Package ‘INLAjoint’

June 2, 2026

Type Package

Title Multivariate Joint Modeling for Longitudinal and Time-to-Event Outcomes with 'INLA'

Version 26.6.2

Description Estimation of joint models for multivariate longitudinal markers (with various distributions available) and survival outcomes (possibly accounting for competing risks) with Integrated Nested Laplace Approximations (INLA). The flexible and user friendly function joint() facilitates the use of the fast and reliable inference technique implemented in the 'INLA' package for joint modeling. More details are given in the help page of the joint() function (accessible via ?joint in the R console) and the vignette associated to the joint() function (accessible via vignette(``INLAjoint'') in the R console).

License GPL-3

URL `https://github.com/DenisRustand/INLAjoint`

BugReports `https://github.com/DenisRustand/INLAjoint/issues`

Encoding UTF-8

LazyData true

Depends R (>= 3.6), utils, stats

Imports numDeriv, reformulas, ggplot2, methods, Matrix, nlme, data.table

Additional_repositories `https://inla.r-inla-download.org/R/testing`

Suggests INLA, knitr, rmarkdown, splines, graphics, grDevices, R.rsp, survival, JMbayes2, rstanarm, frailtypack, smcure, fmesher, MASS, sn

BuildVignettes true

VignetteBuilder R.rsp

Maintainer Denis Rustand <`INLAjoint@gmail.com`>

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Denis Rustand [cre, aut] (ORCID:
<https://orcid.org/0000-0001-9708-5220>),
Elias Teixeira Krainski [aut] (ORCID:
<https://orcid.org/0000-0002-7063-2615>),
Haavard Rue [aut] (ORCID: <https://orcid.org/0000-0002-0222-1881>),
Janet van Niekerk [aut] (ORCID:
<https://orcid.org/0000-0002-4334-2057>)

Repository CRAN

Date/Publication 2026-06-02 18:20:07 UTC

Contents

`coef.INLAjoint` *Extracts model coefficients from a given model fitted with INLAjoint*

Description

This function extracts model coefficients from INLAjoint objects.

Usage

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

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`family.INLAjoint` *Extracts family from a given model fitted with INLAjoint*

Description

This function extracts family from INLAjoint objects.

Usage

```
## S3 method for class 'INLAjoint'
family(object, ...)
```

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`fitted.INLAjoint` *Extracts fitted values from a given model fitted with INLAjoint*

Description

This function extracts fitted values from INLAjoint objects. Values are associated to a name to keep track of the outcome related to each fitted value.

Usage

```
## S3 method for class 'INLAjoint'
fitted(object, ...)
```

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`fixef.INLAjoint` *Extracts fixed effects values from a given model fitted with INLAjoint*

Description

This function extracts fixed effects values from INLAjoint objects.

Usage

```
## S3 method for class 'INLAjoint'
fixef(object, ...)
```

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`formula.INLAjoint` *Extracts formula from a given model fitted with INLAjoint*

Description

This function extracts formula from INLAjoint objects.

Usage

```
## S3 method for class 'INLAjoint'
formula(x, ...)
```

Arguments

`x`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`INLAjoint` *INLAjoint*

Description

INLAjoint is a package that fits joint models for multivariate longitudinal markers (with various distributions available) and survival outcomes (possibly accounting for competing risks) with Integrated Nested Laplace Approximations (INLA). The flexible and user friendly function joint() facilitates the use of the fast and reliable inference technique implemented in INLA package for joint modeling. More details are given in the help page of the joint function (accessible via ?joint in the R console), the vignette associated to the joint() function (accessible via vignette("INLAjoint") in the R console).

Contact: `<INLAjoint@gmail.com>`

Usage

```
INLAjoint()
```

`INLAjoint.ginv` *Setup ginv*

Description

Setup random walk input:

Usage

```
INLAjoint.ginv(x, tol = sqrt(.Machine$double.eps), rankdef = NULL)
```

Arguments

`x`	input
`tol`	tolerance
`rankdef`	rank output:

Value

ginv

`INLAjoint.object` *Fitted* `joint` *object*

Description

An object of class `INLAjoint` returned by the `joint` function that fits a joint model to multivariate longitudinal and time-to-event data. The following functions can apply to objects of this class: `plot`, `print`, `summary` and `priors.used`.

Usage

```
INLAjoint.object
```

Value

A list with the following components:

`names.fixed` a vector with the name of the fixed effects of the model. The corresponding submodel is indicated by the suffix including a letter and a number ("L" for longitudinal and "S" for survival).

`summary.fixed` summary statistics for the fixed effects of the model. The summary statistics sorted by longitudinal and survival components are available by applying the `summary` function to the `INLAjoint` object.

`summary.fixed` marginals for the fixed effects of the model.

`mlik` log marginal-likelihood.

`cpo` Conditional Predictive Ordinate.

`gcpo` Group-Conditional Predictive Ordinate.

`po` Predictive ordinate.

`waic` Widely applicable Bayesian information criterion

`model.random` a vector with the name of the random parameters of the model, possibly including the following components:

  `RW1 model` and `RW2 model` Random walk of order 1 or 2 corresponding to Bayesian smoothing splines for the baseline hazard risk

  `IID model` Univariate random effect.

  `IIDKD model` Multivariate random effects.

  `Copy` association parameter.

`summary.random` summary statistics for the random parameters of the model.

`marginals.random` marginals for the random parameters of the model.

`size.random` size of the random parameters of the model.

`summary.linear.predictor` summary statistics of the linear predictors.

`marginals.linear.predictor` marginals for the linear predictors.

`summary.fitted.values` summary statistics of the fitted values.

`marginals.fitted.values` marginals for the fitted values.

`size.linear.predictor` size of the linear predictors.

