Package: dtms (via r-universe)

May 10, 2026

Type Package

Title Discrete-Time Multistate Models

Version 0.4.4

Description Discrete-time multistate models with a user-friendly workflow. The package provides tools for processing data, several ways of estimating parametric and nonparametric multistate models, and an extensive set of Markov chain methods which use transition probabilities derived from the multistate model. Some of the implemented methods are described in Schneider et al. (2024) <doi:10.1080/00324728.2023.2176535>, Dudel (2021) <doi:10.1177/0049124118782541>, Dudel & Myrskylä (2020) <doi:10.1186/s12963-020-00217-0>, van den Hout (2017) <doi:10.1201/9781315374321>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

URL https://github.com/christiandudel/dtms

BugReports https://github.com/christiandudel/dtms/issues

Depends R (>= 4.1.0)

Imports mclogit, nnet, VGAM, doParallel, foreach, markovchain, methods

Config/pak/sysreqs libglpk-dev make libxml2-dev

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

Date/Publication 2026-04-23 14:04:17 UTC

RemoteUrl https://github.com/christiandudel/dtms

RemoteRef HEAD

RemoteSha 1bd062fa72fe7ffe32970b0eb2f301713353e488

Contents

`dtms` *Create dtms object*

Description

This function creates an object of class 'dtms' to be passed to other functions of the package.

Usage

```
dtms(transient, absorbing, timescale, timestep = NULL, sep = "_")
```

Arguments

`transient`	A character vector of names of the transient states in the state space.
`absorbing`	A character vector of names of the absorbing states in the state space.
`timescale`	A numeric vector with the time scale, including the starting time and the final time.
`timestep`	Numeric (optional), step length of the time scale, will be guessed if NULL (default).
`sep`	Character (optional), separator between short state name and value of time scale. Default is '_'.

Details

`dtms` provides an abstract definition of a multistate model, including the names of the transient states, the names of the absorbing states, the values the time scale can take, and the step length of the time scale.

The names of the absorbing and transient states should be provided as character strings. However, numeric values also work. Factors are not supported. All states should appear in the data, otherwise error messages might occur.

The step length of the time scale can be a vector with several values, which allows for unevenly spaced observations. Note, however, that some functions require one specific value for the step length; e.g., `dtms_transitions()`. For such functions, if several values are provided the first value will be used.

Value

Returns an object of class 'dtms'

Examples

```
dtms(transient=c("A","B"),
     absorbing="X",
     timescale=1:10)
```

`dtms_absorbed` *Calculate the distribution of the time until entering an absorbing state*

Description

Calculates the distribution of the time until entering any of the absorbing states.

Usage

```
dtms_absorbed(
  probs = NULL,
  matrix = NULL,
  dtms,
  start_distr = NULL,
  start_state = NULL,
  start_time = NULL,
  end_time = NULL,
  method = "mid"
)
```

Arguments

`probs`	Data frame with transition probabilities, as created with `dtms_transitions`.
`matrix`	Matrix with transition probabilities, as generated with `dtms_matrix`.
`dtms`	dtms object, as created with `dtms`.
`start_distr`	Numeric (optional), distribution of starting states. If specified, average distribution over all starting states will be calculated.
`start_state`	Character (optional), name of starting states. If NULL (default) all transient states will be used.
`start_time`	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
`end_time`	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.
`method`	Character (optional), do transitions happen mid-interval (‘mid‘, default) or at the end of the interval (‘end‘), see details.

Details

In a discrete-time model, the time spent in a state depends on assumptions about when transitions happen. Currently, this functions supports two variants which can be specified with the argument ‘method‘: mid-interval transitions can be selected with the option ‘mid‘ and imply that transitions happen at the middle of the time interval; and the option ‘end‘ assumes that instead transitions happen at the end of the interval. The calculation takes the step length of the time scale into account as specified by the ‘dtms‘ object. If the #’ step length is not one fixed value, the first entry of ‘dtms$timestep‘ will be used.

If a distribution of the starting states is provided with 'start_distr' the output table has an additional row, showing the waiting time distribution unconditional on the starting state.

Value

A table with the distribution of time until entering any of the absorbing states.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## Distribution of visits
dtms_absorbed(dtms=simple,
              probs=probs,
              start_distr=S)
```

---

dtms_aggregate *Aggregate data*

Description

This function takes any data set and returns a new data frame which only includes the unique rows from the original data set and indicates how often these rows appear in the original data.

Usage

```
dtms_aggregate(data, weights = NULL, idvar = "id", countvar = "count")
```

Arguments

data	Data frame.
weights	Character (optional). Name of variable with weights.
idvar	Character (optional). Name of variable in ‘data‘ with unit ID. Default is "id".
countvar	Character (optional). Name of new variable in data which provides the counts. Default is "count".

Details

Currently, missing values are not supported and will be dropped; consider using factors if you want to keep them. The variable provided with the argument ‘idvar‘ will be dropped from the aggregated data. If ‘weights‘ is specified, the counts will be placed in a variable with the same name. If ‘countvar‘ is used, any existing variable in the original data with the same name will be replaced.

Value

An aggregated data frame

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Aggregate
aggdata <- dtms_aggregate(estdata)
## Fit model
fit <- dtms_fit(data=aggdata,
                weights="count")
```

---

dtms_backward *Carry states backward*

Description

This function carries a state backward after its last occurrence.

Usage

```
dtms_backward(
  data,
  state,
  fromvar = "from",
  tovar = "to",
  statevar = NULL,
  idvar = "id",
  timevar = "time",
  dtms = NULL,
  overwrite = "missing",
  vector = FALSE
)
```

Arguments

data	A data frame in long format.
state	Character, name of the state to be carried forward.
fromvar	Character (optional), name of variable with starting state. Default is 'from'.
tovar	Character (optional), name of variable with receiving state. Default is 'to'.
statevar	Character (optional), name of the variable in the data frame in long format with the states. Default is NULL.
idvar	Character (optional), name of variable with unit ID. Default is 'id'.
timevar	Character (optional), name of variable with time scale. Default is 'time'.
dtms	dtms object (optional), as created with `dtms`. Not required if 'overwrite==transient'.
overwrite	Character (optional), one of 'transient', 'missing', 'absorbing', and 'all', see details. Default is 'transient'.
vector	Logical (optional), return vector (if TRUE) or data frame (if FALSE). Default is FALSE. Argument is only used if argument 'statevar' is specified.

Details

This function carries a state backward after its first occurrence. For instance, carrying the state "A" backward in the sequence 'B, B, A, B, B' will give the sequence 'A, A, A, B, B'. The sequence 'C, B, C, A, B, A, A, B' will give 'A, A, A, A, A, A, A, B'.

This function works with data frames in transition format and in long format. The default is transition format, using the arguments 'fromvar' and 'tovar'. If, however, the argument 'statevar' is specified, it is used instead.

The argument 'overwrite' is used to control what type of information is replaced. If 'overwrite==transient', then only transient states are replaced while missing values and absorbing states remain unchanged. For example, carrying backward state "A" in the sequence 'B, NA, B, B, X, A, X' with X being an absorbing state will give 'A, NA, A, A, X, A, X'. If 'overwrite==missing' then in addition to transient states also missing values are replaced and for the example sequence 'A, A, A, A, X, A, X' would be returned. If 'overwrite==absorbing' then in addition to transient states absorbing states will be replaced; for the example sequence the result would be 'A, NA, A, A, A, A, X'. Finally, if 'overwrite==all' then all values in the sequence will be replaced: 'A, A, A, A, A, A, X'.

Value

The data frame specified with ‘data‘ and the edited state variable (if ‘vector=FALSE‘) or a vector (if ‘vector=TRUE‘).

See Also

dtms_forward to carry states forward.

Examples

```
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:19)

dtms_backward(data=simpledata,
              statevar="state",
              state="A",
              dtms=simple,
              overwrite="transient")
```

---

dtms_boot *Bootstrap and block bootstrap*

Description

This function is a simple wrapper for bootstrapping and block-bootstrapping data in transition format. Parallel processing is supported.

