Package: smmR (via r-universe)

October 7, 2026

Type Package

Title Simulation, Estimation and Reliability of Semi-Markov Models

Version 1.0.5

Date 2025-10-20

Description Performs parametric and non-parametric estimation and simulation for multi-state discrete-time semi-Markov processes. For the parametric estimation, several discrete distributions are considered for the sojourn times: Uniform, Geometric, Poisson, Discrete Weibull and Negative Binomial. The non-parametric estimation concerns the sojourn time distributions, where no assumptions are done on the shape of distributions. Moreover, the estimation can be done on the basis of one or several sample paths, with or without censoring at the beginning or/and at the end of the sample paths. Reliability indicators such as reliability, maintainability, availability, BMP-failure rate, RG-failure rate, mean time to failure and mean time to repair are available as well. The implemented methods are described in Barbu, V.S., Limnios, N. (2008) <doi:10.1007/978-0-387-73173-5>, Barbu, V.S., Limnios, N. (2008) <doi:10.1080/10485250701261913> and Trevezas, S., Limnios, N. (2011) <doi:10.1080/10485252.2011.555543>. Estimation and simulation of discrete-time k-th order Markov chains are also considered.

URL https://plmlab.math.cnrs.fr/lmrs/statistique/smmR, https://lmrs.pages.math.cnrs.fr/statistique/smmR

BugReports https://github.com/corentin-dev/smmR/issues

License GPL-3

Imports DiscreteWeibull, Rcpp, seqinr

Suggests utils, knitr, rmarkdown, testthat (>= 3.0.0), roxytest

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Vlad Stefan Barbu [aut] (ORCID: <https://orcid.org/0000-0002-0840-016X>), Florian Lecocq [aut], Corentin Lothode [aut] (ORCID: <https://orcid.org/0000-0002-8209-317X>), Nicolas Vergne [aut, cre]

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

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

Date/Publication 2025-11-07 11:40:02 UTC

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

RemoteRef HEAD

RemoteSha 8fdf003b7162b9d7381c4032a18fbad584a5e93a

Contents

smmR-package . . . 3
.failureRateBMP . . . 5
availability . . . 6
convolution . . . 8
failureRate . . . 8
fitmm . . . 10
fitsmm . . . 12
get.f . . . 16
get.H . . . 17
get.Kpar . . . 18
get.limitDistribution . . . 18
get.Niuj . . . 19
get.P . . . 19
get.psi . . . 20
get.Py . . . 20
get.qy . . . 21
get.stationaryDistribution . . . 21
getKernel . . . 22
getProcesses . . . 22
is.mm . . . 23
is.mmfit . . . 24
is.smm . . . 24
is.smmfit . . . 25
is.smmnonparametric . . . 25
is.smmparametric . . . 26
maintainability . . . 26
matrixConvolution . . . 27
meanRecurrenceTimes . . . 28

meanSojournTimes 29
mm 30
mttf 31
mttr 32
plot.smm 33
plot.smmfit 34
reliability 35
setSeed 37
simulate.mm 37
simulate.mmfit 38
simulate.smm 39
simulate.smmfit 40
smmnonparametric 41
smmparametric 43

Index **47**

`smmR-package` *smmR : Semi-Markov Models, Markov Models and Reliability*

Description

This package performs parametric and non-parametric estimation and simulation for multi-state discrete-time semi-Markov processes. For the parametric estimation, several discrete distributions are considered for the sojourn times: Uniform, Geometric, Poisson, Discrete Weibull and Negative Binomial. The non-parametric estimation concerns the sojourn time distributions, where no assumptions are done on the shape of distributions. Moreover, the estimation can be done on the basis of one or several sample paths, with or without censoring at the beginning or/and at the end of the sample paths. Estimation and simulation of discrete-time k-th order Markov chains are also considered.

Semi-Markov models are specified by using the functions `smmparametric()` and `smmnonparametric()` for parametric and non-parametric specifications respectively. These functions return objects of S3 class (`smm`, `smmparametric`) and (`smm`, `smmnonparametric`) respectively (`smm` class inherits from S3 classes `smmparametric` or `smmnonparametric`). Thus, `smm` is like a wrapper class for semi-Markov model specifications.

Based on a model specification (an object of class `smm`), it is possible to:

- **simulate** one or several sequences with the method `simulate.smm()`;
- **plot** conditional sojourn time distributions (method `plot.smm()`);
- compute **log-likelihood**, **AIC** and **BIC** criteria (methods `logLik()`, `AIC()`, `BIC()`);
- compute **reliability**, **maintainability**, **availability**, **failure rates** (methods `reliability()`, `maintainability()`, `availability()`, `failureRate()`).

Estimations of parametric and non-parametric semi-Markov models can be done by using the function `fitsmm()`. This function returns an object of S3 class `smmfit`. The class `smmfit` inherits from classes (`smm`, `smmparametric`) or (`smm`, `smmnonparametric`).

Based on a fitted/estimated semi-Markov model (an object of class `smmfit`), it is possible to:

- **simulate** one or several sequences with the method `simulate.smmfit()`;
- **plot** estimated conditional sojourn time distributions (method `plot.smmfit()`);
- compute **log-likelihood**, **AIC** and **BIC** criteria (methods `logLik()`, `AIC()`, `BIC()`);
- compute estimated **reliability**, **maintainability**, **availability**, **failure rates** and their **confidence intervals** (methods `reliability()`, `maintainability()`, `availability()`, `failureRate()`).

Author(s)

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

Authors:

- Vlad Stefan Barbu (ORCID)
- Florian Lecocq
- Corentin Lothode (ORCID)

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

R.E. Barlow, A.W. Marshall, and F. Prochan. (1963). Properties of probability distributions with monotone hazard rate. Ann. Math. Statist., 34, 375-389.

T. Nakagawa and S. Osaki. (1975). The discrete Weibull distribution. IEEE Trans. Reliabil., R-24, 300-301.

D. Roy and R. Gupta. (1992). Classification of discrete lives. Microelectron. Reliabil., 32 (10), 1459-1473.

I. Votsi & A. Brouste (2019) Confidence interval for the mean time to failure in semi-Markov models: an application to wind energy production, Journal of Applied Statistics, 46:10, 1756-1773.

See Also

Useful links:

- https://plmlab.math.cnrs.fr/lmrs/statistique/smmR
- https://lmrs.pages.math.cnrs.fr/statistique/smmR
- Report bugs at https://github.com/corentin-dev/smmR/issues

`.failureRateBMP` *BMP-Failure Rate Function*

Description

Consider a system S_{ystem} starting to work at time $k = 0$. The BMP-failure rate at time $k \in N$ is the conditional probability that the failure of the system occurs at time k, given that the system has worked until time $k - 1$.

Usage

```
.failureRateBMP(
  x,
  k,
  upstates = x$states,
  level = 0.95,
  epsilon = 0.001,
  klim = 10000
)
```

Arguments

`x`	An object of S3 class `smmfit` or `smm`.
`k`	A positive integer giving the period $[0, k]$ on which the BMP-failure rate should be computed.
`upstates`	Vector giving the subset of operational states U.
`level`	Confidence level of the asymptotic confidence interval. Helpful for an object `x` of class `smmfit`.
`epsilon`	Value of the reliability above which the latter is supposed to be 0 because of computation errors (see Details).
`klim`	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the failure rate of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The system starts to work at instant 0 and the state of the system

is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

The BMP-failure rate at time $k \in N$ is the conditional probability that the failure of the system occurs at time k, given that the system has worked until time $k-1$.

Let T_D denote the first passage time in subset D, called the lifetime of the system, i.e.,

$$T_D := \inf\{n \in N;\ Z_n \in D\} \text{ and } \inf \emptyset := \infty.$$

For a discrete-time semi-Markov system, the failure rate at time $k \geq 1$ has the expression:

$$\lambda(k) := P(T_D = k | T_D \geq k)$$

We can rewrite it as follows :

$$\lambda(k) = 1 - \frac{R(k)}{R(k-1)},\ \text{if } R(k-1) \neq 0;\ \lambda(k) = 0, \text{otherwise}$$

The failure rate at time $k = 0$ is defined by $\lambda(0) := 1 - R(0)$, with R being the reliability function (see reliability function).