`summary.hyperpar` summary statistics for the hyperparameters of the model. The summary statistics sorted by longitudinal and survival components are available by applying the `summary` function to the `INLAjoint` object. Particularly, this is the raw output of INLA and therefore the precision of the residual errors and baseline hazard functions hyperparameters are provided. Similarly, the Cholesky matrix is given for the random-effects. The summary function can easily return either variance and covariance or standard deviations and correlations for all these hyperparameters.

`marginals.hyperpar` marginals for the hyperparameters of the model.

`internal.summary.hyperpar` summary of the internal hyperparameters, this is similar to the summary of the hyperparameters but here they are provided as used for the computations (logarithm scale for residual error and baseline risk hyperparameters).

`internal.marginals.hyperpar` marginals for the internal hyperparameters of the model.

`misc` miscellaneous (as provided in the INLA output).

`dic` Deviance Information Criterion.

`mode` .

`joint.hyper` .

`nhyper` .

`version` Version of INLA.

`cpu.used` Computation time of INLA.

`all.hyper` .

`.args` .

`call` INLA call.

`selection` information about parameters for sampling with inla.rjmarginal.

`cureVar` informations about cure fraction submodel for mixture cure survival models.

`variant` information about variant for Weibull baseline hazards.

`SurvInfo` some information about survival submodels (names of event indicator and event time variables as well as baseline hazard).

`famLongi` list of distributions for the longitudinal markers.

`corLong` boolean indicating if random effects are correlated accross markers.

`control.link` informations about link function (1=default).

`longOutcome` name of longitudinal outcomes.

`survOutcome` name of survival outcomes.

`assoc` vector with names of all association parameters (longi-surv).

`id` name of the id variable.

`timeVar` name of time variable.

`range` information about range of X-axis values for non-linear associations.

`REstruc` names of the grouped random effects for the longitudinal markers.

`mat_k` contains the list of random effects covariance matrices when they are fixed as they are not part of the estimated parameters (used for displaying them in summary).

`fixRE` list of the size of number of groups of random effects, each element is a boolean indicating if the random effects of the group is fixed (TRUE) or estimated (FALSE).

`lonFacChar` list of factors and character covariates included in the longitudinal submodels to keep track of modalities (used internally when doing predictions to reconstruct categorical covariates).

`survFacChar` same as lonFacChar but for survival submodels.

`corRE` list indicating if groups of random effects are correlated within longitudinal submodels.

`basRisk` list of the baseline risk used for each survival submodel.

`priors_used` informations about priors used in the model, internally used to display priors in plots (with argument priors=TRUE in the call of the plot function). Note that priors can also be displayed with the function priors.used() applied to an INLAjoint object.

`dataLong` name of the longitudinal dataset.

`dataSurv` name of the survival dataset.

See Also

`joint`.

`INLAjoint.rw` *Setup rw*

Description

Setup random walk input:

Usage

```
INLAjoint.rw(n, order = 1L, sparse = TRUE, scale.model = FALSE, cyclic = FALSE)
```

Arguments

`n`	number of knots
`order`	order 1
`sparse`	boolean, sparsity
`scale.model`	boolean, scale
`cyclic`	boolean, cyclic output:

Value

random walk 1

`INLAjoint.rw2` *Setup rw2*

Description

Setup random walk of order 2 input:

Usage

```
INLAjoint.rw2(n, order = 2L, ...)
```

Arguments

`n`	number of knots
`order`	order 2
`...`	additional arguments passed to INLAjoint.rw output:

Value

random walk 2

`INLAjoint.scopy.define` *Setup scopy*

Description

Setup weights for non-linear effects input:

Usage

```
INLAjoint.scopy.define(n = 5L)
```

Arguments

`n`	number of knots output:

Value

Matrix with weights for scopy

`joint` *Fit a multivariate joint model for longitudinal and/or survival data*

Description

This function fits a multivariate joint model for longitudinal and/or survival data. The longitudinal outcomes are modeled with mixed effects models and can handle various distributions. The survival outcomes (i.e., terminal event with possibly competing risks) are modeled with Cox proportional hazards regression models. Various association structures can be specified between the longitudinal and survival outcomes. The inference is based on Integrated Nested Laplace Approximations (Rue et al., 2009).

Usage

```
joint(
  formSurv = NULL,
  formLong = NULL,
  dataSurv = NULL,
  dataLong = NULL,
  id = NULL,
  timeVar = NULL,
  family = "gaussian",
  link = "default",
  basRisk = "rw1",
  NbasRisk = 15,
```

```
  cutpoints = NULL,
  assoc = NULL,
  assocSurv = NULL,
  corLong = FALSE,
  corRE = TRUE,
  dataOnly = FALSE,
  longOnly = FALSE,
  silentMode = FALSE,
  reorder = TRUE,
  run = TRUE,
  control = list()
)
```

Arguments

`formSurv`	the formula for the time-to-event outcome, with the response given as an inla.surv() object. Keep it as NULL if no survival part is needed and give a list of formulas for competing risks.
`formLong`	the formula for the longitudinal outcome, structured as with the lme4 package (Mächler et al., 2015) for linear mixed-effects models (i.e., including random effects within parenthesis). Keep it as NULL if no longitudinal part is needed and give a list of formulas for multivariate longitudinal.
`dataSurv`	the dataset for the survival part. Keep it as NULL if no survival part is needed or if the survival data is in the longitudinal dataset (it will extract the last line for each individual as the survival dataset).
`dataLong`	the dataset for the longitudinal part. Keep it as NULL if no longitudinal part is needed. For multivariate longitudinal models, either give one dataset with all outcomes and covariates or a list of datasets for each longitudinal marker.
`id`	the name of the variable to identify individuals or grouped repeated measurements. Keep is as NULL if no longitudinal part is needed.
`timeVar`	a character string (or a vector) giving the name of the time-varying variable(s). Functions of time can be included in formulas, they first need to be set up as a univariate function with name fX, where X is a number between 1 and 20. Then the function can be used directly in the formula (see example below).
`family`	a character string (or a vector) giving the name of families for the longitudinal outcomes. The list of the available families is given by names(inla.models()$likelihood).
`link`	a character string (or a vector) giving the link function associated to the families for the longitudinal outcomes. The links available for a family is given in the associated doc: inla.doc("familyName"). The link should be a vector of the same size as the family parameter and should be set to "default" for default (i.e., identity for gaussian, log for poisson, logit for bimomial,...).
`basRisk`	the baseline risk of event (should be a vector in case of competing risks). It can be defined as parametric with either "exponentialsurv" for exponential baseline or "weibullsurv" for Weibull baseline (note that there are two formulations of the Weibull distribution, see ‘inla.doc("weibull")‘ for more details, default is variant = 0). Alternatively, there are two options to avoid parametric assumptions on

