Package 'JMbayes'

July 21, 2025

Title Joint Modeling of Longitudinal and Time-to-Event Data under a Bayesian Approach

Version 0.8-85

Date 2020-01-08

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

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

Description Shared parameter models for the joint modeling of longitudinal and time-to-event data using MCMC; Dimitris Rizopoulos (2016) <doi:10.18637/jss.v072.i07>.

Depends nlme, survival, doParallel, rstan

Imports MASS, foreach, Rcpp, jagsUI, xtable, shiny, splines, Hmisc

SystemRequirements JAGS (http://mcmc-jags.sourceforge.net)

LinkingTo Rcpp, RcppArmadillo

LazyLoad yes

LazyData yes

License GPL (>= 2)

URL https://github.com/drizopoulos/JMbayes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-01-09 01:30:02 UTC

Contents

`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.

---

anova *Anova Method for Fitted Joint Models*

Description

Comparison of (non)nested joint models using information criteria.

Usage

```
## S3 method for class 'JMbayes'
anova(object, ...)
```

Arguments

`object, ...` objects inheriting from class `JMbayes`.

Value

A data frame with rows the different models, and columns the number or parameters in each model, the the log pseudo marginal likelihood value, the deviance information criterion value, and the pD value.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

`jointModelBayes`

Examples

```
## Not run:
# composite event indicator
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# linear mixed model with natural cubic splines for the time
# effect
lmeFit.pbc1 <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
                   random = ~ ns(year, 2) | id, method = "ML")

# Cox regression model with baseline covariates
coxFit.pbc1 <- coxph(Surv(years, status2) ~ drug * age, data = pbc2.id, x = TRUE)

# the standard joint model fit with only the m_i(t) term in
# the linear predictor of the survival submodel
jointFit.pbc1 <- jointModelBayes(lmeFit.pbc1, coxFit.pbc1, timeVar = "year")

# we include the time-dependent slopes term
dForm <- list(fixed = ~ 0 + dns(year, 2), random = ~ 0 + dns(year, 2),
              indFixed = 2:3, indRandom = 2:3)

jointFit.pbc2 <- update(jointFit.pbc1, param = "td-both", extraForm = dForm)

# we include the cumulative effect of the marker
iForm <- list(fixed = ~ 0 + year + ins(year, 2), random = ~ 0 + year + ins(year, 2),
              indFixed = 1:3, indRandom = 1:3)

jointFit.pbc3 <- update(jointFit.pbc1, param = "td-extra", extraForm = iForm)


# we compare the three models
anova(jointFit.pbc1, jointFit.pbc2, jointFit.pbc3)

## End(Not run)
```

---

aucJM *Time-Dependent ROCs and AUCs for Joint Models*

Description

Using the available longitudinal information up to a starting time point, this function computes an estimate of the ROC and the AUC at a horizon time point based on joint models.

Usage

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

## S3 method for class 'JMbayes'
```

```
aucJM(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, idVar = "id", simulate = FALSE, M = 100, ...)

## S3 method for class 'mvJMbayes'
aucJM(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, idVar = "id", M = 100, ...)

rocJM(object, newdata, Tstart, ...)

## S3 method for class 'JMbayes'
rocJM(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, idVar = "id", simulate = FALSE, M = 100, ...)

## S3 method for class 'mvJMbayes'
rocJM(object, newdata, Tstart, Thoriz = NULL,
    Dt = NULL, idVar = "id", M = 100, ...)

predict_eventTime(object, newdata, cut_points, ...)

## S3 method for class 'mvJMbayes'
predict_eventTime(object, newdata, cut_points,
    idVar = "id", M = 500L, low_percentile = 0.025, ...)

find_thresholds(object, newdata, Dt, ...)

## S3 method for class 'mvJMbayes'
find_thresholds(object, newdata, Dt, idVar = "id",
    M = 200L, variability_threshold = NULL,
    n_cores =  max(1, parallel::detectCores() - 2), ...)
```

Arguments

object	an object inheriting from class JMbayes or mvJMbayes.
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 model (using lme()) and the survival model (using coxph()) that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the different subjects (see also argument idVar).
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.

idVar	the name of the variable in newdata that identifies the different subjects.
simulate	logical; if TRUE, a Monte Carlo approach is used to estimate survival probabilities. If FALSE, a first order estimator is used instead. See survfitJM for mote details.
M	a numeric scalar denoting the number of Monte Carlo samples; see survfitJM for mote details.
cut_points	a numeric matrix with first column time-points followed by other columns of optimal cut-points from an ROC curve.
variability_threshold	numeric value denoting the treshold in the spread of the posterior distribution calculated from the 2.5% percentile to the median. Default is the 25% percentile of the event times distribution.
low_percentile	a numeric value indicating the percentile based on which it will be judged whether the spread of the posterior predictive distribution is too large.
n_cores	an integer indicating the number of cores to use for parallel computing.
...	additional arguments; currently none is used.

Details

Based on a fitted joint model (represented by object) and using the data supplied in argument newdata, this function computes the following estimate of the AUC:

$$\text{AUC}(t, \Delta t) = \Pr\big[\pi_i(t + \Delta t \mid t) < \pi_j(t + \Delta t \mid t) \mid \{T_i^* \in (t, t + \Delta t]\} \cap \{T_j^* > t + \Delta t\}\big],$$

with i and j denote a randomly selected pair of subjects, and $\pi_i(t + \Delta t \mid t)$ and $\pi_j(t + \Delta t \mid t)$ denote the conditional survival probabilities calculated by survfitJM for these two subjects, for different time windows Δt specified by argument Dt using the longitudinal information recorded up to time t = Tstart.

The estimate of $\text{AUC}(t, \Delta t)$ provided by aucJM() is in the spirit of Harrell's c-index, that is for the comparable subjects (i.e., the ones whose observed event times can be ordered), we compare their dynamic survival probabilities calculated by survfitJM. For the subjects who due to censoring we do not know if they are comparable, they contribute in the AUC with the probability that they would have been comparable.

Value

A list of class aucJM 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.

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.

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.

See Also

survfitJM, dynCJM, jointModelBayes

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 3),
    random = list(id = pdDiag(form = ~ ns(year, 3))), data = pbc2)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

# AUC using data up to year 5 with horizon at year 8
aucJM(jointFit, pbc2, Tstart = 5, Thoriz = 8)

plot(rocJM(jointFit, pbc2, Tstart = 5, Thoriz = 8))

## End(Not run)
```

bma.combine *Combines Predictions for Bayesian Model Averaging*

Description

Combines estimated survival probabilities or predictions for longitudinal responses.

Usage

```
bma.combine(..., JMlis = NULL, weights = NULL)
```

Arguments

...	objects inheriting from class survfit.JMbayes or predict.JMbayes.
JMlis	a list of survfit.JMbayes or predict.JMbayes objects.
weights	a numeric vector of weights to be applied in each object.

Value

an object of class survfit.JMbayes or predict.JMbayes.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

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., Hatfield, L., Carlin, B. and Takkenberg, J. (2014). Combining dynamic predictions from joint models for longitudinal and time-to-event data using Bayesian model averaging. *Journal of the American Statistical Association* **109**, 1385–1397.

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit two joint models using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function; the first one with the current value parameterization
# and the other with the shared random effects parameterization
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
              random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
```

```
jointFit1 <- jointModelBayes(lmeFit, survFit, timeVar = "year")
jointFit2 <- update(jointFit1, param = "shared-RE")

# we compute survival probabilities for Subject 2 with
# different weights
ND <- pbc2[pbc2$id == 2, ] # the data of Subject 2
survPreds1 <- survfitJM(jointFit1, newdata = ND, weight = 0.4)
survPreds2 <- survfitJM(jointFit2, newdata = ND, weight = 0.6)

survPreds.bma <- bma.combine(survPreds1, survPreds2)
survPreds.bma
plot(survPreds.bma)

## End(Not run)
```

---

coef *Estimated Coefficients and Confidence Intervals for Joint Models*

Description

Extracts estimated coefficients and confidence intervals from fitted joint models.

Usage

```
## S3 method for class 'JMbayes'
coef(object, process = c("Longitudinal", "Event"), ...)

## S3 method for class 'JMbayes'
fixef(object, process = c("Longitudinal", "Event"), ...)

## S3 method for class 'JMbayes'
confint(object, parm = c("all", "Longitudinal", "Event"), ...)
```

Arguments

object	an object inheriting from class JMbayes.
process	for which submodel (i.e., linear mixed model or survival model) to extract the estimated coefficients.
parm	for which submodel (i.e., linear mixed model or survival model) to extract credible intervals.
...	additional arguments; currently none is used.

Details

When process = "Event" both methods return the same output. However, for process = "Longitudinal", the coef() method returns the subject-specific coefficients, whereas fixef() only the fixed effects.

Value

A numeric vector or a matrix of the estimated parameters or confidence intervals for the fitted model.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

ranef.JMbayes, jointModelBayes

Examples

```
## Not run:
# linear mixed model fit
fitLME <- lme(sqrt(CD4) ~ obstime * drug - drug,
    random = ~ 1 | patient, data = aids)
# cox model fit
fitCOX <- coxph(Surv(Time, death) ~ drug, data = aids.id, x = TRUE)

# joint model fit
fitJOINT <- jointModelBayes(fitLME, fitCOX,
    timeVar = "obstime")

# fixed effects for the longitudinal process
fixef(fitJOINT)

# fixed effects + random effects estimates for the longitudinal
# process
coef(fitJOINT)

# fixed effects for the event process
fixef(fitJOINT, process = "Event")
coef(fitJOINT, process = "Event")

## End(Not run)
```

---

cvDCL *Dynamic Information*

Description

Using the available longitudinal information up to a starting time point, this function computes an estimate of the cross-entropy function based on joint models.

Usage

```
cvDCL(object, newdata, Tstart, idVar = "id", M = 300L, seed = 123L)
```

Arguments

object	an object inheriting from class JMBayes.
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 model (using lme()) and the survival model (using coxph()) that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the different subjects (see also argument idVar).
Tstart	a numeric scalar indicating at which time to compute the cross-entropy.
idVar	the name of the variable in newdata that identifies the different subjects.
M	a numeric scalar denoting the number of Monte Carlo samples.
seed	a numeric scalar

Details

This function calculates an estimate of the cross-entropy at any time point t (given in Tstart) that can be used to identify the joint model that best predicts future event giver survival up to t.

Value

A list of class aucJM 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.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

survfitJM, dynCJM, jointModelBayes

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
```

```
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 3),
    random = list(id = pdDiag(form = ~ ns(year, 3))), data = pbc2)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

cvDCL(jointFit, Tstart = 5)

## End(Not run)
```

DerivSplines *Derivatives and Integrals of B-splines and Natural Cubic splines*

Description

Numerical derivatives and integrals of functions bs() and ns() at their first argument.

Usage

```
dns(x, df = NULL, knots = NULL, intercept = FALSE,
    Boundary.knots = range(x), eps = 1e-03)

dbs(x, df = NULL, knots = NULL, intercept = FALSE,
    Boundary.knots = range(x), eps = 1e-03)

ins(x, df = NULL, knots = NULL, intercept = FALSE,
    Boundary.knots = range(x), from = 0, weight.fun = NULL,
    integrand.fun = NULL, ...)

ibs(x, df = NULL, knots = NULL, intercept = FALSE,
    Boundary.knots = range(x), from = 0, weight.fun = NULL,
    integrand.fun = NULL, ...)
```

Arguments

x, df, knots, intercept, Boundary.knots	see the help pages of functions ns() and bs().
eps	a numeric scalar denoting the step length for the central difference approximation, which calculates the derivative.
from	a numeric scalar denoting the lower limit of the integral.
weight.fun	a function to be applied as weights.
integrand.fun	a function to be applied in the integrand.
...	extra arguments passed to weight.fun.

