Package: JMbayes2 (via r-universe)

October 15, 2026

Type Package

Title Extended Joint Models for Longitudinal and Time-to-Event Data

Version 0.6-5

Maintainer Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

Date 2026-10-15

BugReports https://github.com/drizopoulos/JMbayes2/issues

Description Fit joint models for longitudinal and time-to-event data under the Bayesian approach. Multiple longitudinal outcomes of mixed type (continuous/categorical) and multiple event times (competing risks and multi-state processes) are accommodated. Rizopoulos (2012, ISBN:9781439872864).

Suggests lattice, knitr, rmarkdown, pkgdown

Encoding UTF-8

Depends survival, nlme, GLMMadaptive, splines

Imports coda, Rcpp, parallel, parallelly, matrixStats, ggplot2, gridExtra, abind, MASS, survC1

LinkingTo Rcpp, RcppArmadillo

LazyLoad yes

LazyData yes

License GPL (>= 3)

URL https://drizopoulos.github.io/JMbayes2/

RoxygenNote 7.3.3

NeedsCompilation yes

Author Dimitris Rizopoulos [aut, cre] (ORCID: <https://orcid.org/0000-0001-9397-0900>), Pedro Miranda-Afonso [aut] (ORCID: <https://orcid.org/0000-0001-6708-9597>), Grigorios Papageorgiou [aut]

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

Date/Publication 2026-10-15 19:15:35 UTC

RemoteUrl https://github.com/cran/JMbayes2

RemoteRef HEAD

RemoteSha 31c2b97cd8ec929dde205e3ceb587d8944e1901b

Contents

Accuracy Measures . . . 2
aids . . . 6
consensus . . . 7
crisk_setup . . . 8
jm . . . 9
jm coda Methods . . . 17
jm Methods . . . 20
JMbayes2 . . . 24
lme . . . 25
pbc2 . . . 26
ppcheck . . . 28
Predictions . . . 30
prothro . . . 33
rc_setup . . . 34
slicer . . . 35
variogram . . . 36

Index **37**

`Accuracy Measures` *Time-Dependent Predictive Accuracy Measures for Joint Models*

Description

Using the available longitudinal information up to a starting time point, these functions compute estimates of the ROC curve and the AUC, the Brier score and expected predictive cross-entropy at a horizon time point based on joint models.

Usage

```
tvROC(object, newdata, Tstart, ...)

## S3 method for class 'jm'
tvROC(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, type_weights = c("model-based", "IPCW"), ...)

tvAUC(object, newdata, Tstart, ...)

## S3 method for class 'jm'
tvAUC(object, newdata, Tstart, Thoriz = NULL,
```

```
    Dt = NULL, type_weights = c("model-based", "IPCW"), ...)

## S3 method for class 'tvROC'
tvAUC(object, ...)

calibration_plot(object, newdata, Tstart, ...)

## S3 method for class 'jm'
calibration_plot(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, df_ns = NULL, plot = TRUE,
    col = "red", lty = 1, lwd = 1,
    add_CI = TRUE, col_CI = "lightgrey",
    add_density = TRUE, col_dens = "grey",
    xlab = "Predicted Probabilities",
    ylab = "Observed Probabilities", main = "", ...)

calibration_metrics(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, df_ns = NULL, ...)

tvBrier(object, newdata, Tstart, ...)

## S3 method for class 'jm'
tvBrier(object, newdata, Tstart, Thoriz = NULL, Dt = NULL,
    integrated = FALSE, type_weights = c("model-based", "IPCW"),
    model_weights = NULL, eventData_fun = NULL,
    parallel = c("snow", "multicore"),
    cores = parallelly::availableCores(omit = 1L), ...)

tvEPCE(object, newdata, Tstart, Thoriz = NULL, Dt = NULL, eps = 0.001,
    model_weights = NULL, eventData_fun = NULL,
    parallel = c("snow", "multicore"),
    cores = parallelly::availableCores(omit = 1L), ...)

create_folds(data, V = 5, id_var = "id",
    method = c("CV", "Bootstrap"), strata = NULL, seed = 123L)
```

Arguments

object	an object inheriting from class jm, except for tvAUC.tvROC() where this is an object of class tvROC. For tvBrier() and tvEPCE() it can also be a library of joint models.
newdata	a data.frame that contains the longitudinal and covariate information for the subjects for which prediction of survival probabilities is required. The names of the variables in this data.frame must be the same as in the data.frames that were used to fit the linear mixed effects and the event process model that were supplied as the two first argument of jm.
Tstart	numeric scalar denoting the time point up to which longitudinal information is to be used to derive predictions.

Thoriz	numeric scalar denoting the time point for which a prediction of the survival status is of interest; Thoriz must be later than Tstart and either Dt or Thoriz must be specified. If Thoriz is NULL is set equal to Tstart + Dt.
Dt	numeric scalar denoting the length of the time interval of prediction; either Dt or Thoriz must be specified.
integrated	logical; if TRUE the integrated Brier score is calculated.
type_weights	character string denoting the type of weights to use to account for censorting. Options are model-based (default) and inverse probability of censoring weighting (using the Kaplan-Meier estimate of the censoring distribution).
eps	numeric scalar used in the approximation of the hazard function.
model_weights	a numeric vector of weights to combine predictions when object is a list of joint models of class "jmList".
eventData_fun	a function that takes as input the newdata and produces the dataset used for the event process model. This is useful when, for example, the event process model contains other time-varying covariates. It is important that this function does not alter the ordering of the subjects in newdata.
parallel	character string; what type of parallel computing to use.
cores	integer denoting the number of cores to be used when a library of joint models has been provided in object. If cores = 1, no parallel computing is used.
df_ns	the degrees of freedom for the natural cubic spline of the cloglog transformation of the predicted probabilities used in the Cox model that assesses calibration. The default is 3 unless there are less than 25 events in the interval (Tstart, Thoriz] in which case it is 2.
plot	logical; should a plot be produced. If FALSE, a list is returned with the observed and predicted probabilities.
add_CI	logical; should 0.95 pointwise confidence intervals be added around the calibration line.
col_CI	character; the color of the shaded area representing the 0.95 pointwise confidence intervals around the calibration line.
add_density	logical; should the kernal density estimation of the predicted probabilities be superimposed in the calibration plot.
col, lwd, lty, col_dens, xlab, ylab, main	graphical parameters.
data	the data.frame to split in folds.
V	numeric scalar denoting the number of folds for cross-validation or the number of sample for the Bootstrap methods.
id_var	character string denoting the name of the subject id variable in data.
strata	character vector with the names of stratifying variables.
method	character string indicating which method to use to create the training and testing datasets in create_folds(). The default is V-fold cross-validation. For the Bootstrap option, V samples with replacement from the original dataset are proruced as training data. The testing data contains the subjects that were not selected in the respective Bootstrap sample.
seed	integer denoting the seed.
...	additional arguments passed to predict.jm().

Value

A list of class `tvAUC` with components:

`auc`	a numeric scalar denoting the estimated prediction error.
`Tstart`	a copy of the `Tstart` argument.
`Thoriz`	a copy of the `Thoriz` argument.
`nr`	a numeric scalar denoting the number of subjects at risk at time `Tstart`.
`classObject`	the class of `object`.
`nameObject`	the name of `object`.

A list of class `tvROC` with components:

`TP`, `FP`, `nTP`, `nFN`, `nTN`, `qSN`, `qSP`, `qOverall`	accuracy indexes.
`F1score`, `Youden`	numeric scalars with the optimal cut-point using the F1 score and the Youden index.
`thr`	numeric vector of thresholds.
`Tstart`	a copy of the `Tstart` argument.
`Thoriz`	a copy of the `Thoriz` argument.
`nr`	a numeric scalar denoting the number of subjects at risk at time `Tstart`.
`classObject`	the class of `object`.
`nameObject`	the name of `object`.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Antolini, L., Boracchi, P., and Biganzoli, E. (2005). A time-dependent discrimination index for survival data. *Statistics in Medicine* **24**, 3927–3944.

Commenges, D., Liquet, B., and Proust-Lima, C. (2012). Choice of prognostic estimators in joint models by estimating differences of expected conditional Kullback-Leibler risks. *Biometrics* **68**, 380–387.