	the shape of the baseline risk: "rw1" for random walks of order one prior that corresponds to a smooth spline function based on first order differences. The second option "rw2" assigns a random walk order two prior that corresponds to a smooth spline function based on second order differences. This second option provides a smoother spline compared to order one since the smoothing is then done on the second order.
`NbasRisk`	the number of intervals for the baseline risk function, only one value should be provided and the same number of intervals is used for each risk submodel in case of competing risks.
`cutpoints`	a vector with baseline hazard cutpoints if not using equidistant (if not NULL, this replaces the NbasRisk parameter).
`assoc`	a character string that specifies the association between the longitudinal and survival components. The available options are "CV" for sharing the current value of the linear predictor, "CS" for the current slope, "CV_CS" for the current value and the current slope, "SRE" for shared random effects (i.e., sharing the individual deviation from the mean at time t as defined by the random effects), "SRE_ind" for shared random effect independent (each random effect's individual deviation is associated to an association parameter in the survival submodel) and "" (empty string) for no association. When there are either multiple longitudinal markers or multiple competing events, this should be a vector. In case of both multiple markers and events, it should be a list with one element per longitudinal marker and each element is a vector of association for each competing event. Keep it as NULL to have no association between longitudinal and survival components or if there is no survival component.
`assocSurv`	a boolean that indicates if a frailty term (i.e., random effect) from a survival model should be shared into another survival model. The order is important, the first model in the list of survival formulas (‘formSurv‘) should include a random effect and it can be shared in the next formulas. Multiple survival models with random effects can be accommodated and a random effect can be shared in multiple survival models, following the same structure as ‘assoc‘ (i.e., vector of booleans if one random effect is shared in multiple survival and list of vectors if multiple survival models with random effects share their random effects in multiple survival models).
`corLong`	a boolean that only applies when multiple longitudinal markers are fitted: should the random effects accross markers be correlated (TRUE) or independent (FALSE)? Default is FALSE.
`corRE`	list of the size of number of groups of random effects (i.e., equal to 1 if there is only one longitudinal marker or if corLong is TRUE and equal to the number of markers otherwise), each element is a boolean indicating if the random effects of the group must be correlated or independent (i.e., diagonal variance-covariance). Default is FALSE.
`dataOnly`	a boolean to only prepare the data with the correct format without running the model.
`longOnly`	a boolean to only prepare the data for the longitudinal part of a longitudinal-survival joint model with the correct format without running the model.
`silentMode`	a boolean that will stop printing messages during computations if turned to TRUE.

`reorder`	a boolean used to prevent reordering of the data according to id.
`run`	a boolean used to setup a model without running INLA (allows to make modifications prior to run).

`control`	a list of control values that can be set with control=list(), with components:
	`priorFixed` list with mean and standard deviations for the Gaussian prior distribution for the fixed effects. Default is `list(mean=0, prec=0.01, mean.intercept=0, prec.intercept=0.01)`, where mean and prec are the mean and precision (i.e., inverse of the variance) of the fixed effects, respectively and mean.intercept and prec.intercept are the corresponding parameters for the fixed intercept.
	`priorAssoc` list with mean and standard deviations for the Gaussian prior distribution for the association parameters (does not apply to "SRE_ind" association and shared random effect from survival models (frailty), see next item for those two). Default is `list(mean=0, prec=0.01)`
	`priorSRE_ind` list with mean and standard deviations for the Gaussian prior distribution for the association of independent random effects ("SRE_ind" and survival frailty random effects shared). Default is `list(mean=0, prec=1)`
	`priorRandom` list with prior distribution for the multivariate random effects (Inverse Wishart). Default is `list(r=10, R=1). The default behavior automatically adapts the r to be equal to order + 1 if not manually specified, making the prior scaled to the dimension of the random effects.`, see "inla.doc("iidkd") for more details.
	`priorFrailty` list with prior distribution for the frailty iid random effects (Inverse Gamma). Default is `list(alpha=4, beta=1)`, see "inla.doc("iid123") for more details.
	`priorRW` Vector for the Penalised Complexity prior of the log precision of random walk baseline hazards. Default is a vector c(0.5, 0.01). See "inla.doc("RW1")" and "inla.doc("RW2") for more details.
	`initVC` list of the size of number of groups of random effects giving initial values for variance-covariance of random effects, first values are variance and then covariances (as displayed in summary). All the elements of the covariance matrix must be fixed but in case of multiple groups of random effects, it is possible to fix initial values for only some groups, then elements in the list that are not initialized must be an empty string.
	`initSD` same as initVC but to fix standard deviations and correlations instead (only one of these two arguments can be used).
	`strata` list of the same size as the number of survival submodels, giving the name of covariates for stratified proportional hazards model (default is NULL).
	`Kinship` matrix for kinship frailty structure in survival models, this must be the inverse of 2*kinship and only works when a frailty term is included in the model (default is NULL).
	`fixRE` list of the size of number of groups of random effects, each element is a boolean indicating if the random effects of the group must be fixed or estimated.
	`assocInit` Initial value for all the association parameters (default is 0.1).

