Package ‘gmvjoint’

July 22, 2025

Type Package

Title Joint Models of Survival and Multivariate Longitudinal Data

Version 0.4.5

Date 2024-10-05

Description Fit joint models of survival and multivariate longitudinal data. The longitudinal data is specified by generalised linear mixed models. The joint models are fit via maximum likelihood using an approximate expectation maximisation algorithm. Bernhardt (2015) <doi:10.1016/j.csda.2014.11.011>.

License GPL-3

Depends R (>= 3.6.0), glmmTMB, survival

Imports Rcpp (>= 1.0.6), MASS, methods, mvtnorm, pracma, reformulas, stats, statmod, xtable

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.2.3

LazyData true

URL `https://github.com/jamesmurray7/gmvjoint`

BugReports `https://github.com/jamesmurray7/gmvjoint/issues`

NeedsCompilation yes

Author James Murray [aut, cre]

Maintainer James Murray <`james.murray@lshtm.ac.uk`>

Repository CRAN

Date/Publication 2024-10-05 22:40:02 UTC

Contents

`anova.joint` *Anova for joint models*

Description

Perform a likelihood ratio test between two (**nested**) `joint` models. The user must decide whether the models are truly nested.

Usage

```
## S3 method for class 'joint'
anova(object, object2, ...)
```

Arguments

`object`	a joint model fit by the `joint` function. This should be **nested** in `object2`.
`object2`	a joint model fit by the `joint` function. This should be more complex than `object`.
`...`	additional arguments (none used).

Value

A list of class `anova.joint` with elements

`mod0`	the name of `object`.
`l0`	the log-likelihood of the nested model, i.e. fit under the null.
`AIC0`	AIC for `object`.
`BIC0`	BIC for `object`.
`mod1`	the name of `object2`.
`l1`	the log-likelihood under the alternative hypothesis.
`AIC1`	AIC for `object2`.
`BIC1`	BIC for `object2`.
`LRT`	likelihood ratio test statistic.
`p`	the p-value of `LRT`.
`warnSurv`	internal - logical value for printing difference in survival models.
`warnRanefs`	internal - logical value for printing difference in random effects specifications.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>)

See Also

`joint` and `logLik.joint`.

Examples

```
rm(list=ls())
data(PBC)
# Compare quadratic vs linear time specification for log(serum bilirubin) -----
PBC$serBilir <- log(PBC$serBilir)
long.formulas1 <- list(serBilir ~ drug * time + (1 + time|id))
long.formulas2 <- list(serBilir ~ drug * (time + I(time^2)) + (1 + time + I(time^2)|id))
surv.formula <- Surv(survtime, status) ~ drug
family <- list('gaussian')
# Fit the two competing models (fit is nested in fit2) ------------------------
fit <- joint(long.formulas1, surv.formula, PBC, family,
             control = list(verbose = FALSE))
fit2 <- joint(long.formulas2, surv.formula, PBC, family, control = list(verbose = FALSE))
anova(fit, fit2)
# Quadratic terms improve fit significantly.
```

boot.joint — *Bootstrapping a* joint *object*

Description

Use an existing model fit by joint along with the data object originally used and obtain a mean estimate, standard errors and 95% confidence interval using the bootstrap. The original data is resampled by subject, not by observation.

Usage

```
boot.joint(
  fit,
  data,
  boot.size = NULL,
  nboot = 100L,
  replace = TRUE,
  progress = TRUE,
  use.MLEs = TRUE,
  control = list()
)
```

Arguments

fit	a joint model fit by the joint function.
data	the original data used to fit the above joint model.
boot.size	integer, specifies the number of subjects to resample in the bootstrapping approach. The default value is boot.size = NULL which defaults to the number of unique subjects in the joint object.
nboot	integer, specifies the number of bootstrap samples, default value is nboot = 100L.
replace	logical, should sampling be done with replacement? Defaults to replace = TRUE.
progress	logical, should a text progress bar showing overall progress be shown and updated after each successful bootstrapped model fit? Defaults to progress=TRUE.
use.MLEs	logical, should the MLEs of the fit be used as initial conditions in each of the bootstrapped calls to joint? Defaults to use.MLEs=TRUE which should help reduce the computational burden in fitting these bootstrap replicate joint objects.
control	a list of control arguments, with same possible arguments as shown in joint. Note that by default the *same* control arguments used in the joint fit parameter are carried forwards, besides the items return.dmats, post.process, and verbose which are all set to FALSE in boot.joint in order to reduce memory overheads and computation time. Instead, the user could lessen computational burden of this intensive bootstrapping by changing convergence criteria items e.g. conv, tol.rel, tol.abs, tol.thr in order to speed-up convergence of the nboot individual bootstrapped model fits.

