Package ‘msmtools’

June 8, 2017

Type Package

Title Building Augmented Data to Run Multi-State Models with 'msm' Package

Version 1.3

Date 2017-06-08

Description A fast and general method for restructuring classical longitudinal data into augmented ones. The reason for this is to facilitate the modeling of longitudinal data under a multi-state framework using the 'msm' package.

URL https://github.com/contefranz/msmtools

BugReports https://github.com/contefranz/msmtools/issues

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

Depends R (>= 3.0)

Imports data.table (>= 1.9.6), msm (>= 1.6), survival (>= 2.38.0)

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Francesco Grossetti [aut, cre]

Maintainer Francesco Grossetti <francesco.grossetti@unibocconi.it>

Repository CRAN

Date/Publication 2017-06-08 07:10:21 UTC

R topics documented:

`augment` *A fast and general method for building augmented data*

Description

A fast and general method for reshaping standard longitudinal data into a new structure called augmented'. This format is suitable under a multi-state framework using the `msm` package.

Usage

```
augment(data, data_key, n_events, pattern, state = list("IN", "OUT", "DEAD"),
  t_start, t_end, t_cens, t_death, t_augmented, more_status, check_NA = FALSE,
  convert = FALSE, verbose = TRUE)
```

Arguments

`data`	A `data.table` or `data.frame` object in longitudinal format where each row represents an observation in which the exact starting and ending time of the process are known and recorded. If `data` is a `data.frame`, then `augment` internally casts it to a `data.table`.
`data_key`	A keying variable which `augment` uses to define a key for `data`. This represents the subject ID (see `setkey`).
`n_events`	An integer variable indicating the progressive (monotonic) event number of a given ID. `augment` always checks whether `n_events` is monotonic increasing within the provided `data_key` and stops the execution in case the check fails (see 'Details'). If missing, `augment` fastly creates a variable named `"n_events"`.
`pattern`	Either an integer, a factor or a characer with 2 or 3 unique values which provides the ID status at the end of the study. `pattern` has a predefined structure. When 2 values are detected, they must be in the format: 0 = "alive", 1 = "dead". When 3 values are detected, then the format must be: 0 = "alive", 1 = "dead during a transition", 2 = "dead after a transition has ended" (see 'Details').
`state`	A list of three and exactly three possible states which a subject can reach. `state` has a predefined structure as follows: IN, OUT, DEAD (see 'Details').
`t_start`	The starting time of an observation. It can be passed as date, integer, or numeric format.
`t_end`	The ending time of an observation. It can be passed as date, integer, or numeric format.
`t_cens`	The censoring time of the study. This is the date until each ID is observed, if still active in the cohort.
`t_death`	The exact death time of a subject ID. If `t_death` is missing, `t_cens` is assumed to contain both censoring and death times and a warning is raised.

t_augmented
: A variable indicating the name of the new time variable of the process in the augmented format. If `t_augmented` is missing, then the default name 'augmented' is assumed and the corresponding new variable is added to `data`. `t_augmented` is cast to integer or to numeric depending whether `t_start` is a date or a difftime, respectively. The suffix '_int' or '_num' is pasted to `t_augmented` and a new variable is computed accordingly. This is done because `msm` can't correclty deal with date or difftime variables. Both variables are positioned before `t_start`.

more_status
: A variable which marks further transitions beside the default ones given by `state`. `more_status` can be a factor or a character (see 'Details'). If missing, `augment` ignores it.

check_NA
: If `TRUE`, then arguments `data_key`, `n_events`, `pattern`, `t_start` and `t_end` are looked up for any missing data and if the function finds any, it stops with error. Default is `FALSE` because `augment` is not intended for running consistency checks, beside what is mandatory, and because the procedure is computationally onerous and could cause memory overhead for very large datasets. Argument `more_status` is the only one for which `augment` always checks for the presence of missing data and, again, if it finds any it just stops with error.

convert
: If `TRUE`, then the returned object is automatically converted to the class `data.frame`. This is done in place and comes at very low cost both from running time and memory consumption (see `setDF`).

verbose
: If `FALSE`, all information produced by `print`, `cat` and `message` are suppressed. Default is `TRUE`.

