Package: msmtools (via r-universe)

June 8, 2026

Type Package

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

Version 2.2.1

Date 2026-06-01

Description A fast and general method for restructuring classical longitudinal observational data into augmented transition data suitable for multi-state modeling with the 'msm' package. Works with any longitudinal data where subjects accumulate repeated observations with start and end times and an optional terminal outcome. Methods are described in Grossetti, Ieva and Paganoni (2018) <doi:10.1007/s10729-017-9400-z>.

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

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

License GPL-3

LazyData TRUE

Config/roxygen2/version 8.0.0

Depends R (>= 4.1)

Imports data.table (>= 1.18.4), cli (>= 3.6.0), msm (>= 1.8.2), survival (>= 3.8-6), ggplot2 (>= 4.0.3)

Suggests testthat (>= 3.3.2), knitr (>= 1.51), rmarkdown (>= 2.31), roxygen2 (>= 8.0.0), patchwork (>= 1.3.2)

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Francesco Grossetti [aut, cre] (ORCID: <https://orcid.org/0000-0002-5130-7745>)

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

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

Date/Publication 2026-06-08 16:32:35 UTC

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

RemoteRef HEAD

RemoteSha 00d08017b39fc3c8c0ec360eeeb5bca9f1934510

Contents

augment 2
hosp 6
polish 7
prevplot 8
survplot 11

Index **15**

`augment` *Build augmented transition data*

Description

Reshape standard longitudinal data into augmented transition data suitable for multi-state models fitted with **msm**.

Usage

```
augment(
  data,
  data_key,
  n_events,
  pattern,
  state = c("IN", "OUT", "DEAD"),
  t_start,
  t_end,
  t_cens,
  t_death,
  t_augmented,
  more_status = NULL,
  check_NA = FALSE,
  copy = FALSE,
  verbosity = getOption("msmtools.verbosity", "quiet")
)
```

Arguments

data	A data.table or data.frame object in longitudinal format where each row represents an observation with known start and end times. If data is a data.frame, augment() internally casts it to a data.table.
data_key	A keying variable used to identify subjects and define a key for data (see data.table::setkey()).
n_events	An integer variable indicating the progressive (monotonic) event number for each subject. augment() checks whether n_events is monotonically increasing within each data_key and stops if the check fails (see Details). If missing, augment() creates a variable named "n_events".
pattern	Either an integer, a factor, or a character variable with 2 or 3 unique values that gives each subject's terminal outcome schema. When 2 values are detected, they must be in the format: 0 = "alive", 1 = "dead". When 3 values are detected, they must be: 0 = "alive", 1 = "dead during a transition", 2 = "dead after a transition has ended" (see Details).
state	A character vector of exactly three unique, non-missing, non-empty labels used as the generated transition-state vocabulary. Defaults to c("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 in the augmented format. If t_augmented is missing, the default name "augmented" is used and the new variable is added to data. When t_start is a date or difftime, augment() also creates an integer or numeric companion variable. The suffix "_int" or "_num" is added to t_augmented accordingly. This is needed because **msm** does not handle date or difftime variables directly. Both variables are positioned before t_start.
more_status	A variable that marks further transitions beyond the default ones given by state. more_status can be a factor or character (see Details). If NULL (default), augment() ignores it.
check_NA	If TRUE, data_key, n_events, pattern, t_start, and t_end are checked for missing values. If any missing values are found, the function stops with an error. Default is FALSE because augment() is not intended for general consistency checks and the scan can add memory overhead on very large datasets. more_status is always checked for missing values when supplied.
copy	If FALSE (default), augment() keeps the historical memory-efficient behavior and may modify caller-owned data by reference. If TRUE, data is copied before any data.table operation so the input object remains unchanged.

`verbosity` Controls informational output. Use `"quiet"` to suppress status messages, `"summary"` for high-level phase messages and timing, and `"progress"` for phase messages plus progress bars in long status-building loops. The default is `getOption("msmtools.verbosity", "quiet")`.

Details

`augment()` requires a monotonic event sequence within each subject. The data are ordered with `data.table::setkey()` using `data_key` as the primary key and `t_start` as the secondary key. The function then checks the monotonicity of `n_events`; if the check fails, it stops and reports the subjects that violate the condition. If `n_events` is missing, `augment()` first computes a progression number named *n_events* and then runs the same check.