Harrell, F., Kerry, L. and Mark, D. (1996). Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine* **15**, 361–387.

Heagerty, P. and Zheng, Y. (2005). Survival model predictive accuracy and ROC curves. *Biometrics* **61**, 92–105.

Rizopoulos, D. (2016). The R package JMbayes for fitting joint models for longitudinal and time-to-event data using MCMC. *Journal of Statistical Software* **72(7)**, 1–45. doi:10.18637/jss.v072.i07.

Rizopoulos, D. (2012) *Joint Models for Longitudinal and Time-to-Event Data: with Applications in R*. Boca Raton: Chapman and Hall/CRC.

Rizopoulos, D. (2011). Dynamic predictions and prospective accuracy in joint models for longitudinal and time-to-event data. *Biometrics* **67**, 819–829.

Rizopoulos, D., Molenberghs, G. and Lesaffre, E.M.E.H. (2017). Dynamic predictions with time-dependent covariates in survival analysis using joint modeling and landmarking. *Biometrical Journal* **59**, 1261–1276.

See Also

predict, jm

Examples

```
# We fit a multivariate joint model
pbc2.id$status2 <- as.numeric(pbc2.id$status != 'alive')
CoxFit <- coxph(Surv(years, status2) ~ sex, data = pbc2.id)
fm1 <- lme(log(serBilir) ~ ns(year, 3) * sex, data = pbc2,
           random = ~ ns(year, 3) | id, control = lmeControl(opt = 'optim'))
fm2 <- lme(prothrombin ~ ns(year, 2) * sex, data = pbc2,
           random = ~ ns(year, 2) | id, control = lmeControl(opt = 'optim'))
fm3 <- mixed_model(ascites ~ year * sex, data = pbc2,
                   random = ~ year | id, family = binomial())

jointFit <- jm(CoxFit, list(fm1, fm2, fm3), time_var = "year", n_chains = 1L)

roc <- tvROC(jointFit, newdata = pbc2, Tstart = 4, Dt = 3, cores = 1L)
roc
tvAUC(roc)
plot(roc, legend = TRUE, optimal_cutoff = "Youden")
```

---

aids *Didanosine versus Zalcitabine in HIV Patients*

Description

A randomized clinical trial in which both longitudinal and survival data were collected to compare the efficacy and safety of two antiretroviral drugs in treating patients who had failed or were intolerant of zidovudine (AZT) therapy.

Format

A data frame with 1408 observations on the following 9 variables.

`patient` patients identifier; in total there are 467 patients.

`Time` the time to death or censoring.

`death` a numeric vector with 0 denoting censoring and 1 death.

`CD4` the CD4 cells count.

`obstime` the time points at which the CD4 cells count was recorded.

`drug` a factor with levels `ddC` denoting zalcitabine and `ddI` denoting didanosine.

`gender` a factor with levels `female` and `male`.

`prevOI` a factor with levels `AIDS` denoting previous opportunistic infection (AIDS diagnosis) at study entry, and `noAIDS` denoting no previous infection.

`AZT` a factor with levels `intolerance` and `failure` denoting AZT intolerance and AZT failure, respectively.

Note

The data frame `aids.id` contains the first CD4 cell count measurement for each patient. This data frame is used to fit the survival model.

References

Goldman, A., Carlin, B., Crane, L., Launer, C., Korvick, J., Deyton, L. and Abrams, D. (1996) Response of CD4+ and clinical consequences to treatment using ddI or ddC in patients with advanced HIV infection. *Journal of Acquired Immune Deficiency Syndromes and Human Retrovirology* **11**, 161–169.

Guo, X. and Carlin, B. (2004) Separate and joint modeling of longitudinal and event time data using standard computer packages. *The American Statistician* **58**, 16–24.

---

consensus *Consensus Combination of Posterior Draws from Joint Model Fits*

Description

This function combines posterior draws from multiple joint model fits into a single set of draws.

Usage

```
consensus(object, parm, method = c("union", "equal_weight", "var_weight"),
          seed = 123L)
```

Arguments

`object`	an object of class `"sliced_jm"`.
`parm`	a character vector with the names of parameters to combine. These must correspond to elements in `fit$mcmc[[parm]]` for each fit (e.g., `"gammas"`, `"alphas"`, `"betas1"`, `"betas2"`).
`method`	the consensus method used to combine draws. `"union"` concatenate draws across slices (no averaging). `"equal_weight"` compute an iteration-wise simple average across slices. `"var_weight"` compute an iteration-wise weighted average across slices using inverse-variance weights.
`seed`	an integer seed used for the within-slice random permutation step (used by `"equal_weight"` and `"var_weight"`).

Value

An object of class "consensus_jm" with components:

method the selected method.

parm the requested parameter(s) block(s).

n_splits number of fits combined.

draws named list of combined draw matrices (one per parameter block).

weights for "equal_weight" and "var_weight", a named list of weight matrices.

n_draws number of combined draws per parameter block.

summary named list of summary matrices with Mean, StDev, 2.5%, 97.5%, and P).

seed seed used.

Author(s)

Pedro Miranda-Afonso <p.mirandaafonso@erasmusmc.nl>

---

crisk_setup *Transform Competing Risks Data in Long Format*

Description

In a competing risks setting this function expands the data frame with a single row per subject to a data frame in the long format in which each subject has as many rows as the number of competing events.

Usage

```
crisk_setup(data, statusVar, censLevel,
    nameStrata = "strata", nameStatus = "status2")
```

Arguments

data	the data frame containing the competing risk data with a single row per subject.
statusVar	a character string denoting the name of the variable in data that identifies the status variable which equals 1 if the subject had any of the competing events and 0 otherwise.
censLevel	a character string or a scalar denoting the censoring level in the statusVar variable of data.
nameStrata	a character string denoting the variable that will be added in the long version of data denoting the various causes of event.
nameStatus	a character string denoting the variable that will be added in the long version of data denoting if the subject experience any of the competing events.

Value

A data frame in the long format with multiple rows per subject.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Rizopoulos, D. (2012) *Joint Models for Longitudinal and Time-to-Event Data: with Applications in R*. Boca Raton: Chapman and Hall/CRC.

Putter, H., Fiocco, M., and Geskus, R. (2007). Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine* **26**, 2389–2430.

Examples

```
head(crisk_setup(pbc2.id, "status", "alive"))
```

---

jm *Joint Models for Longitudinal and Time-to-Event Data*

Description

Fits multivariate joint models for longitudinal and time-to-event data.

Usage

```
jm(Surv_object, Mixed_objects, time_var, recurrent = FALSE,
  functional_forms = NULL, which_independent = NULL,
  base_hazard = NULL, data_Surv = NULL, id_var = NULL,
  priors = NULL, control = NULL, ...)

value(x, IE_time = NULL)
coefs(x, zero_ind = NULL, IE_time = NULL)
slope(x, eps = 0.001, direction = "both", IE_time = NULL)
velocity(x, eps = 0.001, direction = "both", IE_time = NULL)
acceleration(x, IE_time = NULL)
area(x, time_window = NULL, IE_time = NULL)
Delta(x, time_window = NULL, standardise = TRUE, IE_time = NULL)

vexpit(x)
Dexpit(x)

vexp(x)
Dexp(x)

vabs(x)
```

```
vlog(x)
vlog2(x)
vlog10(x)

vsqrt(x)
poly2(x)
poly3(x)
poly4(x)

tv(x, knots = NULL, ord = 2L)
```

Arguments

`Surv_object`	an object: • of class 'coxph' fitted by function `coxph()` from package **survival**, or • of class 'survreg' fitted by function `survreg()` from package **survival**.
`Mixed_objects`	a `list` of objects or a single object. Objects may be: • of class 'lme' fitted by function `lme()` from package **nlme**, or • of class 'MixMod' fitted by function `mixed_model()` from package **GLMMadaptive**.
`time_var`	a `character` string indicating the time variable in the mixed-effects model(s).
`recurrent`	a `character` string indicating "calendar" or "gap" timescale to fit a recurrent event model.
`functional_forms`	a `list` of formulas. Each formula corresponds to one longitudinal outcome and specifies the association structure between that outcome and the survival submodel as well as any interaction terms between the components of the longitudinal outcome and the survival submodel. See **Examples**.
`which_independent`	a numeric indicator matrix denoting which outcomes are independent. It can also be the character string `"all"` in which case all longitudinal outcomes are assumed independent. Only relevant in joint models with multiple longitudinal outcomes.
`base_hazard`	a `character` vector indicating the type of hazard function.
`data_Surv`	the `data.frame` used to fit the Cox/AFT survival submodel.
`id_var`	a `character` string indicating the id variable in the survival submodel.
`priors`	a named `list` of user-specified prior parameters: `mean_betas_HC` the prior mean vector of the normal prior for the regression coefficients of the covariates of the longitudinal model(s), which were hierarchically centered. `Tau_betas_HC` the prior precision matrix of the normal prior for the regression coefficients of the longitudinal model(s), which were hierarchically centered.

mean_betas_nHC a list of the prior mean vector(s) of the normal prior(s) for the regression coefficients of the covariates of the longitudinal model(s), which were not hierarchically centered.

