tpm=tpm, dur=c(3, 5, 2),
               dur.lb=NA, dur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
               rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
               WBsum.method='direct.evaluation')

# Defining an object of 3-state HSMM with SCPG distribution

obj_hsmm1_m3 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
# =====
# An example of a 2-state HSMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter  $\lambda_2(t)$ , where
#  $\lambda_2(t) = K + M / (1 + \exp(R * h_{t-1} - P))$ 
# falls from about 5.8 at the start of the record towards  $K = 0.5$ , around
# the inflection time  $P/R = 400$  of the last occurrence time  $h_{t-1}$ . The
# sojourns are shifted Poisson with parameters dur (expected sojourns
#  $1 + \text{dur} = 5$  and 4 observations). With  $m = 2$  the zero-diagonal transition
# matrix is fixed: the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5      # K
rate[2, ] <- 6        # M
rate[3, ] <- 0.005    # R
rate[4, ] <- 2        # P: inflection time  $P/R = 400$ 
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,

```

```

rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
WBsum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with TSCPG distribution

obj_hsmm2 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm2)
# =====
# An example of a 3-state HSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 7 seconds), so the examples that fit
# it are wrapped in donttest.

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete States 2 and 3 the number of missing events N between two
# observed events is shifted Poisson with parameter  $\lambda_j(t)$ , where
#  $\lambda_j(t) = K_j + M_j / (1 + \exp(R_j * h_{t-1} - P_j))$ 
# falls around the inflection time  $P_j/R_j$  of the last occurrence time
#  $h_{t-1}$ :  $\lambda_2(t)$  from about 3.4 towards 0.5 around 600, and
#  $\lambda_3(t)$  from about 10 towards 4 around 900. The sojourns are shifted
# Poisson with parameters dur (expected sojourns  $1 + \text{dur} = 5, 4$  and  $4$ 
# observations), and the zero-diagonal transition matrix gives the next
# state of each visit. The shape 2 makes the interevent times of the three
# states less overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3.6, 5.4)    # P: inflection times  $P/R = 600$  and  $900$ 
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBsum.method = 'direct.evaluation')

# Defining an object of 3-state HSMM with TSCPG distribution

obj_hsmm2_m3 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm2_m3)

```

Description

Creates a discrete time IHMM object with class "gamma.scpg", `ihmm.scpg`, or `ihmm.tscpg`.

Usage

```
ihmm.scpg(
  x = NULL,
  imc.par = list(initd = NULL, ttpm = NULL, stationary = TRUE),
  scpg.par = list(rate = NULL, shape = NULL, scale = NULL,
    rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
    WBsum.method = "direct.evaluation")
)
```

Arguments

- | | |
|-----------------------|--|
| <code>x</code> | A vector of positive interevent times in time order, or NULL when the object is only used to simulate. |
| <code>imc.par</code> | named list containing the first-order inhomogeneous Markov chain (IMC) parameters.

<code>initd</code> A numeric vector of length m for the initial probability distribution.
<code>ttpm</code> An $m \times (m - 1) \times 2$ array of time-dependent transition probability parameters.
<code>stationary</code> If TRUE, <code>initd</code> is computed from the transition matrix at time 0. |
| <code>scpg.par</code> | A named list with the parameters of the observation process:

<code>rate</code> the numbers of missing events in States $2, \dots, m$, one column per state: a $1 \times (m - 1)$ matrix of Poisson means λ_j (SCPG), a $4 \times (m - 1)$ matrix with columns (K_j, M_j, R_j, P_j) (TSCPG), or a $1 \times (m - 1)$ matrix of fixed numbers n_j (<code>rate.fix = TRUE</code>).
<code>shape</code> the gamma shape α : one value common to all states, or one value per state.
<code>scale</code> the gamma scale β , common to all states.
<code>rate.fix</code> logical; if TRUE, exactly n_j events are missing in State j , which is then not estimated.
<code>shape.fix</code> logical; if TRUE, the shape is held at its supplied value when fitting, for example 1, which makes the complete process a Poisson process.
<code>time.scpg</code> logical; if TRUE, States $2, \dots, m$ are TSCPG; requires <code>rate.fix = FALSE</code> .
<code>WBsum.method</code> the method used by <code>WBsum</code> for the SCPG and TSCPG densities, one of <ul style="list-style-type: none"> • "direct.evaluation" (default), which suits almost all records; • "direct.asymptotic", for when to use it see dscpg; • "asymptotic.expansion". |

Details

An IHMM is an HMM whose underlying MC has a time-dependent TPM $A = [a_{ij}(t)]$, defined by a multinomial logistic function with State 1 (completeness) as the reference state. Because a record's recent part has fewer missing events, transition probabilities into State 1 increase with time. For $m = 2$ those into State 2 decrease; for $m \geq 3$ a transition probability into a State $j \geq 2$ need not be monotone (it can rise and then fall when r_{ij} is smaller than another r_{ik}), although its odds against State 1, $a_{ij}(t)/a_{i1}(t) = \exp(w_{ij} - r_{ij}h_{t-1})$, always decrease. The transition probabilities at time t are functions of the last occurrence time h_{t-1} as follows:

$$a_{i1}(t) = \frac{1}{1 + \sum_{k=2}^m \exp(-r_{ik}h_{t-1} + w_{ik})},$$

$$a_{ij}(t) = \frac{\exp(-r_{ij}h_{t-1} + w_{ij})}{1 + \sum_{k=2}^m \exp(-r_{ik}h_{t-1} + w_{ik})}, \quad j = 2, \dots, m,$$

where $r_{ij} > 0$ measures the rate of change and $-\infty < w_{ij} < \infty$ is the log-odds at $h_{t-1} = 0$. The location of the change is $h_{t-1} = w_{ij}/r_{ij}$, the time at which $a_{ij}(t) = a_{i1}(t)$; for $m = 2$ it is the inflection time of $a_{i2}(t)$, at which $a_{i2}(t) = 1/2$. For an $m \times m$ TPM, $2m(m-1)$ parameters are estimated, and the matrix remains stochastic for all t . State durations are implicitly time-dependent geometric, $p_j(d, t) = a_{jj}(t)^{d-1}(1 - a_{jj}(t))$. The gamma, SCPG, and TSCPG variants replace the homogeneous MC parameters with $\{\pi_j, r_{ij}, w_{ij}\}$.

`ihmm.scpg` constructs and validates a specification of the m -state IHMM defined above in which the transition probabilities are multinomial logistic functions of the last occurrence time. These probabilities are supplied not as a matrix but through the coefficients that generate it.

The coefficients are held in `ttpm`, an $m \times (m-1) \times 2$ array. The first two margins index the departure state i and the arrival state $j = 2, \dots, m$, State 1 being the reference category and carrying no coefficients; the third margin distinguishes the rate of change r_{ij} from the intercept w_{ij} , so that `ttpm[i, j-1, 1]` is r_{ij} and `ttpm[i, j-1, 2]` is w_{ij} . The $2m(m-1)$ coefficients so arranged determine the transition probability matrix at every time point, and the parameterization is row stochastic for all t by construction.

Since the transition probability matrix varies with time, stationarity requires a convention. If `stationary = TRUE`, the initial distribution is computed as the stationary distribution of the transition probability matrix evaluated at time zero.

The remaining conventions are those of `hmm.scpg`: State 1 denotes completeness and carries the gamma renewal density, `rate` describes the $m-1$ incompleteness states alone, and a single scale parameter is shared across states. The observation process determines the class of the returned object, on which `fit.ihmm.scpg`, `sim.ihmm.scpg`, `vip.ihmm.scpg` and `sp.ihmm.scpg` dispatch.

Value

returns `list` object with class `ihmm.scpg`, `ihmm.tscpg`, or `ihmm.gamma` with arguments as named components.

Examples

```
# =====
# An example of a 2-state IHMM with an SCPG renewal process
# =====
```

```

# The transition probabilities  $a_{12}(t)$  and  $a_{22}(t)$  into the incomplete
# State 2 decrease along the record: the log-odds of moving from state  $i$  to
# State 2 rather than to State 1 is  $w_{i2} - r_{i2} * h_{\{t-1\}}$ , which is zero at
# the crossing time  $w_{i2} / r_{i2}$  (250 for  $a_{12}(t)$  and 500 for  $a_{22}(t)$ ). In
# State 2 the number  $N$  of events missing between two observed events is
# shifted Poisson with parameter  $\lambda_{22}$ . The chain starts mostly in State 2
# (initd = (0.2, 0.8)),
# since the early part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.004, 0.006) # rates of change  $r_{12}$ ,  $r_{22}$ 
ttpm[, , 2] <- c(1, 3) # intercepts  $w_{12}$ ,  $w_{22}$ 
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(3, nrow = 1, ncol = m-1) #  $\lambda_{22}$ 
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

# Defining an object of 2-state IHMM with SCPG distribution

obj_ihmm1 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm1)
# =====
# An example of a 3-state IHMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 6 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(0.45, 0.25, 0.30)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change  $r_{ij}$  (rows  $i = 1, 2, 3$ ; columns  $j = 2, 3$ )
ttpm[, , 1] <- matrix(c(0.0008, 0.0010, 0.0006,
                      0.0006, 0.0008, 0.0010), nrow = m, ncol = m-1)
# intercepts  $w_{ij}$ ;  $a_{ij}(t) = a_{i1}(t)$  at  $h = w_{ij}/r_{ij}$ , between 2000 and 3000
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(2500, 2000, 3000,
                                     2000, 3000, 2500), nrow = m, ncol = m-1)

stationary <- FALSE
imc.par <- list(initd=initd, ttpm=ttpm, stationary=stationary)
rate <- matrix(c(7, 2), nrow = 1, ncol = m-1)
shape <- 1.7
scale <- 3.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

```

```

# Defining an object of 3-state IHMM with SCPG distribution

obj_ihmm1_m3 <- ihmm.scpG(x = NULL, imc.par, scpg.par)
# =====
# An example of a 2-state IHMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scpG):
# - the transition probabilities  $a_{12}(t)$  and  $a_{22}(t)$  into the incomplete
#   State 2 decrease: the log-odds of moving from state  $i$  to State 2
#   rather than to State 1 is  $w_{i2} - r_{i2} * h_{\{t-1\}}$ , which is zero at the
#   crossing time  $w_{i2} / r_{i2}$  (about 304 for  $a_{12}(t)$  and 600 for  $a_{22}(t)$ );
# - the Poisson mean  $\lambda_{22}(t)$  of the TSCPG falls from about 5.4 at the
#   start of the record towards  $K = 0.5$ , around the inflection time
#    $P/R = 400$ .
# The chain starts mostly in State 2 ( $initd = (0.2, 0.8)$ ), since the early
# part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.0028, 0.005) # rates of change  $r_{12}$ ,  $r_{22}$ 
ttpm[, , 2] <- c(0.85, 3) # intercepts  $w_{12}$ ,  $w_{22}$ 
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(c(0.5, 5, 0.01, 4), nrow = 4) # K, M, R, P:  $P/R = 400$ 
shape <- 1
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
  rate.fix=FALSE, shape.fix=TRUE, time.scpG=TRUE,
  WBSum.method='direct.evaluation')

# Defining an object of 2-state IHMM with TSCPG distribution

obj_ihmm2 <- ihmm.scpG(x = NULL, imc.par, scpg.par)
class(obj_ihmm2)
# =====
# An example of a 3-state IHMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2.5 minutes on a laptop (the
# 2-state model above takes about 8 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scpG):
# - the transition probabilities  $a_{ij}(t)$ : moves into the incomplete
#   States 2 and 3 become less likely as time goes on. The log-odds of
#   moving from state  $i$  to state  $j$  rather than to State 1 is
#    $w_{ij} - r_{ij} * h_{\{t-1\}}$ , which is zero at the crossing time  $w_{ij} / r_{ij}$ ;
# - the Poisson means  $\lambda_j(t)$  of the TSCPG: in the incomplete states
#   fewer events are missed after the inflection times  $P_j / R_j$ .
# The shape 2 makes the interevent times of the three states less

```

```

# overlapping than with shape 1, so that the states are easier to tell apart.

m <- 3
initd <- c(0.4, 0.3, 0.3)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.006, 0.006, 0.006,
                        0.006, 0.006, 0.006), nrow = m, ncol = m-1)
# intercepts w_ij; crossing times w_ij/r_ij between 400 and 600
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(500, 600, 500,
                                       400, 500, 600), nrow = m, ncol = m-1)

imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(0.5, 4)      # K: lambda_2(t) falls from about 3.4 towards
rate[2, ] <- c(3, 6)        # M: 0.5, lambda_3(t) from about 10 towards 4
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3, 4.2)      # P: inflection times P/R = 500 and 700
shape <- 2
scale <- 0.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
                 WBsum.method='direct.evaluation')

# Defining an object of 3-state IHMM with TSCPG distribution

obj_ihmm2_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm2_m3)

```

ihsmm.scpg

An m-state IHSM with gamma or SCPG, or TSCPG renewal processes

Description

Creates a discrete time IHSM object with class `ihsmm.scpg`, `ihsmm.tscpg`, or `ihsmm.gamma`.

Usage

```

ihsmm.scpg(
  x = NULL,
  ismc.par = list(initd = NULL, tpm = NULL, tdur = NULL,
                  tdur.lb = NA, tdur.ub = NA, stationary = TRUE),
  scpg.par = list(rate = NULL, shape = NULL, scale = NULL,
                  rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
                  WBsum.method = "direct.evaluation")
)

```

Arguments

<code>x</code>	A vector of positive interevent times in time order, or NULL when the object is only used to simulate.
<code>ismc.par</code>	named list containing inhomogeneous semi-Markov chain.
<code>initd</code>	A numeric vector of length m for initial probability distribution with sum of all probabilities equal to one.
<code>tpm</code>	is an $m \times m$ zero-diagonal numerical row stochastic matrix with each row summing to 1.
<code>tdur</code>	is a matrix of order $4 \times m$ containing parameters for defining time-dependent parameters of shifted Poisson distributions for state durations each of m hidden states.
<code>tdur.lb</code>	is a matrix of order $4 \times m$ to specify the lower bound of parameters in <code>tdur</code> . Default is NA.
<code>tdur.ub</code>	is a matrix of order $4 \times m$ to specify the upper bound of parameters in <code>tdur</code> . Default is NA.
<code>stationary</code>	A logical argument. If TRUE, initial distribution is computed as stationary distribution implied by <code>tpm</code> .
<code>scpg.par</code>	A named list with the parameters of the observation process: <code>rate</code> the numbers of missing events in States $2, \dots, m$, one column per state: a $1 \times (m - 1)$ matrix of Poisson means λ_j (SCPG), a $4 \times (m - 1)$ matrix with columns (K_j, M_j, R_j, P_j) (TSCPG), or a $1 \times (m - 1)$ matrix of fixed numbers n_j (<code>rate.fix = TRUE</code>). <code>shape</code> the gamma shape α: one value common to all states, or one value per state. <code>scale</code> the gamma scale β, common to all states. <code>rate.fix</code> logical; if TRUE, exactly n_j events are missing in State j, which is then not estimated. <code>shape.fix</code> logical; if TRUE, the shape is held at its supplied value when fitting, for example 1, which makes the complete process a Poisson process. <code>time.scpg</code> logical; if TRUE, States $2, \dots, m$ are TSCPG; requires <code>rate.fix = FALSE</code>. <code>WBSum.method</code> the method used by WBSum for the SCPG and TSCPG densities, one of "direct.evaluation" (default), which suits almost all records; "direct.asymptotic", for when to use it see dscpg; "asymptotic.expansion".

Details

An IHSM (Shahzadi et al., 2023) has an underlying inhomogeneous SMC (ISM) whose sojourn time distribution depends on time. Sojourn times in state j_r follow a shifted Poisson distribution with time-dependent parameter $\mu_{j_r}(t)$ as follows:

$$p_{j_r}(d_r, t) = \frac{[\mu_{j_r}(t)]^{d_r-1} \exp(-\mu_{j_r}(t))}{(d_r - 1)!}, \quad t = \tau_{r-1} + 1,$$

with the survival function used for the right-censored sojourn time of the last visited state. Here, $t = \tau_{r-1} + 1$ is the starting time of sojourn in r th visited state j_r . The expected sojourn time in state j is therefore $1 + \mu_j(t)$. Given time inhomogeneity in record completeness, $\mu_1(t)$ is an increasing function of the last occurrence time h_{t-1} ,

$$\mu_1(t) = \frac{B_1}{1 + \exp(-C_1 h_{t-1} + D_1)} + A_1,$$

and in states $j = 2, \dots, m$ a decreasing function,

$$\mu_j(t) = \frac{B_j}{1 + \exp(C_j h_{t-1} - D_j)} + A_j,$$

where $A_j, B_j, C_j > 0$ and $-\infty < D_j < \infty$. The gamma, SCPG, and TSCPG variants use ISMC parameters $\{\pi_j, \{a_{ij}\}_{i \neq j}, A_j, B_j, C_j, D_j\}$ together with the corresponding observation parameters θ_j .

ihsmm.scpkg constructs and validates a specification of the m -state IHSMM of Shahzadi et al. (2023, 2026), in which the expected sojourn times are generalized logistic functions of the last occurrence time, as defined above.

The duration parameters are held in `tdur`, a $4 \times m$ matrix whose columns index the states and whose rows are, in order, A_j, B_j, C_j and D_j . Each $\mu_j(t)$ is thereby confined to the interval $(A_j, A_j + B_j)$, and the expected sojourn time $1 + \mu_j(t)$ to $(1 + A_j, 1 + A_j + B_j)$. C_j governs the rate of change, and the inflection time, at which $\mu_j(t) = A_j + B_j/2$, is $h_{t-1} = D_j/C_j$ (D_j itself is the value of $C_j h_{t-1}$ there, not a time). The first three rows are required to be positive and the fourth is unconstrained. The requirement $C_j > 0$ in every state fixes the direction of each curve in advance: the expected sojourn time in State 1 increases towards $A_1 + B_1$ with elapsed time, and those in States $2, \dots, m$ decrease towards A_j . This imposes the assumption that completeness improves along the record, and precludes the estimation of a state whose direction is reversed. A homogeneous HSMM is recovered in the limit in which $\mu_j(t)$ is constant over the observed span.

As in `hsmm.scpkg`, the transition probability matrix is required to have a zero diagonal. The arguments `tdur.lb` and `tdur.ub` delimit the intervals within which the duration parameters are sought by `fit.ihsmm.scpkg` rather than forming part of the model; entries left as NA are estimated on the logarithmic scale where positivity is required and without transformation otherwise. Setting `stationary = TRUE` computes the initial distribution as the stationary distribution of the embedded chain. It is recommended that `stationary` be specified explicitly, since omitting it from `ismc.par` is not equivalent to the default shown in the function signature.

Value

returns a `list` object with class `ihsmm.scpkg`, `ihsmm.tscpg`, or `ihsmm.gamma` with the above arguments as named components.

References

- Shahzadi A, Wang T, Parry M, Bebbington M (2023). “Inhomogeneous hidden semi-Markov models for incompletely observed point processes.” *Annals of Institute Statistical Mathematics*, **75**, 253-280. doi:10.1007/s10463022008435.
- Shahzadi A, Wang T, Parry M, Bebbington M (2026). “Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models.” *Advances in Data Analysis and Classification*, **20**, 663-691. doi:10.1007/s1163402500632x.

Examples

```
# =====
# An example of a 2-state IHSMM with an SCPG renewal process
# =====

# The sojourn parameters  $\mu_j(t)$  depend on the time at which a visit
# commences (rows A, B, C, D of tdur): the expected sojourn  $1 + \mu_1(t)$  in
# the complete State 1 increases, and that in the incomplete State 2
# decreases, between the levels  $1 + A_j = 1.5$  and  $1 + A_j + B_j = 5.5$ 
# observations, around the inflection time  $D_j/C_j = 500$ . In State 2,
# the number  $N$  of events missing between two observed events is shifted
# Poisson with parameter  $\lambda_2$ .
# With  $m = 2$  the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5,      # A
  4, 4,          # B
  0.005, 0.005, # C
  2.5, 2.5       # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(3, nrow = 1, ncol = m-1) #  $\lambda_2$ 
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHSMM with SCPG distribution

obj_ihsmm1 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm1)
# =====
# An example of a 3-state IHSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 30 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.5, 0.5, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.3, 0.3, 0.3,      # A
  1.5, 2, 1.5,        # B
  0.005, 0.005, 0.005, # C
```

```

      3, 3, 3          # D: inflection times D/C = 600
    ), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHSMM with SCPG distribution

obj_ihsmm1_m3 <- ihsmm.scpg(x = NULL, ismc.par, scpg.par)
# =====
# An example of a 2-state IHSMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scpg):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur): the expected
#   sojourn  $1 + \mu_1(t)$  in the complete State 1 increases, and that in the
#   incomplete State 2 decreases, between the levels  $1 + A_j = 2$  and
#    $1 + A_j + B_j = 8$  observations, around the inflection time
#    $D_j/C_j = 500$ ;
# - the Poisson mean  $\lambda_2(t)$  of the TSCPG (rows K, M, R, P of rate)
#   falls from about 5.8 at the start of the record towards  $K = 0.5$ ,
#   around the inflection time  $P/R = 400$ .