`int.strategy` a character string giving the strategy for the numerical integration over the hyperparameters used to approximate the marginal posterior distributions of the latent field. Available options are "auto" (default), "ccd", "grid" or "eb" (empirical Bayes). The default strategy uses "grid" for dim<=2 and "ccd" otherwise. The empirical Bayes uses only the mode of the approximations for the integration, which speed up and simplifies computations. It can be pictured as a tradeoff between Bayesian and frequentist estimation strategies while the default full Bayesian accounts for uncertainty by using the mode and the curvature at the mode. See ?control.inla for additional details.

`Ntrials` Number of trials for binomial and Betabinomial distributions, default is NULL.

`cpo` TRUE/FALSE: Default is FALSE, set to TRUE to compute the Conditional Predictive Ordinate.

`cfg` TRUE/FALSE: Default is FALSE, set to TRUE to be able to sample from the posterior distribution.

`safemode` TRUE/FALSE: use the INLA safe mode (automatically reruns in case of negative eigenvalue(s) in the Hessian, reruns with adjusted starting values in case of crash). Default is TRUE (activated). The message “‘*** inla.core.safe“‘ appears when the safe mode is running, it improves the inference of the hyperparameters and can be ignored. To remove this safe mode, switch the boolean to FALSE (it can save some computation time but may return slightly less precise estimates for some hyperparameters).

`n_NL` Number of knots for non-linear associations (random walk of order 2).

`NLpriorAssoc` Priors for non-linear effects, it is a list of 3 elements "mean", "slope" and "spline", each element including 3 options: "mean", "prec" and "initial" for the mean, precision of the prior and initial values.

`rerun` TRUE/FALSE: the model reruns to improve numerical stability (default is FALSE).

`tolerance` accuracy in the inner optimization (default is 0.005).

`h` step-size for the hyperparameters (default is 0.005).

`verbose` TRUE/FALSE: prints details of the INLA algorithm. Default is FALSE.

`keep` TRUE/FALSE: keep internal files. Default is FALSE. (expert option)

`lightmode` Controls the amount of memory in the saved object. Default is 0 (save all), when set to 1 a lighter version is saved but keeps required information for plot() and predict() while when set to 2 the light version also removes .args and misc (for an even lighter object).

Value

An object of class `INLAjoint`. See `INLAjoint.object` for details.

References

Rustand, D., van Niekerk, J., Teixeira Krainski, E., Rue, H. and Proust-Lima, C. (2023). Fast and flexible inference for joint models of multivariate longitudinal and survival data using integrated nested Laplace approximations. Biostatistics, 2023, kxad019. https://doi.org/10.1093/biostatistics/kxad019 https://arxiv.org/abs/2203.06256

Rustand, D., van Niekerk, J., Rue, H., Tournigand, C., Rondeau, V. and Briollais, L. (2023). Bayesian Estimation of Two-Part Joint Models for a Longitudinal Semicontinuous Biomarker and a Terminal Event with R-INLA: Interests for Cancer Clinical Trial Evaluation. Biometrical Journal, 65, 2100322. https://doi.org/10.1002/bimj.202100322 https://arxiv.org/abs/2010.13704

Rue, H., Martino, S. and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71: 319-392. https://doi.org/10.1111/j.1467-9868.2008.00700.x

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1–48. https://doi.org/10.18637/jss.v067.i01

Contact: `<INLAjoint@gmail.com>`

Examples

```
# joint model with 3 longitudinal / 3 competing risks of event
data(Longsim)
data(Survsim)
f1 <- function(x) x^2 # quadratic function of time for first marker
Nsplines <- splines::ns(Longsim$time, knots=2) # 2 ns splines for second marker
f2 <- function(x) predict(Nsplines, x)[,1]
f3 <- function(x) predict(Nsplines, x)[,2]

if(requireNamespace("INLA")){
 JMINLA <- joint(
  formLong = list(Y1 ~ time + f1(time) + ctsX + binX + (1 + time + f1(time) | Id),
                 Y2 ~ time + f2(time) + f3(time) + ctsX + binX + (1 | Id),
                 Y3 ~ time + ctsX + binX + (1 | Id)),
  formSurv = list(INLA::inla.surv(deathTimes, Event1) ~ binX + ctsX,
                 INLA::inla.surv(deathTimes, Event2) ~ binX,
                 INLA::inla.surv(deathTimes, Event3) ~ ctsX),
  dataLong = Longsim, dataSurv=Survsim, id = "Id", timeVar = "time", corLong=TRUE,
  family = c("gaussian", "poisson", "binomial"), basRisk = c("rw1", "rw1", "rw1"),
  assoc = list(c("CV", "CS", ""),  c("CV", "", "SRE"), c("", "CV", "")),
  control=list(int.strategy="eb"))

 summary(JMINLA)
 # 'sdcor' to switch from variance-covariance to standard
 # deviation-correlation and 'hazr' to switch parameters
 # in survival submodels from mean to hazard ratios (exp(mean)).
 summary(JMINLA, sdcor=TRUE, hazr=TRUE)
}
```

`joint.compare` *Compare INLAjoint models*

Description

Pairwise comparison of joint models based on pointwise WAIC differences. Positive dWAIC means model2 fits better (lower WAIC).

Usage

```
joint.compare(...)
```

Arguments

`...`	Two or more objects of class `INLAjoint`.

Value

A data.frame with one row per pair of models.

`joint.rerun` *Rerun a model fitted with INLAjoint*

Description

Reruns inla() for an object of class `INLAjoint` returned by the `joint` function. The rerun starts with posterior values from previous run and can sometimes improve the model fit (for very complex models or unstable parameter estimates due to low information in the data)

Usage

```
joint.rerun(model, ...)
```

Arguments

`model`	an object containing a model fitted with the joint() function.
`...`	Extra arguments.

Value

An object of class `INLAjoint` containing a model fitted with the joint() function.

See Also

`joint`.