The calculation of the reliability R involves the computation of many convolutions. It implies that the computation error, may be higher (in value) than the "true" reliability itself for reliability close to 0. In order to avoid inconsistent values of the BMP-failure rate, we use the following formula:

$$\lambda(k) = 1 - \frac{R(k)}{R(k-1)},\ \text{if } R(k-1) \geq \epsilon;\ \lambda(k) = 0, \text{otherwise}$$

with ϵ, the threshold, the parameter `epsilon` in the function `failureRateBMP`.

Value

A matrix with $k+1$ rows, and with columns giving values of the BMP-failure rate, variances, lower and upper asymptotic confidence limits (if `x` is an object of class `smmfit`).

`availability` *Availability Function*

Description

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

Usage

```
availability(x, k, upstates = x$states, level = 0.95, klim = 10000)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the time at which the availability should be computed.
upstates	Vector giving the subset of operational states U.
level	Confidence level of the asymptotic confidence interval. Helpful for an object x of class smmfit.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the availability of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

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

Thus, the pointwise availability of a semi-Markov system at time $k \in N$ is

$$A(k) = P(Z_k \in U) = \sum_{i \in E} \alpha_i A_i(k),$$

where we have denoted by $A_i(k)$ the conditional availability of the system at time $k \in N$, given that it starts in state $i \in E$,

$$A_i(k) = P(Z_k \in U | Z_0 = i).$$

Value

A matrix with $k + 1$ rows, and with columns giving values of the availability, variances, lower and upper asymptotic confidence limits (if x is an object of class smmfit).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

convolution *Discrete-time convolution product of f and g (See definition 2.2 p. 20)*

Description

Discrete-time convolution product of f and g (See definition 2.2 p. 20)

Usage

```
convolution(f, g)
```

Arguments

f	A vector giving the values of the function f for each $k \in N$.
g	A vector giving the values of the function g for each $k \in N$.

Value

A vector giving the values of the discrete-time convolution of f and g for each $k \in N$.

failureRate *Failure Rate Function*

Description

Function to compute the BMP-failure rate or the RG-failure rate.

Consider a system S_{ystem} starting to work at time $k = 0$. The BMP-failure rate at time $k \in N$ is the conditional probability that the failure of the system occurs at time k, given that the system has worked until time $k - 1$.

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

Usage

```
failureRate(
  x,
  k,
  upstates = x$states,
  failure.rate = c("BMP", "RG"),
  level = 0.95,
  epsilon = 0.001,
  klim = 10000
)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the period $[0, k]$ on which the BMP-failure rate should be computed.
upstates	Vector giving the subset of operational states U.
failure.rate	Type of failure rate to compute. If failure.rate = "BMP" (default value), then BMP-failure-rate is computed. If failure.rate = "RG", the RG-failure rate is computed.
level	Confidence level of the asymptotic confidence interval. Helpful for an object x of class smmfit.
epsilon	Value of the reliability above which the latter is supposed to be 0 because of computation errors (see Details).
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the failure rate of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The system starts to work at instant 0 and the state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

The BMP-failure rate at time $k \in N$ is the conditional probability that the failure of the system occurs at time k, given that the system has worked until time $k - 1$.

Let T_D denote the first passage time in subset D, called the lifetime of the system, i.e.,

$$T_D := \inf\{n \in N;\ Z_n \in D\} \text{ and } \inf \emptyset := \infty.$$

For a discrete-time semi-Markov system, the failure rate at time $k \geq 1$ has the expression:

$$\lambda(k) := P(T_D = k | T_D \geq k)$$

We can rewrite it as follows :

$$\lambda(k) = 1 - \frac{R(k)}{R(k-1)}, \text{ if } R(k-1) \neq 0;\ \lambda(k) = 0, \text{otherwise}$$

The failure rate at time $k = 0$ is defined by $\lambda(0) := 1 - R(0)$, with R being the reliability function (see reliability function).

The calculation of the reliability R involves the computation of many convolutions. It implies that the computation error, may be higher (in value) than the "true" reliability itself for reliability close to 0. In order to avoid inconsistent values of the BMP-failure rate, we use the following formula:

$$\lambda(k) = 1 - \frac{R(k)}{R(k-1)}, \text{ if } R(k-1) \geq \epsilon; \ \lambda(k) = 0, \text{otherwise}$$

with ϵ, the threshold, the parameter `epsilon` in the function `failureRate`.

Expressing the RG-failure rate $r(k)$ in terms of the reliability R we obtain that the RG-failure rate function for a discrete-time system is given by:

$$r(k) = -\ln \frac{R(k)}{R(k-1)}, \text{ if } k \geq 1; \ r(k) = -\ln R(0), \text{ if } k = 0$$

for $R(k) \neq 0$. If $R(k) = 0$, we set $r(k) := 0$.

Note that the RG-failure rate is related to the BMP-failure rate by:

$$r(k) = -\ln(1 - \lambda(k)), \ k \in N$$

Value

A matrix with $k + 1$ rows, and with columns giving values of the BMP-failure rate or RG-failure rate, variances, lower and upper asymptotic confidence limits (if `x` is an object of class `smmfit`).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

R.E. Barlow, A.W. Marshall, and F. Prochan. (1963). Properties of probability distributions with monotone hazard rate. Ann. Math. Statist., 34, 375-389.

D. Roy and R. Gupta. (1992). Classification of discrete lives. Microelectron. Reliabil., 32 (10), 1459-1473.

`fitmm` *Maximum Likelihood Estimation (MLE) of a k-th order Markov chain*

Description

Maximum Likelihood Estimation of the transition matrix and initial distribution of a k-th order Markov chain starting from one or several sequences.

Usage

```
fitmm(sequences, states, k = 1, init.estim = "mle")
```

Arguments

`sequences`	A list of vectors representing the sequences.
`states`	Vector of state space (of length s).
`k`	Order of the Markov chain.
`init.estim`	Optional. `init.estim` gives the method used to estimate the initial distribution. The following methods are proposed:

- `init.estim = "mle"`: the classical Maximum Likelihood Estimator is used to estimate the initial distribution `init`;
- `init.estim = "stationary"`: the initial distribution is replaced by the stationary distribution of the Markov chain (if the order of the Markov chain is more than one, the transition matrix is converted into a square block matrix in order to estimate the stationary distribution);
- `init.estim = "freq"`: the initial distribution is estimated by taking the frequencies of the words of length `k` for all sequences;
- `init.estim = "prod"`: `init` is estimated by using the product of the frequencies of each letter (for all the sequences) in the word of length `k`;
- `init.estim = "unif"`: the initial probability of each state is equal to $1/s$, with s the number of states.

Details

Let $X_1, X_2, ..., X_n$ be a trajectory of length n of the Markov chain $X = (X_m)_{m \in N}$ of order $k = 1$ with transition matrix $p_{trans}(i, j) = P(X_{m+1} = j | X_m = i)$. The maximum likelihood estimation of the transition matrix is $\widehat{p_{trans}}(i, j) = \frac{N_{ij}}{N_{i.}}$, where N_{ij} is the number of transitions from state i to state j and $N_{i.}$ is the number of transition from state i to any state. For $k > 1$ we have similar expressions.

The initial distribution of a k-th order Markov chain is defined as $\mu_i = P(X_1 = i)$. Five methods are proposed for the estimation of the latter :

Maximum Likelihood Estimator: The Maximum Likelihood Estimator for the initial distribution. The formula is: $\widehat{\mu_i} = \frac{Nstart_i}{L}$, where $Nstart_i$ is the number of occurences of the word i (of length k) at the beginning of each sequence and L is the number of sequences. This estimator is reliable when the number of sequences L is high.