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  1, 1,          # A
  6, 6,          # B
  0.005, 0.005, # C
  2.5, 2.5       # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5   # K
rate[2, ] <- 6     # M
rate[3, ] <- 0.005 # R
rate[4, ] <- 2     # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state IHSMM with TSCPG distribution

```

```

obj_ihsmm2 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2)
# =====
# An example of a 3-state IHSM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 8 minutes on a laptop (the
# 2-state model above takes about 50 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur), with
#   inflection times  $D_j/C_j = 500$ : the expected sojourn  $1 + \mu_1(t)$  in the
#   complete State 1 increases from about 2.1 towards 6.5 observations,
#   and those in the incomplete States 2 and 3 decrease from about 5.9
#   towards 1.5;
# - the Poisson means  $\lambda_j(t)$  of the TSCPG (rows K, M, R, P of rate):
#    $\lambda_2(t)$  falls from about 3.3 towards 0.5 around 400, and
#    $\lambda_3(t)$  from about 9.8 towards 4 around 600.
# The zero-diagonal transition matrix gives the next state of each visit.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5, 0.5,      # A
  5, 5, 5,           # B
  0.004, 0.004, 0.004, # C
  2, 2, 2           # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(2.4, 3.6)    # P: inflection times P/R = 400 and 600
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 3-state IHSM with TSCPG distribution

obj_ihsmm2_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2_m3)

```

pscp

*Distribution function of the SCPG and TSCPG distributions***Description**

Evaluates the distribution function of the SCPG distribution, of its time-dependent version (TSCPG), or of a gamma distribution with a fixed number of missing events; see [dscpg](#) for the models.

Usage

```
pscp(
  x,
  rate,
  shape,
  scale,
  rate.fix = FALSE,
  time.scp = FALSE,
  lower.tail = TRUE,
  log.p = FALSE
)
```

Arguments

<code>x</code>	A vector of positive, finite interevent times. For the TSCPG (<code>time.scp = TRUE</code>) they must be in time order.
<code>rate</code>	The number of missing events: for the SCPG, the Poisson mean $\lambda > 0$; for the TSCPG, the vector <code>c(K, M, R, P)</code> of $\lambda(t)$ (see dscpg); with <code>rate.fix = TRUE</code> , the fixed number $n \geq 0$ of missing events.
<code>shape</code>	The gamma shape parameter $\alpha > 0$.
<code>scale</code>	The gamma scale parameter $\beta > 0$.
<code>rate.fix</code>	Logical. If TRUE, exactly rate events are missing between consecutive observed events, and X_t is gamma distributed with shape $(n + 1)\alpha$ and scale β . Default is FALSE.
<code>time.scp</code>	Logical. If TRUE, the TSCPG distribution is used, with <code>rate = c(K, M, R, P)</code> , and <code>rate.fix</code> must be FALSE. Default is FALSE (SCPG).
<code>lower.tail</code>	Logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
<code>log.p</code>	Logical; if TRUE, log probabilities are returned.

Details

Given $N = k + 1$, the observed interevent time is the sum of $k + 2$ gamma interevent times, and is therefore gamma with shape $(k + 2)\alpha$ and scale β . The SCPG distribution function is thus a Poisson mixture of gamma distribution functions,

$$F(x) = \sum_{k=0}^{\infty} e^{-\lambda} \frac{\lambda^k}{k!} G\{x; (k + 2)\alpha, \beta\},$$

where $G(x; a, \beta)$ is the gamma distribution function with shape a and scale β . pscpg evaluates this sum directly, without integrating the density. The Poisson terms that are kept are chosen so that the omitted Poisson mass is below 10^{-12} times the smaller tail. That tail, $P(X \leq x)$ or $P(X > x)$, is summed on the log scale, and the other is obtained as its complement, so the two tails add to one to machine accuracy and far-tail log probabilities stay finite.

With `rate.fix = TRUE`, F is the gamma distribution function with shape $(n+1)\alpha$. With `time.scp = TRUE`, the t th value is the probability for x_t given the preceding observations, that is, the SCPG probability at $\lambda(t)$, which depends on $h_{t-1} = x_1 + \dots + x_{t-1}$. For a record generated by the TSCPG these values are independent and uniform on $(0, 1)$, and `qscpg` applied to them in the same order returns the record.

Value

A numeric vector of probabilities, or log probabilities when `log.p = TRUE`, with the same length and names as `x`.

Examples

```
# =====
# Example of an SCPG distribution
# =====
rate <- 3
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from an SCPG distribution

x_scp <- rscpg(nsim=200, rate, shape, scale, seed=900)
print(x_scp)
# Computing CDF of an SCPG distribution

cdf_scp <- pscpg(x_scp, rate, shape, scale, time.scp=FALSE)
print(cdf_scp)
# =====
# Example of a TSCPG distribution
# =====

# rate[1]=K, rate[2]=M, rate[3]=R, rate[4]=P: lambda(t) falls from about K+M
# to K,
# with inflection time P/R = 300
t_rate <- c(1, 2, 0.02, 6)
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from a TSCPG distribution

x_tscpg <- rscpg(nsim=200, t_rate, shape, scale, time.scp=TRUE, seed=700)
```

```

print(x_tscpg)
# Computing CDF of a TSCPG distribution

cdf_tscpg <- pscpg(x_tscpg, t_rate, shape, scale, time.scpg=TRUE)
print(cdf_tscpg)

```

qscpg

*Quantile function of the SCPG and TSCPG distributions***Description**

Computes quantiles of the SCPG distribution, of its time-dependent version (TSCPG), or of a gamma distribution with a fixed number of missing events; see [dscpg](#) for the models.

Usage

```

qscpg(
  p,
  rate,
  shape,
  scale,
  rate.fix = FALSE,
  time.scpg = FALSE,
  lower.tail = TRUE,
  log.p = FALSE,
  mixture.rtol = 1e-12,
  tol = 1e-10,
  maxiter = 200L
)

```

Arguments

p	A numeric vector of probabilities in $[0, 1]$, or of log probabilities (at most 0) when <code>log.p = TRUE</code> . NA values are allowed.
rate	The number of missing events: for the SCPG, the Poisson mean $\lambda > 0$; for the TSCPG, the vector <code>c(K, M, R, P)</code> of $\lambda(t)$ (see dscpg); with <code>rate.fix = TRUE</code> , the fixed number $n \geq 0$ of missing events.
shape	The gamma shape parameter $\alpha > 0$.
scale	The gamma scale parameter $\beta > 0$.
rate.fix	Logical. If TRUE, exactly rate events are missing between consecutive observed events, and X_t is gamma distributed with shape $(n + 1)\alpha$ and scale β . Default is FALSE.
time.scpg	Logical. If TRUE, the TSCPG distribution is used, with <code>rate = c(K, M, R, P)</code> , and <code>rate.fix</code> must be FALSE. Default is FALSE (SCPG).
lower.tail	Logical; if TRUE (default), p is $P(X \leq x)$, otherwise $P(X > x)$.

log.p	Logical; if TRUE, p is given as log probabilities.
mixture.rtol	A number in (0, 1): the Poisson terms omitted from the mixture in Details have total probability below mixture.rtol times the tail being inverted. Default is 10^{-12} .
tol	The tolerance of uniroot on $\log(x/\beta)$, so approximately a relative tolerance on the quantile. Default is 10^{-10} .
maxiter	The maximum number of uniroot iterations. Default is 200.

Details

The quantile is $Q(p) = \min\{x : F(x) \geq p\}$, where F is the distribution function of [pscpg](#): for the SCPG, the Poisson mixture of gamma distribution functions

$$F(x) = \sum_{k=0}^{\infty} e^{-\lambda} \frac{\lambda^k}{k!} G\{x; (k+2)\alpha, \beta\}.$$

There is no closed form, so $F(x) = p$ is solved numerically. The Poisson terms outside a window are omitted, with total probability below mixture.rtol times the tail being inverted, and the equation is solved for $\log(x/\beta)$ with [uniroot](#), working with the smaller of the two tails on the log scale so that extreme probabilities keep full accuracy. With rate.fix = TRUE the quantile is that of the gamma distribution with shape $(n+1)\alpha$.

With time.scpg = TRUE the quantiles are computed in order, each one at the Poisson mean $\lambda(t)$ set by the quantiles before it: starting from $h_0 = 0$,

$$q_t = Q_{\lambda(t)}(p_t), \quad h_t = h_{t-1} + q_t,$$

with $\lambda(t)$ as in [dscpg](#). The order of p therefore matters, and qscpg inverts [pscp](#) applied to a record in time order.

Value

A numeric vector of quantiles with the same length and names as p. $P(X \leq x) = 0$ gives 0 and $P(X \leq x) = 1$ gives Inf. Without time.scpg, NA values of p give NA. With time.scpg = TRUE, the quantiles after an NA or an infinite quantile cannot be computed and are NA.

Examples

```
# =====
# Example of an SCPG distribution
# =====
rate <- 3
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from an SCPG distribution

x_scp <- rscpg(nsim=200, rate, shape, scale, seed=900)
```

```

print(x_scp)
# Computing CDF of an SCPG distribution

cdf_scp <- pscpg(x_scp, rate, shape, scale, time.scp=FALSE)
print(cdf_scp)
# Calculating quantiles of an SCPG distribution

q1_scp <- qscpg(cdf_scp, rate, shape, scale, time.scp=FALSE)
print(q1_scp)
print(x_scp)

q2_scp <- qscpg(seq(0.1, 0.9, 0.1), rate, shape, scale, time.scp=FALSE)
print(q2_scp)

# A very small lower-tail probability supplied on the log scale

qscpg(-100, rate, shape, scale, log.p = TRUE)

# Upper-tail quantile:  $P(X > q) = 1e-8$ 

qscpg(log(1e-8), rate, shape, scale, lower.tail = FALSE, log.p = TRUE)
# =====
# Example of a TSCPG distribution
# =====

# rate[1]=K, rate[2]=M, rate[3]=R, rate[4]=P: lambda(t) falls from about K+M
# to K,
# with inflection time P/R = 300
t_rate <- c(1, 2, 0.02, 6)
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from a TSCPG distribution

x_tscpg <- rscpg(nsim=200, t_rate, shape, scale, time.scp=TRUE, seed=700)
print(x_tscpg)
# Computing CDF of a TSCPG distribution

cdf_tscpg <- pscpg(x_tscpg, t_rate, shape, scale, time.scp=TRUE)
print(cdf_tscpg)
# Quantiles for a TSCPG distribution

qt1_scp <- qscpg(cdf_tscpg, t_rate, shape, scale, time.scp=TRUE)
print(qt1_scp)
print(x_tscpg)

qt2_scp <- qscpg(seq(0.1, 0.5, 0.1), t_rate, shape, scale, time.scp=TRUE)
print(qt2_scp)

```

rscpg

*Random number generation from an SCPG or a TSCPG distribution***Description**

Generates random numbers from an SCPG or a TSCPG distribution.

Usage

```
rscpg(
  nsim,
  rate,
  shape,
  scale,
  rate.fix = FALSE,
  shape.fix = FALSE,
  time.scpg = FALSE,
  seed = NULL
)
```

Arguments

nsim	is the number of points to be simulated for random number generator.
rate	The number of missing events: for the SCPG, the Poisson mean $\lambda > 0$; for the TSCPG, the vector $c(K, M, R, P)$ of $\lambda(t)$ (see dscpg); with <code>rate.fix = TRUE</code> , the fixed number $n \geq 0$ of missing events.
shape	The gamma shape parameter $\alpha > 0$.
scale	The gamma scale parameter $\beta > 0$.
rate.fix	Logical. If TRUE, exactly rate events are missing between consecutive observed events, and X_t is gamma distributed with shape $(n + 1)\alpha$ and scale β . Default is FALSE.
shape.fix	Logical; used only by fit.scpg , where TRUE holds the shape at the supplied value instead of estimating it. Default is FALSE.
time.scpg	Logical. If TRUE, the TSCPG distribution is used, with <code>rate = c(K, M, R, P)</code> , and <code>rate.fix</code> must be FALSE. Default is FALSE (SCPG).
seed	is seed for random number generator. Default is NULL.

Details

Simulation from the SCPG distribution follows a two-step process: first, a random number N is generated from a shifted Poisson distribution with parameter λ ; next, $N + 1$ random numbers $y_1, y_2, \dots, y_{N+1}$ are generated from a gamma distribution with the specified shape and scale parameters and summed to produce a single draw $x = \sum_{i=1}^{N+1} y_i$. This procedure is repeated independently to generate `nsim` observations, and an identical procedure applies to the TSCPG distribution.

Under `rate.fix = TRUE` the counting stage is omitted. The number of missing events being fixed, the additive property of the gamma distribution gives each draw directly as a gamma variate with shape $(N + 1)\alpha$ and scale β .

Under `time.scpg = TRUE` the draws are independent but not identically distributed, the rate governing the i th draw being

$$\lambda(t) = K + M / \{1 + \exp(Rh_{t-1} - P)\}$$

evaluated at the occurrence time accumulated from the draws preceding it. Generation is therefore recursive: an interevent time is drawn, the occurrence time advanced, and the rate re-evaluated. The realization so obtained is an ordered process in which the expected number of missing events declines along the sequence, and its elements are to be retained in the order returned. Specifying `seed` initializes the random number generator and renders the realization reproducible.

Value

returns a sequence of `nsim` observations from an SCPG or a TSCPG distribution.

Examples

```
# =====
# Example of an SCPG distribution
# =====
rate <- 3
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from an SCPG distribution

x_scpg <- rscpg(nsim=200, rate, shape, scale, seed=900)
print(x_scpg)
# =====
# Example of a TSCPG distribution
# =====

# rate[1]=K, rate[2]=M, rate[3]=R, rate[4]=P: lambda(t) falls from about K+M
# to K,
# with inflection time P/R = 300
t_rate <- c(1, 2, 0.02, 6)
shape <- 2
scale <- 0.5
rate.fix <- FALSE
shape.fix <- FALSE

# Simulating from a TSCPG distribution

x_tscpg <- rscpg(nsim=200, t_rate, shape, scale, time.scpg=TRUE, seed=700)
print(x_tscpg)
```

sim.hmm.scpg	<i>Simulation from an m-state HMM with gamma, SCPG or TSCPG renewal processes</i>
--------------	---

Description

Simulates an incomplete record, and its hidden states, from an HMM of class `hmm.gamma`, `hmm.scpg` or `hmm.tscpg` created by [hmm.scpg](#).

Usage

```
sim.hmm.scpg(hmmobj, nsim = 1, seed = NULL)
```

Arguments

<code>hmmobj</code>	An object created by hmm.scpg .
<code>nsim</code>	The number of interevent times to simulate.
<code>seed</code>	Seed for the random number generator, or NULL (default).

Details

The record is generated in time order. The first state is drawn from the initial distribution (the stationary distribution when `stationary = TRUE`); at each step an interevent time is drawn for the current state, as in [rscpg](#): a gamma interevent time in State 1, and in States $2, \dots, m$ the sum of $N + 1$ gamma interevent times with N missing events. The next state is then drawn from the row of the transition matrix for the current state. For the TSCPG, the Poisson mean $\lambda_j(t)$ of each step uses the occurrence time h_{t-1} reached so far.

Only the observed interevent times are returned, so `x` is an incomplete record of the kind the models describe; the missing events themselves are not recorded. The hidden states are returned to assess decoding and parameter recovery, as in the simulation studies of Shahzadi et al. (2026).

Value

An object of the same class as `hmmobj`: a list with the simulated interevent times `x`, the hidden states `states`, the number of transitions of the chain `n_transitions` (`nsim - 1`), and `mc.par` and `scpg.par` of `hmmobj`. It can be passed directly to [fit.hmm.scpg](#).

References

Shahzadi A, Wang T, Parry M, Bebbington M (2026). “Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models.” *Advances in Data Analysis and Classification*, **20**, 663-691. doi:10.1007/s1163402500632x.

Examples

```
# =====
# An example of a 2-state HMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter lambda_2(t), where
#   lambda_2(t) = K + M / (1 + exp(R * h_{t-1} - P))
# falls from about 5.3 at the start of the record towards K = 0.5, around
# the inflection time P/R = 300 of the last occurrence time h_{t-1}. The
# hidden chain is homogeneous, so only lambda_2(t) changes along the record.

m <- 2
tpm <- matrix(c(0.9, 0.1,
                0.2, 0.8), nrow = 2, byrow = TRUE)
mc.par <- list(initd = NULL, tpm = tpm, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5   # K
rate[2, ] <- 5     # M
rate[3, ] <- 0.01  # R
rate[4, ] <- 3     # P: inflection time P/R = 300
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBsum.method = 'direct.evaluation')

# Defining an object for 2-state HMM with TSCPG distribution

obj_hmm2 <- hmm.scpg(x = NULL, mc.par, scpg.par)
class(obj_hmm2)
# Simulating a 2-state HMM with TSCPG distribution

sim_hmm2 <- sim.hmm.scpg(obj_hmm2, nsim = 200, seed = 1)
str(sim_hmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hmm2$x[-200]), h_T = sum(sim_hmm2$x))
# =====
# An example of a 3-state HMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 30 seconds on a laptop (the
# 2-state model above takes about 4 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(1/3, 1/3, 1/3)
tpm <- matrix(c(0.2, 0.65, 0.15, 0.50, 0.30, 0.20, 0.25, 0.50, 0.25),
              nrow = 3, byrow = TRUE)
mc.par <- list(initd=NULL, tpm=tpm, stationary=TRUE)
```

```

rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(1, 1) # K
rate[2, ] <- c(1.5, 2.5) # M
rate[3, ] <- c(0.02, 0.015) # R
rate[4, ] <- c(3, 3) # P: inflection times P/R = 150 and 200
shape <- 1
scale <- 1.7
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
                 WBsum.method='direct.evaluation')

# Defining an object for 3-state HMM with TSCPG distribution
obj_hmm2_m3 <- hmm.scpg(x = NULL, mc.par, scpg.par)
# Simulating a 3-state HMM with TSCPG distribution

sim_hmm2_m3 <- sim.hmm.scpg(obj_hmm2_m3, nsim=100, seed=9)

```

sim.hsmm.scpg	<i>Simulation from an m-state HSMM with gamma or SCPG or TSCPG renewal processes</i>
---------------	--

Description

Generates a sequence of observed process along with hidden state sequence with class `hsmm.gamma`, `hsmm.scpg` and `hsmm.tscpg` as typically returned by [hsmm.scpg](#).

Usage

```
sim.hsmm.scpg(hsmmobj, nsim = 1, seed = NULL)
```

Arguments

<code>hsmmobj</code>	is a model object with class <code>hsmm.gamma</code> , <code>hsmm.scpg</code> and <code>hsmm.tscpg</code> as typically returned by hsmm.scpg .
<code>nsim</code>	is the number of points to be simulated for random number generator.
<code>seed</code>	is seed for random number generator. Default is <code>NULL</code> .

Details

A semi-Markov chain is characterized by its realization occurring visit by visit, as opposed to at each discrete time step, with explicit sojourn times. A state is drawn from the initial distribution; its sojourn time is drawn from the shifted Poisson distribution $p_j(d) = \mu_j^{d-1} \exp(-\mu_j) / (d-1)!$, so that every visit occupies at least one observation; the observations of that visit are drawn from the state-dependent renewal process; and the succeeding state is drawn from the row of the embedded transition probability matrix indexed by the state just left. Since that matrix has a zero diagonal, consecutive visits are necessarily to distinct states.

Generation terminates once `nsim` observations have been accumulated, in general during a visit. The realization is thus right-censored in the sense assumed by the likelihood evaluated in `fit.hsmm.scpG`, which permits a simulation study to assess the censored likelihood as it is applied in practice.

The number of observations and the number of visits constitute distinct sample sizes. A state whose expected sojourn time is large relative to `nsim` may be visited only a few times irrespective of the number of observations generated, and it is the number of visits that determines the precision with which the embedded chain can be recovered. Specifying `seed` renders the realization reproducible.

Value

The returned object is an `hsmm.scpG` with the same components as the input arguments in addition to the simulated sequence x for the observed process and states, which is the corresponding sequence of states of the underlying semi-Markov chain. This also provides the information about the number of transitions for SMC.

Examples

```
# =====
# An example of a 2-state HSMM with an SCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times with shape
# alpha); in the incomplete State 2 the number N of events missing between
# two observed events is shifted Poisson with parameter lambda_2, so the
# observed events are N + 1 such gamma interevent times apart. The sojourns
# are shifted Poisson with
# parameters dur (expected sojourns 1 + dur = 5 and 4 observations). With
# m = 2 the zero-diagonal transition matrix is fixed: the chain alternates
# between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = FALSE, time.scpG = FALSE,
                 WBsum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with SCPG distribution

obj_hsmm1 <- hsmm.scpG(x = NULL, smc.par, scpg.par)
class(obj_hsmm1)
# Simulating a 2-state HSMM with SCPG distribution

sim_hsmm1 <- sim.hsmm.scpG(obj_hsmm1, nsim = 150, seed = 1)
str(sim_hsmm1[c("x", "states")])
```

```

# Number of visits to each state in the simulated path
table(rle(sim_hsmm1$states)$values)
# =====
# An example of a 3-state HSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 15 seconds on a laptop (the
# 2-state model above takes about 1 second), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.3, 0.7, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd=NULL, tpm=tpm, dur=c(3, 5, 2),
               dur.lb=NA, dur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
               rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
               WBsum.method='direct.evaluation')

# Defining an object of 3-state HSMM with SCPG distribution

obj_hsmm1_m3 <- hsmm.scp(x = NULL, smc.par, scpg.par)
# Simulating a 3-state HSMM with SCPG distribution

sim_hsmm1_m3 <- sim.hsmm.scp(obj_hsmm1_m3, nsim = 80, seed = 1)
print(sim_hsmm1_m3)
# =====
# An example of a 2-state HSMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter  $\lambda_2(t)$ , where
#  $\lambda_2(t) = K + M / (1 + \exp(R * h_{t-1} - P))$ 
# falls from about 5.8 at the start of the record towards  $K = 0.5$ , around
# the inflection time  $P/R = 400$  of the last occurrence time  $h_{t-1}$ . The
# sojourns are shifted Poisson with parameters dur (expected sojourns
#  $1 + \text{dur} = 5$  and 4 observations). With  $m = 2$  the zero-diagonal transition
# matrix is fixed: the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5 # K
rate[2, ] <- 6 # M
rate[3, ] <- 0.005 # R

```

```

rate[4, ] <- 2      # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with TSCPG distribution

obj_hsmm2 <- hsmm.scp(x = NULL, smc.par, scpg.par)
class(obj_hsmm2)
# Simulating a 2-state HSMM with a TSCPG distribution

sim_hsmm2 <- sim.hsmm.scp(obj_hsmm2, nsim = 300, seed = 3)
str(sim_hsmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2$x[-300]), h_T = sum(sim_hsmm2$x))
# =====
# An example of a 3-state HSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 7 seconds), so the examples that fit
# it are wrapped in donttest.

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete States 2 and 3 the number of missing events N between two
# observed events is shifted Poisson with parameter  $\lambda_j(t)$ , where
#  $\lambda_j(t) = K_j + M_j / (1 + \exp(R_j * h_{t-1} - P_j))$ 
# falls around the inflection time  $P_j/R_j$  of the last occurrence time
#  $h_{t-1}$ :  $\lambda_2(t)$  from about 3.4 towards 0.5 around 600, and
#  $\lambda_3(t)$  from about 10 towards 4 around 900. The sojourns are shifted
# Poisson with parameters dur (expected sojourns 1 + dur = 5, 4 and 4
# observations), and the zero-diagonal transition matrix gives the next
# state of each visit. The shape 2 makes the interevent times of the three
# states less overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
               0.5, 0, 0.5,
               0.6, 0.4, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3.6, 5.4)    # P: inflection times P/R = 600 and 900
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,

```

```

WBsum.method = 'direct.evaluation')

# Defining an object of 3-state HSMM with TSCPG distribution

obj_hsmm2_m3 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm2_m3)
# Simulating a 3-state HSMM with a TSCPG distribution

sim_hsmm2_m3 <- sim.hsmm.scpg(obj_hsmm2_m3, nsim = 400, seed = 3)
str(sim_hsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rates are evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2_m3$x[-400]), h_T = sum(sim_hsmm2_m3$x))

# Number of visits to each state in the simulated path
table(rle(sim_hsmm2_m3$states)$values)

```

sim.ihmm.scpg	<i>Simulation from an m-state IHMM with gamma, SCPG and TSCPG renewal processes</i>
---------------	---

Description

Generates a sequence of observed process along with hidden state sequence with class `ihmm.gamma`, `ihmm.scpg` and `ihmm.tscpg` as typically returned by `ihmm.scpg`.

Usage

```
sim.ihmm.scpg(ihmmobj, nsim = 1, seed = NULL)
```

Arguments

<code>ihmmobj</code>	a model object with class <code>ihmm.gamma</code> , <code>ihmm.scpg</code> and <code>ihmm.tscpg</code> as typically returned by <code>ihmm.scpg</code> .
<code>nsim</code>	The number of interevent times to simulate.
<code>seed</code>	Seed for the random number generator, or NULL (default).

Details

Realizations are generated as in `sim.hmm.scpg`, with the difference that each state is drawn from the row of the transition probability matrix evaluated at the occurrence time accumulated from the preceding observations rather than from a fixed matrix. The hidden chain and the observations are therefore generated jointly and in time order, each state drawn advancing the occurrence time that determines the transition probabilities governing the next. The initial state is drawn from π , or from the stationary distribution of the transition probability matrix at time zero where the model is specified as stationary.

The resulting realization exhibits the time inhomogeneity for which the specification was introduced: the earlier portion of the record, in which the transition probabilities into State 1 are small,

is dominated by the incompleteness states, while the later portion is increasingly confined to State 1. Simulation therefore provides a direct means of assessing whether a given set of coefficients r_{ij} and w_{ij} locates the change within the intended span. Specifying seed renders the realization reproducible.

Value

The returned object is an `ihmm.scpg` with the same components as input arguments in addition to the simulated sequence x for the observed process and states is the corresponding sequence of hidden states of inhomogeneous Markov chain. This also provides the information about the number of transitions for IMC.

Examples