Value

an object of class dns, dbs, ins or ibs.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

---

dynCJM *A Dynamic Discrimination Index for Joint Models*

Description

This function computes a dynamic discrimination index for joint models based on a weighted average of time-dependent AUCs.

Usage

```
dynCJM(object, newdata, Dt, ...)

## S3 method for class 'JMbayes'
dynCJM(object, newdata, Dt, idVar = "id", t.max = NULL,
    simulate = FALSE, M = 100, weightFun = NULL, ...)
```

Arguments

object	an object inheriting from class JMBayes.
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 model (using lme()) and the survival model (using coxph()) that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the different subjects (see also argument idVar).
Dt	a numeric scalar denoting the time frame within which the occurrence of events is of interest.
idVar	the name of the variable in newdata that identifies the different subjects.
t.max	a numeric scalar denoting the time maximum follow-up time up to which the dynamic discrimination index is to be calculated. If NULL, it is set equal to max(Time) + 1e-05 where Time denotes the observed event times.
simulate	logical; if TRUE, a Monte Carlo approach is used to estimate survival probabilities. If FALSE, a first order estimator is used instead. See survfitJM for mote details.
M	a numeric scalar denoting the number of Monte Carlo samples; see survfitJM for mote details.
weightFun	a function of two arguments the first denoting time and the second the length of the time frame of interest, i.e., Dt.
...	additional arguments; currently none is used.

Details

(**Note:** The following contain some math formulas, which are better viewed in the pdf version of the manual accessible at https://cran.r-project.org/package=JMbayes.)

Function dynC computes the following discrimination index

$$\mathrm{C}_{dyn}^{\Delta t} = \int_0^{t_{max}} \mathrm{AUC}(t, \Delta t)\, \Pr\{\mathcal{E}(t, \Delta t)\}\, dt \Big/ \int_0^{t_{max}} \Pr\{\mathcal{E}(t, \Delta t)\}\, dt,$$

where

$$\mathrm{AUC}(t, \Delta t) = \Pr\big[\pi_i(t + \Delta t \mid t) < \pi_j(t + \Delta t \mid t) \mid \{T_i^* \in (t, t + \Delta t]\} \cap \{T_j^* > t + \Delta t\}\big],$$

and

$$\mathcal{E}(t, \Delta t) = \big[\{T_i^* \in (t, t + \Delta t]\} \cap \{T_j^* > t + \Delta t\}\big],$$

with i and j denote a randomly selected pair subjects, and $\pi_i(t+\Delta t \mid t)$ and $\pi_j(t+\Delta t \mid t)$ denote the conditional survival probabilities calculated by survfitJM for these two subjects, for different time windows Δt specified by argument Dt. The upper limit of integral in specified by argument t.max. The integrals in the numerator and denominator are approximated using a 15-point Gauss-Kronrod quadrature rule.

Value

A list of class dynCJM with components:

dynC	a numeric scalar denoting the dynamic discrimination index.
times	a numeric vector of time points at which the AUC was calculated.
AUCs	a numeric vector of the estimated AUCs at the aforementioned time points.
weights	a numeric vector of the estimated weights at the aforementioned time points.
t.max	a copy of the t.max argument.
Dt	a copy of the Dt argument.
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.

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.

See Also

survfitJM, aucJM, jointModelBayes

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
              random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

# dynamic discrimination index up to year 10 using a two-year interval
dynCJM(jointFit, pbc2, Dt = 2, t.max = 10)

## End(Not run)
```

---

dynInfo *Dynamic Information of an Extra Longitudinal Measurement*

Description

Using the available longitudinal information up to a particular time point, this function computes an estimate of the information we again by obtaining an extra longitudinal measurement.

Usage

```
dynInfo(object, newdata, Dt, K = 5, M = 500, idVar = "id",
    simulateFun = function (eta, scale) rnorm(length(eta), eta, scale),
    seed = 1L)
```

Arguments

object	an object inheriting from class JMBayes.
newdata	a data frame that contains the longitudinal and covariate information for the subject for whom we wish to plan the next measurement. 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 model (using lme()) and the survival model (using coxph() that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the subject (see also argument idVar).
Dt	numeric scalar denoting the length of the time interval to search for the optimal time point of the next measurement, i.e., the interval is $(t, t + Deltat]$ with $Deltat$ given by Dt.
K	numeric scalar denoting the number of time points to cosider in the interval $(t, t + Deltat]$.
idVar	the name of the variable in newdata that identifies the subject.
simulateFun	a function based on which longitudinal measurement can be simulated. This should have as a main argument the variable eta that denotes the subject-specific linear predictor from the mixed model, and possibly a scale parameter.
M	a numeric scalar denoting the number of Monte Carlo samples.
seed	a numeric scalar

Details

This functions computes the following posterior predictive distribution

$$E_Y[E_{T^*|Y}(\log p(T_j^* \mid T_j^* > u, \{Y_j(t), y_j(u)\}, D_n))],$$

where T_j^* denotes the time-to-event for subject j for whom we wish to plan the next visit, $Y_j(t)$ the available longitudinal measurements of this subject up to time t, $y_j(u)$ the future longitudinal measurement we wish to plan at time $u > t$, and D_n the data set that was used to fit the joint model.

Value

A list with components:

summary	a numeric matrix with first column the time points at which the longitudinal measurement is hypothetically taken, second column the estimated information we gain by obtaining the measurement, and third column the estimated cumulative risk of an event up to the particular time point denoted in the first column.
full.results	a numeric matrix with columns representing the time points, rows the Monte Carlo samples, and entries the value of log posterio predictive density.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

survfitJM, jointModelBayes

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 3),
    random = list(id = pdDiag(form = ~ ns(year, 3))), data = pbc2)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

dynInfo(jointFit, newdata = pbc2[pbc2$id == 2, ], Dt = 5)[[1]]

## End(Not run)
```

---

fitted & residuals *Fitted Values and Residuals for Joint Models*

Description

Calculates fitted values for joint models.

Usage

```
## S3 method for class 'JMbayes'
fitted(object,
    process = c("Longitudinal", "longitudinal", "Event", "event"),
    type = c("Marginal", "marginal", "Subject", "subject"), nullY = FALSE, ...)

## S3 method for class 'JMbayes'
residuals(object,
    process = c("Longitudinal", "longitudinal", "Event", "event"),
    type = c("Marginal", "marginal", "Subject", "subject",
             "Martingale", "martingale", "nullMartingale", "nullmartingale"),
    standardized = FALSE, ...)
```

Arguments

object	an object inheriting from class jointModel.
process	for which model (i.e., linear mixed model or survival model) to calculate fitted values or residuals.

type	what type of fitted values or residuals to calculate. See **Details**.
nullY	logical; if TRUE the association parameters that connect the longitudinal and event time process are set to zero.
standardized	logical; if TRUE standardized residuals are calculated.
...	additional arguments; currently none is used.

Details

For process = "Longitudinal", let X denote the design matrix for the fixed effects β, and Z the design matrix for the random effects b. Then for type = "Marginal" the fitted values are $X\hat{\beta}$, whereas for type = "Subject" they are $X\hat{\beta}+Z\hat{b}$, where $\hat{\beta}$ and $\hat{b}$ denote the corresponding posterior means for the fixed and random effects. The corresponding residuals are calculated by subtracting the fitted values from the observed data y. If type = "Subject" and standardized = TRUE, the residuals are divided by the estimated residual standard error.

For process = "Event" function fitted() calculates the cumulative hazard function at each time point a longitudinal measurement has been recorded. If nullY = TRUE, then the cumulative hazard is calculated without the contribution of the longitudinal process. Function residuals() calculates the martingales residuals or the martingale residuals without the contribution of the longitudinal process when type = "nullMartingale".

Value

a numeric vector of fitted values or residuals.

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.

Examples

```
## Not run:
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
              random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ 1, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

fitted(jointFit, process = "Event")
residuals(jointFit, type = "Subject", standardized = TRUE)

## End(Not run)
```

gt *The Generalized Student's t Distribution*

Description

Density, distribution function, quantile function and random generation for the generalized Student's t distribution.

Usage

```
dgt(x, mu = 0, sigma = 1, df = stop("no df arg"), log = FALSE)

pgt(q, mu = 0, sigma = 1, df = stop("no df arg"))

qgt(p, mu = 0, sigma = 1, df = stop("no df arg"))

rgt(n, mu = 0, sigma = 1, df = stop("no df arg"))
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	a numeric scalar denoting the number of observations.
mu	a vector of means.
sigma	a vector of standard deviations.
log	logical; if TRUE the density is computed in the log scale.
df	a numeric scalar denoting the degrees of freedom.

Value

dgt gives the density, pgt gives the distribution function, qgt gives the quantile function, and rgt generates random deviates.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

Examples

```
## Not run:
x <- rnorm(10, mean = 10, sd = 3)
dgt(x, mu = 10, sigma = 3, df = 4)
rgt(10, mu = 10, sigma = 3, df = 4)

## End(Not run)
```

IndvPred_lme *Individualized Predictions from Linear Mixed Models*

Description

Calculates subject-specific predictions for new subjects from a linear mixed model.

Usage

```
IndvPred_lme(lmeObject, newdata, timeVar, times = NULL, M = 200L,
    interval = c("confidence", "prediction"), all_times = FALSE,
    level = 0.95, return_data = FALSE, seed = 1L)

extract_lmeComponents(lmeObject, timeVar)
```

Arguments

lmeObject	an object inheriting from class lme or class lmeComponents.
newdata	a data frame in which to look for variables with which to predict.
timeVar	a character string specifying the time variable in the linear mixed model.
interval	a character string indicating what type of intervals should be computed.
all_times	logical; should predictions be calculated at all times or only at the ones that are after the last observed time of each subject.
level	a numeric scalar denoting the tolerance/confidence level.
times	a numeric vector denoting the time points for which we wish to compute the subject-specific predictions after the last available measurement provided in newdata. Default is a sequence of 100 equally spaced time points from the smallest to the largest follow-up time of all subjects.
M	numeric scalar denoting the number of Monte Carlo samples. See **Details**.
return_data	logical; if TRUE the data frame supplied in newdata is returned augmented with the outputs of the function.
seed	numeric scalar, the random seed used to produce the results.

Value

If return_data = TRUE, a the data frame newdata with extra rows for the time points at which predictions were calculated, and extra columns with the predictions and the limits of the pointwise confidence intervals.

If return_data = FALSE, a list with components

times_to_pred	time points at which predictions were calculated.
predicted_y	the predictions.
low	the lower limits of the pointwise confidence intervals.
upp	the upper limits of the pointwise confidence intervals.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

predict.JMbayes

Examples

```
## Not run:
# linear mixed model fit
fitLME <- lme(log(serBilir) ~ drug * ns(year, 2), data = subset(pbc2, id != 2),
        random = ~ ns(year, 2) | id)

DF <- IndvPred_lme(fitLME, newdata = subset(pbc2, id == 2), timeVar = "year",
    M = 500, return_data = TRUE)

require(lattice)
xyplot(pred + low + upp ~ year | id, data = DF,
    type = "l", col = c(2,1,1), lty = c(1,2,2), lwd = 2,
    ylab = "Average log serum Bilirubin")

# extract_lmeComponents() extract the required components from the lme object
# that are required to calculate the predictions; this is a light weight version of
# the object, e.g.,
fitLME_light <- extract_lmeComponents(fitLME, timeVar = "year")

DF <- IndvPred_lme(fitLME_light, newdata = subset(pbc2, id == 2), timeVar = "year",
    M = 500, return_data = TRUE)