Details

In order to get the data processed, a monotonic increasing process needs to be ensured. In the first place, `augment` checks this both in case `n_events` is missing or not. The data are fastly ordered through `setkey` function with `data_key` as the primary key and `t_start` as the secondary key. In the second place, it checks the monotonicity of `n_events` and if it fails, it stops with error and returns the subjects gived by `data_key` for whom the condition is not met. If `n_events` is missing, then `augment` internally computes the progression number with the name *n_events* and runs the same procedure.

Attention needs to be payed to argument `pattern`. Integer values can be 0 and 1 if only two status are defined and they must correspond to the status 'alive' and 'dead'. If three values are defined, then they must be 0, 1 and 2 if `pattern` is an integer, or 'alive', 'dead inside a transition' and dead outside a transition' if `pattern` is either a character or a factor. The order matters: it is not possible to specify 0 as 'dead' for instance.

When passing a list of states, the order is important so that the first element must be the state corresponding to the starting time (i.e. 'IN', inside the hospital), the second element must correspond to the ending time (i.e. 'OUT', outside the hospital), and the third state is the absorbing state (i.e. 'DEAD').

`more_status` allows to manage multiple transitions beside what already specified in `state`. In particular, if the corresponding observation is a standard admission which adds no other information than what is inside `state`, then `more_status` must be set to 'df' which stands for 'Default' (see

'Examples' or run ?hosp and look at the variable 'rehab_it'). In general, it is always a good practice to fully specify the transition with a bunch of self-explanatory characters in order to quickly understand which is the current transition.

Value

An augmented format dataset of class `data.table`, or `data.frame` when `convert` is `TRUE`, where each row represents a specific transition for a given subject. `augment` returns them after some important variables have been computed:

`augmented` The new timing variable for the process when looking at transitions. If `t_augmented` is missing, then `augment` creates *augmented* by default. *augmented*. The function looks directly to `t_start` and `t_end` to build it and thus it inherits their class. In particular, if `t_start` is a date format, then `augment` computes a new variable cast as integer and names it *augmented_int*. If `t_start` is a difftime format, then `augment` computes a new variable cast as a numeric and names it *augmented_num*.

`status` A status flag which contains the states as specified in `state`. `augment` automatically checks whether argument `pattern` has 2 or 3 unique values and computes the correct structure of a given subject as reported in the vignette. The variable is cast as character.

`status_num` The corresponding integer version of *status*.

`n_status` A mix of `status` and `n_events` cast as character. This becomes useful when a multi-state model on the progression of the process needs to be implemented.

If `more_status` is passed, then `augment` computes some more variables. They mimic the meaning of *status*, *status_num*, and *n_status* but they account for the more complex structure defined. They are: `status_exp`, `status_exp_num`, and `n_status_exp`.

Author(s)

Francesco Grossetti <francesco.grossetti@unibocconi.it>.

References

Jackson, C.H. (2011). Multi-State Models for Panel Data:
The *msm* Package for R. Journal of Statistical Software, 38(8), 1-29.
URL http://www.jstatsoft.org/v38/i08/.

M. Dowle, A. Srinivasan, T. Short, S. Lianoglou with contributions from R. Saporta and E. Antonyan (2016):
data.table: Extension of *data.frame*. R package version 1.9.6
URL https://github.com/Rdatatable/data.table/wiki

