Package: LSJM (via r-universe)

October 15, 2026

Title Estimate Location-Scale Joint Models

Version 0.1.1

Description Estimation of mixed models including a subject-specific variance that can be time- and covariate-dependent or defined for within- and between-visit variability. In the joint modeling framework, the package handles left truncation, interval censoring, and multistate models, and allows a flexible dependence structure between competing events and the longitudinal marker. Estimation is performed in a frequentist framework using the Marquardt-Levenberg algorithm. Methods are described in Courcoul et al. (2025) <doi:10.1002/sim.70244> and in Courcoul et al. (2026) <doi:10.1002/bimj.70123>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, marqLevAlg, survival, flexsurv, ggplot2, splines, spacefillr, survminer, foreach, doParallel, SmoothHazard, parallel, mvtnorm, dplyr

Depends R (>= 3.5.0)

NeedsCompilation yes

Author Léonie Courcoul [aut, cre], Antoine Barbieri [aut], Hélène Jacqmin-Gadda [aut]

Maintainer Léonie Courcoul <courcoul.leonie498@gmail.com>

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

Date/Publication 2026-09-14 21:30:07 UTC

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

RemoteRef HEAD

RemoteSha c28a350bb569c743595259f7ed51e7db9074ccf5

Contents

data.manag.long 2
data.manag.sigma 3
data.manag.surv 3
dynpred 4
initial.long 8
lsjm 8
lsmm 16
plot.lsjm 21
predict.lsjm 27
ranef 31
survmarg 33
threeC 36

Index **38**

`data.manag.long` *Management of longitudinal data*

Description

Management of longitudinal data

Usage

```
data.manag.long(formGroup, formFixed, formRandom, data.long1)
```

Arguments

`formGroup`	A formula which indicates the group variable
`formFixed`	A formula which indicates the fixed effects for the longitudinal submodel
`formRandom`	A formula which indicates the random effects for the longitudinal submodel
`data.long1`	A dataframe with the longitudinal data

Value

A list with the following components :

`data_long` a clean dataframe for the longitudinal data

`y.new.prog` the vector of responses variable

`X` a matrix with the fixed effects

`U` a matrix with the random effects

`id` a vector with the identification of individuals

`offset` a vector with the number of measurements for each individual

`I` an integer, the number of individuals

data.manag.sigma *Management of data for residual part*

Description

Management of data for residual part

Usage

```
data.manag.sigma(formGroup, formFixed, formRandom, data.long1)
```

Arguments

formGroup	A formula which indicates the group variable
formFixed	A formula which indicates the fixed effects for the longitudinal submodel
formRandom	A formula which indicates the random effects for the longitudinal submodel
data.long1	A dataframe with the longitudinal data

Value

A list with the following components :

X a matrix with the fixed effects

U a matrix with the random effects

data.manag.surv *Management of survival data*

Description

Management of survival data

Usage

```
data.manag.surv(formGroup, formSurv, data.long1)
```

Arguments

formGroup	A formula which indicates the group variable
formSurv	A formula which indicates the survival submodel
data.long1	A dataframe with the longitudinal data

dynpred *dynpred: compute the dynamic predictions of an event.*

Description

This function can be used only with an `lsjm` object. Individual dynamic predictions can be computed and plotted. They are defined as the predicted probability of having event k between time s and $s+t$ given that the subject i has not experienced any event before time s, and knowing all marker measures collected until time s, denoted by $Y_i(s)$, and the set of estimated parameters. The prediction is defined for subject i by: #'

$$\pi_i^{0k}(s,t;\widehat{\theta}) = P(s < T_i < s+t, \delta_i = k \mid T_i > s, \mathcal{Y}_i(s), \widehat{\theta}) = \frac{\int \left[\int_s^{s+t} \exp\left(-\sum_{c=1}^{K} \Lambda_i^{0c}(u|r_i,\widehat{\theta})\right) \lambda_i^{0k}(u|r_i,\widehat{\theta}) du\right] f(\mathcal{Y}_i(s)$$
$$\int \exp\left(-\sum_{c=1}^{K} \Lambda_i^{0c}(s|r_i,\widehat{\theta})\right) f(\mathcal{Y}_i(s)|r_i,\widehat{\theta}) f(r_i|\widehat{\theta}$$

For illness-death and competing risks models, $K = 2$, whereas for a model with a single event, $K = 1$. For illness-death and single-event models, only the prediction from state (0) to state (1) can be computed, while for competing risks, transitions (0→1) and (0→2) can both be predicted.

As in the estimation procedure, the integral over the random effects is computed by QMC approximation and the integral over time by the Gauss-Kronrod quadrature.

The 95\ which can be computationally intensive. For L large enough and $l = 1, ..., L$ (e.g. $L = 1000$):

- Generate $\widetilde{\theta}^{(l)} \sim \mathcal{N}(\widehat{\theta}, V(\widehat{\theta}))$, where $V(\widehat{\theta})$ is the inverse of the Hessian matrix at $\widehat{\theta}$;
- Compute $\widetilde{\pi}_i^{(l)}(s,t;\widetilde{\theta}^{(l)})$ from the equation above;
- Compute the 95\ 97.5th percentiles of the L-sample of $\widetilde{\pi}_i^{(l)}(s,t;\widetilde{\theta}^{(l)})$.

Usage

```
dynpred(object, newdata, s, horizon, event, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_classicCR'
dynpred(object, newdata, s, horizon, event, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_classicIDM'
dynpred(object, newdata, s, horizon, event = NULL, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_classicSingle'
dynpred(object, newdata, s, horizon, event = NULL, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_covDepCR'
dynpred(object, newdata, s, horizon, event, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_covDepIDM'
dynpred(object, newdata, s, horizon, event = NULL, CI = 95, nb.draws = 1000)
```

```
## S3 method for class 'lsjm_covDepSingle'
dynpred(object, newdata, s, horizon, event = NULL, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_interintraCR'
dynpred(object, newdata, s, horizon, event, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_interintraIDM'
dynpred(object, newdata, s, horizon, event = NULL, CI = 95, nb.draws = 1000)

## S3 method for class 'lsjm_interintraSingle'
dynpred(object, newdata, s, horizon, event = NULL, CI = 95, nb.draws = 1000)
```

Arguments

object	An lsjm object.
newdata	A dataset containing the individuals for whom predictions are to be computed.
s	The landmark time (a single numeric value).
horizon	The horizon time of prediction. The function computes the probability of experiencing the event between s and horizon. This can be a vector or a single value.
event	An integer indicating for which event the prediction is computed. In the case of a * competing risks model, it can be 1 or 2, whereas for other models it can only be 1.
CI	An integer between 0 and 100 indicating the confidence level (default: 95 for a 95% CI). If CI = NULL, no confidence interval is computed.
nb.draws	An integer giving the number of Monte Carlo draws used to compute the confidence interval (default: 1000).

Value

A dataframe containing the table of predictions (table.pred).

