Package 'icmstate'

May 8, 2026

Type Package

Title Interval Censored Multi-State Models

Version 0.2.0

Maintainer Daniel Gomon <dgstatsoft@gmail.com>

Description Allows for the non-parametric estimation of transition intensities in interval-censored multi-state models using the approach of Gomon and Putter (2024) <doi:10.48550/arXiv.2409.07176> or Gu et al. (2023) <doi:10.1093/biomet/asad073>.

License GPL (>= 2)

Imports Rcpp, mstate, prodlim, igraph (>= 1.3.0), checkmate, ggplot2, deSolve, msm, survival, JOPS

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), icenReg, profvis, knitr, rmarkdown, bookdown, latex2exp

Config/testthat/edition 3

RoxygenNote 7.3.2

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

Author Daniel Gomon [aut, cre] (ORCID: <https://orcid.org/0000-0001-9011-3743>), Hein Putter [aut] (ORCID: <https://orcid.org/0000-0001-5395-1422>)

Repository CRAN

Date/Publication 2025-07-04 10:00:02 UTC

Contents

icmstate-package 2
ageqb 3
agreaterb 4

ainB . . . 4
Aintersectb . . . 5
Alargerb . . . 5
AsubsetB . . . 6
direct_from_observed_intervals . . . 6
estimate_support_msm . . . 7
evalstep . . . 8
extend_msfit . . . 9
get_trans_intervals . . . 9
graphfromIntervals . . . 10
interpol_msfit . . . 11
msm_frydman . . . 12
npmsm . . . 13
plot.npmsm . . . 18
plot.probtrans.subjects . . . 18
plot.smoothmsm . . . 19
plot_probtrans . . . 20
plot_surv . . . 22
predict_tp . . . 23
print.npmsm . . . 25
print.summary.probtrans.subjects . . . 25
probtrans_coxph . . . 26
probtrans_weib . . . 29
prod_lambda_G_base . . . 30
remove_redundant_observations . . . 30
sim_id_weib . . . 31
sim_weibmsm . . . 33
smoothmsm . . . 35
summary.probtrans.subjects . . . 37
supportHudgens . . . 39
support_from_direct_intervals . . . 40
support_npmsm . . . 40
transprob.msm . . . 41
visualise_data . . . 42
visualise_msm . . . 43

Index **45**

`icmstate-package` *icmstate*

Description

Non-parametric estimation of transition intensities in interval-censored multi-state models

Details

Allows for the estimation of transition intensities in interval-censored multi-state models using the approach of Gomon and Putter (2024) (Multinomial likelihood) or Gu et al. (2023) (Poisson likelihood).

Author(s)

Maintainer: Daniel Gomon <dgstatsoft@gmail.com> [aut, cre]
Hein Putter [aut]

References

Y. Gu, D. Zeng, G. Heiss, and D. Y. Lin, Maximum likelihood estimation for semiparametric regression models with interval-censored multistate data, Biometrika, Nov. 2023, doi:10.1093/biomet/asad073

D. Gomon and H. Putter, Non-parametric estimation of transition intensities in interval censored Markov multi-state models without loops, arXiv, 2024, doi:10.48550/arXiv.2409.07176

`ageqb` *Check if event time is larger/equal than other event time*

Description

Function which takes as input a vector a with event times and a vector b with event times and checks whether each entry in a is larger or equal than the entries in b.

Usage

```
ageqb(a, b)
```

Arguments

`a`	Vector of event times
`b`	Vector of event times

Value

Matrix of size (length(a) * length(b)) with binary values indicating whether the event times in a are larger than the event times in b

agreaterb *Check if event time is larger than other event time*

Description

Function which takes as input a vector a with event times and a vector b with event times and checks whether each entry in a is larger than the entries in b.

Usage

```
agreaterb(a, b)
```

Arguments

a	Vector of event times
b	Vector of event times

Value

Matrix of size (length(a) * length(b)) with binary values indicating whether the event times in a are larger than the event times in b

ainB *Check if event time is contained within half-open interval*

Description

Function which takes as input a vector a with event times and a 2 column matrix B representing half-open intervals (l, R] and checks whether each event time is contained in each half-open interval.

Usage

```
ainB(a, B)
```

Arguments

a	Vector of event times
B	Two column matrix containing intervals

Value

Matrix of size (length(a) * nrow(B)) with binary values indicating whether the event times in a are contained in the intervals of B

Aintersectb *Check if half-open intervals intersect with event times*

Description

Function which takes as input a 2 column matrix of half-open intervals and a vector of event times and returns the event times that intersect with the half-open intervals.

Usage

```
Aintersectb(A, b, A.left.open = FALSE)
```

Arguments

A	Two column matrix containing intervals
b	Vector of event times
A.left.open	Are the intervals in A open on the left side? Default = FALSE.

Value

Numeric vector of event times from b that intersect with A.

Alargerb *Check if closed interval is contained in half-open infinite interval*

Description

Function which takes as input a matrix with 2 columns and a vector indicating left points of intervals [b, Infinity) and checks whether each interval in the matrix is contained within the corresponding interval derived from b.

Usage

```
Alargerb(A, b)
```

Arguments

A	Two column matrix containing intervals to be checked for being contained in b
b	Vector indicating left point in corresponding [b, Infinity) interval

Value

Matrix of size (nrow(A) * length(b)) with binary values indicating whether the intervals in A are contained in the ones induced by b

AsubsetB *Check if closed interval is contained in other closed interval*

Description

Function which takes as input two matrices with 2 columns each and checks whether each interval in the first matrix is contained within each interval in the second matrix.

Usage

```
AsubsetB(A, B, B.left.open = FALSE, B.right.open = FALSE)
```

Arguments

A	Two column matrix containing intervals to be checked for being contained in B
B	Two column matrix containing intervals possibly overlapping the intervals in A
B.left.open	Are the intervals in B left-open?
B.right.open	Are the intervals in B right-open?

Value

Matrix of size (nrow(A) * nrow(B)) with binary values indicating whether the intervals in A are contained in B

direct_from_observed_intervals *Translate observed transition intervals into direct transition intervals*

Description

Given observed transition intervals, determine the "worst" (least informative) possible direct transition intervals that could have occurred to form this sample.

Usage

```
direct_from_observed_intervals(observed_intervals, tmat, gd)
```

Arguments

observed_intervals	Output from get_trans_intervals.
tmat	A transition matrix as created by transMat
gd	A data.frame with the following named columns
	id: Subject idenitifier;

`state`: State at which the subject is observed at `time`;

`time`: Time at which the subject is observed;

The true transition time between states is then interval censored between the times.

Value

A `data.frame` with the following named columns

`entry_time`: Time of entry into "from" state;

`time_from`: Last time subject(id) was seen in state "from";

`time_to`: First time subject(id) was seen in state "to";

`from`: State from which a transition was observed;

`to`: State to which the transition was observed;

`id`: Subject identifier;

For right-censored observations, entry_time denotes the first time seen in the censored state, time_from the last time seen in the censored state, time_to is `Inf`, from the censored state and to is `NA`.

---

estimate_support_msm *Estimate the support of a general Markov interval-censored Multi-state model without loops.*

Description

Given a realisation of a multi-state model, estimate the support of the different transitions possible in that MSM. The estimation is performed by viewing each possible state in a competing risks setting and applying the result of Hudgens (2001) to determine the support and left-truncation intervals and Hudgens (2005) to check whether a solution is possible.

Usage

```
estimate_support_msm(gd, tmat)
```

Arguments

`gd`	A `data.frame` with the following named columns `id`: Subject idenitifier; `state`: State at which the subject is observed at `time`; `time`: Time at which the subject is observed; The true transition time between states is then interval censored between the times.
`tmat`	A transition matrix as created by `transMat`

Value

TODO

References

Michael G. Hudgens, On Nonparametric Maximum Likelihood Estimation with Interval Censoring and Left Truncation, Journal of the Royal Statistical Society Series B: Statistical Methodology, Volume 67, Issue 4, September 2005, Pages 573-587, doi:10.1111/j.14679868.2005.00516.x

M. G. Hudgens, G. A. Satten, and I. M. Longini, Nonparametric Maximum Likelihood Estimation for Competing Risks Survival Data Subject to Interval Censoring and Truncation, Biometrics, vol. 57, no. 1, Pages 74-80, March 2001, doi:10.1111/j.0006341x.2001.00074.x

---

`evalstep` *Sample from a markov chain multi state model with exactly observed transition times*

Description

Given a markov chain multi state model with exactly observed transition times, sample from this chain at the observation times, giving interval censored observations (panel data).

Usage

```
evalstep(time, stepf, newtime, subst = -Inf, to.data.frame = FALSE)
```

Arguments

`time`	Times at which a transition occurs
`stepf`	States at which the chain is in at `times`
`newtime`	Observation times of the chain, to create observed states
`subst`	State to return if observation time is before first transition time. Default = -Inf.
`to.data.frame`	Should the result be returned as a `data.frame`?

Value

A numeric `vector` or `data.frame` (if `to.data.frame = TRUE`) containing either the observed states or the named columns `newtime` and `res`, representing the observation times and observed states.