Value

A list of class `boot.joint` which contains the MLEs from supplied `joint` object, as well as the bootstrapped summaries and some model/computation information.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`joint` `vcov.joint`

Examples

```
# Bivariate fit on PBC data -----------------------------------------
data(PBC)

# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'albumin', 'platelets'))
PBC <- na.omit(PBC)

# Specify bivariate fit
long.formulas <- list(
  albumin ~ time*drug + (1 + time|id),
  platelets ~ time * drug + (1 + time|id)
)
surv.formula <- Surv(survtime, status) ~ drug

fit <- joint(long.formulas, surv.formula, PBC, family = list('gaussian', 'poisson'))
# Set 50 bootstraps, with lower absolute tolerance and convergence of 'either'.
BOOT <- boot.joint(fit, PBC, nboot = 50L, control = list(tol.abs = 5e-3, conv = 'either'),
                   use.MLEs = TRUE)
BOOT # Print to console via S3 method
```

`cond.ranefs` *Obtain conditional distribution of the random effects*

Description

Obtain the conditional distribution of the random effects of a `joint` model fit. This is achieved by a Metropolis scheme. Approximate normality across random effects is expected, and could be useful in diagnosing potential issues surrounding model fits.

Usage

```
cond.ranefs(fit, burnin = 500L, N = 3500L, tune = 2)
```

Arguments

`fit`	a joint model fit by the `joint` function.
`burnin`	Number of burn-in iterations to discard, defaults to 500.
`N`	Number of MC iterations to carry out *post* burn-in, defaults to 3500.
`tune`	Tuning parameter, problem-specific, defaults to 2.

Value

A list of class `cond.b.joint` containing:

walks A list of length `n` containing the history of b_i post burn-in.

acceptance A numeric vector containing the acceptance rate for each sampled subject.

M The ModelInfo list from `joint`. Used by S3 methods for class `cond.b.joint`.

bhats Posterior estimates at MLEs for the random effects. Same as `ranef(joint)`.

Sigmahats The covariances of `bhats`.

D The MLE estimate for the variance-covariance matrix of random effects from `fit`.

q Dimension of random effects.

K Number of responses.

qnames The names of the random effects as determined by call to `joint`.

burnin The amount of burn-in used.

N Number of MC iterations.

tune tuning parameter used

nobs The number of observations for each subject for each response.

elapsed.time Time taken for `cond.ranefs` to complete.

See Also

`ranef.joint` `plot.cond.b.joint`

Examples

```
dat <- simData()$data
long.formulas <- list(Y.1 ~ time + cont + bin + (1 + time|id),
                      Y.2 ~ time + cont + bin + (1 + time|id))
surv.formula <- Surv(survtime, status) ~ bin
fit <- joint(long.formulas, surv.formula, dat, list("gaussian","gaussian"))
cond.b <- cond.ranefs(fit, burnin = 50L, N = 1000, tune = 2)
cond.b
plot(cond.b) # Overall
plot(cond.b, id = 1) # Plot the first subject (see plot.cond.b.joint).
```

dynPred — *Dynamic predictions for survival sub-model in a multivariate joint model.*

Description

Calculates individualised conditional survival probabilities for subjects during a period of follow-up using a `joint` model fit along with requisite longitudinal process history.

Note that this function is largely designed for use within the ROC function which assesses discriminatory power of the joint model, however it *does* function by itself with proper use of its arguments.