Examples

```
set.seed(123)
data <- data.frame(
         ID = rep(1:100, each = 3),
         time = rep(1:3, 100),
         y = rnorm(300),
         event = rep(rbinom(100,1,0.5), each = 3),
         time_event = rep(runif(100), each = 3)
         )


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
```

```
  formGroup = ~ ID,
  formVar = "standard",
  timeVar = "time",
  data.long = data,
  S1 = 5,
  S2 = 5,
  nproc = 1
  )

fit <- lsjm(
     Objectlsmm = m0,
     survival_type = "Single",
     formSurv_01 = ~ 1,
     sharedtype_01 = "value",
     hazardBase_01 = "Weibull",
     delta1 = ~ event,
     Time_T = ~ time_event,
     S1 = 10,
     S2 = 10,
     nproc = 1
     )

ind1 <- data[which(data$ID == 1),]
dynpEvent <- dynpred(fit, ind1,  s = 2, horizon = seq(2.1,3,0.1),
                 event = 1, nb.draws = 100)


## Not run:
library(dplyr)

# Begin by running the examples from the lsjm function (see ?lsjm).
data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10


# Example: prediction of the risk to be diagnosed with dementia
# (dynpDementia) and to die (dynpDeath) between 85 and 95 years old
# given the blood pressure measurements before 85 years old for
# individual "10003", with a competing risk model
threeC_ex2 <- threeC[,c("ID", "SBP", "age.visit65", "age0_65",
                        "age.final65", "age.last65", "age.first65",
                        "dem", "death", "sex","num.visit")]

threeC_ex2$age65_CR <- NA
threeC_ex2$age65_CR[which(threeC_ex2$dem == 1)] <-
  (threeC_ex2$age.last65[which(threeC_ex2$dem == 1)] +
     threeC_ex2$age.first65[which(threeC_ex2$dem == 1)])/2
threeC_ex2$age65_CR[which(threeC_ex2$dem == 0)] <-
  threeC_ex2$age.final65[which(threeC_ex2$dem == 0)]

threeC_ex2$demCR <- threeC_ex2$dem
threeC_ex2$deathCR <- NA
```

```
threeC_ex2$deathCR[which(threeC_ex2$dem == 1)] <- 0
threeC_ex2$deathCR[which(threeC_ex2$dem == 0)] <-
  threeC_ex2$death[which(threeC_ex2$dem == 0)]

threeC_ex2 <- threeC_ex2 %>% group_by(ID) %>% dplyr::filter(age.visit65 <= age65_CR)

threeC_ex2 <- threeC_ex2[,c("ID","SBP","age.visit65","num.visit","age0_65",
                            "demCR","deathCR","age65_CR","sex")]

ind2 <- threeC_ex2[which(threeC_ex2$ID == 3),]

ind2 <- as.data.frame(ind2)

m2 <- lsmm(formFixed = SBP ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex2,
               formVar = "inter-intra",
               random_inter = TRUE,
               random_intra = TRUE,
               formGroupVisit = ~num.visit,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)


l2 <- lsjm(Objectlsmm = m2,
           survival_type = 'CR',
           formSurv_01 = ~ sex,
           formSurv_02 = ~ sex,
           sharedtype_01 = c("value", "variability inter"),
           sharedtype_02 = c("value", "variability inter",
                             "variability intra"),
           hazardBase_01 = "Weibull",
           hazardBase_02 = "Weibull",
           delta1 = ~ demCR,
           delta2 = ~ deathCR,
           Time_T0 = ~ age0_65,
           Time_T = ~ age65_CR,
           nproc = 5,
           S1 = 1000,
           S2 = 2000)


dynpDementia <- dynpred(l2, ind2,  s = 2, horizon = seq(2.1,3,0.1),
                  event = 1, nb.draws = 1000)

dynpDeath <- dynpred(l2, ind2,  s = 2, horizon = seq(2.1,3,0.1),
                  event = 2, nb.draws = 1000)

## End(Not run)
```

`initial.long` *Initialisation of Longitudinal Submodel*

Description

Initialisation of Longitudinal Submodel

Usage

```
initial.long(formFixed, formRandom, idVar, data.long1, ncX, nproc = nproc)
```

Arguments

`formFixed`	A formula which indicates the fixed effects for the longitudinal submodel
`formRandom`	A formula which indicates the random effects for the longitudinal submodel
`idVar`	A character, indicates the name of the group variable
`data.long1`	A dataframe with the longitudinal data
`ncX`	An integer, the number of columns of matrix X, ie, the number of fixed effects
`nproc`	An integer, the number of cores for parallel computation

Value

A list with the following components :

`long_model` the result of the hlme function

`priorMean.beta` the estimated parameters for fixed effects in the linear mixed effects model

`sigma` the estimated sigma of the model

`lsjm` *lsjm : Estimation of a location-scale joint model for longitudinal data with flexible subject-specific variability and complex survival structures*

Description

This function fits joint models in which the residual error variance is allowed to be subject-specific and the survival submodel may include either a single event, competing risks, or an illness-death process. Up to nine different model structures can be estimated (see *Details*). Parameters are estimated by maximum likelihood using a Marquardt-Levenberg algorithm.

Usage

```
lsjm(
  Objectlsmm,
  survival_type = c("Single", "CR", "IDM"),
  formSurv_01,
  formSurv_02 = NULL,
  formSurv_12 = NULL,
  sharedtype_01,
  sharedtype_02 = NULL,
  sharedtype_12 = NULL,
  hazardBase_01,
  hazardBase_02 = NULL,
  hazardBase_12 = NULL,
  delta1,
  delta2 = NULL,
  Time_T,
  Time_L = NULL,
  Time_R = NULL,
  Time_T0 = NULL,
  formSlopeFixed = NULL,
  formSlopeRandom = NULL,
  index_beta_slope = NULL,
  index_b_slope = NULL,
  nb.knots.splines = c(1, 1, 1),
  nb_pointsGK = 15,
  S1 = 1000,
  S2 = 5000,
  binit = NULL,
  nproc = 1,
  clustertype = "PSOCK",
  maxiter = 500,
  print.info = FALSE,
  file = NULL,
  epsa = 1e-04,
  epsb = 1e-04,
  epsd = 1e-04
)
```

Arguments

`Objectlsmm`	The result from the `lsmm()` function.
`survival_type`	Character string specifying the survival scheme: `"Single"`, `"CR"` (competing risks), or `"IDM"` (illness–death model).
`formSurv_01`	One-sided formula specifying covariates for the 0–1 transition.
`formSurv_02`	One-sided formula specifying covariates for the 0–2 transition.
`formSurv_12`	One-sided formula specifying covariates for the 1-2 transition.

`sharedtype_01` Character vectors defining the dependence structure between the longitudinal and the survival submodel for the 0-1 transition. For standard mixed models, valid options include `"value"`, `"slope"`, and `"random effects"`. For location–scale mixed models, additional terms such as `"variability"`, `"variability inter"`, or `"variability intra"` can be included when subject-specific variances are modelled.

`sharedtype_02` Character vectors defining the dependence structure between the longitudinal and the survival submodel for the 0-2 transition. For standard mixed models, valid options include `"value"`, `"slope"`, and `"random effects"`. For location–scale mixed models, additional terms such as `"variability"`, `"variability inter"`, or `"variability intra"` can be included when subject-specific variances are modelled.

`sharedtype_12` Character vectors defining the dependence structure between the longitudinal and the survival submodel for the 1-2 transition. For standard mixed models, valid options include `"value"`, `"slope"`, and `"random effects"`. For location–scale mixed models, additional terms such as `"variability"`, `"variability inter"`, or `"variability intra"` can be included when subject-specific variances are modelled.