## End(Not run)
```

JMbayes — *Joint Modeling of Longitudinal and Time-to-Event Data in R under a Bayesian Approach*

Description

This package fits shared parameter models for the joint modeling of normal longitudinal responses and event times under a Bayesian approach. Various options for the survival model and the association structure are provided.

Details

Package:	JMbayes
Type:	Package
Version:	0.8-85
Date:	2020-01-08

License: GPL (>=2)

The package has a single model-fitting function called `jointModelBayes`, which accepts as main arguments a linear mixed effects object fit returned by function `lme()` of package **nlme**, and a Cox model object fit returned by function `coxph()` of package **survival**. The `survMod` argument of specifies the type of survival submodel to be fitted; available options are a relative risk model with a Weibull baseline hazard (default) and a relative risk model with a B-spline approximation of the log baseline risk function. In addition, the `param` specifies the association structure between the longitudinal and survival processes; available options are: `"td-value"` which is the classic formulation used in Wulfsohn and Tsiatis (1997); `"td-extra"` which is a user-defined, possibly time-dependent, term based on the specification of the `extraForm` argument of `jointModelBayes`. This could be used to include terms, such as the time-dependent slope (i.e., the derivative of the subject-specific linear predictor of the linear mixed model) and the time-dependent cumulative effect (i.e., the integral of the subject-specific linear predictor of the linear mixed model); `"td-both"` which is the combination of the previous two parameterizations, i.e., the current value and the user-specified terms are included in the linear predictor of the relative risk model; and `"shared-RE"` where only the random effects of the linear mixed model are included in the linear predictor of the survival submodel.

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

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

References

Guo, X. and Carlin, B. (2004) Separate and joint modeling of longitudinal and event time data using standard computer packages. *The American Statistician* **54**, 16–24.

Henderson, R., Diggle, P. and Dobson, A. (2000) Joint modelling of longitudinal measurements and event time data. *Biostatistics* **1**, 465–480.

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. and Ghosh, P. (2011) A Bayesian semiparametric multivariate joint model for multiple longitudinal outcomes and a time-to-event. *Statistics in Medicine* **30**, 1366–1380.

Rizopoulos, D., Verbeke, G. and Molenberghs, G. (2010) Multiple-imputation-based residuals and diagnostic plots for joint models of longitudinal and survival outcomes. *Biometrics* **66**, 20–29.

Tsiatis, A. and Davidian, M. (2004) Joint modeling of longitudinal and time-to-event data: an overview. *Statistica Sinica* **14**, 809–834.

Wulfsohn, M. and Tsiatis, A. (1997) A joint model for survival and longitudinal data measured with error. *Biometrics* **53**, 330–339.

See Also

jointModelBayes, survfitJM, aucJM, dynCJM, prederrJM, predict.JMbayes, logLik.JMbayes

---

JMbayesObject *Fitted JMbayes Object*

Description

An object returned by the jointModelBayes function, inheriting from class JMbayes and representing a fitted joint model for longitudinal and time-to-event data. Objects of this class have methods for the generic functions coef, confint, fixed.effects, logLik, plot, print, random.effects, summary, and vcov.

Value

The following components must be included in a legitimate JMbayes object.

mcmc	a list with the MCMC samples for each parameter (except from the random effects if control argument keepRE is FALSE).
postMeans	a list with posterior means.
postModes	a list with posterior modes calculated using kernel desnisty estimation.
postVarsRE	a list with the posterior variance-covariance matrix for the random effects of each subject.
StErr	a list with posterior standard errors.
EffectiveSize	a list with effective sample sizes.
StDev	a list with posterior standard deviations.
CIs	a list with 95% credible intervals.
vcov	the variance-covariance matrix of the model's parameters based.
pD	the pD value.
DIC	the deviance information criterion value.
CPO	the conditional predictive ordinate value.
LPML	the log pseudo marginal likelihood value.
time	the time used to fit the model.
scales	a list with scaling constants in the Metropolis algorithm.
Covs	a list with the covariance matrices of the proposals in the Metropolis algorithm.
acceptRates	a list of acceptance rates.
x	a list with the design matrices for the longitudinal and event processes.
y	a list with the response vectors for the longitudinal and event processes.

Data	a list of data frames with the data used to fit the models.
Terms	a list of terms objects for the various parts of the joint model.
Funs	a list of functions used for the various parts of the joint model.
Forms	a list of formulas for the two submodels.
timeVar	the value of the timeVar argument
control	the value of the control argument.
densLongCheck	a logical indicating whether a scale parameter is required in the longitudinal submodel.
param	the value of the param argument.
priors	a list with the specification of the prior distributions for the model's parameters. This has the same components as the priors argument of the jointModelBayes function.
baseHaz	the value of the baseHaz argument.
df.RE	the value of the df.RE argument.
call	the matched call.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

jointModelBayes

---

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

Description

Fits shared parameter joint models for longitudinal and survival outcomes under a Bayesian approach.

Usage

```
jointModelBayes(lmeObject, survObject, timeVar,
    param = c("td-value", "td-extra", "td-both", "shared-betasRE", "shared-RE"),
    extraForm = NULL, baseHaz = c("P-splines", "regression-splines"),
    transFun = NULL, densLong = NULL, lag = 0, df.RE = NULL,
    estimateWeightFun = FALSE, weightFun = NULL, init = NULL,
    priors = NULL, scales = NULL, control = list(), ...)
```

Arguments

`lmeObject`	an object of class 'lme' fitted by function `lme()` from package **nlme** or by function `glmmPQL()` from package **MASS**.
`survObject`	an object of class 'coxph' fitted by function `coxph()` from package **survival**.
`timeVar`	a character string indicating the time variable in the mixed effects model.
`param`	a character string specifying the type of association structure between the longitudinal and survival processes. See **Details**.
`extraForm`	a list with components `fixed` a formula representing the fixed-effects part of the user-defined term, `indFixed` a numeric vector indicating which fixed effects of `lmeObject` are involved in the user-defined term, `random` a formula representing the random-effects part of the user-defined term, and `indRamdom` a numeric vector indicating which random effects of `lmeObject` are involved in the user-defined term. Required only when `param = "td-extra"` or `param = "td-both"`. See **Examples**.
`baseHaz`	a character string specifying the type of the baseline hazard function. See **Details**.
`transFun`	a function or a named list with elements `value` and `extra` which should be functions. In either case the functions should always have two arguments, namely `x` and `data` (even when the second one is not needed). The purpose is to transform the `value` and/or `extra`, for example including an interaction term, a nonlinear function, etc.
`densLong`	a function with arguments `y`, `eta.y`, `scale`, `log`, and `data` that calculates the density of the longitudinal outcome. `y` denotes the longitudinal responses, `eta.y` the linear predictor that includes the fixed and random effects, `scale` a possible scale parameter (e.g., the measurement error standard deviation), `log` a logical argument that controls whether the density should be calculated in the log scale, and `data` a data frame which may be used to extract variables used in the definition of the density function (e.g., a censoring indicator for left censored longitudinal data).
`lag`	a numeric scalar denoting a lag effect in the time-dependent covariate represented by the mixed model; default is 0.
`df.RE`	a numeric scalar denoting the number of degrees of freedom for the Student's-t random-effects distribution. If `NULL` the random effects are assumed to have a multivariate normal distribution.
`estimateWeightFun`	logical; experimental, not in use yet.
`weightFun`	a weight function; experimental, not in use yet.
`init`	a named list of user-specified initial values: **betas** the vector of fixed effects for the linear mixed effects model. **tau** the precision parameter from the linear mixed effects model (i.e., $\tau = 1/\sigma^2$ with σ denoting the error terms standard deviation). **invD** the inverse variance-covariance matrix of the random effects. **b** a matrix of random effects values. **gammas** the vector of baseline covariates for the survival model.

alphas the association parameter(s).

Dalphas the association parameter for the true slopes formulation.

Bs.gammas the vector of spline coefficients for the spline-approximated baseline risk function.

When this list of initial values does not contain some of these components or contains components not of the appropriate length, then the default initial values are used instead.

`priors` a named list of user-specified prior parameters:

priorMean.betas the prior mean vector of the normal prior for the fixed effects of the linear mixed effects model.

priorTau.betas the prior precision matrix of the normal prior for the fixed effects of the linear mixed effects model.

priorA.tau the prior shape parameter of the Gamma prior for the precision parameter of the linear mixed effects model.

priorB.tau the prior rate parameter of the Gamma prior for the precision parameter of the linear mixed effects model.

priorMean.gammas the prior mean vector of the normal prior for the regression coefficients of the survival model.

priorTau.gammas the prior precision matrix of the normal prior for the regression coefficients of the survival model.

priorMean.alphas the prior mean vector of the normal prior for the association parameter in the survival model.

priorTau.alphas the prior precision matrix of the normal prior for the association parameter in the survival model.

priorMean.Dalphas the prior mean vector of the normal prior for the slope association parameter in the survival model.

priorTau.Dalphas the prior precision matrix of the normal prior for the slope association parameter in the survival model.

priorMean.Bs.gammas the prior mean vector of the normal prior for the spline coefficients of the baseline risk function.

priorTau.Bs.gammas the prior precision matrix of the normal prior for the spline coefficients of the baseline risk function.

priorA.tauBs the prior shape parameter of the Gamma prior for the precision parameter of the penalty term when `baseHaz = "P-splines"`.

priorB.tauBs the prior rate parameter of the Gamma prior for the precision parameter of the penal term when `baseHaz = "P-splines"`.

priorR.D the prior precision matrix of the Wishart prior for the precision matrix of the random effects.

priorK.D the degrees of freedom of the Wishart prior for the precision matrix of the random effects.

`scales` a named list with names as in `init` specifying scaling constants for the proposal distributions in the Metropolis algorithm.

`control` a list of control values with components:

adapt logical default `FALSE`; should adaptive Metropolis be used. Currently experimental.

n.iter integer specifying the total number of iterations; default is 20000.

n.burnin integer specifying how many of `n.iter` to discard as burn-in; default is 3000.

n.thin integer specifying the thinning of the chains; default is to set `n.thin` such that 2000 samples are kept.

n.adapt integer specifying the number of iterations to use for adaptation; default is 3000.

keepRE logical; if `TRUE` the MCMC samples for the random effect are kept in the output object.

priorVar integer used as the prior precision in the normal prior for the fixed effects, the regression coefficients of the survival submodel, the association parameters, the extra association parameters, and in the spline coefficients; default is 100.

knots a numeric vector of knots positions for the spline approximation of the log baseline risk function; default is `NULL`, which means that the knots are calculated based on the percentiles of the observed event times.

ObsTimes.knots logical; if `TRUE` (default), the knots are set using the percentiles of the observed event times (i.e., including both true events and censored observations). If `FALSE`, the knots are set based on the percentiles of the true event times alone.

lng.in.kn a numeric scalar indicating the number of knots to use (based on the percentiles); default is 15 for the penalized spline baseline hazard and 5 for the regression spline baseline hazard.

ordSpline a numeric scalar setting the order of the spline function. This is the number of coefficients in each piecewise polynomial segment, thus a cubic spline has order 4; default is 4.

diff a numeric scalar setting the order of the differences in the calculation of the penalty term for the penalized baseline hazard; default is 2.

seed a numeric scalar setting the random seed; default is 1.

verbose logical; if `TRUE` (default), a progress bar is shown in the console.

... options passed to the `control` argument.

Details

Function `jointModelBayes` fits joint models for longitudinal and survival data under a Bayesian approach. For the longitudinal responses a linear mixed effects model represented by the `lmeObject` is assumed, unless the user specifies his own probability density function using argument `densLong`. For the survival times, let w_i denote the vector of baseline covariates in `survObject`, with associated parameter vector γ, $m_i(t)$ the subject-specific linear predictor at time point t as defined by the mixed model (i.e., $m_i(t)$ equals the fixed-effects part + random-effects part of the mixed effects model for sample unit i), $m_i'(t)$ denotes an extra user-defined term (based on the specification of argument `extraForm`) to be included in the linear predictor of the survival submodel, and α and α_e vector of association parameters for $m_i(t)$ and $m_i'(t)$, respectively. Then, `jointModelBayes` assumes a relative risk model of the form

$$h_i(t) = h_0(t) \exp\{\gamma^\top w_i + f(m_i(t), m_i'(t), b_i; \alpha, \alpha_e)\},$$

where the baseline risk function is approximated using splines, i.e.,

$$\log h_0(t) = \sum_k \tilde{\gamma}_k B(t; \lambda),$$

with $B(.)$ denoting a B-spline basis function, λ a vector of knots, and $\tilde{\gamma}_k$ the corresponding splines coefficients ($\tilde{\gamma}$ correspond to `Bs.gammas` above). Argument `baseHaz` specifies whether a penalized- or regression-spline-approximation is employed. For the former the P-splines approach of Eilers and Marx (1996) is used, namely the prior for $\tilde{\gamma}$ is taken to be proportional to

$$p(\tilde{\gamma}) \propto \exp\Big(-\frac{\tau_{bs}}{2}\tilde{\gamma}^\top \Delta^\top \Delta \tilde{\gamma}\Big),$$

where Δ denotes the differences matrix (the order of the differences is set by the control argument `diff`).

Function $f(m_i(t), m_i'(t), b_i; \alpha, \alpha_d)$ specifies the association structure between the two processes. In particular, for `param = "td-value"`,

$$f(m_i(t), m_i'(t), b_i; \alpha, \alpha_d) = f_1(m_i(t))\alpha,$$

for `param = "td-extra"`,

$$f(m_i(t), m_i'(t), b_i; \alpha, \alpha_d) = f_2(m_i'(t))\alpha_e,$$

for `param = "td-both"`,

$$f(m_i(t), m_i'(t), b_i; \alpha, \alpha_d) = f_1(m_i(t))\alpha + f_2(m_i'(t))\alpha_e,$$

for `param = "shared-RE"`,

$$f(m_i(t), m_i'(t), b_i; \alpha, \alpha_d) = \alpha^\top b_i,$$

and for `param = "shared-betasRE"`,

$$f(m_i(t), m_i'(t), b_i; \alpha, \alpha_d) = \alpha^\top (\beta^* + b_i),$$

where $f_1(.)$ and $f_2(.)$ denote possible transformation functions, b_i denotes the vector of random effects for the ith subject and β^* the fixed effects that correspond to the random effects.

Value

See `JMbayesObject` for the components of the fit.

Note

1. The `lmeObject` argument should represent a mixed model object without any special structure in the random-effects covariance matrix (i.e., no use of `pdMats`).

2. The `lmeObject` object should not contain any within-group correlation structure (i.e., `correlation` argument of `lme()`) or within-group heteroscedasticity structure (i.e., `weights` argument of `lme()`).

3. It is assumed that the linear mixed effects model `lmeObject` and the survival model `survObject` have been fitted to the same subjects. Moreover, it is assumed that the ordering of the subjects is the same for both `lmeObject` and `survObject`, i.e., that the first line in the data frame containing the event times corresponds to the first set of lines identified by the grouping variable in the data frame

containing the repeated measurements, and so on. Furthermore, the scale of the time variable (e.g., days, months, years) should be the same in both the `lmeObject` and `survObject` objects.

4. In the `print` and `summary` generic functions for class `jointModel`, the estimated coefficients (and standard errors for the `summary` generic) for the event process are augmented with the element "Assoct" that corresponds to the association parameter α and the element "AssoctE" that corresponds to the parameter α_e when `parameterization` is `"td-extra"` or `"td-both"` (see **Details**).

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Henderson, R., Diggle, P. and Dobson, A. (2000) Joint modelling of longitudinal measurements and event time data. *Biostatistics* **1**, 465–480.

Hsieh, F., Tseng, Y.-K. and Wang, J.-L. (2006) Joint modeling of survival and longitudinal data: Likelihood approach revisited. *Biometrics* **62**, 1037–1043.

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.

Tsiatis, A. and Davidian, M. (2004) Joint modeling of longitudinal and time-to-event data: an overview. *Statistica Sinica* **14**, 809–834.

Wulfsohn, M. and Tsiatis, A. (1997) A joint model for survival and longitudinal data measured with error. *Biometrics* **53**, 330–339.

See Also

`coef.JMbayes`, `ranef.JMbayes`, `logLik.JMbayes`, `survfitJM`, `aucJM`, `dynCJM`, `prederrJM`, `predict.JMbayes`

Examples