Author(s)

Hein Putter

Examples

```
obs_states <- evalstep(time = seq(0, 20, 2), stepf = sample(1:9, 11, replace = TRUE),
                newtime = c(-1, 1, 7, 9, 19))
obs_states
```

extend_msfit *Given a* msfit *object, extend the times considered in the object*

Description

After using this function, use probtrans to get interpolated transition probabilities. This function is useful when you want to obtain transition probabilities at more than just the minimal number of times that strictly have to be considered. The inserted hazard values are simply the hazards at the nearest time that is smaller or equal.

Usage

```
extend_msfit(msfit, times)
```

Arguments

msfit	A msfit object.
times	Times at which to extend the msfit object.

Value

An msfit object containing the extended hazards

Examples

```
library(mstate)
tmat <- trans.illdeath()
times <- seq(0, 5, 0.1)
ms_fit <- list(Haz = data.frame(time = rep(times, 3),
                                Haz = c(replicate(3, cumsum(runif(length(times), 0, 0.02)))),
                                     trans = rep(1:3, each = length(times))),
               trans = tmat)
class(ms_fit) <- "msfit"

ms_fit_interpolated <- extend_msfit(ms_fit, seq(0, 5, 0.01))
```

get_trans_intervals *Get transition intervals from specified data*

Description

Given a sample from a multi-state model, summarize the transitions that have been observed.

Usage

```
get_trans_intervals(gd, tmat)
```

Arguments

gd	A data.frame with the following named columns id: Subject idenitifier; state: State at which the subject is observed at time; time: Time at which the subject is observed; The true transition time between states is then interval censored between the times.
tmat	A transition matrix as created by transMat

Value

A data.frame with the following named columns

entry_time: Time of entry into "from" state;

time_from: Last time subject(id) was seen in state "from";

time_to: First time subject(id) was seen in state "to";

from: State from which a transition was observed;

to: State to which the transition was observed;

id: Subject identifier;

For right-censored observations, entry_time denotes the first time seen in the censored state, time_from the last time seen in the censored state, time_to is Inf, from the censored state and to is NA.

---

graphfromIntervals *Construct Graph from censoring/truncation intervals*

Description

Given intervals, construct a graph containing vertices representing these intervals and edges between the vertices if the intervals intersect. See Hudgens (2005).

Usage

```
graphfromIntervals(intervals)
```

Arguments

intervals	A data.frame with 3 columns containing half-open intervals (left open, right closed) and an indicator whether the interval results from a censored transition or truncation: #' L: Left side of interval; R: Right side of interval; cens: Indicator whether interval resulted from censoring or truncation (1 = censoring, 0 = truncation); Note that the truncation intervals need to be in the form (N, Inf] with N a numeric value.

Value

Returns an 'igraph' object containing the graph with vertices representing the intervals and edges between the vertices if the intervals intersect. The vertices will be named accordingly, starting with a 'T' when representing a truncation interval and 'C' when representing a censoring interval.

References

Michael G. Hudgens, On Nonparametric Maximum Likelihood Estimation with Interval Censoring and Left Truncation, Journal of the Royal Statistical Society Series B: Statistical Methodology, Volume 67, Issue 4, September 2005, Pages 573-587, doi:10.1111/j.14679868.2005.00516.x

---

`interpol_msfit` *Given a* `msfit` *object, linearly interpolate the cumulative hazard taking into account the support sets for* `msfit` *objects.*

Description

After using this function, use probtrans to get interpolated transition probabilities.

Usage

```
interpol_msfit(msfit, times)
```

Arguments

`msfit`	A `msfit` object.
`times`	Times at which to interpolate the `msfit` object.

Value

An `msfit` object containing the interpolated hazards

Examples

```
library(mstate)
tmat <- trans.illdeath()
times <- seq(0, 5, 0.1)
ms_fit <- list(Haz = data.frame(time = rep(times, 3),
                                Haz = c(replicate(3, cumsum(runif(length(times), 0, 0.02)))),
                                    trans = rep(1:3, each = length(times))),
               trans = tmat)
class(ms_fit) <- "msfit"

ms_fit_interpolated <- interpol_msfit(ms_fit, seq(0, 5, 0.01))
```

`msm_frydman` *Determine NPMLE for Multi State illness death Markov model using Frydman (1995)*

Description

Determine NPMLE for Multi State illness death Markov model using Frydman (1995)

Usage

```
msm_frydman(data, tol = 1e-08)
```

Arguments

`data`	A `data.frame` containing the columns named: `delta`: Did a transition from 1 -> 2 occur? (binary: 0 = no, 1 = yes); In the left-truncated case, delta = 2 indicates initially observed in state 2. `Delta`: Was the transition to state 3 observed? (binary: 0 = no, 1 = yes); `L`: Left timepoint of interval censored transition to state 2 (numeric); `R`: Right timepoint of interval censored transition to state 2 (numeric); `time`: Time of event (transition to 3) or right-censoring in state 2 (numeric); `trunc`: (optional) Left-truncation time (numeric); Only used for entries with delta = 2.
`tol`	Tolerance of the EM algorithm. Algorithm will stop when the absolute difference between current mass estimates and new estimates is smaller than the tolerance

Details

For an illness death model (1 = healthy, 2 = ill, 3 = dead) estimate the NPMLE in the following form:

`F12`: Cumulative distribution function of 1->2 transition;

`F13`: Cumulative distribution function of 1->3 transition;

`Lambda23`: Cumulative intensity of 2->3 transition;

Value

A list with the following entries:

`data_idx`: A list containing the data used for the fit (`matdata`), the indices for which group a subject belongs to (`GroupX_idx`), some computational parameters (see Frydman(1995)) and the unique failure times of the 2->3 and 1->3 transitions respectively in `t_n_star` and `e_k_star`;

`supportMSM`: A list containing all transition intervals in `A` and the theoretical support intervals in `Q_mat`;

`z_lambda`: Computational quantities, see Frydman(1995);

`cdf`: A list of functions that allow to recover the cdf for the 1->3 (`F13`) and 1->2 (`F12`) transition and the cumulative hazard for the 2->3 (`Lambda23`) transition.;

References

Frydman, H. (1995). Nonparametric Estimation of a Markov 'Illness-Death' Process from Interval-Censored Observations, with Application to Diabetes Survival Data. Biometrika, 82(4), 773-789. doi:10.2307/2337344

Examples

```
data <- data.frame(delta = c(0, 0, 1, 1), Delta = c(0, 1, 0, 1),
                   L = c(NA, NA, 1, 1.5), R = c(NA, 3, 2, 3),
                   time = c(4, 5, 6, 7))

mod_frydman <- msm_frydman(data)
visualise_data(data, mod_frydman)
```

`npmsm` *NPMLE for general multi-state model with interval censored transitions*

Description

For a general Markov chain multi-state model with interval censored transitions calculate the NPMLE of the transition intensities. The estimates are returned as an `msfit` object.

Usage

```
npmsm(
  gd,
  tmat,
  method = c("multinomial", "poisson"),
  inits = c("equalprob", "homogeneous", "unif", "beta"),
  beta_params,
  support_manual,
  exact,
  maxit = 100,
  tol = 1e-04,
  conv_crit = c("haz", "prob", "lik"),
  verbose = FALSE,
  manual = FALSE,
  newmet = FALSE,
  include_inf = FALSE,
  checkMLE = TRUE,
  checkMLE_tol = 1e-04,
  prob_tol = tol/10,
  remove_redundant = TRUE,
  remove_bins = FALSE,
```

```
  estimateSupport = FALSE,
  init_int = c(0, 0),
  ...
)
```

Arguments

gd	A data.frame with the following named columns id: Subject idenitifier; state: State at which the subject is observed at time; time: Time at which the subject is observed; The true transition time between states is then interval censored between the times.
tmat	A transition matrix as created by transMat
method	Which method should be used for the EM algorithm. Choices are c("multinomial", "poisson"), with multinomial the default. Multinomial will use the EM algorithm described in Gomon and Putter (2024) and Poisson will use the EM algorithm described in Gu et al. (2023).
inits	Which distribution should be used to generate the initial estimates of the intensities in the EM algorithm. One of c("equalprob", "unif", "beta"), with "equalprob" assigning 1/K to each intensity, with K the number of distinct observation times (length(unique(gd[, "time"]))). For "unif", each intensity is sampled from the Unif[0,1] distribution and for "beta" each intensity is sampled from the Beta(a, b) distribution. If "beta" is chosen, the argument beta_params must be specified as a vector of length 2 containing the parameters of the beta distribution. Default = "equalprob".
beta_params	A vector of length 2 specifying the beta distribution parameters for initial distribution generation. First entry will be used as shape1 and second entry as shape2. See help(rbeta). Only used if inits = "beta".
support_manual	Used for specifying a manual support region for the transitions. A list of length the number of transitions in tmat, each list element containing a data frame with 2 named columns L and R indicating the left and right values of the support intervals. When specified, all intensities outside of these intervals will be set to zero for the corresponding transitions. Intensities set to zero cannot be changed by the EM algorithm. Will use inits = "equalprob".
exact	Numeric vector indicating to which states transitions are observed at exact times. Must coincide with the column number in tmat.
maxit	Maximum number of iterations. Default = 100.
tol	Tolerance of the convergence procedure. A change in the value of conv_crit in an iteration of less than tol will make the procedure stop.
conv_crit	Convergence criterion. Stops procedure when the difference in the chosen quantity between two consecutive iterations is smaller than the tolerance level tol. One of the following: **"haz"** Stop when change in maximum estimated intensities (hazards) < tol. **"prob"** Stop when change in estimated probabilities < tol.