`joint.run` *Run a model fitted with INLAjoint*

Description

Runs inla() for an object of class `INLAjoint` returned by the `joint` function with argument ‘run‘ set to FALSE. The rerun starts with posterior values from previous run and can sometimes improve the model fit (for very complex models or unstable parameter estimates due to low information in the data)

Usage

```
joint.run(model, silentMode = FALSE, class = "INLAjoint", ...)
```

Arguments

`model`	an object containing a model fitted with the joint() function.
`silentMode`	boolean to display messages about the fit procedure. Default is FALSE.
`class`	defines the class of the object created. Default is "INLAjoint" but can be switched to "inla".
`...`	Extra arguments.

Value

An object of class `INLAjoint` containing a model fitted with the joint() function.

See Also

`joint`.

`logLik.INLAjoint` *Extracts log-likelihood value from a given model fitted with INLAjoint*

Description

This function extracts log-likelihood value from INLAjoint objects.

Usage

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

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`LongMS` *Simulated univariate longitudinal dataset*

Description

A simulated dataset used to illustrate joint models with univariate longitudinal and multi-state survival component (see joint longitudinal and multi-state model in INLAjoint vignette). The corresponding survival dataset is named 'SurvMS'.

Usage

`LongMS`

Format

'LongMS' A data frame with 96 rows and 7 columns:

id Individual id

time Visit times

X Continuous covariate

y Gaussian longitudinal outcome ...

`Longsim` *Simulated multivariate longitudinal dataset*

Description

A simulated dataset used to illustrate some models (e.g., example of joint() function in the documentation). The corresponding survival dataset is named 'Survsim'.

Usage

`Longsim`

Format

'Longsim' A data frame with 96 rows and 7 columns:

Id Individual id

time Visit times

ctsX Continuous covariate

binX Binary covariate

Y1, Y2, Y3 Gaussian Longitudinal outcomes ...

`nobs.INLAjoint` *Extracts number of observations of each composant from a given model fitted with INLAjoint*

Description

This function extracts number of observations of each composant from INLAjoint objects.

Usage

```
## S3 method for class 'INLAjoint'
nobs(object, ...)
```

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`plot.INLAjoint` *Plot the output from a multivariate joint model for longitudinal and/or survival data*

Description

This function provide plots for the output of a multivariate joint model for longitudinal and/or survival data. The output can be stored into an object and manipulated as a list of ggplot outputs, described bellow.

Usage

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

Arguments

`x`	an object with the output of the the joint function
`...`	Extra arguments including: `sdcor`: logical indicating if the random effects correlation are to be shown. If FALSE the covariance is shown. `priors`: logical indicating if the priors are added to the posterior marginals plots.

Value

return a named list of `ggplot` objects containing:

`Outcomes` as a list of length equal the number of longitudinal outcomes plus the number of survival outcomes, each one including the plot for the posterior marginals of the associated fixed effects and residual or baseline variance (or standard error). Each element contains the plot for the posterior marginal.

`Covariances` the plots for the posterior marginal distribution of the covariance parameters.

`Associations` the plots for the posterior marginal distribution of the association parameters.

`Random` The plot for the fitted baseline risk functions as shown as the posterior mean and credible interval.

`plot.score_predictions` *Plot method for score_predictions objects*

Description

Produces time-dependent AUC and/or Brier score plots.

Usage

```
## S3 method for class 'score_predictions'
plot(
  x,
  which = NULL,
  add = FALSE,
  col = "blue",
  lwd = 2,
  pch = 19,
  main = NULL,
  legend_label = NULL,
  ...
)
```

Arguments

`x`	an object of class `"score_predictions"`.
`which`	character vector specifying which plots to produce. Options are `"auc"`, `"brier"`, or both. Default uses the metrics available in `x`.
`add`	logical; if `TRUE`, add lines to an existing plot instead of creating a new one. Useful for overlaying results from different models. Default is `FALSE`.
`col`	color for the lines and points. Default is `"blue"`.
`lwd`	line width. Default is 2.

`pch`	point character. Default is 19 (filled circle).
`main`	optional title(s) for the plot(s). If a single string is provided, it is used for all panels. If a named list with elements `"auc"` and/or `"brier"`, each is used for the respective panel.
`legend_label`	character string for the legend label when using `add = TRUE`.
`...`	additional graphical parameters passed to `plot()` or `lines()`.

`predict.INLAjoint` *Computes predictions for a given model fitted with INLAjoint*

Description

This function allows to compute predictions for a given model fitted with INLAjoint, the default behavior (without arguments) returns fitted values for each component of the model. It is also possible to supply a dataset for which predictions are required, this dataset must have the same structure as the dataset used for the model fitting (i.e., same columns). The default returned predictions corresponds to the linear predictors for each outcomes.

Usage

```
## S3 method for class 'INLAjoint'
predict(
  object,
  newData = NULL,
  newDataSurv = NULL,
  timePoints = NULL,
  NtimePoints = 50,
  NsampleHY = 20,
  NsampleFE = 20,
  NsampleRE = 50,
  id = NULL,
  Csurv = NULL,
  startTime = NULL,
  horizon = NULL,
  baselineHaz = "interpolation",
  return.samples = FALSE,
  FEonly = FALSE,
  survival = FALSE,
  CIF = FALSE,
  inv.link = FALSE,
  NidLoop = "auto",
  resErrLong = FALSE,
  return.RE = FALSE,
  set.samples = NULL,
  silentMode = FALSE,
  ...
)
```