```
## Not run:
# A joint model for the AIDS dataset:
# First we fit the linear mixed model for the longitudinal measurements of
# sqrt CD4 cell counts
lmeFit.aids <- lme(CD4 ~ obstime * drug, random = ~ obstime | patient, data = aids)
# next we fit the Cox model for the time to death (including the 'x = TRUE' argument)
survFit.aids <- coxph(Surv(Time, death) ~ drug, data = aids.id, x = TRUE)

# the corresponding joint model is fitted by (the default is to assume
# the current value parameterization)
jointFit.aids <- jointModelBayes(lmeFit.aids, survFit.aids, timeVar = "obstime")
summary(jointFit.aids)

# A joint model for the PBC dataset:
```

```
# We first fit the linear mixed and Cox models. In the first we include
# splines to model flexibly the subject-specific longitudinal trajectories
lmeFit.pbc <- lme(log(serBilir) ~ ns(year, 2),
    random = list(id = pdDiag(form = ~ ns(year, 2))), data = pbc2)
survFit.pbc <- coxph(Surv(years, status2) ~ 1, data = pbc2.id, x = TRUE)

# the corresponding joint model is fitted by:
jointFit.pbc <- jointModelBayes(lmeFit.pbc, survFit.pbc, timeVar = "year",
    baseHaz = "regression-splines")
summary(jointFit.pbc)

# we update the joint model fitted for the PBC dataset by including
# the time-dependent slopes term. To achieve this we need to define
# the 'extraForm' argument, in which we use function dns() to numerically
# compute the derivative of the natural cubic spline. In addition, we increase
# the number of MCMC iterations to 35000
dform = list(fixed = ~ 0 + dns(year, 2), random = ~ 0 + dns(year, 2),
    indFixed = 2:3, indRandom = 2:3)
jointFit.pbc2 <- update(jointFit.pbc, param = "td-both", extraForm = dform,
    n.iter = 35000)
summary(jointFit.pbc2)

# we fit the same model with the shared random effects formulation
jointFit.pbc3 <- update(jointFit.pbc, param = "shared-betasRE")
summary(jointFit.pbc3)

# a joint model for left censored longitudinal data
# we create artificial left censoring in serum bilirubin
pbc2$CensInd <- as.numeric(pbc2$serBilir <= 0.8)
pbc2$serBilir2 <- pbc2$serBilir
pbc2$serBilir2[pbc2$CensInd == 1] <- 0.8

censdLong <- function (y, eta.y, scale, log = FALSE, data) {
    log.f <- dnorm(x = y, mean = eta.y, sd = scale, log = TRUE)
    log.F <- pnorm(q = y, mean = eta.y, sd = scale, log.p = TRUE)
    ind <- data$CensInd
    if (log) {
        (1 - ind) * log.f + ind * log.F
    } else {
        exp((1 - ind) * log.f + ind * log.F)
    }
}
lmeFit.pbc2 <- lme(log(serBilir2) ~ ns(year, 2), data = pbc2,
                   random = ~ ns(year, 2) | id, method = "ML")
jointFit.pbc4 <- jointModelBayes(lmeFit.pbc2, survFit.pbc, timeVar = "year",
                                 densLong = censdLong)

summary(jointFit.pbc4)

# a joint model for a binary outcome
pbc2$serBilirD <- 1 * (pbc2$serBilir > 1.8)
lmeFit.pbc3 <- glmmPQL(serBilirD ~ year, random = ~ year | id,
family = binomial, data = pbc2)
```

```
dLongBin <- function (y, eta.y, scale, log = FALSE, data) {
    dbinom(x = y, size = 1, prob = plogis(eta.y), log = log)
}

jointFit.pbc5 <- jointModelBayes(lmeFit.pbc3, survFit.pbc, timeVar = "year",
densLong = dLongBin)

summary(jointFit.pbc5)


# create start-stop counting process variables
pbc <- pbc2[c("id", "serBilir", "drug", "year", "years",
              "status2", "spiders")]
pbc$start <- pbc$year
splitID <- split(pbc[c("start", "years")], pbc$id)
pbc$stop <- unlist(lapply(splitID,
                          function (d) c(d$start[-1], d$years[1]) ))
pbc$event <- with(pbc, ave(status2, id,
                           FUN = function (x) c(rep(0, length(x)-1), x[1])))
pbc <- pbc[!is.na(pbc$spiders), ]

# left-truncation
pbc <- pbc[pbc$start != 0, ]

lmeFit.pbc <- lme(log(serBilir) ~ drug * ns(year, 2),
                  random = ~ ns(year, 2) | id, data = pbc)

tdCox.pbc <- coxph(Surv(start, stop, event) ~ drug * spiders + cluster(id),
                   data = pbc, x = TRUE, model = TRUE)

jointFit.pbc6 <- jointModelBayes(lmeFit.pbc, tdCox.pbc, timeVar = "year")

summary(jointFit.pbc6)

## End(Not run)
```

`logLik.JMbayes` *Log-Likelihood for Joint Models*

Description

Computes the log-likelihood for a fitted joint model.

Usage

```
## S3 method for class 'JMbayes'
logLik(object, thetas, b, priors = TRUE, marginal.b = TRUE,
    marginal.thetas = FALSE, full.Laplace = FALSE, useModes = TRUE, ...)
```

Arguments

object	an object inheriting from class JMBayes.
thetas	a list with values for the joint model's parameters. This should have the same structure as the coefficients component of a fitted joint model. If missing object$postMeans is used.
b	a numeric matrix with random effects value. This should have the same structure as the ranef component of a fitted joint model. If missing ranef(object) is used.
priors	logical, if TRUE the priors are also included in the computation.
marginal.b	logical, if TRUE the marginal log-likelihood over the random effects is returned. This marginalization is done using a Laplace approximation.
marginal.thetas	logical, if TRUE the marginal log-likelihood over the parameters is returned. This marginalization is done using a Laplace approximation.
full.Laplace	logical, if FALSE the posterior means and posterior variances are used in the Laplace approximation instead of the posterior modes and posterior hessian matrix of the random effects. Sacrificing a bit of accuracy, this will be much faster than calculating the posterior modes. Relevant only when marginal.b = TRUE.
useModes	logical, if TRUE the modes are used in the Laplace approximation otherwise the means.
...	extra arguments; currently none is used.

Details

Let y_i denote the vectors of longitudinal responses, T_i the observed event time, and δ_i the event indicator for subject i ($i = 1, \ldots, n$). Let also $p(y_i|b_i;\theta)$ denote the probability density function (pdf) for the linear mixed model, $p(T_i, \delta_i|b_i;\theta)$ the pdf for the survival submodel, and $p(b_i;\theta)$ the multivariate normal pdf for the random effects, where θ denotes the full parameter vector. Then, if priors = TRUE, and marginal.b = TRUE, function logLik() computes

$$\log \int p(y_i|b_i;\theta)p(T_i, \delta_i|b_i;\theta)p(b_i;\theta)db_i + \log p(\theta),$$

where $p(\theta)$ denotes the prior distribution for the parameters. If priors = FALSE the prior is excluded from the computation, i.e.,

$$\log \int p(y_i|b_i;\theta)p(T_i, \delta_i|b_i;\theta)p(b_i;\theta)db_i,$$

and when marginal.b = FALSE, then the conditional on the random effects log-likelihood is computed, i.e.,

$$\log p(y_i|b_i;\theta) + \log p(T_i, \delta_i|b_i;\theta) + \log p(b_i;\theta) + \log p(\theta),$$

when priors = TRUE and

$$\log p(y_i|b_i;\theta) + \log p(T_i, \delta_i|b_i;\theta) + \log p(b_i;\theta),$$

when priors = FALSE.

Value

a numeric scalar of class `logLik` with the value of the log-likelihood. It also has the attributes `df` the number of parameter (excluding the random effects), and `nobs` the number of subjects.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Rizopoulos, D., Hatfield, L., Carlin, B. and Takkenberg, J. (2014). Combining dynamic predictions from joint models for longitudinal and time-to-event data using Bayesian model averaging. *Journal of the American Statistical Association*. to appear.

See Also

`jointModelBayes`

Examples

