Package 'flexmsm'

June 23, 2023

Type Package

Title A General Framework for Flexible Multi-State Survival Modelling

Version 0.1.1

Description A general estimation framework for multi-
state Markov processes with flexible specification of the transition intensities.
The log-transition intensities can be specified through Generalised Additive Models which allow for virtually any type of covariate
effect. Elementary specifications such as time-
homogeneous processes and simple parametric forms are also supported. There are
no limitations on the type of process one can assume, with both forward and backward transitions allowed and virtually any number
of states.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports GJRM, mgcv, stats, trust, msm, matrixStats, parallel

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

NeedsCompilation no

Author Alessia Eletti [aut, cre],
Giampiero Marra [aut],
Rosalba Radice [aut]

Maintainer Alessia Eletti <alessia.eletti.19@ucl.ac.uk>

Repository CRAN

Date/Publication 2023-06-23 12:20:18 UTC

R topics documented:

`flexmsm-package` *flexmsm: Flexible Multi-State Modelling*

Description

Provides a function for fitting any type of multistate survival model, with flexibly defined transition intensities and any type of observation scheme. The package also provides a host of tools for straightforward interpretation and visualisation of the fitted model.

The main fitting function is `fmsm`, which fits the multistate survival model, providing parameter and as key inference quantities (p-values, estimated degrees of freedom, ...), as well as the elements needed to obtain predicted transition intensities and probabilities, along with their confidence intervals.

The main auxiliary functions are `Q.pred` and `P.pred`.

Details

Provides functions for fitting and interpreting the output of general flexible multistate survival models. The process is defined by means of a list of model specifications for the transition intensities, each of which follow syntax similar to that used for GAMs in `mgcv`.

The estimation approach is based on a carefully structured, stable penalised likelihood approach, with the smoothers (representing several types of covariate effects) set up using penalised regression splines. The numerical routine carries out function minimization using a trust region algorithm in combination with an adaptation of an automatic multiple smoothing parameter estimation procedure for Generalised Additive Models (see `mgcv` for more details on this last point). The smooths supported by this package are those available in `mgcv`.

Confidence intervals for smooth components and nonlinear functions of the model parameters are derived using a Bayesian approach. P-values for testing individual smooth terms for equality to the zero function are also provided and based on the approach implemented in `mgcv`. The usual plotting and summary functions are also available.

Plots of the estimated transition intensities and transition probabilities can be obtained along with their respective confidence intervals. This includes 3D plots when two-dimensional splines are included in the model specification of one or more transition intensities.

Author(s)

Alessia Eletti (University College London, Department of Statistical Science), Giampiero Marra (University College London, Department of Statistical Science) and Rosalba Radice (Cass Business School, City, University of London).

Maintainer: Alessia Eletti <alessia.eletti.19@ucl.ac.uk>

References

Eletti, A., Marra, G., Radice, R., (submitted), A General Estimation Framework for Multi-State Markov Processes with Flexible Specification of the Transition Intensities.

See Also

`fmsm`

`conv.check` *Convergence diagnostics on fitted model output.*

Description

Convergence diagnostics on fitted model output.

Usage

```
conv.check(object, also.unpen = FALSE)
```

Arguments

`object`	Fitted model object.
`also.unpen`	If `TRUE`, displays eigenvalues range also for the unpenalised Hessian. Defaults to `FALSE`.

Value

Convergence diagnostics.

`fmsm` *Flexible transition intensity based models for univariate multistate processes*

Description

Fits a flexible multistate survival model. Any type of process is supported, including both forward and backward transitions, and must be specified by providing a list of equations, one for each transition intensity allowed. Any type of observation scheme is allowed: the process can be observed in continuous time, intermittently at fixed times, there can be an absorbing state as well as censored states. Virtually any type of covariate effects are supported and can be specified by means of splines, with the same syntax used to specify Generalised Additive Models (GAMs) in `R`.