```
# =====
# An example of a 2-state IHMM with an SCPG renewal process
# =====

# The transition probabilities a_12(t) and a_22(t) into the incomplete
# State 2 decrease along the record: the log-odds of moving from state i to
# State 2 rather than to State 1 is w_i2 - r_i2 * h_{t-1}, which is zero at
# the crossing time w_i2 / r_i2 (250 for a_12(t) and 500 for a_22(t)). In
# State 2 the number N of events missing between two observed events is
# shifted Poisson with parameter lambda_2. The chain starts mostly in State 2
# (initd = (0.2, 0.8)),
# since the early part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.004, 0.006) # rates of change r_12, r_22
ttpm[, , 2] <- c(1, 3)         # intercepts w_12, w_22
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHMM with SCPG distribution

obj_ihmm1 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm1)
# Simulating a 2-state IHMM with SCPG distribution

sim_ihmm1 <- sim.ihmm.scpg(obj_ihmm1, nsim = 200, seed = 1)
str(sim_ihmm1[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm1$states,
```

```

    half = ifelse(seq_along(sim_ihmm1$states) <= 100, "first", "second"))
# =====
# An example of a 3-state IHMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 6 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(0.45, 0.25, 0.30)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.0008, 0.0010, 0.0006,
                       0.0006, 0.0008, 0.0010), nrow = m, ncol = m-1)
# intercepts w_ij; a_ij(t) = a_i1(t) at h = w_ij/r_ij, between 2000 and 3000
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(2500, 2000, 3000,
                                       2000, 3000, 2500), nrow = m, ncol = m-1)

stationary <- FALSE
imc.par <- list(initd=initd, ttpm=ttpm, stationary=stationary)
rate <- matrix(c(7, 2), nrow = 1, ncol = m-1)
shape <- 1.7
scale <- 3.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

# Defining an object of 3-state IHMM with SCPG distribution

obj_ihmm1_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
# Simulating a 3-state IHMM with SCPG distribution

sim_ihmm1_m3 <- sim.ihmm.scpg(obj_ihmm1_m3, nsim = 200, seed = 2)
print(sim_ihmm1_m3)
# =====
# An example of a 2-state IHMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scpg):
# - the transition probabilities a_12(t) and a_22(t) into the incomplete
#   State 2 decrease: the log-odds of moving from state i to State 2
#   rather than to State 1 is w_i2 - r_i2 * h_{t-1}, which is zero at the
#   crossing time w_i2 / r_i2 (about 304 for a_12(t) and 600 for a_22(t));
# - the Poisson mean lambda_2(t) of the TSCPG falls from about 5.4 at the
#   start of the record towards K = 0.5, around the inflection time
#   P/R = 400.
# The chain starts mostly in State 2 (initd = (0.2, 0.8)), since the early
# part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))

```

```

ttpm[, , 1] <- c(0.0028, 0.005) # rates of change r_12, r_22
ttpm[, , 2] <- c(0.85, 3) # intercepts w_12, w_22
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(c(0.5, 5, 0.01, 4), nrow = 4) # K, M, R, P: P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scp=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHMM with TSCPG distribution

obj_ihmm2 <- ihmm.scp(x = NULL, imc.par, scpg.par)
class(obj_ihmm2)
# Simulating a 2-state IHMM with a TSCPG distribution

sim_ihmm2 <- sim.ihmm.scp(obj_ihmm2, nsim = 200, seed = 2)
str(sim_ihmm2[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm2$states,
      half = ifelse(seq_along(sim_ihmm2$states) <= 100, "first", "second"))
# =====
# An example of a 3-state IHMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2.5 minutes on a laptop (the
# 2-state model above takes about 8 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scp):
# - the transition probabilities a_ij(t): moves into the incomplete
#   States 2 and 3 become less likely as time goes on. The log-odds of
#   moving from state i to state j rather than to State 1 is
#   w_ij - r_ij * h_{t-1}, which is zero at the crossing time w_ij / r_ij;
# - the Poisson means lambda_j(t) of the TSCPG: in the incomplete states
#   fewer events are missed after the inflection times P_j / R_j.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1, so that the states are easier to tell apart.

m <- 3
initd <- c(0.4, 0.3, 0.3)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.006, 0.006, 0.006,
                       0.006, 0.006, 0.006), nrow = m, ncol = m-1)
# intercepts w_ij; crossing times w_ij/r_ij between 400 and 600
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(500, 600, 500,
                                       400, 500, 600), nrow = m, ncol = m-1)
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(NA, nrow = 4, ncol = m-1)

```

```

rate[1, ] <- c(0.5, 4)      # K: lambda_2(t) falls from about 3.4 towards
rate[2, ] <- c(3, 6)        # M: 0.5, lambda_3(t) from about 10 towards 4
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3, 4.2)      # P: inflection times P/R = 500 and 700
shape <- 2
scale <- 0.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHMM with TSCPG distribution

obj_ihmm2_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm2_m3)
# Simulating a 3-state IHMM with a TSCPG distribution

sim_ihmm2_m3 <- sim.ihmm.scpg(obj_ihmm2_m3, nsim = 500, seed = 1)
str(sim_ihmm2_m3[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into the incomplete states become less likely
table(state = sim_ihmm2_m3$states,
      half = ifelse(seq_along(sim_ihmm2_m3$states) <= 250, "first", "second"))

```

sim.ihsmm.scpg	<i>Simulation from an m-state IHSMM with gamma, SCPG and TSCPG renewal processes</i>
----------------	--

Description

Generates a sequence of observed process along with hidden state sequence with class `ihsmm.gamma`, `ihsmm.scpg` and `ihsmm.tscpg` as typically returned by [ihsmm.scpg](#).

Usage

```
sim.ihsmm.scpg(ihsmmobj, nsim = 1, seed = NULL)
```

Arguments

<code>ihsmmobj</code>	a model object with class <code>ihsmm.gamma</code> , <code>ihsmm.scpg</code> and <code>ihsmm.tscpg</code> as typically returned by ihsmm.scpg .
<code>nsim</code>	The number of interevent times to simulate.
<code>seed</code>	Seed for the random number generator, or NULL (default).

Details

Realizations are generated visit by visit as in `sim.ihsmm.scpg`, the succeeding state being drawn from the zero-diagonal embedded transition probability matrix and the observations of each visit from the state-dependent renewal process, with the difference that the sojourn time of a visit is drawn from the shifted Poisson distribution with parameter $\mu_j(t)$ evaluated at the time t at which that visit commences. A given state is therefore governed by different sojourn distributions according to the point of the record at which it is entered.

Since the expected sojourn time in State 1 increases with elapsed time while those in States 2, $\dots$, m decrease, realizations display a characteristic drift: short and frequent visits to State 1 in the earlier portion of the record, giving way to extended visits towards its end. Simulation provides a direct means of assessing whether a given `tdur` locates this drift within the intended span.

Generation terminates once `nsim` observations have been accumulated, so that the sojourn in the last visited state is right-censored, in accordance with the likelihood evaluated in `fit.ihsmm.scpg`. Specifying `seed` renders the realization reproducible.

Value

The returned object is an `ihsmm.scpg` with the same components as input arguments in addition to the simulated sequence x for the observed process and `states` is the corresponding sequence of hidden states of inhomogeneous Markov chain. This also provides the information about the number of transitions of ISMC.

Examples

```
# =====
# An example of a 2-state IHSMM with an SCPG renewal process
# =====

# The sojourn parameters mu_j(t) depend on the time at which a visit
# commences (rows A, B, C, D of tdur): the expected sojourn 1 + mu_1(t) in
# the complete State 1 increases, and that in the incomplete State 2
# decreases, between the levels 1 + A_j = 1.5 and 1 + A_j + B_j = 5.5
# observations, around the inflection time D_j/C_j = 500. In State 2,
# the number N of events missing between two observed events is shifted
# Poisson with parameter lambda_2.
# With m = 2 the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5,      # A
  4, 4,          # B
  0.005, 0.005,  # C
  2.5, 2.5       # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
shape <- 1.5
```

```

scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHSMM with SCPG distribution

obj_ihsmm1 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm1)
# Simulating a 2-state IHSMM with SCPG distribution

sim_ihsmm1 <- sim.ihsmm.scp(obj_ihsmm1, nsim = 150, seed = 3)
str(sim_ihsmm1[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm1$states)$values)
# =====
# An example of a 3-state IHSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 30 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.5, 0.5, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.3, 0.3, 0.3,      # A
  1.5, 2, 1.5,        # B
  0.005, 0.005, 0.005, # C
  3, 3, 3            # D: inflection times D/C = 600
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHSMM with SCPG distribution

obj_ihsmm1_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
# Simulating a 3-state IHSMM with SCPG distribution

sim_ihsmm1_m3 <- sim.ihsmm.scp(obj_ihsmm1_m3, nsim = 150, seed = 1)
print(sim_ihsmm1_m3)
# =====
# An example of a 2-state IHSMM with a TSCPG renewal process

```

```

# =====

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scpg):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur): the expected
#   sojourn 1 +  $\mu_1(t)$  in the complete State 1 increases, and that in the
#   incomplete State 2 decreases, between the levels 1 +  $A_j = 2$  and
#   1 +  $A_j + B_j = 8$  observations, around the inflection time
#    $D_j/C_j = 500$ ;
# - the Poisson mean  $\lambda_2(t)$  of the TSCPG (rows K, M, R, P of rate)
#   falls from about 5.8 at the start of the record towards  $K = 0.5$ ,
#   around the inflection time  $P/R = 400$ .

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  1, 1,      # A
  6, 6,      # B
  0.005, 0.005, # C
  2.5, 2.5    # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5    # K
rate[2, ] <- 6      # M
rate[3, ] <- 0.005  # R
rate[4, ] <- 2      # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state IHSMM with TSCPG distribution

obj_ihsmm2 <- ihsmm.scpg(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2)
# Simulating a 2-state IHSMM with a TSCPG distribution

sim_ihsmm2 <- sim.ihsmm.scpg(obj_ihsmm2, nsim = 200, seed = 1)
str(sim_ihsmm2[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2$states)$values)
# =====
# An example of a 3-state IHSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 8 minutes on a laptop (the
# 2-state model above takes about 50 seconds), so the examples that fit

```

```

# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scpg):
# - the sojourn parameters mu_j(t) (rows A, B, C, D of tdur), with
#   inflection times D_j/C_j = 500: the expected sojourn 1 + mu_1(t) in the
#   complete State 1 increases from about 2.1 towards 6.5 observations,
#   and those in the incomplete States 2 and 3 decrease from about 5.9
#   towards 1.5;
# - the Poisson means lambda_j(t) of the TSCPG (rows K, M, R, P of rate):
#   lambda_2(t) falls from about 3.3 towards 0.5 around 400, and
#   lambda_3(t) from about 9.8 towards 4 around 600.
# The zero-diagonal transition matrix gives the next state of each visit.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5, 0.5,      # A
  5, 5, 5,            # B
  0.004, 0.004, 0.004, # C
  2, 2, 2            # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)        # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(2.4, 3.6)    # P: inflection times P/R = 400 and 600
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 3-state IHSMM with TSCPG distribution

obj_ihsmm2_m3 <- ihsmm.scpg(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2_m3)
# Simulating a 3-state IHSMM with a TSCPG distribution

sim_ihsmm2_m3 <- sim.ihsmm.scpg(obj_ihsmm2_m3, nsim = 250, seed = 2)
str(sim_ihsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the time-dependent components are evaluated
# on [0, h_{T-1}]
c(h_T1 = sum(sim_ihsmm2_m3$x[-250]), h_T = sum(sim_ihsmm2_m3$x))

# The duration parameters are informed by the visits, one sojourn each:

```

```
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2_m3$states)$values)
```

sp.hmm.scpg

Smoothed probabilities and residuals for an m-state HMM with gamma, SCPG or TSCPG renewal processes

Description

Computes the probability of each hidden state at each time, given the whole record, and the rescaled times used for residual analysis, for an HMM of class `hmm.gamma`, `hmm.scpg` or `hmm.tscpg`.

Usage

```
sp.hmm.scpg(hmmobj)
```

Arguments

`hmmobj` An object created by `hmm.scpg`, usually the result of `fit.hmm.scpg`, with the data attached as `hmmobj$x`.

Details

Smoothed probabilities. $\gamma_t(j) = \Pr(S_t = j \mid x_1, \dots, x_T)$ is computed by the forward-backward algorithm (Baum et al., 1970),

$$\gamma_t(j) = \alpha_t(j)\beta_t(j)/L,$$

with the forward variables $\alpha_t(j)$ of `fit.hmm.scpg` and the backward variables $\beta_T(i) = 1$ and $\beta_t(i) = \sum_j a_{ij} f_j(x_{t+1}) \beta_{t+1}(j)$. Both are rescaled at each step on the log scale. Choosing at each t the state with the largest $\gamma_t(j)$ (local decoding) can give a sequence that the model does not allow; `vip.hmm.scpg` gives the most probable sequence as a whole. The smoothed probabilities show how certain the state assignment is at each time.

Residuals. If a point process model is correct, its integrated intensity at the occurrence times, $\tau_k = \Lambda(h_k)$, forms a Poisson process of unit rate (Berman, 1983; Ogata, 1988; Daley and Vere-Jones, 2003, Theorem 7.4.I). Since the state is hidden, the rescaled times are computed by weighting the integrated hazard of each state by the smoothed probabilities,

$$\tau_k = \sum_{t=1}^k \sum_{j=1}^m \gamma_t(j) \{-\log S_j(x_t)\},$$

where S_j is the survival function in State j . To check the fit, plot the cumulative number of events minus the rescaled time, $k - \tau_k$, against τ_k : for a well-fitting model it stays within the Kolmogorov-Smirnov bands $\pm 1.36\sqrt{T}$ (95 per cent) and $\pm 1.63\sqrt{T}$ (99 per cent) (Wang et al., 2012). Because the weights $\gamma_t(j)$ use the whole record, the individual increments $\tau_k - \tau_{k-1}$ are not exactly unit exponential even for the true model, so they should not be tested one by one. The check assesses the fitted model as a whole, not the decoded states.

Value

A list with smoothed.prob, the $T \times m$ matrix of $\gamma_t(j)$, and residuals, the rescaled times $\tau_1, \dots, \tau_T$.

References

- Baum LE, Petrie T, Soules G, Weiss N (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." *The Annals of Mathematical Statistics*, **41**(1), 164-171. doi:10.1214/aoms/1177697196.
- Berman M (1983). "Comment on "Likelihood analysis of point processes and its applications to seismological data" by Ogata." *Bulletin of the International Statistical Institute*, **50**, 412-418.
- Ogata Y (1988). "Statistical models for earthquake occurrences and residual analysis for point processes." *Journal of the American Statistical Association*, **83**(401), 9-27.
- Wang T, Bebbington MS, Harte D (2012). "Markov-modulated Hawkes process with stepwise decay." *Annals of the Institute of Statistical Mathematics*, **64**, 521-544. doi:10.1007/s10463010-03207.
- Daley DJ, Vere-Jones D (2003). *An Introduction to the Theory of Point Processes*. Springer, New York. doi:10.1007/b97277.

Examples

```
# =====
# An example of a 2-state HMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter lambda_2(t), where
#   lambda_2(t) = K + M / (1 + exp(R * h_{t-1} - P))
# falls from about 5.3 at the start of the record towards K = 0.5, around
# the inflection time P/R = 300 of the last occurrence time h_{t-1}. The
# hidden chain is homogeneous, so only lambda_2(t) changes along the record.

m <- 2
tpm <- matrix(c(0.9, 0.1,
                0.2, 0.8), nrow = 2, byrow = TRUE)
mc.par <- list(initd = NULL, tpm = tpm, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5   # K
rate[2, ] <- 5     # M
rate[3, ] <- 0.01  # R
rate[4, ] <- 3     # P: inflection time P/R = 300
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                WBSum.method = 'direct.evaluation')

# Defining an object for 2-state HMM with TSCPG distribution

obj_hmm2 <- hmm.scpG(x = NULL, mc.par, scpg.par)
```

```

class(obj_hmm2)
# Simulating a 2-state HMM with TSCPG distribution

sim_hmm2 <- sim.hmm.scpg(obj_hmm2, nsim = 200, seed = 1)
str(sim_hmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hmm2$x[-200]), h_T = sum(sim_hmm2$x))
# Fitting a 2-state HMM with TSCPG distributions
# (the parameters of obj_hmm2 are the starting values)

fit_hmm2 <- fit.hmm.scpg(sim_hmm2)
# transition matrix (true rows 0.9, 0.1 and 0.2, 0.8)
fit_hmm2$mc.par$tpm
fit_hmm2$scpg.par$rate[, 1] # K, M, R, P of lambda_2(t)
fit_hmm2$scpg.par$scale      # true 1
fit_hmm2$logLik
# Smoothed state probabilities and residuals for the fitted 2-state HMM
# with TSCPG distribution

fit_hmm2$x <- sim_hmm2$x
sp_hmm2 <- sp.hmm.scpg(fit_hmm2)

# Poisson mean lambda_2(t), fitted and true, at the last occurrence times
# h_{t-1}, and the four quarters of the record
lambda_t <- function(r, h) r[1] + r[2] / (1 + exp(r[3] * h - r[4]))
h_hmm2 <- c(0, cumsum(sim_hmm2$x)[-200])
lam_hmm2 <- lambda_t(fit_hmm2$scpg.par$rate[, 1], h_hmm2)
lam0_hmm2 <- lambda_t(obj_hmm2$scpg.par$rate[, 1], h_hmm2)
gam_hmm2 <- sp_hmm2$smoothed.prob
q_hmm2 <- cut(seq_along(h_hmm2), 4)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + Pr(S_t = 2 | x) / (2 + lambda_2(t)),
# averaged over each quarter (row p), with its two parts: the share of
# State 1 (row state.1) and the proportion recorded while the chain is in
# State 2 (row state.2). The true proportion (row p.true) is 1 in State 1
# and 1/(2 + lambda_2(t)) in State 2. p_t follows the true proportion. In an
# HMM the transition probabilities do not depend on time, so the share of
# State 1 varies only by chance; the proportion recorded in State 2 rises
# from quarter to quarter as lambda_2(t) falls.
p_hmm2 <- gam_hmm2[, 1] + gam_hmm2[, 2] / (2 + lam_hmm2)
p0_hmm2 <- ifelse(sim_hmm2$states == 1, 1, 1 / (2 + lam0_hmm2))
rbind(p = tapply(p_hmm2, q_hmm2, mean),
      state.1 = tapply(gam_hmm2[, 1], q_hmm2, mean),
      state.2 = tapply(gam_hmm2[, 2] / (2 + lam_hmm2), q_hmm2, sum) /
        tapply(gam_hmm2[, 2], q_hmm2, sum),
      p.true = tapply(p0_hmm2, q_hmm2, mean))

# Plotting of residuals to check whether the fitted HMM with TSCPG renewal
# process is a good fit

dev.new()

```

```

res_hmm2 <- sp_hmm2$residuals
lres_hmm2 <- length(sp_hmm2$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_hmm2, 1:lres_hmm2-res_hmm2, type="l", xlab="Transformed times",
ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_hmm2, (1:lres_hmm2-sqrt(lres_hmm2)*1.36)-1:lres_hmm2, type="l",
lty=2)
lines(res_hmm2, (1:lres_hmm2+sqrt(lres_hmm2)*1.36)-1:lres_hmm2, type="l",
lty=2)
# 99% CI
lines(res_hmm2, (1:lres_hmm2-sqrt(lres_hmm2)*1.63)-1:lres_hmm2, type="l",
lty=3)
lines(res_hmm2, (1:lres_hmm2+sqrt(lres_hmm2)*1.63)-1:lres_hmm2, type="l",
lty=3)
lines(res_hmm2, rep(0, lres_hmm2), col="red")
par(oldpar)
# =====
# An example of a 3-state HMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 30 seconds on a laptop (the
# 2-state model above takes about 4 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(1/3, 1/3, 1/3)
tpm <- matrix(c(0.2, 0.65, 0.15, 0.50, 0.30, 0.20, 0.25, 0.50, 0.25),
nrow = 3, byrow = TRUE)
mc.par <- list(initd=NULL, tpm=tpm, stationary=TRUE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(1, 1) # K
rate[2, ] <- c(1.5, 2.5) # M
rate[3, ] <- c(0.02, 0.015) # R
rate[4, ] <- c(3, 3) # P: inflection times P/R = 150 and 200
shape <- 1
scale <- 1.7
scpg.par <- list(rate=rate, shape=shape, scale=scale,
rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
WBsum.method='direct.evaluation')

# Defining an object for 3-state HMM with TSCPG distribution
obj_hmm2_m3 <- hmm.scpg(x = NULL, mc.par, scpg.par)
# Simulating a 3-state HMM with TSCPG distribution

sim_hmm2_m3 <- sim.hmm.scpg(obj_hmm2_m3, nsim=100, seed=9)
# Fitting a 3-state HMM with TSCPG distributions
# The fit takes about 30 seconds, so it is wrapped in donttest.

fit_hmm2_m3 <- fit.hmm.scpg(sim_hmm2_m3)
print(fit_hmm2_m3)

# Computing smoothed probabilities of fitted 3-state HMM with TSCPG

```

```

# distribution

fit_hmm2_m3$x=sim_hmm2_m3$x
sp_hmm2_m3 <- sp.hsmm.scpg(fit_hmm2_m3)
print(sp_hmm2_m3)

# Plotting of residuals to check whether the fitted HMM with TSCPG renewal
# process is a good fit.

dev.new()
res_hmm2_m3 <- sp_hmm2_m3$residuals
lres_hmm2_m3 <- length(sp_hmm2_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_hmm2_m3, 1:lres_hmm2_m3-res_hmm2_m3, type="l",
     xlab="Transformed times",
     ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_hmm2_m3, (1:lres_hmm2_m3-sqrt(lres_hmm2_m3)*1.36)-1:lres_hmm2_m3,
     type="l", lty=2)
lines(res_hmm2_m3, (1:lres_hmm2_m3+sqrt(lres_hmm2_m3)*1.36)-1:lres_hmm2_m3,
     type="l", lty=2)
# 99% CI
lines(res_hmm2_m3, (1:lres_hmm2_m3-sqrt(lres_hmm2_m3)*1.63)-1:lres_hmm2_m3,
     type="l", lty=3)
lines(res_hmm2_m3, (1:lres_hmm2_m3+sqrt(lres_hmm2_m3)*1.63)-1:lres_hmm2_m3,
     type="l", lty=3)
lines(res_hmm2_m3, rep(0, lres_hmm2_m3), col="red")
par(oldpar)

```

sp.hsmm.scpg	<i>Smoothed probabilities for an m-state HSMM with gamma or SCPG or TSCPG renewal processes</i>
--------------	---

Description

Calculates the probability of being in State j at time t for an HSMM with class `hsmm.gamma`, `hsmm.scpg` and `hsmm.tscpg` as typically returned by [hsmm.scpg](#).

Usage