Usage

```
dynPred(
  data,
  id,
  fit,
  u = NULL,
  nsim = 200,
  progress = TRUE,
  scale = NULL,
  df = NULL
)
```

Arguments

`data`	the data to which the original `joint` model was fit.
`id`	subject identifier, i.e. for which subject is the conditional survival probabilities desired?
`fit`	a joint model fit by the `joint` function.
`u`	a numeric `vector` of candidate follow-up times for which a survival probability should be calculated. Note that the first item `u[1]` denotes the start of the "window" and is dropped from calculations. If `u=NULL` (the default), then the probability of surviving all failure times after the `id`'s final longitudinal `time` is calculated.
`nsim`	how many Monte Carlo simulations should be carried out? Defaults to `nsim=200`. First-order estimates are calculated if `nsim=0`.
`progress`	a logical, if `progress=TRUE` (the default) then a progress bar displaying the current percentage of simulations have been completed.
`scale`	numeric scales the variance-covariance parameter in the proposal distribution for the Metropolis-Hastings algorithm. Defaults to `scale = NULL` which doesn't scale the variance term at all. Users are encouraged to experiment with values here; this parameter controls the acceptance rate of the MH scheme.
`df`	numeric denotes the degrees of freedom of the proposed t distribution on the random effects; `df=4` is suggested and is the default.

Details

Dynamic predictions for the time-to-event process based on information available on the subject's longitudinal process up to given time t are calculated by Monte Carlo simulation outlined in Rizopoulos (2011). For a subject last observed at time t, the probability that they survive until future time u is

$$Pr(T_i \geq u | T \geq t; \boldsymbol{Y}_i, \boldsymbol{b}_i; \boldsymbol{\Omega}) \approx \frac{S(u|\hat{\boldsymbol{b}}_i; \boldsymbol{\Omega})}{S(t|\hat{\boldsymbol{b}}_i; \boldsymbol{\Omega})}$$

where T_i is the true failure time for subject i, $\boldsymbol{Y}_i$ their longitudinal measurements up to time t, and $S()$ the survival function.

$\boldsymbol{\Omega}$ is drawn from the multivariate normal distribution with mean $\hat{\boldsymbol{\Omega}}$ and its variance taken from a fitted `joint` object. $\hat{\boldsymbol{b}}$ is drawn from the t distribution by means of a Metropolis-Hastings algorithm with `nsim` iterations.

Value

A list of class `dynPred` which consists of three items:

`pi` A `data.frame` which contains each candidate failure time (supplied by `u`), with the mean, median and 2.5% and 97.5% quantiles of probability of survival until this failure time.

`pi.raw` A `matrix` of with `nsim` rows and `length(u)` columns, each row represents the lth conditional survival probability of survival each `u` survival time. This is largely for debugging purposes.

MH.accept The acceptance rate of the Metropolis-Hastings algorithm on the random effects.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

References

Bernhardt PW, Zhang D and Wang HJ. A fast EM Algorithm for Fitting Joint Models of a Binary Response to Multiple Longitudinal Covariates Subject to Detection Limits. *Computational Statistics and Data Analysis* 2015; **85**; 37–53

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

See Also

`ROC` and `plot.dynPred`.

Examples

```
data(PBC)
PBC$serBilir <- log(PBC$serBilir)
# Focus in on id 81, who fails at around 7 years of follow-up. \code{dynPred} allows us to
# infer how the model believes their survival probability would've progressed (ignoring the
# true outcome at start time).
# Univariate -----------------------------------------------------------
long.formulas <- list(serBilir ~ drug * time + (1 + time|id))
surv.formula <- Surv(survtime, status) ~ drug
family <- list('gaussian')
fit <- joint(long.formulas, surv.formula, PBC, family)
preds <- dynPred(PBC, id = 81, fit = fit, u = NULL, nsim = 200,
                 scale = 2)
preds
plot(preds)
# Bivariate ------------------------------------------------------------
# Does introduction of albumin affect conditional survival probability?
long.formulas <- list(
  serBilir ~ drug * time + I(time^2) + (1 + time + I(time^2)|id),
  albumin ~ drug * time + (1 + time|id)
)
fit <- joint(long.formulas, surv.formula, data = PBC, family = list("gaussian", "gaussian"))
bi.preds <- dynPred(PBC, id = 81, fit = fit, u = NULL, nsim = 200,
                    scale = fit$coeffs$D/sqrt(fit$ModelInfo$n))
bi.preds
plot(bi.preds) # Appears to level-off dramatically; perhaps indicative of this id's albumin
               # levels, or acceleration in serBilir trajectory around 8.5 years.
```

`extractAIC.joint` *Extract AIC from a joint model fit.*

Description

Extract AIC from a joint model fit.

