Package 'JLPM'

May 7, 2026

Type Package

Title Joint Latent Process Models

Version 1.0.4

Date 2026-04-30

Description Estimation of extended joint models with shared random effects. Longitudinal data are handled in latent process models for continuous (Gaussian or curvilinear) and ordinal outcomes while proportional hazard models are used for the survival part. We propose a frequentist approach using maximum likelihood estimation. See Saulnier et al, 2022 <doi:10.1016/j.ymeth.2022.03.003>.

License GPL (>= 2.0)

Depends R (>= 3.5.0), lcmm

Imports survival (>= 2.37-2), spacefillr, stringr, marqLevAlg (>= 2.0.6)

URL https://VivianePhilipps.github.io/JLPM/

BugReports https://github.com/VivianePhilipps/JLPM/issues

LazyLoad yes

RoxygenNote 7.3.3

NeedsCompilation yes

Author Viviane Philipps [aut, cre],
Tiphaine Saulnier [aut],
Cecile Proust-Lima [aut]

Maintainer Viviane Philipps <Viviane.Philipps@u-bordeaux.fr>

Repository CRAN

Date/Publication 2026-05-04 13:00:13 UTC

Contents

`JLPM-package` *Estimation of joint latent process models*

Description

Functions for the estimation of joint latent process models (JLPM). Continuous and ordinal outcomes are handled for the longitudinal part, whereas the survival part considers multiple competing events. The likelihood is computed using Monte-Carlo integration. Estimation is achieved by maximizing the log-likelihood using a robust iterative algorithm.

Details

Please report to the JLPM-team any question or suggestion regarding the package via github only (https://github.com/VivianePhilipps/JLPM/issues).

Author(s)

Viviane Philipps, Tiphaine Saulnier and Cecile Proust-Lima

References

Saulnier, Philipps, Meissner, Rascol, Pavy-Le-Traon, Foubert-Samier, Proust-Lima (2022). Joint models for the longitudinal analysis of measurement scales in the presence of informative dropout, Methods; 203:142-51.

Philipps, Hejblum, Prague, Commenges, Proust-Lima (2021). Robust and efficient optimization using a Marquardt-Levenberg algorithm with R package marqLevAlg, The R Journal 13:2.

convert *Conversion*

Description

This function converts a jointLPM object to an object of type `multlcmm` or `Jointlcmm` from lcmm package. The conversion to `multlcmm` unable the use of the dedicated postfit and predictions functions implemented in the lcmm package (plot, predictL, predictY, predictlink, predictYcond, fitY, VarExpl). The conversion to `Jointlcmm` permits the use of functions cuminc and plot (with options "baselinerisk" or "survival").

Usage

```
convert(object, to)
```

Arguments

`object`	an object of class `jointLPM`
`to`	character. Either "multlcmm" or "Jointlcmm", indicating to which type the object should be converted.

Value

an object of class `multlcmm` or `Jointlcmm`.

createFargs *Likelihood arguments*

Description

Create the arguments to be passed to the `loglik` function

Usage

```
createFargs(x, data)
```

Arguments

`x`	a `jointLPM` object
`data`	a data.frame containing the estimation dataset

Value

a named list

createX0 *createX0*

Description

Create the matrix containing all covariates appearing in the formula

Usage

```
createX0(form, data)
```

Arguments

form	a list of formula
data	a data frame containing all variables mentionned in form

Details

Caution 1 : 'data' should not contain any missing values. There is no check for that within the function. Caution 2 : if interaction terms are present, the names are sorted. For example x:age becomes age:x in X0. Caution 3 : intercept is not systematically in the first column. It will only if form[[1]] contains an intercept.

Value

a list containing

X0	the pooled (with cbind) model matrix. The number of lines is the same as in 'data'
idform	a list of the same length as 'form' specifying which term of X0 appear in each formula

diffSojournTime *Difference of sojourn times*

Description

This function computes the difference between two sojourn times.

Usage

```
diffSojournTime(x1, x2)
```

Arguments

x1	a sojournTime object, called with draws = TRUE and returndraws = TRUE, computed for a specified seed.
x2	a sojournTime object, called with draws = TRUE and returndraws = TRUE, computed with the same seed as x1.

Value

returns the median, the 2.5% and 97.5% quantiles, the mean, the standard deviation of the difference between the two sojourn times and the number of removed draws (eventually due to computational issues)

ForInternalUse *For internal use only ...*

Description

For internal use only ...

IRT4Sselecting *Fisher information function, used in Step 4 _ Selecting from the 4Smethod*

Description

This function computes the contribution if each item during stages, based on the Fisher information.

Usage

```
IRT4Sselecting(D, proj)
```

Arguments

D	an object of class jointLPM representing the estimated dimension model with the items
proj	a vector

Value

a list of two matrices

###@examples

Author(s)

Tiphaine Saulnier, Viviane Philipps and Cecile Proust-Lima

References

preprint : arXiv:2407.08278

`IRT4Sstaging` *Projection function, used in Step 3 _ Staging from the 4Smethod*

Description

This function determines the stage thresholds in the latent scale of the dimension process. Specifically, this function predicts the dimension scores corresponding to stage changes and deducts the corresponding thresholds in the dimension scale based on a correspondence between the predicted scores and the expected sum of the items. The output is a vector of the estimated stage transition thresholds.