"lik" Stop when change in observed-data likelihood < `tol`.

Default is "haz". The options "haz" and "lik" can be compared across different `methods`, but "prob" is dependent on the chosen `method`. Most conservative (requiring most iterations) is "prob", followed by "haz" and finally "lik".

`verbose` Should iteration messages be printed? Default is FALSE

`manual` Manually specify starting transition intensities? If `TRUE`, the transition intensity for each bin for each transition must be entered manually. DO NOT USE for large data sets, and in general it is not adviced to use this.

`newmet` Should contributions after last observation time also be used in the likelihood? Default is FALSE.

`include_inf` Should an additional bin from the largest observed time to infinity be included in the algorithm? Default is FALSE.

`checkMLE` Should a check be performed whether the estimate has converged towards a local maximum? This is done by comparing the reduced gradient to the value of `checkMLE_tol`. Default is TRUE.

`checkMLE_tol` Tolerance for checking whether the estimate has converged to local maximum. Whenever an estimated transition intensity is smaller than `prob_tol`, it is assumed to be zero and its reduced gradient is not considered for determining whether the maximum has been reached. Default = `1e-4`.

`prob_tol` If an estimated probability is smaller than `prob_tol`, it will be set to zero during estimation. Default value is `tol/10`.

`remove_redundant`
Should redundant observations be removed before running the algorithm? An observation is redundant when the same state has been observed more than 3 times consecutively, or if it is a repeat observation of an absorbing state. Default is TRUE.

`remove_bins` Should a bin be removed during the algorithm if all estimated intensities are zero for a single bin? Can improve computation speed for large data sets. Note that zero means the estimated intensities are smaller than `prob_tol`. Default is FALSE.

`estimateSupport`
Should the support of the transitions be estimated using the result of Hudgens (2005)? Currently produces incorrect support sets - DO NOT USE. Default = `FALSE`

`init_int` A vector of length 2, with the first entry indicating what percentage of mass should be distributed over (second entry) what percentage of all first bins. Default is c(0, 0), in which case the argument is ignored. This argument has no practical uses and only exists for demonstration purposes in the related article.

`...` Further arguments to `estimate_support_msm`

Details

Denote the unique observation times in the data as $0 = \tau_0, \tau_1, \ldots, \tau_K$ Let $g, h \in H$ denote the possible states in the model and $X(t)$ the state of the process at time t.

Then this function can be used to estimate the transition intensities $\alpha_{gh}^k = \alpha_{gh}(\tau_k)$.

Having obtained these estimated, it is possible to recover the transition probabilities $\mathbf{P}(X(t) = h|X(s) = g)$ for $t > s$ using the `transprob` functions.

Value

A list with the following entries:

- `A:` A list of class `msfit` containing the cumulative intensities for each transition and the transition matrix used;
- `Ainit:` Initial intensities, in an object of class `msfit`;
- `gd:` Data used for the estimation procedure;
- `ll:` Log-likelihood value of the procedure at the last iteration;
- `delta:` Change in log-likelihood value at the last iteration;
- `it:` Number of iterations of the procedure;
- `taus:` Unique time points of the data, the cumulative intensity only makes jumps at these time points.;
- `tmat:` The transition matrix used, see `transMat`;
- `tmat2:` A summary of the transitions in the model, see `to.trans2`;
- `ll_history:` The log-likelihood value at every iteration of the procedure;
- `KKT_violated:` How often were KKT conditions violated during maximisation of the likelihood? In other words, how often did we hit the optimization boundary during the procedure?;
- `min_time:` The smallest time of an observation in the used data. Note that the smallest time in the data is used as zero reference;
- `reduced_gradient:` The reduced gradient at the last iteration. Rows indicate the transitions and columns the unique observation times;
- `isMLE:` Has the procedure converged to the NPMLE? Checked using `checkMLE_tol`;
- `langrangemultiplier:` The lagrange multipliers at the last iteration;
- `aghmat:` A matrix representation of the transition intensities in `A`. Rows represent transitions and columns unique observation times;
- `Ygk:` The summed at-risk indicator for all subjects in the data at the last iteration. Rows represent transitions and columns unique observation times;
- `Dmk:` The summed probability of making a transition for all subjects at the last iteration. Rows represent transitions and columns unique observation times;
- `method:` Method used for the optimization procedure;
- `maxit:` Maximum number of allowed iterations;
- `tol:` Tolerance of the convergence procedure;
- `conv_crit:` Convergence criterion of the procedure;
- `checkMLE:` Was the reduced gradient checked at the last iteration to determine convergence?;

`checkMLE_tol:` The tolerance of the checkMLE procedure;

`prob_tol:` Tolerance for probabilities to be set to zero;

`remove_redundant:` Were redundant observations removed before performing the procedure?;

References

D. Gomon and H. Putter, Non-parametric estimation of transition intensities in interval censored Markov multi-state models without loops, arXiv, 2024, doi:10.48550/arXiv.2409.07176

Y. Gu, D. Zeng, G. Heiss, and D. Y. Lin, Maximum likelihood estimation for semiparametric regression models with interval-censored multistate data, Biometrika, Nov. 2023, doi:10.1093/biomet/asad073

Michael G. Hudgens, On Nonparametric Maximum Likelihood Estimation with Interval Censoring and Left Truncation, Journal of the Royal Statistical Society Series B: Statistical Methodology, Volume 67, Issue 4, September 2005, Pages 573-587, doi:10.1111/j.14679868.2005.00516.x

See Also

`transprob` for calculating transition probabilities, `plot.npmsm` for plotting the cumulative intensities, `print.npmsm` for printing some output summaries, `visualise_msm` for visualising data, `msfit` for details on the output object.

Examples

```
#Create transition matrix using mstate functionality: illness-death
if(require(mstate)){
  tmat <- mstate::trans.illdeath()
}

#Write a function for evaluation times: observe at 0 and uniform inter-observation times.
eval_times <- function(n_obs, stop_time){
  cumsum( c( 0,  runif( n_obs-1, 0, 2*(stop_time-4)/(n_obs-1) ) ) )
}

#Use built_in function to simulate illness-death data
#from Weibull distributions for each transition
sim_dat <- sim_id_weib(n = 50, n_obs = 6, stop_time = 15, eval_times = eval_times,
start_state = "stable", shape = c(0.5, 0.5, 2), scale = c(5, 10, 10/gamma(1.5)))

tmat <- mstate::trans.illdeath()

#Fit the model using method = "multinomial"
mod_fit <- npmsm(gd = sim_dat, tmat = tmat, tol = 1e-2)

#Plot the cumulative intensities for each transition
plot(mod_fit)
```

`plot.npmsm` *Plot a "npmsm" object*

Description

Plot the cumulative intensities of a `'npmsm'` objects. A wrapper for `plot.msfit` from the `mstate` package.

Usage

```
## S3 method for class 'npmsm'
plot(x, ...)
```

Arguments

`x`	An object of class `"npmsm"`
`...`	Additional arguments to `msfit`

Value

A plot will be produced in the plotting window

`plot.probtrans.subjects`
Plot an object of class `"probtrans.subjects"`

Description

Plots the transition probabilities for a specific subject. Wrapper for `plot.probtrans`

Usage

```
## S3 method for class 'probtrans.subjects'
plot(x, id, ...)
```

Arguments

`x`	An object of class `"probtrans.subjects"`
`id`	Subject identifier
`...`	Further arguments to `plot.probtrans`

Details

Note that

Author(s)