Usage

```
## S3 method for class 'joint'
extractAIC(fit, scale, k = 2, conditional = FALSE, ...)
```

Arguments

`fit`	A fitted `joint` object,
`scale`	See `extractAIC`; not used.
`k`	Numeric specifying the "weight" of degrees of freedom (default `k=2`).
`conditional`	Should AIC of conditional or observed log-likelihood be used? Defaults to `conditional = FALSE`.
`...`	additional arguments (none used).

Value

A numeric vector of length 2, with first and second element giving

`df` The degrees of freedom for the fitted model.

`AIC` The Akaike Information Criterion for the fitted model.

fitted.joint *Obtain joint model fitted values*

Description

returns the fitted values from a `joint` object. Note that the **linear predictor** for each $k = 1, \dots, K$ response is returned.

Usage

```
## S3 method for class 'joint'
fitted(object, as = "matrix", ...)
```

Arguments

`object`	a joint model fit by the `joint` function.
`as`	should the fitted values be returned as a `"matrix"` (the default) or as a `"list"`? Note that `as="matrix"` only works for balanced responses.
`...`	Additional arguments (none used).

Value

A matrix (or list) with a column (or list entry) for each of the fitted linear predictors with class `fitted.joint`.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`residuals.joint`

Examples

```
# Bivariate fit on PBC data -----------------------------------------
data(PBC)

# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'albumin', 'platelets'))
PBC <- na.omit(PBC)

# Specify bivariate fit
long.formulas <- list(
  albumin ~ time*drug + (1 + time|id),
  platelets ~ time * drug + (1 + time|id)
)
```

```
surv.formula <- Surv(survtime, status) ~ drug

fit <- joint(long.formulas, surv.formula, PBC, family = list('gaussian', 'poisson'))
fitted(fit)
```

`fixef.joint` *Extract fixed effects from a* `joint` *object.*

Description

Extract fixed effects from a `joint` object.

Usage

```
## S3 method for class 'joint'
fixef(object, what = c("long", "surv"), ...)
```

Arguments

`object`	a joint model fit by the `joint` function.
`what`	character string. Should the `"long"`itudinal process(es) be extracted, or the `"surv"`ival ones?
`...`	additional arguments (none used).

Value

A vector containing requested fixed effects.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`ranef.joint`

Examples

```
# Univariate fit on PBC data -------------------------------------------
data(PBC)

# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'albumin'))
PBC <- na.omit(PBC)

# Specify simple univariate fit
long.formulas <- list(
  albumin ~ time + (1 + time|id)
)
surv.formula <- Surv(survtime, status) ~ drug

fit <- joint(long.formulas, surv.formula, PBC, family = list('gaussian'))

fixef(fit, 'long')
fixef(fit, 'surv')
```

gmvjoint *Joint Models of Survival and Multivariate Longitudinal Data*

Description

gmvjoint allows the user to fit joint models of survival and multivariate longitudinal data. The longitudinal data is specified by generalised linear mixed models (GLMMs). The joint models are fit via maximum likelihood using an approximate EM algorithm first proposed by Bernhardt et al. (2015). The GLMMs are specified using the same syntax as for package `glmmTMB` Brooks et al. (2017). The joint models themselves are then the flexible extensions to those in e.g. Wulfsohn and Tsiatis (1997). The user is able to simulate data under many different response types.

Author(s)

James Murray <j.murray7@ncl.ac.uk>

References

Bernhardt PW, Zhang D and Wang HJ. A fast EM Algorithm for Fitting Joint Models of a Binary Response to Multiple Longitudinal Covariates Subject to Detection Limits. *Computational Statistics and Data Analysis* 2015; **85**; 37–53

Mollie E. Brooks, Kasper Kristensen, Koen J. van Benthem, Arni Magnusson, Casper W. Berg, Anders Nielsen, Hans J. Skaug, Martin Maechler and Benjamin M. Bolker (2017). glmmTMB Balances Speed and Flexibility Among Packages for Zero-inflated Generalized Linear Mixed Modeling. *The R Journal*, **9(2)**, 378-400.