`hazardBase_01` Character strings specifying the baseline hazard function for transition 0-1 one of `"Exponential"`, `"Weibull"`, `"Gompertz"`, or `"Splines"`.

`hazardBase_02` Character strings specifying the baseline hazard function for transition 0-2 one of `"Exponential"`, `"Weibull"`, `"Gompertz"`, or `"Splines"`.

`hazardBase_12` Character strings specifying the baseline hazard function for transition 1-2 one of `"Exponential"`, `"Weibull"`, `"Gompertz"`, or `"Splines"`.

`delta1` One-sided formula defining the event indicator for the first event (1 for event, 0 otherwise).

`delta2` One-sided formula defining the indicator for the second event (1 for event, 0 otherwise).

`Time_T` One-sided formula specifying the event time variable.

`Time_L` For IDM models, one-sided formula giving the time of the last observation in state (0).

`Time_R` For IDM models, one-sided formula giving the first observed time in state (1).

`Time_T0` One-sided formula specifying the time of entry (for delayed entry).

`formSlopeFixed` One-sided formula for the time derivative of the fixed effects (if "slope" is included in `sharedtype`).

`formSlopeRandom` One-sided formula for the time derivative of the random effects (if "slope" is included in `sharedtype`).

`index_beta_slope` Vector indicating the indices of fixed-effect parameters used in the slope association.

`index_b_slope` Vector indicating the indices of random-effect parameters used in the slope association.

`nb.knots.splines` Integer giving the number of internal knots for spline-based baseline hazards.

nb_pointsGK	Integer specifying the number of Gauss–Kronrod quadrature points (between 7 and 15; default is 15).
S1	Integer specifying the number of QMC draws for the first step.
S2	Integer specifying the number of QMC draws for the second step.
binit	Optional vector of initial parameters values.
nproc	Integer specifying the number of processors for parallel computing.
clustertype	Character string indicating the cluster type supported by makeCluster.
maxiter	Optional integer specifying the maximum number of iterations for the Marquardt–Levenberg algorithm (default to 100).
print.info	Logical indicating whether iteration details should be printed (False by default).
file	Optional character string giving the name of the file where iteration outputs are written (if print.info = TRUE).
epsa	Optional numeric threshold for convergence based on parameter stability.
epsb	Optional numeric threshold for convergence based on objective function stability.
epsd	Optional numeric threshold for the relative distance to the maximum. This criterion has the nice interpretation of estimating the ratio of the approximation error over the statistical error, thus it can be used for stopping the iterative process whatever the problem.

Details

A joint model is composed of two submodels: (1) a linear mixed model or location-scale mixed model(see ?LSJM::lsmm), and (2) a survival model.

Three types of survival processes are supported and are described below.

A. A single event: proportional hazards model

In this case, we consider only a single event. Let $T_i = \min(T_{i1}^*, C_i)$ denote the observed time, where T_{i1}^* is the true event time and C_i the censoring time for subject i. Let $\delta_i \in \{0, 1\}$ be the event indicator. The hazard function is given by:

$$\lambda_i^{01}(t|r_i) = \lambda_0(t) \exp\left(W_i^{01\top}\gamma^{01} + g_y^{01}(b_i, t)^\top \alpha_b^{01} + g_\tau^{01}(\tau_i, t)^\top \alpha_\tau^{01}\right)$$

where:

- $\lambda_0(t)$ is the baseline hazard function,
- W_i^{01} is a vector of baseline covariates with associated coefficients γ^{01},
- α_b^{01} are regression coefficients for the function g_y, representing the association between the event risk and the mean trajectory of Y,
- α_τ^{01} are regression coefficients for the function g_τ, representing the association between the event risk and the residual variance.

The association function $g_y(b_i, t)$ can be defined as:

- $g_y(b_i, t) = \tilde{y}_i(t)$ — current value,

- $g_y(b_i, t) = \tilde{y}_i'(t) = \frac{\partial \tilde{y}_i(t)}{\partial t}$ — current slope,
- $g_y(b_i, t) = (\tilde{y}_i(t), \tilde{y}_i'(t))$ — both value and slope,
- $g_y(b_i, t) = b_i$ — random effects.

The association function $g_\tau(\tau_i, t)$ is defined according to the longitudinal model type:

- If a standard linear mixed model with homogeneous residual variance is used: $g_\tau(\tau_i, t) = 0$.
- If a location–scale mixed model with time- or covariate-dependent variance is used: $g_\tau(\tau_i, t) = \sigma_i(t)$.
- If both within- and between-visit variances are modeled: $g_\tau(\tau_i, t) = (\sigma_i, \kappa_i)^\top$, where $\alpha_\tau = (\alpha_\sigma, \alpha_\kappa)$ correspond to between-visit and within-visit variabilities, respectively.

The baseline hazard function $\lambda_0(t)$ can follow different parametric forms:

- Exponential: $\lambda_0(t) = \exp(\alpha_0)$,
- Weibull: $\lambda_0(t) = \zeta^2 t^{\zeta^2 - 1} \exp(\alpha_0)$,
- Gompertz: $\lambda_0(t) = \kappa_1^2 \exp(\kappa_2 t)$,
- Cubic B-splines with Q knots: $\lambda_0(t) = \exp\left(\sum_{q=1}^{Q+4} \eta_q B_q(t, \nu)\right)$, where $B_q(t, \nu)$ denotes the q-th B-spline basis function with knot vector ν.

B. Competing events: cause-specific model

When multiple types of events are possible, two competing causes can be modeled. Let $T_i = \min(T_{i1}^*, T_{i2}^*, C_i)$ be the observed time, where T_{ik}^* is the true event time for cause k (with $k \in \{1, 2\}$), and C_i the censoring time. The event indicator is $\delta_i \in \{0, 1, 2\}$, where $\delta_i = k$ if cause k occurred and 0 otherwise. The cause-specific hazard is defined as:

$$\lambda_i^{0k}(t) = \lambda_0^{0k}(t) \exp\left(W_i^{0k\top} \gamma^{0k} + g_y^{0k}(b_i, t)^\top \alpha_b^{0k} + g_\tau^{0k}(\tau_i, t)^\top \alpha_\tau^{0k}\right)$$

The definitions of λ_0^{0k}, W_i^{0k}, g_y^{0k}, and g_τ^{0k} follow the same principles as in the single-event model.
C. Semi-competing events: illness–death model In this setting, transition to state (1) may be interval-censored, whereas transition to state (2) is observed exactly. The observed event data for subject i are given by: $D_i = (T_{0i}, L_i, R_i, \delta_i^{(1)}, T_i, \delta_i^{(2)})^\top$, where:

- T_{0i} — entry time (in case of delayed entry),
- L_i — time of the last visit where the subject was in state (0),
- R_i — time of the first visit where the subject was observed in state (1),
- T_i — minimum between the transition time to state (2) and the censoring time,
- $\delta_i^{(1)} = \mathbb{1}_{R_i < T_i}$ — indicator of transition to state (1),
- $\delta_i^{(2)}$ — indicator of transition to state (2).

Transition intensities from state $k \in \{0, 1\}$ to state $l \in \{1, 2\}$ follow a proportional hazards model under the Markov assumption:

$$\lambda_i^{kl}(t|b_i, \tau_i) = \lambda_0^{kl}(t) \exp\left(W_i^{kl\top} \gamma^{kl} + g_y^{kl}(b_i, t)^\top \alpha_b^{kl} + g_\tau^{kl}(\tau_i, t)^\top \alpha_\tau^{kl}\right)$$

where the terms are defined analogously to the single-event model.