Argument `pattern` describes the terminal outcome schema and must follow the expected ordering. With two statuses, values must correspond to `0` = `"alive"` and `1` = `"dead"`. With three statuses, integer values must correspond to `0` = `"alive"`, `1` = `"dead inside a transition"`, and `2` = `"dead outside a transition"`. Character and factor values must follow the same order. For example, `0` cannot be used to indicate death.

Argument `state` describes the generated transition-state vocabulary. Its order also matters. The first element is the state at `t_start` (for example, `"IN"`), the second element is the state at `t_end` (for example, `"OUT"`), and the third element is the absorbing state (for example, `"DEAD"`). A two-value `pattern` still requires three `state` labels because `augment()` infers whether death maps to the absorbing state inside or outside the transition window.

`more_status` lets `augment()` represent transitions beyond the defaults in `state`. Standard observations that add no extra information should use `"df"` for "default" (see Examples, or run `?hosp` and inspect `rehab_it`). More complex transitions should use concise, self-explanatory labels.

By default, `augment()` follows **data.table** by-reference semantics to avoid unnecessary copies of large longitudinal datasets. This means the input may have its key changed, and `n_events` may be added when the argument is omitted. Set `copy = TRUE` when the original input object must remain unchanged.

The function always returns a `data.table`. Use `as.data.frame()` on the result if a plain `data.frame` is needed by downstream code.

Value

An augmented dataset of class `data.table`. Each row represents a specific transition for a given subject. `augment()` computes the following key variables:

- `augmented`: The transition time variable. If `t_augmented` is missing, `augment()` creates *augmented* by default. The variable is built from `t_start` and `t_end` and inherits their class. If `t_start` is a date, `augment()` also creates an integer variable named *augmented_int*. If `t_start` is a difftime, it creates a numeric variable named *augmented_num*.
- `status`: A status flag that 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 is useful when modelling process progression.

If more_status is passed, augment() computes additional variables. They mirror 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

Grossetti, F., Ieva, F., and Paganoni, A.M. (2018). A multi-state approach to patients affected by chronic heart failure. *Health Care Management Science*, 21, 281-291. doi:10.1007/s10729017-9400z.

Jackson, C.H. (2011). Multi-State Models for Panel Data: The **msm** Package for R. Journal of Statistical Software, 38(8), 1-29. https://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. https://github.com/Rdatatable/data.table/wiki

See Also

data.table::data.table(), data.table::setkey()

Examples

```
# loading data
data(hosp)

# 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)

# 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)

# requesting progress output
hosp_augmented = augment(data = hosp, data_key = subj, n_events = adm_number,
                         pattern = label_3, t_start = dateIN, t_end = dateOUT,
                         t_cens = dateCENS, verbosity = "summary")
```

hosp *Synthetic Hospital Admissions*

Description

A synthetic longitudinal dataset of hospital admissions for 10 subjects. It includes repeated admissions, admission-level clinical flags, demographic variables, and end-of-study status labels.

Usage

```
data(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`; otherwise `rehab = 0` (integer).
- `it`: Intensive Therapy flag. If the admission has been in intensive therapy, then `it = 1`; otherwise `it = 0` (integer).
- `rehab_it`: String marking the admission type based on `rehab` and `it`. The standard admission is coded as `"df"` (default). Admissions in rehabilitation or intensive therapy are coded as `"rehab"` or `"it"` (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"`, `"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).

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

Description

Remove subjects with transitions to different states occurring at the same exact time in an augmented dataset produced by `augment()`.

Usage