Usage

```
IRT4Sstaging(D, Dss, nsim = 1000, bounds = c(-10, 10))
```

Arguments

`D`	an object of class `jointLPM` representing the estimated dimension model with the items
`Dss`	an object of class `jointLPM` representing the estimated proxy model between the dimension scores and the stages
`nsim`	number of points used in the numerical integration (Monte-Carlo). nsim should be relatively important (nsim=1000 by default).
`bounds`	dimension scale boundaries between which are computed a grid of the dum of the items

Value

a vector of the estimated stage transition thresholds

###@examples

Author(s)

Tiphaine Saulnier, Viviane Philipps and Cecile Proust-Lima

References

preprint : arXiv:2407.08278

jointLPM *Estimation of latent process joint models for multivariate longitudinal outcomes and time-to-event data.*

Description

This function fits extended joint models with shared random effects. The longitudinal submodel handles multiple continuous longitudinal outcomes (Gaussian or non-Gaussian, curvilinear) as well as ordinal longitudinal outcomes in a mixed effect framework. The model assumes that all the outcomes measure the same underlying latent process defined as their common factor, and each outcome is related to this latent common factor by a outcome-specific measurement model whose nature depends on the type of the outcome (linear model for Gaussian outcome, curvilinear model for non-Gaussian outcome, probit model for binary outcome, cumulative probit model for ordinal outcome). At the latent process level, the model assumes a linear mixed model. The survival submodel handles right-censored (possibly left-truncated) time-to-event with competing causes The association between the longitudinal and the survival data is captured by including the random effect from the mixed model or the predicted current level of the underlying process as linear predictors in the cause-specific proportional hazard survival model. Parameters of the measurement models, of the latent process mixed model and of the survival model are simultaneously estimated using a maximum likelihood method, through a Marquardt-Levenberg optimization algorithm.

Usage

```
jointLPM(
  fixed,
  random,
  subject,
  idiag = FALSE,
  cor = NULL,
  link = "linear",
  intnodes = NULL,
  epsY = 0.5,
  randomY = FALSE,
  var.time,
  survival = NULL,
  hazard = "Weibull",
  hazardrange = NULL,
  hazardnodes = NULL,
  TimeDepVar = NULL,
  logscale = FALSE,
  startWeibull = 0,
  sharedtype = "RE",
  centerpoly = 0,
  methInteg = "QMC",
  nMC = 1000,
  data,
  subset = NULL,
```

```
  na.action = 1,
  B,
  posfix = NULL,
  maxiter = 100,
  convB = 1e-04,
  convL = 1e-04,
  convG = 1e-04,
  partialH = NULL,
  nsim = 100,
  range = NULL,
  verbose = TRUE,
  returndata = FALSE,
  nproc = 1,
  clustertype = NULL
)
```

Arguments

fixed	a two-sided linear formula object for specifying the fixed-effects in the linear mixed model at the latent process level. The response outcomes are separated by + on the left of ~ and the covariates are separated by + on the right of the ~. For identifiability purposes, the intercept specified by default should not be removed by a -1. Variables on which a contrast above the different outcomes should also be estimated are included with contrast().
random	a one-sided formula for the random-effects in the latent process mixed model. Covariates with a random-effect are separated by +. An intercept should always be included for identifiability.
subject	name of the covariate representing the grouping structure.
idiag	optional logical for the variance-covariance structure of the random-effects. If FALSE, a non structured matrix of variance-covariance is considered (by default). If TRUE a diagonal matrix of variance-covariance is considered.
cor	optional indicator for inclusion of an autocorrelated Gaussian process in the linear mixed model at the latent process level. Option BM(time) indicates a brownian motion with parameterized variance while option AR(time) specifies an autoregressive process in continuous time with parameterized variance and correlation intensity. In both cases, time is the variable representing the measurement times. By default, no autocorrelated Gaussian process is added.
link	optional vector of families of parameterized link functions defining the measurement models (one by outcome). Option "linear" (by default) specifies a linear link function. Other possibilities include "beta" for estimating a link function from the family of Beta cumulative distribution functions, "Splines" for approximating the link function by I-splines and "thresholds" for ordinal outcomes modelled by cumulative probit models. For splines case, the number of nodes and the nodes location should be also specified. The number of nodes is first entered followed by -, then the location is specified with "equi", "quant" or "manual" for respectively equidistant nodes, nodes at quantiles of the marker distribution or interior nodes entered manually in argument intnodes. It is followed

by -splines. For example, "7-equi-splines" means I-splines with 7 equidistant nodes, "6-quant-splines" means I-splines with 6 nodes located at the quantiles of the marker distribution and "9-manual-splines" means I-splines with 9 nodes, the vector of 7 interior nodes being entered in the argument intnodes.

intnodes — optional vector of interior nodes. This argument is only required for a I-splines link function with nodes entered manually.

epsY — optional positive real used to rescale the marker in (0,1) when the beta link function is used. By default, epsY=0.5.

randomY — optional logical for including an outcome-specific random intercept. If FALSE no outcome-specific random intercept is added (default). If TRUE independent outcome-specific random intercepts with parameterized variance are included.

var.time — name of the variable representing the measurement times.

survival — two-sided formula object. The left side of the formula corresponds to a Surv() object for right-censored (Surv(EntryTime,Time,Indicator)) and possibly left-truncated (Surv(EntryTime,Time,Indicator)). Multiple causes of event can be considered in the Indicator (0 for censored, k for event of cause k). The right side of the formula specifies the covariates to include in the survival model. Cause-specific covariate effect are specified with cause().For example, Surv(Time,Indicator) ~ X1 + cause(X2) indicates a common effect of X1 and a cause-specific effect of X2.