Stationary distribution: The stationary distribution is used as a surrogate of the initial distribution. If the order of the Markov chain is more than one, the transition matrix is converted into a square block matrix in order to estimate the stationary distribution. This method may take time if the order of the Markov chain is high (more than three (3)).

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

Product of the frequencies of each state: The initial distribution is estimated by using the product of the frequencies of each state (for all the sequences) in the word of length k.

Uniform distribution: The initial probability of each state is equal to $1/s$, with s, the number of states.

Value

An object of class S3 `mmfit` (inheriting from the S3 class mm). The S3 class `mmfit` contains:

- All the attributes of the S3 class mm;
- An attribute `M` which is an integer giving the total length of the set of sequences `sequences` (sum of all the lengths of the list `sequences`);
- An attribute `logLik` which is a numeric value giving the value of the log-likelihood of the specified Markov model based on the `sequences`;
- An attribute `sequences` which is equal to the parameter `sequences` of the function `fitmm` (i.e. a list of sequences used to estimate the Markov model).

See Also

mm, simulate.mm

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)
k <- 2
init <- rep.int(1 / s ^ k, s ^ k)
p <- matrix(0.25, nrow = s ^ k, ncol = s)

# Specify a Markov model of order 2
markov <- mm(states = states, init = init, ptrans = p, k = k)

seqs <- simulate(object = markov, nsim = c(1000, 10000, 2000), seed = 150)

est <- fitmm(sequences = seqs, states = states, k = 2)
```

`fitsmm` *Maximum Likelihood Estimation (MLE) of a semi-Markov chain*

Description

Maximum Likelihood Estimation of a semi-Markov chain starting from one or several sequences. This estimation can be parametric or non-parametric, non-censored, censored at the beginning and/or at the end of the sequence, with one or several trajectories. Several parametric distributions are considered (Uniform, Geometric, Poisson, Discrete Weibull and Negative Binomial).

Usage

```
fitsmm(
  sequences,
  states,
  type.sojourn = c("fij", "fi", "fj", "f"),
  distr = "nonparametric",
  init.estim = "mle",
  cens.beg = FALSE,
  cens.end = FALSE
)
```

Arguments

`sequences`	A list of vectors representing the sequences.
`states`	Vector of state space (of length s).
`type.sojourn`	Type of sojourn time (for more explanations, see Details).
`distr`	By default `"nonparametric"` for the non-parametric estimation case. If the parametric estimation case is desired, `distr` should be: • A matrix of distributions of dimension (s, s) if `type.sojourn` = `"fij"`; • A vector of distributions of length s if `type.sojourn` = `"fi"` or `"fj"`; • A distribution if `type.sojourn` = `"f"`. The distributions to be used in `distr` must be one of `"unif"`, `"geom"`, `"pois"`, `"dweibull"`, `"nbinom"`.
`init.estim`	Optional. `init.estim` gives the method used to estimate the initial distribution. The following methods are proposed: • `init.estim` = `"mle"`: the classical Maximum Likelihood Estimator is used to estimate the initial distribution `init`; • `init.estim` = `"limit"`: the initial distribution is replaced by the limit (stationary) distribution of the semi-Markov chain; • `init.estim` = `"freq"`: the initial distribution is replaced by the frequencies of each state in the sequences; • `init.estim` = `"unif"`: the initial probability of each state is equal to $1/s$, with s the number of states.
`cens.beg`	Optional. A logical value indicating whether or not sequences are censored at the beginning.
`cens.end`	Optional. A logical value indicating whether or not sequences are censored at the end.

Details

This function estimates a semi-Markov model in parametric or non-parametric case, taking into account the type of sojourn time and the censoring described in references. The non-parametric estimation concerns sojourn time distributions defined by the user. For the parametric estimation, several discrete distributions are considered (see below).

The difference between the Markov model and the semi-Markov model concerns the modeling of the sojourn time. With a Markov chain, the sojourn time distribution is modeled by a Geometric distribution (in discrete time). With a semi-Markov chain, the sojourn time can be any arbitrary distribution. In this package, the available distribution for a semi-Markov model are :

- Uniform: $f(x) = \frac{1}{n}$ for $1 \leq x \leq n$. n is the parameter;
- Geometric: $f(x) = \theta(1-\theta)^{x-1}$ for $x = 1, 2, \ldots, n$, $0 < \theta < 1$, θ is the probability of success. θ is the parameter;
- Poisson: $f(x) = \frac{\lambda^x exp(-\lambda)}{x!}$ for $x = 0, 1, 2, \ldots, n$, with $\lambda > 0$. λ is the parameter;
- Discrete Weibull of type 1: $f(x) = q^{(x-1)^\beta} - q^{x^\beta}$, $x = 1, 2, \ldots, n$, with $0 < q < 1$, the first parameter and $\beta > 0$ the second parameter. (q, β) are the parameters;
- Negative binomial: $f(x) = \frac{\Gamma(x+\alpha)}{\Gamma(\alpha)x!}p^\alpha(1-p)^x$, for $x = 0, 1, 2, \ldots, n$, Γ is the Gamma function, α is the parameter of overdispersion and p is the probability of success, $0 < p < 1$. (α, p) are the parameters;
- Non-parametric.

We define :

- the semi-Markov kernel $q_{ij}(k) = P(J_{m+1} = j, T_{m+1} - T_m = k | J_m = i)$;
- the transition matrix $(p_{trans}(i,j))_{i,j} \in states$ of the embedded Markov chain $J = (J_m)_m$, $p_{trans}(i,j) = P(J_{m+1} = j | J_m = i)$;
- the initial distribution $\mu_i = P(J_1 = i) = P(Z_1 = i)$, $i \in 1, 2, \ldots, s$;
- the conditional sojourn time distributions $(f_{ij}(k))_{i,j} \in states,\ k \in N,\ f_{ij}(k) = P(T_{m+1} - T_m = k | J_m = i, J_{m+1} = j)$, f is specified by the argument `param` in the parametric case and by `distr` in the non-parametric case.

The maximum likelihood estimation of the transition matrix of the embedded Markov chain is $\widehat{p_{trans}}(i,j) = \frac{N_{ij}}{N_{i.}}$.

Five methods are proposed for the estimation of the initial distribution :

Maximum Likelihood Estimator: The Maximum Likelihood Estimator for the initial distribution. The formula is: $\widehat{\mu}_i = \frac{Nstart_i}{L}$, where $Nstart_i$ is the number of occurences of the word i (of length k) at the beginning of each sequence and L is the number of sequences. This estimator is reliable when the number of sequences L is high.

Limit (stationary) distribution: The limit (stationary) distribution of the semi-Markov chain is used as a surrogate of the initial distribution.

Frequencies of each state: The initial distribution is replaced by taking the frequencies of each state in the sequences.

Uniform distribution: The initial probability of each state is equal to $1/s$, with s, the number of states.

Note that $q_{ij}(k) = p_{trans}(i,j)\ f_{ij}(k)$ in the general case (depending on the present state and on the next state). For particular cases, we replace $f_{ij}(k)$ by $f_{i.}(k)$ (depending on the present state i), $f_{.j}(k)$ (depending on the next state j) and $f_{..}(k)$ (depending neither on the present state nor on the next state).

In this package we can choose different types of sojourn time. Four options are available for the sojourn times:

- depending on the present state and on the next state (`fij`);
- depending only on the present state (`fi`);
- depending only on the next state (`fj`);
- depending neither on the current, nor on the next state (`f`).

If `type.sojourn = "fij"`, `distr` is a matrix of dimension (s, s) (e.g., if the 1st element of the 2nd column is `"pois"`, that is to say we go from the first state to the second state following a Poisson distribution). If `type.sojourn = "fi"` or `"fj"`, `distr` must be a vector (e.g., if the first element of vector is `"geom"`, that is to say we go from (or to) the first state to (or from) any state according to a Geometric distribution). If `type.sojourn = "f"`, `distr` must be one of `"unif"`, `"geom"`, `"pois"`, `"dweibull"`, `"nbinom"` (e.g., if `distr` is equal to `"nbinom"`, that is to say that the sojourn time when going from one state to another state follows a Negative Binomial distribution). For the non-parametric case, `distr` is equal to `"nonparametric"` whatever type of sojourn time given.

If the sequence is censored at the beginning and/or at the end, `cens.beg` must be equal to `TRUE` and/or `cens.end` must be equal to `TRUE`. All the sequences must be censored in the same way.

Value

Returns an object of S3 class `smmfit` (inheriting from the S3 class `smm` and smmnonparametric class if `distr = "nonparametric"` or smmparametric otherwise). The S3 class `smmfit` contains:

- All the attributes of the S3 class smmnonparametric or smmparametric depending on the type of estimation;
- An attribute `M` which is an integer giving the total length of the set of sequences `sequences` (sum of all the lengths of the list `sequences`);
- An attribute `logLik` which is a numeric value giving the value of the log-likelihood of the specified semi-Markov model based on the `sequences`;
- An attribute `sequences` which is equal to the parameter `sequences` of the function `fitsmm` (i.e. a list of sequences used to estimate the Markov model).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

See Also

smmnonparametric, smmparametric, simulate.smm, simulate.smmfit, plot.smm, plot.smmfit

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)

# Creation of the initial distribution
vect.init <- c(1 / 4, 1 / 4, 1 / 4, 1 / 4)
```

```
# Creation of the transition matrix
pij <- matrix(c(0, 0.2, 0.5, 0.3,
                0.2, 0, 0.3, 0.5,
                0.3, 0.5, 0, 0.2,
                0.4, 0.2, 0.4, 0),
              ncol = s, byrow = TRUE)

# Creation of the distribution matrix

distr.matrix <- matrix(c(NA, "pois", "geom", "nbinom",
                         "geom", NA, "pois", "dweibull",
                         "pois", "pois", NA, "geom",
                         "pois", "geom", "geom", NA),
                       nrow = s, ncol = s, byrow = TRUE)

# Creation of an array containing the parameters
param1.matrix <- matrix(c(NA, 2, 0.4, 4,
                          0.7, NA, 5, 0.6,
                          2, 3, NA, 0.6,
                          4, 0.3, 0.4, NA),
                        nrow = s, ncol = s, byrow = TRUE)

param2.matrix <- matrix(c(NA, NA, NA, 0.6,
                          NA, NA, NA, 0.8,
                          NA, NA, NA, NA,
                          NA, NA, NA, NA),
                        nrow = s, ncol = s, byrow = TRUE)

param.array <- array(c(param1.matrix, param2.matrix), c(s, s, 2))

# Specify the semi-Markov model
semimarkov <- smmparametric(states = states, init = vect.init, ptrans = pij,
                            type.sojourn = "fij", distr = distr.matrix,
                            param = param.array)

seqs <- simulate(object = semimarkov, nsim = c(1000, 10000, 2000), seed = 100)

# Estimation of simulated sequences
est <- fitsmm(sequences = seqs, states = states, type.sojourn = "fij",
              distr = distr.matrix)
```