```
## Not run:
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
    random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ 1, data = pbc2.id, x = TRUE)

jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

logLik(jointFit)
logLik(jointFit, priors = FALSE)
logLik(jointFit, marginal.b = FALSE)

## End(Not run)
```

---

marglogLik *Calculates Marginal Subject-specific Log-Likelihood Contributions*

Description

This function computes marginal subject-specific log-likelihood contributions based on a fitted joint model. The marginalization is done with respect to both the random effects and the parameters using a Laplace approximation.

Usage

```
marglogLik(object, newdata, idVar = "id", method = "BFGS", control = NULL)
```

Arguments

object	an object inheriting from class JMBayes.
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 model (using lme()) and the survival model (using coxph()) that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the different subjects (see also argument idVar).
idVar	the name of the variable in newdata that identifies the different subjects.
method	the method argument of optim().
control	the control argument of optim().

Value

a numeric vector of marginal log-likelihood contributions.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit a joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
    random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)

jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

# we compute marginal log-likelihood contribution for Subject 2
ND <- pbc2[pbc2$id == 2, ] # the data of Subject 2
marglogLik(jointFit, newdata = ND)

## End(Not run)
```

mvglmer *Multivariate Mixed Models*

Description

Fits multivariate mixed models under a Bayesian approach using JAGS.

Usage

```
mvglmer(formulas, data, families, engine = c("JAGS", "STAN"),
    overdispersion = FALSE, priors = NULL, init = NULL, control = NULL, ...)
```

Arguments

formulas	a list of R formulas representing the mixed models; these should be **lme4**-type formulas.
data	a data.frame that contains all the variable to be used when fitting the multivariate mixed model.
families	a list of families objects correspond to each outcome.
engine	a character string indicating whether to use JAGS or STAN to fit the model.
overdispersion	logical; for Poisson outcomes, should an overdispersion parameter be included.
priors	a named list of user-specified prior parameters:
	taus_betas the prior precision parameter for the fixed effects; default is 0.001.
	priorK.D degrees of freedom for the wishart prior for the inverse covariance matrix of the random effects; default is number of random effects plus one.
	priorR.D precision matrix of the wishart prior for the inverse covariance matrix of the random effects; default to a diagonal matrix with diagonal ellements given a Gamma prior with parameters A_R.D and A_R.D.
	A_R.D the prior shape parameter of the Gamma prior for the diagonal elements of the precision matrix of the wishart prior for the inverse covariance matrix of the random effects; default is 0.5.
	B_R.D the prior shape parameter of the Gamma prior for the diagonal elements of the precision matrix of the wishart prior for the inverse covariance matrix of the random effects; default is 0.001.
	tau_half_cauchy prior precision parameter of a half-Cauchy distribution for the precision parameter of a random intercept, when only a single outcome is specified with a single random effect; default is 0.1.
	A_tau the prior shape parameter for the precision of the error terms of Gaussian outcomes.
	B_tau the prior rate parameter for the precision of the error terms of Gaussian outcomes.
init	a list of initial values.
control	a list of control values with components:

n.iter integer specifying the total number of iterations after burn in; default is 28000.

n.burnin integer specifying how many of iterations to discard as burn-in; default is 3000.

n.thin integer specifying the thinning of the chains; default is 50.

n.adapt integer specifying the number of adapt iterations in which the acceptance rates are checked; default is 3000.

n.chains integer specifying the number of chains to use; default is 2.

n.processors integer specifying the number of processors to use; default is the number of available processors minus one.

working.directory a character string giving the path on where to save the JAGS model; default is the working directory.

clear.model logical; should the JAGS models be deleted after the model has run; default is `TRUE`.

seed an integer setting the random seed; default is 1.

`...` options passed to the `control` argument.

Details

This function creates a JAGS program representing a multivariate mixed effects that is run with JAGS using the **jagsUI** package. Currently only Gaussian, Bernoulli and Poisson longitudinal outcomes can be handled.

Value

A list of class `mvglmer` with components:

`mcmc`	a list with the MCMC samples for each parameter.
`components`	a list with design matrices and responses vectors extracted by applying the `formulas` in `data`.
`data`	a copy of `data`.
`control`	a copy of the `control` values used in the fit.
`mcmc.info`	a list with information over the MCMC (i.e., time it took, iterations, etc.).
`DIC`	the DIC value for the fitted model.
`pD`	the pD valu for the fitted model.
`Rhat`	a list with the Rhat convergence diagnostics for each parameter.
`priors`	a copy of the priors used.
`postMeans`	a list with posterior means.
`postModes`	a list with posterior modes calculated using kernel desnisty estimation.
`EffectiveSize`	a list with effective sample sizes.
`StErr`	a list with posterior standard errors.
`StDev`	a list with posterior standard deviations.
`CIs`	a list with 95% credible intervals.
`Pvalues`	a list of tail probabilities for containg the zero value.
`call`	the matched call.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

mvJointModelBayes, jointModelBayes

Examples

```
## Not run:
MixedModelFit <- mvglmer(list(log(serBilir) ~ year + (year | id),
                              spiders ~ year + (1 | id)), data = pbc2,
                         families = list(gaussian, binomial))

summary(MixedModelFit)
plot(MixedModelFit)

## End(Not run)
```

---

mvJointModelBayes *Multivariate Joint Models for Longitudinal and Time-to-Event Data*

Description

Fits multivariate shared parameter joint models for longitudinal and survival outcomes under a Bayesian approach.

Usage

```
mvJointModelBayes(mvglmerObject, survObject, timeVar,
    Formulas = list(NULL), Interactions = list(NULL),
    transFuns = NULL, priors = NULL, multiState = FALSE,
    data_MultiState = NULL, idVar_MultiState = "id",
    control = NULL, ...)
```

Arguments

mvglmerObject	an object of class 'mvglmer' fitted by function mvglmer().
survObject	an object of class 'coxph' fitted by function coxph() or 'survreg' fitted by function survreg() from package **survival**.
timeVar	a character string indicating the time variable in the multivariate mixed effects model.
Formulas	a list of lists. Each inner list should have components fixed a formula representing the fixed-effects part of the user-defined term, indFixed a numeric vector indicating which fixed effects of mvglmerObject are involved in the user-defined term, random a formula representing the random-effects part of the user-defined term, and indRamdom a numeric vector indicating which random effects of mvglmerObject are involved in the user-defined term. See **Examples**.

`Interactions` a list specifying interaction terms for the components of the longitudinal outcomes that are included in the survival submodel. See **Examples**.

`transFuns` a character vector providing transformations of the linear predictors of the mixed models that enter in the linear predictor of the relative risk model. Currently available options are `"identity"` (identity function), `"expit"` (logistic transformation), `"exp"`, `"log"`, `"log2"`, `"log10"` and `"sqrt"`.

`priors` a named list of user-specified prior parameters:

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.

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.

mean_alphas the prior mean vector of the normal prior for the association parameters.

Tau_alphas the prior mean vector of the normal prior for the association parameters.

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.

shrink_gammas logical; should the regression coefficients for the baseline covariates be shrinked.

A_tau_gammas the prior shape parameter of the Gamma prior for the precision parameter of the global penalty term baseline regression coefficients. Only relevant when `shrink_gammas = TRUE`.

B_tau_gammas the prior rate parameter of the Gamma prior for the precision parameter of the penalty term for the baseline regression coefficients. Only relevant when `shrink_gammas = TRUE`.

A_phi_gammas the prior shape parameter of the Gamma prior for the precision parameters of each baseline regression coefficients. Only relevant when `shrink_gammas = TRUE`.

B_phi_gammas the prior rate parameter of the Gamma prior for the precision parameters of each baseline regression coefficients. Only relevant when `shrink_gammas = TRUE`.

shrink_alphas logical; should the association parameters be shrinked.

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 `shrink_alphas = TRUE`.

B_tau_alphas the prior rate parameter of the Gamma prior for the precision parameter of the penalty term for the association parameters. Only relevant when `shrink_alphas = TRUE`.

A_phi_alphas the prior shape parameter of the Gamma prior for the precision parameters of each association parameter. Only relevant when `shrink_alphas = TRUE`.

B_phi_alphas the prior rate parameter of the Gamma prior for the precision parameters of each association parameter. Only relevant when `shrink_alphas = TRUE`.

`multiState` logical; if `TRUE` then a joint model for longitudinal and multi-state survival data is fitted.

`data_MultiState`

A data.frame that contains all the variables which were used to fit the multi-state model. This data.frame should be in long format and include one row for each transition for which a subject is at risk. A column called `trans` indicating the transition to which each row corresponds to, must be included in the data.frame. Function `msprep()` from package **mstate** can be used to easily convert datasets from wide format (one row per subject) to long format while including the necessary `trans` column. For more information and examples see the documentation for function `msprep()` from package **mstate**.

`idVar_MultiState`

A character string indicating the id variable in `data_MultiState` for the multi-state model.

`control` a list of control values with components:

n_iter integer specifying the total number of iterations after burn in; default is 300.

n_burnin integer specifying how many of iterations to discard as burn-in; default is 1000.

n_thin integer specifying the thinning of the chains; default is 300.

n_block integer specifying the number of block iterations in which the acceptance rates are checked; default is 100.

target_acc a numeric scalar denoting the target acceptance rate; default is 0.234.

knots a numeric vector of knots positions for the spline approximation of the log baseline risk function; default is `NULL`, which means that the knots are calculated based on the percentiles of the observed event times.

ObsTimes.knots logical; if `TRUE` (default), the knots are set using the percentiles of the observed event times (i.e., including both true events and censored observations). If `FALSE`, the knots are set based on the percentiles of the true event times alone.

lng.in.kn a numeric scalar indicating the number of knots to use (based on the percentiles); default is 15.

ordSpline an integer setting the order of the spline function. This is the number of coefficients in each piecewise polynomial segment, thus a cubic spline has order 4; default is 4.

diff an integer setting the order of the differences in the calculation of the penalty term for the penalized baseline hazard; default is 2.

seed an integer setting the random seed; default is 1.

n_cores an integer specifying the number of cores to use. Default is the the available cores minus one.

GQsurv a character string specifying the type of Gaussian quadrature to be used. Options are "GaussKronrod" (default) and "GaussLegendre".

GQsurv.k an integer specifying the number of quadrature points; default is 15.

... options passed to the `control` argument.