Murray, J and Philipson P. A fast approximate EM algorithm for joint models of survival and multivariate longitudinal data.*Computational Statistics and Data Analysis* 2022

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

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

Description

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

Usage

```
joint(
  long.formulas,
  surv.formula,
  data,
  family,
  disp.formulas = NULL,
  control = list()
)
```

Arguments

`long.formulas`	A list of formula objects specifying the K responses. Each must be usable by `glmmTMB`. A restriction is that unique identifiers must be named `id`, and increment in intervals of at exactly one. The variable for time must be named `time`.
`surv.formula`	A formula specifying the time-to-event sub-model. Must be usable by `coxph`.
`data`	A `data.frame` containing all covariates and responses.
`family`	A list of length K containing strings denoting the exponential families for each longitudinal sub-model, corresponding in order to `long.formulas`. For choices of `family`, see **details**.
`disp.formulas`	An optional list of length K specifying the dispersion models wanted for each longitudinal sub-model, corresponding in order to `long.formulas`. Defaults to `disp.formulas = NULL`. See **details** for more information.
`control`	A list of control values:

`verbose` Logical: If `TRUE`, at each iteration parameter information will be printed to console. Default is `verbose=FALSE`.

`conv` Character: Convergence criterion, see **details**.

`tol.abs` Numeric: Tolerance value used to assess convergence, see **details**. Default is `tol.abs=1e-3`.

`tol.rel` Numeric: Tolerance value used to assess convergence, see **details**. Default is `tol.rel=1e-2`.

`tol.den` Numeric: Tolerance value used to assess convergence, see **details**. Default is `tol.den=1e-3`.

`tol.thr` Numeric: Threshold used when `conv = 'sas'`, see **details**. Default is `tol.thr=1e-1`.

`grad.eps` Numeric: Step size for numerical differentiation routines used to calculate the gradient in updates to dispersion parameters. This defaults to the cube root of machine tolerance. If a different step size is wanted for each response, a list can also be provided, with each of its elements corresponding to each longitudinal response (even if not fitted with a dispersion model).

`hess.eps` Numeric: Step size for numerical differentiation routines used to calculate the hessian in updates to dispersion parameters. This defaults to the fourth root of machine tolerance. Behaves in same way as `grad.eps` for more information.

`inits` List: list of initial conditions, any/all of the following can be specified (largely for bootstrapping purposes). Accepts elements named: `D`, which should be an appropriately-dimensioned variance-covariance matrix; `beta`, a vector containing all fixed effects; `sigma` a list containing all dispersion parameters, with non-applicable elements set to zero; `gamma` a vector containing all association parameters; `zeta` a vector containing the time-invariant survival coefficients.

`maxit` Integer: Maximum number of EM iterations to carry out before exiting the algorithm. Defaults to `maxit=200L`, which is usually sufficient.

`correlated` Logical: Should covariance parameters **between** responses be estimated and used in determination of model convergence? Default is `correlated=TRUE`. A choice of `correlated=FALSE` is equivalent to imposing the belief that deviations in longitudinal trajectories are not correlated across responses, but can decrease computation time, particularly for large K.

`gh.nodes` Integer: Number of weights and abscissae to use in gauss-hermite quadrature. Defaults to `gh.nodes=3`, which is usually sufficient.

`gh.sigma` Numeric: Standard deviation for gauss-hermite approximation of normal distribution. Defaults to `gh.sigma=1`. This should rarely (if ever) need altering.

`return.dmats` Logical: Should data matrices be returned? Defaults to `return.dmats=TRUE`. Note that some S3 methods for `joint.object`s require the returned object to include these data matrices.

`return.inits` Logical: Should a list of inital conditons be returned? Defaults to `return.inits=FALSE`.

`center.ph` Logical: Should the survival covariates be mean-centered? Defaults to `center.ph=TRUE`.

`post.process` Logical: Should model post-processing be carried out (assumes that the model has converged). Defaults to `post.process = TRUE` which then returns posterior modes and their variance for the random effects, as well as approximated standard error. This is largely for internal use (i.e. if bootstrapping to obtain SEs instead).

