Package: joineRML (via r-universe)

October 29, 2026

Type Package

Title Joint Modelling of Multivariate Longitudinal Data and Time-to-Event Outcomes

Version 0.4.8

Encoding UTF-8

Description Fits the joint model proposed by Henderson and colleagues (2000) <doi:10.1093/biostatistics/1.4.465>, but extended to the case of multiple continuous longitudinal measures. The time-to-event data is modelled using a Cox proportional hazards regression model with time-varying covariates. The multiple longitudinal outcomes are modelled using a multivariate version of the Laird and Ware linear mixed model. The association is captured by a multivariate latent Gaussian process. The model is estimated using a Monte Carlo Expectation Maximization algorithm. This project was funded by the Medical Research Council (Grant number MR/M013227/1).

License GPL-3 | file LICENSE

URL https://graemeleehickey.github.io/joineRML/, https://github.com/graemeleehickey/joineRML, https://CRAN.R-project.org/package=joineRML

BugReports https://github.com/graemeleehickey/joineRML/issues

LazyData true

Depends R (>= 4.1.0), nlme, survival

Imports cobs, doParallel, foreach, generics, ggplot2, graphics, lme4 (>= 1.1-8), MASS, Matrix, mvtnorm, parallel, randtoolbox, Rcpp (>= 0.12.7), RcppArmadillo, stats, tibble, utils

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.3

Suggests dplyr, JM, joineR, knitr, pkgdown, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Graeme L. Hickey [cre, aut] (ORCID: <https://orcid.org/0000-0002-4989-0054>), Pete Philipson [aut] (ORCID: <https://orcid.org/0000-0001-7846-0208>), Andrea Jorgensen [ctb] (ORCID: <https://orcid.org/0000-0002-6977-9337>), Ruwanthi Kolamunnage-Dona [aut] (ORCID: <https://orcid.org/0000-0003-3886-6208>), Paula Williamson [ctb] (ORCID: <https://orcid.org/0000-0001-9802-6636>), Dimitris Rizopoulos [ctb, dtc] (data/renal.rda, R/hessian.R, R/vcov.R), Alessandro Gasparini [aut] (ORCID: <https://orcid.org/0000-0002-8319-7624>), Medical Research Council [fnd] (Grant number: MR/M013227/1)

Maintainer Graeme L. Hickey <graemeleehickey@gmail.com>

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

Date/Publication 2026-06-14 07:01:02 UTC

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

RemoteRef HEAD

RemoteSha 177152221f7154d8a47bccdaba647d8d49052b78

Contents

baseHaz . . . 3
bootSE . . . 4
confint.mjoint . . . 8
dynLong . . . 10
dynSurv . . . 13
epileptic.qol . . . 16
fitted.mjoint . . . 17
fixef.mjoint . . . 18
formula.mjoint . . . 20
getVarCov.mjoint . . . 21
heart.valve . . . 22
joineRML . . . 23
logLik.mjoint . . . 24
mjoint . . . 25
mjoint.object . . . 31
mjoint_tidiers . . . 33
pbc2 . . . 36
plot.dynLong . . . 37
plot.dynSurv . . . 38
plot.mjoint . . . 41
plot.ranef.mjoint . . . 42

plotConvergence 43
ranef.mjoint 44
renal 45
residuals.mjoint 47
sampleData 48
sigma.mjoint 49
simData 50
summary.mjoint 52
vcov.mjoint 53

Index **56**

`baseHaz` *The baseline hazard estimate of an* `mjoint` *object*

Description

This function returns the (baseline) hazard increment from a fitted `mjoint` object. In addition, it can report either the *uncentered* or the more ubiquitous *centered* version.

Usage

```
baseHaz(object, centered = TRUE, se = FALSE)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`centered`	logical: should the baseline hazard be for the mean-centered covariates model or not? Default is `centered=TRUE`. See **Details**.
`se`	logical: should standard errors be approximated for the hazard increments? Default is `se=FALSE`.

Details

When covariates are included in the time-to-event sub-model, `mjoint` automatically centers them about their respective means. This also applies to non-continuous covariates, which are first coded using a dummy-transformation for the design matrix and subsequently centered. The reason for the mean-centering is to improve numerical stability, as the survival function involves exponential terms. Extracting the baseline hazard increments from `mjoint.object` returns the Breslow hazard estimate (Lin, 2007) that corresponds to this mean-centered model. This is the same as is done in the R `survival` package when using `coxph.detail` (Therneau and Grambsch, 2000). If the user wants to access the baseline hazard estimate for the model in which no mean-centering is applied, then they can use this function, which scales the mean-centered baseline hazard by

$$\exp\{-\bar{w}^\top \gamma_v\},$$

where $\bar{w}$ is a vector of the means from the time-to-event sub-model design matrix.

Value

A `data.frame` with two columns: the unique failure times and the estimate baseline hazard. If `se=TRUE`, then a third column is appended with the corresponding standard errors (for the centred case).

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Therneau TM, Grambsch PM. *Modeling Survival Data: Extending the Cox Model.* New Jersey: Springer-Verlag; 2000.

Lin DY. On the Breslow estimator. *Lifetime Data Anal.* 2007; **13(4)**: 471-480.

See Also

`mjoint` and `coef`.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)
baseHaz(fit2, centered = FALSE)

## End(Not run)
```

---

bootSE *Standard errors via bootstrap for an* `mjoint` *object*

Description

This function takes a model fit from an `mjoint` object and calculates standard errors and confidence intervals for the main longitudinal and survival coefficient parameters, including the latent association parameters, using bootstrapping (Efron and Tibshirani, 2000).

Usage

```
bootSE(
  object,
  nboot = 100,
  ci = 0.95,
  use.mle = TRUE,
  verbose = FALSE,
  control = list(),
  progress = TRUE,
  ncores = 1L,
  safe.boot = FALSE,
  ...
)
```

Arguments

object	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.
nboot	the number of bootstrap samples. Default is nboot=100.
ci	the confidence interval to be estimated using the percentile-method. Default is ci=0.95 for a 95% confidence interval.
use.mle	logical: should the algorithm use the maximizer from the converged model in object as initial values for coefficients in each bootstrap iteration. Default is use.mle=TRUE.
verbose	logical: if TRUE, the parameter estimates and other convergence statistics are value are printed at each iteration of the MCEM algorithm. Default is FALSE.
control	a list of control values with components:

nMC integer: the initial number of Monte Carlo samples to be used for integration in the burn-in phase of the MCEM. Default is nMC=100K.

nMCscale integer: the scale factor for the increase in Monte Carlo size when Monte Carlo has not reduced from the previous iteration. Default is nMCscale=5.

nMCmax integer: the maximum number of Monte Carlo samples that the algorithm is allowed to reach. Default is nMCmax=20000.

burnin integer: the number of iterations for 'burn-in' phase of the optimization algorithm. It is computationally inefficient to use a large number of Monte Carlo samples early on until one is approximately near the maximum likelihood estimate. Default is burnin=100K for type='antithetic' or type='montecarlo' and burnin=5 for type='sobol' or type='halton'. For standard methods, such a large burn-in will generally be unnecessary and can be reduced on an application-specific basis.

mcmaxIter integer: the maximum number of MCEM algorithm iterations allowed. Default is mcmaxIter=burnin+200.

convCrit character string: the convergence criterion to be used. See **Details**.

gammaOpt character string: by default (gammaOpt='NR'), γ is updated using a one-step Newton-Raphson iteration, with the Hessian matrix calculated exactly. If gammaOpt='GN', a Gauss-Newton algorithm-type iteration is implemented, where the Hessian matrix is approximated based on calculations

similar to those used for calculating the empirical information matrix? If it is used, then the step-length is adjusted by a nominal scaling parameter of 0.5 in order to reduce the chance of over-shooting the maximizer.

`tol0` numeric: tolerance value for convergence in the parameters; see **Details**. Default is `tol0=1e-03`.

`tol1` numeric: tolerance value for convergence in the parameters; see **Details**. Default is `tol1=1e-03`.

`tol2` numeric: tolerance value for convergence in the parameters; see **Details**. Default is `tol2=5e-03` for `type='antithetic'` or `type='montecarlo'` and `tol2=1e-03` for `type='sobol'` or `type='halton'`.

`tol.em` numeric: tolerance value for convergence in the multivariate linear mixed model (MV-LMM). When $K > 1$, the optimal initial parameters are those from the MV-LMM, which is estimated using a separate EM algorithm. Since both the E- and M-steps are available in closed-form, this algorithm convergences relatively rapidly with a high precision. Default is min(`1e-04`, `tol2`).

`rav` numeric: threshold when using `convCrit='sas'` that applies absolute change (when $<$`rav`) or relative change (when $\geq$`rav`) criterion; see **Details**. Default is `0.1`, which is an order of magnitude higher than the SAS implementation.

`type` character: type of Monte Carlo integration method to use. Options are

- `type='montecarlo'` Vanilla Monte Carlo sampling.
- `type='antithetic'` Variance reduction method using antithetic simulation. This is the default option.
- `type='sobol'` Quasi-Monte Carlo with a low deterministic Sobol sequence with Owen-type scrambling.
- `type='halton'` Quasi-Monte Carlo with a low deterministic Halton sequence.

`progress` logical: should a progress bar be shown on the console to indicate the percentage of bootstrap iterations completed? Default is `progress=TRUE`.

`ncores` integer: if more than one core is available, then the `bootSE` function can run in parallel via the `foreach` function. By default, `ncores=1`, which defaults to serial mode. Note that if `ncores>1`, then `progress` is set to `FALSE` by default, as it is not possible to display progress bars for parallel processes at the current time.

`safe.boot` logical: should each bootstrap replication be wrapped in a `tryCatch` statement to catch errors (e.g. during the optimisation progress)? When model fitting throws errors, a new bootstrap sample is drawn for the current iteration and the model is re-fit; this process continues until a model fits successfully. Default is `FALSE`.

