Package 'nhm'

July 22, 2025

Type Package

Title Non-Homogeneous Markov and Hidden Markov Multistate Models

Version 0.1.1

Maintainer Andrew Titman <a.titman@lancaster.ac.uk>

Description Fits non-homogeneous Markov multistate models and misclassification-type hidden Markov models in continuous time to intermittently observed data. Implements the methods in Titman (2011) <doi:10.1111/j.1541-0420.2010.01550.x>. Uses direct numerical solution of the Kolmogorov forward equations to calculate the transition probabilities.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports stats, deSolve, maxLik, mvtnorm

Suggests msm, parallel, splines, R.rsp

VignetteBuilder R.rsp

NeedsCompilation yes

Author Andrew Titman [aut, cre]

Repository CRAN

Date/Publication 2023-11-02 21:00:02 UTC

Contents

`ematrix.nhm` *Compute the misclassification probability matrix from a fitted nhm model*

Description

Outputs the matrix of misclasification probabilities in a misclassification type hidden Markov multi-state model fitted using `nhm`.

Usage

```
ematrix.nhm(object, covvalue=NULL)
```

Arguments

`object`	Fitted model object produced using `nhm`.
`covvalue`	Optional vector of covariate vectors (should be given in the order specified in the `covariate` option in `nhm`). If omitted the function will use the mean values of the covariates.

Details

The `emat_nhm` function used to fit the model is called to obtain the values of the misclassification probabilities at the supplied times for the supplied covariate value.

Value

Returns a list containing a matrix of misclassification probabilities and a matrix of corresponding standard errors computed using the delta method.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

See Also

`nhm`, `plot.nhm`, `predict.nhm`, `qmatrix.nhm`

example_data1 *Example of data on a progressive 4 state process*

Description

The observed states and associated observation times for 1000 patients simulated from a 4 state process non-homogeneous Markov model

Usage

```
data("example_data1")
```

Format

A data frame with 3861 rows and 5 variables:

state Observed state at the time of observation

time Time at which the observation occurred

id Patient identification number

cov1 Binary covariate

cov2 Continuous covariate

example_data2 *Example of data on a progressive 4 state process with state misclassification*

Description

The observed states and associated observation times for 1000 patients simulated from a 4 state process non-homogeneous Markov model with misclassification to adjacent transient states.

Usage

```
data("example_data1")
```

Format

A data frame with 3864 rows and 5 variables:

state Observed state at the time of observation

time Time at which the observation occurred

id Patient identification number

cov1 Binary covariate

cov2 Continuous covariate

`initialprob.nhm` *Compute the initial probability vector from a fitted nhm model*

Description

Outputs the vector of initial state probabilities in a misclassification type hidden Markov multi-state model fitted using `nhm`.

Usage

```
initialprob.nhm(object, covvalue=NULL)
```

Arguments

`object`	Fitted model object produced using `nhm`.
`covvalue`	Optional vector of covariate vectors (should be given in the order specified in the `covariate` option in `nhm`). If omitted the function will use the mean values of the covariates.

Details

The `initp_nhm` function used to fit the model is called to obtain the values of the initial state vector at the supplied times for the supplied covariate value.

Value

Returns a list containing a vector of initial state probabilities and a corresponding vector of standard errors computed using the delta method.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

See Also

`nhm`, `ematrix.nhm`

model.nhm *Model object set up for non-homogeneous Markov models*

Description

Sets up a model object in preparation for fitting a non-homogeneous Markov or misclassification type hidden Markov multi-state model.

Usage

```
model.nhm(formula,  data, subject, covariates=NULL,  type, trans,
nonh=NULL, covm=NULL, centre_time=NULL, emat=NULL, ecovm=NULL,
firstobs=NULL, initp=NULL, initp_value=NULL, initcovm=NULL,
splinelist=NULL,degrees=NULL,censor=NULL,
censor.states=NULL,death=FALSE,death.states=NULL,intens=NULL)
```