Details

Function `joint` fits a joint model to time-to-event data and multivariate longitudinal data. The longitudinal data can be specified by numerous models encompassing a fairly wide range of data. This joint model fit is achieved by the use of an approximate EM algorithm first proposed in Bernhardt et al. (2015), and later used in the 'classic' multivariate joint model in Murray and Philipson (2022). Each longitudinal response is modelled by

$$h_k(E[Y_{ik}|b_{ik};\Omega]) = X_{ik}\beta_k + Z_{ik}b_{ik}$$

where h_k is a known, monotonic link function. An association is induced between the Kth response and the hazard $\lambda_i(t)$ by:

$$\lambda_i(t) = \lambda_0(t)\exp\{S_i^T\zeta + \sum_{k=1}^{K}\gamma_k W_k(t)^T b_{ik}\}$$

where γ_k is the association parameter and $W_k(t)$ is the vector function of time imposed on the Kth random effects structure (i.e. intercept-and-slope; spline).

Value

An object with class `joint`. See `joint.object` for information.

Family specification

Currently, five families are available for implementation, spanning continuous, binary and count data types:

`'gaussian'` Normally distributed. The identity link is used. A term σ_k will be estimated, denoting the *variance* of this response

`'binomial'` For binary data types, a logit link is used.

`'poisson'` For count data types where dispersion is either non-consequential or ignored. A log link is used.

`'genpois'` For count data types where dispersion is at least of some secondary interest. A log link is used. A term σ_k is estimated, denoting the dispersion, φ of the response. This follows interpretation of Zamani & Ismail (2012): $\varphi > 0$: Over-dispersion; $\varphi < 0$: Under-dispersion. $Var[Y] = (1+\varphi)^2\mu$.

`'Gamma'` For continuous data where a Gamma distribution might be sensible. The log link is used. A term σ_k is be estimated, denoting the (log) shape of the distribution, which is then reported as $\varphi_k = \exp\{\sigma_k\}$.

`"negbin"` For count data types where overdispersion is modelled. A log link is used. A term σ_k is estimated, which is then reported as $\varphi_k = \exp\{\sigma_k\}$ which is the overdispersion. The variance of the response is $Var[Y] = \mu + \mu^2/\varphi$.

For families `"negbin"`, `"Gamma"`, `"genpois"`, the user can define the dispersion model desired in `disp.formulas`. For the `"negbin"` and `"Gamma"` cases, we define $\varphi_i = \exp\{W_i\sigma_i\}$ (i.e. the exponent of the linear predictor of the dispersion model; and for `"genpois"` the identity of the linear is used.

Dispersion models

The `disp.formulas` in the function call allows the user to model the dispersion for a given sub-model if wanted. The default value `disp.formulas = NULL` simply imposes an 'intercept only' model. If the kth item in `disp.formulas` corresponds to a longitudinal sub-model with no dispersion term, then it is simply ignored. With this in mind then, if a dispersion model is only required for, say, one sub-model, then the corresponding item in this list of models should be specified as such, with the others set to `~1`.

Standard error estimation

We follow the approximation of the observed empirical information matrix detailed by Mclachlan and Krishnan (2008), and later used in `joineRML` (Hickey et al., 2018). These are only calculated if `post.process=TRUE`. Generally, these SEs are well-behaved, but their reliability will depend on multiple factors: Sample size; number of events; collinearity of REs of responses; number of observed times, and so on. Some more discussion/ references are given in `vcov.joint`.

Convergence of the algorithm

A few convergence criteria (specified by `control$conv`) are available:

- `abs` Convergence reached when maximum absolute change in parameter estimates is `<tol.abs`.
- `rel` Convergence reached when maximum absolute relative change in parameter estimates is `<tol.rel`. A small amount (`tol.den`) is added to the denominator to eschew numerical issues if parameters are nearly zero.
- `either` Convergence is reached when either `abs` or `rel` are met.
- `sas` Assess convergence for parameters $|\Omega_a|$`<tol.thr` by the `abs` criterion, else `rel`. This is the default.

Note that the baseline hazard is updated at each EM iteration, but is not monitored for convergence.

Author(s)

James Murray (`<j.murray7@ncl.ac.uk>`).

References

Bernhardt PW, Zhang D and Wang HJ. A fast EM Algorithm for Fitting Joint Models of a Binary Response to Multiple Longitudinal Covariates Subject to Detection Limits. *Computational Statistics and Data Analysis* 2015; **85**; 37–53

Hickey GL, Philipson P, Jorgensen A, Kolamunnage-Dona R. `joineRML`: a joint model and software package for time-to-event and multivariate longitudinal outcomes. *BMC Med. Res. Methodol.* 2018; **50**

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

Murray, J and Philipson P. A fast approximate EM algorithm for joint models of survival and multivariate longitudinal data.*Computational Statistics and Data Analysis* 2022; **170**; 107438

Zamani H and Ismail N. Functional Form for the Generalized Poisson Regression Model, *Communications in Statistics - Theory and Methods* 2012; **41(20)**; 3666-3675.