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

Details

Standard errors and confidence intervals are obtained by repeated fitting of the requisite joint model to bootstrap samples of the original longitudinal and time-to-event data. Note that bootstrap is done by sampling subjects, not individual records.

Value

An object of class bootSE.

Note

This function is computationally intensive. A dynamic progress bar is displayed showing the percentage of bootstrap models fitted. On computer systems with more than one core available, computational time can be reduced by passing the argument ncores (with integer value >1) to bootSE, which implements parallel processing via the foreach package. **Note:** if parallel processing is implemented, then the progress bar is not displayed.

Due to random sampling, an mjoint model fitted to some bootstrap samples may not converge within the specified control parameter settings. The bootSE code discards any models that failed to converge when calculating the standard errors and confidence intervals. If a large proportion of models have failed to converge, it is likely that it will need to be refitted with changes to the control arguments.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Efron B, Tibshirani R. *An Introduction to the Bootstrap.* 2000; Boca Raton, FL: Chapman & Hall/CRC.

See Also

mjoint for approximate standard errors.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

fit.boot <- bootSE(fit, 50, use.mle = TRUE, control = list(
    burnin = 25, convCrit = "either",
```

```
    tol0 = 6e-03, tol2 = 6e-03, mcmaxIter = 60))

## End(Not run)
```

`confint.mjoint` *Confidence intervals for model parameters of an* `mjoint` *object*

Description

This function computes confidence intervals for one or more parameters in a fitted `mjoint` object.

Usage

```
## S3 method for class 'mjoint'
confint(
  object,
  parm = c("Both", "Longitudinal", "Event"),
  level = 0.95,
  bootSE = NULL,
  ...
)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`parm`	a character string specifying which sub-model parameter confidence intervals should be returned for. Can be specified as `parm='Longitudinal'` (multivariate longitudinal sub-model), `parm='Event'` (time-to-event sub-model), or `parm='both'` (default).
`level`	the confidence level required. Default is `level=0.95` for a 95% confidence interval.
`bootSE`	an object inheriting from class `bootSE` for the corresponding model. If `bootSE=NULL`, the function will attempt to utilize approximate standard error estimates (if available) calculated from the empirical information matrix.
`...`	additional arguments; currently none are used.

Value

A matrix containing the confidence intervals for either the longitudinal, time-to-event, or both sub-models.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

McLachlan GJ, Krishnan T. *The EM Algorithm and Extensions.* Second Edition. Wiley-Interscience; 2008.

Henderson R, Diggle PJ, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics.* 2000; **1(4)**: 465-480.

Lin H, McCulloch CE, Mayne ST. Maximum likelihood estimation in the joint analysis of time-to-event and multiple longitudinal variables. *Stat Med.* 2002; **21**: 2369-2382.

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics.* 1997; **53(1)**: 330-339.

See Also

`mjoint`, `bootSE`, and `confint` for the generic method description.

Examples

```
## Not run:
# Fit a classical univariate joint model with a single longitudinal outcome
# and a single time-to-event outcome

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

gamma <- c(0.1059417, 1.0843359)
sigma2 <- 0.03725999
beta <- c(4.9988669999, -0.0093527634, 0.0004317697)
D <- matrix(c(0.128219108, -0.006665505, -0.006665505, 0.002468688),
            nrow = 2, byrow = TRUE)

set.seed(1)
fit1 <- mjoint(formLongFixed = log.lvmi ~ time + age,
    formLongRandom = ~ time | num,
    formSurv = Surv(fuyrs, status) ~ age,
    data = hvd,
    timeVar = "time",
    inits = list(gamma = gamma, sigma2 = sigma2, beta = beta, D = D),
    control = list(nMCscale = 2, burnin = 5)) # controls for illustration only

confint(fit1, parm = "Longitudinal")

## End(Not run)

## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
```

```
                        "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)
confint(fit2)

## End(Not run)
```

---

dynLong *Dynamic predictions for the longitudinal data sub-model*

Description

Calculates the conditional expected longitudinal values for a *new* subject from the last observation time given their longitudinal history data and a fitted `mjoint` object.

Usage

```
dynLong(
  object,
  newdata,
  newSurvData = NULL,
  u = NULL,
  type = "first-order",
  M = 200,
  scale = 1.6,
  ci,
  progress = TRUE,
  ntimes = 100,
  level = 1
)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`newdata`	a list of `data.frame` objects for each longitudinal outcome for a single new patient in which to interpret the variables named in the `formLongFixed` and `formLongRandom` formulae of `object`. As per `mjoint`, the `list` structure enables one to include multiple longitudinal outcomes with different measurement protocols. If the multiple longitudinal outcomes are measured at the same time points for each patient, then a `data.frame` object can be given instead of a `list`. It is assumed that each data frame is in long format.

newSurvData	a data.frame in which to interpret the variables named in the formSurv formulae from the mjoint object. This is optional, and if omitted, the data will be searched for in newdata. Note that no event time or censoring indicator data are required for dynamic prediction. Defaults to newSurvData=NULL.
u	an optional time that must be greater than the last observed measurement time. If omitted (default is u=NULL), then conditional failure probabilities are reported for *all* observed failure times in the mjoint object data from the last known follow-up time of the subject.
type	a character string for whether a first-order (type="first-order") or Monte Carlo simulation approach (type="simulated") should be used for the dynamic prediction. Defaults to the computationally faster first-order prediction method.
M	for type="simulated", the number of simulations to performs. Default is M=200.
scale	a numeric scalar that scales the variance parameter of the proposal distribution for the Metropolis-Hastings algorithm, which therefore controls the acceptance rate of the sampling algorithm.
ci	a numeric value with value in the interval $(0, 1)$ specifying the confidence interval level for predictions of type='simulated'. If missing, defaults to ci=0.95 for a 95% confidence interval. If type='first-order' is used, then this argument is ignored.
progress	logical: should a progress bar be shown on the console to indicate the percentage of simulations completed? Default is progress=TRUE.
ntimes	an integer controlling the number of points to discretize the extrapolated time region into. Default is ntimes=100.
level	an optional integer giving the level of grouping to be used in extracting the residuals from object. Level values increase from outermost to innermost grouping, with level 0 corresponding to the population model fit and level 1 corresponding to subject-specific model fit. Defaults to level=1.

Details

Dynamic predictions for the longitudinal data sub-model based on an observed measurement history for the longitudinal outcomes of a new subject are based on either a first-order approximation or Monte Carlo simulation approach, both of which are described in Rizopoulos (2011). Namely, given that the subject was last observed at time *t*, we calculate the conditional expectation of each longitudinal outcome at time *u* as

$$E[y_k(u)|T \geq t, y, \theta] \approx x^T(u)\beta_k + z^T(u)\hat{b}_k,$$

where T is the failure time for the new subject, and y is the stacked-vector of longitudinal measurements up to time *t*.

First order predictions

For type="first-order", $\hat{b}$ is the mode of the posterior distribution of the random effects given by

$$\hat{b} = \arg\max_b f(b|y, T \geq t; \theta).$$

The predictions are based on plugging in $\theta = \hat{\theta}$, which is extracted from the `mjoint` object.

Monte Carlo simulation predictions

For `type="simulated"`, θ is drawn from a multivariate normal distribution with means $\hat{\theta}$ and variance-covariance matrix both extracted from the fitted `mjoint` object via the `coef()` and `vcov()` functions. $\hat{b}$ is drawn from the the posterior distribution of the random effects

$$f(b|y, T \geq t; \theta)$$

by means of a Metropolis-Hasting algorithm with independent multivariate non-central *t*-distribution proposal distributions with non-centrality parameter $\hat{b}$ from the first-order prediction and variance-covariance matrix equal to `scale` $\times$ the inverse of the negative Hessian of the posterior distribution. The choice of `scale` can be used to tune the acceptance rate of the Metropolis-Hastings sampler. This simulation algorithm is iterated `M` times, at each time calculating the conditional survival probability.

Value

A list object inheriting from class `dynLong`. The list returns the arguments of the function and a list containing *K* `data.frames` of 2 columns, with first column (named `timeVar[k]`; see `mjoint`) denoting times and the second column (named `y.pred`) denoting the expected outcome at each time point.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

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

See Also

`mjoint`, `dynSurv`.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
```

```
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

hvd2 <- droplevels(hvd[hvd$num == 1, ])
dynLong(fit2, hvd2)
dynLong(fit2, hvd2, u = 7) # outcomes at 7-years only

out <- dynLong(fit2, hvd2, type = "simulated")
out

## End(Not run)
```

---

dynSurv *Dynamic predictions for the time-to-event data sub-model*

Description

Calculates the conditional time-to-event distribution for a *new* subject from the last observation time given their longitudinal history data and a fitted `mjoint` object.