Usage

```
dtms_boot(
  data,
  dtms,
  fun,
  rep,
  method = "simple",
  idvar = "id",
  weights = NULL,
  slack = 1,
  verbose = FALSE,
  progress = FALSE,
  parallel = FALSE,
  cores = 2,
  .packages = c("mclogit", "VGAM", "nnet", "dtms"),
  ...
)
```

Arguments

data	Data frame in transition format as created with dtms_format.
dtms	dtms object, as created with dtms.
fun	Function to be repeatedly applied, see details.
rep	Numeric, number of bootstrap replications.
method	Character (optional), either "simple" for simple bootstrap, "block" for block bootstrap, or "weights" for a weight-based parametric bootstrap. Default is "simple".
idvar	Character (optional), name of ID variable in 'data' identifying units. Only required for block bootstrap. Default is "id".
weights	Character (optional), name of variable with weights. Only used if 'method=weights'. Default is NULL.
slack	Numeric (optional), used to in parametric resampling to replace 0. Default is 1.
verbose	Logical (optional), print output which might be generated when running 'fun'? Default is FALSE.
progress	Logical (optional), indicate progress if simple bootstrap? Default is FALSE.
parallel	Logical (optional), use parallel processing? Default is FALSE.
cores	Numeric (optional), if parallel=TRUE, how many cores should be used? Default is 2.
.packages	Character (optional), packages to be loaded when parallel processing. Default is 'c("mclogit","VGAM","nnet","dtms")'
...	Arguments to be passed to 'fun', only works if 'parallel=FALSE'.

Details

dtms_boot() takes a function specified with the argument 'fun' and applies it several times to resampled data, where the original data is specified with the argument 'data' and 'rep' specifies the number of replications. The argument 'dtms' takes an object created with dtms() and also passes it to 'fun'. 'data' is passed to 'fun' as its first argument, and 'dtms' is passed as the second argument.

The result of this function is a list with as many entries as there are replications. Each entry is the result of calling 'fun' for the respective replication.

Three methods are implemented and selected with the argument 'method'. A simple resampling bootstrap, which assumes that the rows in 'data' are independent of each other ('method=simple'). The block bootstrap which allows for dependent observations; e.g., different units each contributing several transitions ('method=block'). Moreover, a parametric bootstrap using weights is also supported, assuming that observations are i.i.d. multinomial ('method=weights'). If the block bootstrap is used the argument 'idvar' sets which variable in #' 'data' contains information the unit/cluster identifier. In case the parametric bootstrap is used the argument 'weights' is used to specify the name of the variable with the weights.

For parallel computing the packages 'foreach' and 'doParallel' are used. See the documentation of these packages for details.

Value

A list of results, see details

See Also

dtms_boot_summary to help with summarizing the results.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Simple resampling bootstrap
# Bootstrap function
bootfun <- function(data,dtms) {
  fit <- dtms_fit(data=data)
  probs    <- dtms_transitions(dtms=dtms,
                               model = fit)
  S <- dtms_start(dtms=dtms,
                  data=data)
  dtms_expectancy(dtms=dtms,
                  probs=probs,
                  start_distr=S)
}
# Run bootstrap
bootstrap <- dtms_boot(data=estdata,
                       dtms=simple,
                       fun=bootfun,
                       rep=5)
summary(bootstrap,
        probs=c(0.025,0.5,0.975))
# Parametric bootstrap
aggdata <- dtms_aggregate(estdata)
# Bootstrap function
bootfun <- function(data,dtms) {
  fit <- dtms_fit(data=data,weights="count")
  probs <- dtms_transitions(dtms=dtms,
                            model = fit)
  S <- dtms_start(dtms=dtms,
                  data=data,
                  weights="count")
  dtms_expectancy(dtms=dtms,
                  probs=probs,
                  start_distr=S)
}
# Bootstrap
```

```
bootstrap <- dtms_boot(data=aggdata,
                       dtms=simple,
                       fun=bootfun,
                       rep=5,
                       weights="count",
                       method="weights")
# Results
summary(bootstrap,
        probs=c(0.025,0.5,0.975))
```

`dtms_boot_summary` *Summary function for bootstrap results*

Description

Provides bootstrap percentiles for bootstrap replications created with `dtms_boot()`.

Usage

```
dtms_boot_summary(boot, probs = NULL, alpha = 0.05)
```

Arguments

`boot`	Object created with `dtms_boot()`.
`probs`	Numeric (optional), vector of percentiles. Default is NULL.
`alpha`	Numeric (optional), confidence level. Default is 0.05.

Details

Percentiles can be specified with the argument `probs`. This can be as many percentiles as required by the user. If it is not specified, the argument `alpha` is used instead. `alpha` is the confidence level for the confidence intervals.

The function passed to `dtms_boot()` needs to either return a numeric vector, a matrix, or a data.frame, otherwise `dtms_boot_summary()` returns an error #' message. A data.frame will be transformed into a matrix.

Value

Either a vector or a matrix.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
```

```
                      dtms=simple,
                      idvar="id",
                      timevar="time",
                      statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Bootstrap function
bootfun <- function(data,dtms) {
  fit <- dtms_fit(data=data)
  probs    <- dtms_transitions(dtms=dtms,
                               model = fit)
  S <- dtms_start(dtms=dtms,
                  data=data)
  dtms_expectancy(dtms=dtms,
                  probs=probs,
                  start_distr=S)
}
# Run bootstrap
bootstrap <- dtms_boot(data=estdata,
                      dtms=simple,
                      fun=bootfun,
                      rep=5)
summary(bootstrap,
        probs=c(0.025,0.5,0.975))
```

---

dtms_censoring *Left censoring, right censoring, and gaps in data*

Description

This function provides an overview of censoring and gaps in the data. It can do so in several ways: by providing counts of units with left censoring, right censoring, and gaps; by providing a cross-tabulation of the number of units with left censoring and/or right censoring and/or gaps; and by returning a data frame with added indicators on censoring and gaps.

Usage

```
dtms_censoring(
  data,
  dtms,
  fromvar = "from",
  tovar = "to",
  timevar = "time",
  idvar = "id",
  print = TRUE,
  printlong = FALSE,
  add = FALSE,
  addtype = "id",
```

```
  varnames = c("LEFT", "GAP", "RIGHT")
)
```

Arguments

data	Data frame in transition format, as created with dtms_format.
dtms	dtms object, as created with dtms.
fromvar	Character (optional), name of variable in 'data' with starting state. Default is "from".
tovar	Character (optional), name of variable in 'data' with receiving state. Default is "to".
timevar	Character (optional), name of variable in 'data' with time scale. Default is "time".
idvar	Character (optional), name of variable in 'data' with unit ID. Default is "id".
print	Logical (optional), print counts? Default is TRUE.
printlong	Logical (optional), print cross-tabulation? Default is FALSE.
add	Logical (optional), add indicators to data set? Default is FALSE. If TRUE the data frame specified with data is returned with added columns.
addtype	Character (optional), what type of information should be added if add=TRUE. Either "id" or "obs", see details. Default is "id".
varnames	Character vector (optional), names of added variables if add=TRUE. Default is "c("LEFT","GAP","RIGHT")".

Details

Added variables can be at the unit level or at the observation level. This is controlled by the argument "addtype". If it is set to "id" then the unit level is used. In this case the added variables are the same for each observation of a unit. For instance, if a unit experiences any gap, then the added variable has the value TRUE for all observations of that unit. If "addtype" is set to "obs" the observation level is used and the indicators are only set to TRUE if they apply to a specific observation. For instance, if a unit experience right censoring, only the last observation will have TRUE as the value for the right-censoring indicator; i.e., showing that after this last observation there is right censoring. This can be helpful for analyses to understand censoring better.