```
sp.hsmm.scpg(hsmmobj)
```

Arguments

`hsmmobj` is a model object with class `hsmm.gamma`, `hsmm.scpg` and `hsmm.tscpg` as typically returned by [hsmm.scpg](#).

Details

For a hidden semi-Markov model the smoothed probability $\gamma_t(j) = \Pr(S_t = j \mid X_{1:T}; \theta)$ cannot be formed from the forward and backward variables of an HMM, the hidden path being a sequence of visits rather than of states. It is obtained by the forward-backward algorithm, in which the forward variable $\alpha_t(j)$ of [fit.hsmm.scpg](#) records that a visit to State j ends at time t and the backward variable $\beta_t(j) = \Pr(X_{t+1}, \dots, X_T \mid S_t = j, S_{t+1} \neq j; \theta)$ the complementary conditional probability. Writing $\alpha_{t+1}^*(j)$ and $\beta_{t+1}^*(j)$ for the corresponding quantities at the commencement of a visit, the smoothed probability satisfies the backward recursion

$$\gamma_t(j) = \gamma_{t+1}(j) + \frac{\alpha_t(j)\beta_t(j)}{L(\theta)} - \frac{\alpha_{t+1}^*(j)\beta_{t+1}^*(j)}{L(\theta)},$$

with $\gamma_T(j) = \alpha_T(j)/L(\theta)$, the two subtracted terms being the conditional probabilities that a visit to State j ends at t and that one commences at $t + 1$. The shifted Poisson sojourn mass function determines the number of visits completed within the record, while the corresponding survival function addresses visits that are right-censored at T . The recursions are scaled at each time step.

The globally optimal sequence of visits, with their durations, is obtained from [vip.hsmm.scpg](#).

The residual process returned alongside the smoothed probabilities provides the goodness-of-fit assessment of the fitted model as a point process (Berman, 1983; Ogata, 1988; Wang et al., 2012, Shahzadi et al., 2023,2026), see [sp.hmm.scpg](#)

Value

returns an $T \times m$ matrix for the smoothed probabilities and a numeric vector of residuals from an HSMM with class [hsmm.scpg](#).

References

- Berman M (1983). “Comment on “Likelihood analysis of point processes and its applications to seismological data” by Ogata.” *Bulletin of the International Statistical Institute*, **50**, 412-418.
- Ogata Y (1988). “Statistical models for earthquake occurrences and residual analysis for point processes.” *Journal of the American Statistical Association*, **83**(401), 9-27.
- Wang T, Bebbington MS, Harte D (2012). “Markov-modulated Hawkes process with stepwise decay.” *Annals of the Institute of Statistical Mathematics*, **64**, 521-544. [doi:10.1007/s10463010-03207](#).
- Shahzadi A, Wang T, Parry M, Bebbington M (2023). “Inhomogeneous hidden semi-Markov models for incompletely observed point processes.” *Annals of Institute Statistical Mathematics*, **75**, 253-280. [doi:10.1007/s10463022008435](#).
- Shahzadi A, Wang T, Parry M, Bebbington M (2026). “Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models.” *Advances in Data Analysis and Classification*, **20**, 663-691. [doi:10.1007/s1163402500632x](#).

Examples

```
# =====
# An example of a 2-state HSMM with an SCPG renewal process
# =====
```

```

# State 1 is the state of completeness (gamma interevent times with shape
# alpha); in the incomplete State 2 the number N of events missing between
# two observed events is shifted Poisson with parameter lambda_2, so the
# observed events are N + 1 such gamma interevent times apart. The sojourns
# are shifted Poisson with
# parameters dur (expected sojourns 1 + dur = 5 and 4 observations). With
# m = 2 the zero-diagonal transition matrix is fixed: the chain alternates
# between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
               rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
               WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with SCPG distribution

obj_hsmm1 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm1)
# Simulating a 2-state HSMM with SCPG distribution

sim_hsmm1 <- sim.hsmm.scpg(obj_hsmm1, nsim = 150, seed = 1)
str(sim_hsmm1[c("x", "states")])

# Number of visits to each state in the simulated path
table(rle(sim_hsmm1$states)$values)
# Fitting a 2-state HSMM with SCPG distributions
# (the parameters of obj_hsmm1 are the starting values)

fit_hsmm1 <- fit.hsmm.scpg(sim_hsmm1)
fit_hsmm1$smc.par$dur      # sojourn parameters mu_1, mu_2 (true 4, 3)
fit_hsmm1$scpg.par$rate    # lambda_2 (true 3)
fit_hsmm1$scpg.par$shape   # alpha (true 1.5)
fit_hsmm1$scpg.par$scale   # beta (true 1)
fit_hsmm1$logLik
# Smoothed probabilities for a 2-state HSMM with SCPG distribution

fit_hsmm1$x <- sim_hsmm1$x

# Computing smoothed probabilities and residuals

sp_hsmm1 <- sp.hsmm.scpg(fit_hsmm1)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + Pr(S_t = 2 | x) / (2 + lambda_2), averaged over the
# record and compared with the true proportion (1 in State 1 and

```

```

# 1/(2 + lambda_2) in State 2)
p_hsmm1 <- sp_hsmm1$smoothed.prob[, 1] +
  sp_hsmm1$smoothed.prob[, 2] / (2 + fit_hsmm1$scpg.par$rate[1, 1])
c(estimated = mean(p_hsmm1),
  true = mean(ifelse(sim_hsmm1$states == 1, 1,
    1 / (2 + obj_hsmm1$scpg.par$rate[1, 1]))))

# Plotting of residuals to check whether the fitted HSMM with SCPG renewal
# process is a good fit.

dev.new()
res_hsmm1 <- sp_hsmm1$residuals
lres_hsmm1 <- length(sp_hsmm1$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_hsmm1, 1:lres_hsmm1-res_hsmm1, type="l", xlab="Transformed times",
  ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_hsmm1, (1:lres_hsmm1-sqrt(lres_hsmm1)*1.36)-1:lres_hsmm1, type="l",
  lty=2)
lines(res_hsmm1, (1:lres_hsmm1+sqrt(lres_hsmm1)*1.36)-1:lres_hsmm1, type="l",
  lty=2)
# 99% CI
lines(res_hsmm1, (1:lres_hsmm1-sqrt(lres_hsmm1)*1.63)-1:lres_hsmm1, type="l",
  lty=3)
lines(res_hsmm1, (1:lres_hsmm1+sqrt(lres_hsmm1)*1.63)-1:lres_hsmm1, type="l",
  lty=3)
lines(res_hsmm1, rep(0, lres_hsmm1), col="red")
par(oldpar)
# =====
# An example of a 3-state HSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 15 seconds on a laptop (the
# 2-state model above takes about 1 second), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
  0.5, 0, 0.5,
  0.3, 0.7, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd=NULL, tpm=tpm, dur=c(3, 5, 2),
  dur.lb=NA, dur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
  rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
  WBSum.method='direct.evaluation')

# Defining an object of 3-state HSMM with SCPG distribution

obj_hsmm1_m3 <- hsmm.scp(x = NULL, smc.par, scpg.par)
# Simulating a 3-state HSMM with SCPG distribution

```

```

sim_hsmm1_m3 <- sim.hsmm.scpg(obj_hsmm1_m3, nsim = 80, seed = 1)
print(sim_hsmm1_m3)
# Fitting a 3-state HSMM with SCPG distributions
# The fit takes about 15 seconds, so it is wrapped in donttest.

fit_hsmm1_m3 <- fit.hsmm.scpg(sim_hsmm1_m3)
print(fit_hsmm1_m3)

# Smoothed probabilities for a 3-state HSMM with SCPG distribution

fit_hsmm1_m3$x <- sim_hsmm1_m3$x

# Computing smoothed probabilities and residuals

sp_hsmm1_m3 <- sp.hsmm.scpg(fit_hsmm1_m3)

# Plotting of residuals to check whether the fitted HSMM with SCPG renewal
# process is a good fit.

dev.new()
res_hsmm1_m3 <- sp_hsmm1_m3$residuals
lres_hsmm1_m3 <- length(sp_hsmm1_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_hsmm1_m3, 1:lres_hsmm1_m3-res_hsmm1_m3, type="l",
     xlab="Transformed times",
     ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_hsmm1_m3,
      (1:lres_hsmm1_m3-sqrt(lres_hsmm1_m3)*1.36)-1:lres_hsmm1_m3, type="l", lty=2)
lines(res_hsmm1_m3,
      (1:lres_hsmm1_m3+sqrt(lres_hsmm1_m3)*1.36)-1:lres_hsmm1_m3, type="l", lty=2)
# 99% CI
lines(res_hsmm1_m3,
      (1:lres_hsmm1_m3-sqrt(lres_hsmm1_m3)*1.63)-1:lres_hsmm1_m3, type="l", lty=3)
lines(res_hsmm1_m3,
      (1:lres_hsmm1_m3+sqrt(lres_hsmm1_m3)*1.63)-1:lres_hsmm1_m3, type="l", lty=3)
lines(res_hsmm1_m3, rep(0, lres_hsmm1_m3), col="red")
par(oldpar)

# =====
# An example of a 2-state HSMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter  $\lambda_2(t)$ , where
#  $\lambda_2(t) = K + M / (1 + \exp(R * h_{t-1} - P))$ 
# falls from about 5.8 at the start of the record towards  $K = 0.5$ , around
# the inflection time  $P/R = 400$  of the last occurrence time  $h_{t-1}$ . The
# sojourns are shifted Poisson with parameters dur (expected sojourns
#  $1 + \text{dur} = 5$  and 4 observations). With  $m = 2$  the zero-diagonal transition
# matrix is fixed: the chain alternates between the two states.

```

```

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
               dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5      # K
rate[2, ] <- 6        # M
rate[3, ] <- 0.005    # R
rate[4, ] <- 2        # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
               rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
               WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with TSCPG distribution

obj_hsmm2 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm2)
# Simulating a 2-state HSMM with a TSCPG distribution

sim_hsmm2 <- sim.hsmm.scpg(obj_hsmm2, nsim = 300, seed = 3)
str(sim_hsmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2$x[-300]), h_T = sum(sim_hsmm2$x))
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state HSMM with a TSCPG distribution
# (the parameters of obj_hsmm2 are the starting values)

fit_hsmm2 <- fit.hsmm.scpg(sim_hsmm2)
fit_hsmm2$smc.par$dur      # sojourn parameters mu_1, mu_2 (true 4, 3)
fit_hsmm2$scpg.par$rate    # K, M, R, P of lambda_2(t)
fit_hsmm2$scpg.par$scale   # true 1
fit_hsmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# Smoothed state probabilities and residuals for the fitted 2-state HSMM
# with TSCPG distribution

fit_hsmm2$x <- sim_hsmm2$x
sp_hsmm2 <- sp.hsmm.scpg(fit_hsmm2)

# Poisson mean lambda_2(t), fitted and true, at the last occurrence times
# h_{t-1}, and the four quarters of the record
lambda_t <- function(r, h) r[1] + r[2] / (1 + exp(r[3] * h - r[4]))
h_hsmm2 <- c(0, cumsum(sim_hsmm2$x)[-300])
lam_hsmm2 <- lambda_t(fit_hsmm2$scpg.par$rate[, 1], h_hsmm2)
lam0_hsmm2 <- lambda_t(obj_hsmm2$scpg.par$rate[, 1], h_hsmm2)

```

```

gam_hsmm2 <- sp_hsmm2$smoothed.prob
q_hsmm2 <- cut(seq_along(h_hsmm2), 4)

# Proportion of events recorded around each observation,
#  $p_t = \Pr(S_t = 1 \mid x) + \Pr(S_t = 2 \mid x) / (2 + \lambda_2(t))$ ,
# averaged over each quarter (row p), with its two parts: the share of
# State 1 (row state.1) and the proportion recorded while the chain is in
# State 2 (row state.2). The true proportion (row p.true) is 1 in State 1
# and  $1/(2 + \lambda_2(t))$  in State 2. In an HSMM the sojourns do not depend
# on time, so the share of State 1 varies only by chance; the proportion
# recorded in State 2 rises as  $\lambda_2(t)$  falls.
p_hsmm2 <- gam_hsmm2[, 1] + gam_hsmm2[, 2] / (2 + lam_hsmm2)
p0_hsmm2 <- ifelse(sim_hsmm2$states == 1, 1, 1 / (2 + lam0_hsmm2))
rbind(p = tapply(p_hsmm2, q_hsmm2, mean),
      state.1 = tapply(gam_hsmm2[, 1], q_hsmm2, mean),
      state.2 = tapply(gam_hsmm2[, 2] / (2 + lam_hsmm2), q_hsmm2, sum) /
        tapply(gam_hsmm2[, 2], q_hsmm2, sum),
      p.true = tapply(p0_hsmm2, q_hsmm2, mean))

# Plotting the residuals (cumulative number of events minus rescaled time)
# with Kolmogorov-Smirnov bands, to check whether the fitted model is a good
# fit

dev.new()
res_hsmm2 <- sp_hsmm2$residuals
lres_hsmm2 <- length(sp_hsmm2$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_hsmm2, 1:lres_hsmm2-res_hsmm2, type="l", xlab="Transformed times",
      ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_hsmm2, (1:lres_hsmm2-sqrt(lres_hsmm2)*1.36)-1:lres_hsmm2, type="l",
      lty=2)
lines(res_hsmm2, (1:lres_hsmm2+sqrt(lres_hsmm2)*1.36)-1:lres_hsmm2, type="l",
      lty=2)
# 99% CI
lines(res_hsmm2, (1:lres_hsmm2-sqrt(lres_hsmm2)*1.63)-1:lres_hsmm2, type="l",
      lty=3)
lines(res_hsmm2, (1:lres_hsmm2+sqrt(lres_hsmm2)*1.63)-1:lres_hsmm2, type="l",
      lty=3)
lines(res_hsmm2, rep(0, lres_hsmm2), col="red")
par(oldpar)

# =====
# An example of a 3-state HSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 7 seconds), so the examples that fit
# it are wrapped in donttest.

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete States 2 and 3 the number of missing events N between two
# observed events is shifted Poisson with parameter  $\lambda_j(t)$ , where

```

```

# lambda_j(t) = K_j + M_j / (1 + exp(R_j * h_{t-1} - P_j))
# falls around the inflection time P_j/R_j of the last occurrence time
# h_{t-1}: lambda_2(t) from about 3.4 towards 0.5 around 600, and
# lambda_3(t) from about 10 towards 4 around 900. The sojourns are shifted
# Poisson with parameters dur (expected sojourns 1 + dur = 5, 4 and 4
# observations), and the zero-diagonal transition matrix gives the next
# state of each visit. The shape 2 makes the interevent times of the three
# states less overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3.6, 5.4)    # P: inflection times P/R = 600 and 900
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBsum.method = 'direct.evaluation')

# Defining an object of 3-state HSMM with TSCPG distribution

obj_hsmm2_m3 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm2_m3)
# Simulating a 3-state HSMM with a TSCPG distribution

sim_hsmm2_m3 <- sim.hsmm.scpg(obj_hsmm2_m3, nsim = 400, seed = 3)
str(sim_hsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rates are evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2_m3$x[-400]), h_T = sum(sim_hsmm2_m3$x))

# Number of visits to each state in the simulated path
table(rle(sim_hsmm2_m3$states)$values)
# Fitting a 3-state HSMM with a TSCPG distribution
# (the parameters of obj_hsmm2_m3 are the starting values). The fit
# takes about 2 minutes, so it is wrapped in donttest.

fit_hsmm2_m3 <- fit.hsmm.scpg(sim_hsmm2_m3)
fit_hsmm2_m3$smc.par$tpm      # transition matrix of the embedded chain
# sojourn parameters mu_1, mu_2, mu_3 (true 4, 3, 3)
fit_hsmm2_m3$smc.par$dur
fit_hsmm2_m3$scpg.par$rate    # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_hsmm2_m3$scpg.par$scale   # true 0.5
fit_hsmm2_m3$logLik

# Smoothed state probabilities and residuals for the fitted 3-state HSMM

```

```

# with TSCPG distribution

fit_hsmm2_m3$x <- sim_hsmm2_m3$x
sp_hsmm2_m3 <- sp.hsmm.scp(fit_hsmm2_m3)

# Poisson means lambda_2(t) and lambda_3(t), fitted and true, at the last
# occurrence times h_{t-1}, and the four quarters of the record
lambda_t <- function(r, h) r[1] + r[2] / (1 + exp(r[3] * h - r[4]))
h_hsmm2_m3 <- c(0, cumsum(sim_hsmm2_m3$x)[-400])
lam_hsmm2_m3 <- sapply(1:2,
  function(j) lambda_t(fit_hsmm2_m3$scpg.par$rate[, j], h_hsmm2_m3))
lam0_hsmm2_m3 <- sapply(1:2,
  function(j) lambda_t(obj_hsmm2_m3$scpg.par$rate[, j], h_hsmm2_m3))
gam_hsmm2_m3 <- sp_hsmm2_m3$smoothed.prob
q_hsmm2_m3 <- cut(seq_along(h_hsmm2_m3), 4)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + sum_j Pr(S_t = j | x) / (2 + lambda_j(t)),
# averaged over each quarter (row p), with its two parts: the share of
# State 1 (row state.1) and the proportion recorded while the chain is in
# States 2 and 3 (row states.2.3). The true proportion (row p.true) is 1
# in State 1 and 1/(2 + lambda_j(t)) in State j. In an HSMM the sojourns
# do not depend on time, so the share of State 1 varies only by chance;
# the proportion recorded in States 2 and 3 rises from quarter to quarter
# as lambda_2(t) and lambda_3(t) fall.
p_hsmm2_m3 <- gam_hsmm2_m3[, 1] +
  rowSums(gam_hsmm2_m3[, 2:3] / (2 + lam_hsmm2_m3))
s_hsmm2_m3 <- sim_hsmm2_m3$states
p0_hsmm2_m3 <- ifelse(s_hsmm2_m3 == 1, 1,
  1 / (2 + lam0_hsmm2_m3[cbind(seq_along(s_hsmm2_m3),
    pmax(s_hsmm2_m3 - 1, 1))]))
rbind(p = tapply(p_hsmm2_m3, q_hsmm2_m3, mean),
  state.1 = tapply(gam_hsmm2_m3[, 1], q_hsmm2_m3, mean),
  states.2.3 = tapply(rowSums(gam_hsmm2_m3[, 2:3] / (2 + lam_hsmm2_m3)),
    q_hsmm2_m3, sum) /
    tapply(rowSums(gam_hsmm2_m3[, 2:3]), q_hsmm2_m3, sum),
  p.true = tapply(p0_hsmm2_m3, q_hsmm2_m3, mean))

# Plotting the residuals (cumulative number of events minus rescaled time)
# with Kolmogorov-Smirnov bands, to check whether the fitted model is a good
# fit

dev.new()
res_hsmm2_m3 <- sp_hsmm2_m3$residuals
lres_hsmm2_m3 <- length(sp_hsmm2_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_hsmm2_m3, 1:lres_hsmm2_m3-res_hsmm2_m3, type="l",
  xlab="Transformed times",
  ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_hsmm2_m3,
  (1:lres_hsmm2_m3-sqrt(lres_hsmm2_m3)*1.36)-1:lres_hsmm2_m3, type="l", lty=2)
lines(res_hsmm2_m3,

```

```

(1:lres_hsmm2_m3+sqrt(lres_hsmm2_m3)*1.36)-1:lres_hsmm2_m3, type="l", lty=2)
# 99% CI
lines(res_hsmm2_m3,
      (1:lres_hsmm2_m3-sqrt(lres_hsmm2_m3)*1.63)-1:lres_hsmm2_m3, type="l", lty=3)
lines(res_hsmm2_m3,
      (1:lres_hsmm2_m3+sqrt(lres_hsmm2_m3)*1.63)-1:lres_hsmm2_m3, type="l", lty=3)
lines(res_hsmm2_m3, rep(0, lres_hsmm2_m3), col="red")
par(oldpar)

```

sp.ihmm.scpg

Smoothed probabilities for an m-state IHMM with gamma or SCPG or TSCPG renewal processes

Description

Calculates the probability of being in State j at time t for an IHMM with class `ihmm.gamma`, `ihmm.scpg` and `ihmm.tscpg` as typically returned by [ihmm.scpg](#).

Usage

```
sp.ihmm.scpg(ihmmobj)
```

Arguments

`ihmmobj` is a model object with class `ihmm.gamma`, `ihmm.scpg` and `ihmm.tscpg` as typically returned by [ihmm.scpg](#).

Details

The smoothed probability $\gamma_t(j) = \Pr(S_t = j \mid X_{1:T}, H_{1:T}; \theta)$ is obtained by the forward-backward algorithm with the transition probabilities replaced by their time-varying counterparts. With the forward variable $\alpha_t(j)$ defined as in [fit.ihmm.scpg](#) and the backward variable satisfying

$$\beta_T(i) = 1, \quad \beta_t(i) = \sum_{j=1}^m a_{ij}(t+1) f_j(x_{t+1}) \beta_{t+1}(j), \quad t = T-1, \dots, 1,$$

the smoothed probability is $\gamma_t(j) = \alpha_t(j) \beta_t(j) / L(\theta)$. Each step employs the transition probability matrix reconstructed from the logistic coefficients at the occurrence time accumulated to that point. Both recursions are evaluated on the logarithmic scale with scaling at each time step.

Plotted against the occurrence times, the smoothed probability of State 1 provides the most direct evidence of the time inhomogeneity: under a correctly specified model it increases along the record, and the interval over which it rises estimates the period during which the record becomes complete. As in the homogeneous case, the sequence of pointwise maximizers need not be admissible under the model; the globally optimal sequence is obtained from [vip.ihmm.scpg](#).

The residual process returned alongside the smoothed probabilities provides the goodness-of-fit assessment of the fitted model as a point process (Berman, 1983; Ogata, 1988; Wang et al., 2012), see [sp.hmm.scpg](#).

Value

returns an $T \times m$ matrix for the smoothed probabilities and a numeric vector of residuals from an IHMM with class `ihmm.scpg`.

References

- Berman M (1983). “Comment on “Likelihood analysis of point processes and its applications to seismological data” by Ogata.” *Bulletin of the International Statistical Institute*, **50**, 412-418.
- Ogata Y (1988). “Statistical models for earthquake occurrences and residual analysis for point processes.” *Journal of the American Statistical Association*, **83**(401), 9-27.
- Wang T, Bebbington MS, Harte D (2012). “Markov-modulated Hawkes process with stepwise decay.” *Annals of the Institute of Statistical Mathematics*, **64**, 521-544. doi:10.1007/s10463010-03207.
- Daley DJ, Vere-Jones D (2003). *An Introduction to the Theory of Point Processes*. Springer, New York. doi:10.1007/b97277.

Examples

```
# =====
# An example of a 2-state IHMM with an SCPG renewal process
# =====

# The transition probabilities a_12(t) and a_22(t) into the incomplete
# State 2 decrease along the record: the log-odds of moving from state i to
# State 2 rather than to State 1 is w_i2 - r_i2 * h_{t-1}, which is zero at
# the crossing time w_i2 / r_i2 (250 for a_12(t) and 500 for a_22(t)). In
# State 2 the number N of events missing between two observed events is
# shifted Poisson with parameter lambda_2. The chain starts mostly in State 2
# (initd = (0.2, 0.8)),
# since the early part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.004, 0.006) # rates of change r_12, r_22
ttpm[, , 2] <- c(1, 3)        # intercepts w_12, w_22
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

# Defining an object of 2-state IHMM with SCPG distribution

obj_ihmm1 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm1)
# Simulating a 2-state IHMM with SCPG distribution
```

```

sim_ihmm1 <- sim.ihmm.scpg(obj_ihmm1, nsim = 200, seed = 1)
str(sim_ihmm1[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm1$states,
      half = ifelse(seq_along(sim_ihmm1$states) <= 100, "first", "second"))
# The fit takes about 6 s, so the code below is wrapped in donttest.