Tau_betas_nHC a list of the prior precision matrix(ces) of the normal prior(s) for the regression coefficients of the longitudinal model(s), which were not Hierarchically Centered.

mean_bs_gammas the prior mean vector of the normal prior for the B-splines coefficients used to approximate the baseline hazard.

Tau_bs_gammas the prior precision matrix of the normal prior for the B-splines coefficients used to approximate the baseline hazard.

A_tau_bs_gammas the prior shape parameter of the gamma prior for the precision parameter of the penalty term for the B-splines coefficients for the baseline hazard.

B_tau_bs_gammas the prior rate parameter of the gamma prior for the precision parameter of the penalty term for the B-splines coefficients for the baseline hazard.

rank_Tau_bs_gammas the prior rank parameter for the precision matrix of the normal prior for the B-splines coefficients used to approximate the baseline hazard.

mean_gammas the prior mean vector of the normal prior for the regression coefficients of baseline covariates.

Tau_gammas the prior precision matrix of the normal prior for the regression coefficients of baseline covariates.

penalty_gammas a character string with value 'none', 'ridge', or 'horseshoe' indicating whether the coefficients of the baseline covariates included in the survival submodel should not be shrunk, shrank using ridge prior, or shrank using horseshoe prior, respectively.

A_lambda_gammas the prior shape parameter of the gamma prior for the precision parameter of the local penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_lambda_gammas the prior rate parameter of the gamma prior for the precision parameter of the local penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

A_tau_gammas the prior shape parameter of the gamma prior for the precision parameter of the global penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_tau_gammas the prior rate parameter of the gamma prior for the precision parameter of the global penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

A_nu_gammas the prior shape parameter of the gamma prior for the variance hyperparameter for the precision parameter of the local penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_nu_gammas the prior rate parameter of the gamma prior for the variance hyperparameter for the precision parameter of the local penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

A_xi_gammas the prior shape parameter of the gamma prior for the variance hyperparameter for the precision parameter of the global penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_xi_gammas the prior rate parameter of the gamma prior for the variance hyperparameter for the precision parameter of the global penalty term for the baseline regression coefficients. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

mean_alphas the prior mean vector of the normal prior for the association parameter(s).

Tau_alphas the prior mean vector of the normal prior for the association parameter(s).

penalty_alphas a character string with value 'none', 'ridge', 'horseshoe' indicating whether the coefficients association parameters should not be shrunk, shrank using ridge prior, or shrank using horseshoe prior, respectively.

A_lambda_alphas the prior shape parameter of the gamma prior for the precision parameter of the local penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_lambda_alphas the prior rate parameter of the gamma prior for the precision parameter of the local penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

A_tau_alphas the prior shape parameter of the gamma prior for the precision parameter of the global penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_tau_alphas the prior rate parameter of the gamma prior for the precision parameter of the global penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge' or penalty_gammas = 'horseshoe'.

A_nu_alphas the prior shape parameter of the gamma prior for the variance hyperparameter for the precision parameter of the local penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge', or penalty_gammas = 'horseshoe'.

B_nu_alphas the prior rate parameter of the gamma prior for the variance hyperparameter for the precision parameter of the local penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

A_xi_alphas the prior shape parameter of the gamma prior for the variance hyperparameter for the precision parameter of the global penalty term for the association parameters. Only relevant when penalty_gammas = 'ridge' or when penalty_gammas = 'horseshoe'.

B_xi_alphas the prior rate parameter of the gamma prior for the variance hyperparameter for the precision parameter of the global penalty term for the

association parameters. Only relevant when `penalty_gammas = 'ridge'` or when `penalty_gammas = 'horseshoe'`.

`gamma_prior_D_sds` logical; if `TRUE`, a gamma prior will be used for the standard deviations of the D matrix (variance-covariance matrix of the random effects). Defaults to `TRUE`

`D_sds_df` the prior degrees of freedom parameter for the half-t prior for the standard deviations of the D matrix (variance-covariance matrix of the random effects).

`D_sds_sigma` the prior sigma parameter vector for the half-t prior for the standard deviations of the D matrix (variance-covariance matrix of the random effects).

`D_sds_shape` the prior shape parameter for the gamma prior for the standard deviations of the D matrix (variance-covariance matrix of the random effects).

`D_sds_mean` the prior mean parameter vector for the gamma prior for the standard deviations of the D matrix (variance-covariance matrix of the random effects).

`D_L_etaLKJ` the prior eta parameter for the LKJ prior for the correlation matrix of the random effects.

`sigmas_df` the prior degrees of freedom parameter for the half-t prior for the error term(s).

`sigmas_sigma` the prior sigma parameter for the half-t prior for the error term(s).

`control` a list of control values with components:

`GK_k` the number of quadrature points for the Gauss Kronrod rule; options 15 and 7.

`n_chains` an integer specifying the number of chains for the MCMC. Defaults to 3.

`n_burnin` an integer specifying the number of burn-in iterations. Defaults to 500.

`n_iter` an integer specifying the number of total iterations per chain. Defaults to 3500.

`n_thin` an integer specifying the thinning of the chains. Defaults to 1.

`seed` the seed used in the sampling procedures. Defaults to 123.

`MALA logical`; if `TRUE`, the MALA algorithm is used when updating the elements of the Cholesky factor of the D matrix. Defaults to `FALSE`.

`save_random_effects logical`; if `TRUE`, the full MCMC results of the random effects will be saved and returned with the `jm` object. Defaults to `FALSE`.

`save_logLik_contributions logical`; if `TRUE`, the log-likelihood contributions are saved in the `mcmc` component of the `jm` object. Defaults to `FALSE`

`cores` an integer specifying the number of cores to use for running the chains in parallel; no point of setting this greater than `n_chains`.

`parallel` a character string indicating how the parallel sampling of the chains will be performed. Options are `"snow"` (default) and `"multicore"`.

basis character string with possible values "bs" (default) or "ns". When "bs" a B-spline basis is used to approximate the log baseline hazard function with degree of the spline specified by the Bsplines_degree. When "ns" a natrual cubic spline basis is used; in this case the value of the Bsplines_degree control argument is ignored.

Bsplines_degree the degree of the splines in each basis; default is quadratic splines.

base_hazard_segments the number of segments to split the follow-up period for the spline approximation of the log baseline hazard function. Defaults to 10.

timescale_base_hazard character string with possible values "identity" (default) or "log". When "identity" the spline basis is specified for the time variable in its orginal scale. When "log" the spline basis is specified for the logarithm of the time variable.

diff the order of the difference used in the penalty matrix for the coefficients of the splines used to approximate the log baseline hazard function. Defaults to 2.

knots a numeric vector with the position of the knots for the spline approximation of the log baseline hazard function. The default is equally-spaced knots starting from sqrt(.Machine$double.eps) until the maximum follow-up time.

x a numeric input variable.

knots a numeric vector of knots.

ord an integer denoting the order of the spline.

zero_ind a list with integer vectors indicating which coefficients are set to zero in the calculation of the value term. This can be used to include for example only the random intercept; default is NULL.

eps numeric scalar denoting the step-size for the finite difference approximation.

direction character string for the direction of the numerical derivative, options are "both", and "backward".

time_window numeric scalar controlling the time window used by the Delta() and area() functional forms. For area(), time_window specifies the lower limit of the interval over which the integral is evaluated. For Delta(), time_window specifies the length of the time interval (i.e., the contrast between t and t - time_window) over which the finite difference is computed; when set to NULL (the default), the contrast is taken between the current time and time 0.

standardise logical; controls whether the Delta() functional form returns a rate or a raw contrast. If TRUE, the difference between the values at the two time points is divided by the time distance between them, yielding a change per unit time. If FALSE, Delta() returns the raw difference over the specified time window. Defaults to TRUE.