Value

Table or data frame.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:19)
# Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
```

```
                      idvar="id",
                      timevar="time",
                      statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Censoring
dtms_censoring(data=estdata,
               dtms=simple)
```

`dtms_clean` *Cleans data in transition format*

Description

Cleans data in transition format. It can handle issues regularly occurring with such data: transitions starting from or ending in missing states, observations not covered by the time range, transitions starting or ending in a state which is not in the state space, and observations starting in absorbing states.

Usage

```
dtms_clean(
  data,
  dtms,
  fromvar = "from",
  tovar = "to",
  timevar = "time",
  dropTime = TRUE,
  dropState = TRUE,
  dropNA = TRUE,
  dropAbs = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	Data frame, as created with `dtms_format`.
`dtms`	dtms object, as created with `dtms`.
`fromvar`	Character (optional), name of variable with starting state. Default is "from".
`tovar`	Character (optional), name of variable with receiving state. Default is "to".
`timevar`	Character (optional), name of variable with time scale. Default is "time".
`dropTime`	Logical (optional), drop transitions with values of time not covered by the model. Default is TRUE.
`dropState`	Logical (optional), drop transitions with states which are not part of the state space. Default is TRUE.

dropNA	Logical (optional), drop transitions with gaps, last observations, and similar. Default is TRUE.
dropAbs	Logical (optional), drop transitions starting from absorbing states. Default is TRUE.
verbose	Logical (optional), print how many transitions were dropped. Default is TRUE

Details

Transitions starting or ending with a missing state often occur for three reasons. First, the function dtms_format will always create a transition with a missing receiving state for the last observation of a unit, whether due to censoring or not. For instance, if t=20 is the last value of the time scale, and a unit is in state A at that time, then there will be a transition starting at time t=20 from state A, and with receiving state missing. Such transitions can usually be safely ignored, in particular if there is only one absorbing state. Second, if, say, for a unit the last observation is at time t=10 and censored after, there will be a transition starting at time t=10 with missing receiving state. Whether such transitions can be ignored depends on the censoring mechanism. If it is uninformative these transitions can be dropped. Third, there might be missing values in the sequence of states. For instance, a unit might first be in state A, then state B, then the state is missing, and then state is again A, giving the sequence A, B, NA, A. This implies a transition from B to NA, and from NA to A.

Transitions which are out of the time range can occur, for instance, when the researcher is interested in a shorter time frame than covered by data collection. In a clinical trial, the time scale could capture follow-up time since start of the trial in months and data might be available for 60 months. But perhaps the researcher is only interested in the first 36 months.

Transitions which start or end in a state which is not in the state space occur when the states in the transition data are not included in the ‘dtms‘ object. This likely will apply to states which rarely occur and which the researcher does not want to combine with other states.

Value

Cleaned data frame in transition format.

Examples

```
# Define model
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:20)
# Transiton format, filling implicit missings with explicit missings
estdata <- dtms_format(data=simpledata,
dtms=simple,
idvar="id",
timevar="time",
statevar="state",
fill=TRUE)
# Clean data
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
```

`dtms_data_summary` *Summarize data in transition format*

Description

Returns a data frame with number of observed transitions (column COUNT), relative proportion of a transition relative to all transitions (column PROP), and raw transition probabilities Pr(j|i) (column PROB).

Usage

```
dtms_data_summary(
  data,
  dtms = NULL,
  fromvar = "from",
  tovar = "to",
  weights = NULL
)
```

Arguments

`data`	Data frame, as created with `dtms_format`.
`dtms`	dtms object, as created with `dtms`.
`fromvar`	Character (optional), name of variable in ‘data‘ with starting state. Default is "from".
`tovar`	Character (optional), name of variable in ‘data‘with receiving state. Default is "to".
`weights`	Character (optional), name of variable in ‘data‘ with weights. Default is NULL.

Value

A data frame

Examples

```
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
dtms_data_summary(estdata)
```

`dtms_delta` *Calculate delta*

Description

Calculates delta, either to compare transition probabilities from two different models, or to assess how including lags changes transition probabilities.

Usage

```
dtms_delta(
  data,
  dtms = NULL,
  model1 = NULL,
  model2 = NULL,
  lags = 1:5,
  controls = NULL,
  fromvar = "from",
  tovar = "to",
  timevar = "time",
  idvar = "id",
  reference = 1,
  package = "VGAM",
  full = FALSE,
  keepNA = TRUE,
  ...
)
```

Arguments

`data`	Data frame in transition format, as created with `dtms_format`.
`dtms`	dtms object, as created with `dtms`.
`model1`	Name of object containing a model estimated with `dtms_fit`.
`model2`	Name of object containing a model estimated with `dtms_fit`.
`lags`	Numeric (optional), vector containing the lags as positive integers.
`controls`	Character (optional), names of control variables
`fromvar`	Character (optional), name of variable in ‘data‘ with starting state. Default is "from".
`tovar`	Character (optional), name of variable in ‘data‘ with receiving state. Default is "to".
`timevar`	Character (optional), name of variable in ‘data‘ with time scale. Default is "time".
`idvar`	Character (optional), name of variable in ‘data‘ with unit ID. Default is "id".

reference	Numeric or character (optional). Reference level of multinomial logistic regression.
package	Character, chooses package for multinomial logistic regression, currently 'VGAM', 'nnet', and 'mclogit' are supported. Default is 'VGAM'.
full	Logical (optional), estimate fully interacted model? Default is FALSE.
keepNA	Logical (optional), keep missing values of lags as predictor value? Default is TRUE.
...	Further arguments passed to estimation functions.

Details

Delta is the weighted average absolute difference between the predicted transition probabilities from two multistate models. It can attain values between 0 and 1, where 0 indicates perfect similarity and 1 indicates that the two models always give predictions at the opposite extremes; i.e., for all predicted probabilities, one model predicts a probability of 0 and the other predicts a probability of 1.

This function is designed to use delta to assess the impact of including different lags of the state variable in the model.

To compare two different models, the arguments 'data', 'model1', and 'model2' are needed. 'data' specifies the data frame used for predicting transition probabilities. It needs to have all variables required for predicting based on both 'model1' and 'model2'. The latter two arguments are the names of multistate models estimated with `dtms_fit`.

To compare how the inclusion of different lags of the state variable affects predictions, a model needs to be specified using 'data' and 'dtms', as well as potential covariates with 'controls'. The argument 'lags' sets which lags are included These are always including lower lags; e.g., a model including the state at t-3 also has the state at t-2, at t-1, and at t. All resulting models are compared to a model which does not control for the current or any past state. If 'lags=NULL' the Markov model is compared to this model not accounting for the current state or any past states.

The argument 'keepNA' controls how missing values are handled. These will often occur for lagged states. For instance, for the first transition observed for an individual, the state at time t is known, but not at time t-1. In this case, if a first-order lag is used, this observation could either be dropped; or, a missing value of the state at time t-1 could be included as a predictor. 'keepNA=TRUE' will do the latter, while if 'FALSE', all observations with missing states are dropped. This is done for all models, irrespective of the lag, such that they are based on exactly the same observations.

Value

Vector of values of delta

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
```

```
                        dtms=simple,
                        idvar="id",
                        timevar="time",
                        statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit models
fit1 <- dtms_fit(data=estdata,controls="time")
fit2 <- dtms_fullfit(data=estdata,controls="time")

## Compare
dtms_delta(data=estdata,model1=fit1,model2=fit2)
```

`dtms_distr_summary` *Summary for distributional results*

Description

This function provides several summary measures for results obtained with dtms_visits, dtms_first, and dtms_last.

Usage

```
dtms_distr_summary(distr)
```

Arguments

`distr`	An object of class 'dtms_distr' created with `dtms_visits`, `dtms_first`, or `dtms_last`.

Value

A matrix with summary measures

Examples

```
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:19)
estdata <- dtms_format(data=simpledata,
                        dtms=simple,
                        idvar="id",
                        timevar="time",
                        statevar="state")
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
fit <- dtms_fit(data=estdata)
probs <- dtms_transitions(dtms=simple,
                          model = fit)
```

```
example <- dtms_visits(dtms=simple,
                       probs=probs,
                       risk="A")
summary(example)
```

`dtms_duration` *Generate variable with duration*

Description

This function creates a variable which measures the duration in a state.

Usage

```
dtms_duration(
  data,
  dtms,
  newname = "duration",
  statevar = "state",
  idvar = "id",
  timevar = "time",
  ignoreleft = TRUE,
  vector = FALSE
)
```

Arguments

`data`	A data frame in long format.
`dtms`	dtms object, as created with `dtms`.
`newname`	Character (optional), name of new variable if data set is returned. Default is "duration".
`statevar`	Character (optional), name of the variable in the data frame with the states. Default is 'state'.
`idvar`	Character (optional), name of variable with unit ID. Default is 'id'.
`timevar`	Character (optional), name of variable with time scale. Default is 'time'.
`ignoreleft`	Logical (optional), ignore left censoring and start counting at the first observation? Default is TRUE.
`vector`	Logical (optional), return vector (if TRUE) or data frame (if FALSE). Default is FALSE

Details

Counting starts with 1 and the first occurence in a state. For instance, if for an unit the sequence of states A, A, A, B, B, A, C is observed, the duration variable would include 1, 2, 3, 1, 2, 1, 1.

The argument 'ignoreleft' controls how left censoring is handled; i.e., what happens when for a unit there are no observations at the beginning of the time scale. If 'TRUE', left censoring is ignored, and counting starts at the first observation for a unit. For instance, if the time scale starts at t=0, but the first observation for a unit is at time t=2, and the sequence of states is again A, A, A, B, B, A, C, then 'ignoreleft=TRUE' returns 1, 2, 3, 1, 2, 1, 1. If 'ignoreleft=FALSE', then the function would return NA, NA, NA, 1, 2, 1, 1.

The function handles gaps in the data by setting the duration to NA. For instance, if a unit is observed at times 1, 2, 4, 5, and 6, but not at time 3, and the states are A, A, B, C, C, then the duration variable will have the values 1, 2, NA, 1, 2.

Value

The data frame specified with 'data' with an additional column (if 'vector=FALSE') or a vector (if 'vector=TRUE').

Examples