Usage

```
dynSurv(
  object,
  newdata,
  newSurvData = NULL,
  u = NULL,
  horizon = NULL,
  type = "first-order",
  M = 200,
  scale = 2,
  ci,
  progress = TRUE
)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`newdata`	a list of `data.frame` objects for each longitudinal outcome for a single new patient in which to interpret the variables named in the `formLongFixed` and `formLongRandom` formulae of `object`. As per `mjoint`, the `list` structure enables one to include multiple longitudinal outcomes with different measurement protocols. If the multiple longitudinal outcomes are measured at the same time points for each patient, then a `data.frame` object can be given instead of a `list`. It is assumed that each data frame is in long format.

newSurvData	a data.frame in which to interpret the variables named in the formSurv formulae from the mjoint object. This is optional, and if omitted, the data will be searched for in newdata. Note that no event time or censoring indicator data are required for dynamic prediction. Defaults to newSurvData=NULL.
u	an optional time that must be greater than the last observed measurement time. If omitted (default is u=NULL), then conditional failure probabilities are reported for *all* observed failure times in the mjoint object data from the last known follow-up time of the subject.
horizon	an optional horizon time. Instead of specifying a specific time u relative to the time origin, one can specify a horizon time that is relative to the last known follow-up time. The prediction time is essentially equivalent to horizon + t_{obs}, where t_{obs} is the last known follow-up time where the patient had not yet experienced the event. Default is horizon=NULL. If horizon is non-NULL, then the output will be reported still in terms of absolute time (from origin), u.
type	a character string for whether a first-order (type="first-order") or Monte Carlo simulation approach (type="simulated") should be used for the dynamic prediction. Defaults to the computationally faster first-order prediction method.
M	for type="simulated", the number of simulations to performs. Default is M=200.
scale	a numeric scalar that scales the variance parameter of the proposal distribution for the Metropolis-Hastings algorithm, which therefore controls the acceptance rate of the sampling algorithm.
ci	a numeric value with value in the interval $(0, 1)$ specifying the confidence interval level for predictions of type='simulated'. If missing, defaults to ci=0.95 for a 95% confidence interval. If type='first-order' is used, then this argument is ignored.
progress	logical: should a progress bar be shown on the console to indicate the percentage of simulations completed? Default is progress=TRUE.

Details

Dynamic predictions for the time-to-event data sub-model based on an observed measurement history for the longitudinal outcomes of a new subject are based on either a first-order approximation or Monte Carlo simulation approach, both of which are described in Rizopoulos (2011). Namely, given that the subject was last observed at time *t*, we calculate the conditional survival probability at time $u > t$ as

$$P[T \geq u | T \geq t; y, \theta] \approx \frac{S(u|\hat{b}; \theta)}{S(t|\hat{b}; \theta)},$$

where T is the failure time for the new subject, y is the stacked-vector of longitudinal measurements up to time *t* and $S(u|\hat{b}; \theta)$ is the survival function.

First order predictions

For type="first-order", $\hat{b}$ is the mode of the posterior distribution of the random effects given by

$$\hat{b} = \arg\max_b f(b|y, T \geq t; \theta).$$

The predictions are based on plugging in $\theta = \hat{\theta}$, which is extracted from the `mjoint` object.

Monte Carlo simulation predictions

For `type="simulated"`, θ is drawn from a multivariate normal distribution with means $\hat{\theta}$ and variance-covariance matrix both extracted from the fitted `mjoint` object via the `coef()` and `vcov()` functions. $\hat{b}$ is drawn from the the posterior distribution of the random effects

$$f(b|y, T \geq t; \theta)$$

by means of a Metropolis-Hasting algorithm with independent multivariate non-central *t*-distribution proposal distributions with non-centrality parameter $\hat{b}$ from the first-order prediction and variance-covariance matrix equal to `scale` $\times$ the inverse of the negative Hessian of the posterior distribution. The choice of `scale` can be used to tune the acceptance rate of the Metropolis-Hastings sampler. This simulation algorithm is iterated `M` times, at each time calculating the conditional survival probability.

Value

A list object inheriting from class `dynSurv`. The list returns the arguments of the function and a `data.frame` with first column (named `u`) denoting times and the subsequent columns returning summary statistics for the conditional failure probabilities For `type="first-order"`, a single column named `surv` is appended. For `type="simulated"`, four columns named `mean`, `median`, `lower` and `upper` are appended, denoting the mean, median and lower and upper confidence intervals from the Monte Carlo draws. Additional objects are returned that are used in the intermediate calculations.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

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

Taylor JMG, Park Y, Ankerst DP, Proust-Lima C, Williams S, Kestin L, et al. Real-time individual predictions of prostate cancer recurrence using joint models. *Biometrics*. 2013; **69**: 206–13.

See Also

`mjoint`, `dynLong`, and `plot.dynSurv`.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]
```

```
fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

hvd2 <- droplevels(hvd[hvd$num == 1, ])
dynSurv(fit2, hvd2)
dynSurv(fit2, hvd2, u = 7) # survival at 7-years only

out <- dynSurv(fit2, hvd2, type = "simulated")
out

## End(Not run)
```

`epileptic.qol` *Quality of life data following epilepsy drug treatment*

Description

The SANAD (Standard and New Antiepileptic Drugs) study (Marson et al., 2007) is a randomised control trial of standard and new antiepileptic drugs, comparing effects on longer term clinical outcomes. Quality of life (QoL) data were collected by mail at baseline, 3 months, and at 1 and 2 years using validated measures. This data is a subset of the trial for 544 patients randomised to one of 2 drugs: carbamazepine and lamotrigine.

Usage

```
data(epileptic.qol)
```

Format

A data frame with 1853 observations on the following 9 variables:

`id` patients identifier; in total there are 544 patients.

`with.time` number of days between registration and the earlier of treatment failure or study analysis time.

`trt` a factor with levels `CBZ` and `LTG` denoting carbamazepine and lamotrigine, respectively.

`with.status` the reason for treatment failure. Coded as `0`=censored; `1`=unacceptable adverse effects; `2`=inadequate seizure control.

`time` the time the quality of life measures were recorded (days). The first measurement for each subject is the baseline measurement, however there was variability between the time taken to return the questionnaires; hence the reason this is non-zero. Similarly, the second, third, and fourth follow-up times, which were scheduled for 3-months, 1-year, and 2-years, respectively, also had variability in completion times.

`anxiety` a continuous measure of anxiety, as defined according to the NEWQOL (Newly Diagnosed Epilepsy Quality of Life) assessment. Higher scores are indicative of worse QoL.

`depress` a continuous measure of depression, as defined according to the NEWQOL (Newly Diagnosed Epilepsy Quality of Life) assessment. Higher scores are indicative of worse QoL.

`aep` a continuous measure of the Liverpool Adverse Events Profile (AEP), as defined according to the NEWQOL (Newly Diagnosed Epilepsy Quality of Life) assessment. Higher scores are indicative of worse QoL.

`with.status2` a binary indicator of composite treatment failure (for any reason), coded `status2=1`, or right-censoring `status2=0`.

Source

SANAD Trial: University of Liverpool. See Jacoby et al. (2015).

References

Jacoby A, Sudell M, Tudur Smith C, et al. Quality-of-life outcomes of initiating treatment with standard and newer antiepileptic drugs in adults with new-onset epilepsy: Findings from the SANAD trial. *Epilepsia*. 2015; **56(3)**: 460-472.

Marson AG, Appleton R, Baker GA, et al. A randomised controlled trial examining longer-term outcomes of standard versus new antiepileptic drugs. The SANAD Trial. *Health Technology Assessment*. 2007; **11(37)**.

Marson AG, Al-Kharusi AM, Alwaidh M, et al. The SANAD study of effectiveness of carbamazepine, gabapentin, lamotrigine, oxcarbazepine, or topiramate for treatment of partial epilepsy: an unblinded randomised controlled trial. *Lancet*. 2007; **365**: 2007-2013.

Abetz L, Jacoby A, Baker GA, et al. Patient-based assessments of quality of life in newly diagnosed epilepsy patients: validation of the NEWQOL. *Epilepsia*. 2000; **41**: 1119-1128.

See Also

`pbc2`, `heart.valve`, `renal`.

`fitted.mjoint` *Extract* `mjoint` *fitted values*

Description

The fitted values at level *i* are obtained by adding together the population fitted values (based only on the fixed effects estimates) and the estimated contributions of the random effects.

Usage

```
## S3 method for class 'mjoint'
fitted(object, level = 0, ...)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`level`	an optional integer giving the level of grouping to be used in extracting the fitted values from object. Level values increase from outermost to innermost grouping, with level 0 corresponding to the population fitted values and level 1 corresponding to subject-specific fitted values Defaults to level 0.
`...`	additional arguments; currently none are used.

Value

A `list` of length K with each element a vector of fitted values for the k-th longitudinal outcome.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

See Also

`mjoint`, `residuals.mjoint`

`fixef.mjoint` *Extract fixed effects estimates from an* `mjoint` *object*

Description

Extract fixed effects estimates from an `mjoint` object.

Usage

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

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`process`	character string: if `process='Longitudinal'` the fixed effects coefficients from the (multivariate) longitudinal sub-model are returned. Else, if `process='Event'`, the coefficients from the time-to-event sub-model are returned.
`...`	additional arguments; currently none are used.

Value

A named vector of length equal to the number of sub-model coefficients estimated.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics*. 1997; **53(1)**: 330-339.

See Also

`fixef` for the generic method description, and `ranef.mjoint`.

Examples

```
## Not run:
# Fit a classical univariate joint model with a single longitudinal outcome
# and a single time-to-event outcome

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

set.seed(1)
fit1 <- mjoint(formLongFixed = log.lvmi ~ time + age,
    formLongRandom = ~ time | num,
    formSurv = Surv(fuyrs, status) ~ age,
    data = hvd,
    timeVar = "time",
    control = list(nMCscale = 2, burnin = 5)) # controls for illustration only

fixef(fit1, process = "Longitudinal")
fixef(fit1, process = "Event")

## End(Not run)

## Not run:
# Fit a joint model with bivariate longitudinal outcomes
```

```
data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

fixef(fit2, process = "Longitudinal")
fixef(fit2, process = "Event")

## End(Not run)
```

`formula.mjoint` *Extract model formulae from an* `mjoint` *object*

Description

Extract model formulae from an `mjoint` object.

Usage

```
## S3 method for class 'mjoint'
formula(x, process = c("Longitudinal", "Event"), k = 1, ...)
```

Arguments

`x`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`process`	character string: if `process='Longitudinal'` a fixed effects formula from the (multivariate) longitudinal sub-model is returned for the `k`-th outcome. Else, if `process='Event'`, the time-to-event model formula is returned.
`k`	integer: a number between 1 and K (the total number of longitudinal outcomes) that specifies the longitudinal outcome of interest.
`...`	additional arguments; currently none are used.

Value

An object of class "formula" which contains a symbolic model formula for the separate sub-model fixed effect terms only.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics*. 1997; **53(1)**: 330-339.

See Also

`formula` for the generic method description, and `ranef.mjoint`.

`getVarCov.mjoint` *Extract variance-covariance matrix of random effects from an* `mjoint` *object*

Description

Extract variance-covariance matrix of random effects from an `mjoint` object.

Usage

```
## S3 method for class 'mjoint'
getVarCov(obj, ...)
```

Arguments

`obj`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`...`	additional arguments; currently none are used.

Value

A variance-covariance matrix.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

See Also

`getVarCov` for the generic method description.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

getVarCov(fit2)

## End(Not run)
```

---

heart.valve *Aortic valve replacement surgery data*

Description

This is longitudinal data on an observational study on detecting effects of different heart valves, differing on type of tissue, implanted in the aortic position. The data consists of longitudinal measurements on three different heart function outcomes, after surgery occurred. There are several baseline covariates available, and also survival data.