Usage

```
fmsm(formula, data, id, state, death, pmethod = 'eigendecomp',
       aggregate = TRUE, params.0 = NULL, sp.0 = NULL,
       constraint = NULL, sp.method = 'perf', iterlimsp = 50,
       Q.diagnostics = TRUE, fit = TRUE, iterlim = 100,
       tolsp = 1e-7, tolsp.EFS = 0.1, parallel = FALSE, no_cores = 2,
       cens.state = NULL, living.exact = NULL, verbose = FALSE)
```

Arguments

`formula`	Model specification for the transition intensities.
`data`	Dataset.
`id`	Name of the variable in the dataset representing the unique code associated with each patient.
`state`	Name of the variable in the dataset representing the state occupied by the patient at the given time.
`death`	`TRUE` if the last state is an absorbing state, `FALSE` otherwise.
`pmethod`	Which method should be used for the computation of the transition probability matrix. Available options are • `'eigendecomp'` (default): this method is based on the eigendecomposition of the transition intensity matrix (from Kalbfleisch & Lawless 1985); • `'analytic'`: uses analytic expressions of the transition probabilities, obtained by solving the Kolmogorov forward differential equation, only implemented for IDMs for now; • `'scaling&squaring'`: this is the scaling and squaring method implemented as proposed in Fung (2004).This is inefficient, so its use is not recommended. Can be used to investigate convergence errors.
`aggregate`	Whether or not data should be aggregated (this slightly improves efficiency as redundancies in the data are eliminated). The default is `TRUE`.
`params.0`	Starting values for the model parameters. Defaults to `NULL`, i.e. they are computed internally.

sp.0	Starting values for the smoothing parameters. Defaults to NULL, i.e. they are computed internally.
constraint	A list containing the constraints to be applied to the model parameters. This follows the form list(x1 = (1,1,1), x2 = c())
sp.method	Method to be used for smoothing parameter estimation. The default is magic, the automatic multiple smoothing parameter selection algorithm. Alternatively, efs can be used for the Fellner-Schall method.
iterlimsp	Maximum allowed iterations for smoothing parameter estimation.
Q.diagnostics	If TRUE, diagnostics information on the Q matrix are saved. The default TRUE.
fit	If FALSE, fitting is not carried. May be useful to extract model setup quantities.
iterlim	Maximum allowed iterations for trust region algorithm.
tolsp	Convergence criterion used in magic based smoothing parameter estimation.
tolsp.EFS	Convergence criterion used in efs based smoothing parameter estimation.
parallel	If TRUE parallel computing is used during estimation. This can only be used by Windows users for now.
no_cores	Number of cores used if parallel computing chosen. The default is 2. If NULL, all available cores are used.
cens.state	Code used in the dataset to indicate the censored states.
living.exact	Name of the variable in the dataset indicating whether an observation is exactly observed or not.
verbose	If TRUE, prints the convergence criterion obtained at each iteration of the full algorithm. The default is FALSE.

Value

The function returns an object of class fmsm as described in fmsmObject.

Examples

```
## Not run:

################################################
# MULTISTATE SURVIVAL MODELLING with CAV DATA ####
################################################

library(flexmsm)

# Import data
Data <- IDM_cav

# MODEL SPECIFICATION ####
formula <- list(years ~ s(years, bs = 'cr', k = 10) + dage + pdiag, # 1-2
                years ~ s(years, bs = 'cr', k = 10) + dage + pdiag, # 1-3
                0,                                                  # 2-1
                years ~ s(years, bs = 'cr', k = 10) + dage + pdiag, # 2-3
                0,                                                  # 3-1
```

```
               0                                                    # 3-2
)

# Counts of pairs of consecutive states observed (C = counts, T = times)
counts.CT <- state.pairs.CT(formula = formula, data = Data, time = 'years',
                            state = 'state', id = 'PTNUM')
counts.CT$counts


# MODEL FITTING ###

# NOTE ***
# Takes about 18 minutes on a machine with Windows 10,
# Intel 2.20 GHz core, 16 GB of RAM and 8 cores, using all cores.
# The default is to use 2 cores, this takes about 26 minutes.
# To use all available cores on your device input no_cores = NULL.
# ****

fmsm.out <- fmsm(formula = formula, data = Data,
                 id = PTNUM, state = state, death = TRUE,
                 fit = TRUE, parallel = TRUE, no_cores = 2,
                 pmethod = 'analytic')

print(fmsm.out)

AIC(fmsm.out)
BIC(fmsm.out)

# FITTING SUMMARY ####
summary(fmsm.out)
conv.check(fmsm.out)

####################
# VISUALISATION ####
####################

# PLOT THE SMOOTHS OF TIME FOR EACH TRANSITION ####
# par(mfrow = c(1,3))
plot(fmsm.out)


# Consider a patient with:
dage.pred <- 16      # - 16 year old donor
pdiag.pred <- 0      # - IDC as principal diagnosis
start.pred <- 0      # - start observation at time t = 0
stop.pred <- 15      # - t = 15 years for time horizon
n.pred <- 21         # - 21 time points
no.state.pred <- -13 # - (because we don't need this, so anything is fine)

newdata <- data.frame(PTNUM = rep(1, n.pred),
                      years = seq(start.pred, stop.pred, length.out = n.pred),
                      state = rep(no.state.pred, n.pred),
                      dage = rep(dage.pred, n.pred), pdiag = rep(pdiag.pred, n.pred))
```

```
# ESTMATED TRANSITION INTENSITIES ####

# Plot of estimated transition intensities
# par(mfrow = c(1,3))
Q.hat <- Q.pred(fmsm.out, newdata = newdata, get.CI = TRUE, plot.Q = TRUE, rug = TRUE,
                ylim = c(0, 1.5))

# Estimated transition intensity matrix at, e.g., t = 0
round(Q.hat$Q.hist[,,1], 3)



# ESTMATED TRANSITION PROBABILITIES ####

# Plot of estimated transition probabilities
# par(mfrow = c(2,3))
P.hat <- P.pred(fmsm.out, newdata = newdata, get.CI = TRUE, plot.P = TRUE, rug = TRUE)

# Estimated 15 year transition probability matrix
round(P.hat$P.pred, 3)
# e.g., there is a 6.2% chance of observing CAV onset 15 years after transplant



## End(Not run)
```