```
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:19)

dtms_duration(data=simpledata,
              dtms=simple)
```

`dtms_expectancy` *Calculate state expectancy*

Description

This function calculates the expected time spent in the transient states (state expectancy).

Usage

```
dtms_expectancy(
  probs = NULL,
  matrix = NULL,
  dtms,
  risk = NULL,
  start_distr = NULL,
  start_time = NULL,
  start_state = NULL,
  end_time = NULL,
  correction = 0.5,
  total = TRUE,
```

```
  fundamental = FALSE,
  verbose = FALSE
)
```

Arguments

probs	Data frame with transition probabilities, as created with dtms_transitions.
matrix	Matrix with transition probabilities, as generated with dtms_matrix.
dtms	dtms object, as created with dtms.
risk	Character (otpional), name of one transient state. If specified expectancies are only shown for this state but by values of the time scale.
start_distr	Numeric (optional), distribution of starting states. If specified, average expectancy over all starting states will be calculated. Only applied if risk=NULL.
start_time	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
start_state	Character (optional), name of starting states. If NULL (default) all transient states will be used.
end_time	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.
correction	Numeric (optional), correction for expectancy when starting state and state under consideration match, see details. Defaults to 0.5.
total	Logical (optional), calculate total expectancy. Default is TRUE. Only applied if risk=NULL.
fundamental	Logical (optional), return fundamental matrix? Default is FALSE.
verbose	Logical (optional), print some information on what is computed. Default is FALSE.

Details

If the argument 'start_distr' is specified, the average of the state expectancies over all starting states is calculated. The names and length of 'start_distr' need to match the starting states generated by this function which are based on the 'dtms' object.

The partial expectancy for the time spent in the transient states can be calculated using the arguments 'start_time' and 'end_time'.

IF the argument 'risk' is specified, then only the remaining life expectancy for the state specified with this argument is shown, but for all time units of the time scale.

Two corrections to the results will be applied per default. Both corrections are required as the underlying formulas do actually not provide the expected time spent in a state, but the number of visits to a state. Time and visits are only equal under certain conditions; in particular, only if transitions between states happen mid-interval and the step length of the time scale is equal to one. The first correction will remove a certain amount of time spent in a certain state if its equal to the starting state. This is controlled with the argument 'correction' which is applied multiplicative. For instance, its default value 0.5 means that the state expectancy in some state X starting from state X is reduced by 0.5 time steps. The second correction uses the entry 'timestep' of 'dtms', and multiplies results with its value.

Value

A matrix with state expectancies.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## State expectancies
dtms_expectancy(dtms=simple,
                probs=probs,
                start_distr=S)
```

---

dtms_first *Time needed to reach a subset of states for the first time*

Description

This function calculates the distribution of the time needed to reach a subset of states for the first time.

Usage

```
dtms_first(
  probs = NULL,
  matrix = NULL,
  dtms,
  risk,
  start_time = NULL,
  start_state = NULL,
  start_distr = NULL,
```

```
  end_time = NULL,
  method = "mid",
  total = TRUE,
  rescale = TRUE
)
```

Arguments

probs	Data frame with transition probabilities, as created with dtms_transitions.
matrix	Matrix with transition probabilities, as generated with dtms_matrix.
dtms	dtms object, as created with dtms.
risk	Character, name of state(s) for which risk is of interest.
start_time	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
start_state	Character (optional), name of starting states. If NULL (default) all transient states will be used.
start_distr	Numeric (optional), distribution of starting states. If specified, average distribution over all starting states will be calculated.
end_time	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.
method	Character (optional), do transitions happen mid-interval (‘mid‘, default) or at the end of the interval (‘end‘), see details.
total	Logical (optional), should total of distribution be shown? See details. Default is FALSE.
rescale	Logical (optional), should distribution be rescaled to sum to 1? See details. Default is TRUE.

Details

The resulting distribution is conditional on ever reaching the subset of states, as it is not defined if the set is never reached. If the argument ‘rescale‘ is set to FALSE, the distribution will not sum to one but to the lifetime risk of ever reaching the subset.

The state(s) which count to the time are specified with the argument ‘risk‘. If several states are specified, the resulting distribution refers to the lifetime spent in any of the specified states.

In a discrete-time model, the time spent in a state depends on assumptions about when transitions happen. Currently, this functions supports two variants which can be specified with the argument ‘method‘: mid-interval transitions can be selected with the option ‘mid‘ and imply that transitions happen at the middle of the time interval; and the option ‘end‘ assumes that instead transitions happen at the end of the interval. In this latter case the distribution of the time spent in a state is equivalent to the number of visits to that state. The calculation takes the step length of the time scale into account as specified by the ‘dtms‘ object. If the step length is not one fixed value, the first entry of ‘dtms$timestep‘ will be used.

If a distribution of the starting states is provided with ‘start_distr‘ the output table has two additional rows. One shows the distribution unconditional on the starting state. The other shows the distribution conditional on not starting in any state of the risk set.

Value

A table of the distribution of the time needed to reach the subset of states

See Also

dtms_distr_summary to help with summarizing the resulting distribution.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## First visit
dtms_first(dtms=simple,
           probs=probs,
           risk="A",
           start_distr=S)
```

---

dtms_fit *Estimate (un)constrained discrete-time multistate model*

Description

This function estimates a (un)constrained discrete-time multistate model using multinomial logistic regression.

Usage

```
dtms_fit(
  data,
  controls = NULL,
```

```
  formula = NULL,
  weights = NULL,
  fromvar = "from",
  tovar = "to",
  reference = 1,
  package = "VGAM",
  full = FALSE,
  ...
)
```

Arguments

data	Data frame in transition format, as created with dtms_format.
controls	Character (optional), names of control variables
formula	Formula (optional). If no formula is specified, it will be build from the information specified with controls, fromvar, tovar, and timevar.
weights	Character (optional). Name of variable with survey weights.
fromvar	Character (optional), name of variable in 'data' with starting state. Default is "from".
tovar	Character (optional), name of variable in 'data' with receiving state. Default is "to".
reference	Numeric or character (optional). Reference level of multinomial logistic regression.
package	Character, chooses package for multinomial logistic regression, currently 'VGAM', 'nnet', and 'mclogit' are supported. Default is 'VGAM'.
full	Logical (optional), estimate fully interacted model? Default is FALSE.
...	Further arguments passed to estimation functions.

Details

The argument 'data' takes a data set in transition format. The model formula can either be specified by using the argument 'formula'. Alternatively, it can be specified with the arguments 'fromvar', 'tovar', and 'controls'. These are used if 'formula' is not specified. 'fromvar' takes the name of the variable with the starting state as a character string, 'tovar' the same for the receiving state, and 'controls' is an optional vector of control variables. 'fromvar' and 'tovar' have default values which match other functions of this package, making them a convenient alternative to 'formula' (see example).

If 'full=TRUE' a fully interacted model will be estimated in which each control variable is interacted with all starting states. This is equivalent to a full or unconstrained multistate model in which each transition is a regression equation.

The argument 'package' is used choose the package used for estimation. Currently, 'VGAM' (default), 'nnet', and 'mclogit' are supported. The functions used for estimation are, respectively, 'vgam', 'multinom', and 'mblogit'. Arguments for these functions are passed via '...'.

The argument 'reference' sets the reference category for the multinomial logistic regression. Weights for the regression can be passed via the arguments 'weights'. See the documentation of the package and function used for estimation for details.

Value

Returns an object with class depending on the package used.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
```

---

dtms_format *Reshape data to transition format*

Description

Takes a data frame in long format and reshapes it into transition format.