Usage

```
data(heart.valve)
```

Format

This is a data frame in the unbalanced format, that is, with one row per observation. The data consists in columns for patient identification, time of measurements, longitudinal multiple longitudinal measurements, baseline covariates, and survival data. The column names are identified as follows:

`num` number for patient identification.

`sex` gender of patient (`0`=Male and `1`=Female).

`age` age of patient at day of surgery (years).

`time` observed time point, with surgery date as the time origin (years).

`fuyrs` maximum follow up time, with surgery date as the time origin (years).

`status` censoring indicator (`1`=died and `0`=lost at follow up).

`grad` valve gradient at follow-up visit.

`log.grad` natural log transformation of `grad`.

`lvmi` left ventricular mass index (standardised) at follow-up visit.

`log.lvmi` natural log transformation of `lvmi`.

`ef` ejection fraction at follow-up visit.

`bsa` preoperative body surface area.

`lvh` preoperative left ventricular hypertrophy.

`prenyha` preoperative New York Heart Association (NYHA) classification (`1`=I/II and `3`=III/IV).

`redo` previous cardiac surgery.

`size` size of the valve (millimeters).

`con.cabg` concomitant coronary artery bypass graft.

`creat` preoperative serum creatinine (μmol/mL).

`dm` preoperative diabetes.

`acei` preoperative use of ace inhibitor.

`lv` preoperative left ventricular ejection fraction (LVEF) (`1`=good, `2`=moderate, and `3`=poor).

`emergenc` operative urgency (`0`=elective, `1` = urgent, and `3`=emergency).

`hc` preoperative high cholesterol (`0`=absent, `1` =present treated, and `2`=present untreated).

`sten.reg.mix` aortic valve haemodynamics (`1`=stenosis, `2`=regurgitation, `3`=mixed).

`hs` implanted aortic prosthesis type (`1`=homograft and `0`=stentless porcine tissue).

References

Lim E, Ali A, Theodorou P, Sousa I, Ashrafian H, Chamageorgakis T, Duncan M, Diggle P, Pepper J. A longitudinal study of the profile and predictors of left ventricular mass regression after stentless aortic valve replacement. *Ann Thorac Surg.* 2008; **85(6)**: 2026-2029.

See Also

`pbc2`, `renal`, `epileptic.qol`.

`joineRML` *joineRML*

Description

joineRML is an extension of the joineR package for fitting joint models of time-to-event data and multivariate longitudinal data. The model fitted in joineRML is an extension of the Wulfsohn and Tsiatis (1997) and Henderson et al. (2000) models, which is comprised of $(K + 1)$-sub-models: a Cox proportional hazards regression model (Cox, 1972) and a K-variate linear mixed-effects model - a direct extension of the Laird and Ware (1982) regression model. The model is fitted using a Monte Carlo Expectation-Maximization (MCEM) algorithm, which closely follows the methodology presented by Lin et al. (2002).

References

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics.* 1997; **53(1)**: 330-339.

Henderson R, Diggle PJ, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics.* 2000; **1(4)**: 465-480.

Cox DR. Regression models and life-tables. *J R Stat Soc Ser B Stat Methodol.* 1972; **34(2)**: 187-220.

Laird NM, Ware JH. Random-effects models for longitudinal data. *Biometrics.* 1982; **38(4)**: 963-974.

`logLik.mjoint` *Extract log-likelihood from an* `mjoint` *object*

Description

Extract log-likelihood from an `mjoint` object.

Usage

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

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`...`	additional arguments; currently none are used.

Value

Returns an object of class `logLik`. This is a number with two attributes: `df` (degrees of freedom), giving the number of parameters in the model, and `nobs`, the number of observations used in estimation.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Henderson R, Diggle PJ, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics.* 2000; **1(4)**: 465-480.

See Also

`logLik` for the generic method description.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]
fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

logLik(fit2)

## End(Not run)
```

`mjoint` *Fit a joint model to time-to-event data and multivariate longitudinal data*

Description

This function fits the joint model proposed by Henderson et al. (2000), but extended to the case of multiple continuous longitudinal measures. The time-to-event data is modelled using a Cox proportional hazards regression model with time-varying covariates. The multiple longitudinal outcomes are modelled using a multivariate version of the Laird and Ware linear mixed model. The association is captured by a multivariate latent Gaussian process. The model is estimated using a (quasi-) Monte Carlo Expectation Maximization (MCEM) algorithm.

Usage