`fmsmObject` *Fitted fmsmObject object*

Description

The `fmsm` function returns the fitted model object `fmsmObject`. This is of class "fmsm" and includes the components listed below. These are intended for confident users. To extract results from the fitted model objects, functions such as `summary.fmsm`, `plot.fmsm`, `Q.pred` and `P.pred` should be used instead.

Value

`suStf`	A list with all of the quantities used for estimation and post-estimation computations. This includes the full design matrix `full.X`, the starting parameters used, `params.0` and `sp.0`, and more technical quantities such as the positions of the smooths' parameters and of the parametric coefficients.
`msm.fit.object`	This contains all of the details of the model fitting.
`msm.post.object`	This contains all of the post-estimation details.

formula	Formula used in the model specification.
short.formula	Short version of the model specification, i.e. only non-zero transition specifications are included.
n	Number of observations in the dataset.
N	Number of unique individuals.
logLik	The value of the log-likelihood at convergence.
t.edf	Total effective degrees of freedom.

See Also

fmsm, summary.fmsm

IDM_cav *Cardiac allograft vasculopathy (CAV) data*

Description

A series of approximately yearly angiographic examinations of heart transplant recipients. Onset of cardiac allograft vasculopathy, a deterioration of the arterial wall, and death are monitored. This is a subset of data from the cav dataset in R package msm.

Usage

```
IDM_cav
```

Format

A data frame with 2,803 observations of 614 patients and 5 variables. These are grouped by patient number and ordered by years after transplant.

PTNUM Unique number identifying each patient.

years Examination time (years after transplant).

state State of the examination. State 1 represents no CAV, state 2 represents CAV, state 3 represents death.

dage Age of the heart donor (years).

pdiag Primary diagnosis (reason for transplant). IHD = ischaemic heart disease, IDC = idiopathic dilated cardiomyopathy.

Source

Papworth Hospital, U.K.

`LikGradHess.general` *Likelihood, gradient and Hessian for univariate transition intensity based models*

Description

Likelihood, gradient and Hessian for univariate transition intensity based models

Usage

```
LikGradHess.general(
  params,
  data = NULL,
  full.X = NULL,
  MM,
  pen.matr.S.lambda,
  aggregated.provided = FALSE,
  do.gradient = TRUE,
  do.hessian = TRUE,
  pmethod = "analytic",
  death,
  Qmatr.diagnostics.list = NULL,
  verbose = FALSE,
  parallel = FALSE,
  no_cores = 2
)
```

Arguments

`params`	Parameters vector.
`data`	Dataset in proper format.
`full.X`	Full design matrix.
`MM`	List of necessary setup quantities.
`pen.matr.S.lambda`	Penalty matrix multiplied by smoothing parameter lambda.
`aggregated.provided`	Whether aggregated form was provided (may become obsolete in the future if we see original dataset as special case of aggregated where `nrep = 1`).
`do.gradient`	Whether or not to compute the gradient.
`do.hessian`	Whether or not to compute the Hessian.
`pmethod`	Method to be used for computation of transition probability matrix. See help of `msm()` for further details.
`death`	Whether the last state is an absorbing state.