Usage

```
dtms_format(
  data,
  dtms,
  idvar = "id",
  timevar = "time",
  statevar = "state",
  fromvar = "from",
  tovar = "to",
  absorbing = TRUE,
  keepnames = FALSE,
  fill = FALSE,
  verbose = TRUE,
  steplength = FALSE,
  stepvar = "steplength"
)
```

Arguments

`data`	Data frame in long format.
`dtms`	dtms object, as created with `dtms`.
`idvar`	Character (optional), name of variable in ‘data‘ with unit ID. Default is "id".
`timevar`	Character (optional), name of variable in ‘data‘ with time scale. Default is "time".
`statevar`	Character (optional), name of variable in ‘data‘ with state, default is ’state’.
`fromvar`	Character (optional), name of variable‘with starting state in reshaped data. Default is "from".
`tovar`	Character (optional), name of variable with receiving state in reshaped data. Default is "to".
`absorbing`	Logical (optional), use first observed absorbing state consistently? See details. Default is TRUE.
`keepnames`	Logical (optional), keep original names for id and time variable? Default is FALSE; i.e., not keeping original names.
`fill`	Logical (optional), fill implicit missing values with explicit NA? Default is FALSE.
`verbose`	Logical (optional), create output to console if changing variable names is not possible? Default is TRUE.
`steplength`	Logical (optional), if true, the time to the next state is returned as a variable. Default is FALSE.
`stepvar`	Character (optional), if `steplength==TRUE`, this specifies the name of the variable with the step length. Default is ‘steplength‘.

Details

The data frame supplied with the ‘data‘ argument has to be in long format, where X is a time-constant covariate and Z(t) is a time-dependent covariate:

idvar	timevar	statevar	X	Z(t)
1	0	A	x_1	z_1(0)
1	1	A	x_1	z_1(1)
1	2	B	x_1	z_1(2)
1	3	A	x_1	z_1(3)
2	0	B	x_2	z_2(0)
2	1	A	x_2	z_2(1)
...	...	...	...	...

If it is not in long format it has to be reshaped. The state variable should provide the states as character strings or numbers; factors are not supported.

‘dtms_format‘ turns the data set above into a data frame in transition format:

id	time	fromvar	tovar	X	Z(t)
1	0	A	A	x_1	z_1(0)

1	1	A	B	x_1	z_1(1)
1	2	B	A	x_1	z_1(2)
2	0	B	A	x_2	z_2(0)
...	...	...	...	...	...

Covariates do not need to be specified and are handled implicitly. The transition from time t to t+1 takes covariate values from time t. By default the variable names of the ID variable and the time variable are changed to 'id' and 'time', as the other functions of the package use these as default names. If renaming the variables is not possible because these variable names already appear in the data then the original names are used. If 'keepnames=TRUE' the original names for 'id' and 'time' are kept.

'dtms_format' by default drops gaps in the data, as no transitions are observed. For instance, in the following example there is no observation at time 4, and thus no transition is observed from t=3 to t=4; and no transition from t=4 to t=5:

idvar	timevar	statevar
1	0	A
1	1	A
1	2	B
1	3	A
1	5	A

In this example, 'dtms_format' will return the following:

id	time	from	to
1	0	A	A
1	1	A	B
1	2	B	A

If 'fill=TRUE', then 'dtms_format' will return the following:

id	time	from	to
1	0	A	A
1	1	A	B
1	2	B	A
1	3	A	NA
1	4	NA	A

The argument `absorbing` controls if the first observed absorbing state is carried over to later observations. For instance, if a unit is first observed to be in transient state 'A' at time 1, then in absorbing state 'X' at time 2, and then in transient state 'A' at time 3, `absorbing=TRUE` will lead to replacement of the state at time 3 with 'X'.

Value

A data set reshaped to transition format

See Also

dtms_data_summary to summarize data in transition format. dtms_censoring for descriptive information on censoring. dtms_clean for fast data cleaning.

Examples

```
# Define model
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:20)
# Transiton format
estdata <- dtms_format(data=simpledata,
dtms=simple,
idvar="id",
timevar="time",
statevar="state")
```

---

dtms_forward *Carry states forward*

Description

This function carries a state forward after its first occurrence.

Usage

```
dtms_forward(
  data,
  state,
  fromvar = "from",
  tovar = "to",
  statevar = NULL,
  idvar = "id",
  timevar = "time",
  dtms = NULL,
  overwrite = "missing",
  vector = FALSE
)
```

Arguments

data	A data frame in long format.
state	Character, name of the state to be carried forward.
fromvar	Character (optional), name of variable with starting state. Default is 'from'.
tovar	Character (optional), name of variable with receiving state. Default is 'to'.
statevar	Character (optional), name of the variable in the data frame in long format with the states. Default is NULL.
idvar	Character (optional), name of variable with unit ID. Default is 'id'.
timevar	Character (optional), name of variable with time scale. Default is 'time'.
dtms	dtms object (optional), as created with dtms. Not required if 'overwrite==transient'.
overwrite	Character (optional), one of 'transient', 'missing', 'absorbing', and 'all', see details. Default is 'transient'.
vector	Logical (optional), return vector (if TRUE) or data frame (if FALSE). Default is FALSE. Argument is only used if argument 'statevar' is specified.

Details

This function carries a state forward after its first occurrence. For instance, carrying the state "A" forward in the sequence 'B, B, A, B, B' will give the sequence 'B, B, A, A, A'. The sequence 'C, B, C, A, B, A, A, B' will give 'C, B, C, A, A, A, A, A'.

This function works with data frames in transition format and in long format. The default is transition format, using the arguments 'fromvar' and 'tovar'. If, however, the argument 'statevar' is specified, it is used instead.

The argument 'overwrite' is used to control what type of information is replaced. If 'overwrite==transient', then only transient states are replaced while missing values and absorbing states remain unchanged. For example, carrying forward state "A" in the sequence 'B, B, A, B, NA, X, X' with X being an absorbing state will give 'B, B, A, A, NA, X, X'. If 'overwrite==missing' then in addition to transient states also missing values are replaced and for the example sequence 'B, B, A, A, A, X, X' would be returned. If 'overwrite==absorbing' then in addition to transient states absorbing states will be replaced; for the example sequence the result would be 'B, B, A, A, NA, A, A'. Finally, if 'overwrite==all' then all values in the sequence will be replaced: 'B, B, A, A, A, A, A'.

Value

The data frame specified with 'data' and the edited state variable (if 'vector=FALSE') or a vector (if 'vector=TRUE').

See Also

dtms_backward to carry states backward.

Examples

```
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:19)

dtms_forward(data=simpledata,
             statevar="state",
             state="A",
             dtms=simple,
             overwrite="transient")
```

`dtms_fullfit` *Estimate unconstrained discrete-time multistate model*

Description

This function estimates an unconstrained discrete-time multistate model using multinomial logistic regression. This is achieved by interacting the starting state with all predictors in the model. It is a wrapper for `dtms_fit()` with 'full=TRUE' and otherwise slightly less arguments.

Usage

```
dtms_fullfit(
  data,
  controls = NULL,
  formula = NULL,
  weights = NULL,
  fromvar = "from",
  tovar = "to",
  reference = 1,
  package = "VGAM",
  ...
)
```

Arguments

`data`	Data frame in transition format, as created with `dtms_format`.
`controls`	Character (optional), names of control variables
`formula`	Formula (optional). If no formula is specified, it will be build from the information specified with controls, fromvar, tovar, and timevar.
`weights`	Character (optional), name of variable in 'data' with survey weights.
`fromvar`	Character (optional), name of variable in 'data' with starting state. Default is "from".
`tovar`	Character (optional), name of variable in 'data' with receiving state. Default is "to".

reference	Numeric or character (optional). Reference level of multinomial logistic regression.
package	Character, chooses package for multinomial logistic regression, currently 'VGAM', 'nnet', and 'mclogit' are supported. Default is 'VGAM'.
...	Further arguments passed to estimation functions.