Hein Putter and Daniel Gomon

Examples

```
if(require("mstate")){
  data(ebmt3)
  n <- nrow(ebmt3)
  tmat <- transMat(x = list(c(2, 3), c(3), c()), names = c("Tx",
                                                           "PR", "RelDeath"))
  ebmt3$prtime <- ebmt3$prtime/365.25
  ebmt3$rfstime <- ebmt3$rfstime/365.25
  covs <- c("dissub", "age", "drmatch", "tcd", "prtime")
  msbmt <- msprep(time = c(NA, "prtime", "rfstime"), status = c(NA,
                  "prstat", "rfsstat"), data = ebmt3, trans = tmat, keep = covs)
  #Expand covariates so that we can have transition specific covariates
  msbmt <- expand.covs(msbmt, covs, append = TRUE, longnames = FALSE)

  #Simple model, transition specific covariates, each transition own baseline hazard
  c1 <- coxph(Surv(Tstart, Tstop, status) ~ dissub1.1 + dissub2.1 +
                age1.1 + age2.1 + drmatch.1 + tcd.1 + dissub1.2 + dissub2.2 +
                age1.2 + age2.2 + drmatch.2 + tcd.2 + dissub1.3 + dissub2.3 +
                age1.3 + age2.3 + drmatch.3 + tcd.3 + strata(trans), data = msbmt,
                method = "breslow")
  #We need to make a data.frame containing all subjects of interest
  ttmat <- to.trans2(tmat)[, c(2, 3, 1)]
  names(ttmat)[3] <- "trans"
  nd_n <- NULL
  for (j in 1:30) {
    # Select global covariates of subject j
    cllj <- ebmt3[j, covs]
    nd2 <- cbind(ttmat, rep(j, 3), rbind(cllj, cllj, cllj))
    colnames(nd2)[4] <- "id"
    # Make nd2 of class msdata to use expand.covs
    attr(nd2, "trans") <- tmat
    class(nd2) <- c("msdata", "data.frame")
    nd2 <- expand.covs(nd2, covs=covs, longnames = FALSE)
    nd_n <- rbind(nd_n, nd2)
   }

   icmstate_pt <- probtrans_coxph(c1, predt = 0, direction = "forward",
                                  newdata = nd_n, trans = tmat)

   #plot transition probabilities for subject 2
   plot(icmstate_pt, id = 2)
}
```

`plot.smoothmsm` *Plot a "smoothmsm" object*

Description

Plot the cumulative intensities of a 'npmsm' objects. A wrapper for plot.msfit from the mstate package.

Usage

```
## S3 method for class 'smoothmsm'
plot(x, ...)
```

Arguments

x	An object of class "smoothmsm"
...	Additional arguments to plot

Value

A plot will be produced in the plotting window

---

plot_probtrans *Plot the transition probabilities for a fitted* npmsm *model*

Description

For a fitted npmsm model plot the transition probabilities from a certain state for all possible (direct and indirect) transitions.

Usage

```
plot_probtrans(
  npmsmlist,
  from = NULL,
  to = NULL,
  transitions = NULL,
  landmark,
  interpolate = TRUE,
  facet = TRUE,
  times_interpol = NULL,
  c.legend = TRUE,
  c.names = NULL
)
```

Arguments

npmsmlist	An "npmsm" object or a list containing multiple "npmsm" objects
from	A numeric value indicating the state from which we consider the transition probabilities. Default is NULL, meaning we consider transition probabilities from all states from which a direct transition is possible.
to	A numeric vector indicating to which states we consider the transition probabilities. Only states that can be reached from the "from" state are considered.
transitions	A numeric vector indicating which transitions to consider (plot). Can only be used if "from" is not specified, as it only works for direct transitions.
landmark	A landmark time indicating from which time on survival should be determined. If missing, the smallest between the time in the first "npmsm" object or 0 will be used.
interpolate	Should the cumulative hazard be linearly interpolated before determining transition probabilities? Default is TRUE.
facet	Should the resulting plot be faceted (one panel per transition)? Default is TRUE.
times_interpol	At which times should the cumulative hazard be interpolated? Only necessary to specify if interpolate = TRUE.
c.legend	Should legend be displayed for colour (different entries in npmsmlist)? Default is TRUE.
c.names	A character vector indicating the names to display in the legend. These names should represent the entries in npmsmlist.. Default = NULL.

Value

A plot will be produced in the plotting window. When assigning the output to an object, the underlying data frame used for plotting and a 'ggplot' object will be returned in a list.

Examples

```
require(mstate)
require(ggplot2)
#Generate from an illness-death model with exponential transitions with
#rates 1/2, 1/10 and 1 for 10 subjects over a time grid.
gd <- sim_weibmsm(tmat = trans.illdeath(), shape = c(1,1,1),
                  scale = c(2, 10, 1), n_subj = 10, obs_pars = c(2, 0.5, 20),
                  startprobs = c(0.9, 0.1, 0))
#Fit 2 models: 1 with at most 4 iterations and 1 with at most 20
mod1 <- npmsm(gd, trans.illdeath(), maxit = 4)
mod2 <- npmsm(gd, trans.illdeath(), maxit = 20)

#Plot the transition probabilities from state 1, without interpolating
#the cumulative hazard for the npmsm runs with max 4 and 20 iterations.
plot_probtrans(list(mod1, mod2), from = 1, interpolate = FALSE,
               c.names = c("4 iterations", "20 iterations"))
```

plot_surv *Plot the transition specific survival probabilities for a fitted* npmsm *model*

Description

For a fitted npmsm model plot the transition specific survival probabilities. These are given by the product integral of the hazard increments estimated for a single transition. This is equivalent to a Kaplan-Meier estimator ignoring the existence of all other transitions.

Usage

```
plot_surv(npmsmlist, landmark, support = FALSE, sup_cutoff = 1e-08)
```

Arguments

npmsmlist	An "npmsm" object or a list containing multiple "npmsm" objects
landmark	A landmark time indicating from which time on survival should be determined. If missing, the smallest time in the first "npmsm" object will be used.
support	Should the support regions be displayed as rectangles?
sup_cutoff	Cutoff to be used for determining the support intervals.

Value

A plot will be produced in the plotting window. When assigning the output to an object, the underlying data frame used for plotting and a 'ggplot' object will be returned in a list.

Examples

```
require(mstate)
require(ggplot2)
#Generate from an illness-death model with exponential transitions with
#rates 1/2, 1/10 and 1 for 10 subjects over a time grid.
gd <- sim_weibmsm(tmat = trans.illdeath(), shape = c(1,1,1),
                  scale = c(2, 10, 1), n_subj = 10, obs_pars = c(2, 0.5, 20),
                  startprobs = c(0.9, 0.1, 0))
mod1 <- npmsm(gd, trans.illdeath(), maxit = 4)
mod2 <- npmsm(gd, trans.illdeath(), maxit = 20)

#Plot the transition specific Kaplan-Meier estimators and their numerically
#determined support sets.
plot_surv(list(mod1, mod2), support = TRUE)
```

predict_tp *Calculate subject specific transition probabilities from a multi-state proportional hazards model.*

Description

Given a coxph model fit on multi-state data (prepared with msprep), determine transition probabilities for subjects in newdata.

Usage

```
predict_tp(
  object,
  predt,
  direction = c("forward", "fixedhorizon"),
  newdata,
  trans
)
```

Arguments

object	A coxph object fit on multi-state data. Must contain a strata(X) term. Data used for the coxph() fit preferably prepared using msprep.
predt	A positive number indicating the prediction time. This is either the time at which the prediction is made (if direction = "forward") or the time for which the prediction is to be made (if direction = "backward").
direction	One of "forward" (default) or "fixedhorizon", indicating whether prediction is forward or for a fixed horizon
newdata	A data.frame containing a single row per subject in the data. It must contain the following named columns: id: (optional) Unique identifier of the subject, can be numeric or character; "variables": The variables and their values for the subject(s) (optionally identified by "id"). After using expand.covs, the names of the new data must match the names of the variables in the coxph object. Note that newdata must contain a column containing the variable which was used to determine the stratum of a transition in object. Usually the stratum is determined from a column that is generated automatically in the pre-processing steps of this function.
trans	A transition matrix as created by transMat.

Details

When using this function for newdata with many subjects, consider running the function multiple times for parts of newdata to negate the risk of running our of memory. Using this function, it is only possible to consider models with transition specific covariates. If you would like to have covariates shared over transitions or proportional hazards assumptions between transitions, see probtrans_coxph.

Value

An object of class "probtrans.subjects". This is a list of length n (number of subjects in newdata), with each list element an object of class probtrans for the associated subject. List elements can be accessed using [[x]], with x ranging from 1 to n. Additionally, each list element has an element $id, representing the subject id and the output object also has an element $subject_ids representing the subject ids in order.