`get.f` *Method to get the conditional sojourn time distribution f*

Description

Computes the conditional sojourn time distribution $f(k)$, $f_i(k)$, $f_j(k)$ or $f_{ij}(k)$.

Usage

```
get.f(x, k)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A strictly positive integer giving the time horizon (k must be different from 0 since we assume that there are not instantaneous transitions).

Value

A vector, matrix or array giving the value of f at each time between 1 and k.

---

get.H — *Function to compute the value of the sojourn time cumulative distribution H*

Description

Function to compute the value of H (See equation (3.4) p.46).

Usage

```
get.H(q)
```

Arguments

q	An array giving the values of the kernel for a giving time horizon $[0, \ldots, k]$ (This kernel q is the output of the method getKernel or get.qy).

Value

An array giving the value of $H(k)$ at each time between 0 and k.

get.Kpar *Method to get the number of parameters of a Markov or semi-Markov chain*

Description

Method to get the number of parameters of a Markov or such as AIC and BIC.

Usage

```
get.Kpar(x)
```

Arguments

x	An object for which the number of parameters can be returned (An object of class smm or mm).

Value

A positive integer giving the number of parameters.

get.limitDistribution *Method to get the limit (stationary) distribution*

Description

Method to get the limit (stationary) distribution of a semi-Markov chain.

Usage

```
get.limitDistribution(x, klim = 10000)
```

Arguments

x	An object of S3 class smmfit or smm.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function).

Value

A vector of length card(E) giving the values of the limit distribution.

get.Niuj

Function giving the value of the counting process Niuj used in the estimation of the kernel and the transition matrix of censored and non-parametric semi-markov chains (cf. article Exact MLE and asymptotic properties for nonparametric semi-Markov models)

Description

Function giving the value of the counting process Niuj used in the estimation of the kernel and the transition matrix of censored and non-parametric semi-markov chains (cf. article Exact MLE and asymptotic properties for nonparametric semi-Markov models)

Usage

```
get.Niuj(sequences)
```

Arguments

sequences	A list of sequences of states.

Value

An array giving the values of the counting process N_{iuj}.

get.P

Method to compute the value of P

Description

Method to compute the value of P (See equation (3.33) p.59).

Usage

```
get.P(x, k, states = x$states, var = FALSE, klim = 10000)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the time horizon.
states	Vector giving the states for which the mean sojourn time should be computed. states is a subset of E.
var	Logical. If TRUE the asymptotic variance is computed.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Value

An array giving the value of $P_{i,j}(k)$ at each time between 0 and k if var = FALSE. If var = TRUE, a list containing the following components:

x: an array giving the value of $P_{ij}(k)$ at each time between 0 and k;

sigma2: an array giving the asymptotic variance of the estimator $\sigma^2_P(i,j,k)$.

---

get.psi *Function to compute the value of the matrix-valued function ψ*

Description

Function to compute the value of ψ, the matrix-valued function (See equation (3.16) p.53).

Usage

```
get.psi(q)
```

Arguments

q	An array giving the values of the kernel for a giving time horizon $[0,\ldots,k]$ (This kernel q is the output of the method getKernel or .get.qy).

Value

An array giving the value of $\psi(k)$ at each time between 0 and k.

---

get.Py *Method to compute the value of P_Y*

Description

Method to compute the value of P_Y (See Proposition 5.1 p.105-106).

Usage

```
get.Py(x, k, upstates = x$states)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the time horizon.
upstates	Vector giving the subset of operational states U.

Value

An array giving the value of $P_Y(k)$ at each time between 0 and k.

get.qy *Method to get the semi-Markov kernel q_Y*

Description

Computes the semi-Markov kernel $q_Y(k)$ (See proposition 5.1 p.106).

Usage

```
get.qy(x, k, upstates = x$states)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the time horizon.
upstates	Vector giving the subset of operational states U.

Value

An array giving the value of $q_Y(k)$ at each time between 0 and k.

get.stationaryDistribution *Method to get the stationary distribution*

Description

Method to get the stationary distribution of a Markov chain. If the order of the Markov chain is higher than one, a block matrix is computed to get the stationary distribution.

Usage

```
get.stationaryDistribution(x)
```

Arguments

x	An object of S3 class mm or mmfit.

Value

A vector of length card(E) giving the values of the stationary distribution.

getKernel *Method to get the semi-Markov kernel q*

Description

Computes the semi-Markov kernel $q_{ij}(k)$.