Value

Returns an object with class depending on the package used.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fullfit(data=estdata)
```

---

dtms_last *Calculate the distribution of the time until a subset of states is left for the last time.*

Description

Calculates the distribution of the until a subset of states is left for the very last time.

Usage

```
dtms_last(
  probs = NULL,
  matrix = NULL,
  dtms,
  risk,
  risk_to = NULL,
  start_time = NULL,
  start_state = NULL,
  start_distr = NULL,
  end_time = NULL,
```

```
  method = "mid",
  total = TRUE,
  rescale = TRUE
)
```

Arguments

probs	Data frame with transition probabilities, as created with dtms_transitions.
matrix	Matrix with transition probabilities, as generated with dtms_matrix.
dtms	dtms object, as created with dtms.
risk	Character, name of state(s) for which risk is of interest.
risk_to	Character (optional), names of one or several states to which the states specified in 'risk' are left. See details.
start_time	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
start_state	Character (optional), name of starting states. If NULL (default) all transient states will be used.
start_distr	Numeric (optional), distribution of starting states. If specified, average distribution over all starting states will be calculated.
end_time	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.
method	Character (optional), do transitions happen mid-interval ('mid', default) or at the end of the interval ('end'), see details.
total	Logical, should total of distribution be shown (always sums to 1)? Default is FALSE.
rescale	Logical (optional), should distribution be rescaled to sum to 1? See details. Default is TRUE.

Details

The resulting distribution is conditional on ever experiencing the final exit, as the waiting time otherwise is not a finite number. The argument 'rescale' can be used to control whether the distribution is rescaled to sum to 1; it usually will do without rescaling.

The state(s) which count to the time are specified with the argument 'risk'. If several states are specified, the resulting distribution refers to the lifetime spent in any of the specified states. The optional argument 'risk_to' can be used to restrict results to exits from the set 'risk' to another specific subset defined by 'risk_to'; i.e., this way, not all transitions out of 'risk' count for the final exit, but only those to specific states.

In a discrete-time model, the time spent in a state depends on assumptions about when transitions happen. Currently, this functions supports two variants which can be specified with the argument 'method': mid-interval transitions can be selected with the option 'mid' and imply that transitions happen at the middle of the time interval; and the option 'end' assumes that instead transitions happen at the end of the interval. In this latter case the distribution of the time spent in a state is equivalent to the number of visits to that state. The calculation takes the step length of the time

scale into account as specified by the ‘dtms‘ object. If the step length is not one fixed value, the first entry of ‘dtms$timestep‘ will be used.

If a distribution of the starting states is provided with ‘start_distr‘ the output table has two additional rows. One shows the distribution unconditional on the starting state. The other shows the distribution conditional on not starting in any state of the risk set.

The distribution of partial waiting times can be generated using the arguments ‘start_state‘ and ‘start_time‘ in combination with ‘end_time‘.

Value

Matrix with the distribution(s) of the waiting time.

See Also

dtms_distr_summary to help with summarizing the resulting distribution.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs <- dtms_transitions(dtms=simple,
                          model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## First visit
dtms_last(dtms=simple,
          probs=probs,
          risk="A",
          start_distr=S)
```

`dtms_matrix` *Creates a transition matrix from transition probabilities*

Description

This function creates a transition matrix based on transition probabilities predicted using the function ‘dtms_transitions‘.

Usage

```
dtms_matrix(
  probs,
  dtms = NULL,
  fromvar = "from",
  tovar = "to",
  Pvar = "P",
  enforcedeath = TRUE,
  rescale = TRUE,
  reshapesep = ":"
)
```

Arguments

`probs`	Data frame with transition probabilities, as created with `dtms_transitions`.
`dtms`	dtms object, as created with `dtms`.
`fromvar`	Character (optional), name of variable in ‘probs‘ with starting state. Default is "from".
`tovar`	Character (optional), name of variable in ‘probs‘ with receiving state. Default is "to".
`Pvar`	Character (optional), name of variable in ‘probs‘ with transition probabilities. Default is ‘P‘.
`enforcedeath`	Logical (optional), make sure that every unit moves to absorbing state after last value of time scale? Default is TRUE.
`rescale`	Logical (optional), rescale transition probabilities to sum to 1? Default is TRUE.
`reshapesep`	Character (optional), used in re-arranging the transition probabilities; should not appear in any state name. Default is ‘:‘.

Value

Returns a transition matrix.

Examples

```
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get transition matrix
Tp <- dtms_matrix(dtms=simple,
                  probs=probs)
```

`dtms_nonparametric` *Nonparametric estimates of transition probabilities*

Description

This function calculates nonparametric estimates of transition probabilities for each value of the time scale.

Usage

```
dtms_nonparametric(
  data,
  dtms,
  fromvar = "from",
  tovar = "to",
  timevar = "time",
  Pvar = "P",
  weights = NULL,
  se = TRUE,
  ci = FALSE,
  alpha = 0.05
)
```

Arguments

`data`	Data frame in transition format, as created with `dtms_format`.

dtms	dtms object, as created with dtms.
fromvar	Character (optional), name of variable with starting state in 'data'. Default is 'from'.
tovar	Character (optional), name of variable with receiving state in 'data'. Default is 'to'.
timevar	Character (optional), name of variable with time scale in 'data'. Default is 'time'.
Pvar	Character (optional), name of variable with transition probabilities in the returned data frame. Default is 'P'.
weights	Character (optional). Name of variable with survey weights.
se	Logical (optional), return standard errors of predicted probabilities. Default is 'TRUE'.
ci	Logical (optional), return confidence intervals? See details. Default is FALSE.
alpha	Numeric (optional), if ci=TRUE, what confidence level is used? Default is 0.05.

Details

The argument 'data' takes a data set in transition format. Predicted transition probabilities are returned as a data frame. Standard errors are approximated using binomial standard errors. In case of small cell counts this might be inaccurate.

Value

A data frame with transition probabilities.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Nonparametric transition probabilities
probs <- dtms_nonparametric(data=estdata,
                            dtms=simple)
```

`dtms_occurrence` *Generate variable with number of occurrence of state*

Description

This function creates a variable which measures the number of occurrence of the states.

Usage

```
dtms_occurrence(
  data,
  dtms,
  newname = "occurrence",
  statevar = "state",
  idvar = "id",
  timevar = "time",
  ignoreleft = TRUE,
  vector = FALSE
)
```

Arguments

`data`	A data frame in long format.
`dtms`	dtms object, as created with `dtms`.
`newname`	Character (optional), name of new variable if data set is returned. Default is "duration".
`statevar`	Character (optional), name of the variable in the data frame with the states. Default is 'state'.
`idvar`	Character (optional), name of variable with unit ID. Default is 'id'.
`timevar`	Character (optional), name of variable with time scale. Default is 'time'.
`ignoreleft`	Logical (optional), ignore left censoring and start counting at the first observation? Default is TRUE.
`vector`	Logical (optional), return vector (if TRUE) or data frame (if FALSE). Default is FALSE

Details

Counting starts with 1 and the first occurrence of a state. For instance, if for an unit the sequence of states A, A, A, B, B, A, C is observed, the occurrence variable would include 1, 1, 1, 1, 1, 2, 1.

The argument 'ignoreleft' controls how left censoring is handled; i.e., what happens when for a unit there are no observations at the beginning of the time scale. If 'TRUE', left censoring is ignored, and counting starts at the first observation for a unit. For instance, if the time scale starts at t=0, but the first observation for a unit is at time t=2, and the sequence of states is again A, A, A, B, B, A, C, then 'ignoreleft=TRUE' returns 1, 2, 3, 1, 2, 1, 1. If 'ignoreleft=FALSE', then the function would return NA, NA, NA, NA, NA, NA, NA for this unit.

The function handles gaps in the data by setting all further occurrences to NA. For #' instance, if a unit is observed at times 1, 2, 4, 5, and 6, but not at time #' 3, and the states are A, A, B, C, C, then the occurrence variable will have the values 1, 1, NA, NA, NA. Note that in this case it would be possible to return 1, 1, NA, 1, NA, NA, but the function currently does not have this capability.

Value

The data frame specified with 'data' with an additional column (if 'vector=FALSE') or a vector (if 'vector=TRUE').

Examples

```
simple <- dtms(transient=c("A","B"),
absorbing="X",
timescale=0:19)

dtms_occurrence(data=simpledata,
                dtms=simple)
```

`dtms_plot` *Plotting transition probabilities*

Description

A simple function for plotting transition probabilities with base R. This is fast, but it is much easier to produce nicer looking results with dtms_simplify.

Usage