Value

An object of class `lsjm` containing:

`table.res` Table of parameter estimates and standard errors.

`result_step1` A `marqLevAlg` object with first-step estimation results.

`result_step2` A `marqLevAlg` object with second-step estimation results.

`info_conv_step1` Information on first-step convergence (criteria and computation time).

`info_conv_step2` Information on second-step convergence (criteria and computation time).

`control` List of control parameters used during estimation.

Examples

```
set.seed(123)

data <- data.frame(
         ID = rep(1:100, each = 3),
         time = rep(1:3, 100),
         y = rnorm(300),
         event = rep(rbinom(100,1,0.5), each = 3),
         time_event = rep(runif(100), each = 3)
         )


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
   formGroup = ~ ID,
   formVar = "standard",
   timeVar = "time",
   data.long = data,
   S1 = 5,
   S2 = 5,
   nproc = 1
   )

 fit <- lsjm(
      Objectlsmm = m0,
      survival_type = "Single",
      formSurv_01 = ~ 1,
      sharedtype_01 = "value",
      hazardBase_01 = "Weibull",
      delta1 = ~ event,
      Time_T = ~ time_event,
      S1 = 100,
      S2 = 100,
      nproc = 1
      )

## Not run:
```

```
library(dplyr)

# First, run the examples from the lsmm function (see ?lsmm).

# Example 1: Illness-death model with time-dependent subject-specific
# variability
data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10
threeC <- threeC
threeC <- dplyr::group_by(threeC, ID, num.visit)
threeC <- dplyr::mutate(threeC, SBPvisit = mean(SBP))
threeC_ex1 <- threeC[!duplicated(threeC[, c("ID", "num.visit")]),
                                 c("ID", "SBPvisit", "age.visit65", "sex","dem","death",
                                    "age.first65","age.last65","age.final65","age0_65")]

m1 <- lsmm(formFixed = SBPvisit ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex1,
               formVar = "cov-dependent",
               formFixedVar = ~ age.visit65+sex,
               formRandomVar = ~ age.visit65,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)


l1 <- lsjm(m1,
           survival_type = 'IDM',
           formSurv_01=~1,
           formSurv_02=~sex,
           formSurv_12=~sex,
           sharedtype_01 = c("value", "variability"),
           sharedtype_02 = c("value", "slope", "variability"),
           sharedtype_12 = c("value", "variability"),
           hazardBase_01 = "Weibull",
           hazardBase_02 = "Weibull",
           hazardBase_12 = "Splines",
           delta1=~dem,
           delta2=~death,
           Time_T =~age.final65,
           Time_L =~age.last65,
           Time_R =~age.first65,
           Time_T0 =~age0_65,
           formSlopeFixed =~1,
           formSlopeRandom =~1,
           index_beta_slope = c(2),
           index_b_slope = c(2),
           nb.knots.splines = c(0,0,1),
           S1 = 1000,
```

```
            S2 = 2000,
            nproc = 5)

summary(l1)


# Example 2: Competing risks model with between- and within-visit
# subject-specific variabilities

threeC_ex2 <- threeC[,c("ID", "SBP", "age.visit65", "age0_65",
                        "age.final65", "age.last65", "age.first65",
                        "dem", "death", "sex","num.visit")]

threeC_ex2$age65_CR <- NA
threeC_ex2$age65_CR[which(threeC_ex2$dem == 1)] <-
  (threeC_ex2$age.last65[which(threeC_ex2$dem == 1)] +
     threeC_ex2$age.first65[which(threeC_ex2$dem == 1)])/2
threeC_ex2$age65_CR[which(threeC_ex2$dem == 0)] <-
  threeC_ex2$age.final65[which(threeC_ex2$dem == 0)]

threeC_ex2$demCR <- threeC_ex2$dem
threeC_ex2$deathCR <- NA
threeC_ex2$deathCR[which(threeC_ex2$dem == 1)] <- 0
threeC_ex2$deathCR[which(threeC_ex2$dem == 0)] <-
  threeC_ex2$death[which(threeC_ex2$dem == 0)]

threeC_ex2 <- threeC_ex2 %>% group_by(ID) %>% dplyr::filter(age.visit65 <= age65_CR)

threeC_ex2 <- threeC_ex2[,c("ID","SBP","age.visit65","num.visit","age0_65",
                            "demCR","deathCR","age65_CR","sex")]

m2 <- lsmm(formFixed = SBP ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex2,
               formVar = "inter-intra",
               random_inter = TRUE,
               random_intra = TRUE,
               formGroupVisit = ~num.visit,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)


l2 <- lsjm(Objectlsmm = m2,
           survival_type = 'CR',
           formSurv_01 = ~ sex,
           formSurv_02 = ~ sex,
           sharedtype_01 = c("value", "variability inter"),
           sharedtype_02 = c("value", "variability inter",
                             "variability intra"),
```

```
            hazardBase_01 = "Weibull",
            hazardBase_02 = "Weibull",
            delta1 = ~ demCR,
            delta2 = ~ deathCR,
            Time_T0 = ~ age0_65,
            Time_T = ~ age65_CR,
            nproc = 5,
            S1 = 1000,
            S2 = 2000)

summary(l2)

## End(Not run)
```

`lsmm` *lsmm : Estimation of a linear mixed model for longitudinal data with flexible subject-specific variability.*

Description

This function fits linear mixed effects models for longitudinal data, allowing the residual variance to be subject-specific. Three different model types can be estimated (see *Details*).Parameters are estimated by maximum likelihood using a Marquardt-Levenberg algorithm.

Usage

```
lsmm(
  formFixed,
  formRandom,
  formGroup,
  timeVar,
  formVar = "standard",
  formFixedVar = NULL,
  formRandomVar = NULL,
  random_inter = FALSE,
  random_intra = FALSE,
  formGroupVisit = NULL,
  correlated_re = FALSE,
  data.long,
  S1 = 500,
  S2 = 5000,
  nproc = 1,
  clustertype = "PSOCK",
  maxiter = 100,
  print.info = FALSE,
  file = "",
```

```
  epsa = 1e-04,
  epsb = 1e-04,
  epsd = 1e-04,
  binit = NULL
)
```