Arguments

formula	A formula identifying the state and time variables within data, for instance state ~ time would imply the variables are state and time, respectively.
data	data frame containing the observed states, observation times, subject identifiers and covariates. Should include initial observation/recruitment times.
subject	Name of the subject identifier variable within the data data frame.
covariates	A character vector giving the variable names of the covariates to be used in the model
type	type of intensity model. 'bespoke': user supplied, 'weibull': Model with Weibull transition intensity functions with respect to time. 'gompertz': Gompertz/exponential growth intensity models. 'bspline': b-spline function of time model.
trans	Square matrix of viable transitions with dimension equal to the number of states. Impossible transitions should be 0. Others should be labelled consecutively from 1. Labelling transitions with the same value assumes the parameter is shared.
nonh	Square matrix to indicate non-homogeneous transitions with dimension equal to the number of states. Impossible transitions or homogeneous transitions should be 0. Otherwise label consecutively from 1. Labelling the same value implies the same non-homogeneity. Not required if type='bespoke'. If otherwise omitted a time homogeneous model is fitted.
covm	Either a named list of nstate x nstates indicating the covariate effects with respect to a particular covariate OR an nstate x nstate x ncov array to indicate covariate effects, where ncov is the length of the supplied covariates vector. 0 implies no covariate effect. Otherwise label consecutively from 1. Labelling the same value implies a common covariate effect. Not required if type='bespoke'.
centre_time	Value by which to centre time for Gompertz models. By default the model is of the form $h(t) = exp(a+bt)$, centring by c reparametrizes this to $h(t) = exp(a+b(t-c))$. Centring can improve the convergence of optimization routines.

emat	Square matrix of viable misclassification errors. Must be supplied if the model has misclassification. Impossible errors should be 0. Others should be labelled consecutively. Labelling the same implies a common parameter on the logit scale.
ecovm	Either a named list of nstate x nstates indicating the covariate effects with respect to a particular covariate OR an nstate x nstate x ncov array to indicate indicate covariate effects on misclassification, where ncov is the length of the supplied `covariates` vector. 0 implies no covariate effect. Otherwise label consecutively from 1. Labelling the same value implies a common covariate effect.
firstobs	For misclassification models: Form of the first observation for each subject in the data. `'exact'`: Initial state not subject to misclassification (default) `'absent'`: No initial state. First observation is ignored and state occupied is based on initial probabilities model. `'misc'`: Initial state is subject to misclassification.
initp	For misclassification models: Numerical vector of length nstate to define the model for the initial probabilities. The first entry should be zero. Should be numbered consecutively. If the same number is repeated implies a shared parameter. If absent then initial probabilities taken from `initp_value`.
initp_value	For misclassification models where `firstobs="absent"` or `"misc"`: Fixed value of initial probabilities is missing. Should be a numerical vector of length nstate. Ignored if `initp` is present. Default if absent is `c(1,0,...)`.
initcovm	For misclassification models; Either a named list of vectors of length nstate, or an nstate x ncovs matrix to specify the covariate effects on misclassification probabilities. 0 implies no covariate effect. Otherwise label consecutively from 1. Labelling the same value implies a common covariate effect.
splinelist	For bspline models only: list (of length equal to the number of nonhomogeneous transitions) of knot point locations including the boundary knots.
degrees	For bspline models only: optional vector (of length equal to number of nonhomogeneous transitions) of degrees of splines. Defaults to 3 if not specified.
censor	Vector of censor state indicators in the data. Note that censored observations can only occur as the last observation for a subject.
censor.states	List of vectors of states in which subject occupy if censored by corresponding censor state indicator. Can be a vector if only one censor state marker is present.
death	Setting `TRUE` assumes exact death times are present in the data set
death.states	Vector specifying which states have exact death times. Should only correspond to absorbing states.
intens	Optional supplied intensity function. See below for details.

Details

The function allows the model to be specified and creates the metadata needed to use `nhm` to fit it. The function automatically generates a function `intens` which defines the generator matrix of the model and its first derivatives as a function of time `t`, covariates `z` and the underlying parameters `x`, provided the model is of Weibull, Gompertz or B-spline type.