See Also

probtrans_coxph, plot.probtrans.subjects, summary.probtrans.subjects

Examples

```
#Example from the mstate vignette
#We determine the subject specific transition probabilities for subjects
#in the ebmt3 data-set
if(require("mstate")){
  data(ebmt3)
  n <- nrow(ebmt3)
  tmat <- transMat(x = list(c(2, 3), c(3), c()), names = c("Tx",
                                                           "PR", "RelDeath"))
  #From days to years
  ebmt3$prtime <- ebmt3$prtime/365.25
  ebmt3$rfstime <- ebmt3$rfstime/365.25
  #Covariates we will use
  covs <- c("dissub", "age", "drmatch", "tcd", "prtime")
  msbmt <- msprep(time = c(NA, "prtime", "rfstime"), status = c(NA,
                  "prstat", "rfsstat"), data = ebmt3, trans = tmat, keep = covs)
  #Expand covariates so that we can have transition specific covariates
  msbmt2 <- expand.covs(msbmt, covs, append = TRUE, longnames = FALSE)

  #-------------Model--------------------#

  #Simple model, transition specific covariates, each transition own baseline hazard
  c1 <- coxph(Surv(Tstart, Tstop, status) ~ dissub1.1 + dissub2.1 +
                age1.1 + age2.1 + drmatch.1 + tcd.1 + dissub1.2 + dissub2.2 +
                age1.2 + age2.2 + drmatch.2 + tcd.2 + dissub1.3 + dissub2.3 +
                age1.3 + age2.3 + drmatch.3 + tcd.3 + strata(trans), data = msbmt2,
                method = "breslow")

   #Predict transition probabilities for first 30 subjects.
   tp_subjects <- predict_tp(c1, predt = 0, direction = "forward",
                                  newdata = ebmt3[1:30,], trans = tmat)

   #Now we can plot the transition probabilities for each subject separately:
   plot(tp_subjects, id = 1)
   #tp_subjects has length number of subjects in newdata + 1
   #And tp_subjects[[i]] is an object of class "probtrans", so you can
   #use all probtrans functions: summary, plot etc.
}
```

print.npmsm *Print a "npmsm" object*

Description

Print some details of a npmsm fit

Usage

```
## S3 method for class 'npmsm'
print(x, ...)
```

Arguments

x	An object of class "npmsm"
...	Additional arguments to print

Value

A summary of the fitted model will be displayed in the console

print.summary.probtrans.subjects *Print method for a summary.probtrans.subjects object*

Description

Print method for a summary.probtrans.subjects object

Usage

```
## S3 method for class 'summary.probtrans.subjects'
print(x, complete = FALSE, ...)
```

Arguments

x	Object of class 'summary.probtrans.subjects', to be printed
complete	Whether or not the complete estimated transition probabilities should be printed (TRUE) or not (FALSE); default is FALSE, in which case the estimated transition probilities will be printed for the first and last 6 time points of each starting state or of the selected times (or all when there are at most 12 of these time points
...	Further arguments to print

Examples

```
## Not run:
# If all time points should be printed, specify complete=TRUE in the print statement
print(x, complete=TRUE)

## End(Not run)
```

`probtrans_coxph` *Calculate subject specific transition probabilities from a multi-state* `coxph` *model.*

Description

Given a coxph model fit on multi-state data (prepared with `msprep`), determine transition probabilities for subjects in `newdata`.

Usage

```
probtrans_coxph(
  object,
  predt,
  direction = c("forward", "fixedhorizon"),
  newdata,
  trans
)
```

Arguments

`object`	A `coxph` object fit on multi-state data. Must contain a `strata(X)` term. Data used for the coxph() fit preferably prepared using `msprep`.
`predt`	A positive number indicating the prediction time. This is either the time at which the prediction is made (if `direction = "forward"`) or the time for which the prediction is to be made (if `direction = "backward"`).
`direction`	One of `"forward"` (default) or `"fixedhorizon"`, indicating whether prediction is forward or for a fixed horizon
`newdata`	A `data.frame` containing a single row for each transition per subject in the data. For a model with m possible transitions, and n subjects `newdata` must have m*n rows. It must contain the following named columns: `id`: Unique identifier of the subject, can be numeric or character; `from`: State from which the transition takes place; `to`: State to which the transition takes place; `trans`: Transition number in the `'transMat'` `trans` this transition relates to; `"variables"`: The variables and their values for the subject identified by "id" for the transition this entry relates to. Names must match the names of the variables in coxph `object`.

Note that newdata must contain a column containing the variable which was used to determine the stratum of a transition in `object`. Usually the stratum is determined from one of the required columns. The "variables" columns can usually be obtained using `expand.covs`.

`trans` A transition matrix as created by `transMat`.

Details

When using this function for `newdata` with many subjects, consider running the function multiple times for parts of `newdata` to negate the risk of running our of memory.

Value

An object of class `"probtrans.subjects"`. This is a list of length n (number of subjects in newdata), with each list element an object of class `probtrans` for the associated subject. List elements can be accessed using `[[x]]`, with `x` ranging from 1 to n. Additionally, each list element has an element `$id`, representing the subject id and the output object also has an element `$subject_ids` representing the subject ids in order.

Examples

```
#Example from the mstate vignette
#We determine the subject specific transition probabilities for subjects
#in the ebmt3 data-set
if(require("mstate")){
  data(ebmt3)
  n <- nrow(ebmt3)
  tmat <- transMat(x = list(c(2, 3), c(3), c()), names = c("Tx",
                                                           "PR", "RelDeath"))
  ebmt3$prtime <- ebmt3$prtime/365.25
  ebmt3$rfstime <- ebmt3$rfstime/365.25
  covs <- c("dissub", "age", "drmatch", "tcd", "prtime")
  msbmt <- msprep(time = c(NA, "prtime", "rfstime"), status = c(NA,
                  "prstat", "rfsstat"), data = ebmt3, trans = tmat, keep = covs)
  #Expand covariates so that we can have transition specific covariates
  msbmt <- expand.covs(msbmt, covs, append = TRUE, longnames = FALSE)

  #Create extra variable to allow gender mismatch to have the same effect
  #for transitions 2 and 3.
  msbmt$drmatch.2.3 <- msbmt$drmatch.2 + msbmt$drmatch.3

  #Introduce pr covariate for proportionality assumption of transitions 2 and 3
  #(only used in models 2 and 4)
  msbmt$pr <- 0
  msbmt$pr[msbmt$trans == 3] <- 1



  #-------------Models--------------------#

  #Simple model, transition specific covariates, each transition own baseline hazard
```

```
c1 <- coxph(Surv(Tstart, Tstop, status) ~ dissub1.1 + dissub2.1 +
              age1.1 + age2.1 + drmatch.1 + tcd.1 + dissub1.2 + dissub2.2 +
              age1.2 + age2.2 + drmatch.2 + tcd.2 + dissub1.3 + dissub2.3 +
              age1.3 + age2.3 + drmatch.3 + tcd.3 + strata(trans), data = msbmt,
              method = "breslow")

#Model with same baseline hazards for transitions 2 (1->3) and 3(2->3)
#pr then gives the ratio of the 2 hazards for these transitions
c2 <- coxph(Surv(Tstart, Tstop, status) ~ dissub1.1 + dissub2.1 +
              age1.1 + age2.1 + drmatch.1 + tcd.1 + dissub1.2 + dissub2.2 +
              age1.2 + age2.2 + drmatch.2 + tcd.2 + dissub1.3 + dissub2.3 +
              age1.3 + age2.3 + drmatch.3 + tcd.3 + pr + strata(to), data = msbmt,
              method = "breslow")

#Same as c2, but now Gender mismatch has the same effect for both
c3 <- coxph(Surv(Tstart, Tstop, status) ~ dissub1.1 + dissub2.1 +
              age1.1 + age2.1 + drmatch.1 + tcd.1 + dissub1.2 + dissub2.2 +
              age1.2 + age2.2 + drmatch.2.3 + tcd.2 + dissub1.3 + dissub2.3 +
              age1.3 + age2.3  + tcd.3 + pr + strata(to), data = msbmt,
            method = "breslow")


#Predict for first 30 people in ebmt data
tmat2 <- to.trans2(tmat)[, c(2,3,1)]
names(tmat2)[3] <- "trans"
n_transitions <- nrow(tmat2)

newdata <- ebmt3[rep(seq_len(30), each = nrow(tmat2)),]
newdata <- cbind(tmat2[rep(seq_len(nrow(tmat2)), times = 30), ], newdata)
#Make of class "msdata"
attr(newdata, "trans") <- tmat
class(newdata) <- c("msdata", "data.frame")
#Covariate names to expand
covs <- names(newdata)[5:ncol(newdata)]
newdata <- expand.covs(newdata, covs, append = TRUE, longnames = FALSE)

newdata$drmatch.2.3 <- newdata$drmatch.2 + newdata$drmatch.3

newdata$pr <- 0
newdata$pr[newdata$trans == 3] <- 1

#Calculate transition probabilities for the Cox fits
icmstate_pt1 <- probtrans_coxph(c1, predt = 0, direction = "forward",
                                newdata = newdata, trans = tmat)
icmstate_pt2 <- probtrans_coxph(c2, predt = 0, direction = "forward",
                                newdata = newdata, trans = tmat)
icmstate_pt3 <- probtrans_coxph(c3, predt = 0, direction = "forward",
                                newdata = newdata, trans = tmat)

#Now we can plot the transition probabilities for each subject separately:
plot(icmstate_pt1[[1]])
#icmstate_pt has length number of subjects in newdata
#And icmstate_pt1[[i]] is an object of class "probtrans", so you can
```

```
  #use all probtrans functions: summary, plot etc.

  #Alternatively, use the plotting function directly:
  plot(icmstate_pt2, id = 2)
}
```

---

probtrans_weib *Determine transition probabilities for a multi-state model with Weibull hazards for the transitions.*

Description

Determine transition probabilities for a multi-state model with Weibull hazards for the transitions.

Usage

```
probtrans_weib(transMat, times, shapes, scales, type = c("prodint", "ODE"))
```

Arguments

transMat	A transition matrix as generated by `mstate::transMat` describing the possible transitions for the multi-state model.
times	The times at which the transition probabilities should be determined. Will always determine the probabilities forward in time starting from `min(times)`.
shapes	The Weibull shapes corresponding to the numbered transitions in `transMat`. See `?pweibull` for more info.
scales	The Weibull scales corresponding to the numbered transitions in `transMat`. See `?pweibull` for more info.
type	Should the transition probabilities be determined using product integration `"prodint"` or by solving the Kolmogorov forward ordinary differential equation `"ODE"`.