# Fitting a 2-state IHMM with SCPG distributions
# (the parameters of obj_ihmm1 are the starting values)

fit_ihmm1 <- fit.ihmm.scpg(sim_ihmm1)
fit_ihmm1$imc.par$ttpm[, 1, ] # r_i2 (first column) and w_i2 (second)
fit_ihmm1$scpg.par$rate      # lambda_2 (true 3)
fit_ihmm1$scpg.par$shape     # true 1.5
fit_ihmm1$scpg.par$scale     # true 1
fit_ihmm1$logLik

# The fit takes about 6 s, so the code below is wrapped in donttest.

# Smoothed probabilities for a 2-state IHMM with an SCPG distribution

fit_ihmm1$x <- sim_ihmm1$x

# Computing smoothed probabilities and residuals

sp_ihmm1 <- sp.ihmm.scpg(fit_ihmm1)

# Proportion of events recorded around each observation,
#  $p_t = \Pr(S_t = 1 \mid x) + \Pr(S_t = 2 \mid x) / (2 + \lambda_2)$ , averaged over the
# four quarters of the record (row p) and compared with the true proportion
# (row p.true: 1 in State 1 and  $1/(2 + \lambda_2)$  in State 2). With a
# constant  $\lambda_2$  the rise comes from the transitions alone, which make
# State 1 more frequent (row state.1).
p_ihmm1 <- sp_ihmm1$smoothed.prob[, 1] +
  sp_ihmm1$smoothed.prob[, 2] / (2 + fit_ihmm1$scpg.par$rate[1, 1])
p0_ihmm1 <- ifelse(sim_ihmm1$states == 1, 1,
  1 / (2 + obj_ihmm1$scpg.par$rate[1, 1]))
q_ihmm1 <- cut(seq_along(p_ihmm1), 4)
rbind(p = tapply(p_ihmm1, q_ihmm1, mean),
      state.1 = tapply(sp_ihmm1$smoothed.prob[, 1], q_ihmm1, mean),
      p.true = tapply(p0_ihmm1, q_ihmm1, mean))

# Plotting of residuals to check whether the fitted IHMM with SCPG renewal
# process is a good fit.

dev.new()
res_ihmm1 <- sp_ihmm1$residuals
lres_ihmm1 <- length(sp_ihmm1$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihmm1, 1:lres_ihmm1-res_ihmm1, type="l", xlab="Transformed times",
     ylab="Transformed cumsum", ylim=c(-50, 50))

```

```

# 95% CI
lines(res_ihmm1, (1:lres_ihmm1-sqrt(lres_ihmm1)*1.36)-1:lres_ihmm1, type="l",
      lty=2)
lines(res_ihmm1, (1:lres_ihmm1+sqrt(lres_ihmm1)*1.36)-1:lres_ihmm1, type="l",
      lty=2)
# 99% CI
lines(res_ihmm1, (1:lres_ihmm1-sqrt(lres_ihmm1)*1.63)-1:lres_ihmm1, type="l",
      lty=3)
lines(res_ihmm1, (1:lres_ihmm1+sqrt(lres_ihmm1)*1.63)-1:lres_ihmm1, type="l",
      lty=3)
lines(res_ihmm1, rep(0, lres_ihmm1), col="red")
par(oldpar)

# =====
# An example of a 3-state IHMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 6 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(0.45, 0.25, 0.30)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.0008, 0.0010, 0.0006,
                        0.0006, 0.0008, 0.0010), nrow = m, ncol = m-1)
# intercepts w_ij; a_ij(t) = a_i1(t) at h = w_ij/r_ij, between 2000 and 3000
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(2500, 2000, 3000,
                                       2000, 3000, 2500), nrow = m, ncol = m-1)

stationary <- FALSE
imc.par <- list(initd=initd, ttpm=ttpm, stationary=stationary)
rate <- matrix(c(7, 2), nrow = 1, ncol = m-1)
shape <- 1.7
scale <- 3.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

# Defining an object of 3-state IHMM with SCPG distribution

obj_ihmm1_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
# Simulating a 3-state IHMM with SCPG distribution

sim_ihmm1_m3 <- sim.ihmm.scpg(obj_ihmm1_m3, nsim = 200, seed = 2)
print(sim_ihmm1_m3)
# Fitting a 3-state IHMM with SCPG distributions
# The fit takes about 1.5 minutes, so it is wrapped in donttest.

fit_ihmm1_m3 <- fit.ihmm.scpg(sim_ihmm1_m3)
print(fit_ihmm1_m3)

# Smoothed probabilities for a 3-state IHMM with an SCPG distribution

```

```

fit_ihmm1_m3$x=sim_ihmm1_m3$x

# Computing smoothed probabilities and residuals

sp_ihmm1_m3 <- sp.ihmm.scp(fit_ihmm1_m3)

# Plotting of residuals to check whether the fitted IHMM with SCPG renewal
# process is a good fit.

dev.new()
res_ihmm1_m3 <- sp_ihmm1_m3$residuals
lres_ihmm1_m3 <- length(sp_ihmm1_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihmm1_m3, 1:lres_ihmm1_m3-res_ihmm1_m3, type="l",
      xlab="Transformed times",
      ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihmm1_m3,
      (1:lres_ihmm1_m3-sqrt(lres_ihmm1_m3)*1.36)-1:lres_ihmm1_m3, type="l", lty=2)
lines(res_ihmm1_m3,
      (1:lres_ihmm1_m3+sqrt(lres_ihmm1_m3)*1.36)-1:lres_ihmm1_m3, type="l", lty=2)
# 99% CI
lines(res_ihmm1_m3,
      (1:lres_ihmm1_m3-sqrt(lres_ihmm1_m3)*1.63)-1:lres_ihmm1_m3, type="l", lty=3)
lines(res_ihmm1_m3,
      (1:lres_ihmm1_m3+sqrt(lres_ihmm1_m3)*1.63)-1:lres_ihmm1_m3, type="l", lty=3)
lines(res_ihmm1_m3, rep(0, lres_ihmm1_m3), col="red")
par(oldpar)

# =====
# An example of a 2-state IHMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scp):
# - the transition probabilities  $a_{12}(t)$  and  $a_{22}(t)$  into the incomplete
#   State 2 decrease: the log-odds of moving from state  $i$  to State 2
#   rather than to State 1 is  $w_{i2} - r_{i2} * h_{\{t-1\}}$ , which is zero at the
#   crossing time  $w_{i2} / r_{i2}$  (about 304 for  $a_{12}(t)$  and 600 for  $a_{22}(t)$ );
# - the Poisson mean  $\lambda_{22}(t)$  of the TSCPG falls from about 5.4 at the
#   start of the record towards  $K = 0.5$ , around the inflection time
#    $P/R = 400$ .
# The chain starts mostly in State 2 ( $initd = (0.2, 0.8)$ ), since the early
# part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.0028, 0.005) # rates of change  $r_{12}$ ,  $r_{22}$ 
ttpm[, , 2] <- c(0.85, 3) # intercepts  $w_{12}$ ,  $w_{22}$ 
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(c(0.5, 5, 0.01, 4), nrow = 4) # K, M, R, P:  $P/R = 400$ 

```

```

shape <- 1
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHMM with TSCPG distribution

obj_ihmm2 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm2)
# Simulating a 2-state IHMM with a TSCPG distribution

sim_ihmm2 <- sim.ihmm.scpg(obj_ihmm2, nsim = 200, seed = 2)
str(sim_ihmm2[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm2$states,
      half = ifelse(seq_along(sim_ihmm2$states) <= 100, "first", "second"))
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state IHMM with a TSCPG distribution
# (the parameters of obj_ihmm2 are the starting values)

fit_ihmm2 <- fit.ihmm.scpg(sim_ihmm2)
fit_ihmm2$imc.par$ttpm[, 1, ] # r_i2 (first column) and w_i2 (second)
fit_ihmm2$scpg.par$rate[, 1] # K, M, R, P of lambda_2(t)
fit_ihmm2$scpg.par$scale      # true 1
fit_ihmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# Smoothed state probabilities and residuals for the fitted 2-state IHMM
# with TSCPG distribution

fit_ihmm2$x <- sim_ihmm2$x
sp_ihmm2 <- sp.ihmm.scpg(fit_ihmm2)

# Poisson mean lambda_2(t), fitted and true, at the last occurrence times
# h_{t-1}, and the four quarters of the record
lambda_t <- function(r, h) r[1] + r[2] / (1 + exp(r[3] * h - r[4]))
h_ihmm2 <- c(0, cumsum(sim_ihmm2$x)[-200])
lam_ihmm2 <- lambda_t(fit_ihmm2$scpg.par$rate[, 1], h_ihmm2)
lam0_ihmm2 <- lambda_t(obj_ihmm2$scpg.par$rate[, 1], h_ihmm2)
gam_ihmm2 <- sp_ihmm2$smoothed.prob
q_ihmm2 <- cut(seq_along(h_ihmm2), 4)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + Pr(S_t = 2 | x) / (2 + lambda_2(t)),
# averaged over each quarter (row p), with its two parts: the share of
# State 1 (row state.1) and the proportion recorded while the chain is in
# State 2 (row state.2). The true proportion (row p.true) is 1 in State 1
# and 1/(2 + lambda_2(t)) in State 2. Both kinds of time dependence raise

```

```

# p_t: the share of State 1 rises as the transitions into State 2 become
# less likely, and the proportion recorded in State 2 rises as lambda_2(t)
# falls. p_t rises as the true proportion does.
p_ihmm2 <- gam_ihmm2[, 1] + gam_ihmm2[, 2] / (2 + lam_ihmm2)
p0_ihmm2 <- ifelse(sim_ihmm2$states == 1, 1, 1 / (2 + lam0_ihmm2))
rbind(p = tapply(p_ihmm2, q_ihmm2, mean),
      state.1 = tapply(gam_ihmm2[, 1], q_ihmm2, mean),
      state.2 = tapply(gam_ihmm2[, 2] / (2 + lam_ihmm2), q_ihmm2, sum) /
        tapply(gam_ihmm2[, 2], q_ihmm2, sum),
      p.true = tapply(p0_ihmm2, q_ihmm2, mean))

# Plotting of residuals to check whether the fitted IHMM with TSCPG renewal
# process is a good fit
dev.new()
res_ihmm2 <- sp_ihmm2$residuals
lres_ihmm2 <- length(sp_ihmm2$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihmm2, 1:lres_ihmm2-res_ihmm2, type="l", xlab="Transformed times",
     ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihmm2, (1:lres_ihmm2-sqrt(lres_ihmm2)*1.36)-1:lres_ihmm2, type="l",
      lty=2)
lines(res_ihmm2, (1:lres_ihmm2+sqrt(lres_ihmm2)*1.36)-1:lres_ihmm2, type="l",
      lty=2)
# 99% CI
lines(res_ihmm2, (1:lres_ihmm2-sqrt(lres_ihmm2)*1.63)-1:lres_ihmm2, type="l",
      lty=3)
lines(res_ihmm2, (1:lres_ihmm2+sqrt(lres_ihmm2)*1.63)-1:lres_ihmm2, type="l",
      lty=3)
lines(res_ihmm2, rep(0, lres_ihmm2), col="red")
par(oldpar)

# =====
# An example of a 3-state IHMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2.5 minutes on a laptop (the
# 2-state model above takes about 8 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scp):
# - the transition probabilities a_ij(t): moves into the incomplete
#   States 2 and 3 become less likely as time goes on. The log-odds of
#   moving from state i to state j rather than to State 1 is
#   w_ij - r_ij * h_{t-1}, which is zero at the crossing time w_ij / r_ij;
# - the Poisson means lambda_j(t) of the TSCPG: in the incomplete states
#   fewer events are missed after the inflection times P_j / R_j.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1, so that the states are easier to tell apart.

m <- 3
initd <- c(0.4, 0.3, 0.3)

```

```

ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.006, 0.006, 0.006,
                        0.006, 0.006, 0.006), nrow = m, ncol = m-1)
# intercepts w_ij; crossing times w_ij/r_ij between 400 and 600
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(500, 600, 500,
                                       400, 500, 600), nrow = m, ncol = m-1)
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(0.5, 4)      # K: lambda_2(t) falls from about 3.4 towards
rate[2, ] <- c(3, 6)       # M: 0.5, lambda_3(t) from about 10 towards 4
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3, 4.2)     # P: inflection times P/R = 500 and 700
shape <- 2
scale <- 0.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scp=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHMM with TSCPG distribution

obj_ihmm2_m3 <- ihmm.scp(x = NULL, imc.par, scpg.par)
class(obj_ihmm2_m3)
# Simulating a 3-state IHMM with a TSCPG distribution

sim_ihmm2_m3 <- sim.ihmm.scp(obj_ihmm2_m3, nsim = 500, seed = 1)
str(sim_ihmm2_m3[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into the incomplete states become less likely
table(state = sim_ihmm2_m3$states,
      half = ifelse(seq_along(sim_ihmm2_m3$states) <= 250, "first", "second"))
# Fitting a 3-state IHMM with a TSCPG distribution
# (the parameters of obj_ihmm2_m3 are the starting values). The fit
# takes about 2.5 minutes, so it is wrapped in donttest.

fit_ihmm2_m3 <- fit.ihmm.scp(sim_ihmm2_m3)
fit_ihmm2_m3$imc.par$ttpm[, , 1] # r_ij (rows i, columns j = 2, 3)
fit_ihmm2_m3$imc.par$ttpm[, , 2] # w_ij
fit_ihmm2_m3$scpg.par$rate      # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_ihmm2_m3$logLik

# Smoothed state probabilities and residuals for the fitted 3-state IHMM
# with TSCPG distribution

fit_ihmm2_m3$x <- sim_ihmm2_m3$x
sp_ihmm2_m3 <- sp.ihmm.scp(fit_ihmm2_m3)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + sum_j Pr(S_t = j | x) / (2 + lambda_j(t)),
# averaged over the four quarters of the record. It combines the two kinds
# of time dependence (fewer visits to the incomplete states and fewer
# missing events in them) in one summary of how completeness changes.

```

```

h_ihmm2_m3 <- c(0, cumsum(sim_ihmm2_m3$x)[-500])
r_ihmm2_m3 <- fit_ihmm2_m3$scpg.par$rate
lam_ihmm2_m3 <- sapply(1:2, function(j)
  r_ihmm2_m3[1, j] + r_ihmm2_m3[2, j] /
  (1 + exp(r_ihmm2_m3[3, j] * h_ihmm2_m3 - r_ihmm2_m3[4, j])))
p_ihmm2_m3 <- sp_ihmm2_m3$smoothed.prob[, 1] +
  rowSums(sp_ihmm2_m3$smoothed.prob[, 2:3] / (2 + lam_ihmm2_m3))
tapply(p_ihmm2_m3, cut(seq_along(p_ihmm2_m3), 4), mean)

# Plotting of residuals to check whether the fitted IHMM with TSCPG renewal
# process is a good fit
dev.new()
res_ihmm2_m3 <- sp_ihmm2_m3$residuals
lres_ihmm2_m3 <- length(sp_ihmm2_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihmm2_m3, 1:lres_ihmm2_m3-res_ihmm2_m3, type="l",
  xlab="Transformed times",
  ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihmm2_m3,
  (1:lres_ihmm2_m3-sqrt(lres_ihmm2_m3)*1.36)-1:lres_ihmm2_m3, type="l", lty=2)
lines(res_ihmm2_m3,
  (1:lres_ihmm2_m3+sqrt(lres_ihmm2_m3)*1.36)-1:lres_ihmm2_m3, type="l", lty=2)
# 99% CI
lines(res_ihmm2_m3,
  (1:lres_ihmm2_m3-sqrt(lres_ihmm2_m3)*1.63)-1:lres_ihmm2_m3, type="l", lty=3)
lines(res_ihmm2_m3,
  (1:lres_ihmm2_m3+sqrt(lres_ihmm2_m3)*1.63)-1:lres_ihmm2_m3, type="l", lty=3)
lines(res_ihmm2_m3, rep(0, lres_ihmm2_m3), col="red")
par(oldpar)

```

sp.ihsmm.scpg

Smoothed probabilities for an m-state IHSMM with a gamma or an SCPG or a TSCPG renewal process

Description

Calculates the probability of being in State j at time t for an IHSMM with class `ihsmm.gamma`, `ihsmm.scpg` and `ihsmm.tscpg` as typically returned by [ihsmm.scpg](#).

Usage

```
sp.ihsmm.scpg(ihsmmobj)
```

Arguments

`ihsmmobj` is a model object with class `ihsmm.gamma`, `ihsmm.scpg` and `ihsmm.tscpg` as typically returned by [ihsmm.scpg](#).

Details

The smoothed probability $\gamma_t(j) = \Pr(S_t = j \mid X_{1:T}; \theta)$ is obtained by the extension of the forward-backward algorithm of Guedon (2003) given by Shahzadi et al. (2023). With $\alpha_t(j)$ and $\beta_t(j)$ recording respectively that a visit to State j ends at time t and the conditional probability of the remaining observations, and with $\alpha_{t+1}^*(j)$ and $\beta_{t+1}^*(j)$ the corresponding quantities at the commencement of a visit, the smoothed probability satisfies the backward recursion

$$\gamma_t(j) = \gamma_{t+1}(j) + \frac{\alpha_t(j)\beta_t(j)}{L(\theta)} - \frac{\alpha_{t+1}^*(j)\beta_{t+1}^*(j)}{L(\theta)},$$

with $\gamma_T(j) = \alpha_T(j)/L(\theta)$. The sojourn mass and survival functions are evaluated at each admissible commencement time from the sojourn parameter $\mu_j(t)$ prevailing there, so that visits beginning at different points of the record are weighted differently. The recursions are scaled at each time step.

Plotted against the occurrence times, the smoothed probabilities exhibit the drift for which the inhomogeneous specification was introduced: the probability of State 1 consolidates into progressively longer intervals as the record approaches the present, while the incompleteness states are confined to its earlier portion. The globally optimal sequence of visits is obtained from [vip.ihsmm.scpg](#).

Value

returns an $T \times m$ matrix for the smoothed probabilities and a numeric vector of residuals from an IHSMM with class [ihsmm.scpg](#).

References

- Guédon Y (2003). “Estimating hidden semi-Markov chains from discrete sequences.” *Journal of Computational and Graphical Statistics*, **12**(3), 604–639. doi:[10.1198/1061860032030](#).
- Shahzadi A, Wang T, Parry M, Bebbington M (2023). “Inhomogeneous hidden semi-Markov models for incompletely observed point processes.” *Annals of Institute Statistical Mathematics*, **75**, 253–280. doi:[10.1007/s10463022008435](#).
- Daley DJ, Vere-Jones D (2003). *An Introduction to the Theory of Point Processes*. Springer, New York. doi:[10.1007/b97277](#).

Examples

```
# =====
# An example of a 2-state IHSMM with an SCPG renewal process
# =====

# The sojourn parameters mu_j(t) depend on the time at which a visit
# commences (rows A, B, C, D of tdur): the expected sojourn 1 + mu_1(t) in
# the complete State 1 increases, and that in the incomplete State 2
# decreases, between the levels 1 + A_j = 1.5 and 1 + A_j + B_j = 5.5
# observations, around the inflection time D_j/C_j = 500. In State 2,
# the number N of events missing between two observed events is shifted
# Poisson with parameter lambda_2.
# With m = 2 the chain alternates between the two states.
```

```

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5,      # A
  4, 4,          # B
  0.005, 0.005, # C
  2.5, 2.5      # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHSMM with SCPG distribution

obj_ihsmm1 <- ihsmm.scpg(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm1)
# Simulating a 2-state IHSMM with SCPG distribution

sim_ihsmm1 <- sim.ihsmm.scpg(obj_ihsmm1, nsim = 150, seed = 3)
str(sim_ihsmm1[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm1$states)$values)
# The fit takes about 30 s, so the code below is wrapped in donttest.

# Fitting a 2-state IHSMM with SCPG distributions
# (the parameters of obj_ihsmm1 are the starting values)

fit_ihsmm1 <- fit.ihsmm.scpg(sim_ihsmm1)
fit_ihsmm1$ismc.par$tdur      # A, B, C, D of mu_1(t) and mu_2(t)
fit_ihsmm1$scpg.par$rate      # lambda_2 (true 3)
fit_ihsmm1$scpg.par$shape     # true 1.5
fit_ihsmm1$scpg.par$scale     # true 1
fit_ihsmm1$logLik

# The fit takes about 30 s, so the code below is wrapped in donttest.

# Smoothed probabilities and residuals for a 2-state IHSMM with SCPG
# distribution

fit_ihsmm1$x <- sim_ihsmm1$x
sp_ihsmm1 <- sp.ihsmm.scpg(fit_ihsmm1)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + Pr(S_t = 2 | x) / (2 + lambda_2), averaged over the
# four quarters of the record (row p) and compared with the true proportion

```

```

# (row p.true: 1 in State 1 and 1/(2 + lambda_2) in State 2). With a
# constant lambda_2 the rise comes from the sojourns alone: the visits to
# State 1 become longer and those to State 2 shorter (row state.1).
p_ihsmm1 <- sp_ihsmm1$smoothed.prob[, 1] +
  sp_ihsmm1$smoothed.prob[, 2] / (2 + fit_ihsmm1$scpg.par$rate[1, 1])
p0_ihsmm1 <- ifelse(sim_ihsmm1$states == 1, 1,
  1 / (2 + obj_ihsmm1$scpg.par$rate[1, 1]))
q_ihsmm1 <- cut(seq_along(p_ihsmm1), 4)
rbind(p = tapply(p_ihsmm1, q_ihsmm1, mean),
  state.1 = tapply(sp_ihsmm1$smoothed.prob[, 1], q_ihsmm1, mean),
  p.true = tapply(p0_ihsmm1, q_ihsmm1, mean))

# Plotting of residuals to check whether the fitted IHSMM with SCPG renewal
# process is a good fit.

dev.new()
res_ihsmm1 <- sp_ihsmm1$residuals
lres_ihsmm1 <- length(sp_ihsmm1$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihsmm1, 1:lres_ihsmm1-res_ihsmm1, type="l", xlab="Transformed times",
  ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihsmm1, (1:lres_ihsmm1-sqrt(lres_ihsmm1)*1.36)-1:lres_ihsmm1,
  type="l", lty=2)
lines(res_ihsmm1, (1:lres_ihsmm1+sqrt(lres_ihsmm1)*1.36)-1:lres_ihsmm1,
  type="l", lty=2)
# 99% CI
lines(res_ihsmm1, (1:lres_ihsmm1-sqrt(lres_ihsmm1)*1.63)-1:lres_ihsmm1,
  type="l", lty=3)
lines(res_ihsmm1, (1:lres_ihsmm1+sqrt(lres_ihsmm1)*1.63)-1:lres_ihsmm1,
  type="l", lty=3)
lines(res_ihsmm1, rep(0, lres_ihsmm1), col="red")
par(oldpar)

# =====
# An example of a 3-state IHSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 30 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
  0.5, 0, 0.5,
  0.5, 0.5, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.3, 0.3, 0.3, # A
  1.5, 2, 1.5, # B
  0.005, 0.005, 0.005, # C
  3, 3, 3 # D: inflection times D/C = 600
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,

```

```

      tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHSMM with SCPG distribution

obj_ihsmm1_m3 <- ihsmm.scpg(x = NULL, ismc.par, scpg.par)
# Simulating a 3-state IHSMM with SCPG distribution

sim_ihsmm1_m3 <- sim.ihsmm.scpg(obj_ihsmm1_m3, nsim = 150, seed = 1)
print(sim_ihsmm1_m3)
# Fitting a 3-state IHSMM with SCPG distributions
# The fit takes about 2 minutes, so it is wrapped in donttest.

fit_ihsmm1_m3 <- fit.ihsmm.scpg(sim_ihsmm1_m3)
print(fit_ihsmm1_m3)

# Computing smoothed probabilities and residuals

fit_ihsmm1_m3$x <- sim_ihsmm1_m3$x
sp_ihsmm1_m3 <- sp.ihsmm.scpg(fit_ihsmm1_m3)

# Plotting of residuals to check whether the fitted IHSMM with SCPG renewal
# process is a good fit.

dev.new()
res_ihsmm1_m3 <- sp_ihsmm1_m3$residuals
lres_ihsmm1_m3 <- length(sp_ihsmm1_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihsmm1_m3, 1:lres_ihsmm1_m3-res_ihsmm1_m3, type="l",
     xlab="Transformed times",
     ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihsmm1_m3,
      (1:lres_ihsmm1_m3-sqrt(lres_ihsmm1_m3)*1.36)-1:lres_ihsmm1_m3, type="l",
      lty=2)
lines(res_ihsmm1_m3,
      (1:lres_ihsmm1_m3+sqrt(lres_ihsmm1_m3)*1.36)-1:lres_ihsmm1_m3, type="l",
      lty=2)
# 99% CI
lines(res_ihsmm1_m3,
      (1:lres_ihsmm1_m3-sqrt(lres_ihsmm1_m3)*1.63)-1:lres_ihsmm1_m3, type="l",
      lty=3)
lines(res_ihsmm1_m3,
      (1:lres_ihsmm1_m3+sqrt(lres_ihsmm1_m3)*1.63)-1:lres_ihsmm1_m3, type="l",
      lty=3)
lines(res_ihsmm1_m3, rep(0, lres_ihsmm1_m3), col="red")
par(oldpar)

```

```

# =====
# An example of a 2-state IHSMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur): the expected
#   sojourn 1 +  $\mu_1(t)$  in the complete State 1 increases, and that in the
#   incomplete State 2 decreases, between the levels 1 +  $A_j = 2$  and
#   1 +  $A_j + B_j = 8$  observations, around the inflection time
#    $D_j/C_j = 500$ ;
# - the Poisson mean  $\lambda_2(t)$  of the TSCPG (rows K, M, R, P of rate)
#   falls from about 5.8 at the start of the record towards  $K = 0.5$ ,
#   around the inflection time  $P/R = 400$ .