Alternatively, `type='bespoke'` can be chosen. In which case it is necessary for the user to supply a function `intens`. This must have arguments `t, z, x` and return a list consisting of a component `q` which is the nstate x nstate generator matrix, and `dq` which is the nstate x nstate x nparQ first derivatives of the generator matrix with respect to the parameters of the model, where nparQ is the number of parameters in the model for the intensities only (excludes parameters for the emat or initp). Since unrestricted maximization is used so the parameters must take values on `-Inf, Inf`. Note that using a hard-coded version via `type='bespoke'` can be substantially faster than the analogous automatically generated function, so for large models or datasets it may be advantageous to code directly.

For misclassification type models, the function also automatically creates functions `emat_nhm` and `initp_nhm`, to allow the misclassification probability matrix and the initial probability vectors and their derivatives to be calculated at given parameter and covariate values. In each case, a multinomial logistic regression is used for the covariate model. User specification of the misclassification probability function or initial probability vector is not currently possible.

Value

Returns an object of class `nhm_model` containing the necessary metadata needed to use `nhm` to fit the model.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

See Also

nhm

`nhm` *Fit a non-homogeneous Markov model using maximum likelihood*

Description

Fit a continuous-time Markov or hidden Markov multi-state model by maximum likelihood. Observations of the process can be made at arbitrary times, or the exact times of transition between states can be known. Covariates can be fitted to the Markov chain transition intensities or to the hidden Markov observation process.

Usage

```
nhm(model_object, initial=NULL, gen_inits=FALSE,
control, score_test=FALSE, fixedpar=NULL)
```

Arguments

model_object	Model object created using model.nhm
initial	Vector of initial parameter values
gen_inits	If TRUE, then initial values for the transition intensities are generated automatically using the method in crudeinits.msm from the **msm** package. This is not available for models with misclassified states. If FALSE a BHHH algorithm implemented using **maxLik** is used.
control	Object of class nhm.control specifying various settings for the solution of the KFEs and the optimization. See nhm.control for default settings.
score_test	If TRUE just the gradient and Fisher information at the supplied values will be computed to allow score tests to be performed.
fixedpar	Numerical vector indicating which parameters are taken as fixed at the value specified by initial.

Details

For more details about the methodology behind the **nhm** package, see Titman (2011) and the package vignette.

Value

By default returns an object of class nhm containing model output data such as the estimated parameters, maximized likelihood value, information matrix etc. The object can be used with print, predict, plot and anova.

If score.test=TRUE then returns an object of class nhm_score. See print.nhm_score for more details.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

References

Titman AC. Flexible Nonhomogeneous Markov Models for Panel Observed Data. *Biometrics*, 2011. 67, 780-787.

See Also

model.nhm, nhm.control, plot.nhm, predict.nhm, print.nhm_score

Examples

```
### Example dataset
### For further examples, see the vignette
trans <- rbind(c(0,1,0,0),c(0,0,2,0),c(0,0,0,3),rep(0,4))
nonh <- rbind(c(0,1,0,0),c(0,0,2,0),c(0,0,0,3),rep(0,4))
gomp_model <- model.nhm(state~time, data=example_data1, subject = id,
                        type="gompertz",trans=trans,nonh=nonh)
```

```
initial_val <- c(-0.65,-0.45,-0.55,0,0,0)
gomp_fit <- nhm(gomp_model,initial=initial_val,control=nhm.control(obsinfo=FALSE))
gomp_fit
plot(gomp_fit)
plot(gomp_fit,what="intensities")
```

`nhm.control` *Ancillary arguments for controlling nhm fits*

Description

This is used to set various logical or numeric parameters controlling a non-homogeneous Markov model fit. Usually to be used within a call to `nhm`.