See Also

`data.table` `setkey`

Examples

```
# loading data
data( hosp )

# 1.
# augmenting hosp
hosp_augmented = augment( data = hosp, data_key = subj, n_events = adm_number,
                          pattern = label_3, t_start = dateIN, t_end = dateOUT,
                          t_cens = dateCENS )

# 2.
# augmenting hosp by passing more information regarding transitions
# with argument more_status
hosp_augmented_more = augment( data = hosp, data_key = subj, n_events = adm_number,
                               pattern = label_3, t_start = dateIN, t_end = dateOUT,
                               t_cens = dateCENS, more_status = rehab_it )
# 3.
# augmenting hosp and returning a data.frame
hosp_augmented = augment( data = hosp, data_key = subj, n_events = adm_number,
                          pattern = label_3, t_start = dateIN, t_end = dateOUT,
                          t_cens = dateCENS, convert = TRUE )
class( hosp_augmented )
```

---

hosp *Synthetic Hospital Admissions*

Description

A dataset containing synthetic hospital admissions in the classic longitudinal format. The dataset counts imaginary 10 patients who undergo different (re)admission into a hospital. Some demographic and clinical variables are also included.

Usage

```
hosp
```

Format

A `data.table` with 53 rows and 12 variables:

subj Subject ID (integer)

adm_number Hospital admissions counter (integer)

gender Gender of patient (factor with 2 levels: "F" = females, "M" = males)

age Age of patient in years at the given observation (integer)

rehab Rehabilitation flag: if the admission has been in rehabilitation, then rehab = 1, else = 0 (integer)

it Intensive Therapy flag: if the admission has been in intensive therapy, then it = 1, else = 0 (integer)

rehab_it String which in one place marks the hospital admission types based on rehab and it. The standard admission is coded as "df" (default). If admission was in rehabilitation or in intensive therapy, rehab_it = "rehab" or "it", respectively (character)

label_2 Subject status at the end of the study. It takes 2 values: "alive" and "dead" (character)

label_3 Subject status at the end of the study. It takes 3 values: "alive" and "dead_in" and "dead_out" (character)

dateIN Exact admission date (date)

dateOUT Exact discharge date (date)

dateCENS Either censoring time or exact death time (date)

`msmtools` *Building augmented data for multi-state models: the* `msmtools` *package*

Description

`msmtools` introduces a fast and general method for restructuring classical longitudinal datasets into *augmented* ones. Augmented data enhances longitudinal datasets and allow to model each transition under a multi-state framework. `msmtools` works in symbiosis with the `msm` package. It also provides two graphical goodness-of-fit tools to inspect the model performances using survival curves and prevalences under the Markov assumption. `msmtools` comes with 4 functions: `augment`, `polish`, `prevplot`, and `survplot`.

`polish` *Remove observations with different states occurring at the same time*

Description

Fast algorithm to get rid of transitions to different states occurring at the same exact time in an augmented data structure as computed by `augment` (see 'Details').

Usage

```
polish(data, data_key, pattern, time, check_NA = FALSE, convert = FALSE,
  verbose = TRUE)
```

Arguments

- `data` — A `data.table` or `data.frame` object in longitudinal format where each row represents an observation in which the exact starting and ending time of the process are known and recorded. If `data` is a `data.frame`, then `augment` internally casts it to a `data.table`.
- `data_key` — A keying variable which `augment` uses to define a key for `data`. This represents the subject ID (see `setkey`).
- `pattern` — Either an integer, a factor or a characer with 2 or 3 unique values which provides the ID status at the end of the study. `pattern` has a predefined structure. When 2 values are detected, they must be in the format: 0 = "alive", 1 = "dead". When 3 values are detected, then the format must be: 0 = "alive", 1 = "dead during a transition", 2 = "dead after a transition has ended" (see 'Details').
- `time` — The target time variable to check duplicates. By default it is set to 'augmented_int'.
- `check_NA` — If `TRUE`, then arguments `data_key`, `pattern`, and `time` are looked up for any missing data and if the function finds any, it stops with error. Default is `FALSE`.
- `convert` — If `TRUE`, then the returned object is automatically converted to the class `data.frame`. This is done in place and comes at very low cost both from running time and memory consumption (see `setDF`).
- `verbose` — If `FALSE`, all information produced by `print`, `cat` and `message` are suppressed. Default is `TRUE`.