IE_time a character string specifying the name of the intermediate event time variable in the data.frame used to fit the Cox/AFT survival submodel. For groups/subjects who did not experience the intermediate event, the time should be set to Inf. The same IE_time variable should be used when specifying multiple functional forms for the same longitudinal outcome.

...	arguments passed to control.

Details

The mathematical details regarding the definition of the multivariate joint model, and the capabilities of the package can be found in the vignette in the doc directory.

Notes:

- The ordering of the subjects in the datasets used to fit the mixed and Cox regression models needs to be the same.
- The units of the time variables in the mixed and Cox models need to be the same.

Value

A list of class jm with components:

mcmc	a list of the MCMC samples for each parameter.
acc_rates	a list of the acceptance rates for each parameter.
logLik	a matrix of dimensions [((n_iter - n_burnin)/n_thin)*n_thin, number of individuals], with element [i, j] being the conditional log-Likelihood value of the i^{th} iteration for the j^{th} individual.
mlogLik	a matrix of dimensions [((n_iter - n_burnin)/n_thin)*n_thin, number of individuals], with element [i, j] being the marginal log-Likelihood value of the i^{th} iteration for the j^{th} individual.
running_time	an object of class proc_time with the time used to run jm.
statistics	a list with posterior estimates of the parameters (means, medians, standard deviations, standard errors, effective sample sizes, tail probabilities, upper and lower bounds of credible intervals, etc.).
fit_stats	a list of lists with fit statistics (DIC, pD, LPML, CPO, WAIC) for both conditional and marginal formulations.
model_data	a list of data used to fit the model.
model_info	a list of components of the fit useful to other functions.
initial_values	a list with the initial values of the parameters.
control	a copy of the control values used to fit the model.
priors	a copy of the priors used to fit the model.
call	the matched call.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

methods.jm, coda_methods.jm

Examples

```
##############################################################################

##############################################
# Univariate joint model for serum bilirubin #
# 1 continuous outcome                       #
##############################################

# [1] Fit the mixed model using lme().
fm1 <- lme(fixed = log(serBilir) ~ year * sex + I(year^2) +
           age + prothrombin, random =  ~ year | id, data = pbc2)

# [2] Fit a Cox model, specifying the baseline covariates to be included in the
# joint model.
fCox1 <- coxph(Surv(years, status2) ~ drug + age, data = pbc2.id)

# [3] The basic joint model is fitted using a call to jm() i.e.,
joint_model_fit_1 <- jm(fCox1, fm1, time_var = "year",
        n_chains = 1L, n_iter = 11000L, n_burnin = 1000L)
summary(joint_model_fit_1)
traceplot(joint_model_fit_1)

##############################################################################

#########################################################################
# Multivariate joint model for serum bilirubin, hepatomegaly and ascites #
# 1 continuous outcome, 2 categorical outcomes                          #
#########################################################################

# [1] Fit the mixed-effects models using lme() for continuous
# outcomes and mixed_model() for categorical outcomes.
fm1 <- lme(fixed = log(serBilir) ~ year * sex,
           random = ~ year | id, data = pbc2)

fm2 <- mixed_model(hepatomegaly ~ sex + age + year, data = pbc2,
                   random = ~ year | id, family = binomial())

fm3 <- mixed_model(ascites ~ year + age, data = pbc2,
                   random = ~ year | id, family = binomial())

# [2] Save all the fitted mixed-effects models in a list.
Mixed <- list(fm1, fm2, fm3)

# [3] Fit a Cox model, specifying the baseline covariates to be included in the
# joint model.
fCox1 <- coxph(Surv(years, status2) ~ drug + age, data = pbc2.id)

# [4] The joint model is fitted using a call to jm() i.e.,
joint_model_fit_2 <- jm(fCox1, Mixed, time_var = "year",
      n_chains = 1L, n_iter = 11000L, n_burnin = 1000L)
summary(joint_model_fit_2)
traceplot(joint_model_fit_2)
```

```
################################################################################

######################
# Slope & Area Terms #
######################

# We extend model 'joint_model_fit_2' by including the value and slope term for
# bilirubin, the area term for hepatomegaly (in the log-odds scale), and the
# value and area term for spiders (in the log-odds scale).
# To include these terms into the model, we specify the 'functional_forms'
# argument. This should be a list of right side formulas. Each component of the
# list should have as name the name of the corresponding outcome variable. In
# the right side formula we specify the functional form of the association using
# functions 'value()', 'slope()' and 'area()'.
# Notes: (1) For terms not specified in the 'functional_forms' list, the default
# value functional form is used.

# [1] Fit the mixed-effects models using lme() for continuous outcomes
# and mixed_model() for categorical outcomes.
fm1 <- lme(fixed = log(serBilir) ~ year * sex, random = ~ year | id, data = pbc2)

fm2 <- mixed_model(hepatomegaly ~ sex + age + year, data = pbc2,
                   random = ~ year | id, family = binomial())

fm3 <- mixed_model(ascites ~ year + age, data = pbc2,
                   random = ~ year | id, family = binomial())

# [2] Save all the fitted mixed-effects models in a list.
Mixed <- list(fm1, fm2, fm3)

# [3] Fit a Cox model, specifying the baseline covariates to be included in the
# joint model.
fCox1 <- coxph(Surv(years, status2) ~ drug + age, data = pbc2.id)

# [4] Specify the list of formulas to be passed to the functional_forms argument
# of jm().
fForms <- list("log(serBilir)" = ~ value(log(serBilir)) + slope(log(serBilir)),
               "hepatomegaly" = ~ area(hepatomegaly),
               "ascites" = ~ value(ascites) + area(ascites))

# [5] The joint model is fitted using a call to jm() and passing the list
# to the functional_forms argument.
joint_model_fit_2 <- jm(fCox1, Mixed, time_var = "year",
                        functional_forms = fForms, n_chains = 1L,
                        n_iter = 11000L, n_burnin = 1000L)
summary(joint_model_fit_2)
```

---

jm coda Methods *Various Methods for Functions from the* coda *Package*

Description

Methods for an object of class "jm" for diagnostic functions.

Usage

```
traceplot(object, ...)

## S3 method for class 'jm'
traceplot(object,
  parm = c("all", "betas", "sigmas", "D", "bs_gammas",
           "tau_bs_gammas", "gammas", "alphas"), ...)

ggtraceplot(object, ...)

## S3 method for class 'jm'
ggtraceplot(object,
  parm = c("all", "betas", "sigmas", "D", "bs_gammas",
           "tau_bs_gammas", "gammas", "alphas"),
  linewidth = 1, alpha = 0.8,
  theme = c('standard', 'catalog', 'metro',
                'pastel', 'beach', 'moonlight', 'goo', 'sunset', 'custom'),
  grid = FALSE, gridrows = 3, gridcols = 1, custom_theme = NULL, ...)

gelman_diag(object, ...)

## S3 method for class 'jm'
gelman_diag(object,
  parm = c("all", "betas", "sigmas", "D", "bs_gammas",
           "tau_bs_gammas", "gammas", "alphas"), ...)

densplot(object, ...)

## S3 method for class 'jm'
densplot(object,
  parm = c("all", "betas", "sigmas", "D", "bs_gammas",
           "tau_bs_gammas", "gammas", "alphas"), ...)

ggdensityplot(object, ...)

## S3 method for class 'jm'
ggdensityplot(object,
  parm = c("all", "betas", "sigmas", "D", "bs_gammas",
           "tau_bs_gammas", "gammas", "alphas"),
  linewidth = 1, alpha = 0.6,
  theme = c('standard', 'catalog', 'metro', 'pastel',
                'beach', 'moonlight', 'goo', 'sunset', 'custom'),
  grid = FALSE, gridrows = 3, gridcols = 1, custom_theme = NULL, ...)
```

```
cumuplot(object, ...)

## S3 method for class 'jm'
cumuplot(object,
  parm = c("all", "betas", "sigmas", "D", "bs_gammas",
           "tau_bs_gammas", "gammas", "alphas"), ...)
```