```
dtms_plot(
  probs,
  dtms,
  fromvar = "from",
  tovar = "to",
  timevar = "time",
  Pvar = "P",
  ...
)
```

Arguments

`probs`	Object with transition probabilities as created with `dtms_transitions`.
`dtms`	dtms object, as created with `dtms`.
`fromvar`	Character (optional), name of variable in 'probs' with starting state. Default is 'from'.
`tovar`	Character (optional), name of variable in 'probs' with receiving state. Default is 'to'.

timevar	Character (optional), name of variable in ‘probs‘ with time scale. Default is ‘time‘.
Pvar	Character (optional), name of variable in ‘probs‘ with transition probabilities. Default is ‘P‘.
...	Further arguments passed to plot().

Value

No return value, called for side effects

Examples

```
## Model setup
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Plot
dtms_plot(probs=probs,
          dtms=simple)
```

---

dtms_probs_summary *Summarize transition probabilities*

Description

Provides several summary statistics on transition probabilities.

Usage

```
dtms_probs_summary(
  probs,
  fromvar = "from",
  tovar = "to",
  timevar = "time",
```

```
  Pvar = "P",
  digits = 4,
  format = "decimal",
  sep = "_"
)
```

Arguments

probs	Object with transition probabilities as created with dtms_transitions.
fromvar	Character (optional), name of variable with starting state in 'probs'. Default is 'from'.
tovar	Character (optional), name of variable with receiving state in 'probs'. Default is 'to'.
timevar	Character (optional), name of variable with time scale in 'probs'. Default is 'time'.
Pvar	Character (optional), name of variable with transition probabilities in 'probs'. Default is 'P'.
digits	Numeric (optional), number of digits to return, default is 6.
format	Character (optional), show results in decimal format or percentage, either 'decimal' or 'percent'. Default is 'decimal'.
sep	Character (optional), separator between short state name and value of time scale. Default is '_'.

Value

A data frame

Examples

```
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
fit <- dtms_fit(data=estdata)
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
summary(probs)
```

dtms_reward *Markov chain with rewards*

Description

This function calculates the expected rewards by starting state in a Markov chain with rewards.

Usage

```
dtms_reward(probs = NULL, matrix = NULL, reward, dtms)
```

Arguments

probs	Data frame with transition probabilities, as created with dtms_transitions.
matrix	Matrix with transition probabilities, as generated with dtms_matrix.
reward	Matrix with rewards, has to be of same dimensions as ‘matrix‘ or the matrix which will result from ‘probs‘.
dtms	dtms object, as created with dtms.

Value

A matrix with expected rewards.

See Also

[dtms_rewardmatrix()]

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get transition matrix
```

```
Tp <- dtms_matrix(dtms=simple,
                  probs=probs)
## Reward matrix
Rw <- diag(1,dim(Tp)[1])
## State expectancies
dtms_reward(dtms=simple,
            matrix=Tp,
            reward=Rw)
```

`dtms_rewardmatrix` *Generate the reward matrix for a Markov chain with rewards*

Description

This function generates a reward matrix which can be used with `dtms_reward`.

Usage

```
dtms_rewardmatrix(
  dtms,
  starting = NULL,
  receiving,
  reward,
  start_time = NULL,
  end_time = NULL
)
```

Arguments

`dtms`	dtms object, as created with `dtms`.
`starting`	Character (optional), name or names of starting states. If NULL (default) any transition to the state or states specified with `receiving` will get the reward.
`receiving`	Character, name or names of states to which transitioning generates the reward. Can be both transient or absorbing states.
`reward`	Numeric, reward value to be placed in matrix.
`start_time`	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
`end_time`	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.

Value

A matrix with rewards.

See Also

[dtms_reward()]

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
dtms_rewardmatrix(dtms=simple,receiving="B",reward=0.3)
```

`dtms_risk` *Calculate the lifetime risk of ever reaching a state*

Description

The function 'dtms_risk' calculates the (partial) lifetime risk of ever reaching a state specified with the argument 'risk'.

Usage

```
dtms_risk(
  probs = NULL,
  matrix = NULL,
  risk,
  dtms,
  start_distr = NULL,
  start_state = NULL,
  start_time = NULL,
  end_time = NULL
)
```

Arguments

`probs`	Data frame with transition probabilities, as created with `dtms_transitions`.
`matrix`	Matrix with transition probabilities, as generated with `dtms_matrix`.
`risk`	Character, name of state(s) for which risk is of interest.
`dtms`	dtms object, as created with `dtms`.
`start_distr`	Numeric (optional), distribution of starting states. If specified, average distribution over all starting states will be calculated.
`start_state`	Character (optional), name of starting states. If NULL (default) all transient states will be used.
`start_time`	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
`end_time`	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.

Value

Probability of ever reaching state 'risk'.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
# Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## Lifetime risk
dtms_risk(dtms=simple,
          probs=probs,
          risk="A")
```

`dtms_simplify` *Simplify state names*

Description

This function turns long state names into short state names. It is particularly useful for plotting and when used in pipes, see the example.

Usage

```
dtms_simplify(probs, fromvar = "from", tovar = "to", sep = "_")
```

Arguments

`probs`	Object with transition probabilities as created with `dtms_transitions`.
`fromvar`	Character (optional), name of variable in 'probs' with starting state. Default is 'from'.
`tovar`	Character (optional), name of variable in 'probs' with receiving state. Default is 'to'.
`sep`	Character (optional), separator between short state name and value of time scale. Default is '_'.

Value

Data frame

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Simplify
probs |>  dtms_simplify()
## NOT RUN: requires ggplot2
# library(ggplot2)
# probs |>  dtms_simplify() |>
#   ggplot(aes(x=time,y=P,color=to)) +
#   geom_line() +
#   facet_wrap(~from)
```

---

dtms_simulate *Simulation of Markov chains*

Description

This function simulates trajectories based on a Markov chain using the ‘markovchain‘ package.

Usage

```
dtms_simulate(
  probs = NULL,
  matrix = NULL,
  dtms,
  size = 100,
  start_distr = NULL,
  droplast = TRUE,
  varnames = "T_"
)
```

Arguments

probs	Data frame with transition probabilities, as created with dtms_transitions.
matrix	Matrix, a matrix of transition probabilities as created with dtms_matrix(),
dtms	dtms object, as created with dtms.
size	Numeric, number of trajectories which will be simulated. Default is 100.
start_distr	Numeric (optional), distribution of starting states. If NULL, starting states will be assumed to be equally distributed.
droplast	Logical (optional), drop final time step after the time scale in which every unit is absorbed? Default is TRUE.
varnames	Character (optional), suffix for variable names in simulated data. Will be pasted with values of the timescale. Default is "T_".

Value

A data frame with simulated trajectories in wide format.

Examples

```
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:19)
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
fit <- dtms_fit(data=estdata,package="mclogit")
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
dtms_simulate(probs=probs,dtms=simple)
```

---

dtms_source *Source code only from specific lines*

Description

This is a helper function which allows to source only specific lines from a file.

Usage

```
dtms_source(file, start, end, local = FALSE)
```

Arguments

file	Name of the file to read (character).
start	First line to read (numeric).
end	Last line to read (numeric).
local	Logical or an environment. Default is 'FALSE'. See function source for details.

Value

No return value, called for side effects

Source

User Matthew Plourde on Stackoverflow https://stackoverflow.com/questions/12214963/source-only-part-of-a-file

See Also

[func(source)]

---

dtms_start *Tabulate starting distribution*

Description

Tabulates the starting distribution.

Usage

```
dtms_start(
  data,
  dtms,
  variables = NULL,
  start_state = NULL,
  start_time = NULL,
  fromvar = "from",
  timevar = "time",
  weights = NULL
)
```

Arguments

data	Data frame in transition format, as created with dtms_format.
dtms	dtms object, as created with dtms.
variables	List (optional), a named list with covariate values which are used to restrict the data.

start_state	Character (optional), name of starting states. If NULL (default) all transient states will be used.
start_time	Numeric (optional), value of time scale for start. If several values are specified, the average distribution over all these values is calculated. In this case the first value specified with this argument is used to construct the long state name. If NULL (default) first value of time scale will be used.
fromvar	Character (optional), name of variable in 'data' with starting state. Default is 'from'.
timevar	Character (optional), name of variable in 'data' with time scale. Default is 'time'.
weights	Character (optional). Name of variable with survey weights.

Details

Per default, the starting distribution is the distribution of transient states at the first value of the time scale in the data. This can be changed to any value of the time scale, and any set of states. The distribution can also be conditional on further covariate values which can be specified with the argument 'variables'.

'variables' takes a named list where each entry of the list is named like the corresponding variable and with the values to be selected.

Value

Returns a table of the starting distribution.

Examples