Arguments

formFixed	Formula specifying the fixed effects of the longitudinal model.
formRandom	Formula specifying the random effects of the longitudinal model.
formGroup	Formula specifying the grouping variable (typically the subject ID).
timeVar	Character string specifying the time variable.
formVar	Character string indicating the type of variability: **"standard"** for a standard LMM, **"cov-dependent"** for a covariate-dependent residual variance, or **"inter-intra"** to distinguish inter- and intra-visit variability.
formFixedVar	Formula specifying the fixed effects for the variance predictor (if formVar == "cov-dependent").
formRandomVar	Formula specifying the random effects for the variance predictor (if formVar == "cov-dependent").
random_inter	Logical indicating whether the between-visit variability is subject-specific (used when formVar = "inter-intra").
random_intra	Logical indicating whether the within-visit variability is subject-specific (used when formVar = "inter-intra").
formGroupVisit	Formula specifying the visit indicator variable (used when formVar = "inter-intra").
correlated_re	Logical indicating whether the random effects for the mean and variance submodels are correlated (used when formVar is in c("cov-dependent", "inter-intra")).
data.long	Data frame containing the longitudinal data.
S1	Integer specifying the number of QMC draws for the first step.
S2	Integer specifying the number of QMC draws for the second step.
nproc	Integer specifying the number of processors for parallel computing.
clustertype	Character string indicating the cluster type supported by makeCluster.
maxiter	Optional integer specifying the maximum number of iterations for the Marquardt–Levenberg algorithm (default to 100).
print.info	Logical indicating whether iteration details should be printed (False by default).
file	Optional character string giving the name of the file where iteration outputs are written (if print.info = TRUE).
epsa	Optional numeric threshold for convergence based on parameter stability.
epsb	Optional numeric threshold for convergence based on objective function stability.
epsd	Optional numeric threshold for the relative distance to the maximum. This criterion has the nice interpretation of estimating the ratio of the approximation error over the statistical error, thus it can be used for stopping the iterative process whatever the problem.
binit	Optional vector of initial parameters values.

Details

The model is defined as:

$Y_{ij} = Y_i(t_{ij}) = \widetilde{Y}_i(t_{ij}) + \epsilon_{ij} = X_{ij}^\top \beta + Z_{ij}^\top b_i + \epsilon_{ij}$, where X_{ij} and Z_{ij} are vectors of explanatory variables for subject i at time t_{ij}, associated with the fixed-effect vector β and the subject-specific random-effect vector b_i, respectively.

A. Standard linear mixed model

In this case, $b_i \sim \mathcal{N}(0, B)$, with B an unstructured covariance matrix, and the measurement errors ϵ_{ij} are independent Gaussian errors with variance σ_ϵ^2.

B. Location-scale mixed model with time and/or covariate-dependent variability

In this model, the residual error variance can vary across subjects and over time: we assume the following specification for the residual error:

$\epsilon_{ij} \sim \mathcal{N}(0, \sigma_i^2)$ with $\log(\sigma_i(t_{ij})) = O_{ij}^\top \mu + M_{ij}^\top \tau_i$.#' where O_{ij} and M_{ij} are vectors of explanatory variables for subject i at visit j, associated with the fixed-effect vector μ and the subject-specific random-effect vector τ_i, respectively.

The random effects are assumed jointly Gaussian:

$$\begin{pmatrix} b_i \\ \tau_i \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_b & \Sigma_{\tau b} \\ \Sigma'_{\tau b} & \Sigma_\tau \end{pmatrix}\right)$$

By convention, random effects for the mean (b_i) can be assumed independent from those for the variance (τ_i) by setting $\Sigma_{\tau b} = 0$.

C. Location-scale mixed model distinguishing within and between visits variabilities

In some studies, multiple measurements of the same marker are collected during each visit. For instance, in clinical research, two or three blood pressure readings are typically recorded per visit, and within-visit variability may carry information.

To account for this, we introduce an LSMM that distinguishes within- and between-visit variabilities.

We introduce an additional level in the data hierarchy, grouping repeated measures by visit.

For each subject i ($i = 1, ..., N$), Y_{ijl} represents the l-th measurement ($l = 1, ..., n_{ij}$) at visit j ($j = 1, ..., n_i$) and time t_{ij}. We then define

$$\begin{cases} Y_{ijl} = \widetilde{Y}_i(t_{ij}) + \epsilon_{ij} + \nu_{ijl} \\ \quad\quad = X_{ij}^\top \beta + Z_{ij}^\top b_i + \epsilon_{ij} + \nu_{ijl}, \\ \epsilon_{ij} \sim \mathcal{N}(0, \sigma_i^2), \quad \text{with } \log(\sigma_i) = \mu_\sigma + \tau_{\sigma i}, \\ \nu_{ijl} \sim \mathcal{N}(0, \kappa_i^2), \quad \text{with } \log(\kappa_i) = \mu_\kappa + \tau_{\kappa i}, \end{cases}$$

where μ_σ and μ_κ are fixed intercepts for the between-visits and within-visit variances, respectively. The subject-specific random-effect b_i and $\tau_i = (\tau_{\sigma i}, \tau_{\kappa i})^\top$ are assumed to be Gaussian as in model (B).

Value

An object of class `lsmm` containing:

`table.res` Table of parameter estimates and standard errors.

`result_step1` A `marqLevAlg` object with first-step estimation results.

`result_step2` A `marqLevAlg` object with second-step estimation results.

`info_conv_step1` Information on first-step convergence (criteria and computation time).

`info_conv_step2` Information on second-step convergence (criteria and computation time).

`control` List of control parameters used during estimation.

Examples

```
data <- data.frame(
         ID = rep(1:5, each = 3),
         time = rep(1:3, 5),
         y = rnorm(15),
         event = rep(rbinom(5,1,0.5), each = 3),
         time_event = rep(runif(5), each = 3)
         )


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
   formGroup = ~ ID,
   formVar = "standard",
   timeVar = "time",
   data.long = data,
   S1 = 5,
   S2 = 5,
   nproc = 1
   )

 summary(m0)




data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10
threeC <- dplyr::group_by(threeC, ID, num.visit)
threeC <- dplyr::mutate(threeC, SBPvisit = mean(SBP))
threeC_ex1 <- threeC[!duplicated(threeC[, c("ID", "num.visit")]),
                                  c("ID", "SBPvisit", "age.visit65", "sex")]

#First example : a standard linear mixed model (constant residual variance,
# in time and between subjects, case A in details)

m1 <- lsmm(formFixed = SBPvisit ~ age.visit65+I(age.visit65^2),
               formRandom = ~ age.visit65+I(age.visit65^2),
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex1,
               formVar = "standard",
               S1 = 500,
               S2 = 1000,
               nproc = 1)
```

```
summary(m1)


#Second example : a linear mixed model with subject-specific time-dependent
# and covariate-dependent variability (case B in details)

#We adjust the individual residual variability on age and the sex.

m2 <- lsmm(formFixed = SBPvisit ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex1,
               formVar = "cov-dependent",
               formFixedVar = ~ age.visit65+sex,
               formRandomVar = ~ age.visit65,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)

summary(m2)

#Third example : a linear mixed model with subject-specific inter-visits and
# intra-visits variabilities

threeC_ex2 <- threeC[, c("ID", "SBP", "age.visit65", "sex", "num.visit")]

m3 <- lsmm(formFixed = SBP ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex2,
               formVar = "inter-intra",
               random_inter = TRUE,
               random_intra = TRUE,
               formGroupVisit = ~num.visit,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)

summary(m3)



#Fourth example : a linear mixed model with subject-specific inter-visits
# variability and constant intra-visit variability

m4 <- lsmm(formFixed = SBP ~ age.visit65+sex,
               formRandom = ~ age.visit65,
```

```
                formGroup = ~ ID,
                timeVar = 'age.visit65',
                data.long = threeC_ex2,
                formVar = "inter-intra",
                random_inter = TRUE,
                random_intra = FALSE,
                formGroupVisit = ~num.visit,
                correlated_re = FALSE,
                S1 = 500,
                S2 = 1000,
                nproc = 1)

summary(m4)
```

---

plot.lsjm *plot: Plot methods for* lsjm *or* lsmm *objects*

Description

This function produces different plots (longitudinal and survival goodness-of-fit, individual trajectory) of a fitted object of class lsmm or lsjm.