```
polish(
  data,
  data_key,
  pattern,
  time = NULL,
  check_NA = FALSE,
  copy = FALSE,
  verbosity = getOption("msmtools.verbosity", "quiet")
)
```

Arguments

`data`	A `data.table` or `data.frame` object in longitudinal format where each row represents an observation with known start and end times. If `data` is a `data.frame`, `augment()` internally casts it to a `data.table`.
`data_key`	A keying variable used to identify subjects and define a key for `data` (see `data.table::setkey()`).
`pattern`	Either an integer, a factor, or a character variable with 2 or 3 unique values that gives each subject's terminal outcome schema. When 2 values are detected, they must be in the format: 0 = "alive", 1 = "dead". When 3 values are detected, they must be: 0 = "alive", 1 = "dead during a transition", 2 = "dead after a transition has ended" (see Details).
`time`	The time variable used to identify duplicate transition times. If omitted or set to `NULL`, `polish()` uses `"augmented_int"` when it is available, then `"augmented_num"`. If neither column exists, `time` must be supplied explicitly.
`check_NA`	If `TRUE`, `data_key`, `pattern`, and `time` are checked for missing values. If any missing values are found, the function stops with an error. Default is `FALSE`.
`copy`	If `FALSE` (default), `polish()` keeps the historical memory-efficient behavior and may modify caller-owned `data` by reference. If `TRUE`, `data` is copied before any `data.table` operation so the input object remains unchanged.
`verbosity`	Controls informational output. Use `"quiet"` to suppress status messages, `"summary"` for high-level phase messages and timing, and `"progress"` for phase messages plus progress bars in long status-building loops. The default is `getOption("msmtools.verbosity", "quiet")`.

Details

The function searches for cases where two subsequent events for the same 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 function reports how many subjects were removed.

By default, `polish()` follows **data.table** by-reference semantics to avoid unnecessary copies of large augmented datasets. This means the input may have its key changed while duplicate subjects are identified. Set `copy = TRUE` when the original input object must remain unchanged.

The function always returns a `data.table`. Use `as.data.frame()` on the result if a plain `data.frame` is needed by downstream code.

Value

A `data.table` with the same columns as the input `data`. Subjects whose pattern transitions occur at the same `time` on different states are removed in full (every row sharing the same `data_key`); rows from unaffected subjects are kept as-is. When no duplicated transitions are found, the input `data` is returned unchanged.

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 targeted duplicate transitions
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

Plot observed and expected state prevalences from a fitted multi-state model. The function can also compute a rough diagnostic for where the data depart from the estimated Markov model.

Usage

```
prevplot(
  x,
  prev.obj,
  exacttimes = TRUE,
  M = FALSE,
  ci = FALSE,
  print_plot = TRUE,
  verbosity = getOption("msmtools.verbosity", "quiet")
)
```

Arguments

x	A fitted **msm** model object.
prev.obj	A list computed by msm::prevalence.msm(). It may include confidence intervals; prevplot() adapts automatically.
exacttimes	If TRUE (default), transition times are known and exact. This should match the value used when fitting the model with **msm**.
M	If TRUE, a rough indicator of deviance from the model is computed (see Details). Default is FALSE.
ci	If TRUE, confidence intervals are plotted when available. Default is FALSE.
print_plot	If TRUE (default), the plot is printed before being returned. If FALSE, the plot is returned without printing.
verbosity	Controls informational output. Use "quiet" to suppress status messages and "summary" or "progress" for high-level messages.

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 subject k in the state s between observed counts O_is and expected counts E_is is built as M_is = (O_is - E_is)^2 / E_is.

The deviance M plot is returned together with the standard prevalence plot in the second row. This layout is fixed.

When M = TRUE, the combined layout is built with **patchwork**, which is an optional dependency of **msmtools**. Install it with install.packages("patchwork") if it is not already available; prevplot() raises an informative error otherwise. The default M = FALSE path has no such requirement.

Value

When M = FALSE, a gg/ggplot object with observed and expected prevalences is returned. When M = TRUE, a patchwork object is returned with the prevalence plot and the deviance M plot.

The returned object also carries a $prevalence field with the long-format data.table used to build the plot. It always includes time, state, obs, and hat; it also includes lwr and upr when ci = TRUE, and M when M = TRUE. Access it directly:

```
p <- prevplot(model, prev_obj)
p$prevalence
```

`print_plot` only controls whether the plot is printed as a side effect. Returned objects are unchanged: use `print_plot = FALSE` to create the plot silently.

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. https://www.jstatsoft.org/v38/i08/.

See Also

`msm::plot.prevalence.msm()`, `msm::msm()`, `msm::prevalence.msm()`

Examples

```
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')

# fitting 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()
gof = prevplot(x = msm_model, prev.obj = prev, ci = TRUE, M = TRUE)
```

`survplot` *Plot fitted survival and Kaplan-Meier curves from a multi-state model*

Description

Plot fitted survival probabilities from an `msm::msm()` model and compare them with Kaplan-Meier estimates. The function can also return the data used to build each curve.

Usage