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  1, 1,          # A
  6, 6,          # B
  0.005, 0.005,  # C
  2.5, 2.5       # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5   # K
rate[2, ] <- 6     # M
rate[3, ] <- 0.005 # R
rate[4, ] <- 2     # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state IHSMM with TSCPG distribution

obj_ihsmm2 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2)
# Simulating a 2-state IHSMM with a TSCPG distribution

sim_ihsmm2 <- sim.ihsmm.scp(obj_ihsmm2, nsim = 200, seed = 1)
str(sim_ihsmm2[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2$states)$values)
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state IHSMM with a TSCPG distribution
# (the parameters of obj_ihsmm2 are the starting values)

```

```

fit_ihsmm2 <- fit.ihsmm.scp(sim_ihsmm2)
fit_ihsmm2$ismc.par$tdur      # A, B, C, D of mu_1(t) and mu_2(t)
fit_ihsmm2$scpg.par$rate      # K, M, R, P of lambda_2(t)
fit_ihsmm2$scpg.par$scale     # true 1
fit_ihsmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# Smoothed state probabilities and residuals for the fitted 2-state IHSMM
# with TSCPG distribution

fit_ihsmm2$x <- sim_ihsmm2$x
sp_ihsmm2 <- sp.ihsmm.scp(fit_ihsmm2)

# Poisson mean lambda_2(t), fitted and true, at the last occurrence times
# h_{t-1}, and the four quarters of the record
lambda_t <- function(r, h) r[1] + r[2] / (1 + exp(r[3] * h - r[4]))
h_ihsmm2 <- c(0, cumsum(sim_ihsmm2$x)[-200])
lam_ihsmm2 <- lambda_t(fit_ihsmm2$scpg.par$rate[, 1], h_ihsmm2)
lam0_ihsmm2 <- lambda_t(obj_ihsmm2$scpg.par$rate[, 1], h_ihsmm2)
gam_ihsmm2 <- sp_ihsmm2$smoothed.prob
q_ihsmm2 <- cut(seq_along(h_ihsmm2), 4)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + Pr(S_t = 2 | x) / (2 + lambda_2(t)),
# averaged over each quarter (row p), with its two parts: the share of
# State 1 (row state.1) and the proportion recorded while the chain is in
# State 2 (row state.2). The true proportion (row p.true) is 1 in State 1
# and 1/(2 + lambda_2(t)) in State 2. Both kinds of time dependence raise
# p_t: the visits to State 1 become longer and those to State 2 shorter,
# and fewer events are missed in State 2.
p_ihsmm2 <- gam_ihsmm2[, 1] + gam_ihsmm2[, 2] / (2 + lam_ihsmm2)
p0_ihsmm2 <- ifelse(sim_ihsmm2$states == 1, 1, 1 / (2 + lam0_ihsmm2))
rbind(p = tapply(p_ihsmm2, q_ihsmm2, mean),
      state.1 = tapply(gam_ihsmm2[, 1], q_ihsmm2, mean),
      state.2 = tapply(gam_ihsmm2[, 2] / (2 + lam_ihsmm2), q_ihsmm2, sum) /
        tapply(gam_ihsmm2[, 2], q_ihsmm2, sum),
      p.true = tapply(p0_ihsmm2, q_ihsmm2, mean))

# Plotting the residuals (cumulative number of events minus rescaled time)
# with Kolmogorov-Smirnov bands, to check whether the fitted model is a good
# fit

dev.new()
res_ihsmm2 <- sp_ihsmm2$residuals
lres_ihsmm2 <- length(sp_ihsmm2$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihsmm2, 1:lres_ihsmm2-res_ihsmm2, type="l", xlab="Transformed times",
     ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihsmm2, (1:lres_ihsmm2-sqrt(lres_ihsmm2)*1.36)-1:lres_ihsmm2,
     type="l", lty=2)

```

```

lines(res_ihsmm2, (1:lres_ihsmm2+sqrt(lres_ihsmm2)*1.36)-1:lres_ihsmm2,
      type="l", lty=2)
# 99% CI
lines(res_ihsmm2, (1:lres_ihsmm2-sqrt(lres_ihsmm2)*1.63)-1:lres_ihsmm2,
      type="l", lty=3)
lines(res_ihsmm2, (1:lres_ihsmm2+sqrt(lres_ihsmm2)*1.63)-1:lres_ihsmm2,
      type="l", lty=3)
lines(res_ihsmm2, rep(0, lres_ihsmm2), col="red")
par(oldpar)

# =====
# An example of a 3-state IHSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 8 minutes on a laptop (the
# 2-state model above takes about 50 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur), with
#   inflection times  $D_j/C_j = 500$ : the expected sojourn  $1 + \mu_1(t)$  in the
#   complete State 1 increases from about 2.1 towards 6.5 observations,
#   and those in the incomplete States 2 and 3 decrease from about 5.9
#   towards 1.5;
# - the Poisson means  $\lambda_j(t)$  of the TSCPG (rows K, M, R, P of rate):
#    $\lambda_2(t)$  falls from about 3.3 towards 0.5 around 400, and
#    $\lambda_3(t)$  from about 9.8 towards 4 around 600.
# The zero-diagonal transition matrix gives the next state of each visit.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5, 0.5,      # A
  5, 5, 5,           # B
  0.004, 0.004, 0.004, # C
  2, 2, 2           # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(2.4, 3.6)    # P: inflection times P/R = 400 and 600
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,

```

```

WBsum.method = 'direct.evaluation')

# Defining an object of 3-state IHSMM with TSCPG distribution

obj_ihsmm2_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2_m3)
# Simulating a 3-state IHSMM with a TSCPG distribution

sim_ihsmm2_m3 <- sim.ihsmm.scp(obj_ihsmm2_m3, nsim = 250, seed = 2)
str(sim_ihsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the time-dependent components are evaluated
# on [0, h_{T-1}]
c(h_T1 = sum(sim_ihsmm2_m3$x[-250]), h_T = sum(sim_ihsmm2_m3$x))

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2_m3$states)$values)
# Fitting a 3-state IHSMM with a TSCPG distribution
# (the parameters of obj_ihsmm2_m3 are the starting values). The fit
# takes about 8 minutes, so it is wrapped in donttest.

fit_ihsmm2_m3 <- fit.ihsmm.scp(sim_ihsmm2_m3)
fit_ihsmm2_m3$ismc.par$tpm      # transition matrix of the embedded chain
fit_ihsmm2_m3$ismc.par$tdur     # A, B, C, D of mu_1(t), mu_2(t) and mu_3(t)
fit_ihsmm2_m3$scpg.par$rate     # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_ihsmm2_m3$scpg.par$scale    # true 0.5
fit_ihsmm2_m3$logLik

# Smoothed state probabilities and residuals for the fitted 3-state IHSMM
# with TSCPG distribution

fit_ihsmm2_m3$x <- sim_ihsmm2_m3$x
sp_ihsmm2_m3 <- sp.ihsmm.scp(fit_ihsmm2_m3)

# Poisson means lambda_2(t) and lambda_3(t), fitted and true, at the last
# occurrence times h_{t-1}, and the four quarters of the record
lambda_t <- function(r, h) r[1] + r[2] / (1 + exp(r[3] * h - r[4]))
h_ihsmm2_m3 <- c(0, cumsum(sim_ihsmm2_m3$x)[-250])
lam_ihsmm2_m3 <- sapply(1:2,
  function(j) lambda_t(fit_ihsmm2_m3$scpg.par$rate[, j], h_ihsmm2_m3))
lam0_ihsmm2_m3 <- sapply(1:2,
  function(j) lambda_t(obj_ihsmm2_m3$scpg.par$rate[, j], h_ihsmm2_m3))
gam_ihsmm2_m3 <- sp_ihsmm2_m3$smoothed.prob
q_ihsmm2_m3 <- cut(seq_along(h_ihsmm2_m3), 4)

# Proportion of events recorded around each observation,
# p_t = Pr(S_t = 1 | x) + sum_j Pr(S_t = j | x) / (2 + lambda_j(t)),
# averaged over each quarter (row p), with its two parts: the share of
# State 1 (row state.1) and the proportion recorded while the chain is in
# States 2 and 3 (row states.2.3). The true proportion (row p.true) is 1
# in State 1 and 1/(2 + lambda_j(t)) in State j. Here p_t follows the true
# proportion, and most of its rise comes from the sojourns: the visits to

```

```

# the complete State 1 become longer and those to States 2 and 3 shorter,
# so the share of State 1 rises steeply, while the proportion recorded in
# States 2 and 3 changes less.
p_ihsmm2_m3 <- gam_ihsmm2_m3[, 1] +
  rowSums(gam_ihsmm2_m3[, 2:3] / (2 + lam_ihsmm2_m3))
s_ihsmm2_m3 <- sim_ihsmm2_m3$states
p0_ihsmm2_m3 <- ifelse(s_ihsmm2_m3 == 1, 1,
  1 / (2 + lam0_ihsmm2_m3[cbind(seq_along(s_ihsmm2_m3),
    pmax(s_ihsmm2_m3 - 1, 1))]))
rbind(p = tapply(p_ihsmm2_m3, q_ihsmm2_m3, mean),
  state.1 = tapply(gam_ihsmm2_m3[, 1], q_ihsmm2_m3, mean),
  states.2.3 = tapply(rowSums(gam_ihsmm2_m3[, 2:3] / (2 + lam_ihsmm2_m3)),
    q_ihsmm2_m3, sum) /
    tapply(rowSums(gam_ihsmm2_m3[, 2:3]), q_ihsmm2_m3, sum),
  p.true = tapply(p0_ihsmm2_m3, q_ihsmm2_m3, mean))

# Plotting the residuals (cumulative number of events minus rescaled time)
# with Kolmogorov-Smirnov bands, to check whether the fitted model is a good
# fit

dev.new()
res_ihsmm2_m3 <- sp_ihsmm2_m3$residuals
lres_ihsmm2_m3 <- length(sp_ihsmm2_m3$residuals)
oldpar <- par(mgp=c(2.6, 0.8, 0), mar=c(4, 4.5, 0.5, 0.5), cex.lab=1.4, cex.axis=1.2)
plot(res_ihsmm2_m3, 1:lres_ihsmm2_m3-res_ihsmm2_m3, type="l",
  xlab="Transformed times",
  ylab="Transformed cumsum", ylim=c(-50, 50))
# 95% CI
lines(res_ihsmm2_m3,
  (1:lres_ihsmm2_m3-sqrt(lres_ihsmm2_m3)*1.36)-1:lres_ihsmm2_m3, type="l",
  lty=2)
lines(res_ihsmm2_m3,
  (1:lres_ihsmm2_m3+sqrt(lres_ihsmm2_m3)*1.36)-1:lres_ihsmm2_m3, type="l",
  lty=2)
# 99% CI
lines(res_ihsmm2_m3,
  (1:lres_ihsmm2_m3-sqrt(lres_ihsmm2_m3)*1.63)-1:lres_ihsmm2_m3, type="l",
  lty=3)
lines(res_ihsmm2_m3,
  (1:lres_ihsmm2_m3+sqrt(lres_ihsmm2_m3)*1.63)-1:lres_ihsmm2_m3, type="l",
  lty=3)
lines(res_ihsmm2_m3, rep(0, lres_ihsmm2_m3), col="red")
par(oldpar)

```

Description

Calculates the Viterbi path for an HMM with class `hmm.gamma`, `hmm.scp` and `hmm.tscpg` as typically returned by `hmm.scp`.

Usage

```
vip.hmm.scp(hmmobj)
```

Arguments

`hmmobj` An object created by `hmm.scp`, usually the result of `fit.hmm.scp`, with the data attached as `hmmobj$x`.

Details

Global decoding determines the single most probable hidden state sequence given the observations,

$$S^* = \arg \max_S \Pr(S \mid X; \theta) = \arg \max_S \Pr(S, X \mid \theta),$$

the denominator $\Pr(X)$ being constant in S . This differs from the local decoding obtained from the smoothed probabilities of `sp.hmm.scp`, which maximizes at each time separately; the globally optimal sequence is always admissible under the model, whereas the sequence of pointwise maximizers need not be.

The maximization is performed by the Viterbi algorithm (Viterbi, 1967), which replaces the summation of the forward recursion by a maximization. Defining $\delta_t(j) = \max_{s_1, \dots, s_{t-1}} \Pr(S_1 = s_1, \dots, S_t = j, X_1, \dots, X_t \mid \theta)$,

$$\delta_1(j) = \pi_j f_j(x_1), \quad \delta_t(j) = \left\{ \max_i \delta_{t-1}(i) a_{ij} \right\} f_j(x_t), \quad t = 2, \dots, T,$$

and the optimal sequence is recovered by backtracking from $s_T^* = \arg \max_j \delta_T(j)$ through $s_t^* = \arg \max_i \{ \delta_t(i) a_{is_{t+1}^*} \}$. The recursion is evaluated on the logarithmic scale, products becoming sums and underflow being thereby avoided.

Since State 1 denotes completeness and States $2, \dots, m$ successive degrees of incompleteness, the decoded sequence is interpreted as an estimate of which portions of the record are complete and of the number of events missing elsewhere.

Value

An integer vector of length T with the most probable state for each interevent time.

References

Viterbi AJ (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Transactions on Information Theory*, **13**(2), 260-269.

Examples

```
# =====
# An example of a 2-state HMM with an SCPG renewal process
# =====

# State 1 is the state of completeness: its interevent times are gamma with
# shape alpha and scale beta. In the incomplete State 2 the number of
# missing events N between two observed events is shifted Poisson with
# parameter lambda_2, so on average 1 + lambda_2 = 4 events are missing,
# and the observed interevent time is the sum of N + 1 such gamma
# interevent times. The shape is common to both states and, with
# shape.fix = FALSE, estimated; a separate shape for each state (a vector
# of shapes) is also possible but is less well determined.

m <- 2
tpm <- matrix(c(0.9, 0.1,
                0.2, 0.8), nrow = 2, byrow = TRUE)
mc.par <- list(initd = NULL, tpm = tpm, stationary = TRUE)
rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
                 WBSum.method = 'direct.evaluation')

# Defining an object for 2-state HMM with SCPG distribution

obj_hmm1 <- hmm.scpg(x = NULL, mc.par, scpg.par)
print(obj_hmm1)
# Simulating a 2-state HMM with SCPG distribution

sim_hmm1 <- sim.hmm.scpg(obj_hmm1, nsim = 200, seed = 8)
str(sim_hmm1[c("x", "states")])

# Observations in each state
table(sim_hmm1$states)
# Fitting a 2-state HMM with SCPG distributions
# (the parameters of obj_hmm1 are the starting values)

fit_hmm1 <- fit.hmm.scpg(sim_hmm1)
# transition matrix (true rows 0.9, 0.1 and 0.2, 0.8)
fit_hmm1$mc.par$tpm
fit_hmm1$scpg.par$rate # lambda_2 (true 3)
fit_hmm1$scpg.par$shape # alpha (true 1.5)
fit_hmm1$scpg.par$scale # beta (true 1)
fit_hmm1$logLik
# The Viterbi path of the fitted 2-state HMM with SCPG distribution,
# compared with the simulated states

fit_hmm1$x <- sim_hmm1$x
vip_hmm1 <- vip.hmm.scpg(fit_hmm1)
table(true = sim_hmm1$states, decoded = vip_hmm1)
```

```

# Plotting of the Viterbi path for the fitted model

dev.new()
oldpar <- par(mfrow=c(2, 1), cex.lab = 1.3, cex.axis = 1.2, mgp=c(2.3,0.8,0))
par(mar=c(0, 4.0, 2, 3)) # bottom margin is 0.
plot(cumsum(sim_hmm1$x), sim_hmm1$x, type="l", ylab = "Interevent times" ,
     xaxt="n")
mtext("Simulated data for a 2-state HMM with SCPG", side = 4, line = 0.8,
     cex = 1.1, las = 0)
par(mar=c(3.5, 4.0, 0, 3)) # top is 0.
plot(cumsum(sim_hmm1$x), vip_hmm1, type="s", yaxt="n",
     xlab="Occurrence Time", ylab="States")
axis(2, at=1:2, labels=1:2)
mtext("Viterbi path for a 2-state HMM with SCPG", side = 4, line = 0.8,
     cex = 1.1, las = 0)
par(oldpar)
# =====
# An example of a 3-state HMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 2 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(1/3, 1/3, 1/3)
tpm <- matrix(c(0.2, 0.65, 0.15, 0.60, 0.30, 0.10, 0.45, 0.30, 0.25),
              nrow = 3, byrow = TRUE)
mc.par <- list(initd=NULL, tpm=tpm, stationary=TRUE)
rate <- matrix(c(7, 3), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBSum.method='direct.evaluation')

# Defining an object for 3-state HMM with SCPG distribution

obj_hmm1_m3 <- hmm.scpg(x = NULL, mc.par, scpg.par)
print(obj_hmm1_m3)
# Simulating a 3-state HMM with SCPG distribution

sim_hmm1_m3 <- sim.hmm.scpg(obj_hmm1_m3, nsim = 200, seed = 2)
print(sim_hmm1_m3)
# Fitting a 3-state HMM with SCPG distributions
# The fit takes about 1.5 minutes, so it is wrapped in donttest.

fit_hmm1_m3 <- fit.hmm.scpg(sim_hmm1_m3)
print(fit_hmm1_m3)

# The Viterbi path of fitted 3-state HMM with SCPG distribution

```

```

fit_hmm1_m3$x <- sim_hmm1_m3$x
vip_hmm1_m3 <- vip.hsmm.scpG(fit_hmm1_m3)
print(vip_hmm1_m3)

# Plotting of the Viterbi path for the fitted model

dev.new()
oldpar <- par(mfrow=c(2, 1), cex.lab = 1.3, cex.axis = 1.2, mgp=c(2.3,0.8,0))
par(mar=c(0, 4.0, 2, 3)) # bottom margin is 0.
plot(cumsum(sim_hmm1_m3$x), sim_hmm1_m3$x, type="l",
     ylab = "Interevent times" , xaxt="n")
mtext("Simulated data for a 3-state HMM with SCPG", side = 4, line = 0.8,
     cex = 1.1, las = 0)
par(mar=c(3.5, 4.0, 0, 3)) # top is 0.
plot(cumsum(sim_hmm1_m3$x), vip_hmm1_m3, type="s", yaxt="n",
     xlab="Occurrence Time", ylab="States")
axis(2, at=1:3, labels=1:3)
mtext("Viterbi path for a 3-state HMM with SCPG", side = 4, line = 0.8,
     cex = 1.1, las = 0)
par(oldpar)

```

vip.hsmm.scpG

*Viterbi path for an m-state HSMM with gamma or SCPG or TSCPG
renewal processes*

Description

Calculates the Viterbi path for an HMM with class `hsmm.gamma`, `hsmm.scpG` and `hsmm.tscpg` as typically returned by [hsmm.scpG](#).

Usage

```
vip.hsmm.scpG(hsmmobj)
```

Arguments

`hsmmobj` is a model object with class `hsmm.gamma`, `hsmm.scpG` and `hsmm.tscpg` as typically returned by [hsmm.scpG](#).

Details

In a hidden semi-Markov model the hidden path is a sequence of visits, each with an associated sojourn time, so that global decoding maximizes over states and durations jointly. Following Guedon (2003) and Yu (2010), define

$$\delta_t(j, d) = \max_{S_{1:(t-d)}} \Pr(S_{1:(t-d)}, S_{(t-d+1):t} = j, S_{t+1} \neq j, X_{1:t} \mid \theta),$$

the maximum probability of a path in which a visit to State j of length d ends at time t . For $2 \leq t \leq T - 1$,

$$\delta_t(j, d) = \max_{i \neq j} \max_h \{ \delta_{t-d}(i, h) a_{ij} \} p_j(d) \prod_{l=t-d+1}^t f_j(x_l),$$

with initial condition $\delta_t(j, d) = \pi_j p_j(d) \prod_{l=1}^t f_j(x_l)$ for $t = d$. At $t = T$ the sojourn in the last visited state is right-censored and the sojourn mass function $p_j(d)$ is replaced by the survival function $S_j(d)$, so that the decoded path is optimal under the same right-censored likelihood as is maximized by `fit.hsmm.scp`. The maximizing predecessor and its duration are recorded in $\psi(t, j, d) = (t - d, i^*, h^*)$ and the path recovered by backtracking from $\max_{j,d} \delta_T(j, d)$. The recursion is evaluated on the logarithmic scale.

The decoded path is returned as a sequence of visits rather than as one state per observation, each row giving the index of the last observation of a visit (column "t-d"), the state occupied ("j"), and its duration ("d"); the rows run backwards from the end of the record, and the visit of a row covers observations t-d - d + 1 to t-d. Consecutive rows necessarily occupy distinct states, the embedded chain admitting no self-transitions.

Value

returns the most probable state sequence responsible for the observed sequence using the Viterbi algorithm with a fitted HSMM with class `hsmm.scp`.

References

- Guédon Y (2003). "Estimating hidden semi-Markov chains from discrete sequences." *Journal of Computational and Graphical Statistics*, **12**(3), 604-639. doi:10.1198/1061860032030.
- Yu SZ (2010). "Hidden semi-Markov models." *Artificial Intelligence*, **174**, 215-243. doi:10.1016/j.artint.2009.11.011.

Examples

```
# =====
# An example of a 2-state HSMM with an SCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times with shape
# alpha); in the incomplete State 2 the number N of events missing between
# two observed events is shifted Poisson with parameter lambda_2, so the
# observed events are N + 1 such gamma interevent times apart. The sojourns
# are shifted Poisson with
# parameters dur (expected sojourns 1 + dur = 5 and 4 observations). With
# m = 2 the zero-diagonal transition matrix is fixed: the chain alternates
# between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
```

```

rate <- matrix(3, nrow = 1, ncol = m - 1) # lambda_2
shape <- 1.5                               # common shape alpha
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = FALSE, time.scpg = FALSE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with SCPG distribution

obj_hsmm1 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm1)
# Simulating a 2-state HSMM with SCPG distribution

sim_hsmm1 <- sim.hsmm.scpg(obj_hsmm1, nsim = 150, seed = 1)
str(sim_hsmm1[c("x", "states")])

# Number of visits to each state in the simulated path
table(rle(sim_hsmm1$states)$values)
# Fitting a 2-state HSMM with SCPG distributions
# (the parameters of obj_hsmm1 are the starting values)

fit_hsmm1 <- fit.hsmm.scpg(sim_hsmm1)
fit_hsmm1$smc.par$dur      # sojourn parameters mu_1, mu_2 (true 4, 3)
fit_hsmm1$scpg.par$rate    # lambda_2 (true 3)
fit_hsmm1$scpg.par$shape   # alpha (true 1.5)
fit_hsmm1$scpg.par$scale   # beta (true 1)
fit_hsmm1$logLik
# The fit takes about 6 s, so the code below is wrapped in donttest.