Value

An object containing the "true" transition probabilities for the specified Weibull hazards.

Examples

```
#Illness-death model
tmat <- mstate::trans.illdeath()
IDM <- probtrans_weib(tmat, seq(0, 15, 0.01), shapes = c(0.5, 0.5, 2),
                      scales = c(5, 10, 10/gamma(1.5)), type = "prodint")
IDM2 <- probtrans_weib(tmat, seq(0, 15, 0.01), shapes = c(0.5, 0.5, 2),
                       scales = c(5, 10, 10/gamma(1.5)), type = "ODE")
plot(IDM)
plot(IDM2)
```

```
#Extended illness-death model
tmat <- mstate::transMat(list(c(2, 3), c(4), c(), c()))
IDM <- probtrans_weib(tmat, seq(0, 15, 0.01), shapes = c(0.5, 0.5, 2),
                      scales = c(5, 10, 10/gamma(1.5)), type = "prodint")
IDM2 <- probtrans_weib(tmat, seq(0, 15, 0.01), shapes = c(0.5, 0.5, 2),
                       scales = c(5, 10, 10/gamma(1.5)), type = "ODE")
plot(IDM)
plot(IDM2)
```

`prod_lambda_G_base` *Calculate the product of intensities over interval decided by failure times*

Description

This function calculates $\prod_\lambda$ G as defined in Frydman (1995), with t_n the failure times. Note that length(t_n) must be equal to length(lambda)

Usage

```
prod_lambda_G_base(lambda, t_n, Q, A)
```

Arguments

`lambda`	Intensities of the 2->3 transition
`t_n`	Unique failure times, same length as lambda
`Q`	Matrix (2 column) containing support intervals as rows
`A`	Matrix (2 column) containing censoring intervals as rows

`remove_redundant_observations` *Remove redundant observations from supplied data frame*

Description

Remove redundant observed states from a supplied data frame. Observations are redundant either when we observe an absorbing state multiple times (as we cannot leave an absorbing state), or when a transient state is observed multiple times between transitions (as we cannot have loops, therefore no extra information is provided when we observe a transient state multiple times).

Usage

```
remove_redundant_observations(gd, tmat)
```

Arguments

gd	A data.frame with the following named columns id: Subject idenitifier; state: State at which the subject is observed at time; time: Time at which the subject is observed; The true transition time between states is then interval censored between the times.
tmat	A transition matrix as created by transMat

Value

A data.frame containing the information contained in the input data.frame gd, but without redundant observations. Depending on whether tmat was specified the function may remove more observations.

Examples

```
#We simulate some data
#Function to generate evaluation times: at 0 and uniform inter-observation
eval_times <- function(n_obs, stop_time){
 cumsum( c( 0,  runif( n_obs-1, 0, 2*(stop_time-4)/(n_obs-1) ) ) )
}

#Simulate illness-death model data with Weibull transitions
sim_dat <- sim_id_weib(n = 20, n_obs = 6, stop_time = 15, eval_times = eval_times,
                       start_state = "stable", shape = c(0.5, 0.5, 2),
                       scale = c(5, 10, 10/gamma(1.5)))
visualise_msm(sim_dat)
require(mstate)
sim_dat_clean <- remove_redundant_observations(sim_dat, trans.illdeath())
visualise_msm(sim_dat_clean)
```

---

sim_id_weib *Simulate panel data from an illness-death model with Weibull transition hazards*

Description

An illness-death model has 3 transitions:

1: State 1 (Healthy) to State 2 (Illness);

2: State 1 (Healthy) to State 3 (Death);

3: State 2 (Illness) to State 3 (Death);

Using this function, it is possible to simulate data from an illness-death model with Weibull transition intensities. Requires the use of an external (self-written) function to generate observation times.

Usage

```
sim_id_weib(
  n,
  n_obs,
  stop_time,
  eval_times,
  start_state = c("stable", "equalprob"),
  shape,
  scale,
  ...
)
```

Arguments

n	Number of subjects to generate paths for.
n_obs	Number of observations in time period for each subject.
stop_time	Largest time at which the model is considered.
eval_times	A function which returns the evaluation times for a subject. Must have as arguments at least n_obs and stop_time.
start_state	In which states can subjects start? Either everyone starts in state 1 ("stable") or equal probability to start in state 1 or 2 ("equalprob").
shape	Vector of shape parameters for the 3 transitions. See rweibull. The first entry will be used for the first transition and so on.
scale	Vector of scale parameters for the 3 transitions. See rweibull The first entry will be used for the first transition and so on.
...	Further parameters to eval_times function.

Details

Taking shape = 1 we get an exponential distribution with rate 1/scale

Value

Panel data in the form of a data.frame with 3 named columns id, time and state. These represent the subject identifier, the observation time and the state at the observation time.

Examples

```
#Function to generate evaluation times: at 0 and uniform inter-observation
eval_times <- function(n_obs, stop_time){
  cumsum( c( 0,  runif( n_obs-1, 0, 2*(stop_time-4)/(n_obs-1) ) ) )
}

#Simulate illness-death model data with Weibull transitions
sim_dat <- sim_id_weib(n = 20, n_obs = 6, stop_time = 15, eval_times = eval_times,
start_state = "stable", shape = c(0.5, 0.5, 2), scale = c(5, 10, 10/gamma(1.5)))
```

```
visualise_msm(sim_dat)
```

`sim_weibmsm` *Simulate multiple trajectories from an interval-censored multi-state model with Weibull transition intensities*

Description

Simulate multiple trajectories from a multi-state model quantified by a transition matrix, with interval-censored transitions and Weibull distributed transition intensities. Allows for Weibull censoring in each of the states.

Usage

```
sim_weibmsm(
  data,
  tmat,
  startprobs,
  exact,
  shape,
  scale,
  censshape,
  censscale,
  n_subj,
  obs_pars,
  true_trajec = FALSE
)
```

Arguments

`data`	A `data.frame` or `matrix` with named columns `time` and `id`, representing the observation times and corresponding subject id(entifier).
`tmat`	A transition matrix as created by `transMat`, with H rows and H columns indicating the states. The total number of possible transitions will be indicated by M.
`startprobs`	A numeric vector of length H indicating the probability of each subject to start in any of the possible states. Must sum to 1. By default, all subjects will start in state 1.
`exact`	A numeric vector indicating which states are exactly observed. The transition time to exact states will be observed at exact times, regardless of the times in `obstimes`. No exact states if missing.
`shape`	A numeric vector of length M indicating the shape of the Weibull transition intensity for the corresponding transition in `tmat`. See `help(dweibull)`.
`scale`	A numeric vector of length M indicating the scale of the Weibull transition intensity for the corresponding transition in `tmat`. See `help(dweibull)`.

censshape	A numeric vector of length H indicating the Weibull censoring shape in each of the states. If no censoring is required in some states, set corresponding entries to NA. If left missing, no censoring is applied. See details.
censscale	A numeric vector of length H indicating the Weibull censoring scale in each of the states. If no censoring is required in some states, set corresponding entries to NA. If left missing, no censoring is applied. See details.
n_subj	(Optional) Instead of specifying data, specify the number of subjects to generate trajectories for. Requires obs_pars to also be specified.
obs_pars	(Optional) A numeric vector of length 3 specifying what the time is between planned assessments, what the uniform deviation from this time is at the visits and the maximum visit time. Specifying obs_pars = c(2, 0.5, 20) will generate a grid of observation times (0, 2, 4, ..., 20) with a uniform[-0.5, 0.5] random variable added to each observation time, and cut-off at the end-points 0 and 20. The observation times may not overlap, so the first argument must be at least twice as large as the second.
true_trajec	Should the true (right-censored) trajectory be returned for the subjects as well? Default = FALSE.

Details

Taking (cens)shape to be 1 for all transitions, we obtain exponential (censoring)/transitions with rate 1/(cens)scale.

If right-censoring parameters are specified, a right-censoring time is generated in each of the visited states. If the subject is right-censored, we assume the subject is no longer observed at later obstimes. Due to the interval-censored nature of the generation process, it may therefore appear as if the subject was right-censored in an earlier state.

Suppose a subject arrives in state g at time s. If we wish to generate a survival time from that state according to a Weibull intensity in a clock forward model, we can use the inverse transform of the conditional Weibull intensity. More specifically, letting a denote the shape and σ denote the scale, the conditional survival function for $t > s$ is given by

$$S(t|s) = \mathbf{P}(T \geq t | T \geq s) = \exp(\left(\frac{s}{\sigma}\right)^a - \left(\frac{t}{\sigma}\right)^a)$$

The corresponding cumulative intensity is then given by:

$$A(t|s) = -\log(S(t|s)) = \left(\frac{t}{\sigma}\right)^a - \left(\frac{s}{\sigma}\right)^a$$

And the inverse cumulative intensity is then:

$$A^{-1}(t|s) = \sigma \sqrt[a]{t + \left(\frac{s}{\sigma}\right)^a}$$

A conditional survival time is then generated by:

$$T|s = A^{-1}(-\log(U)|s)$$

with U a sample from the standard uniform distribution. If we additionally have covariates (or frailties), the $-\log(U)$ above should be replaced by $\frac{-\log(U)}{\exp(\beta X)}$ with β and X the coefficients and covariates respectively.