hazard — optional family of hazard function assumed for the survival model. By default, "Weibull" specifies a Weibull baseline risk function. Other possibilities are "piecewise" for a piecewise constant risk function or "splines" for a cubic M-splines baseline risk function. For these two latter families, the number of nodes and the location of the nodes should be specified as well, separated by -. The number of nodes is entered first followed by -, then the location is specified with "equi", "quant" or "manual" for respectively equidistant nodes, nodes at quantiles of the times of event distribution or interior nodes entered manually in argument hazardnodes. It is followed by - and finally "piecewise" or "splines" indicates the family of baseline risk function considered. Examples include "5-equi-splines" for M-splines with 5 equidistant nodes, "6-quant-piecewise" for piecewise constant risk over 5 intervals and nodes defined at the quantiles of the times of events distribution and "9-manual-splines" for M-splines risk function with 9 nodes, the vector of 7 interior nodes being entered in the argument hazardnodes. In the presence of competing events, a vector of hazards should be provided such as hazard=c("Weibull","5-quant-splines") with 2 causes of event, the first one modelled by a Weibull baseline cause-specific risk function and the second one by splines.

hazardrange — optional vector indicating the range of the survival times (that is the minimum and maximum). By default, the range is defined according to the minimum and maximum observed values of the survival times. The option should be used only for piecewise constant and Splines hazard functions.

hazardnodes — optional vector containing interior nodes if splines or piecewise is specified for the baseline hazard function in hazard.

TimeDepVar — optional vector containing an intermediate time corresponding to a change in the risk of event. This time-dependent covariate can only take the form of a time

variable with the assumption that there is no effect on the risk before this time and a constant effect on the risk of event after this time (example: initiation of a treatment to account for).

`logscale` optional boolean indicating whether an exponential (logscale=TRUE) or a square (logscale=FALSE -by default) transformation is used to ensure positivity of parameters in the baseline risk functions. See details section

`startWeibull` optional numeric with Weibull hazard functions only. Indicates the shift in the Weibull distribution.

`sharedtype` indicator of shared random function type. `'RE'` indicates an association through the random effects included in the linear mixed model. `'CL'` defines a association through the predicted current level of the latent process.

`centerpoly` for polynomial associations only, value(s) to center degree 2 and 3 polynomial terms.

`methInteg` character indicating the type of integration to compute the log-likelihood. 'MCO' for ordinary Monte Carlo, 'MCA' for antithetic Monte Carlo, 'QMC' for quasi Monte Carlo. Default to "QMC".

`nMC` integer, number of Monte Carlo simulations. Default to 1000.

`data` data frame containing all variables named in `fixed`, `random`, `cor`, `survival` and `subject`.

`subset` optional vector giving the subset of observations in `data` to use. By default, all lines.

`na.action` Integer indicating how NAs are managed. The default is 1 for 'na.omit'. The alternative is 2 for 'na.fail'. Other options such as 'na.pass' or 'na.exclude' are not implemented in the current version.

`B` optional specification for the initial values for the parameters. Initial values should be entered in the order detailed in `details` section.

`posfix` Optional vector giving the indices in vector B of the parameters that should not be estimated. Default to NULL, all parameters are estimated.

`maxiter` optional maximum number of iterations for the Marquardt iterative algorithm. By default, maxiter=100.

`convB` optional threshold for the convergence criterion based on the parameter stability. By default, convB=0.0001.

`convL` optional threshold for the convergence criterion based on the log-likelihood stability. By default, convL=0.0001.

`convG` optional threshold for the convergence criterion based on the derivatives. By default, convG=0.0001.

`partialH` optional vector giving the indices in vector B of parameters that can be dropped from the Hessian matrix to define convergence criteria.

`nsim` number of points used to plot the estimated link functions. By default, nsim=100.

`range` optional vector indicating the range of the outcomes (that is the minimum and maximum). By default, the range is defined according to the minimum and maximum observed values of the outcome. The option should be used only for Beta and Splines transformations.

verbose	logical indicating if information about computation should be reported. Default to TRUE.
returndata	logical indicating if data used for computation should be returned. Default to FALSE, data are not returned.
nproc	number of cores for parallel computation.
clustertype	the type of cluster that should internally be created. See parallel::makeCluster for possible values.

Details

A. THE MEASUREMENT MODELS

jointLPM function estimates one measurement model per outcome to link each outcome Y_k(t) with the underlying latent common factor L(t) they measure. To fix the latent process dimension, we chose to constrain the latent process dimension with the intercept of the mixed model fixed at 0 and the standard error of the first random effect fixed at 1. The nature of each measurement model adapts to the type of the outcome it models.

1. For continuous Gaussian outcomes, linear models are used and require 2 parameters for the transformation (Y(t) - b1)/b2

2. For continuous non-Gaussian outcomes, curvilinear models use parameterized link functions to link outcomes to the latent process. With the "beta" link function, 4 parameters are required for the following transformation: [h(Y(t)',b1,b2) - b3]/b4 where h is the Beta CDF with canonical parameters c1 and c2 that can be derived from b1 and b2 as c1=exp(b1)/[exp(b2)*(1+exp(b1))] and c2=1/[exp(b2)*(1+exp(b1))], and Y(t)' is the rescaled outcome i.e. Y(t)'= [Y(t) - min(Y(t)) + epsY] / [max(Y(t)) - min(Y(t)) +2*epsY]. With the "splines" link function, n+2 parameters are required for the following transformation b_1 + b_2*I_1(Y(t)) + ... + b_(n+2)*I_(n+1)(Y(t)), where I_1,...,I_(n+1) is the basis of quadratic I-splines. To constrain the parameters to be positive, except for b_1, the program estimates b_k^* (for k=2,...,n+2) so that b_k=(b_k^*)^2.