Usage

```
getKernel(x, k, var = FALSE, klim = 10000)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the time horizon.
var	Logical. If TRUE the asymptotic variance is computed.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Value

An array giving the value of $q_{ij}(k)$ at each time between 0 and k if var = FALSE. If var = TRUE, a list containing the following components:

x: an array giving the value of $q_{ij}(k)$ at each time between 0 and k;

sigma2: an array giving the asymptotic variance of the estimator $\sigma_q^2(i,j,k)$.

getProcesses *Function to compute processes based on a list of sequences*

Description

Function to compute processes based on a list of sequences

Usage

```
getProcesses(sequences)
```

Arguments

sequences	A list of sequences of states.

Value

A list giving:

s: Cardinal of the states space;

states: Vector of state space of length s;

M: The total length (sum of the length of each sequence);

L: The number of sequences;

kmax : Maximum length of the sojourn times;

Ym: List of semi-Markov chains;

Jm: List of successively visited states for each sequence;

Tm: List of successive time points when state changes for each sequence;

Lm: List of sojourn time for each sequence;

Um: List of backward recurrence time for each sequence;

counting: List giving the counting processes such as $N_{ij}(k)$, $N_i(k)$, $N_j(k)$...

`is.mm` *Function to check if an object is of class* `mm`

Description

`is.mm` returns `TRUE` if `x` is an object of class `mm`.

Usage

```
is.mm(x)
```

Arguments

`x`	An arbitrary R object.

Value

`is.mm` returns `TRUE` or `FALSE` depending on whether `x` is an object of class `mm` or not.

is.mmfit *Function to check if an object is of class* mmfit

Description

is.mmfit returns TRUE if x is an object of class mmfit.

Usage

```
is.mmfit(x)
```

Arguments

x	An arbitrary R object.

Value

is.mmfit returns TRUE or FALSE depending on whether x is an object of class mmfit or not.

is.smm *Function to check if an object is of class* smm

Description

is.smm returns TRUE if x is an object of class smm.

Usage

```
is.smm(x)
```

Arguments

x	An arbitrary R object.

Value

is.smm returns TRUE or FALSE depending on whether x is an object of class smm or not.

is.smmfit *Function to check if an object is of class* smmfit

Description

is.smmfit returns TRUE if x is an object of class smmfit.

Usage

```
is.smmfit(x)
```

Arguments

x	An arbitrary R object.

Value

is.smmfit returns TRUE or FALSE depending on whether x is an object of class smmfit or not.

is.smmnonparametric *Function to check if an object is of class* smmnonparametric

Description

is.smmnonparametric returns TRUE if x is an object of class smmnonparametric.

Usage

```
is.smmnonparametric(x)
```

Arguments

x	An arbitrary R object.

Value

is.smmnonparametric returns TRUE or FALSE depending on whether x is an object of class smmnonparametric or not.

is.smmparametric *Function to check if an object is of class* smmparametric

Description

is.smmparametric returns TRUE if x is an object of class smmparametric.

Usage

```
is.smmparametric(x)
```

Arguments

x	An arbitrary R object.

Value

is.smmparametric returns TRUE or FALSE depending on whether x is an object of class smmparametric or not.

maintainability *Maintainability Function*

Description

For a reparable system S_{ystem} for which the failure occurs at time $k = 0$, its maintainability at time $k \in N$ is the probability that the system is repaired up to time k, given that it has failed at time $k = 0$.

Usage

```
maintainability(x, k, upstates = x$states, level = 0.95, klim = 10000)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the period $[0, k]$ on which the maintainability should be computed.
upstates	Vector giving the subset of operational states U.
level	Confidence level of the asymptotic confidence interval. Helpful for an object x of class smmfit.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the maintainability of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The system starts to fail at instant 0 and the state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

Thus, we take $(\alpha_i := P(Z_0 = i))_{i \in U} = 0$ and we denote by T_U the first hitting time of subset U, called the duration of repair or repair time, that is,

$$T_U := \text{inf}\{n \in N;\ Z_n \in U\} \text{ and inf } \emptyset := \infty.$$

The maintainability at time $k \in N$ of a discrete-time semi-Markov system is

$$M(k) = P(T_U \leq k) = 1 - P(T_U > k) = 1 - P(Z_n \in D,\ n = 0, \ldots, k).$$

Value

A matrix with $k + 1$ rows, and with columns giving values of the maintainability, variances, lower and upper asymptotic confidence limits (if `x` is an object of class `smmfit`).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

`matrixConvolution` *Discrete-time matrix convolution product (See definition 3.5 p. 48)*

Description

Discrete-time matrix convolution product (See definition 3.5 p. 48)

Usage

```
matrixConvolution(a, b)
```

Arguments

a	An array of dimension $(S, S, k+1)$.
b	An array of dimension $(S, S, k+1)$.

Value

An array of dimension $(S, S, k+1)$ giving the discrete-time matrix convolution product for each $k \in N$.

`meanRecurrenceTimes` *Method to get the mean recurrence times* μ

Description

Method to get the mean recurrence times μ.

Usage

```
meanRecurrenceTimes(x, klim = 10000)
```

Arguments

x	An object of S3 class `smmfit` or `smm`.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function).

Details

Consider a system (or a component) S_{ystem} whose possible states during its evolution in time are $E = \{1, \ldots, s\}$.

We are interested in investigating the mean recurrence times of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$.

Let $T = (T_n)_{n \in N}$ denote the successive time points when state changes in $(Z_n)_{n \in N}$ occur and let also $J = (J_n)_{n \in N}$ denote the successively visited states at these time points.

The mean recurrence of an arbitrary state $j \in E$ is given by:

$$\mu_{jj} = \frac{\sum_{i \in E} \nu(i) m_i}{\nu(j)}$$

where $(\nu(1), \ldots, \nu(s))$ is the stationary distribution of the embedded Markov chain $(J_n)_{n \in N}$ and m_i is the mean sojourn time in state $i \in E$ (see meanSojournTimes function for the computation).

Value

A vector giving the mean recurrence time $(\mu_i)_{i \in [1,\ldots,s]}$.

`meanSojournTimes` *Mean Sojourn Times Function*

Description

The mean sojourn time is the mean time spent in each state.

Usage

```
meanSojournTimes(x, states = x$states, klim = 10000)
```

Arguments

`x`	An object of S3 class `smmfit` or `smm`.
`states`	Vector giving the states for which the mean sojourn time should be computed. `states` is a subset of E.
`klim`	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function).

Details

Consider a system (or a component) S_{ystem} whose possible states during its evolution in time are $E = \{1, \ldots, s\}$.

We are interested in investigating the mean sojourn times of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$.

Let $T = (T_n)_{n \in N}$ denote the successive time points when state changes in $(Z_n)_{n \in N}$ occur and let also $J = (J_n)_{n \in N}$ denote the successively visited states at these time points.

The mean sojourn times vector is defined as follows:

$$m_i = E[T_1 | Z_0 = j] = \sum_{k \geq 0} (1 - P(T_{n+1} - T_n \leq k | J_n = j)) = \sum_{k \geq 0} (1 - H_j(k)),\ i \in E$$

Value

A vector of length card(E) giving the values of the mean sojourn times for each state $i \in E$.

mm *Markov model specification*

Description

Creates a model specification of a Markov model.

Usage

```
mm(states, init, ptrans, k = 1)
```

Arguments

states	Vector of state space of length s.
init	Vector of initial distribution of length s ^ k.
ptrans	Matrix of transition probabilities of dimension (s, s).
k	Order of the Markov chain.

Value

An object of class mm.