Usage

```
nhm.control(tmax=NULL, coarsen=FALSE, coarsen.vars=NULL, coarsen.lv=NULL,
checks=FALSE,rtol=1e-6, atol=1e-6, fishscore=NULL, linesearch=FALSE, damped=FALSE,
damppar=0,obsinfo=TRUE,splits=NULL,ncores=1,print.level=2, maxLikcontrol=NULL)
```

Arguments

`tmax`	Optional parameter to set the maximum time to which the Kolmogorov Forward equations should be integrated. Defaults to 1+max(time) if left unspecified.
`coarsen`	If `TRUE` the covariate values will be subjected to coarsening using K-means clustering, so there are fewer unique values. This is useful for large datasets with continuous covariates.
`coarsen.vars`	Vector of the index of covariates which require coarsening. Must be supplied if `coarsen=TRUE`.
`coarsen.lv`	Number of unique covariate values to which the covariates should be coarsened.
`checks`	If `TRUE` some basic checks will be performed to ensure the accuracy of the supplied `intens` function. Mainly useful if a user defined `type="bespoke"` intensity function is used for which the default is `TRUE`, otherwise default is `FALSE`
`rtol`	Relative error tolerance to be passed to lsoda, default is 1e-6
`atol`	Absolute error tolerance to be passed to lsoda, default is 1e-6
`fishscore`	If `TRUE` then the Fisher scoring algorithm will be used provided the model has no censoring, exact death times or misclassification. This is generally faster, but less robust than the BHHH algorithm.
`linesearch`	If `TRUE` and `fishscore=TRUE` then a line search will be performed to find the best step length in the Fisher scoring algorithm.
`damped`	If `TRUE` the Fisher scoring algorithm will be damped (e.g. Levenberg type algorithm). Useful if some parameters are close to being unidentifiable.
`damppar`	Numerical damping parameter to be applied if `damped=TRUE`

obsinfo	If TRUE the observed Fisher information will be computed in addition to the expected information when the Fisher scoring algorithm is used. For optimization with maxLik the observed Fisher information will be used as the Hessian rather than the squared gradient vectors.
splits	Optional vector of intermediate split times for solving the ODEs. Only needed if P(0,t) becomes singular for some t causing the optimization to stop. Should be a set of consecutive values less than tmax.
ncores	Number of cores to use. 1= no parallelization, 2 or more: Uses mclapply when solving ODEs with different covariates patterns.
print.level	For maxLik optimization; level of detail to print. Integer from 0 to 3. Defaults to 2.
maxLikcontrol	For maxLik optimization; optional list of control parameters to be passed to maxLik.

Details

tmax, rtol and atol refer directly to parameters with the lsoda function in deSolve and relate to how the Kolmogorov Forward Equations are numerically solved.

coarsen, coarsen.vars and coarsen.lv are useful in situations where it is computationally infeasible (or unattractive) to compute the exact solution for all covariate patterns. Implements an approximate solution in which the covariates are coarsened using K-means clustering (as proposed in Titman (2011)).

linesearch, damped, damppar are specific to the Fisher scoring algorithm.

Setting obsinfo=TRUE will tend to give more accurate standard error estimates and gives more opportunity to check for non-convergence of the maximum likelihood procedure.

The option splits modifies the way in which the transition probabilities are computed. By default, nhm solves a single system of differential equations starting from 0 to obtain $P(0,t)$ and then uses inversion of the Chapman-Kolmogorov equation $P(0,t) = P(0,t_0)P(t_0,t)$ to find $P(t_0,t)$ for a given $t_0 > 0$. In some cases $P(0,t_0)$ will be singular or effectively singular. If a split is specified at s then nhm will find $P(t_0,t)$ for $t_0 > t*$ by solving the system of equations $P(t*,t)$ where $t*$ is the smallest interval start time greater than or equal to s within the data. If nhm fails due to the lack of split times, the error message will advise on the interval in which the split should be introduced. Note that the need for splits can also arise if the initial parameters specified are inappropriate. It may often be better to find more appropriate initial parameter estimates,for instance by fitting the analogous homogeneous model in msm, rather than adding multiple split times.

ncores allows parallel processing to be used, through the **parallel** package, to simultaneously solve the systems of differential equations for each covariate pattern. If ncores > 1 then ncores defines the mc.cores value in mclapply. Note that the data needs to include multiple covariate patterns for this to successfully increase computation speed.

Value

A list containing the values of each of the above constants

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

References

Titman AC. Flexible Nonhomogeneous Markov Models for Panel Observed Data. *Biometrics*, 2011. 67, 780-787.

See Also

nhm

`plot.nhm` *Plot transition probabilities or intensities from a fitted nhm model.*

Description

Produces plots of the transition probabilites or intensities from a non-homogeneous Markov or misclassification type hidden Markov multi-state model fitted using nhm.