# The Viterbi path of the fitted 2-state HSMM with SCPG distribution.
# Each row describes one decoded visit: the index of its last observation,
# its state and its duration (rows run from the end of the record).

fit_hsmm1$x <- sim_hsmm1$x
vip_hsmm1 <- vip.hsmm.scpg(fit_hsmm1)
head(vip_hsmm1)

# Expanding the visits into one state per observation and comparing the
# decoded states with the simulated ones
path_hsmm1 <- integer(length(sim_hsmm1$x))
for (r in seq_len(nrow(vip_hsmm1))) {
  last <- vip_hsmm1[r, 1]
  path_hsmm1[(last - vip_hsmm1[r, 3] + 1):last] <- vip_hsmm1[r, 2]
}
table(true = sim_hsmm1$states, decoded = path_hsmm1)

# =====
# An example of a 3-state HSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 15 seconds on a laptop (the
# 2-state model above takes about 1 second), so the examples that fit
# it are wrapped in donttest.

```

```

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.3, 0.7, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd=NULL, tpm=tpm, dur=c(3, 5, 2),
                dur.lb=NA, dur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state HSMM with SCPG distribution

obj_hsmm1_m3 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
# Simulating a 3-state HSMM with SCPG distribution

sim_hsmm1_m3 <- sim.hsmm.scpg(obj_hsmm1_m3, nsim = 80, seed = 1)
print(sim_hsmm1_m3)
# Fitting a 3-state HSMM with SCPG distributions
# The fit takes about 15 seconds, so it is wrapped in donttest.

fit_hsmm1_m3 <- fit.hsmm.scpg(sim_hsmm1_m3)
print(fit_hsmm1_m3)

# The Viterbi path of fitted 3-state HSMM with SCPG distribution

fit_hsmm1_m3$x <- sim_hsmm1_m3$x
vip_hsmm1_m3 <- vip.hsmm.scpg(fit_hsmm1_m3)
print(vip_hsmm1_m3)

# =====
# An example of a 2-state HSMM with a TSCPG renewal process
# =====

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete State 2 the number of missing events N between two observed
# events is shifted Poisson with parameter  $\lambda_2(t)$ , where
#  $\lambda_2(t) = K + M / (1 + \exp(R * h_{\{t-1\}} - P))$ 
# falls from about 5.8 at the start of the record towards  $K = 0.5$ , around
# the inflection time  $P/R = 400$  of the last occurrence time  $h_{\{t-1\}}$ . The
# sojourns are shifted Poisson with parameters dur (expected sojourns
#  $1 + \text{dur} = 5$  and 4 observations). With  $m = 2$  the zero-diagonal transition
# matrix is fixed: the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)

```

```

rate[1, ] <- 0.5    # K
rate[2, ] <- 6      # M
rate[3, ] <- 0.005  # R
rate[4, ] <- 2      # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBsum.method = 'direct.evaluation')

# Defining an object of 2-state HSMM with TSCPG distribution

obj_hsmm2 <- hsmm.scp(x = NULL, smc.par, scpg.par)
class(obj_hsmm2)
# Simulating a 2-state HSMM with a TSCPG distribution

sim_hsmm2 <- sim.hsmm.scp(obj_hsmm2, nsim = 300, seed = 3)
str(sim_hsmm2[c("x", "states")])

# The record covers [0, h_T]; the TSCPG rate is evaluated on [0, h_{T-1}]
c(h_T1 = sum(sim_hsmm2$x[-300]), h_T = sum(sim_hsmm2$x))
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state HSMM with a TSCPG distribution
# (the parameters of obj_hsmm2 are the starting values)

fit_hsmm2 <- fit.hsmm.scp(sim_hsmm2)
fit_hsmm2$smc.par$dur      # sojourn parameters mu_1, mu_2 (true 4, 3)
fit_hsmm2$scpg.par$rate    # K, M, R, P of lambda_2(t)
fit_hsmm2$scpg.par$scale   # true 1
fit_hsmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# The Viterbi path of the fitted 2-state HSMM with TSCPG distribution.
# Each row describes one decoded visit: the index of its last observation,
# its state and its duration (rows run from the end of the record).

fit_hsmm2$x <- sim_hsmm2$x
vip_hsmm2 <- vip.hsmm.scp(fit_hsmm2)
head(vip_hsmm2)

# Expanding the visits into one state per observation and comparing the
# decoded states with the simulated ones
path_hsmm2 <- integer(length(sim_hsmm2$x))
for (r in seq_len(nrow(vip_hsmm2))) {
  last <- vip_hsmm2[r, 1]
  path_hsmm2[(last - vip_hsmm2[r, 3] + 1):last] <- vip_hsmm2[r, 2]
}
table(true = sim_hsmm2$states, decoded = path_hsmm2)

# =====
# An example of a 3-state HSMM with a TSCPG renewal process

```

```

# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 7 seconds), so the examples that fit
# it are wrapped in donttest.

# State 1 is the state of completeness (gamma interevent times). In the
# incomplete States 2 and 3 the number of missing events N between two
# observed events is shifted Poisson with parameter  $\lambda_j(t)$ , where
#  $\lambda_j(t) = K_j + M_j / (1 + \exp(R_j * h_{t-1} - P_j))$ 
# falls around the inflection time  $P_j/R_j$  of the last occurrence time
#  $h_{t-1}$ :  $\lambda_2(t)$  from about 3.4 towards 0.5 around 600, and
#  $\lambda_3(t)$  from about 10 towards 4 around 900. The sojourns are shifted
# Poisson with parameters dur (expected sojourns  $1 + \text{dur} = 5, 4$  and  $4$ 
# observations), and the zero-diagonal transition matrix gives the next
# state of each visit. The shape 2 makes the interevent times of the three
# states less overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
smc.par <- list(initd = NULL, tpm = tpm, dur = c(4, 3, 3),
                dur.lb = NA, dur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4)      # K
rate[2, ] <- c(3, 6)       # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3.6, 5.4)    # P: inflection times P/R = 600 and 900
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBsum.method = 'direct.evaluation')

# Defining an object of 3-state HSMM with TSCPG distribution

obj_hsmm2_m3 <- hsmm.scpg(x = NULL, smc.par, scpg.par)
class(obj_hsmm2_m3)
# Simulating a 3-state HSMM with a TSCPG distribution

sim_hsmm2_m3 <- sim.hsmm.scpg(obj_hsmm2_m3, nsim = 400, seed = 3)
str(sim_hsmm2_m3[c("x", "states")])

# The record covers  $[0, h_T]$ ; the TSCPG rates are evaluated on  $[0, h_{T-1}]$ 
c(h_T1 = sum(sim_hsmm2_m3$x[-400]), h_T = sum(sim_hsmm2_m3$x))

# Number of visits to each state in the simulated path
table(rle(sim_hsmm2_m3$states)$values)
# Fitting a 3-state HSMM with a TSCPG distribution
# (the parameters of obj_hsmm2_m3 are the starting values). The fit
# takes about 2 minutes, so it is wrapped in donttest.

```

```

fit_hsmm2_m3 <- fit.hsmm.scpg(sim_hsmm2_m3)
fit_hsmm2_m3$smc.par$tpm      # transition matrix of the embedded chain
# sojourn parameters mu_1, mu_2, mu_3 (true 4, 3, 3)
fit_hsmm2_m3$smc.par$dur
fit_hsmm2_m3$scpg.par$rate     # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_hsmm2_m3$scpg.par$scale    # true 0.5
fit_hsmm2_m3$logLik

# The Viterbi path of the fitted 3-state HSMM with TSCPG distribution.
# Each row describes one decoded visit: the index of its last observation,
# its state and its duration (rows run from the end of the record).

fit_hsmm2_m3$x <- sim_hsmm2_m3$x
vip_hsmm2_m3 <- vip.hsmm.scpg(fit_hsmm2_m3)
head(vip_hsmm2_m3)

# Expanding the visits into one state per observation and comparing the
# decoded states with the simulated ones
path_hsmm2_m3 <- integer(length(sim_hsmm2_m3$x))
for (r in seq_len(nrow(vip_hsmm2_m3))) {
  last <- vip_hsmm2_m3[r, 1]
  path_hsmm2_m3[(last - vip_hsmm2_m3[r, 3] + 1):last] <- vip_hsmm2_m3[r, 2]
}
table(true = sim_hsmm2_m3$states, decoded = path_hsmm2_m3)

```

vip.ihmm.scpg	<i>Viterbi path for an m-state IHMM with gamma or SCPG or TSCPG renewal processes</i>
---------------	---

Description

Calculates the Viterbi path for an IHMM with class `ihmm.gamma`, `ihmm.scpg` and `ihmm.tscpg` as typically returned by [ihmm.scpg](#).

Usage

```
vip.ihmm.scpg(ihmmobj)
```

Arguments

`ihmmobj` is a model object with class `ihmm.gamma`, `ihmm.scpg` and `ihmm.tscpg` as typically returned by [ihmm.scpg](#).

Details

Global decoding of an IHMM determines the most probable hidden state sequence,

$$S^* = \arg \max_S \Pr(S, X \mid H; \theta),$$

by the Viterbi algorithm (Viterbi, 1967) with the transition probabilities replaced by their time-varying counterparts. Defining $\delta_t(j) = \max_{s_1, \dots, s_{t-1}} \Pr(S_1 = s_1, \dots, S_t = j, X_1, \dots, X_t \mid H_{t-1}; \theta)$,

$$\delta_1(j) = \pi_j f_j(x_1), \quad \delta_t(j) = \left\{ \max_i \delta_{t-1}(i) a_{ij}(t) \right\} f_j(x_t), \quad t = 2, \dots, T,$$

and the optimal sequence is recovered by backtracking from $s_T^* = \arg \max_j \delta_T(j)$, each step employing the transition probability matrix belonging to that time. The recursion is evaluated on the logarithmic scale.

Because the time-varying transition probabilities into State 1 increase with elapsed time, the decoded sequence of a correctly specified model ordinarily leaves the incompleteness states at some point and remains in State 1 thereafter. The occurrence time at which this change takes place estimates the point from which the record may be treated as complete, and is the principal quantity of interest in the applications reported by Shahzadi et al. (2026).

Value

returns the most probable state sequence responsible for the observed sequence using the Viterbi algorithm with a fitted IHMM with class `ihmm.scpG`.

References

- Viterbi AJ (1967). “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.” *IEEE Transactions on Information Theory*, **13**(2), 260-269.
- Shahzadi A, Wang T, Parry M, Bebbington M (2026). “Modelling time-inhomogeneous incomplete records of point processes using variants of hidden Markov models.” *Advances in Data Analysis and Classification*, **20**, 663-691. doi:10.1007/s1163402500632x.

Examples

```
# =====
# An example of a 2-state IHMM with an SCPG renewal process
# =====

# The transition probabilities a_12(t) and a_22(t) into the incomplete
# State 2 decrease along the record: the log-odds of moving from state i to
# State 2 rather than to State 1 is w_i2 - r_i2 * h_{t-1}, which is zero at
# the crossing time w_i2 / r_i2 (250 for a_12(t) and 500 for a_22(t)). In
# State 2 the number N of events missing between two observed events is
# shifted Poisson with parameter lambda_2. The chain starts mostly in State 2
# (initd = (0.2, 0.8)),
# since the early part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.004, 0.006) # rates of change r_12, r_22
ttpm[, , 2] <- c(1, 3)        # intercepts w_12, w_22
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
```

```

shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                 WBsum.method='direct.evaluation')

# Defining an object of 2-state IHMM with SCPG distribution

obj_ihmm1 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm1)
# Simulating a 2-state IHMM with SCPG distribution

sim_ihmm1 <- sim.ihmm.scpg(obj_ihmm1, nsim = 200, seed = 1)
str(sim_ihmm1[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into State 2 become less likely
table(state = sim_ihmm1$states,
      half = ifelse(seq_along(sim_ihmm1$states) <= 100, "first", "second"))
# The fit takes about 6 s, so the code below is wrapped in donttest.

# Fitting a 2-state IHMM with SCPG distributions
# (the parameters of obj_ihmm1 are the starting values)

fit_ihmm1 <- fit.ihmm.scpg(sim_ihmm1)
fit_ihmm1$imc.par$ttpm[, 1, ] # r_i2 (first column) and w_i2 (second)
fit_ihmm1$scpg.par$rate      # lambda_2 (true 3)
fit_ihmm1$scpg.par$shape     # true 1.5
fit_ihmm1$scpg.par$scale     # true 1
fit_ihmm1$logLik

# The fit takes about 6 s, so the code below is wrapped in donttest.

# The Viterbi path of the fitted 2-state IHMM with SCPG distribution,
# compared with the simulated states

fit_ihmm1$x <- sim_ihmm1$x
vip_ihmm1 <- vip.ihmm.scpg(fit_ihmm1)
table(true = sim_ihmm1$states, decoded = vip_ihmm1)

# Plotting of the Viterbi path with observed data

dev.new()
oldpar <- par(mfrow=c(2, 1), cex.lab = 1.3, cex.axis = 1.2, mgp=c(2.3,0.8,0))
par(mar=c(0, 4.0, 2, 3)) # bottom margin is 0.
plot(cumsum(sim_ihmm1$x), sim_ihmm1$x, type="l", ylab = "Interevent times",
     xaxt="n")
mtext("Simulated data for IHMM with SCPG", side = 4, line = 0.8, cex = 1.1,
     las = 0)
par(mar=c(3.5, 4.0, 0, 3)) # top is 0.
plot(cumsum(sim_ihmm1$x), vip_ihmm1, type="s", yaxt="n",
     xlab="Occurrence Time", ylab="States")
axis(2, at=1:2, labels=1:2)

```

```

mtext("Viterbi path for IHMM with SCPG", side = 4, line = 0.8, cex = 1.1,
      las = 0)
par(oldpar)

# =====
# An example of a 3-state IHMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 1.5 minutes on a laptop (the
# 2-state model above takes about 6 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
initd <- c(0.45, 0.25, 0.30)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change r_ij (rows i = 1, 2, 3; columns j = 2, 3)
ttpm[, , 1] <- matrix(c(0.0008, 0.0010, 0.0006,
                       0.0006, 0.0008, 0.0010), nrow = m, ncol = m-1)
# intercepts w_ij; a_ij(t) = a_i1(t) at h = w_ij/r_ij, between 2000 and 3000
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(2500, 2000, 3000,
                                       2000, 3000, 2500), nrow = m, ncol = m-1)

stationary <- FALSE
imc.par <- list(initd=initd, ttpm=ttpm, stationary=stationary)
rate <- matrix(c(7, 2), nrow = 1, ncol = m-1)
shape <- 1.7
scale <- 3.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                rate.fix=FALSE, shape.fix=FALSE, time.scpg=FALSE,
                WBsum.method='direct.evaluation')

# Defining an object of 3-state IHMM with SCPG distribution

obj_ihmm1_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
# Simulating a 3-state IHMM with SCPG distribution

sim_ihmm1_m3 <- sim.ihmm.scpg(obj_ihmm1_m3, nsim = 200, seed = 2)
print(sim_ihmm1_m3)
# Fitting a 3-state IHMM with SCPG distributions
# The fit takes about 1.5 minutes, so it is wrapped in donttest.

fit_ihmm1_m3 <- fit.ihmm.scpg(sim_ihmm1_m3)
print(fit_ihmm1_m3)

# The Viterbi path of fitted 3-state IHMM with SCPG distribution

fit_ihmm1_m3$x=sim_ihmm1_m3$x
vip_ihmm1_m3 <- vip.ihmm.scpg(fit_ihmm1_m3)
print(vip_ihmm1_m3)

# Plotting of the Viterbi path with observed data

dev.new()
oldpar <- par(mfrow=c(2, 1), cex.lab = 1.3, cex.axis = 1.2, mgp=c(2.3,0.8,0))

```

```

par(mar=c(0, 4.0, 2, 3)) # bottom margin is 0.
plot(cumsum(sim_ihmm1_m3$x), sim_ihmm1_m3$x, type="l",
     ylab = "Interevent times" , xaxt="n")
mtext("Simulated data for IHMM with SCPG", side = 4, line = 0.8, cex = 1.1,
      las = 0)
par(mar=c(3.5, 4.0, 0, 3)) # top is 0.
plot(cumsum(sim_ihmm1_m3$x), vip_ihmm1_m3, type="s", yaxt="n",
     xlab="Occurrence Time",
     ylab="States")
axis(2, at=1:3, labels=1:3)
mtext("Viterbi path for IHMM with SCPG", side = 4, line = 0.8, cex = 1.1,
      las = 0)
par(oldpar)

# =====
# An example of a 2-state IHMM with a TSCPG renewal process
# =====

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scp):
# - the transition probabilities  $a_{12}(t)$  and  $a_{22}(t)$  into the incomplete
#   State 2 decrease: the log-odds of moving from state  $i$  to State 2
#   rather than to State 1 is  $w_{i2} - r_{i2} * h_{\{t-1\}}$ , which is zero at the
#   crossing time  $w_{i2} / r_{i2}$  (about 304 for  $a_{12}(t)$  and 600 for  $a_{22}(t)$ );
# - the Poisson mean  $\lambda_{22}(t)$  of the TSCPG falls from about 5.4 at the
#   start of the record towards  $K = 0.5$ , around the inflection time
#    $P/R = 400$ .
# The chain starts mostly in State 2 ( $initd = (0.2, 0.8)$ ), since the early
# part of a record is usually the least complete.

m <- 2
initd <- c(0.2, 0.8)
ttpm <- array(NA, dim = c(m, m-1, 2))
ttpm[, , 1] <- c(0.0028, 0.005) # rates of change  $r_{12}$ ,  $r_{22}$ 
ttpm[, , 2] <- c(0.85, 3) # intercepts  $w_{12}$ ,  $w_{22}$ 
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(c(0.5, 5, 0.01, 4), nrow = 4) # K, M, R, P:  $P/R = 400$ 
shape <- 1
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scp=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHMM with TSCPG distribution

obj_ihmm2 <- ihmm.scp(x = NULL, imc.par, scpg.par)
class(obj_ihmm2)
# Simulating a 2-state IHMM with a TSCPG distribution

sim_ihmm2 <- sim.ihmm.scp(obj_ihmm2, nsim = 200, seed = 2)
str(sim_ihmm2[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the

```

```

# transitions into State 2 become less likely
table(state = sim_ihmm2$states,
      half = ifelse(seq_along(sim_ihmm2$states) <= 100, "first", "second"))
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state IHMM with a TSCPG distribution
# (the parameters of obj_ihmm2 are the starting values)

fit_ihmm2 <- fit.ihmm.scp(sim_ihmm2)
fit_ihmm2$imc.par$ttpm[, 1, ] # r_i2 (first column) and w_i2 (second)
fit_ihmm2$scpg.par$rate[, 1] # K, M, R, P of lambda_2(t)
fit_ihmm2$scpg.par$scale      # true 1
fit_ihmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# The Viterbi path of the fitted 2-state IHMM with TSCPG distribution

fit_ihmm2$x <- sim_ihmm2$x
vip_ihmm2 <- vip.ihmm.scp(fit_ihmm2)
table(true = sim_ihmm2$states, decoded = vip_ihmm2)

# Plotting of the Viterbi path with observed data

dev.new()
oldpar <- par(mfrow=c(2, 1), cex.lab = 1.3, cex.axis = 1.2, mgp=c(2.3,0.8,0))
par(mar=c(0, 4.0, 2, 3)) # bottom margin is 0.
plot(cumsum(sim_ihmm2$x), sim_ihmm2$x, type="l", ylab = "Interevent times",
     xaxt="n")
mtext("Simulated data for IHMM with TSCPG", side = 4, line = 0.8, cex = 1.1,
     las = 0)
par(mar=c(3.5, 4.0, 0, 3)) # top is 0.
plot(cumsum(sim_ihmm2$x), vip_ihmm2, type="s", yaxt="n",
     xlab="Occurrence Time", ylab="States")
axis(2, at=1:2, labels=1:2)
mtext("Viterbi path for IHMM with TSCPG", side = 4, line = 0.8, cex = 1.1,
     las = 0)
par(oldpar)

# =====
# An example of a 3-state IHMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 2.5 minutes on a laptop (the
# 2-state model above takes about 8 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along the
# record (see the Details of ?fit.ihmm.scp):
# - the transition probabilities  $a_{ij}(t)$ : moves into the incomplete
#   States 2 and 3 become less likely as time goes on. The log-odds of
#   moving from state  $i$  to state  $j$  rather than to State 1 is
#    $w_{ij} - r_{ij} * h_{\{t-1\}}$ , which is zero at the crossing time  $w_{ij} / r_{ij}$ ;

```

```

# - the Poisson means  $\lambda_j(t)$  of the TSCPG: in the incomplete states
#   fewer events are missed after the inflection times  $P_j / R_j$ .
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1, so that the states are easier to tell apart.

m <- 3
initd <- c(0.4, 0.3, 0.3)
ttpm <- array(NA, dim = c(m, m-1, 2))
# rates of change  $r_{ij}$  (rows  $i = 1, 2, 3$ ; columns  $j = 2, 3$ )
ttpm[, , 1] <- matrix(c(0.006, 0.006, 0.006,
                        0.006, 0.006, 0.006), nrow = m, ncol = m-1)
# intercepts  $w_{ij}$ ; crossing times  $w_{ij}/r_{ij}$  between 400 and 600
ttpm[, , 2] <- ttpm[, , 1] * matrix(c(500, 600, 500,
                                       400, 500, 600), nrow = m, ncol = m-1)
imc.par <- list(initd=initd, ttpm=ttpm, stationary=FALSE)
rate <- matrix(NA, nrow = 4, ncol = m-1)
rate[1, ] <- c(0.5, 4)      # K:  $\lambda_2(t)$  falls from about 3.4 towards
rate[2, ] <- c(3, 6)       # M: 0.5,  $\lambda_3(t)$  from about 10 towards 4
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(3, 4.2)     # P: inflection times  $P/R = 500$  and  $700$ 
shape <- 2
scale <- 0.5
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=TRUE, time.scpg=TRUE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHMM with TSCPG distribution

obj_ihmm2_m3 <- ihmm.scpg(x = NULL, imc.par, scpg.par)
class(obj_ihmm2_m3)
# Simulating a 3-state IHMM with a TSCPG distribution

sim_ihmm2_m3 <- sim.ihmm.scpg(obj_ihmm2_m3, nsim = 500, seed = 1)
str(sim_ihmm2_m3[c("x", "states")])

# Observations in each state: State 1 becomes more frequent as the
# transitions into the incomplete states become less likely
table(state = sim_ihmm2_m3$states,
      half = ifelse(seq_along(sim_ihmm2_m3$states) <= 250, "first", "second"))
# Fitting a 3-state IHMM with a TSCPG distribution
# (the parameters of obj_ihmm2_m3 are the starting values). The fit
# takes about 2.5 minutes, so it is wrapped in donttest.

fit_ihmm2_m3 <- fit.ihmm.scpg(sim_ihmm2_m3)
fit_ihmm2_m3$imc.par$ttpm[, , 1] #  $r_{ij}$  (rows  $i$ , columns  $j = 2, 3$ )
fit_ihmm2_m3$imc.par$ttpm[, , 2] #  $w_{ij}$ 
fit_ihmm2_m3$scpg.par$rate      # K, M, R, P of  $\lambda_2(t)$  and  $\lambda_3(t)$ 
fit_ihmm2_m3$logLik

# The Viterbi path of the fitted 3-state IHMM with TSCPG distribution

fit_ihmm2_m3$x <- sim_ihmm2_m3$x
vip_ihmm2_m3 <- vip.ihmm.scpg(fit_ihmm2_m3)

```

```

table(true = sim_ihmm2_m3$states, decoded = vip_ihmm2_m3)

# Plotting of the Viterbi path with observed data

dev.new()
oldpar <- par(mfrow=c(2, 1), cex.lab = 1.3, cex.axis = 1.2, mgp=c(2.3,0.8,0))
par(mar=c(0, 4.0, 2, 3)) # bottom margin is 0.
plot(cumsum(sim_ihmm2_m3$x), sim_ihmm2_m3$x, type="l",
     ylab = "Interevent times" , xaxt="n")
mtext("Simulated data for IHMM with TSCPG", side = 4, line = 0.8, cex = 1.1,
     las = 0)
par(mar=c(3.5, 4.0, 0, 3)) # top is 0.
plot(cumsum(sim_ihmm2_m3$x), vip_ihmm2_m3, type="s", yaxt="n",
     xlab="Occurrence Time",
     ylab="States")
axis(2, at=1:3, labels=1:3)
mtext("Viterbi path for IHMM with TSCPG", side = 4, line = 0.8, cex = 1.1,
     las = 0)
par(oldpar)

```

vip.ihsmm.scpG	<i>Viterbi path for an m-state IHSM with gamma or SCPG or TSCPG renewal processes</i>
----------------	---

Description

Calculates the Viterbi path for an IHSM with class `ihsmm.gamma`, `ihsmm.scpG` and `ihsmm.tscpg` as typically returned by [ihsmm.scpG](#).

Usage