```
mjoint(
  formLongFixed,
  formLongRandom,
  formSurv,
  data,
  survData = NULL,
  timeVar,
  inits = NULL,
  verbose = FALSE,
  pfs = TRUE,
  control = list(),
  ...
)
```

Arguments

formLongFixed	a list of formulae for the fixed effects component of each longitudinal outcome. The left-hand side defines the response, and the right-hand side specifies the fixed effect terms. If a single formula is given (either as a list of length 1 or a formula), then it is assumed that a standard univariate joint model is being fitted.
formLongRandom	a list of one-sided formulae specifying the model for the random effects effects of each longitudinal outcome. The length of the list must be equal to formLongFixed.
formSurv	a formula specifying the proportional hazards regression model (not including the latent association structure). See coxph for examples.
data	a list of data.frame objects for each longitudinal outcome in which to interpret the variables named in the formLongFixed and formLongRandom. The list structure enables one to include multiple longitudinal outcomes with different measurement protocols. If the multiple longitudinal outcomes are measured at the same time points for each patient, then a data.frame object can be given instead of a list. It is assumed that each data frame is in long format. tibble objects are automatically converted to plain data.frame objects.
survData	a data.frame in which to interpret the variables named in the formSurv. This is optional, and if not given, the required data is searched for in data. Default is survData=NULL.
timeVar	a character string indicating the time variable in the linear mixed effects model. If there are multiple longitudinal outcomes and the time variable is labelled differently in each model, then a character string vector can be given instead.
inits	a list of initial values for some or all of the parameters estimated in the model. Default is NULL, with initial values estimated using separate multivariate linear mixed effects and Cox proportional hazard regression models.
verbose	logical: if TRUE, the parameter estimates and other convergence statistics are value are printed at each iteration of the MCEM algorithm. Default is FALSE.
pfs	logical: if TRUE, then assuming the MCEM algorithm has converged, post-fit statistics including the posterior means and variances of the random effects, and the approximate standard errors are calculated and returned as part of the model object. Default is TRUE. If FALSE, then these additional calculations are not performed, which can reduce the overall computational time. This option is intended to be used with computationally intensive routines such as simulation and bootstrap standard error estimation where these calculations are not required.
control	a list of control values with components:
	nMC integer: the initial number of Monte Carlo samples to be used for integration in the burn-in phase of the MCEM. Default is nMC=100K.
	nMCscale integer: the scale factor for the increase in Monte Carlo size when Monte Carlo has not reduced from the previous iteration. Default is nMCscale=5.
	nMCmax integer: the maximum number of Monte Carlo samples that the algorithm is allowed to reach. Default is nMCmax=20000.
	burnin integer: the number of iterations for 'burn-in' phase of the optimization algorithm. It is computationally inefficient to use a large number of

Monte Carlo samples early on until one is approximately near the maximum likelihood estimate. Default is `burnin`=100K for `type='antithetic'` or `type='montecarlo'` and `burnin`=5 for `type='sobol'` or `type='halton'`. For standard methods, such a large burn-in will generally be unnecessary and can be reduced on an application-specific basis.

`mcmaxIter` integer: the maximum number of MCEM algorithm iterations allowed. Default is `mcmaxIter=burnin+200`.

`convCrit` character string: the convergence criterion to be used. See **Details**.

`gammaOpt` character string: by default (`gammaOpt='NR'`), γ is updated using a one-step Newton-Raphson iteration, with the Hessian matrix calculated exactly. If `gammaOpt='GN'`, a Gauss-Newton algorithm-type iteration is implemented, where the Hessian matrix is approximated based on calculations similar to those used for calculating the empirical information matrix? If it is used, then the step-length is adjusted by a nominal scaling parameter of 0.5 in order to reduce the chance of over-shooting the maximizer.

`tol0` numeric: tolerance value for convergence in the parameters; see **Details**. Default is `tol0=1e-03`.

`tol1` numeric: tolerance value for convergence in the parameters; see **Details**. Default is `tol1=1e-03`.

`tol2` numeric: tolerance value for convergence in the parameters; see **Details**. Default is `tol2=5e-03` for `type='antithetic'` or `type='montecarlo'` and `tol2=1e-03` for `type='sobol'` or `type='halton'`.

`tol.em` numeric: tolerance value for convergence in the multivariate linear mixed model (MV-LMM). When $K > 1$, the optimal initial parameters are those from the MV-LMM, which is estimated using a separate EM algorithm. Since both the E- and M-steps are available in closed-form, this algorithm convergences relatively rapidly with a high precision. Default is min(`1e-04`, `tol2`).

`rav` numeric: threshold when using `convCrit='sas'` that applies absolute change (when $<$`rav`) or relative change (when $\geq$`rav`) criterion; see **Details**. Default is `0.1`, which is an order of magnitude higher than the SAS implementation.

`type` character: type of Monte Carlo integration method to use. Options are

- `type='montecarlo'` Vanilla Monte Carlo sampling.
- `type='antithetic'` Variance reduction method using antithetic simulation. This is the default option.
- `type='sobol'` Quasi-Monte Carlo with a low deterministic Sobol sequence with Owen-type scrambling.
- `type='halton'` Quasi-Monte Carlo with a low deterministic Halton sequence.

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

Details

Function `mjoint` fits joint models for time-to-event and multivariate longitudinal data. A review of relevant statistical methodology for joint models of multivariate data is given in Hickey et al. (2016). This is a direct extension of the models developed in the seminal works of Wulfsohn and Tsiatis

(1997) and Henderson et al. (2000), with the calculations based largely on Lin et al. (2002) who also extended the model to multivariate joint data. A more detailed explanation of the model formulation is given in the technical vignette. Each longitudinal outcome is modelled according to a linear mixed model (LMM), akin to the models fit by `lme`, with independent and identically distributed Gaussian errors. The latent term in each model (specified by `formLongRandom`) is a linear combination of subject-specific zero-mean Gaussian random effects with outcome-specific variance components. We denote these as $W_{i1}(t, b_{i1}), W_{i2}(t, b_{i2}), \ldots, W_{iK}(t, b_{iK})$, for the *K*-outcomes. Usually, $W(t, b)$ will be specified as either b_0 (a random-intercepts model) or $b_0 + b_1 t$ (a random-intercepts and random-slopes model); however, more general structures are allowed The time-to-event model is modelled as per the usual Cox model formulation, with an additional (possibly) time-varying term given by

$$\gamma_{y1} W_{i1}(t, b_{i1}) + \gamma_{y2} W_{i2}(t, b_{i2}) + \cdots + \gamma_{yK} W_{iK}(t, b_{iK}),$$

where γ_y is a parameter vector of proportional latent association parameters of length *K* for estimation.

The optimization routine is based on a Monte Carlo Expectation Maximization algorithm (MCEM) algorithm, as described by Wei and Tanner (1990). With options for using antithetic simulation for variance reduction in the Monte Carlo integration, or quasi-Monte Carlo based on low order deterministic sequences.

Value

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

Convergence criteria

The routine internally scales and centers data to avoid overflow in the argument to the exponential function. These actions do not change the result, but lead to more numerical stability. Several convergence criteria are available:

`abs` the maximum absolute parameter change is $<$`tol0`. The baseline hazard parameters are not included in this convergence statistic.

`rel` the maximum (absolute) relative parameter change is $<$`tol2`. A small value (`tol1`) is added to the denominator of the relative change statistic to avoid numerical problems when the parameters are close to zero.

`either` *either* the `abs` or `rel` criteria are satisfied.

`sas` if $|\theta_p| <$`rav`, then the `abs` criteria is applied for the *l*-th parameter; otherwise, `rel` is applied. This is the approach used in the SAS EM algorithm program for missing data.

Due to the Monte Caro error, the algorithm could spuriously declare convergence. Therefore, we require convergence to be satisfied for 3 consecutive iterations. The algorithm starts with a low number of Monte Carlo samples in the 'burn-in' phase, as it would be computationally inefficient to use a large sample whilst far away from the true maximizer. After the algorithm moves out of this adaptive phase, it uses an automated criterion based on the coefficient of variation of the relative parameter change of the last 3 iterations to decide whether to increase the Monte Carlo sample size. See the technical vignette and Ripatti et al. (2002) for further details.

Standard error estimation

Approximate standard errors (SEs) are calculated (if `pfs=TRUE`). These are based on the empirical observed information function (McLachlan & Krishnan, 2008). Through simulation studies, we have found that this approximation does not work particularly well for $n < 100$ (where n is the number of subjects). In these cases, one would need to appeal to the bootstrap SE estimation approach. However, in practice, the reliability of the approximate SEs will depend of a multitude of factors, including but not limited to, the average number of repeated measurements per subject, the total number of events, and the convergence of the MCEM algorithm.

Bootstrap SEs are also available, however they are not calculated using the `mjoint` function due to the intense computational time. Instead, a separate function is available: `bootSE`, which takes the fitted joint model as its main argument. Given a fitted joint model (of class `mjoint`) and a bootstrap fit object (of class `bootSE`), the SEs reported in the model can be updated by running `summary(fit_obj, bootSE = boot_obj)`. For details, consult the `bootSE` documentation.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Henderson R, Diggle PJ, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics.* 2000; **1(4)**: 465-480.

Hickey GL, Philipson P, Jorgensen A, Kolamunnage-Dona R. Joint modelling of time-to-event and multivariate longitudinal outcomes: recent developments and issues. *BMC Med Res Methodol.* 2016; **16(1)**: 117.

Lin H, McCulloch CE, Mayne ST. Maximum likelihood estimation in the joint analysis of time-to-event and multiple longitudinal variables. *Stat Med.* 2002; **21**: 2369-2382.

McLachlan GJ, Krishnan T. *The EM Algorithm and Extensions.* Second Edition. Wiley-Interscience; 2008.

Ripatti S, Larsen K, Palmgren J. Maximum likelihood inference for multivariate frailty models using an automated Monte Carlo EM algorithm. *Lifetime Data Anal.* 2002; **8**: 349-360.

Wei GC, Tanner MA. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *J Am Stat Assoc.* 1990; **85(411)**: 699-704.

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics.* 1997; **53(1)**: 330-339.

See Also

`mjoint.object`, `bootSE`, `plot.mjoint`, `summary.mjoint`, `getVarCov.mjoint`, `simData`.

Examples

```
## Not run:
# Fit a classical univariate joint model with a single longitudinal outcome
# and a single time-to-event outcome

data(heart.valve)
```

```
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

set.seed(1)
fit1 <- mjoint(formLongFixed = log.lvmi ~ time + age,
    formLongRandom = ~ time | num,
    formSurv = Surv(fuyrs, status) ~ age,
    data = hvd,
    timeVar = "time",
    control = list(nMCscale = 2, burnin = 5)) # controls for illustration only
summary(fit1)

## End(Not run)

## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)
summary(fit2)

## End(Not run)

## Not run:
# Fit a univariate joint model and compare to the joineR package

data(pbc2)
pbc2$log.b <- log(pbc2$serBilir)

# joineRML package
fit.joineRML <- mjoint(
    formLongFixed = list("log.bil" = log.b ~ year),
    formLongRandom = list("log.bil" = ~ 1 | id),
    formSurv = Surv(years, status2) ~ age,
    data = pbc2,
    timeVar = "year")
summary(fit.joineRML)

# joineR package
pbc.surv <- UniqueVariables(pbc2, var.col = c("years", "status2"),
                            id.col = "id")
pbc.long <- pbc2[, c("id", "year", "log.b")]
pbc.cov <- UniqueVariables(pbc2, "age", id.col = "id")
```

```
pbc.jd <- jointdata(longitudinal = pbc.long, baseline = pbc.cov,
                    survival = pbc.surv, id.col = "id", time.col = "year")
fit.joineR <- joint(data = pbc.jd,
    long.formula = log.b ~ 1 + year,
    surv.formula = Surv(years, status2) ~ age,
    model = "int")
summary(fit.joineR)

## End(Not run)
```

`mjoint.object` *Fitted* `mjoint` *object*

Description

An object returned by the `mjoint` function, inheriting from class `mjoint` and representing a fitted joint model for multivariate longitudinal and time-to-event data. Objects of this class have methods for the generic functions `coef`, `logLik`, `plot`, `print`, `ranef`, `fixef`, `summary`, `AIC`, `getVarCov`, `vcov`, `confint`, `sigma`, `fitted`, `residuals`, and `formula`.

Usage