Usage

```
## S3 method for class 'lsjm_classicCR'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_classicIDM'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
```

```
  ObjectSmoothHazard = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_classicSingle'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_covDepCR'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_covDepIDM'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  ObjectSmoothHazard = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_covDepSingle'
plot(
  x,
  which = "long.fit",
  Objectpredict,
```

```
  break.times = NULL,
  ID.ind = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_interintraCR'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_interintraIDM'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  ObjectSmoothHazard = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsjm_interintraSingle'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'lsmm_classic'
plot(
  x,
  which = "long.fit",
```

```
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  ylim = NULL,
  xlim = NULL,
  ...
)

## S3 method for class 'lsmm_covDep'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  ylim = NULL,
  xlim = NULL,
  ...
)

## S3 method for class 'lsmm_interintra'
plot(
  x,
  which = "long.fit",
  Objectpredict,
  break.times = NULL,
  ID.ind = NULL,
  ylim = NULL,
  xlim = NULL,
  ...
)
```

Arguments

x	A lsmm or a lsjm object
which	A character indicating which plot must be display,
Objectpredict	An object of the predict function
break.times	A vector of breaking times to create windows if which = 'long.fit'
ID.ind	A vector providing the id of subjects for whom we we wish to trace the individual trajectory.
xlim	the x limits (x1, x2) of the plot. The default value, NULL, indicates that the range of the finite values to be plotted should be used.
ylim	the y limits (y1, y2) of the plot. The default value, NULL, indicates that the range of the finite values to be plotted should be used.
...	Further arguments passed to graphics::plot() or other methods.
ObjectSmoothHazard	A SmoothHazard object (only for IDM survival model)

Details

With `which="long.fit"`, this function allows to assess the fit of the longitudinal submodel comparing the mean of marker predictions collected in some windows of times (defined by `break.times` or using percentiles) to the mean of the observed measurements and its 95\

With `which="traj.ind"`, represents the individual trajectory with its prediction interval.

For a `lsjm` object only, with `which="survival.fit"`, the function allows to assess the fit of the survival submodel. For each transition, the predicted cumulative hazard function at each event time is computed given the predicted random effects. Then the mean of the predicted cumulative hazard functions are compared with there Nelson-Aalen estimator in the case of a single event or with competing risks. For an illness-death model, the predicted cumulative hazard function for each transition is compared to an illness-death model estimated by penalized likelihood accounting for interval censoring with the `SmoothHazard` package

Value

A list containing one or more plots:

`long.fit` A plot comparing the mean marker predictions with the mean observed measurements and their 95 confidence intervals.

`graph.traj.ind` One or more plots representing individual trajectories together with their prediction intervals.

`survB` A plot comparing the predicted cumulative hazard function with either the Nelson–Aalen estimator or an Illness–Death model.

Examples

```
set.seed(123)

data <- data.frame(
         ID = rep(1:5, each = 3),
         time = rep(1:3, 5),
         y = rnorm(15),
         event = rep(rbinom(5,1,0.5), each = 3),
         time_event = rep(runif(5), each = 3)
         )


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
   formGroup = ~ ID,
   formVar = "standard",
   timeVar = "time",
   data.long = data,
   S1 = 5,
   S2 = 5,
   nproc = 1
   )
```

```
pred.m0 <- predict(m0, which = c("RE","Y"), data.long = data)

plot(m0,
     which = "long.fit",
     Objectpredict = pred.m0,
     break.times = c(0,1,2,3))

plot(m0,
     which = "traj.ind",
     Objectpredict = pred.m0,
     ID.ind = c(1,2))




## Not run:

library(dplyr)
data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10
threeC <- threeC %>% group_by(ID, num.visit) %>% mutate(SBPvisit = mean(SBP))
threeC$age65_CR <- NA
threeC$age65_CR[which(threeC$dem == 1)] <- (threeC$age.last65[which(threeC$dem == 1)]
                                          +threeC$age.first65[which(threeC$dem == 1)])/2
threeC$age65_CR[which(threeC$dem == 0)] <- threeC$age.final65[which(threeC$dem == 0)]
threeC$demCR <- threeC$dem
threeC$deathCR <- NA
threeC$deathCR[which(threeC$dem == 1)] <- 0
threeC$deathCR[which(threeC$dem == 0)]<-threeC$death[which(threeC$dem == 0)]
threeC <- threeC %>% group_by(ID) %>% filter(age.visit65 <= age65_CR)


threeC_ex1 <- threeC[!duplicated(threeC[, c("ID", "num.visit")]), c("ID",
     "SBPvisit", "age.visit65", "sex","age0_65", "demCR","deathCR",
      "age65_CR")]


# Estimation of a lsmm with time-dependent variability model:
m2 <- lsmm(formFixed = SBPvisit ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex1,
               formVar = "cov-dependent",
               formFixedVar = ~ age.visit65+sex,
               formRandomVar = ~ age.visit65,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)
```

```
# Predictions:
pred.m2 <- predict(m2, which = c("RE", "Y"))

#Plot:
plot(m2, which = "long.fit", Objectpredict = pred.m2,
                           break.times = (seq(65,95,by = 2.5)-65)/10)
plot(m2, which = "traj.ind", Objectpredict = pred.m2,
                                            ID.ind = c(3,120))

# Estimation of a lsjm with time-dependent variability model and
# competing events:
l2 <- lsjm(Objectlsmm = m2,
           survival_type = 'CR',
           formSurv_01 = ~ sex,
           formSurv_02 = ~ sex,
           sharedtype_01 = c("value", "variability"),
           sharedtype_02 = c("value", "variability"),
           hazardBase_01 = "Weibull",
           hazardBase_02 = "Weibull",
           delta1 = ~ demCR,
           delta2 = ~ deathCR,
           Time_T0 = ~ age0_65,
           Time_T = ~ age65_CR,
           nproc = 5,
           S1 = 1000,
           S2 = 2000)

# Predictions:
pred.l2 <- predict(l2, which = c("RE", "Y", "Cum"))

# Plot:
plot(l2, which = "long.fit", Objectpredict = pred.l2,
                         break.times = (seq(65,95,by = 2.5)-65)/10)
plot(l2, which = "traj.ind", Objectpredict = pred.l2,
                          ID.ind = c(3,120))
plot(l2, which = "survival.fit", Objectpredict = pred.l2)

## End(Not run)
```

`predict.lsjm` *predict: Prediction of some quantities for each subjects*

Description

This function computes different predicted quantities. For a `lsmm` object, it is possible to compute, for each subject, their random effects, their value of the marker and the residual variability (for each measurement time)). For a `lsjm` object we could also compute cumulative hazard function for each risk transition.

Usage

```
## S3 method for class 'lsjm_classicCR'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_classicIDM'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_classicSingle'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_covDepCR'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_covDepIDM'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_covDepSingle'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_interintraCR'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_interintraIDM'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsjm_interintraSingle'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsmm_classic'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsmm_covDep'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)

## S3 method for class 'lsmm_interintra'
predict(object, which = "RE", Objectranef = NULL, data.long = NULL, ...)
```

Arguments

`object`	Either a `lsmm` object or a `lsjm` object
`which`	A vector of characters to indicate which predictions are computed. `"RE"` corresponds to the random effects, `"Y"` to the marker and `"Cum"` to the cumulative risk function(s) (only in the case of a `lsjm` object)
`Objectranef`	Optional `ranef` object containing the predicted random effects for each individual. The Default is NULL and the predict functions should compute the random effects.
`data.long`	A dataframe containing the longitudinal data for making predictions.