```
vip.ihsmm.scpG(ihsmmobj)
```

Arguments

`ihsmmobj` is a model object with class `ihsmm.gamma`, `ihsmm.scpG` and `ihsmm.tscpg` as typically returned by [ihsmm.scpG](#).

Details

Global decoding of an IHSM maximizes over sequences of visits and their durations jointly, following the algorithm of Shahzadi et al. (2023), which extends that of Guedon (2003) and Yu (2010) to time-dependent sojourn distributions. With $\delta_t(j, d)$ the maximum probability of a path in which a visit to State j of length d ends at time t ,

$$\delta_t(j, d) = \max_{i \neq j} \max_h \{ \delta_{t-d}(i, h) a_{ij} \} p_j(d, t-d+1) \prod_{l=t-d+1}^t f_j(x_l),$$

for $2 \leq t \leq T - 1$, where $p_j(d, t)$ is the shifted Poisson sojourn mass function with parameter $\mu_j(t)$, and with initial condition $\delta_t(j, d) = \pi_j p_j(d, 1) \prod_{l=1}^t f_j(x_l)$ for $t = d$. At $t = T$ the sojourn in the last visited state is right-censored and $p_j(d, T - d + 1)$ is replaced by the corresponding survival function, so that the decoded path is optimal under the same right-censored likelihood as is maximized by `fit.ihsmm.scp`. The maximizing predecessor and its duration are recorded in $\psi(t, j, d) = (t - d, i^*, h^*)$ and the path recovered by backtracking from $\max_{j,d} \delta_T(j, d)$. The recursion is evaluated on the logarithmic scale.

Each row of the result gives the index of the last observation of a visit (column "t-d"), the state occupied ("j"), and its duration ("d"); the rows run backwards from the end of the record, and the visit of a row covers observations $t-d-d+1$ to $t-d$. Since the expected sojourn time in State 1 increases with elapsed time while those in the incompleteness states decrease, the decoded path of a correctly specified model exhibits visits to State 1 of increasing length towards the end of the record, the incompleteness states being confined to its earlier portion. The occurrence time from which State 1 is no longer interrupted estimates the point at which the record becomes complete, and is the principal quantity reported in the applications of Shahzadi et al. (2023).

Value

returns the most probable state sequence responsible for the observed sequence using the Viterbi algorithm with a fitted IHSM with class `ihsmm.scp`.

References

- Shahzadi A, Wang T, Parry M, Bebbington M (2023). "Inhomogeneous hidden semi-Markov models for incompletely observed point processes." *Annals of Institute Statistical Mathematics*, **75**, 253-280. doi:10.1007/s10463022008435.
- Guédon Y (2003). "Estimating hidden semi-Markov chains from discrete sequences." *Journal of Computational and Graphical Statistics*, **12**(3), 604-639. doi:10.1198/1061860032030.
- Yu SZ (2010). "Hidden semi-Markov models." *Artificial Intelligence*, **174**, 215-243. doi:10.1016/j.artint.2009.11.011.

Examples

```
# =====
# An example of a 2-state IHSM with an SCPG renewal process
# =====

# The sojourn parameters mu_j(t) depend on the time at which a visit
# commences (rows A, B, C, D of tdur): the expected sojourn 1 + mu_1(t) in
# the complete State 1 increases, and that in the incomplete State 2
# decreases, between the levels 1 + A_j = 1.5 and 1 + A_j + B_j = 5.5
# observations, around the inflection time D_j/C_j = 500. In State 2,
# the number N of events missing between two observed events is shifted
# Poisson with parameter lambda_2.
# With m = 2 the chain alternates between the two states.

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
```

```

tdur <- matrix(c(
  0.5, 0.5,      # A
  4, 4,         # B
  0.005, 0.005, # C
  2.5, 2.5      # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(3, nrow = 1, ncol = m-1) # lambda_2
shape <- 1.5
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 2-state IHSMM with SCPG distribution

obj_ihsmm1 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm1)
# Simulating a 2-state IHSMM with SCPG distribution

sim_ihsmm1 <- sim.ihsmm.scp(obj_ihsmm1, nsim = 150, seed = 3)
str(sim_ihsmm1[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm1$states)$values)
# The fit takes about 30 s, so the code below is wrapped in donttest.

# Fitting a 2-state IHSMM with SCPG distributions
# (the parameters of obj_ihsmm1 are the starting values)

fit_ihsmm1 <- fit.ihsmm.scp(sim_ihsmm1)
fit_ihsmm1$ismc.par$tdur      # A, B, C, D of mu_1(t) and mu_2(t)
fit_ihsmm1$scpg.par$rate      # lambda_2 (true 3)
fit_ihsmm1$scpg.par$shape     # true 1.5
fit_ihsmm1$scpg.par$scale     # true 1
fit_ihsmm1$logLik

# The fit takes about 30 s, so the code below is wrapped in donttest.

# The Viterbi path of the fitted 2-state IHSMM with SCPG distribution.
# Each row describes one decoded visit: the index of its last observation,
# its state and its duration (rows run from the end of the record).

fit_ihsmm1$x <- sim_ihsmm1$x
vip_ihsmm1 <- vip.ihsmm.scp(fit_ihsmm1)
head(vip_ihsmm1)

# Expanding the visits into one state per observation and comparing the
# decoded states with the simulated ones
path_ihsmm1 <- integer(length(sim_ihsmm1$x))
for (r in seq_len(nrow(vip_ihsmm1))) {

```

```

    last <- vip_ihsmm1[r, 1]
    path_ihsmm1[(last - vip_ihsmm1[r, 3] + 1):last] <- vip_ihsmm1[r, 2]
  }
  table(true = sim_ihsmm1$states, decoded = path_ihsmm1)

# =====
# An example of a 3-state IHSMM with an SCPG renewal process
# =====

# Fitting this 3-state model takes about 2 minutes on a laptop (the
# 2-state model above takes about 30 seconds), so the examples that fit
# it are wrapped in donttest.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.5, 0.5, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.3, 0.3, 0.3,      # A
  1.5, 2, 1.5,        # B
  0.005, 0.005, 0.005, # C
  3, 3, 3            # D: inflection times D/C = 600
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd=NULL, tpm=tpm, tdur=tdur,
                 tdur.lb=NA, tdur.ub=NA, stationary=TRUE)
rate <- matrix(c(8, 5), nrow = 1, ncol = m-1)
shape <- c(2, 1, 1.7)
scale <- 1
scpg.par <- list(rate=rate, shape=shape, scale=scale,
                 rate.fix=FALSE, shape.fix=FALSE, time.scp=FALSE,
                 WBSum.method='direct.evaluation')

# Defining an object of 3-state IHSMM with SCPG distribution

obj_ihsmm1_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
# Simulating a 3-state IHSMM with SCPG distribution

sim_ihsmm1_m3 <- sim.ihsmm.scp(obj_ihsmm1_m3, nsim = 150, seed = 1)
print(sim_ihsmm1_m3)
# Fitting a 3-state IHSMM with SCPG distributions
# The fit takes about 2 minutes, so it is wrapped in donttest.

fit_ihsmm1_m3 <- fit.ihsmm.scp(sim_ihsmm1_m3)
print(fit_ihsmm1_m3)

# The Viterbi path of fitted 3-state IHSMM with SCPG distribution

fit_ihsmm1_m3$x <- sim_ihsmm1_m3$x
vip_ihsmm1_m3 <- vip.ihsmm.scp(fit_ihsmm1_m3)
print(vip_ihsmm1_m3)

# =====
# An example of a 2-state IHSMM with a TSCPG renewal process

```

```
# =====

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of tdur): the expected
#   sojourn 1 +  $\mu_1(t)$  in the complete State 1 increases, and that in the
#   incomplete State 2 decreases, between the levels 1 +  $A_j = 2$  and
#   1 +  $A_j + B_j = 8$  observations, around the inflection time
#    $D_j/C_j = 500$ ;
# - the Poisson mean  $\lambda_2(t)$  of the TSCPG (rows K, M, R, P of rate)
#   falls from about 5.8 at the start of the record towards  $K = 0.5$ ,
#   around the inflection time  $P/R = 400$ .

m <- 2
tpm <- matrix(c(0, 1,
                1, 0), nrow = 2, byrow = TRUE)
tdur <- matrix(c(
  1, 1,      # A
  6, 6,      # B
  0.005, 0.005, # C
  2.5, 2.5    # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- 0.5    # K
rate[2, ] <- 6      # M
rate[3, ] <- 0.005  # R
rate[4, ] <- 2      # P: inflection time P/R = 400
shape <- 1
scale <- 1
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scpg = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 2-state IHSMM with TSCPG distribution

obj_ihsmm2 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2)
# Simulating a 2-state IHSMM with a TSCPG distribution

sim_ihsmm2 <- sim.ihsmm.scp(obj_ihsmm2, nsim = 200, seed = 1)
str(sim_ihsmm2[c("x", "states")])

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2$states)$values)
# The fit takes several seconds, so the code below is wrapped in donttest.

# Fitting a 2-state IHSMM with a TSCPG distribution
# (the parameters of obj_ihsmm2 are the starting values)

fit_ihsmm2 <- fit.ihsmm.scp(sim_ihsmm2)
```

```

fit_ihsmm2$ismc.par$tdur      # A, B, C, D of  $\mu_1(t)$  and  $\mu_2(t)$ 
fit_ihsmm2$scpg.par$rate     # K, M, R, P of  $\lambda_2(t)$ 
fit_ihsmm2$scpg.par$scale    # true 1
fit_ihsmm2$logLik

# The fit takes several seconds, so the code below is wrapped in donttest.

# The Viterbi path of the fitted 2-state IHSMM with TSCPG distribution.
# Each row describes one decoded visit: the index of its last observation,
# its state and its duration (rows run from the end of the record).

fit_ihsmm2$x <- sim_ihsmm2$x
vip_ihsmm2 <- vip.ihsmm.scp(fit_ihsmm2)
head(vip_ihsmm2)

# Expanding the visits into one state per observation and comparing the
# decoded states with the simulated ones
path_ihsmm2 <- integer(length(sim_ihsmm2$x))
for (r in seq_len(nrow(vip_ihsmm2))) {
  last <- vip_ihsmm2[r, 1]
  path_ihsmm2[(last - vip_ihsmm2[r, 3] + 1):last] <- vip_ihsmm2[r, 2]
}
table(true = sim_ihsmm2$states, decoded = path_ihsmm2)

# =====
# An example of a 3-state IHSMM with a TSCPG renewal process
# =====

# Fitting this 3-state model takes about 8 minutes on a laptop (the
# 2-state model above takes about 50 seconds), so the examples that fit
# it are wrapped in donttest.

# Two kinds of time dependence describe how completeness improves along
# the record (see the Details of ?fit.ihsmm.scp):
# - the sojourn parameters  $\mu_j(t)$  (rows A, B, C, D of  $\mu$ ), with
#   inflection times  $D_j/C_j = 500$ : the expected sojourn  $1 + \mu_1(t)$  in the
#   complete State 1 increases from about 2.1 towards 6.5 observations,
#   and those in the incomplete States 2 and 3 decrease from about 5.9
#   towards 1.5;
# - the Poisson means  $\lambda_j(t)$  of the TSCPG (rows K, M, R, P of  $\lambda$ ):
#    $\lambda_2(t)$  falls from about 3.3 towards 0.5 around 400, and
#    $\lambda_3(t)$  from about 9.8 towards 4 around 600.
# The zero-diagonal transition matrix gives the next state of each visit.
# The shape 2 makes the interevent times of the three states less
# overlapping than with shape 1.

m <- 3
tpm <- matrix(c(0, 0.6, 0.4,
                0.5, 0, 0.5,
                0.6, 0.4, 0), nrow = 3, byrow = TRUE)
tdur <- matrix(c(
  0.5, 0.5, 0.5,      # A
  5, 5, 5,           # B

```

```

0.004, 0.004, 0.004, # C
2, 2, 2 # D: inflection times D/C = 500
), nrow = 4, byrow = TRUE)
ismc.par <- list(initd = NULL, tpm = tpm, tdur = tdur,
                 tdur.lb = NA, tdur.ub = NA, stationary = TRUE)
rate <- matrix(NA, nrow = 4, ncol = m - 1)
rate[1, ] <- c(0.5, 4) # K
rate[2, ] <- c(3, 6) # M
rate[3, ] <- c(0.006, 0.006) # R
rate[4, ] <- c(2.4, 3.6) # P: inflection times P/R = 400 and 600
shape <- 2
scale <- 0.5
scpg.par <- list(rate = rate, shape = shape, scale = scale,
                 rate.fix = FALSE, shape.fix = TRUE, time.scp = TRUE,
                 WBSum.method = 'direct.evaluation')

# Defining an object of 3-state IHSMM with TSCPG distribution

obj_ihsmm2_m3 <- ihsmm.scp(x = NULL, ismc.par, scpg.par)
class(obj_ihsmm2_m3)
# Simulating a 3-state IHSMM with a TSCPG distribution

sim_ihsmm2_m3 <- sim.ihsmm.scp(obj_ihsmm2_m3, nsim = 250, seed = 2)
str(sim_ihsmm2_m3[c("x", "states")])

# The record covers [0, h_T]; the time-dependent components are evaluated
# on [0, h_{T-1}]
c(h_T1 = sum(sim_ihsmm2_m3$x[-250]), h_T = sum(sim_ihsmm2_m3$x))

# The duration parameters are informed by the visits, one sojourn each:
# number of visits to each state in the simulated path
table(rle(sim_ihsmm2_m3$states)$values)
# Fitting a 3-state IHSMM with a TSCPG distribution
# (the parameters of obj_ihsmm2_m3 are the starting values). The fit
# takes about 8 minutes, so it is wrapped in donttest.

fit_ihsmm2_m3 <- fit.ihsmm.scp(sim_ihsmm2_m3)
fit_ihsmm2_m3$ismc.par$tpm # transition matrix of the embedded chain
fit_ihsmm2_m3$ismc.par$tdur # A, B, C, D of mu_1(t), mu_2(t) and mu_3(t)
fit_ihsmm2_m3$scpg.par$rate # K, M, R, P of lambda_2(t) and lambda_3(t)
fit_ihsmm2_m3$scpg.par$scale # true 0.5
fit_ihsmm2_m3$logLik

# The Viterbi path of the fitted 3-state IHSMM with TSCPG distribution.
# Each row describes one decoded visit: the index of its last observation,
# its state and its duration (rows run from the end of the record).

fit_ihsmm2_m3$x <- sim_ihsmm2_m3$x
vip_ihsmm2_m3 <- vip.ihsmm.scp(fit_ihsmm2_m3)
head(vip_ihsmm2_m3)

# Expanding the visits into one state per observation and comparing the
# decoded states with the simulated ones

```

```

path_ihsmm2_m3 <- integer(length(sim_ihsmm2_m3$x))
for (r in seq_len(nrow(vip_ihsmm2_m3))) {
  last <- vip_ihsmm2_m3[r, 1]
  path_ihsmm2_m3[(last - vip_ihsmm2_m3[r, 3] + 1):last] <- vip_ihsmm2_m3[r, 2]
}
table(true = sim_ihsmm2_m3$states, decoded = path_ihsmm2_m3)

```

WBsum

*Wright's generalized Bessel function of the SCPG density***Description**

Evaluates the series

$$W(x; \alpha) = \sum_{k=0}^{\infty} \frac{x^k}{k! \Gamma\{\alpha(k+2)\}},$$

which appears in the SCPG and TSCPG densities of [dscpg](#).

Usage

```

WBsum(
  x,
  alpha,
  WBsum.method = c("direct.evaluation", "direct.asymptotic",
                    "asymptotic.expansion"),
  tol = 1e-12,
  max.direct.terms = 2000L,
  log = FALSE
)

```

Arguments

x	A vector of finite non-negative numbers (the argument z in Details).
alpha	A single finite positive number, the gamma shape α .
WBsum.method	The method, one of • "direct.evaluation" (the default), • "direct.asymptotic" or • "asymptotic.expansion"; see Details.
tol	Relative tolerance. For the direct sum it is a rigorous bound on the omitted terms; for the asymptotic expansion it is compared with an empirical error estimate. Values below <code>.Machine\$double.eps</code> are treated as machine epsilon. Default is 10^{-12} .

`max.direct.terms`

Used only by `WBsum.method = "direct.asymptotic"`: the number of direct-series terms above which the asymptotic expansion is tried; see Details. A positive integer; default is 2000.

`log`

Logical; if TRUE, return $\log W$.

Details

The series. $W(z; \alpha) = \phi(\alpha, 2\alpha; z)$ is a special case of Wright's generalized Bessel function (Wright, 1935; Paris, 2017). In the SCPG density $z = \lambda(x_t/\beta)^\alpha$. For $z \geq 0$ every term is positive, so there is no cancellation, and there is a single largest term, near

$$n^* = (z/\alpha^\alpha)^{1/(1+\alpha)}$$

(at $k = 0$ when z is small), and fall away on both sides of it. The terms, divided by their sum, are the probabilities of the possible numbers of missing events (minus one) given the observation, so n^* is about the most probable number of missing events. The number of terms that contribute grows only like $\sqrt{n^*}$ (about $10\sqrt{n^*}$ terms exceed 10^{-12} times the largest one when $\alpha = 1$).

The three methods.

"`direct.evaluation`" sums the series outwards in both directions from its largest term, as Dunn and Smyth (2005) do for the Tweedie densities. The terms decrease geometrically on each side, so the omitted terms are bounded, and summation stops when both bounds are below `tol` times the sum. The result is accurate to `tol` apart from rounding, which limits the accuracy of $\log W$ to a small multiple of $\epsilon|\log W|$, with ϵ the machine epsilon (at most about $20\epsilon|\log W|$ for n^* up to 10^{12}). The time per value is about 0.02 s at $n^* = 10^8$, 0.2 s at 10^{10} and about 2 s at 10^{12} . If any value has $n^* > 10^{12}$, the call is refused with an error before any sum is computed.

"`asymptotic.expansion`" uses the large-argument expansion of Paris (2010, 2016) in $X = (1 + \alpha)n^*$, with ten coefficients and, for $\alpha > 1$, the $\lfloor(\alpha - 1)/4\rfloor + 1$ pairs of exponentially small terms; at $\alpha = 1$ the exact identity $W(z; 1) = z^{-1/2}I_1(2\sqrt{z})$ is used instead. It is tried only when $X \geq 5$ and accepted only when its error estimate is below `tol`; otherwise it stops with an error. With the default `tol` it is accepted from about $z = 72$ ($\alpha = 0.5$), 117 (0.75), 810 (1.5), 6.2×10^3 (2), 4.4×10^5 (3) and 1.8×10^{11} (5), that is, from X between about 29 and 117. Its error estimate is empirical, not a rigorous bound: just above these points the observed error was up to about 1.3 times `tol`.

"`direct.asymptotic`" chooses between the two for each value. It predicts the number of direct-series terms as $2\lceil 9\sqrt{(n^* + 1)/(1 + \alpha)} \rceil + 21$, a window of about nine standard deviations on each side of the largest term. If this is at most `max.direct.terms`, the direct sum is used; otherwise the asymptotic expansion is tried, and if it is not accepted the direct sum is used after all. With the default `max.direct.terms = 2000` the expansion is therefore used when $n^* > 1.21 \times 10^4(1 + \alpha)$, far above the points where it becomes accurate, and the time per value stays below about 0.01 s. The values agree with those of the direct sum to within `tol`, or within the rounding error of the direct sum (a small multiple of $\epsilon|\log W|$) if that is larger.

The script `verify_WBsum.R` in the `examples` directory of the package reproduces these numbers; run it in the console with

```
f <- system.file("examples", "verify_WBsum.R",
```

```

      package = "HMMscpg")
source(f, echo = TRUE)

```

When `dscpg` needs "direct.asymptotic" is explained there. For every method $z = 0$ is evaluated exactly, and `log = TRUE` avoids overflow of W itself; `dscpg` always works with $\log W$.

Value

The values of $W(z; \alpha)$, or of $\log W(z; \alpha)$ when `log = TRUE`.

References

- Wright EM (1935). "The asymptotic expansion of the generalized hypergeometric function." *Journal of the London Mathematical Society*, **10**(4), 286–293.
- Paris RB (2017). "The asymptotics of the generalized Bessel function." *Mathematica Aeterna*, **7**(4), 381–406. doi:10.48550/arXiv.1711.03006.
- Dunn PK, Smyth GK (2005). "Series evaluation of Tweedie exponential dispersion model densities." *Stat Comput*, **15**(4), 267–280. doi:10.1007/s112220054070y.
- Paris RB (2010). "Exponentially small expansions in the asymptotics of the Wright function." *Journal of Computational and Applied Mathematics*, **234**(2), 488–504.
- Paris RB, Vinogradov V (2016). "Asymptotic and structural properties of special cases of the Wright function arising in probability theory." *Lithuanian Mathematical Journal*, **56**(3), 377–409.

Index

dscpg, [5](#), [5](#), [9](#), [37](#), [38](#), [42](#), [46](#), [50](#), [55](#), [60](#), [62](#),
[63](#), [65](#), [136](#), [138](#)

fit.hmm.scp, [8](#), [21](#), [22](#), [30](#), [42](#), [67](#), [82](#), [113](#)
fit.hsmm.scp, [13](#), [29](#), [30](#), [47](#), [70](#), [87](#), [117](#)
fit.ihmm.scp, [15](#), [20](#), [30](#), [31](#), [51](#), [95](#)
fit.ihsmm.scp, [28](#), [56](#), [78](#), [130](#)
fit.scp, [5–7](#), [9](#), [10](#), [14](#), [15](#), [21](#), [29](#), [37](#), [38](#), [65](#)

hmm.scp, [8–10](#), [41](#), [46](#), [47](#), [51](#), [67](#), [82](#), [113](#)
HMMscpg (HMMscpg-package), [3](#)
HMMscpg-package, [3](#)
hsmm.scp, [13](#), [45](#), [56](#), [69](#), [86](#), [87](#), [116](#), [117](#)

ihmm.scp, [20](#), [21](#), [49](#), [73](#), [95](#), [96](#), [122](#), [123](#)
ihsmm.scp, [28](#), [54](#), [77](#), [103](#), [104](#), [129](#), [130](#)

list, [47](#), [51](#), [56](#)

optim, [9](#), [10](#), [14](#), [15](#), [20–22](#), [28](#), [29](#), [31](#), [38](#), [39](#)

pscpg, [5](#), [60](#), [63](#)

qscpg, [61](#), [62](#)

rscpg, [65](#), [67](#)

sim.hmm.scp, [42](#), [67](#), [73](#)
sim.hsmm.scp, [69](#), [78](#)
sim.ihmm.scp, [51](#), [73](#)
sim.ihsmm.scp, [31](#), [77](#)
sp.hmm.scp, [10](#), [42](#), [82](#), [87](#), [95](#), [113](#)
sp.hsmm.scp, [14](#), [86](#)
sp.ihmm.scp, [51](#), [95](#)
sp.ihsmm.scp, [29](#), [103](#)

uniroot, [63](#)

vip.hmm.scp, [10](#), [42](#), [82](#), [112](#)
vip.hsmm.scp, [14](#), [87](#), [116](#)
vip.ihmm.scp, [51](#), [95](#), [122](#)
vip.ihsmm.scp, [29](#), [104](#), [129](#)

WBsum, [5–7](#), [38](#), [42](#), [46](#), [50](#), [55](#), [136](#)