Details

The function finds all those cases where two subsequent events for a given subject land on different states but occur at the same time. When this happens, the whole subject, as identified by `data_key`, is removed from the data. The total number of subjects to be removed is printed out in order to be more informative.

Author(s)

Francesco Grossetti <francesco.grossetti@unibocconi.it>.

See Also

`augment`

Examples

```
# loading data
data( hosp )

# augmenting longitudinal data
hosp_aug = augment( data = hosp, data_key = subj, n_events = adm_number,
                    pattern = label_3, t_start = dateIN, t_end = dateOUT,
                    t_cens = dateCENS )

# cleaning any targeted occurrence
```

```
hosp_aug_clean = polish( data = hosp_aug, data_key = subj, pattern = label_3 )
```

---

prevplot *Plot observed and expected prevalences for a multi-state model*

Description

Provides a graphical indication of goodness of fit of a multi-state model computed by msm using observed and expected prevalences. It also computes a rough indicator of where the data depart from the fitted Markov model.

Usage

```
prevplot(x, prev.obj, M = FALSE, exacttimes = TRUE, ci = FALSE,
  grid = 100L, x.lab.grid = 500L, xlab = "Time",
  ylab = "Prevalence (%)", lty.fit = 1, lwd.fit = 1, col.fit = "red",
  lty.ci.fit = 2, lwd.ci.fit = 1, col.ci.fit = col.fit, lwd.obs = 1,
  lty.obs = 1, col.obs = "darkblue", legend.pos = "topright",
  par.col = 3, plot.width = 10, plot.height = 5, max.m = 0.1,
  devnew = TRUE, verbose = TRUE)
```

Arguments

x	A msm object.
prev.obj	A list computed by prevalence.msm. It can be with or without confidence intervals. prevplot will behaves accordingly.
M	If TRUE, then a rough indicator of deviance from the model is computed (see 'Details'). Default is FALSE.
exacttimes	If TRUE (default) then transition times are known and exact. This is inherited from msm and should be set the same way.
ci	If TRUE, then confidence intervals, if they exist, are plotted. Default is FALSE.
grid	Define how many points should be used to build the *x* axis. Defaul is 100.
x.lab.grid	Define the interval on the *x* axis at which draw tick marks. Default is 500.
xlab	*x* axis label.
ylab	*y* axis label.
lty.fit	Line type for the expected prevalences. See par.
lwd.fit	Line width for the expected prevalences. See par.
col.fit	Line color for the expected prevalences. See par.
lty.ci.fit	Line type for the expected prevalences confidence limits. See par.
lwd.ci.fit	Line width for the expected prevalences confidence limits. See par.
col.ci.fit	Line color for the expected prevalences confidence limits. See par.
lwd.obs	Line width for the observed prevalences. See par.

lty.obs	Line type for the observed prevalences. See par.
col.obs	Line color for the observed prevalences. See par.
legend.pos	Where to position the legend. Default is "topright", but *x* and *y* coordinate can be passed. If NULL, then legend is not shown.
par.col	The number of columns of the plot. Default is 3.
plot.width	Width of new graphical device. Default is 7. See par.
plot.height	Height of new graphical device. Default is 7. See par.
max.m	If M = TRUE, it adjusts the upper *y* limit when plotting M.
devnew	Set the graphical device where to plot. By default, prevplot plots on a new device by setting dev.new. If FALSE, then a plot is drawn onto the current device as specified by dev.cur. If FALSE and no external devices are opened, then a plot is drawn using internal graphics. See dev.
verbose	If FALSE, all information produced by print, cat and message are suppressed. Default is TRUE.

Details

When M = TRUE, a rough indicator of the deviance from the Markov model is computed according to Titman and Sharples (2008). A comparison at a given time t_i of a patient *k* in the state *s* between observed counts O_{is} with expected ones E_{is} is build as follows:

$$M_{is} = \frac{(O_{is} - E_{is})^2}{E_{is}}$$

Author(s)

Francesco Grossetti <francesco.grossetti@unibocconi.it>.