```
mjoint.object
```

Format

An object of class `NULL` of length 0.

Value

A list with the following components.

`coefficients` a list with the estimated coefficients. The components of this list are:

- `beta` the vector of fixed effects for the linear mixed effects sub-model.
- `D` the variance-covariance matrix of the random effects.
- `sigma2` the measurement error standard deviations for the linear mixed effects sub-model.
- `haz` the estimated baseline hazard values for each unique failure time. Note that this is the *centered* hazard, equivalent to that returned by `coxph.detail`.
- `gamma` the vector of baseline covariates for the survival model and the latent association coefficient parameter estimates.

`history` a matrix with parameter estimates at each iteration of the MCEM algorithm.

`nMC.hx` a vector with the number of Monte Carlo samples for each MCEM algorithm iteration.

`formLongFixed` a list of formulae for the fixed effects component of each longitudinal outcome.

`formLongRandom` a list of formulae for the fixed effects component of each longitudinal outcome. The length of the list will be equal to `formLongFixed`.

`formSurv` a formula specifying the proportional hazards regression model (not including the latent association structure).

`data` a list of data.frames for each longitudinal outcome.

`survData` a data.frame of the time-to-event dataset.

`timeVar` a character string vector of length K denoting the column name(s) for time in `data`.

`id` a character string denoting the column name for subject IDs in `data` and `survData`.

`dims` a list giving the dimensions of model parameters with components:

- `p` a vector of the number of fixed effects for each longitudinal outcome.
- `r` a vector of the number of random effects for each longitudinal outcome.
- `K` an integer of the number of different longitudinal outcome types.
- `q` an integer of the number of baseline covariates in the time-to-event sub-model.
- `n` an integer of the total number of subjects in the study.
- `nk` a vector of the number of measurements for each longitudinal outcome.

`sfit` an object of class `coxph` for the separate time-to-event model fit. See `coxph` for details.

`lfit` a list of objects each of class `lme` from fitting separate linear mixed effects models; one per each longitudinal outcome type. See `lme` for details.

`log.lik0` the combined log-likelihood from separate sub-model fits.

`log.lik` the log-likelihood from the joint model fit.

`ll.hx` a vector of the log-likelihood values for each MCEM algorithm interaction.

`control` a list of control parameters used in the estimation of the joint model. See `mjoint` for details.

`finalnMC` the final number of Monte Carlo samples required prior to convergence.

`call` the matched call.

`inits` the initial values passed as an argument in the `mjoint` function.

`inits.long` the computed initial values from fitting a multivariate longitudinal model.

`inits.surv` the computed initial values from fitting a Cox proportional hazards model with time-dependent covariates calculated from the fitted multivariate LME model.

`conv` logical: did the MCEM algorithm converge within the specified maximum number of iterations?

`comp.time` a vector of length 2 with each element an object of class `difftime` that reports the *total* time taken for model fitting (including all stages) and the time spent in the *EM algorithm*.

Post model fit statistics

If `pfs=TRUE`, indicating that post-fit statistics are to be returned, then the output also includes the following objects.

`vcov` the variance-covariance matrix of model parameters, as approximated by the empirical information matrix, is reported. See `mjoint` for details.

`SE.approx` the square-root of the diagonal of `vcov` is returned, which are estimates of the standard errors for the parameters.

`Eb` a matrix with the estimated random effects values for each subject.

`Vb` an array with the estimated variance-covariance matrices for the random effects values for each subject.

`dmats` a list of length 3 containing the design matrices, data frames, and vectors used in the MCEM algorithm. These are required for prediction and to calculate the residuals and . The 3 items in the list are `l` (longitudinal data), `t` (time-to-event data), and `z` (design matrices expanded over unique failure times). These are not intended to be extracted by the user.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

See Also

`mjoint`.

`mjoint_tidiers` *Tidying methods for joint models for time-to-event data and multivariate longitudinal data*

Description

These methods tidy the coefficients of joint models for time-to-event data and multivariate longitudinal data of the `mjoint` class from the `joineRML` package.

Usage

```
## S3 method for class 'mjoint'
tidy(
  x,
  component = "survival",
  bootSE = NULL,
  conf.int = FALSE,
  conf.level = 0.95,
  ...
)

## S3 method for class 'mjoint'
augment(x, data = x$data, ...)

## S3 method for class 'mjoint'
glance(x, ...)
```

Arguments

`x`	An object of class `mjoint`.
`component`	Either `survival` (the survival component of the model, default) or `longitudinal` (the longitudinal component).

bootSE	An object of class bootSE for the corresponding model. If bootSE = NULL (the default), the function will use approximate standard error estimates calculated from the empirical information matrix.
conf.int	Include (1 - conf.level)% confidence intervals? Defaults to FALSE.
conf.level	The confidence level required.
...	extra arguments (not used)
data	Original data this was fitted on, in a list (e.g. list(data)). This will be extracted from x if not given.

Value

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

tidy returns one row for each estimated fixed effect depending on the component parameter. It contains the following columns:

term	The term being estimated
estimate	Estimated value
std.error	Standard error
statistic	Z-statistic
p.value	P-value computed from Z-statistic
conf.low	The lower bound of a confidence interval on estimate, if required
conf.high	The upper bound of a confidence interval on estimate, if required

.

augment returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns:

.fitted_j_0	population-level fitted values for the j-th longitudinal process
.fitted_j_1	individuals-level fitted values for the j-th longitudinal process
.resid_j_0	population-level residuals for the j-th longitudinal process
.resid_j_1	individual-level residuals for the j-th longitudinal process

See fitted.mjoint and residuals.mjoint for more information on the difference between population-level and individual-level fitted values and residuals.

glance returns one row with the columns

sigma2_j	the square root of the estimated residual variance for the j-th longitudinal process
AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion
logLik	the data's log-likelihood under the model

.

Note

If fitting a joint model with a single longitudinal process, please make sure you are using a named `list` to define the formula for the fixed and random effects of the longitudinal submodel.

Author(s)

Alessandro Gasparini (<alessandro.gasparini@ki.se>)

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes
library(joineRML)
data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) &
  !is.na(heart.valve$log.lvmi) &
  heart.valve$num <= 50, ]
fit <- mjoint(
  formLongFixed = list(
    "grad" = log.grad ~ time + sex + hs,
    "lvmi" = log.lvmi ~ time + sex
  ),
  formLongRandom = list(
    "grad" = ~ 1 | num,
    "lvmi" = ~ time | num
  ),
  formSurv = Surv(fuyrs, status) ~ age,
  data = hvd,
  inits = list("gamma" = c(0.11, 1.51, 0.80)),
  timeVar = "time"
)

# Extract the survival fixed effects
tidy(fit)

# Extract the longitudinal fixed effects
tidy(fit, component = "longitudinal")

# Extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# Extract the survival fixed effects with confidence intervals based on
# bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, bootSE = bSE, ci = TRUE)

# Augment original data with fitted longitudinal values and residuals
hvd2 <- augment(fit)

# Extract model statistics
glance(fit)
```

```
## End(Not run)
```

---

pbc2 *Mayo Clinic primary biliary cirrhosis data*

Description

This data is from the Mayo Clinic trial in primary biliary cirrhosis (PBC) of the liver conducted between 1974 and 1984. A total of 424 PBC patients, referred to Mayo Clinic during that ten-year interval met eligibility criteria for the randomized placebo controlled trial of the drug D-penicillamine, but only the first 312 cases in the data set participated in the randomized trial. Therefore, the data here are for the 312 patients with largely complete data.

Usage

```
data(pbc2)
```

Format

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

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

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

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

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

`age` at registration in years.

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

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

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

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

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

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

`serBilir` serum bilirubin in mg/dl.

`serChol` serum cholesterol in mg/dl.

`albumin` albumin in mg/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.

Source

`pbc2` and `pbc`.

References

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

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

See Also

`heart.valve`, `renal`, `epileptic.qol`.

`plot.dynLong` *Plot a* `dynLong` *object*

Description

Plots the conditional longitudinal expectations for a *new* subject calculated using the `dynLong` function.

Usage

```
## S3 method for class 'dynLong'
plot(x, main = NULL, xlab = NULL, ylab = NULL, grid = TRUE, estimator, ...)
```

Arguments

`x`	an object of class `dynLong` calculated by the `dynLong` function.
`main`	an overall title for the plot: see `title`.
`xlab`	a title for the x [time] axis: see `title`.
`ylab`	a character vector of the titles for the K longitudinal outcomes y-axes: see `title`.
`grid`	adds a rectangular grid to an existing plot: see `grid`.
`estimator`	a character string that can take values `mean` or `median` to specify what prediction statistic is plotted from an objecting inheritting of class `dynSurv`. Default is `estimator='median'`. This argument is ignored for non-simulated `dynSurv` objects, i.e. those of `type='first-order'`, as in that case a mode-based prediction is plotted.
`...`	additional plotting arguments; currently limited to `lwd` and `cex`. See `par` for details.

Value

A dynamic prediction plot.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

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

See Also

dynLong

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

hvd2 <- droplevels(hvd[hvd$num == 1, ])
out <- dynLong(fit2, hvd2)
plot(out, main = "Patient 1")

## End(Not run)
```

---

plot.dynSurv *Plot a* dynSurv *object*

Description

Plots the conditional time-to-event distribution for a *new* subject calculated using the dynSurv function.

Usage

```
## S3 method for class 'dynSurv'
plot(
  x,
  main = NULL,
  xlab = NULL,
  ylab1 = NULL,
  ylab2 = NULL,
  grid = TRUE,
  estimator,
  smooth = FALSE,
  ...
)
```

Arguments

x	an object of class dynSurv calculated by the dynSurv function.
main	an overall title for the plot: see title.
xlab	a title for the x [time] axis: see title.
ylab1	a character vector of the titles for the K longitudinal outcomes y-axes: see title.
ylab2	a title for the event-time outcome axis: see title.
grid	adds a rectangular grid to an existing plot: see grid.
estimator	a character string that can take values mean or median to specify what prediction statistic is plotted from an objecting inheritting of class dynSurv. Default is estimator='median'. This argument is ignored for non-simulated dynSurv objects, i.e. those of type='first-order', as in that case a mode-based prediction is plotted.
smooth	logical: whether to overlay a smooth survival curve (see **Details**). Defaults to FALSE.
...	additional plotting arguments; currently limited to lwd and cex. See par for details.

Details

The joineRML package is based on a semi-parametric model, such that the baseline hazards function is left unspecified. For prediction, it might be preferable to have a smooth survival curve. Rather than changing modelling framework *a prior*, a constrained B-splines non-parametric median quantile curve is estimated using cobs, with a penalty function of $\lambda = 1$, and subject to constraints of monotonicity and $S(t) = 1$.

Value

A dynamic prediction plot.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Ng P, Maechler M. A fast and efficient implementation of qualitatively constrained quantile smoothing splines. *Statistical Modelling*. 2007; **7(4)**: 315-328.

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

See Also

dynSurv

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

hvd2 <- droplevels(hvd[hvd$num == 1, ])
out1 <- dynSurv(fit2, hvd2)
plot(out1, main = "Patient 1")

## End(Not run)

## Not run:
# Monte Carlo simulation with 95% confidence intervals on plot

out2 <- dynSurv(fit2, hvd2, type = "simulated", M = 200)
plot(out2, main = "Patient 1")

## End(Not run)
```

plot.mjoint *Plot diagnostics from an* mjoint *object*

Description

Plot diagnostics from an mjoint object.