See Also

simulate.mm, fitmm

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)
k <- 1
init <- rep.int(1 / s, s)
p <- matrix(c(0, 0, 0.3, 0.4, 0, 0, 0.5, 0.2, 0.7, 0.5,
              0, 0.4, 0.3, 0.5, 0.2, 0), ncol = s)

# Specify a Markov model of order 1
markov <- mm(states = states, init = init, ptrans = p, k = k)
```

`mttf` *Mean Time To Failure (MTTF) Function*

Description

Consider a system S_{ystem} starting to work at time $k = 0$. The mean time to failure (MTTF) is defined as the mean lifetime.

Usage

```
mttf(x, upstates = x$states, level = 0.95, klim = 10000)
```

Arguments

`x`	An object of S3 class `smmfit` or `smm`.
`upstates`	Vector giving the subset of operational states U.
`level`	Confidence level of the asymptotic confidence interval. Helpful for an object `x` of class `smmfit`.
`klim`	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the mean time to failure of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The system starts to work at instant 0 and the state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

Let T_D denote the first passage time in subset D, called the lifetime of the system, i.e.,

$$T_D := \inf\{n \in N;\ Z_n \in D\} \text{ and } \inf \emptyset := \infty.$$

The mean time to failure (MTTF) is defined as the mean lifetime, i.e., the expectation of the hitting time to down set D,

$$MTTF = E[T_D]$$

Value

A matrix with card$(U) = s_1$ rows, and with columns giving values of the mean time to failure for each state $i \in U$, variances, lower and upper asymptotic confidence limits (if `x` is an object of class `smmfit`).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

I. Votsi & A. Brouste (2019) Confidence interval for the mean time to failure in semi-Markov models: an application to wind energy production, Journal of Applied Statistics, 46:10, 1756-1773

`mttr` *Mean Time To Repair (MTTR) Function*

Description

Consider a system S_{ystem} that has just failed at time $k = 0$. The mean time to repair (MTTR) is defined as the mean of the repair duration.

Usage

```
mttr(x, upstates = x$states, level = 0.95, klim = 10000)
```

Arguments

`x`	An object of S3 class `smmfit` or `smm`.
`upstates`	Vector giving the subset of operational states U.
`level`	Confidence level of the asymptotic confidence interval. Helpful for an object `x` of class `smmfit`.
`klim`	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the mean time to repair of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The system has just failed at instant 0 and the state of the

system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

Let T_U denote the first passage time in subset U, called the duration of repair or repair time, i.e.,

$$T_U := \text{inf}\{n \in N;\ Z_n \in U\} \text{ and inf } \emptyset := \infty.$$

The mean time to repair (MTTR) is defined as the mean of the repair duration, i.e., the expectation of the hitting time to up set U,

$$MTTR = E[T_U]$$

Value

A matrix with card$(U) = s_1$ rows, and with columns giving values of the mean time to repair for each state $i \in U$, variances, lower and upper asymptotic confidence limits (if `x` is an object of class `smmfit`).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

I. Votsi & A. Brouste (2019) Confidence interval for the mean time to failure in semi-Markov models: an application to wind energy production, Journal of Applied Statistics, 46:10, 1756-1773

`plot.smm` *Plot function for an object of class smm*

Description

Displays the densities for the conditional sojourn time distributions depending on the current state `i` and on the next state `j`.

Usage

```
## S3 method for class 'smm'
plot(x, i, j, klim = NULL, ...)
```

Arguments

x	An object of S3 class smm (inheriting from the S3 class smmnonparametric or smmparametric).
i	An element of the state space vector x$states giving the current state in the following cases: type.sojourn = "fij" or type.sojourn = "fi", otherwise, i is ignored.
j	An element of the state space vector x$states giving the next state in the following cases: type.sojourn = "fij" or type.sojourn = "fj", otherwise, j is ignored.
klim	An integer giving the limit value for which the density will be plotted. If klim is NULL, then quantile or order 0.95 is used.
...	Arguments passed to plot.

Value

None.

---

plot.smmfit *Plot function for an object of class smmfit*

Description

Displays the densities for the conditional sojourn time distributions depending on the current state i and on the next state j.

Usage

```
## S3 method for class 'smmfit'
plot(x, i, j, klim = NULL, ...)
```

Arguments

x	An object of class smmfit (inheriting from the S3 classes smm, smmnonparametric or smmparametric).
i	An element of the state space vector x$states giving the current state in the following cases: type.sojourn = "fij" or type.sojourn = "fi", otherwise, i is ignored.
j	An element of the state space vector x$states giving the next state in the following cases: type.sojourn = "fij" or type.sojourn = "fj", otherwise, j is ignored.
klim	An integer giving the limit value for which the density will be plotted. If klim is NULL, then quantile or order 0.95 is used.
...	Arguments passed to plot.

Value

None.

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)

# Creation of the initial distribution
vect.init <- c(1 / 4, 1 / 4, 1 / 4, 1 / 4)

# Creation of the transition matrix
pij <- matrix(c(0, 0.2, 0.3, 0.4, 0.2, 0, 0.5, 0.2, 0.5,
                0.3, 0, 0.4, 0.3, 0.5, 0.2, 0), ncol = s)

# Creation of the distribution matrix
distr.vec <- c("pois", "geom", "geom", "geom")
parameters <- matrix(c(2, 0.6, 0.8, 0.8, NA, NA, NA, NA),
                     ncol = 2, byrow = FALSE)

# Specify the semi-Markov model
smm <- smmparametric(states = states, init = vect.init, ptrans = pij,
                     type.sojourn = "fi", distr = distr.vec, param = parameters)

seqs <- simulate(object = smm, nsim = rep(5000, 100), seed = 10)

est <- fitsmm(sequences = seqs, states = states, type.sojourn = "fi", distr = distr.vec)

class(est)

plot(x = est, i = "a", col = "blue", pch = "+")
```

`reliability` *Reliability Function*

Description

Consider a system S_{ystem} starting to function at time $k = 0$. The reliability or the survival function of S_{ystem} at time $k \in N$ is the probability that the system has functioned without failure in the period $[0, k]$.

Usage

```
reliability(x, k, upstates = x$states, level = 0.95, klim = 10000)
```

Arguments

x	An object of S3 class smmfit or smm.
k	A positive integer giving the period $[0, k]$ on which the reliability should be computed.
upstates	Vector giving the subset of operational states U.
level	Confidence level of the asymptotic confidence interval. Helpful for an object x of class smmfit.
klim	Optional. The time horizon used to approximate the series in the computation of the mean sojourn times vector m (cf. meanSojournTimes function) for the asymptotic variance.

Details

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

We are interested in investigating the reliability of a discrete-time semi-Markov system S_{ystem}. Consequently, we suppose that the evolution in time of the system is governed by an E-state space semi-Markov chain $(Z_k)_{k \in N}$. The system starts to work at instant 0 and the state of the system is given at each instant $k \in N$ by Z_k: the event $\{Z_k = i\}$, for a certain $i \in U$, means that the system S_{ystem} is in operating mode i at time k, whereas $\{Z_k = j\}$, for a certain $j \in D$, means that the system is not operational at time k due to the mode of failure j or that the system is under the repairing mode j.

Let T_D denote the first passage time in subset D, called the lifetime of the system, i.e.,

$$T_D := \inf\{n \in N;\ Z_n \in D\} \text{ and } \inf \emptyset := \infty.$$

The reliability or the survival function at time $k \in N$ of a discrete-time semi-Markov system is:

$$R(k) := P(T_D > k) = P(Zn \in U, n = 0, \ldots, k)$$

which can be rewritten as follows:

$$R(k) = \sum_{i \in U} P(Z_0 = i) P(T_D > k | Z_0 = i) = \sum_{i \in U} \alpha_i P(T_D > k | Z_0 = i)$$

Value

A matrix with $k + 1$ rows, and with columns giving values of the reliability, variances, lower and upper asymptotic confidence limits (if x is an object of class smmfit).

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

`setSeed` *Set the RNG Seed from within Rcpp*

Description

Set the RNG Seed from within Rcpp

Usage

```
setSeed(seed)
```

Arguments

`seed`	An `unsigned int` that is the seed one wishes to use.

Value

A set RNG scope.

Examples

```
set.seed(10)
x <- rnorm(5, 0, 1)
setSeed(10)
y <- rnorm(5, 0, 1)
all.equal(x, y, check.attributes = FALSE)
```

`simulate.mm` *Simulates k-th order Markov chains*

Description

Simulates k-th order Markov chains.