`Qmatr.diagnostics.list`
: List of maximum absolute values of the Q matrices computed during model fitting.

`verbose`
: Whether to print out the progress being made in computing the likelihood, gradient and Hessian.

`parallel`
: Whether or not to use parallel computing (only for Windows users for now).

`no_cores`
: Number of cores used if parallel computing chosen. The default is 2. If `NULL`, all available cores are used.

Value

Penalized likelihood, gradient and Hessian associated with model at given parameters, for use by trust region algorithm.

---

logLik.fmsm *Extract the log likelihood for the fitted multistate model*

Description

It extracts the log-likelihood for a fitted `fmsm` model.

Usage

```
## S3 method for class 'fmsm'
logLik(object, ...)
```

Arguments

`object`
: Fitted model object of class `fmsm` produced by function `fmsm`.

`...`
: Unused in this case.

Value

Standard logLik object.

P.pred *Predict and plot the transition probabilities*

Description

Function to predict and plot the estimated transition probabilities (and confidence intervals).

Usage

```
P.pred(object, newdata, get.CI = TRUE,
       n.sim.CI = 1000, prob.lev = 0.05,
       plot.P = FALSE, which.plots = NULL,
       rug = FALSE, ...)
```

Arguments

object	Fitted model object.
newdata	Dataframe containing the profile for which one wished to obtain the predicted transition probabilities.
get.CI	Whether to compute the confidence intervals.
n.sim.CI	Number of simulations to be used for confidence intervals computation.
prob.lev	Probability level of confidence intervals.
plot.P	Whether to output plots of transition probabilities.
which.plots	Number between 1 and the maximum number of non-null transition probabilities. This can be used if only some plots are to be plotted.
rug	Whether to include a rugplot of the observed transition times.
...	Other graphical arguments.

Value

Estimated transition probabilities (and confidence intervals).

P.pred	Predicted transition probability matrix corresponding to the time horizon specified in newdata. This is a nstates x nstates matrix.
P.CI.lower	Matrix containing the lower bounds of the confidence intervals for the predicted transition probability matrix.
P.CI.upper	Matrix containing the upper bounds of the confidence intervals for the predicted transition probability matrix.
P.hist	List of predicted transition probability matrices computed at each time point specified in newdata. This is a nstates x nstates x n.pred array, where n.pred is the number of rows in newdata.
P.CI.lower.hist	List of matrices containing the lower bounds of the confidence intervals for each predicted transition probability matrix in P.hist..

`P.CI.upper.hist`
: List of matrices containing the upper bounds of the confidence intervals for each predicted transition probability matrix in `P.hist`.

`full.X`
: Full design matrix corresponding to the `newdata` provided.

`P.sim.hist`
: List of transition probability matrices simulated to obtain the confidence intervals at each time point from `newdata`. May be useful to quickly obtain intervals for a different confidence level.

See Also

`fmsm`

`plot.fmsm` *Function to plot the smooths included in the model specifications.*

Description

Function to plot the smooths included in the model specifications.

Usage

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

Arguments

`x`
: Fitted model object.

`...`
: Other graphical arguments.

Value

Plots the smooths.

`print.fmsm` *Print a fmsm object*

Description

The print method for the `fmsmObject` produced by `fmsm`.

Usage

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

Arguments

x	fmsm object produced by function fmsm.
...	Unused in this case.

Value

print.fmsm prints out a matrix summarising the positions of the transition intensities, the transition intensities formulae, the total number of observations, etc for the fitted multistate survival model.

---

print.summary.fmsm *Flexible transition intensity based models for univariate multistate processes*

Description

Flexible transition intensity based models for univariate multistate processes

Usage

```
## S3 method for class 'summary.fmsm'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	Fitted model object.
digits	Number of digits printed in the output.
signif.stars	By default significance stars are printed alongside the output.
...	Other arguments.

Value

Prints model term summaries.

`Q.matr.setup.general` *Internal function*

Description

Internal function needed for setup of Q matrix and its first and second derivative.

Usage