Usage

```
## S3 method for class 'nhm'
plot(x, what="probabilities",time0=0, state0=1, times=NULL,
covvalue=NULL, ci=TRUE, sim=FALSE, coverage=0.95, B=1000, rtol=1e-6,
atol=1e-6, main_arg=NULL, xlab="Time", ...)
```

Arguments

`x`	Fitted model object produced using nhm.
`what`	Character string to indicate what should be plotted. Options are `probabilities` (the default which produces transition probabilities) or `intensities` (to produce a plot of the intensities)
`time0`	Starting time from which to compute the transition probabilities or intensities. Defaults to 0.
`state0`	Starting state from which to compute the transition probabilities. Defaults to 1. Not required for transition intensities
`times`	Optional vector of times at which to compute the transition probabilities or intensities. If omitted, the probabilities/intensities will be computed at a sequence of times of length 100 from `time0` to the maximum observed time in the data.
`covvalue`	Optional vector of covariate vectors (should be given in the order specified in the `covariate` option in `nhm`). If omitted the function will use the mean values of the covariates.
`ci`	If `TRUE` pointwise confidence intervals will be shown in addition to the point estimates.
`sim`	If `TRUE` a simulation Delta method (Mandel, 2013) will be used to calculate the confidence intervals. Otherwise the standard Delta method will be applied.
`coverage`	Coverage level (should be a value between 0 and 1) for the confidence intervals. Defaults to 0.95.

B	Number of simulations to be performed to compute the simulation Delta method.
rtol	Relative tolerance parameter to be used by lsoda when solving the differential equations for the transition probabilites.
atol	Absolute tolerance parameter to be used by lsoda when solving the differential equations for the transition probabilites.
main_arg	Character string specifying beginning of title to be given to each of the plot panes generated.
xlab	Character string specifying x-axis label to be given to each plot.
...	Other items to be passed to the function. Currently not used.

Details

Computation is performed by calling predict.nhm, for the transition probabilities, or qmatrix.nhm for the intensities (see for more details).

Value

Generates a multi-pane plot for each state. If values are required they can be obtained using predict.nhm.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

References

Mandel M. Simulation-based confidence intervals for functions with complicated derivatives. 2013. *The American Statistician*, 67. 76-81.

See Also

nhm, predict.nhm, qmatrix.nhm

---

predict.nhm *Compute state occupation or transition probabilities from a fitted nhm model*

Description

Outputs the transition probabilites from a non-homogeneous Markov or misclassification type hidden Markov multi-state model fitted using nhm.

Usage

```
## S3 method for class 'nhm'
predict(object, time0=0, state0=1, times=NULL, covvalue=NULL,
ci=TRUE, sim=FALSE, coverage=0.95, B=1000, rtol=1e-6,
atol=1e-6, ...)
```

Arguments

object	Fitted model object produced using nhm.
time0	Starting time from which to compute the transition probabilities. Defaults to 0.
state0	Starting state from which to compute the transition probabilities. Defaults to 1.
times	Optional vector of times at which to compute the transition probabilities. If omitted, the probabilities will be computed at a sequence of times from time0 to the maximum observed time in the data.
covvalue	Optional vector of covariate vectors (should be given in the order specified in the covariate option in nhm). If omitted the function will use the mean values of the covariates.
ci	If TRUE pointwise confidence intervals will be shown in addition to the point estimates.
sim	If TRUE a simulation Delta method (Mandel, 2013) will be used to calculate the confidence intervals. Otherwise the standard Delta method will be applied.
coverage	Coverage level (should be a value between 0 and 1) for the confidence intervals. Defaults to 0.95.
B	Number of simulations to be performed to compute the simulation Delta method.
rtol	Relative tolerance parameter to be used by lsoda when solving the differential equations
atol	Absolute tolerance parameter to be used by lsoda when solving the differential equations
...	Other items to be passed to the function. Currently not used.

Details

The same approach as in the main nhm function of numerically solving the system of differential equations is used to compute transition probabilities based on the maximum likelihood estimates found in nhm and assuming a specific vector of covariates.