Usage

```
## S3 method for class 'mm'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	An object of class mm.
nsim	An integer or vector of integers (for multiple sequences) specifying the length of the sequence(s).
seed	Optional. seed for the random number generator. If no seed is given, then seed is set by using the command set.seed(round(as.numeric(Sys.time())).
...	further arguments passed to or from other methods.

Details

If nsim is a single integer then a chain of that length is produced. If nsim is a vector of integers, then length(nsim) sequences are generated with respective lengths.

Value

A list of vectors representing the sequences.

See Also

mm, fitmm

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)
k <- 2
init <- rep.int(1 / s ^ k, s ^ k)
p <- matrix(0.25, nrow = s ^ k, ncol = s)

# Specify a Markov model of order 1
markov <- mm(states = states, init = init, ptrans = p, k = k)

seqs <- simulate(object = markov, nsim = c(1000, 10000, 2000), seed = 150)
```

---

simulate.mmfit *Simulates Markov chains*

Description

Simulates sequences from a fitted Markov model.

Usage

```
## S3 method for class 'mmfit'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	An object of class mmfit.
nsim	An integer or vector of integers (for multiple sequences) specifying the length of the sequence(s).
seed	Optional. seed for the random number generator. If no seed is given, then seed is set by using the command set.seed(round(as.numeric(Sys.time()))).
...	further arguments passed to or from other methods.

Details

If nsim is a single integer then a chain of that length is produced. If nsim is a vector of integers, then length(nsim) sequences are generated with respective lengths.

Value

A list of vectors representing the sequences.

See Also

mm, fitmm

---

simulate.smm *Simulates semi-Markov chains*

Description

Simulates sequences from a semi-Markov model.

Usage

```
## S3 method for class 'smm'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	An object of S3 class smm (inheriting from the S3 class smmnonparametric or smmparametric).
nsim	An integer or vector of integers (for multiple sequences) specifying the length of the sequence(s).
seed	Optional. seed for the random number generator. If no seed is given, then seed is set by using the command set.seed(round(as.numeric(Sys.time()))).
...	further arguments passed to or from other methods.

Details

If nsim is a single integer then a chain of that length is produced. If nsim is a vector of integers, then length(nsim) sequences are generated with respective lengths.

Value

A list of vectors representing the sequences.

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

See Also

smmparametric, smmnonparametric, fitsmm

`simulate.smmfit` *Simulates semi-Markov chains*

Description

Simulates sequences from a fitted semi-Markov model.

Usage

```
## S3 method for class 'smmfit'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`object`	An object of class `smmfit` (inheriting from the S3 classes `smm`, smmnonparametric or smmparametric).
`nsim`	An integer or vector of integers (for multiple sequences) specifying the length of the sequence(s).
`seed`	`seed` for the random number generator.
`...`	further arguments passed to or from other methods.

Details

If `nsim` is a single integer then a chain of that length is produced. If `nsim` is a vector of integers, then `length(nsim)` sequences are generated with respective lengths.

Value

A list of vectors representing the sequences.

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

See Also

smmnonparametric, smmparametric, fitsmm

`smmnonparametric` *Non-parametric semi-Markov model specification*

Description

Creates a non-parametric model specification for a semi-Markov model.

Usage

```
smmnonparametric(
  states,
  init,
  ptrans,
  type.sojourn = c("fij", "fi", "fj", "f"),
  distr,
  cens.beg = FALSE,
  cens.end = FALSE
)
```

Arguments

`states`	Vector of state space of length s.
`init`	Vector of initial distribution of length s.
`ptrans`	Matrix of transition probabilities of the embedded Markov chain $J = (J_m)_m$ of dimension (s, s).
`type.sojourn`	Type of sojourn time (for more explanations, see Details).
`distr`	• Array of dimension $(s, s, kmax)$ if `type.sojourn = "fij"`; • Matrix of dimension $(s, kmax)$ if `type.sojourn = "fi"` or `"fj"`; • Vector of length $kmax$ if the `type.sojourn = "f"`. $kmax$ is the maximum length of the sojourn times.
`cens.beg`	Optional. A logical value indicating whether or not sequences are censored at the beginning.
`cens.end`	Optional. A logical value indicating whether or not sequences are censored at the end.

Details

This function creates a semi-Markov model object in the non-parametric case, taking into account the type of sojourn time and the censoring described in references. The non-parametric specification concerns sojourn time distributions defined by the user.

The difference between the Markov model and the semi-Markov model concerns the modeling of the sojourn time. With a Markov chain, the sojourn time distribution is modeled by a Geometric

distribution (in discrete time). With a semi-Markov chain, the sojourn time can be any arbitrary distribution.

We define :

- the semi-Markov kernel $q_{ij}(k) = P(J_{m+1} = j, T_{m+1} - T_m = k | J_m = i)$;
- the transition matrix $(p_{trans}(i,j))_{i,j} \in states$ of the embedded Markov chain $J = (J_m)_m$, $p_{trans}(i,j) = P(J_{m+1} = j | J_m = i)$;
- the initial distribution $\mu_i = P(J_1 = i) = P(Z_1 = i)$, $i \in 1, 2, \ldots, s$;
- the conditional sojourn time distributions $(f_{ij}(k))_{i,j} \in states,\ k \in N,\ f_{ij}(k) = P(T_{m+1} - T_m = k | J_m = i, J_{m+1} = j)$, f is specified by the argument `distr` in the non-parametric case.

In this package we can choose different types of sojourn time. Four options are available for the sojourn times:

- depending on the present state and on the next state (f_{ij});
- depending only on the present state (f_i);
- depending only on the next state (f_j);
- depending neither on the current, nor on the next state (f).

Let define $kmax$ the maximum length of the sojourn times. If `type.sojourn = "fij"`, `distr` is an array of dimension $(s, s, kmax)$. If `type.sojourn = "fi"` or `"fj"`, `distr` must be a matrix of dimension $(s, kmax)$. If `type.sojourn = "f"`, `distr` must be a vector of length $kmax$.

If the sequence is censored at the beginning and/or at the end, `cens.beg` must be equal to `TRUE` and/or `cens.end` must be equal to `TRUE`. All the sequences must be censored in the same way.

Value

Returns an object of class `smm`, smmnonparametric.

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

See Also

simulate, fitsmm, smmparametric

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)

# Creation of the initial distribution
vect.init <- c(1 / 4, 1 / 4, 1 / 4, 1 / 4)

# Creation of the transition matrix
pij <- matrix(c(0, 0.2, 0.5, 0.3,
```

```
                0.2, 0, 0.3, 0.5,
                0.3, 0.5, 0, 0.2,
                0.4, 0.2, 0.4, 0),
              ncol = s, byrow = TRUE)

# Creation of a matrix corresponding to the
# conditional sojourn time distributions
kmax <- 6
nparam.matrix <- matrix(c(0.2, 0.1, 0.3, 0.2,
                          0.2, 0, 0.4, 0.2,
                          0.1, 0, 0.2, 0.1,
                          0.5, 0.3, 0.15, 0.05,
                          0, 0, 0.3, 0.2,
                          0.1, 0.2, 0.2, 0),
                        nrow = s, ncol = kmax, byrow = TRUE)

semimarkov <- smmnonparametric(states = states, init = vect.init, ptrans = pij,
                               type.sojourn = "fj", distr = nparam.matrix)

semimarkov
```

`smmparametric` *Parametric semi-Markov model specification*

Description

Creates a parametric model specification for a semi-Markov model.

Usage

```
smmparametric(
  states,
  init,
  ptrans,
  type.sojourn = c("fij", "fi", "fj", "f"),
  distr,
  param,
  cens.beg = FALSE,
  cens.end = FALSE
)
```

Arguments

`states`	Vector of state space of length s.
`init`	Vector of initial distribution of length s.
`ptrans`	Matrix of transition probabilities of the embedded Markov chain $J = (J_m)_m$ of dimension (s, s).

`type.sojourn` Type of sojourn time (for more explanations, see Details).

`distr`
- Matrix of distributions of dimension (s, s) if `type.sojourn = "fij"`;
- Vector of distributions of length s if `type.sojourn = "fi"` or `"fj"`;
- A distribution if `type.sojourn = "f"`.

where the distributions to be used can be one of `unif`, `geom`, `pois`, `dweibull` or `nbinom`.