... Further arguments

Value

A table for each type of prediction (RE/Y/Cum)

predictRE A table with predicted random effects for each subjects.

predictY A table with predicted marker and residual variability for each subjects and at each measurement time.

predictCum_01 A table for predicted cumulative hazard function for transition 0-1.

predictCum_02 A table for predicted cumulative hazard function for transition 0-2.

predictCum_12 A table for predicted cumulative hazard function for transition 1-2.

grid.time.Cum A vector of times for which the cumulative hazard functions are computed.

Examples

```
data <- data.frame(
         ID = rep(1:5, each = 3),
         time = rep(1:3, 5),
         y = rnorm(15),
         event = rep(rbinom(5,1,0.5), each = 3),
         time_event = rep(runif(5), each = 3)
         )


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
   formGroup = ~ ID,
   formVar = "standard",
   timeVar = "time",
   data.long = data,
   S1 = 5,
   S2 = 5,
   nproc = 1
   )

pred.m0 <- predict(m0, which = c("RE","Y"), data.long = data)


## Not run:
library(dplyr)

data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10
threeC <- threeC
threeC <- dplyr::group_by(threeC, ID, num.visit)
threeC <- dplyr::mutate(threeC, SBPvisit = mean(SBP))
threeC_ex1 <- threeC[!duplicated(threeC[, c("ID", "num.visit")]),
                                  c("ID", "SBPvisit", "age.visit65", "sex","dem","death",
```

```
                                    "age.first65","age.last65","age.final65","age0_65")]

m1 <- lsmm(formFixed = SBPvisit ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex1,
               formVar = "cov-dependent",
               formFixedVar = ~ age.visit65+sex,
               formRandomVar = ~ age.visit65,
               correlated_re = FALSE,
               S1 = 500,
               S2 = 1000,
               nproc = 1)

l1 <- lsjm(m1,
           survival_type = 'IDM',
           formSurv_01=~1,
           formSurv_02=~sex,
           formSurv_12=~sex,
           sharedtype_01 = c("value", "variability"),
           sharedtype_02 = c("value", "slope", "variability"),
           sharedtype_12 = c("value", "variability"),
           hazardBase_01 = "Weibull",
           hazardBase_02 = "Weibull",
           hazardBase_12 = "Splines",
           delta1=~dem,
           delta2=~death,
           Time_T =~age.final65,
           Time_L =~age.last65,
           Time_R =~age.first65,
           Time_T0 =~age0_65,
           formSlopeFixed =~1,
           formSlopeRandom =~1,
           index_beta_slope = c(2),
           index_b_slope = c(2),
           nb.knots.splines = c(0,0,1),
           S1 = 1000,
           S2 = 2000,
           nproc = 5)

pred.m1 <- predict(m1, which = c("RE","Y"), data.long = threeC_ex1)

pred.l1 <- predict(l1, which = c("RE","Y","Cum"), data.long = threeC_ex1)



## End(Not run)
```

ranef *ranef: the predicted random effects for all subjects*

Description

ranef: the predicted random effects for all subjects

Usage

```
## S3 method for class 'lsjm_classicCR'
ranef(object, ...)

## S3 method for class 'lsjm_classicIDM'
ranef(object, ...)

## S3 method for class 'lsjm_classicSingle'
ranef(object, ...)

## S3 method for class 'lsjm_covDepCR'
ranef(object, ...)

## S3 method for class 'lsjm_covDepIDM'
ranef(object, ...)

## S3 method for class 'lsjm_covDepSingle'
ranef(object, ...)

## S3 method for class 'lsjm_interintraCR'
ranef(object, ...)

## S3 method for class 'lsjm_interintraIDM'
ranef(object, ...)

## S3 method for class 'lsjm_interintraSingle'
ranef(object, ...)

## S3 method for class 'lsmm_classic'
ranef(object, ...)

## S3 method for class 'lsmm_covDep'
ranef(object, ...)

## S3 method for class 'lsmm_interintra'
ranef(object, ...)
```

Arguments

object	Either a lsmm object or a lsjm object

... Further arguments

Value

A table with predicted random effects for each subjects.

Examples

```
data <- data.frame(
         ID = rep(1:5, each = 3),
         time = rep(1:3, 5),
         y = rnorm(15),
         event = rep(rbinom(5,1,0.5), each = 3),
         time_event = rep(runif(5), each = 3)
         )


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
   formGroup = ~ ID,
   formVar = "standard",
   timeVar = "time",
   data.long = data,
   S1 = 5,
   S2 = 5,
   nproc = 1
   )

ranef.m0 <- ranef(m0)

## Not run:

library(dplyr)

data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10
threeC <- threeC
threeC <- dplyr::group_by(threeC, ID, num.visit)
threeC <- dplyr::mutate(threeC, SBPvisit = mean(SBP))
threeC_ex1 <- threeC[!duplicated(threeC[, c("ID", "num.visit")]),
                                c("ID", "SBPvisit", "age.visit65", "sex","dem","death",
                                  "age.first65", "age.last65","age.final65","age0_65")]

m1 <- lsmm(formFixed = SBPvisit ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
               data.long = threeC_ex1,
               formVar = "cov-dependent",
               formFixedVar = ~ age.visit65+sex,
               formRandomVar = ~ age.visit65,
```

```
                correlated_re = FALSE,
                S1 = 500,
                S2 = 1000,
                nproc = 1)

l1 <- lsjm(m1,
           survival_type = 'IDM',
           formSurv_01=~1,
           formSurv_02=~sex,
           formSurv_12=~sex,
           sharedtype_01 = c("value", "variability"),
           sharedtype_02 = c("value", "slope", "variability"),
           sharedtype_12 = c("value", "variability"),
           hazardBase_01 = "Weibull",
           hazardBase_02 = "Weibull",
           hazardBase_12 = "Splines",
           delta1=~dem,
           delta2=~death,
           Time_T =~age.final65,
           Time_L =~age.last65,
           Time_R =~age.first65,
           Time_T0 =~age0_65,
           formSlopeFixed =~1,
           formSlopeRandom =~1,
           index_beta_slope = c(2),
           index_b_slope = c(2),
           nb.knots.splines = c(0,0,1),
           S1 = 1000,
           S2 = 2000,
           nproc = 5)

r1 <- ranef(m1)
r1.1 <- ranef(l1)


## End(Not run)
```

`survmarg` *The population-average probability of being free of any events at a given time t and covariates*

Description

The population-average probability of being free of any events at a given time t and covariates

Usage

```
## S3 method for class 'lsjm_classicCR'
survmarg(object, individual, time)
```

```
## S3 method for class 'lsjm_classicIDM'
survmarg(object, individual, time)

## S3 method for class 'lsjm_classicSingle'
survmarg(object, individual, time)

## S3 method for class 'lsjm_covDepCR'
survmarg(object, individual, time)

## S3 method for class 'lsjm_covDepIDM'
survmarg(object, individual, time)

## S3 method for class 'lsjm_covDepSingle'
survmarg(object, individual, time)

## S3 method for class 'lsjm_interintraCR'
survmarg(object, individual, time)

## S3 method for class 'lsjm_interintraIDM'
survmarg(object, individual, time)

## S3 method for class 'lsjm_interintraSingle'
survmarg(object, individual, time)
```