3. For ordinal outcomes (and binary outcomes), cumulative probit models are used. For a (n+1)-level outcome, the model consist of determining n thresholds t_k in the latent process scale which correspond to the outcome level changes. Then, Y(t) = n' <=> t_n' < L(t) + e <= t_(n'+1) with e the standard error of the outcome. To ensure that t_1 < t_2 < ... < t_n, the program estimates t'_1, t'_2, ..., t'_n such that t_1=t'_1, t_2=t_1+(t'_2)^2, t_3=t_2+(t'_3)^2, ...

B. THE SURVIVAL MODEL

a. BASELINE RISK FUNCTIONS

For the baseline risk functions, the following parameterizations are considered.

1. With the "Weibull" function: 2 parameters are necessary w_1 and w_2 so that the baseline risk function a_0(t) = w_1^2*w_2^2*(w_1^2*t)^(w_2^2-1) if logscale=FALSE and a_0(t) = exp(w_1)*exp(w_2)(t)^(exp(w_2)-1) if logscale=TRUE.

2. with the "piecewise" step function and nz nodes (y_1,...y_nz), nz-1 parameters are necesssary p_1,...p_nz-1 so that the baseline risk function a_0(t) = p_j^2 for y_j < t =< y_j+1 if logscale=FALSE and a_0(t) = exp(p_j) for y_j < t =< y_j+1 if logscale=TRUE.

3. with the "splines" function and nz nodes (y_1,...y_nz), nz+2 parameters are necessary s_1,...s_nz+2 so that the baseline risk function a_0(t) = sum_j s_j^2 M_j(t) if logscale=FALSE and a_0(t) = sum_j exp(s_j) M_j(t) if logscale=TRUE where M_j is the basis of cubic M-splines. Two parameterizations of the baseline risk function are proposed (logscale=TRUE or FALSE) because in some cases,

especially when the instantaneous risks are very close to 0, some convergence problems may appear with one parameterization or the other. As a consequence, we recommend to try the alternative parameterization (changing logscale option) when a model does not converge (maximum number of iterations reached) although convergence criteria based on the parameters and likelihood are small.

b. ASSOCIATION BETWEEN LONGITUDINAL AND SURVIVAL DATA

The association between the longitudinal and the survival data is captured by including a function of the elements from the latent process mixed model as a predictor in the survival model. We implemented so far two association structures, that should be specified through `sharedtype` argument.

1. the random effect from the latent process linear mixed model (`sharedtype='RE'`): the q random effects modeling the individual deviation in the longitudinal model are also included in the survival model, so that a q-vector of parameters measures the association between the risk of event and the longitudinal outcome(s).

2. the predicted current level of the underlying process (`sharedtype='CL'`): the predicted latent process defined by the mixed model at time t is included as time-dependent covariate in the risk of event model for time t. The association between the longitudinal process and the risk of event is then quantified by a unique parameter.

C. THE VECTOR OF PARAMETERS B

The parameters in the vector of initial values `B` or in the vector of maximum likelihood estimates `best` are included in the following order: (1) parameters for the baseline risk function: 2 parameters for each Weibull, nz-1 for each piecewise constant risk and nz+2 for each splines risk. In the presence of competing events, the number of parameters should be adapted to the number of causes of event; (2) for all covariates in survival, one parameter is required. Covariates parameters should be included in the same order as in survival. In the presence of cause-specific effects, the number of parameters should be multiplied by the number of causes; (3) parameter(s) of association between the longitudinal and the survival process: for `sharedtype='RE'`, one parameter per random effect and per cause of event is required; for `sharedtype='CL'`, one parameter per cause of event is required; (4) for all covariates in fixed, one parameter is required. Parameters should be included in the same order as in fixed; (5)for all covariates included with `contrast()` in `fixed`, one supplementary parameter per outcome is required excepted for the last outcome for which the parameter is not estimated but deduced from the others; (6) the variance of each random-effect specified in random (excepted the intercept which is constrained to 1) if idiag=TRUE and the inferior triangular variance-covariance matrix of all the random-effects if idiag=FALSE; (7) if `cor` is specified, the standard error of the Brownian motion or the standard error and the correlation parameter of the autoregressive process; (8) parameters of each measurement model: 2 for "linear", 4 for "beta", n+2 for "splines" with n nodes, n for "thresholds" for a (n+1)-level outcome; (9) if `randomY=TRUE`, the standard error of the outcome-specific random intercept (one per outcome); (10) the outcome-specific standard errors (one per outcome)

C. CAUTIONS REGARDING THE USE OF THE PROGRAM

Some caution should be made when using the program. Convergence criteria are very strict as they are based on the derivatives of the log-likelihood in addition to the parameter and log-likelihood stability. In some cases, the program may not converge and reach the maximum number of iterations fixed at 100 by default. In this case, the user should check that parameter estimates at the last iteration are not on the boundaries of the parameter space.

If the parameters are on the boundaries of the parameter space, the identifiability of the model is critical. This may happen especially with splines parameters that may be too close to 0 (lower boundary). When identifiability of some parameters is suspected, the program can be run again

from the former estimates by fixing the suspected parameters to their value with option posfix. This usually solves the problem. An alternative is to remove the parameters of the Beta of Splines link function from the inverse of the Hessian with option partialH. If not, the program should be run again with other initial values, with a higher maximum number of iterations or less strict convergence tolerances.