Usage

```
## S3 method for class 'mjoint'
plot(x, type = "convergence", ...)
```

Arguments

x	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.
type	currently the only option is type='convergence' for graphical examination of convergence over MCEM iteration.
...	other parameters passed to plotConvergence.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

See Also

plot.default, par, abline.

Examples

```
## Not run:
# Fit a classical univariate joint model with a single longitudinal outcome
# and a single time-to-event outcome

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

set.seed(1)
fit1 <- mjoint(formLongFixed = log.lvmi ~ time + age,
    formLongRandom = ~ time | num,
    formSurv = Surv(fuyrs, status) ~ age,
    data = hvd,
    timeVar = "time",
    control = list(nMCscale = 2, burnin = 5)) # controls for illustration only

plot(fit1, param = "beta")  # LMM fixed effect parameters
plot(fit1, param = "gamma") # event model parameters
```

```
## End(Not run)

## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    control = list(burnin = 50),
    verbose = TRUE)

plot(fit2, type = "convergence", params = "gamma")

## End(Not run)
```

`plot.ranef.mjoint` *Plot a* `ranef.mjoint` *object*

Description

Displays a plot of the BLUPs and approximate 95% prediction interval for each subject.

Usage

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

Arguments

`x`	an object inheriting from class `ranef.mjoint`, representing the estimated random effects for the `mjoint` object from which it was produced.
`...`	additional arguments; currently none are used.

Value

an object inheriting from class `ggplot`, which displays a trellis plot with a separate panel for each effect, showing a dotplot (with optional error bars indicating approximate 95% prediction intervals if the argument `postVar=TRUE` is set in the call to `ranef`) for each subject (by row).

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

See Also

ranef.mjoint.

Examples

```
## Not run:
require(ggplot2)
data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]
set.seed(1)

fit1 <- mjoint(formLongFixed = log.lvmi ~ time,
    formLongRandom = ~ time | num,
    formSurv = Surv(fuyrs, status) ~ 1,
    data = hvd,
    timeVar = "time")

plot(ranef(fit1, postVar = TRUE))

## End(Not run)
```

---

plotConvergence *Plot convergence time series for parameter vectors from an* mjoint *object*

Description

Plot convergence time series for parameter vectors from an mjoint object.

Usage

```
plotConvergence(object, params = "gamma", discard = FALSE)
```

Arguments

object	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.

params	a string indicating what parameters are to be shown. Options are params='gamma' for the time-to-event sub-model covariate coefficients, including the latent association parameters; params='beta' for the longitudinal sub-model fixed effects coefficients; params='sigma2' for the residual error variances from the longitudinal sub-model; params='D' for the lower triangular matrix of the variance-covariance matrix of random effects; params='loglik' for the log-likelihood.
discard	logical; if TRUE then the 'burn-in' phase iterations of the MCEM algorithm are discarded. Default is discard=FALSE.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Wei GC, Tanner MA. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *J Am Stat Assoc.* 1990; **85(411)**: 699-704.

See Also

plot.mjoint, plot.default, par, abline.

---

ranef.mjoint *Extract random effects estimates from an* mjoint *object*

Description

Extract random effects estimates from an mjoint object.

Usage

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

Arguments

object	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.
postVar	logical: if TRUE the variance of the posterior distribution is also returned.
...	additional arguments; currently none are used.

Value

A data.frame (also of class ranef.mjoint) 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)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS.* New York: Springer Verlag; 2000.

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics.* 1997; **53(1)**: 330-339.

See Also

`ranef` for the generic method description, and `fixef.mjoint`. To plot `ranef.mjoint` objects, see `plot.ranef.mjoint`.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

ranef(fit2)

## End(Not run)
```

---

renal *Renal transplantation data*

Description

This is a dataset on 407 patients suffering from chronic kidney disease who underwent a primary renal transplantation with a graft from a deceased or living donor in the University Hospital of the Catholic University of Leuven (Belgium) between 21 January 1983 and 16 August 2000. Chronic kidney (renal) disease is a progressive loss of renal function over a period of months or years through five stages. Each stage is a progression through an abnormally low and progressively worse glomerular filtration rate (GFR). The dataset records 3 repeated measures (2 continuous and 1 binary), and an event time.

Usage

```
data(renal)
```

Format

This is a list with 4 data frames:

1. `prot`: repeated measurement data for proteinuria (binary) that measures whether the kidneys succeed in sustaining the proteins in the blood and not discard them in the urine.
2. `haem`: repeated measurement data for blood haematocrit level (continuous) that measures whether the kidneys produce adequate amounts of the hormone erythropoietin that regulates the red blood cell production.
3. `gfr`: repeated measurement data for GFR (continuous) that measures the filtration rate of the kidneys.
4. `surv`: time-to-event data for renal graft failure.

All datasets have the common data columns, which are in long format for the 3 longitudinal data data frames, and 1-per-subject for the time-to-event data frame:

`id` number for patient identification.

`age` age of patient at day of surgery (years).

`weight` preoperative weight of patient (kg).

`sex` gender of patient.

`fuyears` maximum follow up time, with transplant date as the time origin (years).

`failure` censoring indicator (`1`=graft failure and `0`=censored).

The longitudinal datasets only contain 2 further columns:

`time` observed time point, with surgery date as the time origin (years).

biomarker value a recorded measurement of the biomarker taken at time `time`. The 3 biomarkers (one per data frame) are:

- `proteinuria`: recorded as binary indicator: present or not-present. Present in the `prot` data.
- `haematocrit`: recorded as percentage (%) of the ratio of the volume of red blood cells to the total volume of blood. Present in the `haem` data.
- `gfr`: measured as ml/min/1.73m^2. Present in the `gfr` data.

Source

Dr Dimitris Rizopoulos (<d.rizopoulos@erasmusmc.nl>).

References

Rizopoulos D, Ghosh, P. A Bayesian semiparametric multivariate joint model for multiple longitudinal outcomes and a time-to-event. *Stat Med.* 2011; **30(12)**: 1366-80.

See Also

`pbc2`, `heart.valve`, `epileptic.qol`.

`residuals.mjoint` *Extract* `mjoint` *residuals*

Description

The residuals at level i are obtained by subtracting the fitted levels at that level from the response vector.

Usage

```
## S3 method for class 'mjoint'
residuals(object, level = 0, ...)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`level`	an optional integer giving the level of grouping to be used in extracting the residuals from object. Level values increase from outermost to innermost grouping, with level 0 corresponding to the population residuals and level 1 corresponding to subject-specific residuals. Defaults to `level=0`.
`...`	additional arguments; currently none are used.

Value

A `list` of length K with each element a vector of residuals for the k-th longitudinal outcome.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

See Also

`mjoint`, `fitted.mjoint`

sampleData *Sample from an* mjoint *object*

Description

Generic function used to sample a subset of data from an object of class mjoint with a specific number of subjects.

Usage

```
sampleData(object, size = NULL, replace = TRUE)
```

Arguments

object	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.
size	number of subjects to include in the sampled subset. If size=NULL (default), then size is set equal to the number of subjects used to fit the mjoint model.
replace	use replacement when sampling subjects? Default is TRUE. If replacement is used, then the subjects are re-labelled from 1 to size.

Details

This function is primarily intended for internal use in the bootSE function in order to permit bootstrapping. However, it can be used for other purposes given a fitted mjoint object.

Value

A list of 2 data.frames: one recording the requisite longitudinal outcomes data, and the other recording the time-to-event data.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

See Also

mjoint.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
```

```
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)
sampleData(fit2, size = 10)

## End(Not run)
```

`sigma.mjoint` *Extract residual standard deviation(s) from an* `mjoint` *object*

Description

Extract residual standard deviation(s) from an `mjoint` object.

Usage

```
## S3 method for class 'mjoint'
sigma(object, ...)
```

Arguments

`object`	an object inheriting from class `mjoint` for a joint model of time-to-event and multivariate longitudinal data.
`...`	additional arguments; currently none are used.

Value

a number (standard deviation) if $K = 1$ (univariate model), or a vector if $K > 1$ (multivariate model).

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*. New York: Springer Verlag; 2000.

See Also

`sigma` in the **lme4** package.

simData *Simulate data from a joint model*

Description

This function simulates multivariate longitudinal and time-to-event data from a joint model.

Usage