Arguments

object	A lsjm object
individual	Subject profile for which we want to compute the probability
time	a numeric, the time t at which we want to compute the probability

Value

A numeric, the population-average probability of being free of any events at a given time t and chosen covariates

Examples

```
set.seed(456)

data <- data.frame(
  ID = rep(1:100, each = 3),
  time = rep(0:2, 100),
  y = rnorm(300),
  event = rep(rbinom(100, 1, 0.5), each = 3),
  time_event = rep(runif(100, min = 0, max = 5), each = 3),
  covariate = rep(rbinom(100, size = 1, prob = 0.5), each = 3)
)

# Keep only the observed time before time_event
```

```
data <- subset(data, time < time_event)


m0 <- lsmm(
   formFixed = y ~ time,
   formRandom = ~ time,
   formGroup = ~ ID,
   formVar = "standard",
   timeVar = "time",
   data.long = data,
   S1 = 5,
   S2 = 5,
   nproc = 1
   )

 fit <- lsjm(
      Objectlsmm = m0,
      survival_type = "Single",
      formSurv_01 = ~ covariate,
      sharedtype_01 = "value",
      hazardBase_01 = "Weibull",
      delta1 = ~ event,
      Time_T = ~ time_event,
      S1 = 100,
      S2 = 100,
      nproc = 1,
      binit = c(1.44498682, -4.10877891, 0.23741874, -10.11422733, 0.06706379,
                -0.05305361, 0.94307564, 0.34785575, -0.08949800, 0.00814575)
      )

individual <- data[1, ]
survmarg(fit, individual, time = 2)


## Not run:

# Begining by estimating the examples from \code{lsmm} and \code{lsjm}, we then compute
# the average-probability of being alive and without dementia at age 85 for a Women

data(threeC)
threeC$age.visit65 <- (threeC$age.visit-65)/10
threeC$SBP <- threeC$SBP/10
threeC <- threeC
threeC <- dplyr::group_by(threeC, ID, num.visit)
threeC <- dplyr::mutate(threeC, SBPvisit = mean(SBP))
threeC_ex1 <- threeC[!duplicated(threeC[, c("ID", "num.visit")]),
                                  c("ID", "SBPvisit", "age.visit65", "sex","dem","death",
                                     "age.first65","age.last65","age.final65","age0_65")]

m1 <- lsmm(formFixed = SBPvisit ~ age.visit65,
               formRandom = ~ age.visit65,
               formGroup = ~ ID,
               timeVar = 'age.visit65',
```

```
                data.long = threeC_ex1,
                formVar = "cov-dependent",
                formFixedVar = ~ age.visit65+sex,
                formRandomVar = ~ age.visit65,
                correlated_re = FALSE,
                S1 = 500,
                S2 = 1000,
                nproc = 1)

l1 <- lsjm(m1,
           survival_type = 'IDM',
           formSurv_01=~1,
           formSurv_02=~sex,
           formSurv_12=~sex,
           sharedtype_01 = c("value", "variability"),
           sharedtype_02 = c("value", "slope", "variability"),
           sharedtype_12 = c("value", "variability"),
           hazardBase_01 = "Weibull",
           hazardBase_02 = "Weibull",
           hazardBase_12 = "Splines",
           delta1=~dem,
           delta2=~death,
           Time_T =~age.final65,
           Time_L =~age.last65,
           Time_R =~age.first65,
           Time_T0 =~age0_65,
           formSlopeFixed =~1,
           formSlopeRandom =~1,
           index_beta_slope = c(2),
           index_b_slope = c(2),
           nb.knots.splines = c(0,0,1),
           S1 = 1000,
           S2 = 2000,
           nproc = 5)

individual <- threeC_ex1[1,]
survmarg(l1, individual, time = 2)



## End(Not run)
```

---

threeC *ThreeC dataset*

Description

A random subsample of 500 subjects from the French Three-Cities cohort, aimed at assessing the relation between vascular factors and dementia in the elderly, for the example of the LSJM package.

Usage

```
data(threeC)
```

Format

A data frame with 5248 rows and 13 variables:

ID id of each subject

Apoe4 indicator of the apoe4 gene

Edu Educational level (0 = less than 10 years, 1 = more than 10 years)

SBP systolic blood pressure in mmHg

age.visit age at measurement of SBP

age.final age at death or censoring (last news)

age0 age at entry in the cohort

age.last age of last visit without dementia

age.first age at visit diagnosis for dementia

sex sex of the subject

dem indicator of dementia

death indicator of death

num.visit visit identifier

age.visit65 (age.visit-65)/10

age.final65 (age.final-65)/10

age.last65 (age.last-65)/10

age.first65 (age.first-65)/10

age0_65 (age0-65)/10

Details

The sample includes participants without dementia at baseline and aged 65 years old or older. Repeated measures of systolic blood pressure were collected over a maximum period of 20 years. At each visit, systolic blood pressure was measured two or three times.

Index

∗ **datasets**
threeC, 36

data.manag.long, 2
data.manag.sigma, 3
data.manag.surv, 3
dynpred, 4

graphics::plot(), *24*

initial.long, 8

lsjm, 8
lsmm, 16

plot.lsjm, 21
plot.lsjm_classicCR *(*plot.lsjm*)*, 21
plot.lsjm_classicIDM *(*plot.lsjm*)*, 21
plot.lsjm_classicSingle *(*plot.lsjm*)*, 21
plot.lsjm_covDepCR *(*plot.lsjm*)*, 21
plot.lsjm_covDepIDM *(*plot.lsjm*)*, 21
plot.lsjm_covDepSingle *(*plot.lsjm*)*, 21
plot.lsjm_interintraCR *(*plot.lsjm*)*, 21
plot.lsjm_interintraIDM *(*plot.lsjm*)*, 21
plot.lsjm_interintraSingle *(*plot.lsjm*)*, 21
plot.lsmm_classic *(*plot.lsjm*)*, 21
plot.lsmm_covDep *(*plot.lsjm*)*, 21
plot.lsmm_interintra *(*plot.lsjm*)*, 21
predict.lsjm, 27
predict.lsjm_classicCR *(*predict.lsjm*)*, 27
predict.lsjm_classicIDM *(*predict.lsjm*)*, 27
predict.lsjm_classicSingle *(*predict.lsjm*)*, 27
predict.lsjm_covDepCR *(*predict.lsjm*)*, 27
predict.lsjm_covDepIDM *(*predict.lsjm*)*, 27
predict.lsjm_covDepSingle *(*predict.lsjm*)*, 27
predict.lsjm_interintraCR *(*predict.lsjm*)*, 27
predict.lsjm_interintraIDM *(*predict.lsjm*)*, 27
predict.lsjm_interintraSingle *(*predict.lsjm*)*, 27
predict.lsmm_classic *(*predict.lsjm*)*, 27
predict.lsmm_covDep *(*predict.lsjm*)*, 27
predict.lsmm_interintra *(*predict.lsjm*)*, 27

ranef, 31

survmarg, 33

threeC, 36