To reduce the computation time, this program can be carried out in parallel mode, ie. using multiple cores which number can be specified with argument `nproc`.

Value

An object of class "jointLPM" is returned containing some internal information used in related functions. Users may investigate the following elements :

`ns`	number of grouping units in the dataset
`loglik`	log-likelihood of the model
`best`	vector of parameter estimates in the same order as specified in `B` and detailed in section `details`
`V`	vector containing the upper triangle matrix of variance-covariance estimates of `best` with exception for variance-covariance parameters of the random-effects for which `V` contains the variance-covariance estimates of the Cholesky transformed parameters displayed in `cholesky`
`gconv`	vector of convergence criteria: 1. on the parameters, 2. on the likelihood, 3. on the derivatives
`conv`	status of convergence: =1 if the convergence criteria were satisfied, =2 if the maximum number of iterations was reached, =4 or 5 if a problem occured during optimisation
`call`	the matched call
`niter`	number of Marquardt iterations
`nevent`	number of occured event
`pred`	table of individual predictions and residuals in the underlying latent process scale; it includes marginal predictions (pred_m), marginal residuals (resid_m), subject-specific predictions (pred_ss) and subject-specific residuals (resid_ss) and finally the transformed observations in the latent process scale (obs).
`predRE`	table containing individual predictions of the random-effects : a column per random-effect, a line per subject.
`predRE_Y`	table containing individual predictions of the outcome-specific random intercept
`predSurv`	table containing the predicted baseline risk function and the predicted cumulative baseline risk function
`cholesky`	vector containing the estimates of the Cholesky transformed parameters of the variance-covariance matrix of the random-effects
`estimlink`	table containing the simulated values of each outcome and the corresponding estimated link function
`epsY`	definite positive reals used to rescale the markers in (0,1) when the beta link function is used. By default, epsY=0.5.

AIC	the Akaike's information criterion
BIC	the Bayesian information criterion
CPUtime	the runtime in seconds
data	the original data set (if returndata is TRUE)

Author(s)

Viviane Philipps, Tiphaine Saulnier and Cecile Proust-Lima

References

Saulnier, Philipps, Meissner, Rascol, Pavy-Le-Traon, Foubert-Samier, Proust-Lima (2022). Joint models for the longitudinal analysis of measurement scales in the presence of informative dropout, Methods; 203:142-51.

Philipps, Hejblum, Prague, Commenges, Proust-Lima (2021). Robust and efficient optimization using a Marquardt-Levenberg algorithm with R package marqLevAlg, The R Journal 13:2.

Examples

```
#### Examples with paquid data from R-package lcmm
library(lcmm)
paq <- paquid[which(paquid$age_init<paquid$agedem),]
paq$age65 <- (paq$age-65)/10

#### Example with one Gaussian marker :
## We model the cognitive test IST according to age, sexe and eduction level. We assume
## a Weibull distribution for the time to dementia and link the longitudinal and survival
## data using the random effects.
## We provide here the call to the jointLPM function without optimization (maxiter=0). The
## results should therefore not be interpreted.
M0 <- jointLPM(fixed = IST~age65*(male+CEP),
                random=~age65,
                idiag=FALSE,
                subject="ID",
                link="linear",
                survival=Surv(age_init,agedem,dem)~male,
                sharedtype='RE',
                hazard="Weibull",
                data=paq,
                var.time="age65",
                maxiter=0)
M0$best ## these are the initial values of each of the 15 parameters


## Estimation with one Gaussian marker
## We remove the maxiter=0 option to estimate the model. We specify initial values
## to reduce the runtime, but this can take several minutes.
binit1 <- c(0.1039, 5.306, -0.1887, -1.0355, -4.3817, -1.0543, -0.1161, 0.8588,
0.0538, -0.1722, -0.2224, 0.3296, 30.7768, 4.6169, 0.7396)
M1 <- jointLPM(fixed = IST~age65*(male+CEP),
                random=~age65,
```

```
                idiag=FALSE,
                subject="ID",
                link="linear",
                survival=Surv(age_init,agedem,dem)~male,
                sharedtype='RE',
                hazard="Weibull",
                data=paq,
                var.time="age65",
                B=binit1)
## Optimized the parameters to be interpreted :
summary(M1)


#### Estimation with one ordinal marker :
## We consider here the 4-level hierarchical scale of dependence HIER and use "thresholds"
## to model it as an ordinal outcome. We assume an association between the current level
## of dependency and the risk of dementia through the option sharedtype="CL".
## We use a parallel optimization on 2 cores to reduce computation time.
binit2 <- c(0.0821, 2.4492, 0.1223, 1.7864, 0.0799, -0.2864, 0.0055, -0.0327, 0.0017,
0.3313, 0.9763, 0.9918, -0.4402)
M2 <- jointLPM(fixed = HIER~I(age-65)*male,
                random = ~I(age-65),
                subject = "ID",
                link = "thresholds",
                survival = Surv(age_init,agedem,dem)~male,
                sharedtype = 'CL',
                var.time = "age",
                data = paq,
                methInteg = "QMC",
                nMC = 1000,
                B=binit2,
                nproc=2)
summary(M2)
```