References

Titman, A. and Sharples, L.D. (2010). Model diagnostics for multi-state models, *Statistical Methods in Medical Research*, 19, 621-651.

Titman, A. and Sharples, L.D. (2008). A general goodness-of-fit test for Markov and hidden Markov models, *Statistics in Medicine*, 27, 2177-2195.

Gentleman RC, Lawless JF, Lindsey JC, Yan P. (1994). Multi-state Markov models for analysing incomplete disease data with illustrations for HIV disease. *Statistics in Medicine*, 13:805-821.

Jackson, C.H. (2011). Multi-State Models for Panel Data:
The *msm* Package for R. Journal of Statistical Software, 38(8), 1-29.
URL http://www.jstatsoft.org/v38/i08/.

See Also

plot.prevalence.msm msm prevalence.msm

Examples

```
## Not run:
data( hosp )

# augmenting the data
hosp_augmented = augment( data = hosp, data_key = subj, n_events = adm_number,
                          pattern = label_3, t_start = dateIN, t_end = dateOUT,
                          t_cens = dateCENS )

# let's define the initial transition matrix for our model
Qmat = matrix( data = 0, nrow = 3, ncol = 3, byrow = TRUE )
Qmat[ 1, 1:3 ] = 1
Qmat[ 2, 1:3 ] = 1
colnames( Qmat ) = c( 'IN', 'OUT', 'DEAD' )
rownames( Qmat ) = c( 'IN', 'OUT', 'DEAD' )

# attaching the msm package and running the model using
# gender and age as covariates
library( msm )
msm_model = msm( status_num ~ augmented_int, subject = subj,
                 data = hosp_augmented, covariates = ~ gender + age,
                 exacttimes = TRUE, gen.inits = TRUE, qmatrix = Qmat,
                 method = 'BFGS', control = list( fnscale = 6e+05, trace = 0,
                 REPORT = 1, maxit = 10000 ) )

# defining the times at which compute the prevalences
t_min = min( hosp_augmented$augmented_int )
t_max = max( hosp_augmented$augmented_int )
steps = 100L

# computing prevalences
prev = prevalence.msm( msm_model, covariates = 'mean', ci = 'normal',
                       times = seq( t_min, t_max, steps ) )

# and plotting them using prevplot()
prevplot( msm_model, prev, ci = TRUE, devnew = FALSE, verbose = FALSE )

## End(Not run)
```

---

survplot *Plot and get survival data from a multi-state model*

Description

Plot a Kaplan-Meier curve and compare it with the fitted survival probability computed from a msm model. Fast builds and returns the associated datasets.

Usage

```
survplot(x, from = 1, to = NULL, range = NULL, covariates = "mean",
  exacttimes = TRUE, times, grid = 100L, km = FALSE, return.all = FALSE,
  return.km = NULL, return.p = NULL, convert = FALSE, add = FALSE,
  ci = c("none", "normal", "bootstrap"), interp = c("start", "midpoint"),
  B = 100L, legend.pos = "topright", xlab = "Time",
  ylab = "Survival Probability", main = NULL, lty.fit = 1, lwd.fit = 1,
  col.fit = "red", lty.ci.fit = 3, lwd.ci.fit = 1, col.ci.fit = col.fit,
  mark.time = FALSE, lty.km = 5, lwd.km = 1, col.km = "darkblue",
  do.plot = TRUE, plot.width = 7, plot.height = 7, devnew = TRUE,
  verbose = TRUE)
```

