Package ‘JMbayes2’

June 26, 2023

Type Package
Title Extended Joint Models for Longitudinal and Time-to-Event Data
Version 0.4-5
Maintainer Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>
Date 2023-06-23

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

LinkingTo Rcpp, RcppArmadillo

LazyLoad yes
LazyData yes
License GPL (>= 3)

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

NeedsCompilation yes

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

Repository CRAN
Date/Publication 2023-06-26 13:30:05 UTC
**R topics documented:**

Accuracy Measures  ................................................................. 2
aids  ................................................................. 6
crisk_setup  ................................................................. 7
jm  ................................................................. 8
jm coda Methods  ................................................................. 15
jm Methods  ................................................................. 18
JMbayes2  ................................................................. 22
pbc2  ................................................................. 23
Predictions  ................................................................. 24
protho  ................................................................. 28
rc_setup  ................................................................. 28

**Index**  

30

---

**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**

```r
tvROC(object, newdata, Tstart, ...)  
## S3 method for class 'jm'
tvROC(object, newdata, Tstart, Thoriz = NULL,  
       Dt = NULL, ...)

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

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

calibration_plot(object, newdata, Tstart, Thoriz = NULL,  
                  Dt = NULL, df_ns = 3, plot = TRUE, add_density = TRUE,  
                  col = "red", lty = 1, lwd = 1,  
                  col_dens = "grey", xlab = "Predicted Probabilities",  
                  ylab = "Observed Probabilities", main = "", ...)
```

---
Accuracy Measures

calibration_metrics(object, newdata, Tstart, Thoriz = NULL,
  Dt = NULL, df_ns = 3, ...)

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", 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 sub-
  jects 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).

esps 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.
Accuracy Measures

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 assess calibration.
plot logical; should a plot be produced. If FALSE, a list is returned with the observed and predicted probabilities.
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.
id_var character string denoting the name of the subject id variable in data.
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.
Thriz 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.
Thriz 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>
Accuracy Measures

References


See Also

`predict.jm`

Examples

```r
# 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")
```
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


**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**

```r
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**


**Examples**

```r
head(crisk_setup(pbc2.id, "status", "alive"))
```
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, data_Surv = NULL, id_var = NULL, priors = NULL,
  control = NULL, ...)

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

evexpit(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 GLM-Madaptive.

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.

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, shrunk using ridge prior, or shrunk 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, shrunk using ridge prior, or shrunk 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.
parameter of the local penalty term for the association parameters. Only relevant when `penalty_gammas = 'ridge'` or `penalty_gammas = 'horseshoe'`.

`A_t_au_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 `penalty_gammas = 'horseshoe'`.

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

base_hazard_segments  the number of segments to split the follow-up period. Defaults to 10.

diff  the order of the difference used in the penalty matrix for the B-splines for h_0. Defaults to 2.

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  a 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  a logical; if TRUE, the full MCMC results of the random effects will be saved and returned with 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.

knots  a numeric vector with the position of the knots for the B-spline approximation of the log baseline hazard function.

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 denoting the lower limit for calculating the integral.

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.

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 \[\frac{((n_{\text{iter}} - n_{\text{burnin}})/n_{\text{thin}}) \times n_{\text{thin}}, \text{number of individuals}}{\text{}}\], 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 \[\frac{((n_{\text{iter}} - n_{\text{burnin}})/n_{\text{thin}}) \times n_{\text{thin}}, \text{number of individuals}}{\text{}}\], 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"),
        size = 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"),
        size = 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 values are 'all', 'betas', 'sigmas', 'D', 'bs_gammas', 'tau_bs_gammas', 'gammas', 'alphas', 'standard', 'catalog', 'metro', 'pastel', 'beach', 'moonlight', 'goo', 'sunset', 'custom', 'custom_theme', 'NULL', and any other valid theme.
Possible options are 'all', 'betas', 'alphas', 'sigmas', 'D', 'bs_gammas', 'tau_bs_gammas', or 'gammas'.

size 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
# 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('l' = '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('l' = 'black'))

---

## jm Methods

### Various Methods for Standard Generics

### Description

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

### Usage

```r
coef(object, ...)
```

```r
## 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-effects.

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

coe() Extracts estimated fixed effects for the event process from a fitted joint model.
fixe() 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
coe() a list with the elements:

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

fixe() 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:  0.4-5  
Date:  2023-06-23  
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’\textquoteleft\textquoteright\textquotesingle\-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-state 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


See Also

jm, methods.jm, coda_methods.jm

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, transplantation, 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.
albmin 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.
Predictions

Note

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

References


Predictions from Joint Models

Description

Predict method for object of class "jm".