Arguments

object	an object inheriting from class "jm".
parm	a character string specifying which parameters of the joint model to plot. Possible options are 'all', 'betas', 'alphas', 'sigmas', 'D', 'bs_gammas', 'tau_bs_gammas', or 'gammas'.
linewidth	the width of the traceplot line in mm. Defaults to 1.
alpha	the opacity level of the traceplot line. Defaults to 0.8.
theme	a character string specifying the color theme to be used. Possible options are 'standard', 'catalog', 'metro', 'pastel', 'beach', 'moonlight', 'goo', or 'sunset'. Note that this option supports fitted objects with three chains. If the object was fitted using a different number of chains then the colors are either automatically chosen, or can be specified by the user via the argument custom_theme.
grid	logical; defaults to FALSE. If TRUE, the plots are returned in grids split over multiple pages. For more details see the documentation for gridExtra::marrangeGrob().
gridrows	number of rows per page for the grid. Only relevant when using grid = TRUE. Defaults to 3.
gridcols	number of columns per page for the grid. Only relevant when using grid = TRUE. Defaults to 1.
custom_theme	A named character vector with elements equal to the number of chains (n_chains). The name of each element should be the number corresponding to the respective chain. Defaults to NULL.
...	further arguments passed to the corresponding function from the **coda** package.

Value

traceplot() Plots the evolution of the estimated parameter vs. iterations in a fitted joint model.

ggtraceplot() Plots the evolution of the estimated parameter vs. iterations in a fitted joint model using **ggplot2**.

gelman_diag() Calculates the potential scale reduction factor for the estimated parameters in a fitted joint model, together with the upper confidence limits.

densplot() Plots the density estimate for the estimated parameters in a fitted joint model.

ggdensityplot() Plots the evolution of the estimated parameter vs. iterations in a fitted joint model using **ggplot2**.

cumuplot() Plots the evolution of the sample quantiles vs. iterations in a fitted joint model.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

jm

Examples

```
# linear mixed model fits
fit_lme1 <- lme(log(serBilir) ~ year:sex + age,
                random = ~ year | id, data = pbc2)

fit_lme2 <- lme(prothrombin ~ sex,
                random = ~ year | id, data = pbc2)

# cox model fit
fit_cox <- coxph(Surv(years, status2) ~ age, data = pbc2.id)

# joint model fit
fit_jm <- jm(fit_cox, list(fit_lme1, fit_lme2), time_var = "year", n_chains = 1L)

# trace plot for the fixed effects in the linear mixed submodels
traceplot(fit_jm, parm = "betas")

# density plot for the fixed effects in the linear mixed submodels
densplot(fit_jm, parm = "betas")

# cumulative quantile plot for the fixed effects in the linear mixed submodels
cumuplot(fit_jm, parm = "betas")

# trace plot for the fixed effects in the linear mixed submodels
ggtraceplot(fit_jm, parm = "betas")
ggtraceplot(fit_jm, parm = "betas", grid = TRUE)
ggtraceplot(fit_jm, parm = "betas", custom_theme = c('1' = 'black'))

# trace plot for the fixed effects in the linear mixed submodels
ggdensityplot(fit_jm, parm = "betas")
ggdensityplot(fit_jm, parm = "betas", grid = TRUE)
ggdensityplot(fit_jm, parm = "betas", custom_theme = c('1' = 'black'))
```

---

jm Methods *Various Methods for Standard Generics*

Description

Methods for object of class "jm" for standard generic functions.

Usage

```
coef(object, ...)

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

fixef(object, ...)

## S3 method for class 'jm'
fixef(object, outcome = Inf, ...)

ranef(object, ...)

## S3 method for class 'jm'
ranef(object, outcome = Inf, post_vars = FALSE, ...)

terms(x, ...)

## S3 method for class 'jm'
terms(x, process = c("longitudinal", "event"),
                      type = c("fixed", "random"), ...)

model.frame(formula, ...)

## S3 method for class 'jm'
model.frame(formula, process = c("longitudinal", "event"),
                            type = c("fixed", "random"), ...)

model.matrix(object, ...)

## S3 method for class 'jm'
model.matrix(object, ...)

family(object, ...)

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

compare_jm(..., type = c("marginal", "conditional"),
  order = c("WAIC", "DIC", "LPML", "none"))
```

Arguments

object, x, formula	object inheriting from class "jm".
outcome	the index of the linear mixed submodel to extract the estimated fixed effects. If greater than the total number of submodels, extracts from all of them.

post_vars — logical; if TRUE, returns the variance of the posterior distribution.

process — which submodel(s) to extract the terms:

- if "longitudinal", the linear mixed model(s), or
- if "event", the survival model.

type — in terms() and model.frame(), which effects to select in the longitudinal process:

- if "fixed", the fixed-effects, or
- if "random", the random-efects.

in compare_jm(), which log-likelihood function use to calculate the criteria:

- if "marginal", the marginal log-likelihood, or
- if "conditional", the conditional log-likelihood.

... — further arguments; currently, none is used.
in compare_jm(), a series of jm objects.

order — which criteria use to sort the models in the output.

Details

coef() Extracts estimated fixed effects for the event process from a fitted joint model.

fixef() Extracts estimated fixed effects for the longitudinal processes from a fitted joint model.

ranef() Extracts estimated random effects from a fitted joint model.

terms() Extracts the terms object(s) from a fitted joint model.

model.frame() Creates the model frame from a fitted joint model.

model.matrix() Creates the design matrices for linear mixed submodels from a fitted joint model.

family() Extracts the error distribution and link function used in the linear mixed submodel(s) from a fitted joint model.

compare_jm() Compares two or more fitted joint models using the criteria WAIC, DIC, and LPML.

Value

coef() a list with the elements:

- gammas: estimated baseline fixed effects, and
- association: estimated association parameters.

fixef() a numeric vector of the estimated fixed effects for the outcome selected. If the outcome is greater than the number of linear mixed submodels, it returns a list of numeric vectors for all outcomes.

ranef() a numeric matrix with rows denoting the individuals and columns the random effects. If postVar = TRUE, the numeric matrix has the extra attribute "postVar".

terms() if process = "longitudinal", a list of the terms object(s) for the linear mixed model(s). if process = "event", the terms object for the survival model.

`model.frame()` if `process = "longitudinal"`, a list of the model frames used in the linear mixed model(s).
if `process = "event"`, the model frame used in the survival model.

`model.matrix()` a list of the design matrix(ces) for the linear mixed submodel(s).

`family()` a list of `family` objects.

`compare_jm()` a list with the elements:

- `table`: a table with the criteria calculated for each joint model, and
- `type`: the log-likelihood function used to calculate the criteria.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

jm

Examples

```
# linear mixed model fits
fit_lme1 <- lme(log(serBilir) ~ year:sex + age,
                random = ~ year | id, data = pbc2)

fit_lme2 <- lme(prothrombin ~ sex,
                random = ~ year | id, data = pbc2)

# cox model fit
fit_cox <- coxph(Surv(years, status2) ~ age, data = pbc2.id)

# joint model fit
fit_jm <- jm(fit_cox, list(fit_lme1, fit_lme2), time_var = "year",
    n_chains = 1L, n_iter = 11000L, n_burnin = 1000L)

# coef(): fixed effects for the event process
coef(fit_jm)

# fixef(): fixed effects for the first linear mixed submodel
fixef(fit_jm, outcome = 1)

# ranef(): random effects from all linear mixed submodels
head(ranef(fit_jm))

# terms(): random effects terms for the first linear mixed submodel
terms(fit_jm, process = "longitudinal", type = "random")[[1]]

# mode.frame(): model frame for the fixed effects in the second
# linear mixed submodel
head(model.frame(fit_jm, process = "longitudinal", type = "fixed")[[2]])
```

```
# model.matrix(): fixed effects design matrix for the first linear
# mixed submodel
head(model.matrix(fit_jm)[[1]])

# family(): family objects from both linear mixed submodels
family(fit_jm)

# compare_jm(): compare two fitted joint models
fit_lme1b <- lme(log(serBilir) ~ 1,
                 random = ~ year | id, data = pbc2)

fit_jm2 <- jm(fit_cox, list(fit_lme1b, fit_lme2), time_var = "year",
    n_chains = 1L, n_iter = 11000L, n_burnin = 1000L)

compare_jm(fit_jm, fit_jm2)
```

JMbayes2 *Extended Joint Models for Longitudinal and Time-to-Event Data*

Description

Fit joint models for longitudinal and time-to-event data under the Bayesian approach. Multiple longitudinal outcomes of mixed type (continuous/categorical) and multiple event times (competing risks and multi-state processes) are accommodated.