Arguments

`x`	A `msm` object.
`from`	State from which to compute the estimated survival. Default to state 1.
`to`	The absorbing state to which compute the estimated survival. Default to the highest state found by `absorbing.msm`.
`range`	A numeric vector of two elements which gives the time range of the plot.
`covariates`	Covariate values for which to evaluate the expected probabilities. These can either be: the string `"mean"`, denoting the means of the covariates in the data (this is the default), the number 0, indicating that all the covariates should be set to zero, or a list of values, with optional names. For example: `list (75, 1)` where the order of the list follows the order of the covariates originally given in the model formula, or a named list: `list (age = 75, gender = "M")`.
`exacttimes`	If `TRUE` (default) then transition times are known and exact. This is inherited from `msm` and should be set the same way.
`times`	An optional numeric vector giving the times at which to compute the fitted survival.
`grid`	An integer which tells at how many points to compute the fitted survival (see 'Details'). If `times` is passed, `grid` is ignored. It has a default of 100 points.
`km`	If `TRUE`, then the Kaplan-Meier curve is shown. Default is `FALSE`.
`return.all`	If `TRUE`, then all the datasets used to draw the plot will be return to the environment. This argument saves you some typing time since you do not have to pass neither `return.km` nor `return.p`. Default is `FALSE` (see 'Details').
`return.km`	If `TRUE`, then the dataset used for building the Kaplan-Meier is returned as an object of class `data.table` unless `convert` is set to `TRUE` (see `convert`). Default is `FALSE`. `survplot` must be assigned to an object in order to get the data in the environment (see 'Details').
`return.p`	If `TRUE`, then the dataset used for building the fitted survival curve is returned as an object of class `data.table` unless `convert` is set to `TRUE` (see `convert`). Default is `FALSE`. `survplot` must be assigned to an object in order to get the data in the environment (see 'Details').

convert	If TRUE, then any returned object is automatically converted to the class data.frame. This is done in place and comes at very low cost both from running time and memory consumption (see setDF).
add	If TRUE, then a new layer is added to the current plot. Default is FALSE.
ci	If "none" (the default), then no confidence intervals are plotted. If "normal" or "bootstrap", confidence intervals are plotted based on the respective method in pmatrix.msm. This is very computationally-intensive, since intervals must be computed at a series of times.
interp	If "start" (the default), then the entry time into the absorbing state is assumed to be the time it is first observed in the data. If "midpoint", then the entry time into the absorbing state is assumed to be halfway between the time it is first observed and the previous observation time. This is generally more reasonable for "progressive" models with observations at arbitrary times.
B	Number of bootstrap or normal replicates for the confidence interval. The default is 100 rather than the usual 1000, since these plots are for rough diagnostic purposes.
legend.pos	Where to position the legend. Default is "topright", but x and y coordinate can be passed. If NULL, then legend is not shown.
xlab	x axis label.
ylab	y axis label.
main	The main title of the plot(s) as character. Default is NULL.
lty.fit	Line type for the fitted curve. See par.
lwd.fit	Line width for the fitted curve. See par.
col.fit	Line color for the fitted curve. See par.
lty.ci.fit	Line type for the fitted curve confidence limits. See par.
lwd.ci.fit	Line width for the fitted curve confidence limits. See par.
col.ci.fit	Line color for the fitted curve confidence limits. See par.
mark.time	Mark the empirical survival curve at each censoring point. See lines.survfit.
lty.km	Line type for the Kaplan-Meier passed to lines.survfit. See par.
lwd.km	Line width for the Kaplan-Meier passed to lines.survfit. See par.
col.km	Line color for the Kaplan-Meier passed to lines.survfit. See par.
do.plot	If FALSE, then no plot is shown at all. Default is TRUE.
plot.width	Width of new graphical device. Default is 7. See par.
plot.height	Height of new graphical device. Default is 7. See par.
devnew	Set the graphical device where to plot. By default, survplot plots on a new device by setting dev.new. If FALSE, then a plot is drawn onto the current device as specified by dev.cur. If FALSE and no external devices are opened, then a plot is drawn using internal graphics. See dev.
verbose	If FALSE, all information produced by print, cat and message are suppressed. Default is TRUE.