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 `mvJMbayes` with components:

`mcmc`	a list with the MCMC samples for each parameter.
`components`	a copy of the `components` element of `mvglmerObject`.
`Data`	a list of data used to fit the model.
`control`	a copy of the `control` values used in the fit.
`mcmc.info`	a list with information over the MCMC (i.e., time it took, iterations, etc.).
`priors`	a copy of the priors used.
`postwMeans`	a list with posterior weighted means.
`postMeans`	a list with posterior means.
`postModes`	a list with posterior modes calculated using kernel desnisty estimation.
`EffectiveSize`	a list with effective sample sizes.
`StErr`	a list with posterior standard errors.
`StDev`	a list with posterior standard deviations.
`CIs`	a list with 95% credible intervals.
`Pvalues`	a list of tail probabilities for containg the zero value.
`call`	the matched call.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Henderson, R., Diggle, P. and Dobson, A. (2000) Joint modelling of longitudinal measurements and event time data. *Biostatistics* **1**, 465–480.

Hsieh, F., Tseng, Y.-K. and Wang, J.-L. (2006) Joint modeling of survival and longitudinal data: Likelihood approach revisited. *Biometrics* **62**, 1037–1043.

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.

Tsiatis, A. and Davidian, M. (2004) Joint modeling of longitudinal and time-to-event data: an overview. *Statistica Sinica* **14**, 809–834.

Wulfsohn, M. and Tsiatis, A. (1997) A joint model for survival and longitudinal data measured with error. *Biometrics* **53**, 330–339.

See Also

mvglmer, jointModelBayes

Examples

```
## Not run:
pbc2.id$Time <- pbc2.id$years
pbc2.id$event <- as.numeric(pbc2.id$status != "alive")

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

###############################################
# Univariate joint model for serum bilirubin #
###############################################

# [1] Fit the mixed model using mvglmer(). The main arguments of this function are:
# 'formulas' a list of lme4-like formulas (a formular per outcome),
# 'data' a data.frame that contains all the variables specified in 'formulas' (NAs allowed),
# 'families' a list of family objects specifying the type of each outcome (currently only
# gaussian, binomial and poisson are allowed).
MixedModelFit1 <- mvglmer(list(log(serBilir) ~ year + (year | id)), data = pbc2,
                          families = list(gaussian))

# [2] Fit a Cox model, specifying the baseline covariates to be included in the joint
# model; you need to set argument 'model' to TRUE.
CoxFit1 <- coxph(Surv(Time, event) ~ drug + age, data = pbc2.id, model = TRUE)

# [3] The basic joint model is fitted using a call to mvJointModelBayes(), which is very
# similar to JointModelBayes(), i.e.,
JMFit1 <- mvJointModelBayes(MixedModelFit1, CoxFit1, timeVar = "year")
summary(JMFit1)
plot(JMFit1)

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

#########################################################
# Bivariate joint model for serum bilirubin and spiders #
#########################################################

MixedModelFit2 <- mvglmer(list(log(serBilir) ~ year + (year | id),
                               spiders ~ year + (1 | id)), data = pbc2,
                          families = list(gaussian, binomial))

CoxFit2 <- coxph(Surv(Time, event) ~ drug + age, data = pbc2.id, model = TRUE)
```

```
JMFit2 <- mvJointModelBayes(MixedModelFit2, CoxFit2, timeVar = "year")
summary(JMFit2)
plot(JMFit2)

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

######################
# slopes & area terms #
######################

# We extend model 'JMFit2' by including the value and slope term for bilirubin, and
# the area term for spiders (in the log-odds scale). To include these terms into the model
# we specify the 'Formulas' argument. This is specified in a similar manner as the
# 'derivForms' argument of jointModelBayes(). In particular, it should be a list of lists.
# Each component of the outer list should have as name the name of the corresponding
# outcome variable. Then in the inner list we need to specify four components, namely,
# 'fixed' & 'random' R formulas specifying the fixed and random effects part of the term
# to be included, and 'indFixed' & 'indRandom' integer indicices specifying which of the
# original fixed and random effects are involved in the claculation of the new term. In
# the inner list you can also optionally specify a name for the term you want to include.
# Notes: (1) For terms not specified in the 'Formulas' list, the default value functional
# form is used. (2) If for a particular outcome you want to include both the value
# functional form and an extra term, then you need to specify that in the 'Formulas'
# using two entries. To include the value functional form you only need to set the
# corresponding to 'value', and for the second term to specify the inner list. See
# example below on how to include the value and slope for serum bilirubin (for example,
# if the list below the entry '"log(serBilir)" = "value"' was not give, then only the
# slope term would have been included in the survival submodel).

Forms <- list("log(serBilir)" = "value",
              "log(serBilir)" = list(fixed = ~ 1, random = ~ 1,
                                     indFixed = 2, indRandom = 2, name = "slope"),
              "spiders" = list(fixed = ~ 0 + year + I(year^2/2), random = ~ 0 + year,
                               indFixed = 1:2, indRandom = 1, name = "area"))

JMFit3 <- update(JMFit2, Formulas = Forms)
summary(JMFit3)
plot(JMFit3)

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

#####################
# Interaction terms #
#####################

# We further extend the previous model by including the interactions terms between the
# terms specified in 'Formulas' and 'drug'. The names specified in the list that defined
# the interaction factors should match with the names of the output from 'JMFit3'; the
# only exception is with regard to the 'value' functional form. See specification below
# (to include the interaction of the value term of 'log(serBilir)' with 'drug', in the
# list we can either specify as name of the corresponding formula 'log(serBilir)_value'
# or just 'log(serBilir)'):
```

```
Ints <- list("log(serBilir)" = ~ drug, "log(serBilir)_slope" = ~ drug,
             "spiders_area" = ~ drug)

# because of the many association parameters we have, we place a shrinkage prior on the
# alpha coefficients. In particular, if we have K association parameters, we assume that
# alpha_k ~ N(0, tau * phi_k), k = 1, ..., K. The precision parameters tau and phi_k are
# given Gamma priors. Precision tau is global shrinkage parameter, and phi_k a specific
# per alpha coefficient.
JMFit4 <- update(JMFit3, Interactions = Ints, priors = list(shrink_alphas = TRUE))
summary(JMFit4)
plot(JMFit4)

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

########################
# Time-varying effects #
########################

# We allow the association parameter to vary with time; the time-varying coefficients are
# approximated with P-splines
JMFit_tveffect <- mvJointModelBayes(MixedModelFit1, CoxFit1, timeVar = "year",
                    Interactions = list("log(serBilir)_value" = ~ 0 + tve(Time, df = 8)))

plot(JMFit_tveffect, "tv_effect")


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

############################
# Interval censoring terms #
############################

# create artificial interval censoring in the PBC data set
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")
pbc2$status3 <- as.character(pbc2$status)
ff <- function (x) {
    out <- if (x[1L] %in% c('dead', 'transplanted')) 'dead' else
        switch(sample(1:3, 1), '1' = "alive", '2' = "left", '3' = "interval")
    rep(out, length(x))
}
pbc2$status3 <- unlist(with(pbc2, lapply(split(status3, id), ff)), use.names = FALSE)
pbc2$status3 <- unname(with(pbc2, sapply(status3, function (x)
    switch(x, 'dead' = 1, 'alive' = 0, 'left' = 2, 'interval' = 3))))
pbc2$yearsU <- as.numeric(NA)
pbc2$yearsU[pbc2$status3 == 3] <- pbc2$years[pbc2$status3 == 3] +
    runif(sum(pbc2$status3 == 3), 0, 4)
pbc2.id <- pbc2[!duplicated(pbc2$id), ]

# next we fit a weibull model for interval censored data
survFit <- survreg(Surv(years, yearsU, status3, type = "interval") ~ drug + age,
                   data = pbc2.id, model = TRUE)
```

```
# next we fit the joint model (we use 'MixedModelFit1' from above)
JMFit_intcens <- mvJointModelBayes(MixedModelFit1, survFit, timeVar = "year")
summary(JMFit_intcens)

## End(Not run)
```

---

pbc2 *Mayo Clinic Primary Biliary Cirrhosis Data*

Description

Followup 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 gm/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.

`plot` *MCMC Diagnostics for Joint Models*

Description

Produces MCMC diagnostics plots.

Usage

```
## S3 method for class 'JMbayes'
plot(x, which = c("trace", "autocorr", "density", "CPO", "weightFun"),
    param = c("betas", "sigma", "D", "gammas", "alphas", "Dalphas",
        "shapes", "Bs.gammas", "tauBs"), ask = TRUE, max.t = NULL,
    from = 0, ...)
```

Arguments

`x`	an object inheriting from class `JMbayes`.
`which`	which types of plots to produce.
`param`	for which parameter to produce the MCMC diagnostic plots; default is for all parameters.
`ask`	logical, if `TRUE` the user is asked for input, before a new figure is drawn.
`max.t`	numeric scalar; up to which time point to plot the weight function, default is up to the third quantile of the observed event times.
`from`	numeric scalar; from which time point to start plotting the weight function, default is zero.
`...`	additional arguments; currently none is used.

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.

See Also

jointModelBayes

Examples

```
## Not run:
# linear mixed model fit
fitLME <- lme(log(serBilir) ~ drug * year, random = ~ 1 | id, data = pbc2)

# survival regression fit
fitSURV <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)

# joint model fit, under the (default) Weibull model
fitJOINT <- jointModelBayes(fitLME, fitSURV, timeVar = "year")

plot(fitJOINT)

## End(Not run)
```

---

plot.survfitJM *Plot Method for survfit.JMbayes and survfit.mvJMbayes Objects*

Description

Produces plots of conditional probabilities of survival.

Usage

```
## S3 method for class 'survfit.JMbayes'
plot(x, estimator = c("both", "mean", "median"),
    which = NULL, fun = NULL, invlink = NULL, conf.int = FALSE,
   fill.area = FALSE, col.area = "grey", col.abline = "black", col.points = "black",
    add.last.time.axis.tick = FALSE, include.y = FALSE, main = NULL,
    xlab = NULL, ylab = NULL, ylab2 = NULL, lty = NULL, col = NULL,
    lwd = NULL, pch = NULL, ask = NULL, legend = FALSE, ...,
    cex.axis.z = 1, cex.lab.z = 1, xlim = NULL)

## S3 method for class 'survfit.mvJMbayes'
plot(x, split = c(1, 1), which_subjects = NULL,
    which_outcomes = NULL, surv_in_all = TRUE, include.y = TRUE, fun = NULL,
    abline = NULL,
    main = NULL, xlab = "Time", ylab = NULL, zlab = "Event-Free Probability",
    include_CI = TRUE, fill_area_CI = TRUE, col_points = "black",
```

```
    pch_points = 1, col_lines = "red", col_lines_CI = "black",
    col_fill_CI = "lightgrey", lwd_lines = 2, lty_lines_CI = 2,
    cex_xlab = 1, cex_ylab = 1, cex_zlab = 1, cex_main = 1,
    cex_axis = 1, ...)
```

Arguments

x	an object inheriting from class survfit.JMbayes or class survfit.mvJMbayes.
estimator	character string specifying, whether to include in the plot the mean of the conditional probabilities of survival, the median or both. The mean and median are taken as estimates of these conditional probabilities over the M replications of the Monte Carlo scheme described in survfitJM.
which	an integer or character vector specifying for which subjects to produce the plot. If a character vector, then is should contain a subset of the values of the idVar variable of the newdata argument of survfitJM.
which_subjects	an integer vector specifying for which subjects to produce the plot.
split	a integer vector of length 2 indicating in how many panels to construct, i.e., number of rows and number of columns.
which_outcomes	integer vector indicating which longitudinal outcomes to include in the plot.
surv_in_all	logical; should the survival function be included in all panels.
fun	a vectorized function defining a transformation of the survival curve. For example, with fun=log the log-survival curve is drawn.
abline	a list with arguments to abline().
invlink	a function to transform the fitted values of the longitudinal outcome.
conf.int, include_CI	logical; if TRUE, then a pointwise confidence interval is included in the plot.
fill.area, fill_area_CI	logical; if TRUE the area defined by the confidence interval of the survival function is put in color.
col.area, col_fill_CI	the color of the area defined by the confidence interval of the survival function.
col.abline, col.points, col_points, col_lines, col_lines_CI	the color for the vertical line and the points when include.y is TRUE.
add.last.time.axis.tick	logical; if TRUE, a tick is added in the x-axis for the last available time point for which a longitudinal measurement was available.
include.y	logical; if TRUE, two plots are produced per subject, i.e., the plot of conditional probabilities of survival and a scatterplot of his longitudinal measurements.
main	a character string specifying the title in the plot.
xlab	a character string specifying the x-axis label in the plot.
ylab	a character string specifying the y-axis label in the plot.
ylab2	a character string specifying the y-axis label in the plot, when include.y = TRUE.

zlab	a character string specifying the z-axis (vertical right-hand side) label in the plot.
lty, lty_lines_CI	what types of lines to use.
col	which colors to use.
lwd, lwd_lines	the thickness of the lines.
pch, pch_points	the type of points to use.
ask	logical; if TRUE, the user is asked before each plot, see par().
legend	logical; if TRUE, a legend is included in the plot.
cex.axis.z, cex.lab.z	the par cex argument for the axis at side 4, when include.y = TRUE.
cex_xlab, cex_ylab, cex_zlab, cex_main, cex_axis	the par cex argument for the axis in side 1 (x-axis), side 2 (y-axis), side 4 (z-axis) and the title of the plot.
xlim	the par xlim argument.
...	extra graphical parameters passed to plot().