Details

Package:	JMbayes2
Type:	Package
Version:	0.6-5
Date:	2026-10-15
License:	GPL (>=3)

This package fits joint models for longitudinal and time-to-event data. It can accommodate multiple longitudinal outcomes of different type (e.g., continuous, dichotomous, ordinal, counts), and assuming different distributions, i.e., Gaussian, Student's-t, Gamma, Beta, unit Lindley, censored Normal, Binomial, Poisson, Negative Binomial, and Beta-Binomial. For the event time process, right, left and interval censored data can be handled, while competing risks and multi-sate processes are also covered.

JMbayes2 fits joint models using Markov chain Monte Carlo algorithms implemented in C++. The package also offers several utility functions that can extract useful information from fitted joint models. The most important of those are included in the **See also** Section below.

Author(s)

Dimitris Rizopoulos, Grigorios Papageorgiou, Pedro Miranda-Afonso

Maintainer: Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Rizopoulos, D. (2012). Joint Models for Longitudinal and Time-to-Event Data With Applications in R. Boca Raton: Chapman & Hall/CRC.

See Also

jm, methods.jm, coda_methods.jm

---

lme *Slice-aware model fitting generics*

Description

These functions are S3 generics exported by **JMbayes2** to enable method dispatch on the class of the `data` argument. When `data` is a `"sliced_data"` object (as produced by `slicer`), the corresponding slice-wise fitting methods are used.

Usage

```
lme(fixed, data, ...)
coxph(formula, data, ...)
mixed_model(fixed, data, ...)
```

Arguments

`fixed`	For `lme()` and `mixed_model()`, a model formula for the fixed effects.
`formula`	For `coxph()`, a survival model formula.
`data`	A data frame (default methods) or a `"sliced_data"` object (slice-wise methods).
`...`	Further arguments passed to the underlying model-fitting functions. For slice-wise methods, additional arguments include `parallel_out` and `cores` to control parallel execution across slices.

Details

When `data` is a regular data frame, the default methods call `nlme::lme()`, `survival::coxph()`, and `GLMMadaptive::mixed_model()`, respectively.

When `data` is a `"sliced_data"` object, the corresponding `*.sliced_data` methods fit the requested model independently within each slice and return a list of fits (one per slice).

Note

JMbayes2 exports S3 generics `lme()`, `coxph()`, and `mixed_model()` to enable dispatch on `"sliced_data"`. When JMbayes2 is attached, these names mask `nlme::lme`, `survival::coxph`, and `GLMMadaptive::mixed_model`. Use the `pkg::fun` form to call the original functions.

See Also

`slicer`, `nlme::lme`, `survival::coxph`, `GLMMadaptive::mixed_model`.

Examples

```
## Not run:

slc <- slicer(n_slices = 2, id_var = "id", data_long = pbc2, data_surv = pbc2.id)
n_cores <- max(parallel::detectCores() - 1L, 1L)

lme_fit <- lme(fixed  = log(serBilir) ~ year * sex,
               data   = slc$long,
               random = ~ year | id,
               cores  = n_cores)

cox_fit <- coxph(formula = Surv(years, status2) ~ sex,
                 data    = slc$surv,
                 cores   = n_cores)

mxm_fit <- mixed_model(fixed  = ascites ~ year + sex,
                       data   = slc$long,
                       random = ~ year | id,
                       family = binomial(),
                       cores  = n_cores)

## End(Not run)
```

---

pbc2 *Mayo Clinic Primary Biliary Cirrhosis Data*

Description

Follow up of 312 randomised patients with primary biliary cirrhosis, a rare autoimmune liver disease, at Mayo Clinic.

Format

A data frame with 1945 observations on the following 20 variables.

`id` patients identifier; in total there are 312 patients.

`years` number of years between registration and the earlier of death, transplantion, or study analysis time.

`status` a factor with levels `alive`, `transplanted` and `dead`.

`drug` a factor with levels `placebo` and `D-penicil`.

`age` at registration in years.

`sex` a factor with levels `male` and `female`.

`year` number of years between enrollment and this visit date, remaining values on the line of data refer to this visit.

`ascites` a factor with levels `No` and `Yes`.

`hepatomegaly` a factor with levels `No` and `Yes`.

`spiders` a factor with levels `No` and `Yes`.

`edema` a factor with levels `No edema` (i.e., no edema and no diuretic therapy for edema), `edema no diuretics` (i.e., edema present without diuretics, or edema resolved by diuretics), and `edema despite diuretics` (i.e., edema despite diuretic therapy).

`serBilir` serum bilirubin in mg/dl.

`serChol` serum cholesterol in mg/dl.

`albumin` albumin in g/dl.

`alkaline` alkaline phosphatase in U/liter.

`SGOT` SGOT in U/ml.

`platelets` platelets per cubic ml / 1000.

`prothrombin` prothrombin time in seconds.

`histologic` histologic stage of disease.

`status2` a numeric vector with the value 1 denoting if the patient was dead, and 0 if the patient was alive or transplanted.

Note

The data frame `pbc2.id` contains the first measurement for each patient. This data frame is used to fit the survival model.

References

Fleming, T. and Harrington, D. (1991) *Counting Processes and Survival Analysis*. Wiley, New York.

Therneau, T. and Grambsch, P. (2000) *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York.

ppcheck *Posterior Predictive Checks for Joint Models*

Description

It computes various posterior predictive checks for joint models.

Usage

```
ppcheck(object, nsim = 40L, newdata = NULL, seed = 123L,
    process = c("longitudinal", "event", "joint"),
    type = c("ecdf", "average-evolution", "variance-function",
             "variogram", "surv-uniform"),
   CI_ecdf = c("none", "binomial", "Dvoretzky-Kiefer-Wolfowitz"), CI_loess = FALSE,
    outcomes = Inf, percentiles = c(0.025, 0.975),
    random_effects = c("posterior_means", "mcmc", "prior"),
    params_mcmc = NULL, Fforms_fun = NULL, plot = TRUE, add_legend = TRUE,
    pos_legend = c("bottomright", "right"), main = "", xlab = NULL,
    ylab = NULL, col_obs = "black", col_rep = "lightgrey", lty_obs = 1,
    lty_rep = 1, lwd_obs = 1.5, lwd_rep = 1, line_main = NA,
    cex.main = 1.2, ylim = NULL, ...)
```

Arguments

object	an object inheriting from class "jm".
nsim	a numeric scalar denoting the number of replicated data.
newdata	a data.frame based on which to simulate replicated data. Relevant for cross-validated posterior predictive checks.
seed	the seed to use.
process	a character string indicating for which process to do the checks.
type	a character string indicating the type of checks. The default ecdf compares the empirical distribution function of the replicated data with the one of the observed data. For continous longitudinal outcomes, the average evolution, variance function and the sample variogram can also be used.
CI_ecdf	character string indicating the type of confidence interval for the emprical distribution function.
CI_loess	logical; if TRUE a 0.95 confidence interval for the loess curve is plotted.
outcomes	a numeric vector of indices indicating for which longitudinal outcomes to do the checks. The deault value Inf implies checking all longitudinal outcomes.
percentiles	a numeric vector of length two indicating the percentiles of the observed data to use when depicting the checks.
random_effects	a character string indicating what to do with the random effects.
params_mcmc	a list with an MCMC sample of the model parameters.

Fforms_fun	a function that calculates the functional forms.
plot	logical; if TRUE a plot is produced, otherwise the function returns a list with the values to create the figure.
add_legend	logical; if TRUE a legend is added.
pos_legend	character string indicating the position of the legend.
main, xlab, ylab, col_obs, col_rep, lty_obs, lty_rep, lwd_obs, lwd_rep, line_main, cex.main, ylim	graphical parameters.
...	extra argument passed to plot method.