`param` Parameters of sojourn time distributions:
- Array of distribution parameters of dimension $(s, s, 2)$ (2 corresponds to the maximal number of distribution parameters) if `type.sojourn = "fij"`;
- Matrix of distribution parameters of dimension $(s, 2)$ if `type.sojourn = "fi"` or `"fj"`;
- Vector of distribution parameters of length 2 if `type.sojourn = "f"`.

When parameters/values are not necessary (e.g. the Poisson distribution has only one parameter that is λ, leave the value `NA` for the second parameter in the argument `param`).

`cens.beg` Optional. A logical value indicating whether or not sequences are censored at the beginning.

`cens.end` Optional. A logical value indicating whether or not sequences are censored at the end.

Details

This function creates a semi-Markov model object in the parametric case, taking into account the type of sojourn time and the censoring described in references. For the parametric specification, several discrete distributions are considered (see below).

The difference between the Markov model and the semi-Markov model concerns the modeling of the sojourn time. With a Markov chain, the sojourn time distribution is modeled by a Geometric distribution (in discrete time). With a semi-Markov chain, the sojourn time can be any arbitrary distribution. In this package, the available distribution for a semi-Markov model are :

- Uniform: $f(x) = 1/n$ for $a \leq x \leq b$, with $n = b - a + 1$;
- Geometric: $f(x) = \theta(1-\theta)^x$ for $x = 0, 1, 2, \ldots, n$, $0 < \theta < 1$, with $n > 0$ and θ is the probability of success;
- Poisson: $f(x) = (\lambda^x exp(-\lambda))/x!$ for $x = 0, 1, 2, \ldots, n$, with $n > 0$ and $\lambda > 0$;
- Discrete Weibull of type 1: $f(x) = q^{(x-1)^\beta} - q^{x^\beta}, x = 1, 2, 3, \ldots, n$, with $n > 0$, q is the first parameter and β is the second parameter;
- Negative binomial: $f(x) = \frac{\Gamma(x+\alpha)}{\Gamma(\alpha)x!}p^\alpha(1-p)^x$, for $x = 0, 1, 2, \ldots, n$, Γ is the Gamma function, α is the parameter of overdispersion and p is the probability of success, $0 < p < 1$;
- Non-parametric.

We define :

- the semi-Markov kernel $q_{ij}(k) = P(J_{m+1} = j, T_{m+1} - T_m = k | J_m = i)$;
- the transition matrix $(p_{trans}(i,j))_{i,j} \in states$ of the embedded Markov chain $J = (J_m)_m$, $p_{trans}(i,j) = P(J_{m+1} = j | J_m = i)$;

- the initial distribution $\mu_i = P(J_1 = i) = P(Z_1 = i), i \in 1, 2, \ldots, s$;
- the conditional sojourn time distributions $(f_{ij}(k))_{i,j} \in states,\ k \in N,\ f_{ij}(k) = P(T_{m+1} - T_m = k | J_m = i, J_{m+1} = j)$, f is specified by the argument `param` in the parametric case.

In this package we can choose different types of sojourn time. Four options are available for the sojourn times:

- depending on the present state and on the next state (f_{ij});
- depending only on the present state (f_i);
- depending only on the next state (f_j);
- depending neither on the current, nor on the next state (f).

If `type.sojourn = "fij"`, `distr` is a matrix of dimension (s, s) (e.g., if the row 1 of the 2nd column is `"pois"`, that is to say we go from the first state to the second state following a Poisson distribution). If `type.sojourn = "fi"` or `"fj"`, `distr` must be a vector (e.g., if the first element of vector is `"geom"`, that is to say we go from the first state to any state according to a Geometric distribution). If `type.sojourn = "f"`, `distr` must be one of `"unif"`, `"geom"`, `"pois"`, `"dweibull"`, `"nbinom"` (e.g., if `distr` is equal to `"nbinom"`, that is to say that the sojourn times when going from any state to any state follows a Negative Binomial distribution). For the non-parametric case, `distr` is equal to `"nonparametric"` whatever type of sojourn time given.

If the sequence is censored at the beginning and/or at the end, `cens.beg` must be equal to `TRUE` and/or `cens.end` must be equal to `TRUE`. All the sequences must be censored in the same way.

Value

Returns an object of class smmparametric.

References

V. S. Barbu, N. Limnios. (2008). Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications - Their Use in Reliability and DNA Analysis. New York: Lecture Notes in Statistics, vol. 191, Springer.

See Also

simulate, fitsmm, smmnonparametric

Examples

```
states <- c("a", "c", "g", "t")
s <- length(states)

# Creation of the initial distribution
vect.init <- c(1 / 4, 1 / 4, 1 / 4, 1 / 4)

# Creation of the transition matrix
pij <- matrix(c(0, 0.2, 0.5, 0.3,
                0.2, 0, 0.3, 0.5,
                0.3, 0.5, 0, 0.2,
                0.4, 0.2, 0.4, 0),
```

```
              ncol = s, byrow = TRUE)

# Creation of the distribution matrix

distr.matrix <- matrix(c(NA, "pois", "geom", "nbinom",
                         "geom", NA, "pois", "dweibull",
                         "pois", "pois", NA, "geom",
                         "pois", "geom", "geom", NA),
                       nrow = s, ncol = s, byrow = TRUE)

# Creation of an array containing the parameters
param1.matrix <- matrix(c(NA, 2, 0.4, 4,
                          0.7, NA, 5, 0.6,
                          2, 3, NA, 0.6,
                          4, 0.3, 0.4, NA),
                        nrow = s, ncol = s, byrow = TRUE)

param2.matrix <- matrix(c(NA, NA, NA, 0.6,
                          NA, NA, NA, 0.8,
                          NA, NA, NA, NA,
                          NA, NA, NA, NA),
                        nrow = s, ncol = s, byrow = TRUE)

param.array <- array(c(param1.matrix, param2.matrix), c(s, s, 2))

# Specify the semi-Markov model
semimarkov <- smmparametric(states = states, init = vect.init, ptrans = pij,
                            type.sojourn = "fij", distr = distr.matrix,
                            param = param.array)
semimarkov
```

Index

∗ **Markov**
smmR-package, 3
∗ **availability**
smmR-package, 3
∗ **censored**
smmR-package, 3
∗ **estimation**
smmR-package, 3
∗ **failure**
smmR-package, 3
∗ **maintainability**
smmR-package, 3
∗ **reliability**
smmR-package, 3
∗ **semi-Markov**
smmR-package, 3
∗ **simulation**
smmR-package, 3
.failureRateBMP, 5

availability, 6

convolution, 8

failureRate, 8
fitmm, 10, *30*, *38, 39*
fitsmm, 12, *40–42*, *45*

get.f, 16
get.H, 17
get.Kpar, 18
get.limitDistribution, 18
get.Niuj, 19
get.P, 19
get.psi, 20
get.Py, 20
get.qy, 21
get.stationaryDistribution, 21
getKernel, 22
getProcesses, 22

is.mm, 23
is.mmfit, 24
is.smm, 24
is.smmfit, 25
is.smmnonparametric, 25
is.smmparametric, 26

maintainability, 26
matrixConvolution, 27
meanRecurrenceTimes, 28
meanSojournTimes, *5*, *7*, *9*, *18, 19*, *22*, *26*, *28*, 29, *29*, *31, 32*, *36*
mm, *12*, 30, *30*, *38, 39*
mttf, 31
mttr, 32

plot.smm, *15*, 33
plot.smmfit, *15*, 34

reliability, *6*, *10*, 35

setSeed, 37
simulate, *42*, *45*
simulate.mm, *12*, *30*, 37
simulate.mmfit, 38
simulate.smm, *15*, 39
simulate.smmfit, *15*, 40
smmnonparametric, *15*, *34*, *39, 40*, 41, *41, 42*, *45*
smmparametric, *15*, *34*, *39–42*, 43, *45*
smmR *(*smmR-package*)*, 3
smmR-package, 3