Value

A `matrix` with 3 columns `time`, `state` and `id`, indicating the observation time, the corresponding state and subject identifier. If `true_trajec = TRUE`, a `list` with the matrix described above and a matrix representing the underlying right-censored trajectory.

Examples

```
require(mstate)
require(ggplot2)
#Generate from an illness-death model with exponential transitions with
#rates 1/2, 1/10 and 1 for 10 subjects over a time grid.
gd <- sim_weibmsm(tmat = trans.illdeath(), shape = c(1,1,1),
                  scale = c(2, 10, 1), n_subj = 10, obs_pars = c(2, 0.5, 20),
                  startprobs = c(0.9, 0.1, 0), true_trajec = TRUE)

#Observed trajectories
visualise_msm(gd$observed)
#True trajectories
visualise_msm(gd$true)


#Can supply data-frame with specified observation times
obs_df <- data.frame(time = c(0, 1, 3, 5, 0.5, 6, 9),
                     id = c(1, 1, 1, 1, 2, 2, 2))
gd <- sim_weibmsm(data = obs_df, tmat = trans.illdeath(), shape = c(1, 1, 1),
                  scale = c(2, 10, 1))
visualise_msm(gd)
```

`smoothmsm` *Smooth hazard estimation for general multi-state model with interval censored transitions*

Description

For a general Markov chain multi-state model with interval censored transitions calculate the NPMLE of the transition intensities. The estimates are returned as an `msfit` object. The smallest time in the data will be set to zero.

Usage

```
smoothmsm(
  gd,
  tmat,
  exact,
  formula,
  data,
  deg_splines = 3,
  n_segments = 20,
```

```
  ord_penalty = 2,
  maxit = 100,
  tol = 1e-04,
  Mtol = 1e-04,
  conv_crit = c("haz", "prob", "lik"),
  verbose = FALSE,
  prob_tol = tol/10,
  ode_solver = "lsoda",
  ridge_penalty = 1e-06
)
```

Arguments

gd	A `data.frame` with the following named columns `id`: Subject idenitifier; `state`: State at which the subject is observed at `time`; `time`: Time at which the subject is observed; The true transition time between states is then interval censored between the times.
tmat	A transition matrix as created by `transMat`
exact	Numeric vector indicating to which states transitions are observed at exact times. Must coincide with the column number in `tmat`.
formula	Formula to interpret in data for covariates.
data	A `data.frame` containing a column called `'id'` (identifying the subjects in `gd`) and variables which to interpret in `formula`.
deg_splines	Degree to use for the B-spline basis functions. Defaults to 3 (cubic B-splines).
n_segments	Number of segments to use for the P-splines. The segments will space the domain evenly. According to Eilers & Marx (2021), it it OK to choose this number very large. Default = 20.
ord_penalty	Order of the P-spline penalty (penalty on the difference between d-order differences of spline coefficients). See Eilers & Marx Section 2.3. Defaults to 2.
maxit	Maximum number of iterations. Default = 100.
tol	Tolerance of the convergence procedure in the E-step. A change in the value of `conv_crit` in an iteration of less than `tol` will make the procedure stop.
Mtol	Tolerance of the convergence procedure of the M-step. The M-step consists of an Iteratively Reweighted Least Squares (IRLS) procedure, where the (unobserved) complete-data likelihood is maximized. Default is `1e-4`.
conv_crit	Convergence criterion. Stops procedure when the difference in the chosen quantity between two consecutive iterations is smaller than the tolerance level `tol`. One of the following: **"haz"** Stop when change in maximum estimated intensities (hazards) < `tol`. **"prob"** Stop when change in estimated probabilities < `tol`. **"lik"** Stop when change in observed-data likelihood < `tol`.

	Default is "haz". The options "haz" and "lik" can be compared across different methods, but "prob" is dependent on the chosen method. Most conservative (requiring most iterations) is "prob", followed by "haz" and finally "lik".
verbose	Should iteration messages be printed? Default is FALSE
prob_tol	If an estimated probability is smaller than prob_tol, it will be set to zero during estimation. Default value is tol/10.
ode_solver	The integrator to use for solving the ODE's. See ode(). By default, the "lsoda" solver will be used.
ridge_penalty	The ridge penalty to use for estimating risk-adjustment coefficients. Default = 1e-06.

References

Eilers, P.H.C. and Marx, B.D., Practical Smoothing: The Joys of P-splines, Cambridge University Press (2021)

---

summary.probtrans.subjects

Summary method for a probtrans.subjects object

Description

Summary method for an object of class 'probtrans.subjects'. It prints a selection of the estimated transition probabilities. Wrapper for summary.probtrans.

Usage

```
## S3 method for class 'probtrans.subjects'
summary(object, id, times, from = 1, to = 0, extend = FALSE, ...)
```

Arguments

object	Object of class 'probtrans.subjects', containing estimated transition probabilities from and to all states in a multi-state model
id	Subject identifier
times	Time points at which to evaluate the transition probabilites
from	Specifies from which state the transition probabilities are to be printed. Should be subset of 1:S, with S the number of states in the multi-state model. Default is print from state 1 only. User can specify from=0 to print transition probabilities from all states
to	Specifies the transition probabilities to which state are to be printed. User can specify to=0 to print transition probabilities to all states. This is also the default
extend	logical value: if TRUE, prints information for all specified times, even if there are no subjects left at the end of the specified times. This is only valid if the times argument is present
...	Further arguments to summary.probtrans

Value

Function `summary.probtrans` returns an object of class "summary.probtrans.subjects", which is a list (for each `from` state) of transition probabilities at the specified (or all) time points. The `print` method of a `summary.probtrans.subjects` doesn't return a value.

Author(s)

Hein Putter and Daniel Gomon

See Also

`summary.probtrans`, `predict_tp`

Examples

```
if(require("mstate")){
  data(ebmt3)
  n <- nrow(ebmt3)
  tmat <- transMat(x = list(c(2, 3), c(3), c()), names = c("Tx",
                                                           "PR", "RelDeath"))
  ebmt3$prtime <- ebmt3$prtime/365.25
  ebmt3$rfstime <- ebmt3$rfstime/365.25
  covs <- c("dissub", "age", "drmatch", "tcd", "prtime")
  msbmt <- msprep(time = c(NA, "prtime", "rfstime"), status = c(NA,
                  "prstat", "rfsstat"), data = ebmt3, trans = tmat, keep = covs)
  #Expand covariates so that we can have transition specific covariates
  msbmt <- expand.covs(msbmt, covs, append = TRUE, longnames = FALSE)

  #Simple model, transition specific covariates, each transition own baseline hazard
  c1 <- coxph(Surv(Tstart, Tstop, status) ~ dissub1.1 + dissub2.1 +
                age1.1 + age2.1 + drmatch.1 + tcd.1 + dissub1.2 + dissub2.2 +
                age1.2 + age2.2 + drmatch.2 + tcd.2 + dissub1.3 + dissub2.3 +
                age1.3 + age2.3 + drmatch.3 + tcd.3 + strata(trans), data = msbmt,
                method = "breslow")
  #We need to make a data.frame containing all subjects of interest
  ttmat <- to.trans2(tmat)[, c(2, 3, 1)]
  names(ttmat)[3] <- "trans"
  nd_n <- NULL
  for (j in 1:30) {
    # Select global covariates of subject j
    cllj <- ebmt3[j, covs]
    nd2 <- cbind(ttmat, rep(j, 3), rbind(cllj, cllj, cllj))
    colnames(nd2)[4] <- "id"
    # Make nd2 of class msdata to use expand.covs
    attr(nd2, "trans") <- tmat
    class(nd2) <- c("msdata", "data.frame")
    nd2 <- expand.covs(nd2, covs=covs, longnames = FALSE)
    nd_n <- rbind(nd_n, nd2)
   }

   icmstate_pt <- probtrans_coxph(c1, predt = 0, direction = "forward",
                                  newdata = nd_n, trans = tmat)
```

```
   #Obtain summary of probtrans.subjects object
   plot(icmstate_pt, id = 2)
}
```

---

supportHudgens *Determine the support of the NPMLE for interval censored data.*

Description

Given censoring/truncation intervals, find the maxcliques and determine the support of the interval censored problem.