```
work <- dtms(transient=c("Working","Non-working","Retired"),
             absorbing="Dead",
             timescale=50:99)
## Reshape
estdata <- dtms_format(data=workdata,
                       dtms=work,
                       idvar="ID",
                       timevar="Age",
                       statevar="State")
## Drop dead-to-dead transitions etc
estdata <- dtms_clean(data=estdata,
                      dtms=work)
## Starting distributions
# Men
Sm <- dtms_start(dtms=work,
                 data=estdata,
                 variables=list(Gender=0))
# Women
Sw <- dtms_start(dtms=work,
                 data=estdata,
                 variables=list(Gender=1))
```

dtms_survivor *Calculate the survivorship function*

Description

Calculates the proportion of units surviving up to certain values of the time scale.

Usage

```
dtms_survivor(
  probs = NULL,
  matrix = NULL,
  dtms,
  start_distr = NULL,
  start_state = NULL,
  start_time = NULL,
  end_time = NULL
)
```

Arguments

probs	Data frame with transition probabilities, as created with dtms_transitions.
matrix	Matrix with transition probabilities, as generated with dtms_matrix.
dtms	dtms object, as created with dtms.
start_distr	Numeric (optional), distribution of starting states. If specified, average distribution over all starting states will be calculated.
start_state	Character (optional), name of starting states. If NULL (default) all transient states will be used.
start_time	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
end_time	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.

Details

If a distribution of the starting states is provided with 'start_distr' the output table has an additional row, showing the waiting time distribution unconditional on the starting state.

Value

A table with the survivorship function.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## Distribution of visits
dtms_survivor(dtms=simple,
              probs=probs,
              start_distr=S)
```

`dtms_transitions` *Predict transition probabilities*

Description

'dtms_transitions()' predicts transition probabilities based on a model estimated with 'dtms_fit()'.

Usage

```
dtms_transitions(
  model,
  dtms,
  controls = NULL,
  dropvar = TRUE,
  timevar = "time",
  fromvar = "from",
  tovar = "to",
  Pvar = "P",
  se = TRUE,
  vcov = FALSE,
  ci = FALSE,
  alpha = 0.05
)
```

Arguments

model	Model estimated with dtms_fit.
dtms	dtms object, as created with dtms.
controls	List (optional) with values for predictors (see details).
dropvar	Logical (optional), should covariate values used for prediction be returned (see details). Default is 'TRUE'.
timevar	Character (optional), name of variable with time scale in the returned data frame. Default is 'time'.
fromvar	Character (optional), name of variable with starting state in the returned data frame. Default is 'from'.
tovar	Character (optional), name of variable with receiving state in the returned data frame. Default is 'to'.
Pvar	Character (optional), name of variable with transition probabilities in the returned data frame. Default is 'P'.
se	Logical (optional), return standard errors of predicted probabilities. Default is 'TRUE'.
vcov	Logical (optional), return variance-covariance matrix of predicted probabilities. Default is 'FALSE'.
ci	Logical (optional), return confidence intervals? See details. Default is FALSE.
alpha	Numeric (optional), if ci=TRUE, what confidence level is used? Default is 0.05.

Details

Depending on the model specification, the prediction of transition probabilities will require values for predictor variables which can be specified with the argument 'controls'. This is done using a named list where each entry name must correspond to a variable name in the model. For time-constant variables, each list entry is of length one and provides a value for the corresponding time-constant variable. For time-varying variables, each entry must have the length of the time scale minus one, and provide a value for each (potential) transition in the model; i.e., starting from time t=0, starting from time t=1, etc., until time t=T-1. Alternatively, it can be of the same length as the time scale; in this case, the last value is dismissed.

If 'vcov=TRUE' the full variance-covariance matrix of the transition probabilities will be returned instead of the transition probabilities. If 'ci=TRUE', confidence intervals will be returned. Note that the calculation uses a normal approximation and results below 0 or above 1 are possible.

The argument 'dropvar' controls whether the covariate values used for prediction are dropped. If 'FALSE' each row of the resulting data frame will have the covariate values which were used to predict the corresponding probability.

Value

A data frame with transition probabilities.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
                       idvar="id",
                       timevar="time",
                       statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
```

`dtms_visits` *Calculate the distribution of the time spent in a subset of states*

Description

Calculates the distribution of the time spent in a state or a subset of states.

Usage

```
dtms_visits(
  probs = NULL,
  matrix = NULL,
  dtms,
  risk,
  start_time = NULL,
  start_state = NULL,
  start_distr = NULL,
  end_time = NULL,
  method = "mid",
  total = FALSE
)
```

Arguments

`probs`	Data frame with transition probabilities, as created with `dtms_transitions`.
`matrix`	Matrix with transition probabilities, as generated with `dtms_matrix`.
`dtms`	dtms object, as created with `dtms`.

risk	Character, name of state(s) for which risk is of interest.
start_time	Numeric (optional), value of time scale for start. If NULL (default) first value of time scale will be used.
start_state	Character (optional), name of starting states. If NULL (default) all transient states will be used.
start_distr	Numeric (optional), distribution of starting states. If specified, average distribution over all starting states will be calculated.
end_time	Numeric (optional), last value of time scale to consider. If NULL (default) all values of time scale starting from start_time will be used.
method	Character (optional), do transitions happen mid-interval (‘mid‘, default) or at the end of the interval (‘end‘), see details.
total	Logical, should total of distribution be shown (always sums to 1)? Default is FALSE.

Details

The state(s) which count to the time are specified with the argument ‘risk‘. If several states are specified, the resulting distribution refers to the lifetime spent in any of the specified states.

In a discrete-time model, the time spent in a state depends on assumptions about when transitions happen. Currently, this functions supports two variants which can be specified with the argument ‘method‘: mid-interval transitions can be selected with the option ‘mid‘ and imply that transitions happen at the middle of the time interval; and the option ‘end‘ assumes that instead transitions happen at the end of the interval. In this latter case the distribution of the time spent in a state is equivalent to the number of visits to that state. The calculation takes the step length of the time scale into account as specified by the ‘dtms‘ object. If the step length is not one fixed value, the first entry of ‘dtms$timestep‘ will be used.

If a distribution of the starting states is provided with ‘start_distr‘ the output table has two additional rows. One shows the distribution unconditional on the starting state. The other shows the distribution conditional on not starting in any state of the risk set.

Value

A table with the distribution of time spent in a subset of states.

See Also

dtms_distr_summary to help with summarizing the resulting distribution.

Examples

```
## Define model: Absorbing and transient states, time scale
simple <- dtms(transient=c("A","B"),
               absorbing="X",
               timescale=0:20)
## Reshape to transition format
estdata <- dtms_format(data=simpledata,
                       dtms=simple,
```

```
                      idvar="id",
                      timevar="time",
                      statevar="state")
## Clean
estdata <- dtms_clean(data=estdata,
                      dtms=simple)
## Fit model
fit <- dtms_fit(data=estdata)
## Predict probabilities
probs    <- dtms_transitions(dtms=simple,
                             model = fit)
## Get starting distribution
S <- dtms_start(dtms=simple,
                data=estdata)
## Distribution of visits
dtms_visits(dtms=simple,
            probs=probs,
            risk="A",
            start_distr=S,
            total=TRUE)
```

`simpledata` *simpledata: an artificial data set with abstract trajectories*

Description

An artificial data set with abstract states and time scale. The state space consists of two transient states (A,B) and one absorbing state (X).

Usage

```
simpledata
```

Format

'simpledata' A data frame with 12,179 rows and 3 columns:

id Identifier of the units

time Time scale

state The state occupied by an unit at a given value of the time scale

workdata *workdata: simulated working trajectories*

Description

A simulated data set of individuals' working trajectories during late working life and retirement age. The state space consists of three transient states (working; retired; not working) and one absorbing state (dead). The age range covers ages 50 to 99. The data is simulated using transition probabilities published as part of Dudel & Myrskylä (2017).

Usage

```
workdata
```

Format

'workdata' A data frame with 250,000 rows and 4 columns:

ID Person identifier

Gender Individuals' gender (0=men, 1=women)

Age Age, the time scale of this example

State The state occupied by an unit at a given age

Source

<https://doi.org/10.1007/s13524-017-0619-6>

Index