Details

The function is a wrapper of `plot.survfit.msm` and does more things. `survplot` manages correctly the plot of a fitted survival in an exact times framework (when `exacttimes = TRUE`) by just resetting the time scale and looking at the follow-up time. It can fastly build and return to the user the datasets used to compute the Kaplan-Meier and the fitted survival by setting `return.all = TRUE`. When this is `TRUE`, setting `return.km` or `return.p` to `FALSE` produces an error and `survplot` does not conclude the job. If these are set to `TRUE`, a warning is raised but the job is taken to the end. For more details about how `survplot` returns objects, please refer to the vignette with `vignette("msmtools")`.

The user can defined custom times (through `times`) or let `survplot` choose them on its own (through `grid`). In the latter case, `survplot` looks for the follow-up time and divides it by `grid`. The higher it is, the finer the grid will be so that computing the fitted survival will take longer, but will be more precise.

Value

If `return.all` is set to `TRUE`, then `survplot` returns a named list with `$km` and `$fitted` as `data.table` or as `data.frame` when `convert = TRUE`. To save them in the working environment, assign `survplot` to an object (see 'Examples').

`$km` contains up to 4 columns:

`subject`	The ordered subject ID as passed in the `msm` function.
`mintime`	The times at which to compute the fitted survival.
`mintime_exact`	If `exacttimes` is `TRUE`, then the relative timing is reported.
`anystate`	State of transition to compute the Kaplan-Meier.

———

`$fitted` contains 2 columns:

`time`	Times at which to compute the fitted survival.
`probs`	The corresponding values of the fitted survival.

Author(s)

Francesco Grossetti <francesco.grossetti@unibocconi.it>.

References

Titman, A. and Sharples, L.D. (2010). Model diagnostics for multi-state models, *Statistical Methods in Medical Research*, 19, 621-651.

Titman, A. and Sharples, L.D. (2008). A general goodness-of-fit test for Markov and hidden Markov models, *Statistics in Medicine*, 27, 2177-2195.

Jackson, C.H. (2011). Multi-State Models for Panel Data:
The *msm* Package for R. Journal of Statistical Software, 38(8), 1-29.
URL http://www.jstatsoft.org/v38/i08/.

See Also

plot.survfit.msm msm, pmatrix.msm, setDF

Examples

```
## Not run:
data( hosp )

# augmenting the data
hosp_augmented = augment( data = hosp, data_key = subj, n_events = adm_number,
                          pattern = label_3, t_start = dateIN, t_end = dateOUT,
                          t_cens = dateCENS )

# let's define the initial transition matrix for our model
Qmat = matrix( data = 0, nrow = 3, ncol = 3, byrow = TRUE )
Qmat[ 1, 1:3 ] = 1
Qmat[ 2, 1:3 ] = 1
colnames( Qmat ) = c( 'IN', 'OUT', 'DEAD' )
rownames( Qmat ) = c( 'IN', 'OUT', 'DEAD' )

# attaching the msm package and running the model using
# gender and age as covariates
library( msm )
msm_model = msm( status_num ~ augmented_int, subject = subj,
                 data = hosp_augmented, covariates = ~ gender + age,
                 exacttimes = TRUE, gen.inits = TRUE, qmatrix = Qmat,
                 method = 'BFGS', control = list( fnscale = 6e+05, trace = 0,
                 REPORT = 1, maxit = 10000 ) )

# plotting the fitted and empirical survival from state = 1
survplot( msm_model, km = TRUE, ci = 'none',
          verbose = FALSE )

# plotting the fitted and empirical survival from state = 2 and
# adding it to the previous plot
survplot( msm_model, from = 2, km = TRUE, ci = 'none', add = TRUE,
          verbose = FALSE )

# returning fitted and empirical data
all_data = survplot( msm_model, ci = 'none', return.all = TRUE,
                     verbose = FALSE, do.plot = FALSE )

# saving them separately
km_data = all_data[[ 1 ]]
fitted_data = all_data[[ 2 ]]

## End(Not run)
```

Index