Usage

```R
## S3 method for class 'jm'
predict(object, newdata = NULL, newdata2 = NULL, times = NULL,
times_per_id = FALSE, process = c("longitudinal", "event"),
type_pred = c("response", "link"),
type = c("subject_specific", "mean_subject"),
level = 0.95, return_newdata = FALSE, return_mcmc = FALSE,
n_samples = 200L, n_mcmc = 55L, parallel = c("snow", "multicore"),
cores = NULL, seed = 123L,
...)

## 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_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,
...)
```

```R
# S3 method for class 'jm'
predict(object, newdata = NULL, newdata2 = NULL, times = NULL,
times_per_id = FALSE, process = c("longitudinal", "event"),
type_pred = c("response", "link"),
type = c("subject_specific", "mean_subject"),
level = 0.95, return_newdata = FALSE, return_mcmc = FALSE,
n_samples = 200L, n_mcmc = 55L, parallel = c("snow", "multicore"),
cores = NULL, seed = 123L,
...)

# 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_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,
...)
```
predictions = NULL, times_per_id = FALSE, process = c("longitudinal", "event"),
type_pred = c("response", "link"),
type = c("subject_specific", "mean_subject"),
level = 0.95, return_newdata = FALSE,
return_mcmc = FALSE, n_samples = 200L, n_mcmc = 55L,
parallel = c("snow", " multicore"),
cores = parallelly::availableCores(omit = 1L),
)

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.
times_per_id logical; if TRUE the times argument is a vector of times equal to the number of subjects in newdata.
process for which process to calculation predictions, for the longitudinal outcomes or the event times.
type_pred type of predictions; options are "response" using the inverse link function in GLMMs, and "link" that correspond to the linear predictor.
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.
level the level of the credible interval.
return_newdata logical; should predict() return the predictions as extra columns in newdata and newdata2.
return_mcmc logical; if TRUE the mcmc sample for the predictions is returned. It can be TRUE only in conjunction 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.
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

character strings or a character 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_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.

... extra arguments; currently none is used.

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'))
Predictions

```
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

# 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))
```
**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`: prothrombin measurements.
- `time`: for data frame `prothrob` the time points at which the prothrombin 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".

**Source**


**References**


---

**rc_setup**

Combine Recurring and Terminal Event Data in Long Format

**Description**

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

**Usage**

```r
rc_setup(rc_data, trm_data,
         rc_idVar = "id", rc_statusVar = "status",
         rc_startVar = "start", rc_stopVar = "stop",
         trm_idVar = "id", trm_statusVar = "status",
         trm_stopVar = "stop",
         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 data with a single row per subject.
- **rc_idVar**: a character string denoting the name of the variable in `rc_data` that identifies the subject.
- **rc_statusVar**: a character string denoting the name of the variable in `rc_data` that identifies the status variable which equals 1 if the subject had an event and 0 otherwise.
- **rc_startVar**: a character string denoting the name of the variable in `rc_data` that identifies the starting time for the interval.
- **rc_stopVar**: a character string denoting the name of the variable in `rc_data` that identifies the stopping time for the interval.
- **trm_idVar**: a character string denoting the name of the variable in `trm_data` that identifies the subject.
- **trm_statusVar**: a character string denoting the name of the variable in `trm_data` that identifies the status variable which equals 1 if the subject had the event and 0 otherwise.
- **trm_stopVar**: a character string denoting the name of the variable in `trm_data` that identifies the follow up time.
- **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>
Index

* datasets
  aids, 6
  pbc2, 23
  prothro, 28
* methods
  Accuracy Measures, 2
  crisk_setup, 7
  rc_setup, 28
* multivariate
  JMbayes2, 22
* package
  JMbayes2, 22
  gridExtra::marrangeGrob(), 17
  jm, 3, 5, 8, 17, 21, 23, 26
  jm coda Methods, 15
  jm Methods, 18
  JMbayes2, 22
  JMbayes2-package (JMbayes2), 22
  methods.jm, 13, 23
  methods.jm (jm Methods), 18
  model.frame (jm Methods), 18
  model.matrix (jm Methods), 18
  pbc2, 23
  plot.predict_jm (Predictions), 24
  poly2 (jm), 8
  poly3 (jm), 8
  poly4 (jm), 8
  predict, 5
  predict.jm (Predictions), 24
  predict.jmList (Predictions), 24
  Predictions, 24
  prothro, 28
  prothros (prothro), 28
  ranef (jm Methods), 18
  rc_setup, 28
  slope (jm), 8
  terms (jm Methods), 18
  traceplot (jm coda Methods), 15
  tv (jm), 8
  tvAUC (Accuracy Measures), 2
  tvBrier (Accuracy Measures), 2
  tvEPCE (Accuracy Measures), 2
  tvROC (Accuracy Measures), 2
  vabs (jm), 8
  value (jm), 8
  velocity (jm), 8
vexp(jm), 8
vexpit(jm), 8
vlog(jm), 8
vlog10(jm), 8
vlog2(jm), 8
vsqrt(jm), 8