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

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.

See Also

survfitJM

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
    random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")
```

```
# we will compute survival probabilities for Subject 2 in a dynamic manner,
# i.e., after each longitudinal measurement is recorded
ND <- pbc2[pbc2$id == 2, ] # the data of Subject 2
survPreds <- vector("list", nrow(ND))
for (i in 1:nrow(ND)) {
    survPreds[[i]] <- survfitJM(jointFit, newdata = ND[1:i, ])
}

# the default call to the plot method using the first three
# longitudinal measurements
plot(survPreds[[3]])

# we produce the corresponding plot
par(mfrow = c(2, 2), oma = c(0, 2, 0, 2))
for (i in c(1,3,5,7)) {
    plot(survPreds[[i]], estimator = "median", conf.int = TRUE,
        include.y = TRUE, main = paste("Follow-up time:",
            round(survPreds[[i]]$last.time, 1)), ylab = "", ylab2 = "")
}
mtext("log serum bilirubin", side = 2, line = -1, outer = TRUE)
mtext("Survival Probability", side = 4, line = -1, outer = TRUE)

## End(Not run)
```

---

prederrJM *Prediction Errors for Joint Models*

Description

Using the available longitudinal information up to a starting time point, this function computes an estimate of the prediction error of survival at a horizon time point based on joint models.

Usage

```
prederrJM(object, newdata, Tstart, Thoriz, ...)

## S3 method for class 'JMbayes'
prederrJM(object, newdata, Tstart, Thoriz,
    lossFun = c("square", "absolute"), interval = FALSE, idVar = "id",
    simulate = FALSE, M = 100, ...)

## S3 method for class 'mvJMbayes'
prederrJM(object, newdata, Tstart, Thoriz,
    lossFun = c("square", "absolute"), interval = FALSE, idVar = "id",
    M = 100, ...)
```

Arguments

object	an object inheriting from class JMbayes or mvJMbayes.
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 model (using lme()) and the survival model (using coxph()) that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the different subjects (see also argument idVar).
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 mast be later than Tstart.
lossFun	either the options "absolute" or "square" (default), or a user-specified loss function. As the names suggest, when lossFun = "absolute" the loss function is $L(x) = \|x\|$, whereas when lossFun = "square" the loss function is $L(x) = x^2$. If a user-specified function is supplied, this should have a single argument and be vectorized.
interval	logical; if TRUE the weighted prediction error in the interval [Tstart, Thoriz] is calculated, while if FALSE the prediction error at time Thoriz is calculated using the longitudinal information up to time Tstart.
idVar	the name of the variable in newdata that identifies the different subjects.
simulate	logical; if TRUE, a Monte Carlo approach is used to estimate survival probabilities. If FALSE, a first order estimator is used instead. See survfitJM for mote details.
M	a numeric scalar denoting the number of Monte Carlo samples; see survfitJM for mote details.
...	additional arguments; currently none is used.

Details

Based on a fitted joint model (represented by object) and using the data supplied in argument newdata, this function computes the following estimate of the prediction:

$$PE(u|t) = \{R(t)\}^{-1} \sum_{i:T_i \geq s} I(T_i \geq u) L\{1 - Pr(T_i > u | T_i > t, \tilde{y}_i(t), x_i)\}$$

$$+\delta_i I(T_i < u) L\{0 - Pr(T_i > u | T_i > t, \tilde{y}_i(t), x_i)\}$$

$$+(1-\delta_i) I(T_i < u)[S_i(u \mid T_i, \tilde{y}_i(t)) L\{1 - Pr(T_i > u | T_i > t, \tilde{y}_i(t), x_i)\}$$

$$+\{1 - S_i(u \mid T_i, \tilde{y}_i(t))\} L\{0 - Pr(T_i > u | T_i > t, \tilde{y}_i(t), x_i)\}],$$

where $R(t)$ denotes the number of subjects at risk at time $t =$ Tstart, $\tilde{y}_i(t) = \{y_i(s), 0 \leq s \leq t\}$ denotes the available longitudinal measurements up to time t, T_i denotes the observed event time for subject i, δ_i is the event indicator, s is the starting time point Tstart up to which the longitudinal information is used, and $u > s$ is the horizon time point Thoriz. Function $L(.)$ is the loss function

that can be the absolute value (i.e., $L(x) = |x|$), the squared value (i.e., $L(x) = x^2$), or a user-specified function. The probabilities $Pr(T_i > u|T_i > t, \tilde{y}_i(t), x_i)$ are calculated by `survfitJM`.

When `interval` is set to `TRUE`, then function `prederrJM` computes the integrated prediction error in the interval $(u, t) =$ `(Tstart, Thoriz)` defined as

$$IPE(u|t) = \sum_{i:t \leq T_i \leq u} w_i(T_i) PE(T_i|t),$$

where

$$w_i(T_i) = \frac{\delta_i G(T_i)/G(t)}{\sum_{i:t \leq T_i \leq u} \delta_i G(T_i)/G(t)},$$

with $G(.)$ denoting the Kaplan-Meier estimator of the censoring time distribution.

Value

A list of class `prederrJM` with components:

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

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

Henderson, R., Diggle, P. and Dobson, A. (2002). Identification and efficacy of longitudinal markers for survival. *Biostatistics* **3**, 33–50.

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.

See Also

`survfitJM`, `aucJM`, `dynCJM`, `jointModelBayes`

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
    random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

# prediction error at year 10 using longitudinal data up to year 5
prederrJM(jointFit, pbc2, Tstart = 5, Thoriz = 10)
prederrJM(jointFit, pbc2, Tstart = 5, Thoriz = 6.5, interval = TRUE)

## End(Not run)
```

---

predict *Predictions for Joint Models*

Description

Calculates predicted values for the longitudinal part of a joint model.

Usage

```
## S3 method for class 'JMbayes'
predict(object, newdata, type = c("Marginal", "Subject"),
    interval = c("none", "confidence", "prediction"), level = 0.95, idVar = "id",
    FtTimes = NULL, last.time = NULL, LeftTrunc_var = NULL,
    M = 300, returnData = FALSE, scale = 1.6,
    weight = rep(1, nrow(newdata)), invlink = NULL, seed = 1, ...)
```

Arguments

object	an object inheriting from class JMBayes.
newdata	a data frame in which to look for variables with which to predict.
type	a character string indicating the type of predictions to compute, marginal or subject-specific. See **Details**.
interval	a character string indicating what type of intervals should be computed.
level	a numeric scalar denoting the tolerance/confidence level.
idVar	a character string indicating the name of the variable in newdata that corresponds to the subject identifier; required when type = "Subject".

FtTimes	a list with components numeric vectors denoting the time points for which we wish to compute subject-specific predictions after the last available measurement provided in newdata. For each subject in newdata the default is a sequence of 25 equally spaced time points from the last available measurement to the maximum follow-up time of all subjects (plus a small quantity). This argument is only used when type = "Subject".
last.time	a numeric vector. This specifies the known time at which each of the subjects in newdata was known to be alive. If NULL, then this is automatically taken as the last time each subject provided a longitudinal measurement. If a numeric vector, then it is assumed to contain this last time point for each subject.
LeftTrunc_var	character string indicating the name of the variable in newdata that denotes the left-truncation time.
M	numeric scalar denoting the number of Monte Carlo samples. See **Details**.
returnData	logical; if TRUE the data frame supplied in newdata is returned augmented with the outputs of the function.
scale	a numeric value setting the scaling of the covariance matrix of the empirical Bayes estimates in the Metropolis step during the Monte Carlo sampling.
weight	a numeric vector of weights to be applied to the predictions of each subject.
invlink	a function to tranform the linear predictor of the mixed model to fitted means; relevant when the user has specified her own density for the longitudinal outcome in jointModelBayes.
seed	numeric scalar, the random seed used to produce the results.
...	additional arguments; currently none is used.

Details

When type = "Marginal", this function computes predicted values for the fixed-effects part of the longitudinal submodel. In particular, let X denote the fixed-effects design matrix calculated using newdata. The predict() calculates $\hat{y} = X\hat{\beta}$, and if interval = "confidence", then it calculates the confidence intervals based on the percentiles of the MCMC sample for β.

When type = "Subject", this functions computes subject-specific predictions for the longitudinal outcome based on the joint model. This accomplished with a Monte Carlo simulation scheme, similar to the one described in survfitJM. The only difference is in Step 3, where for interval = "confidence" $y_i^* = X_i\beta^* + Z_i b_i^*$, whereas for interval = "prediction" y_i^* is a random vector from a normal distribution with mean $X_i\beta^* + Z_i b_i^*$ and standard deviation σ^*. Based on this Monte Carlo simulation scheme we take as estimate of $\hat{y}_i$ the average of the M estimates y_i^* from each Monte Carlo sample. Confidence intervals are constructed using the percentiles of y_i^* from the Monte Carlo samples.

Value

If se.fit = FALSE a numeric vector of predicted values, otherwise a list with components pred the predicted values, se.fit the standard error for the fitted values, and low and upp the lower and upper limits of the confidence interval. If returnData = TRUE, it returns the data frame newdata with the previously mentioned components added.

Note

The user is responsible to appropriately set the `invlink` argument when a user-specified mixed effects model has been fitted.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

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.

See Also

`survfitJM.JMbayes`, `jointModelBayes`

Examples

```
## Not run:
# linear mixed model fit
fitLME <- lme(log(serBilir) ~ drug * year, data = pbc2,
    random = ~ year | id)
# survival regression fit
fitSURV <- coxph(Surv(years, status2) ~ drug, data = pbc2.id,
    x = TRUE)
# joint model fit, under the (default) Weibull model
fitJOINT <- jointModelBayes(fitLME, fitSURV, timeVar = "year")

DF <- with(pbc2, expand.grid(drug = levels(drug),
    year = seq(min(year), max(year), len = 100)))
Ps <- predict(fitJOINT, DF, interval = "confidence", return = TRUE)
require(lattice)
xyplot(pred + low + upp ~ year | drug, data = Ps,
    type = "l", col = c(2,1,1), lty = c(1,2,2), lwd = 2,
    ylab = "Average log serum Bilirubin")


# Subject-specific predictions
ND <- pbc2[pbc2$id == 2, ]
Ps.ss <- predict(fitJOINT, ND, type = "Subject",
  interval = "confidence", return = TRUE)
xyplot(pred + low + upp ~ year | id, data = Ps.ss,
    type = "l", col = c(2,1,1), lty = c(1,2,2), lwd = 2,
    ylab = "Average log serum Bilirubin")

## End(Not run)
```

prothro *Prednisone versus Placebo in Liver Cirrhosis Patients*

Description

A randomized trial on 488 liver cirrhosis patients

Format

Two data frames with the following variable.