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`newData`	a dataset with the same columns as those used to fit the model. When using a longitudinal marker to predict longitudinal and subsequent survival outcomes, only the longitudinal information (i.e., structure of the longitudinal data) is required. It is also possible to predict the average trajectories conditional on covariates by setting the value of the longitudinal outcomes included in the model to NA.
`newDataSurv`	a dataset for survival information (only useful when both longitudinal and survival data are provided for the predictions, otherwise using the argument newData is working too).
`timePoints`	a vector of the time points at which predictions are computed (for both longitudinal and survival outcomes), this also control the precision of the integration for time-dependent shared terms and the computation of cumulative risks (e.g., for survival or CIF curves), thus many time points will increase the accuracy of predictions. Default is NULL as these time points are automatically computed when not defined manually.
`NtimePoints`	number of time points at which the predictions are computed (for both longitudinal and survival outcomes), these time points are equidistant between time 0 and horizon time. This also control the precision of the integration for time-dependent shared terms and the computation of cumulative risks (e.g., for survival or CIF curves), thus many time points will increase the accuracy of predictions.
`NsampleHY`	number of samples for hyperparameters used to assess uncertainty when computing predictions. Default is 20.
`NsampleFE`	number of samples of fixed effects for each hyperparameters samples used to assess uncertainty when computing predictions. Default is 30 (i.e., 30 x NsampleHY).
`NsampleRE`	number of random effects realizations for each sample specified in 'NsampleHY' and 'NsampleFE'. Default is 50 (i.e., 50 x NsampleFE x NsampleHY, resulting in 20000 random effects samples per new individual with default values). These random effects realizations are conditional on observed longitudinal outcomes values provided in 'newData' and survival time provided in 'newDataSurv' when a survival model is included. If 'newDataSurv' is NULL, they are conditional on survival up to latest longitudinal recorded measurement. When outcomes are set to NA, the realizations are sampled from the marginal distribution of random effects.
`id`	name of the individual id variable, default is NULL as it is automatically grabbed from the fitted model but when fitting simple survival models, providing id when fitting the model is not mandatory and thus this can be useful (an explicit message is printed in this specific case).
`Csurv`	conditional survival, gives the starting value of the at-risk period (i.e., starting value at which risk predictions for survival models are computed). Default is the last longitudinal observation time provided in 'newData' but this is replaced by the value of 'Csurv' when provided. Can be one value or one value per predicted id.

`startTime`	define a starting time for predictions.
`horizon`	horizon of the prediction.
`baselineHaz`	method used to evaluate the baseline hazard value, default is 'interpolation' which is currently recommended. Experimental alternatives are being developed, including 'splines' for an interpolation with splines but has not been properly validated with simulations yet.
`return.samples`	boolean, when set to TRUE the samples are returned instead of summary statistics over the samples. Default is FALSE.
`FEonly`	boolean, when set to TRUE, only fixed effects are involved for predictions computations.
`survival`	boolean, when set to TRUE the summary statistics over survival functions are computed in addition to the summary statistics over the risk functions.
`CIF`	boolean, when set to TRUE the summary statistics over cumulative incidence functions are computed in addition to the summary statistics over the risk functions. Only applies to competing risks.
`inv.link`	boolean, when set to TRUE the summary statistics are computed over the predictions of longitudinal components after applying the inverse link function for each samples in addition to the summary statistics over the linear predictors.
`NidLoop`	Gives the number of individuals for which we compute predictions at once. For large number of individuals, this will loop over groups of 'NidLoop' individuals and could make predictions computations faster.
`resErrLong`	boolean, when set to TRUE the residual error for Gaussian or lognormal longitudinal outcomes is added to the uncertainty of predictions (default is FALSE which predicts the true underlying value of the longitudinal marker, i.e., error-free).
`return.RE`	boolean, when set to TRUE returns summary of random effects posterior distribution for each predicted individual. The summary is obtained from fitted random effects summaries across hyperparameter samples and includes mean, sd, mode and quantiles (0.025, 0.5, 0.975).
`set.samples`	replace random effects with pre-sampled values.
`silentMode`	a boolean that will stop printing messages during computations if turned to TRUE.
`...`	Extra arguments.

`print.plot.INLAjoint` *Prints plot the output from a multivariate joint model for longitudinal and/or survival data*

Description

The plots of a model are grouped by categories, first are the fixed effects and residual error of longitudinal and survival submodels, referred to as 'Outcomes' or 'O'. Then the variance-covariance of random effects (or standard deviations and correlations when the argument 'sdcor' is set to TRUE in the call of the plot function), referred to as 'Covariances' or 'C'. Association parameters referred to as 'Associations' or 'A' for linear associations and 'NL_Associations' or 'N' for non-linear associations. Baseline hazard curves referred to as 'Baseline' or 'B' and baseline hazard related parameters referred to as 'ParamBaseline' or 'P'. It is possible to select specific plots to print by specifying the names or corresponding letters in the argument 'which'.

Usage

```
## S3 method for class 'plot.INLAjoint'
print(
  x,
  which = c("all", "Outcomes", "Covariances", "Associations", "Baseline"),
  ...
)
```

Arguments

`x`	an object with the output of the the joint function
`which`	name of required plots. Default is "all". It can be a character or named list. If it is a list, each element can be either character or numeric to select from the elements to be visualized. Ex.: which = list(Outcomes = "L1") and list(Outcomes = 1) will produce the same output.
`...`	Extra arguments.

`print.score_predictions`

Print method for score_predictions objects

Description

Print method for score_predictions objects

Usage

```
## S3 method for class 'score_predictions'
print(x, digits = 4, ...)
```

Arguments

`x`	an object of class `"score_predictions"`.
`digits`	number of digits to display. Default is 4.
`...`	additional arguments (ignored).

`ranef.INLAjoint` *Extracts random effects values from a given model fitted with INLA-joint*

Description

This function extracts random effects values from INLAjoint objects.

Usage

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

Arguments

`object`	an object that contains a model fitted with INLAjoint.
`...`	Extra arguments.

`score_predictions` *Compute time-dependent predictive accuracy metrics (AUC and Brier Score)*

Description

Computes time-dependent AUC and Brier score from predicted risks and observed outcomes.