```
simData(
  n = 100,
  ntms = 5,
  beta = rbind(c(1, 1, 1, 1), c(1, 1, 1, 1)),
  gamma.x = c(1, 1),
  gamma.y = c(0.5, -1),
  sigma2 = c(1, 1),
  D = NULL,
  df = Inf,
  model = "intslope",
  theta0 = -3,
  theta1 = 1,
  censoring = TRUE,
  censlam = exp(-3),
  truncation = TRUE,
  trunctime = (ntms - 1) + 0.1
)
```

Arguments

n	the number of subjects to simulate data for.
ntms	the maximum number of (discrete) time points to simulate repeated longitudinal measurements at.
beta	a matrix of dim=c(K,4) specifying the coefficients of the fixed effects. The order in each row is intercept, time, a continuous covariate, and a binary covariate.
gamma.x	a vector of length=2 specifying the coefficients for the time-to-event baseline covariates, in the order of a continuous covariate and a binary covariate.
gamma.y	a vector of length=K specifying the latent association parameters for each longitudinal outcome.
sigma2	a vector of length=K specifying the residual standard errors.
D	a positive-definite matrix specifying the variance-covariance matrix. If model='int', the matrix has dimension dim=c(K, K), else if model='intslope', the matrix has dimension dim =c(2K,2K). If D=NULL (default), an identity matrix is assumed.
df	a non-negative scalar specifying the degrees of freedom for the random effects if sampled from a multivariate *t*-distribution. The default is df=Inf, which corresponds to a multivariate normal distribution.

model	follows the model definition in the joint function. See **Details** for choices.
theta0, theta1	parameters controlling the failure rate. See Details.
censoring	logical: if TRUE, includes an independent censoring time.
censlam	a scale (> 0) parameter for an exponential distribution used to simulate random censoring times for when censoring=TRUE.
truncation	logical: if TRUE, adds a truncation time for a maximum event time.
trunctime	a truncation time for use when truncation=TRUE.

Details

The function simData simulates data from a joint model, similar to that performed in Henderson et al. (2000). It works by first simulating multivariate longitudinal data for all possible follow-up times using random draws for the multivariate Gaussian random effects and residual error terms. Data can be simulated assuming either random-intercepts only in each of the longitudinal sub-models, or random-intercepts and random-slopes. Currently, all models must have the same structure. The failure times are simulated from proportional hazards time-to-event models using the following methodologies:

model="int" The baseline hazard function is specified to be an exponential distribution with

$$\lambda_0(t) = \exp \theta_0.$$

Simulation is conditional on known time-independent effects, and the methodology of Bender et al. (2005) is used to simulate the failure time.

model="intslope" The baseline hazard function is specified to be a Gompertz distribution with

$$\lambda_0(t) = \exp \theta_0 + \theta_1 t.$$

In the usual representation of the Gompertz distribution, θ_1 is the shape parameter, and the scale parameter is equivalent to $\exp(\theta_0)$. Simulation is conditional on on a predictable (linear) time-varying process, and the methodology of Austin (2012) is used to simulate the failure time.

Value

A list of 2 data.frames: one recording the requisite longitudinal outcomes data, and one recording the time-to-event data.

Author(s)

Pete Philipson (<peter.philipson1@newcastle.ac.uk>) and Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Austin PC. Generating survival times to simulate Cox proportional hazards models with time-varying covariates. *Stat Med.* 2012; **31(29)**: 3946-3958.

Bender R, Augustin T, Blettner M. Generating survival times to simulate Cox proportional hazards models. *Stat Med.* 2005; **24**: 1713-1723.

Henderson R, Diggle PJ, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics.* 2000; **1(4)**: 465-480.

Examples

```
beta <- rbind(c(0.5, 2, 1, 1),
c(2, 2, -0.5, -1))
D <- diag(4)
D[1, 1] <- D[3, 3] <- 0.5
D[1, 2] <- D[2, 1] <- D[3, 4] <- D[4, 3] <- 0.1
D[1, 3] <- D[3, 1] <- 0.01

sim <- simData(n = 250, beta = beta, D = D, sigma2 = c(0.25, 0.25),
               censlam = exp(-0.2), gamma.y = c(-.2, 1), ntms = 8)
```

---

summary.mjoint *Summary of an* mjoint *object*

Description

This function provides a summary of an mjoint object.

Usage

```
## S3 method for class 'mjoint'
summary(object, bootSE = NULL, ...)
```

Arguments

object	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.
bootSE	an object inheriting from class bootSE for the corresponding model. If bootSE=NULL, the function will attempt to utilize approximate standard error estimates (if available) calculated from the empirical information matrix.
...	additional arguments; currently none are used.

Value

A list containing the coefficient matrices for the longitudinal and time-to-event sub-models; variance-covariance matrix for the random effects; residual error variances; log-likelihood of joint model; AIC and BIC statistics; and model fit objects.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics.* 1997; **53(1)**: 330-339.

Henderson R, Diggle PJ, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics.* 2000; **1(4)**: 465-480.

Lin H, McCulloch CE, Mayne ST. Maximum likelihood estimation in the joint analysis of time-to-event and multiple longitudinal variables. *Stat Med.* 2002; **21**: 2369-2382.

See Also

`mjoint`, `mjoint.object`, and `summary` for the generic method description.

Examples

```
## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)
summary(fit2)

## End(Not run)
```

`vcov.mjoint` *Extract an approximate variance-covariance matrix of estimated parameters from an* `mjoint` *object*

Description

Returns the variance-covariance matrix of the main parameters of a fitted `mjoint` model object.

Usage

```
## S3 method for class 'mjoint'
vcov(object, correlation = FALSE, ...)
```

Arguments

object	an object inheriting from class mjoint for a joint model of time-to-event and multivariate longitudinal data.
correlation	logical: if TRUE returns the correlation matrix, otherwise returns the variance-covariance matrix (default).
...	additional arguments; currently none are used.

Details

This is a generic function that extracts the variance-covariance matrix of parameters from an mjoint model fit. It is based on a profile likelihood, so no estimates are given for the baseline hazard function, which is generally considered a nuisance parameter. It is based on the empirical information matrix (see Lin et al. 2002, and McLachlan and Krishnan 2008 for details), so is only approximate.

Value

A variance-covariance matrix.

Note

This function is not to be confused with getVarCov, which returns the extracted variance-covariance matrix for the random effects distribution.

Author(s)

Graeme L. Hickey (<graemeleehickey@gmail.com>)

References

Lin H, McCulloch CE, Mayne ST. Maximum likelihood estimation in the joint analysis of time-to-event and multiple longitudinal variables. *Stat Med.* 2002; **21**: 2369-2382.

McLachlan GJ, Krishnan T. *The EM Algorithm and Extensions*. Second Edition. Wiley-Interscience; 2008.

See Also

vcov for the generic method description, and cov2cor for details of efficient scaling of a covariance matrix into the corresponding correlation matrix.

Examples

```
## Not run:
# Fit a classical univariate joint model with a single longitudinal outcome
# and a single time-to-event outcome

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

set.seed(1)
```

```
fit1 <- mjoint(formLongFixed = log.lvmi ~ time + age,
    formLongRandom = ~ time | num,
    formSurv = Surv(fuyrs, status) ~ age,
    data = hvd,
    timeVar = "time",
    control = list(nMCscale = 2, burnin = 5)) # controls for illustration only

vcov(fit1)

## End(Not run)

## Not run:
# Fit a joint model with bivariate longitudinal outcomes

data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) & !is.na(heart.valve$log.lvmi), ]

fit2 <- mjoint(
    formLongFixed = list("grad" = log.grad ~ time + sex + hs,
                         "lvmi" = log.lvmi ~ time + sex),
    formLongRandom = list("grad" = ~ 1 | num,
                          "lvmi" = ~ time | num),
    formSurv = Surv(fuyrs, status) ~ age,
    data = list(hvd, hvd),
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time",
    verbose = TRUE)

vcov(fit2)

## End(Not run)
```

Index

∗ **datagen**
  sampleData, 48
  simData, 50
∗ **datasets**
  epileptic.qol, 16
  heart.valve, 22
  pbc2, 36
  renal, 45
∗ **dplot**
  plot.mjoint, 41
  plotConvergence, 43
∗ **hplot**
  plot.dynLong, 37
  plot.dynSurv, 38
∗ **methods**
  baseHaz, 3
  bootSE, 4
  confint.mjoint, 8
  fitted.mjoint, 17
  fixef.mjoint, 18
  formula.mjoint, 20
  getVarCov.mjoint, 21
  logLik.mjoint, 24
  mjoint, 25
  plot.mjoint, 41
  plot.ranef.mjoint, 42
  plotConvergence, 43
  ranef.mjoint, 44
  residuals.mjoint, 47
  sampleData, 48
  sigma.mjoint, 49
  summary.mjoint, 52
  vcov.mjoint, 53
∗ **multivariate**
  bootSE, 4
  dynLong, 10
  mjoint, 25
  mjoint.object, 31
  sampleData, 48
  simData, 50
∗ **survival**
  baseHaz, 3
  bootSE, 4
  dynSurv, 13
  mjoint, 25
  mjoint.object, 31
  sampleData, 48
  simData, 50

abline, *41*, *44*
augment.mjoint *(*mjoint_tidiers*)*, 33

baseHaz, 3
bootSE, 4, *9*, *29*, *48*

cobs, *39*
coef, *4*
confint, *9*
confint.mjoint, 8
cov2cor, *54*
coxph, *26*, *32*
coxph.detail, *3*, *31*

dynLong, 10, *15*, *37*, *38*
dynSurv, *12*, 13, *38–40*

epileptic.qol, 16, *23*, *37*, *46*

fitted.mjoint, 17, *34*, *47*
fixef, *19*
fixef.mjoint, 18, *45*
foreach, *6*, *7*
formula, *21*
formula.mjoint, 20

getVarCov, *21*, *54*
getVarCov.mjoint, 21, *29*
glance.mjoint *(*mjoint_tidiers*)*, 33
grid, *37*, *39*

heart.valve, *17*, 22, *37*, *46*

joineRML, 23
joint, *51*

lme, *28*, *32*
logLik, *24*
logLik.mjoint, 24

mjoint, *3*, *4*, *7*, *9*, *10*, *12*, *13*, *15*, *18*, 25, *32*, *33*, *47*, *48*, *53*
mjoint.object, *3*, *28*, *29*, 31, *53*
mjoint_tidiers, 33

par, *37*, *39*, *41*, *44*
pbc, *37*
pbc2, *17*, *23*, 36, *37*, *46*
plot.default, *41*, *44*
plot.dynLong, 37
plot.dynSurv, *15*, 38
plot.mjoint, *29*, 41, *44*
plot.ranef.mjoint, 42, *45*
plotConvergence, *41*, 43

ranef, *42*, *45*
ranef.mjoint, *19*, *21*, *43*, 44
renal, *17*, *23*, *37*, 45
residuals.mjoint, *18*, *34*, 47

sampleData, 48
sigma, *49*
sigma.mjoint, 49
simData, *29*, 50
summary, *53*
summary.mjoint, *29*, 52

tidy.mjoint *(*mjoint_tidiers*)*, 33
title, *37*, *39*
tryCatch, *6*

vcov, *54*
vcov.mjoint, 53