`loglik` *Wrapper to the Fortran subroutine that computes the log-likelihood*

Description

Computes the log-likelihood of a jointLPM model

Usage

```
loglik(
  b0,
  Y0,
  X0,
```

```
    Tentr0,
    Tevt0,
    Devt0,
    ind_survint0,
    idea0,
    idg0,
    idcor0,
    idcontr0,
    idsurv0,
    idtdv0,
    typrisq0,
    nz0,
    zi0,
    nbevt0,
    idtrunc0,
    logspecif0,
    ny0,
    ns0,
    nv0,
    nobs0,
    nmes0,
    idiag0,
    ncor0,
    nalea0,
    npm0,
    nfix0,
    bfix0,
    epsY0,
    idlink0,
    nbzitr0,
    zitr0,
    uniqueY0,
    indiceY0,
    indiceYinf0,
    nvalSPLORD0,
    fix0,
    methInteg0,
    nMC0,
    dimMC0,
    seqMC0,
    idst0,
    nXcl0,
    Xcl_Ti0,
    Xcl_GK0,
    Xcs_Ti0,
    Xcs_GK0,
    nonlin0,
    centerpoly0,
```

```
  expectancy0
)
```

Arguments

b0	b0
Y0	Y0
X0	X0
Tentr0	Tentr0
Tevt0	Tevt0
Devt0	Devt0
ind_survint0	ind_survint0
idea0	idea0
idg0	idg0
idcor0	idcor0
idcontr0	idcontr0
idsurv0	idsurv0
idtdv0	idtdv0
typrisq0	typrisq0
nz0	nz0
zi0	zi0
nbevt0	nbevt0
idtrunc0	idtrunc0
logspecif0	logspecif0
ny0	ny0
ns0	ns0
nv0	nv0
nobs0	nobs0
nmes0	nmes0
idiag0	idiag0
ncor0	ncor0
nalea0	nalea0
npm0	npm0
nfix0	nfix0
bfix0	bfix0
epsY0	epsY0
idlink0	idlink0
nbzitr0	nbzitr0
zitr0	zitr0

uniqueY0	uniqueY0
indiceY0	indiceY0
indiceYinf0	indiceYinf0
nvalSPLORD0	nvalSPLORD0
fix0	fix0
methInteg0	methInteg0
nMC0	nMC0
dimMC0	dimMC0
seqMC0	seqMC0
idst0	idst0
nXcl0	nXcl0
Xcl_Ti0	Xcl_Ti0
Xcl_GK0	Xcl_GK0
Xcs_Ti0	Xcs_Ti0
Xcs_GK0	Xcs_GK0
nonlin0	nonlin0
centerpoly0	centerpoly0
expectancy0	expectancy0

Value

the log-likelihood

`predictRE` *Random effects prediction*

Description

The random effects of a `jointLPM` model are predicted as the mode of the conditional distribution of the random effects given the subject's longitudinal and survival observations.

Usage

```
predictRE(x, data, control = NULL, variance = FALSE)
```

Arguments

x	a `jointLPM` object
data	a data.frame object containing the data from the predictions are to be computed
control	optional list of arguments to be passed to the `marqLevAlg` optimization function.
variance	logical indicating whether the variance of the estimated predictions should be returned. Default ot FALSE.

Value

a matrix containing the predicted random effects (in columns) for each subject (in rows). If `variance = TRUE`, returns a list containing the previous matrix and a vector containing the upper part of each variance matrix.

`print.jointLPM` *Brief summary of a joint latent process model*

Description

This function provides a brief summary of model estimated with the `jointLPM` function.

Usage

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

Arguments

`x`	an object inheriting from class `jointLPM` for a joint latent process model.
`...`	further arguments to be passed to or from other methods. They are ignored in this function.

Value

the function is used for its side effects, no value is returned.

Author(s)

Viviane Philipps, Tiphaine Saulnier and Cecile Proust-Lima

`REconddensity` *REconddendity*

Description

Wrapper to the Fortran subroutine computing the conditional density of the random effects

Usage

```
REconddensity(
  ui0,
  Y0,
  X0,
  Tentr0,
  Tevt0,
  Devt0,
  ind_survint0,
  idea0,
  idg0,
  idcor0,
  idcontr0,
  idsurv0,
  idtdv0,
  typrisq0,
  nz0,
  zi0,
  nbevt0,
  idtrunc0,
  logspecif0,
  ny0,
  nv0,
  nobs0,
  nmes0,
  idiag0,
  ncor0,
  nalea0,
  npm0,
  b0,
  epsY0,
  idlink0,
  nbzitr0,
  zitr0,
  uniqueY0,
  indiceY0,
  nvalSPLORD0,
  idst0,
  nXcl0,
  Xcl_Ti0,
  Xcl_GK0,
  Xcs_Ti0,
  Xcs_GK0,
  nonlin0,
  centerpoly0
)
```

Arguments

ui0	ui0
Y0	Y0
X0	X0
Tentr0	Tentr0
Tevt0	Tevt0
Devt0	Devt0
ind_survint0	ind_survint0
idea0	idea0
idg0	idg0
idcor0	idcor0
idcontr0	idcontr0
idsurv0	idsurv0
idtdv0	idtdv0
typrisq0	typrisq0
nz0	nz0
zi0	zi0
nbevt0	nbevt0
idtrunc0	idtrunc0
logspecif0	logspecif0
ny0	ny0
nv0	nv0
nobs0	nobs0
nmes0	nmes0
idiag0	idiag0
ncor0	ncor0
nalea0	nalea0
npm0	npm0
b0	b0
epsY0	epsY0
idlink0	idlink0
nbzitr0	nbzitr0
zitr0	zitr0
uniqueY0	uniqueY0
indiceY0	indiceY0
nvalSPLORD0	nvalSPLORD0
idst0	idst0

nXcl0	nXcl0
Xcl_Ti0	Xcl_Ti0
Xcl_GK0	Xcl_GK0
Xcs_Ti0	Xcs_Ti0
Xcs_GK0	Xcs_GK0
nonlin0	nonlin0
centerpoly0	centerpoly0

Value

the log-density

`removeNA` *removeNA*

Description

Remove missing values

Usage

```
removeNA(form, data)
```

Arguments

form	a list of formula
data	a data frame containing all variables mentionned in form

Details

Caution : if several response variables are included in 'form', then the resulting data frame will contain one column (named outcome) grouping all these response variables. The irst lines will refer to the first response variables, a second block to the second, etc. All covariates will be repeated for each response variable.