The function handles censoring through the IPCW (Inverse Probability of Censoring Weighted) method, which reweights contributions by the inverse of the Kaplan-Meier estimate of the censoring distribution.

Usage

```
score_predictions(
  risk,
  pred_times,
  time,
  event,
  event_all = NULL,
  metrics = c("auc", "brier")
)
```

Arguments

`risk`	a numeric matrix of dimension $n \times T$ giving the predicted risk (i.e., event probability) for each subject (row) at each evaluation time (column). For survival predictions, this should be $1 - S(t)$. For competing risks, this should be the cause-specific cumulative incidence function (CIF).
`pred_times`	a numeric vector of length T giving the evaluation time points corresponding to the columns of `risk`.
`time`	a numeric vector of length n giving the observed event or censoring time for each subject.
`event`	a numeric or integer vector of length n giving the event indicator for each subject (1 = event occurred, 0 = censored). For competing risks, this should be the cause-specific indicator (1 = cause of interest, 0 = otherwise); see Details.
`event_all`	optional numeric or integer vector of length n giving the overall event indicator (1 = any event occurred, 0 = truly censored). Only needed for competing risks, where the censoring distribution must distinguish true censoring from competing events. When `NULL` (default), `event` is used (appropriate for single-risk settings).
`metrics`	character vector specifying which metrics to compute. Options are `"auc"`, `"brier"`, or both (default is `c("auc", "brier")`).

Details

Brier Score (IPCW):

The time-dependent Brier score at evaluation time t is defined as:

$$BS(t) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\hat{\pi}_i(t)^2 \cdot I(T_i > t)}{\hat{G}(t)} + \frac{(1 - \hat{\pi}_i(t))^2 \cdot I(T_i \leq t) \cdot \delta_i}{\hat{G}(T_i)} \right]$$

where $\hat{\pi}_i(t)$ is the predicted risk for subject i at time t, T_i is the observed time, δ_i is the event indicator, and $\hat{G}(\cdot)$ is the Kaplan-Meier estimate of the censoring survival function.

AUC (IPCW):

The time-dependent AUC at evaluation time t measures the concordance between predicted risks and observed outcomes over all case-control pairs:

$$AUC(t) = \frac{\sum_i \sum_j I(T_i \leq t)\delta_i I(T_j > t) I(\hat{\pi}_i(t) > \hat{\pi}_j(t)) w_{ij}(t)}{\sum_i \sum_j I(T_i \leq t)\delta_i I(T_j > t) w_{ij}(t)}$$

where the IPCW weight is $w_{ij}(t) = 1/(\hat{G}(T_i) \cdot \hat{G}(t))$. Ties in predicted risk contribute 0.5.

Competing Risks:

For competing risks, pass the cause-specific CIF as the `risk` matrix. Set `event` to the cause-specific indicator (1 = cause of interest, 0 = otherwise) and `event_all` to the overall event indicator (1 = any event occurred, 0 = truly censored). The `event_all` argument is needed because the IPCW censoring distribution must only count true independent censoring, not competing events. Each cause is scored separately with its own call to `score_predictions()`.

Value

An object of class `"score_predictions"`, which is a list containing:

scores A data.frame with columns `time` and the requested metrics (`AUC` and/or `Brier`).

metrics Character vector of metrics computed.

pred_times The evaluation time points.

n_subjects Number of subjects.

n_events Total number of events observed.

n_censored Total number of censored observations.

event_rate Proportion of subjects with events.

n_cases Number of cases (events before each `pred_times`) used for AUC.

n_controls Number of controls (event-free at each `pred_times`) used for AUC.

See Also

`predict.INLAjoint` for computing predictions.

Examples

```
## Not run:
if(requireNamespace("INLA")){

# simulate longitudinal + survival data
set.seed(1)
n <- 500
sex <- rbinom(n, 1, 0.5)
b <- MASS::mvrnorm(n, mu = c(0, 0),
                   Sigma = matrix(c(4, 1.2, 1.2, 0.6), 2, 2))
long_data <- do.call(rbind, lapply(1:n, function(i) {
  times <- seq(0, 10, by = 0.5)
  Y <- 5 - 0.3 * times + 0.8 * sex[i] + b[i,1] + b[i,2] * times +
       rnorm(length(times), 0, 0.5)
  data.frame(id = i, time = times, Y = Y, sex = sex[i])
}))
surv_data <- do.call(rbind, lapply(1:n, function(i) {
  tg <- seq(0, 15, length.out = 500)
  Yt <- 5 - 0.3*tg + 0.8*sex[i] + b[i,1] + b[i,2]*tg
  h <- exp(-4 + 0.4 * Yt)
  H <- cumsum(c(0, (h[-1]+h[-length(h)])/2 * diff(tg)))
  et <- tg[which(H >= rexp(1))[1]]
  st <- min(ifelse(is.na(et), 10, et), 10)
  data.frame(id = i, survtime = st, event = as.integer(st < 10), sex = sex[i])
}))
long_data <- merge(long_data, surv_data[, c("id","survtime")], by="id")
long_data <- long_data[long_data$time <= long_data$survtime, ]
long_data$survtime <- NULL

# train/test split
train_ids <- 1:480
```

```
test_ids <- 481:500
train_long <- long_data[long_data$id %in% train_ids, ]
train_surv <- surv_data[surv_data$id %in% train_ids, ]
test_long  <- long_data[long_data$id %in% test_ids, ]
test_surv  <- surv_data[surv_data$id %in% test_ids, ]

# fit joint model on training data
fit <- joint(
  formSurv = INLA::inla.surv(survtime, event) ~ sex,
  dataSurv = train_surv,
  formLong = list(Y ~ time + sex + (1 + time | id)),
  dataLong = train_long,
  id = "id", timeVar = "time", assoc = "CV"
)

# predict for test subjects from a landmark
landmark <- 2
horizon <- 9
eval_times <- seq(4, 8)
valid_ids <- test_surv$id[test_surv$survtime >= landmark]
newdata <- test_long[test_long$id %in% valid_ids & test_long$time <= landmark, ]
obs_per_id <- table(newdata$id)
valid_ids <- as.integer(names(obs_per_id[obs_per_id >= 2]))
newdata <- newdata[newdata$id %in% valid_ids, ]
test_surv_valid <- test_surv[test_surv$id %in% valid_ids, ]

# fine grid that includes the integer evaluation times
fine_grid <- sort(unique(c(seq(landmark, horizon, length.out = 50), eval_times)))
pred <- predict(fit, newData = newdata, horizon = horizon,
                Csurv = landmark, survival = TRUE,
                timePoints = fine_grid)

# reshape predictions into a risk matrix
predS <- pred$PredS
ids <- unique(predS$id)
all_times <- sort(unique(predS$time))
surv_mat <- matrix(predS$Surv_quant0.5, nrow = length(ids),
                   ncol = length(all_times), byrow = TRUE)
risk_mat <- 1 - surv_mat

# select integer evaluation times for scoring
keep <- match(eval_times, all_times)
risk_mat <- risk_mat[, keep]
pred_times <- eval_times

# compute scores
sc <- score_predictions(
  risk = risk_mat,
  pred_times = pred_times,
  time = test_surv_valid$survtime,
  event = test_surv_valid$event
)
print(sc)
```

```
plot(sc)
}

## End(Not run)
```