`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".

Source

http://www.gllamm.org/books/readme.html#14.6,

References

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

ranef *Random Effects Estimates for Joint Models*

Description

Extracts the random effects estimates from a fitted joint model.

Usage

```
## S3 method for class 'JMbayes'
ranef(object, postVar = FALSE, ...)
```

Arguments

`object`	an object inheriting from class `JMbayes`.
`postVar`	logical; if `TRUE` the variance of the posterior distribution is also returned.
`...`	additional arguments; currently none is used.

Value

a numeric matrix with rows denoting the individuals and columns the random effects (e.g., intercepts, slopes, etc.). If `postVar = TRUE`, the numeric matrix has an extra attribute "postVar".

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.

See Also

`coef.JMbayes`, `jointModelBayes`

Examples

```
## Not run:
# linear mixed model fit
fitLME <- lme(log(serBilir) ~ drug * year, random = ~ 1 | id, data = pbc2)
# survival regression fit
fitSURV <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)

# joint model fit, under the (default) Weibull model
fitJOINT <- jointModelBayes(fitLME, fitSURV, timeVar = "year")
ranef(fitJOINT)

## End(Not run)
```

---

runDynPred *Shiny Application for Dynamic Predictions*

Description

This function loads the shiny package and runs the application for calculating dynamic predictions using package JMbayes.

Usage

```
runDynPred(type = c("JM", "lme"))
```

Arguments

`type`	character string indicating whether dynamic predictions are based on joint models or mixed models alone.

Value

No value returned.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

Examples

```
## Not run:
runDynPred()

## End(Not run)
```

---

survfitJM *Prediction in Joint Models*

Description

This function computes the conditional probability of surviving later times than the last observed time for which a longitudinal measurement was available.

Usage

```
survfitJM(object, newdata, ...)

## S3 method for class 'JMbayes'
survfitJM(object, newdata,
   type = c("SurvProb", "Density"), idVar = "id",
   simulate = TRUE, survTimes = NULL, last.time = NULL,
   LeftTrunc_var = NULL, M = 200L,
   CI.levels = c(0.025, 0.975), log = FALSE, scale = 1.6,
   weight = rep(1, nrow(newdata)),
   init.b = NULL, seed = 1L, ...)

## S3 method for class 'mvJMbayes'
survfitJM(object, newdata,
    survTimes = NULL, idVar = "id", last.time = NULL,
    M = 200L, scale = 1.6, log = FALSE,
    CI.levels = c(0.025, 0.975), seed = 1L, ...)
```

Arguments

object	an object inheriting from class JMBayes or mvJMBayes.

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 model (using lme()) and the survival model (using coxph()) that were supplied as the two first argument of jointModelBayes. In addition, this data frame should contain a variable that identifies the different subjects (see also argument idVar).
type	character string indicating what to compute, i.e., survival probabilities or the log conditional density.
idVar	the name of the variable in newdata that identifies the different subjects.
simulate	logical; if TRUE, a Monte Carlo approach is used to estimate survival probabilities. If FALSE, a first order estimator is used instead. (see **Details**)
survTimes	a numeric vector of times for which prediction survival probabilities are to be computed.
last.time	a numeric vector or character string. This specifies the known time at which each of the subjects in newdata was known to be alive. If NULL, then this is automatically taken as the last time each subject provided a longitudinal measurement. If a numeric vector, then it is assumed to contain this last time point for each subject. If a character string, then it should be a variable in the data frame newdata.
LeftTrunc_var	character string indicating the name of the variable in newdata that denotes the left-truncation time.
M	integer denoting how many Monte Carlo samples to use – see **Details**.
CI.levels	a numeric vector of length two that specifies which quantiles to use for the calculation of confidence interval for the predicted probabilities – see **Details**.
log	logical, should results be returned in the log scale.
scale	a numeric scalar that controls the acceptance rate of the Metropolis-Hastings algorithm – see **Details**.
weight	a numeric vector of weights to be applied to the predictions of each subject.
init.b	a numeric matrix of initial values for the random effects. These are used in the optimization procedure that finds the mode of the posterior distribution described in Step 2 below.
seed	numeric scalar, the random seed used to produce the results.
...	additional arguments; currently none is used.

Details

Based on a fitted joint model (represented by object), and a history of longitudinal responses $\tilde{y}_i(t) = \{y_i(s), 0 \leq s \leq t\}$ and a covariates vector x_i (stored in newdata), this function provides estimates of $Pr(T_i > u | T_i > t, \tilde{y}_i(t), x_i)$, i.e., the conditional probability of surviving time u given that subject i, with covariate information x_i, has survived up to time t and has provided longitudinal the measurements $\tilde{y}_i(t)$.

To estimate $Pr(T_i > u | T_i > t, \tilde{y}_i(t), x_i)$ and if simulate = TRUE, a Monte Carlo procedure is followed with the following steps:

Step 1: Take randomly a realization, say θ^* from the MCMC sample of posterior of the joint model represented by `object`.

Step 2: Simulate random effects values, say b_i^*, from their posterior distribution given survival up to time t, the vector of longitudinal responses $\tilde{y}_i(t)$ and θ^*. This is achieved using a Metropolis-Hastings algorithm with independent proposals from a properly centered and scaled multivariate t distribution. The `scale` argument controls the acceptance rate for this algorithm.

Step 3 Using θ^* and b_i^*, compute $Pr(T_i > u | T_i > t, b_i^*, x_i; \theta^*)$.

Step 4: Repeat Steps 1-3 `M` times.

Based on the `M` estimates of the conditional probabilities, we compute useful summary statistics, such as their mean, median, and percentiles (to produce a confidence interval).

If `simulate = FALSE`, then survival probabilities are estimated using the formula

$$Pr(T_i > u | T_i > t, \hat{b}_i, x_i; \hat{\theta}),$$

where $\hat{\theta}$ denotes the posterior means for the parameters, and $\hat{b}_i$ denotes the posterior means for the random effects.

Value

A list of class `survfit.JMbayes` with components:

`summaries`	a list with elements numeric matrices with numeric summaries of the predicted probabilities for each subject.
`survTimes`	a copy of the `survTimes` argument.
`last.time`	a numeric vector with the time of the last available longitudinal measurement of each subject.
`obs.times`	a list with elements numeric vectors denoting the timings of the longitudinal measurements for each subject.
`y`	a list with elements numeric vectors denoting the longitudinal responses for each subject.
`full.results`	a list with elements numeric matrices with predicted probabilities for each subject in each replication of the Monte Carlo scheme described above.
`success.rate`	a numeric vector with the success rates of the Metropolis-Hastings algorithm described above for each subject.
`scale`	a copy of the `scale` argument.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

References

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.

See Also

plot.survfit.JMbayes, predict.JMbayes, aucJM, dynCJM, prederrJM, jointModelBayes

Examples

```
## Not run:
# we construct the composite event indicator (transplantation or death)
pbc2$status2 <- as.numeric(pbc2$status != "alive")
pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")

# we fit the joint model using splines for the subject-specific
# longitudinal trajectories and a spline-approximated baseline
# risk function
lmeFit <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
    random = ~ ns(year, 2) | id)
survFit <- coxph(Surv(years, status2) ~ drug, data = pbc2.id, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "year")

# we will compute survival probabilities for Subject 2 in a dynamic manner,
# i.e., after each longitudinal measurement is recorded
ND <- pbc2[pbc2$id == 2, ] # the data of Subject 2
survPreds <- vector("list", nrow(ND))
for (i in 1:nrow(ND)) {
    survPreds[[i]] <- survfitJM(jointFit, newdata = ND[1:i, ])
}
survPreds

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

# Predictions from multivariate models

pbc2 <- pbc2[!is.na(pbc2$serChol), ]
pbc2.id <- pbc2[!duplicated(pbc2$id), ]
pbc2.id$Time <- pbc2.id$years
pbc2.id$event <- as.numeric(pbc2.id$status != "alive")

# Fit a trivariate joint model
MixedModelFit <- mvglmer(list(log(serBilir) ~ year + (year | id),
                              sqrt(serChol) ~ year + (year | id),
                              hepatomegaly ~ year + (year | id)), data = pbc2,
                         families = list(gaussian, gaussian, binomial), engine = "STAN")

CoxFit <- coxph(Surv(Time, event) ~ drug + age, data = pbc2.id, model = TRUE)

JMFit <- mvJointModelBayes(MixedModelFit, CoxFit, timeVar = "year")

# We want survival probabilities for three subjects
ND <- pbc2[pbc2$id %in% c(2, 25, 81), ]

sprobs <- survfitJM(JMFit, ND)
sprobs
```

```
# Basic plot
plot(sprobs)

# split in a 2 rows 2 columns and include the survival function in
# a separate panel; plot only the third & first subjects; change various defaults
plot(sprobs, split = c(3, 2), surv_in_all = FALSE, which_subjects = c(3, 1),
     lty_lines_CI = 3, col_lines = "blue", col_fill_CI = "red",
     col_points = "pink", pch_points = 12)

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

# run Shiny app
runDynPred()

## End(Not run)
```

---

tve *Time-Varying Effects using P-splines*

Description

A B-spline expansion of the input variables to be used for a time-varying effect in the specification of joint model.

Usage

```
tve(x, df = NULL, knots = NULL, ord = 3)
```

Arguments

x	a numeric input variable.
df	integer denoting the degrees of freedom.
knots	a numeric vector of knots.
ord	an integer denoting the order of the spline.

Value

an object of class tve.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

xtable *xtable Method from Joint Models.*

Description

produces a LaTeX table with the results of a joint model using package xtable.

Usage

```
## S3 method for class 'JMbayes'
xtable(x, caption = NULL, label = NULL, align = NULL,
    digits = NULL, display = NULL, which = c("all", "Longitudinal", "Event"),
    varNames.Long = NULL, varNames.Event = NULL, ...)
```

Arguments

x	an object inheriting from class JMbayes.
caption	the caption argument of xtable().
label	the label argument of xtable().
align	the align argument of xtable().
digits	the digits argument of xtable().
display	the display argument of xtable().
which	a character string indicating which results to include in the LaTeX table. Options are all results, the results of longitudinal submodel or the results of the survival submodel.
varNames.Long	a character vector of the variable names for the longitudinal submodel.
varNames.Event	a character vector of the variable names for the survival submodel.
...	additional arguments; currently none is used.

Value

A LaTeX code chunk with the results of the joint modeling analysis.

Author(s)

Dimitris Rizopoulos <d.rizopoulos@erasmusmc.nl>

See Also

jointModelBayes

Examples

```
## Not run:
prothro$t0 <- as.numeric(prothro$time == 0)
lmeFit <- lme(pro ~ treat * (time + t0), random = ~ time | id, data = prothro)
survFit <- coxph(Surv(Time, death) ~ treat, data = prothros, x = TRUE)
jointFit <- jointModelBayes(lmeFit, survFit, timeVar = "time")

if (require("xtable")) {
    xtable:::xtable(jointFit, math.style.negative = TRUE)
}

## End(Not run)
```

Index