Value

a figure with the checks.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

Examples

```
# Cox model for the composite event death or transplantation
pbc2.id$status2 <- as.numeric(pbc2.id$status != 'alive')
pbc2$status2 <- as.numeric(pbc2$status != 'alive')
CoxFit <- coxph(Surv(years, status2) ~ sex, data = pbc2.id)

# a linear mixed model for log serum bilirubin
fm1 <- lme(log(serBilir) ~ ns(year, 3) * sex, data = pbc2,
           random = list(id = pdDiag(~ ns(year, 3))))

# the joint model
jointFit <- jm(CoxFit, fm1, time_var = "year", save_random_effects = TRUE)

ppcheck(jointFit)

FF <- function (t, betas, bi, data) {
    sex <- as.numeric(data$sex == "female")
    NS <- ns(t, k = c(0.9911, 3.9863), B = c(0, 14.10579))
    X <- cbind(1, NS, sex, NS * sex)
    Z <- cbind(1, NS)
    eta <- c(X %*% betas[[1]]) + rowSums(Z * bi)
    cbind(eta)
}

ppcheck(jointFit, process = "event", Fforms_fun = FF)
```

`Predictions` *Predictions from Joint Models*

Description

Predict method for object of class `"jm"`.

Usage

```
## S3 method for class 'jm'
predict(object,
    newdata = NULL, newdata2 = NULL, times = NULL,
    process = c("longitudinal", "event"),
    type_pred = c("response", "link"),
    type = c("subject_specific", "mean_subject"),
    control = NULL, ...)

## S3 method for class 'predict_jm'
plot(x, x2 = NULL, subject = 1, outcomes = 1,
  fun_long = NULL, fun_event = NULL, CI_long = TRUE, CI_event = TRUE,
  xlab = "Follow-up Time", ylab_long = NULL, ylab_event = "Cumulative Risk",
  main = "", lwd_long = 2, lwd_event = 2, ylim_event = c(0, 1),
  ylim_long_outcome_range = TRUE,
  col_line_long = "#0000FF",
  col_line_event = c("#FF0000", "#03BF3D", "#8000FF"), pch_points = 16,
  col_points = "blue", cex_points = 1, fill_CI_long = "#0000FF4D",
  fill_CI_event = c("#FF00004D", "#03BF3D4D", "#8000FF4D"), cex_xlab = 1,
  cex_ylab_long = 1, cex_ylab_event = 1, cex_main = 1, cex_axis = 1,
  col_axis = "black", pos_ylab_long = c(0.1, 2, 0.08), bg = "white",
  ...)

## S3 method for class 'jmList'
predict(object,
  weights, newdata = NULL, newdata2 = NULL,
  times = NULL, process = c("longitudinal", "event"),
  type_pred = c("response", "link"),
  type = c("subject_specific", "mean_subject"),
  control = NULL, ...)
```

Arguments

`object`	an object inheriting from class `"jm"` or a list of `"jm"` objects.
`weights`	a numeric vector of model weights.
`newdata, newdata2`	data.frames.
`times`	a numeric vector of future times to calculate predictions.

process — for which process to calculation predictions, for the longitudinal outcomes or the event times.

type — level of predictions; only relevant when type_pred = "longitudinal". Option type = "subject_specific" combines the fixed- and random-effects parts, whereas type = "mean_subject" uses only the fixed effects.

type_pred — type of predictions; options are "response" using the inverse link function in GLMMs, and "link" that correspond to the linear predictor.

control — a named list of control parameters:

all_times logical; if TRUE predictions for the longitudinal outcomes are calculated for all the times given in the times argumet, not only the ones after the last longitudinal measurement..

times_per_id logical; if TRUE the times argument is a vector of times equal to the number of subjects in newdata.

level the level of the credible interval.

return_newdata logical; should predict() return the predictions as extra columns in newdata and newdata2.

use_Y logical; should the longitudinal measurements be used in the posterior of the random effects.

return_mcmc logical; if TRUE the mcmc sample for the predictions is returned. It can be TRUE only in conjuction with return_newdata being FALSE.

n_samples the number of samples to use from the original MCMC sample of object.

n_mcmc the number of Metropolis-Hastings iterations for sampling the random effects per iteration of n_samples; only the last iteration is retained.

parallel character string; what type of parallel computing to use. Options are "snow" (default) and "multicore".

cores how many number of cores to use. If there more than 20 subjects in newdata, parallel computing is invoked with four cores by default. If cores = 1, no parallel computing is used.

seed an integer denoting the seed.

x, x2 — objects returned by predict.jm() with argument return_data set to TRUE.

subject — when multiple subjects are included in the data.frames x and x2, it selects which one to plot. Only a single subject can be plotted each time.

outcomes — when multiple longitudinal outcomes are included in the data.frames x and x2, it selects which ones to plot. A maximum of three outcomes can be plotted each time.

fun_long, fun_event — function to apply to the predictions for the longitudinal and event outcomes, respectively. When multiple longitudinal outcomes are plotted, fun_long can be a list of functions; see examples below.

CI_long, CI_event — logical; should credible interval areas be plotted.

xlab, ylab_long, ylab_event — characture strings or a chracter vector for ylab_long when multiple longitudinal outcomes are considered with the labels for the horizontal axis, and the two vertical axes.

`lwd_long`, `lwd_event`, `col_line_long`, `col_line_event`, `main`, `fill_CI_long`, `fill_CI_event`, `cex_xlab`, `cex_ylab_long`, `cex_ylab_event`, `cex_main`, `cex_axis`, `pch_points`, `col_points`, `cex_points`, `col_axis`, `bg`
: graphical parameters; see `par`.

`pos_ylab_long`
: controls the position of the y-axis labels when multiple longitudinal outcomes are plotted.

`ylim_event`
: the `ylim` for the event outcome.

`ylim_long_outcome_range`
: logical; if `TRUE`, the range of the y-axis spans across the range of the outcome in the data used to fit the model; not only the range of values of the specific subject being plotted.

`...`
: aguments passed to control.

Details

A detailed description of the methodology behind these predictions is given here: https://drizopoulos.github.io/JMbayes2/articles/Dynamic_Predictions.html.

Value

Method `predict()` returns a list or a data.frame (if `return_newdata` was set to `TRUE`) with the predictions.

Method `plot()` produces figures of the predictions from a single subject.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

`jm`

Examples

```
# We fit a multivariate joint model
pbc2.id$status2 <- as.numeric(pbc2.id$status != 'alive')
CoxFit <- coxph(Surv(years, status2) ~ sex, data = pbc2.id)
fm1 <- lme(log(serBilir) ~ ns(year, 3) * sex, data = pbc2,
           random = ~ ns(year, 3) | id, control = lmeControl(opt = 'optim'))
fm2 <- lme(prothrombin ~ ns(year, 2) * sex, data = pbc2,
           random = ~ ns(year, 2) | id, control = lmeControl(opt = 'optim'))
fm3 <- mixed_model(ascites ~ year * sex, data = pbc2,
                   random = ~ year | id, family = binomial())

jointFit <- jm(CoxFit, list(fm1, fm2, fm3), time_var = "year", n_chains = 1L)

# we select the subject for whom we want to calculate predictions
# we use measurements up to follow-up year 3; we also set that the patients
# were alive up to this time point
t0 <- 3
```

```
ND <- pbc2[pbc2$id %in% c(2, 25), ]
ND <- ND[ND$year < t0, ]
ND$status2 <- 0
ND$years <- t0

# predictions for the longitudinal outcomes using newdata
predLong1 <- predict(jointFit, newdata = ND, return_newdata = TRUE)

# predictions for the longitudinal outcomes at future time points
# from year 3 to 10
predLong2 <- predict(jointFit, newdata = ND,
                     times = seq(t0, 10, length.out = 51),
                     return_newdata = TRUE)

# predictions for the event outcome at future time points
# from year 3 to 10
predSurv <- predict(jointFit, newdata = ND, process = "event",
                    times = seq(t0, 10, length.out = 51),
                    return_newdata = TRUE)

plot(predLong1)
# for subject 25, outcomes in reverse order
plot(predLong2, outcomes = 3:1, subject = 25)

# prediction for the event outcome
plot(predSurv)

# combined into one plot, the first longitudinal outcome and cumulative risk
plot(predLong2, predSurv, outcomes = 1)

# the first two longitudinal outcomes
plot(predLong1, predSurv, outcomes = 1:2)

# all three longitudinal outcomes, we display survival probabilities instead
# of cumulative risk, and we transform serum bilirubin to the original scale
plot(predLong2, predSurv, outcomes = 1:3, fun_event = function (x) 1 - x,
     fun_long = list(exp, identity, identity),
     ylab_event = "Survival Probabilities",
     ylab_long = c("Serum Bilirubin", "Prothrombin", "Ascites"),
     pos_ylab_long = c(1.9, 1.9, 0.08))
```

`prothro` *Prednisone versus Placebo in Liver Cirrhosis Patients*

Description

A randomized trial on 488 liver cirrhosis patients.