`setup_FE_model` *Setup fixed effects part for longitudinal marker k*

Description

Setup fixed effects part for longitudinal marker k (internal function) input:

Usage

```
setup_FE_model(formula, dataset, timeVar, k, dataOnly)
```

Arguments

`formula`	with lme4 format (fixed effects and random effects in the same object)
`dataset`	that contains the outcome
`timeVar`	name of time variable
`k`	identifies the longitudinal marker among 1:K markers
`dataOnly`	boolean for internal use, indicates if only preparing data (i.e., not fitting the model) output:

Value

colnames(FE) names of the fixed effects (interactions are separated by ".X." instead of ":" to facilitate their manipulation)

FE values of the fixed effects

`setup_RE_model` *Setup random effects part for longitudinal marker k*

Description

Setup random effects part for longitudinal marker k (internal function) input:

Usage

```
setup_RE_model(formula, dataset, k)
```

Arguments

`formula`	with lme4 format (fixed effects and random effects in the same object)
`dataset`	that contains the outcome
`k`	identifies the longitudinal marker among 1:K markers output:

Value

colnames(RE_mat) names of the random effects

RE_mat values of the random effects

`setup_S_model` *Setup survival part for outcome m*

Description

Setup survival part for outcome m (internal function) input:

Usage

```
setup_S_model(
  formula,
  formLong,
  dataSurv,
  LSurvdat,
  timeVar,
  assoc,
  id,
  m,
  K,
  M,
  NFT,
  corLong,
  dataOnly,
  SurvInfo,
  strata
)
```

Arguments

`formula`	with inla.surv() object as an outcome
`formLong`	formula from the longitudinal part, if any
`dataSurv`	dataset(s) for the survival part
`LSurvdat`	dataset for the longitudinal part converted to survival format (internal, used to get covariates if missing in the survival dataset when sharing linear predictors including covariates from longitudinal into survival)

`timeVar`	names of the variables that are time-dependent (only linear for now)
`assoc`	association parameters between K longitudinal outcomes and M survival outcomes (list of K vectors of size M)
`id`	name of the variable that gives the individual id
`m`	identifies the outcome among 1:M time-to-event outcomes
`K`	number of longitudinal outcomes
`M`	number of survival outcomes
`NFT`	maximum number of functions of time (fixed value)
`corLong`	boolean that indicates if random effects across longitudinal markers are correlated, when multiple longitudinal markers are included in the model
`dataOnly`	boolean for internal use, indicates if only preparing data (i.e., not fitting the model)
`SurvInfo`	information about survival submodels for internal use
`strata`	covariate for stratified proportional hazards model output:

Value

YS_data includes the values of the survival outcome and covariates associated to this survival part, with the association parameters but the provided id are temporary and they will be updated after the cox expansion to make them unique and allow for time dependency

YSformF formula for this survival outcome (not including association parameters)

`setup_Y_model` *Setup outcome for longitudinal marker*

Description

Setup outcome for longitudinal marker (internal function) input:

Usage

```
setup_Y_model(formula, dataset, family, k)
```

Arguments

`formula`	with lme4 format (fixed effects and random effects in the same object)
`dataset`	that contains the outcome
`family`	of the outcome (given to check if the distribution matches but the check is not done yet)
`k`	identifies the longitudinal marker among 1:K markers output:

Value

YL.name name of the outcome

YL values of the outcome

`SurvMS` *Simulated multi-state survival dataset*

Description

A simulated dataset used to illustrate multi-state models (see multi-state model and joint longitudinal and multi-state model in INLAjoint vignette). The corresponding longitudinal dataset is named 'LongMS'.

Usage

```
SurvMS
```

Format

'SurvMS' A list of 3 data frames with 50 rows and 5 columns, 50 rows and 5 columns, 28 rows and 5 columns for each of the 3 transition intensities, respectively:

id Individual id

Tstart Beginning of the 'at-risk' period

Tstop Event time

status Transition indicator

X Continuous covariate ...

`Survsim` *Simulated competing risks survival dataset*

Description

A simulated dataset used to illustrate some models (e.g., example of joint() function in the documentation). The corresponding longitudinal dataset is named 'Longsim'.

Usage

```
Survsim
```

Format

'Survsim' A data frame with 15 rows and 5 columns:

Id Individual id

deathTimes Event time

ctsX Continuous covariate

binX Binary covariate

Event1, Event2, Event3 Event indicator for the 3 competing risks (individual is censored when the 3 events are 0) ...

Index