If the simulation delta method approach is specified then the function will generate B parameter vectors from the asymptotic distribution of the MLE and solve the system of equations for each of them, before finding pointwise percentile bootstrap confidence intervals from them.

Value

Returns a list containing the vector of times at which the probabilities are computed, a matrix of probabilities for each state at each of the times. If confidence intervals are requested then the lower and upper limits are also provided.

If transition intensity (as opposed to probability) estimates are required then qmatrix.nhm should be used.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

References

Mandel M. Simulation-based confidence intervals for functions with complicated derivatives. 2013. *The American Statistician*, 67. 76-81.

See Also

nhm, plot.nhm, qmatrix.nhm

---

print.nhm_score *Print output from a score test of a nhm object*

Description

Print output from a score test based on parameters supplied to nhm with score_test=TRUE specified.

Usage

```
## S3 method for class 'nhm_score'
print(x, which_comp = NULL, ...)
```

Arguments

x	An object of class nhm_code generated using nhm.
which_comp	Optional vector to specify which of the parameters are to be tested. If omitted, the function will assume all parameters governing non-homogeneity are to be tested. Must be supplied if type='bespoke' was specified when creating the object.
...	Other parameters to be supplied. Currently ignored.

Details

The function provides usable output from specifying score_test=TRUE when using nhm. It is most useful to provide a quick(er) test of whether there may be non-homogeneity in a specific model. Note that the model assumes the initial parameters correspond to the constrained maximum likelihood estimate (for instance a model with all the parameters relating to time homogeneity).

The method can be used to compute the local score tests of homogeneity proposed by de Stavola (1988) if type="gompertz" is specified in nhm.

If fisherscore=TRUE in nhm then the expected Fisher information is used. Otherwise, the empirical mean of the squared gradient terms (as used in the BHHH algorithm) is used to estimate the information.

Value

Prints the results of a score test.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

References

de Stavola BL. Testing Departures from Time Homogeneity in Multistate Markov Processes. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 1988. 37. 242-250.

See Also

nhm

---

qmatrix.nhm *Compute transition intensities from a fitted nhm model*

Description

Outputs the transition intensities from a non-homogeneous Markov or misclassification type hidden Markov multi-state model fitted using nhm.

Usage

```
qmatrix.nhm(object, time0=0, times=NULL, covvalue=NULL, ci=TRUE, sim=FALSE,
coverage=0.95, B=1000)
```

Arguments

object	Fitted model object produced using nhm.
time0	Starting time from which to compute the transition intensities. Defaults to 0.
times	Optional vector of times at which to compute the transition intensities. If omitted, the intensities will be computed at a sequence of times from time0 to the maximum observed time in the data.
covvalue	Optional vector of covariate vectors (should be given in the order specified in the covariate option in nhm). If omitted the function will use the mean values of the covariates.
ci	If TRUE pointwise confidence intervals will be shown in addition to the point estimates.
sim	If TRUE a simulation Delta method (Mandel, 2013) will be used to calculate the confidence intervals. Otherwise the standard Delta method will be applied.
coverage	Coverage level (should be a value between 0 and 1) for the confidence intervals. Defaults to 0.95.
B	Number of simulations to be performed to compute the simulation Delta method.

Details

The `intens` function used to fit the model is called to obtain the values of the transition intensities at the supplied times for the supplied covariate value.

If the simulation delta method approach is specified then the function will generate `B` parameter vectors from the asymptotic distribution of the MLE and compute the intensities for each of them, before finding pointwise percentile bootstrap confidence intervals from them.

Value

Returns a list containing the vector of times at which the intensities are computed, a matrix of probabilities for each state at each of the times. If confidence intervals are requested then the lower and upper limits are also provided.

If transition probability (as opposed to intensity) estimates are required then `predict.nhm` should be used.

Author(s)

Andrew Titman <a.titman@lancaster.ac.uk>

References

Mandel M. Simulation-based confidence intervals for functions with complicated derivatives. 2013. *The American Statistician*, 67. 76-81.

See Also

`nhm`, `plot.nhm`, `predict.nhm`

Index