Format

Two data frames with the following variables.

`id` patients identifier; in total there are 467 patients.

`pro` prothrobin measurements.

`time` for data frame `prothro` the time points at which the prothrobin measurements were taken; for data frame `prothros` the time to death or censoring.

`death` a numeric vector with 0 denoting censoring and 1 death.

`treat` randomized treatment; a factor with levels "placebo" and "prednisone".

References

Andersen, P. K., Borgan, O., Gill, R. D. and Keiding, N. (1993). *Statistical Models Based on Counting Processes*. New York: Springer.

---

rc_setup *Combine Recurring and Terminal Event Data in Long Format*

Description

This function combines two data frames, the recurring-event and terminal-event/competing-risks datasets, into one. Each subject has as many rows in the new data frame as the number of recurrent risk periods plus one for each terminal event/competing risk.

Usage

```
rc_setup(rc_data, trm_data,
    idVar = "id", statusVar = "status",
    startVar = "start", stopVar = "stop",
    trm_censLevel,
    nameStrata = "strata", nameStatus = "status")
```

Arguments

`rc_data`	the data frame containing the recurring-event data with multiple rows per subject.
`trm_data`	the data frame containing the terminal-event/competing-risks data with a single row per subject.
`idVar`	a character scalar denoting the name of the variable in `rc_data` and `trm_data` that identifies the subject/group.
`statusVar`	a character string denoting the name of the variable in `rc_data` and `trm_data` that identifies the status variable. In `rc_data` equals 1 if the subject had an event and 0 otherwise. In `trm_data` equals to the event or censoring level.
`startVar`	a character string denoting the name of the variable in `rc_data` that identifies the starting time for the risk interval.

stopVar	a character string denoting the name of the variable in rc_data and trm_data that identifies the event or censoring time.
trm_censLevel	a character string or a scalar denoting the censoring level in the statusVar variable of trm_data.
nameStrata	a character string denoting the variable that will be added in the long version of data denoting the various causes of event.
nameStatus	a character string denoting the variable that will be added in the long version of data denoting if the subject had an event.

Value

A data frame in the long format with multiple rows per subject.

Author(s)

Pedro Miranda-Afonso <p.mirandaafonso@erasmusmc.nl>

---

slicer *Split Longitudinal and Survival Data into Subject-level Samples*

Description

This function partitions a longitudinal dataset and a survival dataset at the subject level, returning two lists of sliced datasets for model fitting in parallel or series.

Usage

```
slicer(n_slices, id_var, data_long, data_surv, seed = 123L)
```

Arguments

n_slices	an integer scalar giving the number of data slices (subsamples) to create.
id_var	a character scalar with the name of the subject identifier variable in both data_long and data_surv.
data_long	a data frame containing the longitudinal measurements.
data_surv	a data frame containing the survival (time-to-event) information.
seed	an integer seed used to randomize the assignment of subject IDs to slices.

Value

A list with two components:

long a list of length n_slices with class "sliced_data". Each element is a data frame containing the longitudinal rows for the subjects assigned to that slice.

surv a list of length n_slices with class "sliced_data". Each element a data frame containing the survival rows for the subjects assigned to that slice.

Author(s)

Pedro Miranda-Afonso <p.mirandaafonso@erasmusmc.nl>

Examples

```
data(pbc2, package = "JMbayes2")
data(pbc2.id, package = "JMbayes2")

pbc2_slc <- slicer(n_slices = 2, id_var = "id", data_long = pbc2,
                   data_surv = pbc2.id, seed = 123L)
length(pbc2_slc$long) # 2
length(pbc2_slc$surv) # 2
```

---

variogram *Variogram for Longitudinal Data*

Description

It computes the semi-variogram for longitudinal data.

Usage

```
variogram(y, times, id)
```

Arguments

y	a numeric vector of longitudinal responses.
times	a numeric vector of times at which the longitudinal responses were collected.
id	a numeric vector of a factor of subject id numbers.

Value

A list with two components, i.e., svar, a two-column matrix with the time lags and the variogram values, and sigma2 the total variance.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

Examples

```
ind <- aids$patient == 2
yy <- aids$CD4[ind]
tt <- aids$obstime[ind]
ids <- aids$patient[ind]
variogram(yy, tt, ids)
```

Index

∗ **datasets**
aids, 6
pbc2, 26
prothro, 33
∗ **methods**
Accuracy Measures, 2
consensus, 7
crisk_setup, 8
lme, 25
ppcheck, 28
rc_setup, 34
slicer, 35
variogram, 36
∗ **multivariate**
JMbayes2, 24
∗ **package**
JMbayes2, 24

acceleration *(*jm*)*, 9
Accuracy Measures, 2
aids, 6
area *(*jm*)*, 9

calibration_metrics *(*Accuracy Measures*)*, 2
calibration_plot *(*Accuracy Measures*)*, 2
coda_methods.jm, *15*, *25*
coda_methods.jm *(*jm coda Methods*)*, 18
coef *(*jm Methods*)*, 20
coefs *(*jm*)*, 9
compare_jm *(*jm Methods*)*, 20
consensus, 7
coxph *(*lme*)*, 25
create_folds *(*Accuracy Measures*)*, 2
crisk_setup, 8
cumuplot *(*jm coda Methods*)*, 18

Delta *(*jm*)*, 9
densplot *(*jm coda Methods*)*, 18
Dexp *(*jm*)*, 9
Dexpit *(*jm*)*, 9

family *(*jm Methods*)*, 20
fixef *(*jm Methods*)*, 20

gelman_diag *(*jm coda Methods*)*, 18
get_links *(*jm Methods*)*, 20
ggdensityplot *(*jm coda Methods*)*, 18
ggtraceplot *(*jm coda Methods*)*, 18
gridExtra::marrangeGrob(), *19*

jm, *3*, *6*, 9, *20*, *23*, *25*, *32*
jm coda Methods, 17
jm Methods, 20
JMbayes2, 24
JMbayes2-package *(*JMbayes2*)*, 24

lme, 25

methods.jm, *15*, *25*
methods.jm *(*jm Methods*)*, 20
mixed_model *(*lme*)*, 25
model.frame *(*jm Methods*)*, 20
model.matrix *(*jm Methods*)*, 20

pbc2, 26
plot.predict_jm *(*Predictions*)*, 30
poly2 *(*jm*)*, 9
poly3 *(*jm*)*, 9
poly4 *(*jm*)*, 9
ppcheck, 28
predict, *6*
predict.jm *(*Predictions*)*, 30
predict.jmList *(*Predictions*)*, 30
Predictions, 30
prothro, 33
prothros *(*prothro*)*, 33

ranef *(*jm Methods*)*, 20
rc_setup, 34

slicer, *25*, *26*, 35
slope *(*jm*)*, 9

terms *(*jm Methods*)*, 20
traceplot *(*jm coda Methods*)*, 18
tv *(*jm*)*, 9
tvAUC *(*Accuracy Measures*)*, 2
tvBrier *(*Accuracy Measures*)*, 2
tvEPCE *(*Accuracy Measures*)*, 2
tvROC *(*Accuracy Measures*)*, 2

vabs *(*jm*)*, 9
value *(*jm*)*, 9
variogram, 36
velocity *(*jm*)*, 9
vexp *(*jm*)*, 9
vexpit *(*jm*)*, 9
vlog *(*jm*)*, 9
vlog10 *(*jm*)*, 9
vlog2 *(*jm*)*, 9
vsqrt *(*jm*)*, 9