Value

a list containing

newdata	a data frame created from data, containing only the variables used in the formula, and without any missing values
Y	the names of the response variables
YlinesNA	the lines removed from each response variables
nmes	the number of measures for each response variable
X	the names of the covariates
XlinesNA	the lines removed from data because of missing values in the covariates

Examples

```
 d <- data.frame(y1=c(3,6,4,2), y2=c(6.5,NA,9.5,4.5), x=c(0,0,1,0))
 removeNA(form=list(y1+y2~x), data=d)
## 1      3.0 0   | first block referring to y1
## 2      6.0 0   |
## 3      4.0 1   |
## 4      2.0 0   |
## 11     6.5 0     | second block referring to y2
## 31     9.5 1     |
## 41     4.5 0     |
```

`sojournTime` *Computation of sojourn time under specific condition*

Description

This function computes posterior computations from a joint shared random-effect model with ordinal longitudinal outcomes, aka a joint item response theory model. Specifically, the function computes the expected time spent before reaching a given level of impairment specified by one or multiple items for a covariate profile.

Usage

```
sojournTime(
  x,
  maxState,
  condState = NULL,
  newdata,
  var.time,
  startTime = 0,
  nMC = 1000,
  upperTime = 150,
  subdivisions = 100L,
  rel.tol = .Machine$double.eps^0.25,
  draws = FALSE,
  ndraws = 2000,
  returndraws = FALSE,
  cl = NULL,
  seed = NULL
)
```

Arguments

`x`	an object of class `jointLPM` representing of joint shared random effects model with binary or ordinal longitudinal outcome(s)

maxState	a list specifying the items and the corresponding levels defining the maximum state for the computation of the sojourn time. For instance `maxState=list(Y = 3)` will compute the expected sojourn time corresponding to an impairment of Y lower or equal to 3.
condState	an optional list specifying the initial state at start time (argument `startTime`) from which to compute the residual sojourn time. The state can be defined either as a single value or as an interval. For example `condState=list(Y = 2)` means that the state at start time is equal to 2, whereas `condState=list(Y = c(1, 2))` means that the state at start time is between 1 and 2 (both included).
newdata	a one line data frame specifying the covariate profile for which the sojourn time is computed.
var.time	a character string specifying the name of the time variable in the longitudinal submodel. Note that this time covariate should not be included in newdata.
startTime	a numeric value specifying the time from which the residual sojourn time is computed (the lower bound of the integral over time). Default to 0 for the expected sojourn time over lifetime.
nMC	an integer giving the number of Monte Carlo simulations used to compute the integral over the random effects.
upperTime	a numeric specifying the upper bound of the integral over time. Default to 150 (150 years as an approximation of infinity).
subdivisions	passed to the `integrate` function.
rel.tol	passed to the `integrate` function.
draws	logical indicating if 95% confidence interval should be computed. Default to FALSE.
ndraws	integer giving the number of draws to be used to compute the 95% confidence interval. Default to 2000.
returndraws	logical indicating if the `ndraws` results should be returned. Default to FALSE.
cl	either a cluster created with `makeCluster` or an integer specifying the number of cores that should be used for computation. Only used with draws = TRUE.
seed	integer only used with `draws = TRUE` to set the random seed.

Details

1. Lifetime expected sojourn time: the expected time before reaching level k at item(s) Y is the integral over time t, from 0 to infinity, of P(Y(t) <= k, T > t), i.e., int_0^infty P(Y(t) <= k, T > t) dt.

2. Residual expected sojourn time from a time s: conditionally on being below m at item(s) Y at time s and being alive at time s, the sojourn time below level k at item(s) Y is computed as: int_s^infty P(Y(t) <= k, T > t | Y(s) <= m, T > s) dt = int_s^infty P(Y(t) <= k, T > t, Y(s) <= m) dt / P(Y(s) <= m, T > s)

Value

if `draws = FALSE`, returns a single value. If `draws = TRUE` and `returndraws = FALSE`, returns the median, the 2.5% and 97.5% quantiles, the mean, the standard deviation and the number of removed draws (eventually due to computational issues). If `draws = TRUE` and `returndraws = TRUE`, returns the `ndraws` values.