```
Q.matr.setup.general(
  params,
  nstates,
  full.X,
  start.pos.par,
  l.short.formula,
  whereQ,
  firstD = TRUE,
  secondD = TRUE,
  bound.eta = FALSE,
  pos.optparams,
  pos.optparams2
)
```

Arguments

`params`	Parameters vector.
`nstates`	Number of states.
`full.X`	Full design matrix.
`start.pos.par`	Positions within full parameters vector of starting point for each sub-parameters vector corresponding to each transition intensity specification.
`l.short.formula`	Number of transitions.
`whereQ`	Positions within Q matrix of not-null transition intensities.
`firstD`	Whether the first derivative of the Q matrix should be computed.
`secondD`	Whether the second derivative of the Q matrix should be computed.
`bound.eta`	Whether to bound the additive predictor, defaults to `FALSE`. This is only used for debugging purposes, do not change.
`pos.optparams`	Vector with positions of parameters vector in the form used by the optimization algorithm (i.e. when one or more parameters are constrained to be equal these will only appear once).
`pos.optparams2`	Like `pos.optparams` but the count is not stopped at the constrained parameters.

Value

Q matrix and its first and second derivatives with respect to the parameters vector.

Q.pred *Predict and plot the transition intensities*

Description

Function to predict and plot the estimated transition intensities (and confidence intervals).

Usage

```
Q.pred(object, newdata, get.CI = TRUE,
       n.sim.CI = 1000, prob.lev = 0.05,
       plot.Q = FALSE, which.plots = NULL,
       cond.list.2d = NULL, plot.Q.2d = FALSE,
       rug = TRUE, ...)
```

Arguments

object	Fitted model object.
newdata	Dataframe containing the profile for which one wished to obtain the predicted transition intensities.
get.CI	Whether to compute the confidence intervals.
n.sim.CI	Number of simulations to be used for confidence intervals computation.
prob.lev	Probability level of confidence intervals.
plot.Q	Whether to output plots of transition intensities.
which.plots	Number between 1 and the maximum number of non-null transition intensities. This can be used if only some plots are to be plotted.
cond.list.2d	Value of covariate(s) to be kept fixed in the plotting of 3D-transition intensities.
plot.Q.2d	Whether to plot 3D transition intensities (only valid if 2D-smooths are present).
rug	Whether to include a rugplot of the observed transition times.
...	Other graphical parameters.

Value

Estimated transition intensities (and confidence intervals).

Q.hist	List of predicted transition intensity matrices computed at each time point specified in newdata. This is a nstates x nstates x n.pred array, where n.pred is the number of rows in newdata.
Q.CI.lower	Matrix containing the lower bounds of the confidence intervals for the predicted transition intensity matrix.
Q.CI.upper	Matrix containing the upper bounds of the confidence intervals for the predicted transition intensity matrix.
full.X	Full design matrix corresponding to the newdata provided.
Q.sim.hist	List of transition intensity matrices simulated to obtain the confidence intervals at each time point from newdata. May be useful to quickly obtain intervals for a different confidence level.

See Also

fmsm

`simulateIDM` *Function to predict and plot the estimated transition intensities (and confidence intervals).*

Description

Function to predict and plot the estimated transition intensities (and confidence intervals).

Usage

```
simulateIDM(N = N, seed = seed, og.12 = TRUE)
```

Arguments

`N`	Total number of individuals.
`seed`	Seed used for the simulation.
`og.12`	If `TRUE` a common shape for the first transition is used. If `FALSE` a sinusoid is used.

Value

Simulated data generated from an Illness-Death model (IDM).

`state.pairs.CT` *Function to extract state pair counts and observed (right) times.*

Description

Function to extract state pair counts and observed (right) times.

Usage

```
state.pairs.CT(
  formula = NULL,
  data = NULL,
  whereQ = NULL,
  nstates = NULL,
  time = NULL,
  state = NULL,
  id = NULL
)
```

Arguments

formula	Model specification.
data	Data.
whereQ	Placement of allowed transition intensities. Only for internal use. Defaults to NULL and is obtained automatically when formula is provided.
nstates	Total number of states. Only for internal use. Defaults to NULL and is obtained automatically when formula is provided.
time	Name of variable containing the time-to-event.
state	Name of variable containing the states.
id	Name of variable containing the unique code identifying the individuals.

Value

A table with the state-pair counts and a list with the observed (right) times for each transition.

---

summary.fmsm *Summary for fitted model ouput.*

Description

Summary for fitted model ouput.

Usage

```
## S3 method for class 'fmsm'
summary(object, ...)
```

Arguments

object	Fitted model object.
...	Other arguments.

Value

Summary of fitted model object.

Index