```
survplot(
  x,
  from = 1,
  to = NULL,
  range = NULL,
  covariates = "mean",
  exacttimes = TRUE,
  times,
  grid = 100L,
  km = FALSE,
  ci = c("none", "normal", "bootstrap"),
  interp = c("start", "midpoint"),
  B = 100L,
  ci_km = c("none", "plain", "log", "log-log", "logit", "arcsin"),
  print_plot = TRUE,
  verbosity = getOption("msmtools.verbosity", "quiet"),
  ...
)
```

Arguments

`x`	A fitted **msm** model object.
`from`	State from which to compute the estimated survival. Defaults to state 1.
`to`	The absorbing state to which compute the estimated survival. Defaults to the highest state found by `msm::absorbing.msm()`.

range	A numeric vector of two elements giving the time range of the plot.
covariates	Covariate values for which to evaluate the expected probabilities. These can be "mean", denoting the means of the covariates in the data (default); the number 0, indicating that all covariates should be set to zero; or a list of values, with optional names. For example: list(75, 1) The unnamed list must follow the order of the covariates in the original model formula. A named list is also accepted: list(age = 75, gender = "M").
exacttimes	If TRUE (default), transition times are known and exact. This should match the value used when fitting the model with **msm**.
times	An optional numeric vector giving the times at which to compute the fitted survival.
grid	An integer specifying the grid points at which to compute the fitted survival curve (see Details). If times is passed, grid is ignored. Defaults to 100 points.
km	If TRUE, the Kaplan-Meier curve is plotted. Default is FALSE.
ci	A character vector with the type of confidence intervals to compute for the fitted survival curve. Specify either "none" (default), for no confidence intervals, "normal" or "bootstrap", for confidence intervals computed with the respective method in msm::pmatrix.msm(). This is computationally intensive, since intervals must be computed at a series of times.
interp	If "start" (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.
ci_km	A character vector with the type of confidence intervals to compute for the Kaplan-Meier curve. Specify either "none", "plain", "log", "log-log", "logit", or "arcsin", as coded in survival::survfit().
print_plot	If TRUE (default), the plot is printed before being returned. If FALSE, the plot is returned without printing.
verbosity	Controls informational output. Use "quiet" to suppress status messages and "summary" or "progress" for high-level messages.
...	Reserved for the migration trampoline. Passing the legacy out argument here raises an informative error pointing to the new $fitted / $km access pattern. The trampoline will be removed in v2.3.0.

Details

The function wraps msm::plot.survfit.msm() and adds support for exact-time plots by resetting the time scale to follow-up time. It returns a gg/ggplot object so the plot composes directly with ggplot2::ggsave(), ggplot2::theme(), and other ggplot operations.

You can pass custom evaluation times through `times`, or let `survplot()` define them from `grid`. Larger `grid` values produce a finer grid and increase computation time.

Value

A `gg/ggplot` object. The fitted and (when `km = TRUE`) Kaplan-Meier data tables are attached to the returned plot as named fields:

- `$fitted` — a `data.table` with columns `time`, `surv`, and (when `ci` is not `"none"`) `lwr` / `upr`. Always present.
- `$km` — a `data.table` with the Kaplan-Meier curve, exposed only when `km = TRUE`.

Access the data through the standard `$` operator:

```
p <- survplot(model, km = TRUE)
p           # prints the plot
p$fitted    # fitted survival data
p$km        # Kaplan-Meier data
```

`print_plot` only controls whether the plot is printed as a side effect. Returned objects are unchanged: use `print_plot = FALSE` to create the plot or returned data silently.

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. https://www.jstatsoft.org/v38/i08/.

See Also

`msm::plot.survfit.msm()`, `msm::msm()`, `msm::pmatrix.msm()`, `data.table::setDF()`

Examples

```
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')

# fitting 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
theplot = survplot(x = msm_model, km = TRUE)

# the fitted and Kaplan-Meier data tables are attached to the plot
head(theplot$fitted)
head(theplot$km)
```

Index

∗ **datasets**
hosp, 6

as.data.frame(), *4*, *8*
augment, 2
augment(), *8*

data.table::data.table(), *5*
data.table::setDF(), *13*
data.table::setkey(), *3–5*, *7*

ggplot2::ggsave(), *12*
ggplot2::theme(), *12*

hosp, 6

msm::absorbing.msm(), *11*
msm::msm(), *10, 11*, *13*
msm::plot.prevalence.msm(), *10*
msm::plot.survfit.msm(), *12, 13*
msm::pmatrix.msm(), *12, 13*
msm::prevalence.msm(), *9, 10*

polish, 7
prevplot, 8

survival::survfit(), *12*
survplot, 11