Author(s)

Viviane Philipps and Cecile Proust-Lima

Examples

```
library(lcmm)
paq <- paquid[which(paquid$age_init < paquid$agedem), ]
paq$age65 <- (paq$age - 65) / 10
paq$ageinit65 <- (paq$age_init - 65) / 10
paq$agedem65 <- (paq$agedem - 65) / 10

#### Estimation of the joint model with one ordinal longitudinal item
## Not run:
M2 <- jointLPM(fixed = HIER ~ age65 * male,
                random = ~ age65,
                subject = "ID",
                link = "thresholds",
                survival = Surv(ageinit65, agedem65, dem) ~ male,
                sharedtype = 'RE',
                var.time = "age65",
                data = paq,
                methInteg = "QMC",
                nMC = 1000,
                B = c(0.6, 2.399, -0.409, 2.076, 6.338, 0.994, -0.223, -0.005,
                     -0.299, 0.174, 0.523, 1.044, 1.064, 0.506))

#### Computation of the expected lifetime sojourn time with HIER impairment
up to 2 (HIER = 2)
#### (=int_0^150 P(HIER(t) <= 2, T > t) dt)
sojournTime(M2, list(HIER = 2), newdata = data.frame(male = 0),
var.time = "age65")

#### Computation of the expected residual time with maximum HIER impairment
 of 2 (HIER = 2), given the impairment was 1 at time 0.5
#### (=int_0.5^150 P(HIER(t) <= 2, T > t | HIER(0.5) <= 1, T > 0.5) dt)
sojournTime(M2, list(HIER = 2), condState = list(HIER = 1), startTime = 0.5,
newdata = data.frame(male = 0), var.time = "age65")

## End(Not run)
```

StandardMethods *Standard methods for estimated models*

Description

coef and vcov methods for estimated jointLPM models.

Usage

```
## S3 method for class 'jointLPM'
coef(object, ...)
```

Arguments

object	an object of class jointLPM
...	other arguments. There are ignored in these functions.

Value

For coef, the vector of the estimated parameters.

For vcov, the variance-covariance matrix of the estimated parameters.

Author(s)

Viviane Philipps

---

summary.jointLPM *Summary of a joint latent process model*

Description

This function provides a summary of model estimated with the jointLPM function.

Usage

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

Arguments

object	an object inheriting from class jointLPM for a joint latent process model.
...	further arguments to be passed to or from other methods. They are ignored in this function.

Value

The function is mainly used for its side effects. It returns invisibily a list of two matrices containing the estimates, their standard errors, Wald statistics and associated p-values for the survival submodel (first element of the list) and for the mixed model's fixed effects (second element).

Author(s)

Viviane Philipps, Tiphaine Saulnier and Cecile Proust-Lima

`WaldMult` *WaldMult Performs multivariate Wald tests.*

Description

Given a vector of estimated parameters (coef = (theta_1, ... theta_n)') and their associated variance matrix (vcov = Var(coef)), the WaldMult function performs either the test of the null hypothesis :

Usage

```
WaldMult(
  Mod,
  coef,
  vcov,
  pos = NULL,
  contrasts = NULL,
  A = NULL,
  name = NULL,
  value = 0
)
```

Arguments

`Mod`	an object for which the `coef` and `vcov` methods are diefined.
`coef`	if `Mod` is missing, the vector of estimated parameters
`vcov`	if `Mod` is missing, the matrix of variance of coef
`pos`	vector containing the indices of the parameters involved in the test. Required if A is not specified.
`contrasts`	optional vector containing the the c1, ..., cm value in case 2 (see details). Should be of same lengtht as pos.
`A`	matrix A of case 3 (see details). A should have as many columns as parameters in coef.
`name`	characters string giving the name of the test printed in the output (the row names of the output). By default, the name's test is the null hypothesis.
`value`	the value against which to test. By default, value=0.

Details

1. H0 : theta_m = (theta_j1, ..., theta_jm)' = 0 ie, tests if a subset of m parameters are simultaneously equal to zero. The Wald statistic is w = theta_m' * Var(theta_m)^(-1) * theta_m and the associated p-value is obtained by p = P(chi2_(dof=m) > w)

2. H0 : theta = c1*theta_j1 + ... + cm*theta_jm = 0 ie, tests if a combination of m parameters is equal to zero. The Wald statistic is w = theta^2 / Var(theta) = (c1*theta_j1 + ... + cm*theta_jm)^2 / Var(c1*theta_j1 + ... + cm*theta_jm) and the associated p-value is obtained by p = P(chi2_(dof=1) > w)

3. H0 : Theta = A*coef = 0 with A a matrix with m lines and n columns. ie, tests if multiple combinations of parameters are simultaneously equal to zero. The Wald statistic is w = Theta * Var(Theta)^(-1) * Theta' = (A*coef)' * (A*vcov*A')^(-1) * (A*coef) and the associated p-value is obtained by p = P(chi2_(dof=m) > w)

The function can be applied to any object having a `coef`

Value

If contrasts is NULL, the function returns a matrix with 1 row and 2 columns containing the value of the Wald test's statistic and the associated p-value.

If contrasts is not NULL, the function returns a matrix with 1 row and 4 columns containing the value of the coefficient (dot product of pos and contrasts), his standard deviation, the value of the Wald test's statistic and the associated p-value.

If A is specified, the function returns a matrix with m rows and 2 columns containing the value of the Wald test's statistic and the associated p-value.

Examples

```
require(lcmm)
m <- hlme(Y ~ Time + X2, random = ~ 1, subject = "ID", data = data_hlme)
summary(m)

mm <- hlme(Y ~ -1 + Time + factor(X2), random = ~ 1, subject = "ID", data = data_hlme)
summary(mm)

## Retrieve from model mm the difference between X2 levels as in model m :
WaldMult(Mod = mm, pos = c(2, 3), contrasts = c(-1, 1))
## or
WaldMult(coef = coef(mm), vcov = vcov(mm), A = matrix(c(0, -1, 1, 0, 0), 1, 5))
```

Index