Usage

```
supportHudgens(intervals, reduction = TRUE, existence = FALSE)
```

Arguments

intervals	A data.frame with 3 columns containing half-open intervals (left open, right closed) and an indicator whether the interval results from a censored transition or truncation: **L:** Left side of interval; **R:** Right side of interval; **cens:** Indicator whether interval resulted from interval censoring or left truncation (1 = censoring, 0 = truncation); **id:** (optional) Identifier for the observation this interval belongs to (numeric/integer). Only required if existence = TRUE; Note that the truncation intervals need to be in the form (N, Inf] with N a numeric value.
reduction	Should the support be reduced using Lemma 3 from Hudgens (2005)? This requires checking an extra condition. Default is TRUE.
existence	Should the existence of the NPMLE be checked using Theorem 1/Lemma 4 from Hudgens (2005)? Requires id to be present in intervals. Default is FALSE.

Value

- graph: An igraph object representing the censoring/truncation intervals
- support: Support estimated from the censoring intervals
- dir_graph: A directed igraph object used to determine whether the NPMLE exists in the presence of left-truncation.
- exist_mle: Logical output indicating whether the NPMLE exists.

References

Michael G. Hudgens, On Nonparametric Maximum Likelihood Estimation with Interval Censoring and Left Truncation, Journal of the Royal Statistical Society Series B: Statistical Methodology, Volume 67, Issue 4, September 2005, Pages 573-587, doi:10.1111/j.14679868.2005.00516.x

---

`support_from_direct_intervals` *Estimate support of multiple transitions given direct transition intervals*

Description

Given only direct transition intervals, determine the support for each transition separately using Hudgens(2001) result. Each state is considered from a competing-risks viewpoint. Hudgens(2005) result is applied to see if the NPMLE for any of the transitions does not exist.

Usage

```
support_from_direct_intervals(direct_intervals, tmat)
```

Arguments

`direct_intervals`	Output from `direct_from_observed_intervals`.
`tmat`	A transition matrix as created by `transMat`

Value

A list containing the estimated support sets for each possible transition in `tmat`.

`support_npmsm` *Numerically find the support of the transitions from a converged npmsm algorithm*

Description

For each transition in `tmat`, determine all consecutive bins with non-zero (higher than `cutoff`) transition intensities. These then determine the numerical support of the transition.

Usage

```
support_npmsm(npmsm, cutoff = 1e-08)
```

Arguments

npmsm	Output from npmsm function or an msfit object.
cutoff	Above which value is a mass in a bin considered to be non-zero? Default = 1e-8. Note that this is independent of bin size, so can be tricky!!

Value

A list containing a list for each transition. Each transition specific list contains the support intervals for that transition in a matrix with 3 named columns L, R and dA, indicating the left/right-endpoints of the support intervals and the change in the estimated intensities over this support interval.

Examples

```
require(mstate)
require(ggplot2)
#Generate from an illness-death model with exponential transitions with
#rates 1/2, 1/10 and 1 for 10 subjects over a time grid.
gd <- sim_weibmsm(tmat = trans.illdeath(), shape = c(1,1,1),
                  scale = c(2, 10, 1), n_subj = 10, obs_pars = c(2, 0.5, 20),
                  startprobs = c(0.9, 0.1, 0))
#Fit 2 models: 1 with at most 4 iterations and 1 with at most 20
mod1 <- npmsm(gd, trans.illdeath(), maxit = 4)

#Determine support numerically:
mod1_supp <- support_npmsm(mod1)
mod1_supp[[1]]
```

---

transprob.msm *Wrapper for the probtrans function*

Description

For 'msm' objects: determine transition probabilities (as in probtrans) from an msm object. Currently only direction = "forward" is supported.

For 'npmsm' objects: Determine transition probabilities for an 'npmsm' object using the probtrans function.

For 'msfit' objects: Wrapper for probtrans

Usage

```
## S3 method for class 'msm'
transprob(object, predt, times, ...)

## S3 method for class 'npmsm'
transprob(object, ...)

## S3 method for class 'msfit'
```

```
transprob(object, ...)

transprob(object, ...)
```

Arguments

object	Object of compatible class
predt	A positive number indicating the prediction time. This is the time at which the prediction is made. If missing, smallest time of times is chosen.
times	A vector of times at which the transition probabilities should be determined.
...	Further arguments to probtrans

Details

Can be used for objects of class 'npmsm', 'msm' and 'msfit'

Value

A probtrans object containing the estimated transition probabilities.

---

visualise_data *Visualise data for illness-death model, only applicable to Frydman(1995) setting.*

Description

Visualise data for illness-death model, only applicable to Frydman(1995) setting.

Usage

```
visualise_data(data, msmFrydman)
```

Arguments

data	A data.frame containing the columns named: **delta:** Did a transition from 1 -> 2 occur? (binary: 0 = no, 1 = yes); In the left-truncated case, delta = 2 indicates initially observed in state 2. **Delta:** Was the transition to state 3 observed? (binary: 0 = no, 1 = yes); **L:** Left timepoint of interval censored transition to state 2 (numeric); **R:** Right timepoint of interval censored transition to state 2 (numeric); **time:** Time of event (transition to 3) or right-censoring in state 2 (numeric); **trunc:** (optional) Left-truncation time (numeric); Only used for entries with delta = 2.
msmFrydman	A fitted model from msm_frydman

Value

Returns a visualisation of illness-death data, with the transition from healthy to illness interval-censored and the other two transitions observed exactly or right-censored. If `msmFrydman` is specified, the support intervals from the fit are additionally plotted at the top of the data visualisation.

References

Frydman, H. (1995). Nonparametric Estimation of a Markov 'Illness-Death' Process from Interval-Censored Observations, with Application to Diabetes Survival Data. Biometrika, 82(4), 773-789. doi:10.2307/2337344

See Also

See `msm_frydman` for fitting a model.

Examples

```
data <- data.frame(delta = c(0, 0, 1, 1), Delta = c(0, 1, 0, 1),
                   L = c(NA, NA, 1, 1.5), R = c(NA, 3, 2, 3),
                   time = c(4, 5, 6, 7))

mod_frydman <- msm_frydman(data)
visualise_data(data, mod_frydman)
```

---

visualise_msm *Visualise multi-state data*

Description

Produce a plot with the y-axis representing subjects in the data and the x-axis representing the time at which states have been observed.

Usage

```
visualise_msm(gd, npmsm, tmat, neat = TRUE, cutoff)
```

Arguments

`gd`	A `data.frame` containing the following named columns: `id`: Identifier of subject; `state`: state of subject at `time`; `time`: time at which subject is observed;
`npmsm`	Output from `npmsm` function
`tmat`	A transition matrix as created by `transMat`
`neat`	Boolean indicating whether redundant observations should be removed in the plot. Default is TRUE
`cutoff`	cutoff value for numerically determining the support using `support_npmsm`

Value

A plot will be produced in the plotting window.

Examples

```
#Write a function for evaluation times: observe at 0 and uniform inter-observation times.
eval_times <- function(n_obs, stop_time){
  cumsum( c( runif(1, 0, 0.5),  runif( n_obs-1, 0, 2*(stop_time-4)/(n_obs-1) ) ) )
}
#Use built_in function to simulate illness-death data
#from Weibull distributions for each transition
sim_dat <- sim_id_weib(n = 50, n_obs = 6, stop_time = 15, eval_times = eval_times,
                       start_state = "stable", shape = c(0.5, 0.5, 2),
                       scale = c(5, 10, 10/gamma(1.5)))

#Visualise the data
visualise_msm(sim_dat)
```

Index

∗ **print**
summary.probtrans.subjects, 37

ageqb, 3
agreaterb, 4
ainB, 4
Aintersectb, 5
Alargerb, 5
AsubsetB, 6

coxph, *23*, *26*

direct_from_observed_intervals, 6, *40*

estimate_support_msm, 7, *15*
evalstep, 8
expand.covs, *23*, *27*
extend_msfit, 9

get_trans_intervals, *6*, 9
graphfromIntervals, 10

icmstate *(*icmstate-package*)*, 2
icmstate-package, 2
interpol_msfit, 11

msfit, *13*, *16–18*, *35*, *41*
msm, *41*
msm_frydman, 12, *42*, *43*
msprep, *23*, *26*
mstate, *18*, *20*

npmsm, 13, *25*, *41*, *43*

ode(), *37*

plot, *20*
plot.msfit, *18*, *20*
plot.npmsm, *17*, 18
plot.probtrans, *18*
plot.probtrans.subjects, 18, *24*
plot.smoothmsm, 19
plot_probtrans, 20
plot_surv, 22
predict_tp, 23, *38*
print.npmsm, *17*, 25
print.summary.probtrans.subjects, 25
probtrans, *24*, *27*, *41*, *42*
probtrans_coxph, *23*, *24*, 26
probtrans_weib, 29
prod_lambda_G_base, 30

remove_redundant_observations, 30
rweibull, *32*

sim_id_weib, 31
sim_weibmsm, 33
smoothmsm, 35
summary.probtrans, *37*, *38*
summary.probtrans.subjects, *24*, 37
support_from_direct_intervals, 40
support_npmsm, 40, *43*
supportHudgens, 39

to.trans2, *16*
transMat, *14*, *16*, *23*, *27*, *33*, *36*
transprob, *16*, *17*
transprob *(*transprob.msm*)*, 41
transprob.msm, 41

visualise_data, 42
visualise_msm, *17*, 43