See Also

`summary.joint`, `logLik.joint`, `boot.joint`, `extractAIC.joint`, `fixef.joint`, `ranef.joint`, `vcov.joint`, `joint.object` and `xtable.joint`. For data simulation see `simData`.

Examples

```
# 1) Fit on simulated bivariate data, (1x gaussian, 1x poisson) --------
beta <- do.call(rbind, replicate(2, c(2, -0.1, 0.1, -0.2), simplify = FALSE))
gamma <- c(0.3, -0.3)
D <- diag(c(0.25, 0.09, 0.25, 0.05))
family <- list('gaussian', 'poisson')
data <- simData(ntms = 10, beta = beta, D = D, n = 100,
                family = family, zeta = c(0, -0.2),
                sigma = list(0.16, 0), gamma = gamma)$data

# Specify formulae and target families
long.formulas <- list(
  Y.1 ~ time + cont + bin + (1 + time|id),  # Gaussian
  Y.2 ~ time + cont + bin + (1 + time|id)   # Poisson
)
surv.formula <- Surv(survtime, status) ~ bin

fit <- joint(long.formulas, surv.formula, data, family)


# 2) Fit on PBC data -----------------------------------------------------
data(PBC)
# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'serBilir', 'albumin', 'spiders', 'platelets'))
PBC <- na.omit(PBC)

# Specify GLMM sub-models, including interaction and quadratic time terms
long.formulas <- list(
  log(serBilir) ~ drug * (time + I(time^2)) + (1 + time + I(time^2)|id),
  albumin ~ drug * time + (1 + time|id),
  platelets ~ drug * time + (1 + time|id),
  spiders ~ drug * time + (1|id)
)
surv.formula <- Surv(survtime, status) ~ drug

fit <-  joint(long.formulas, surv.formula, PBC,
              family = list("gaussian", "gaussian", "poisson", "binomial"),
              control = list(verbose = TRUE))
fit


# 3) Fit with dispersion models ----------------------------------------
beta <- do.call(rbind, replicate(2, c(2, -0.1, 0.1, -0.2), simplify = FALSE))
gamma <- c(0.3, -0.3)
D <- diag(c(0.25, 0.09, 0.25, 0.05))
family <- list('negbin', 'poisson')   # As an example; only requires one dispersion model.
sigma <- list(c(1, 0.2), 0)           # Need to specify the model in simData call too.
disp.formulas = list(~time, ~1)      # Even though poisson doesn't model dispersion, need to
                                      # populate this element in disp.formulas!
# Simulate some data
data <- simData(ntms = 10, beta = beta, D = D, n = 500,
                family = family, zeta = c(0, -0.2), sigma = sigma,
                disp.formulas = disp.formulas, gamma = gamma)$data

# Now fit using joint
long.formulas <- list(
  Y.1 ~ time + cont + bin + (1+time|id),
  Y.2 ~ time + cont + bin + (1+time|id)
)
fit <- joint(
  long.formulas, Surv(survtime, status) ~ bin,
  data, family, disp.formulas = disp.formulas
)
fit
summary(fit)
```

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

Description

An object returned by the `joint` function, with class `joint` a fitted joint model. Objects of this class currently have methods for: `logLik`, `print`, `ranef`, `fixef`, `summary`, `AIC`, and `vcov`.

Usage

```
joint.object
```

Format

An object of class `NULL` of length 0.

Value

A list with the following components.

`coeffs` A list containing parameter estimates:

- `D` The variance-covariance matrix of the random effects.
- `beta` Vector of fixed effects for longitudinal processes.
- `sigma` List of dispersion parameters, families with no dispersion parameter are returned as an unnamed zero value.
- `gamma` Vector of association parameters.
- `zeta` Vector of time-invariant survival coefficients.

`hazard` A matrix containing unique failure times `ft`, their hazard contribution `haz` and the number of events at that failure time `nev`.

`ModelInfo` A list containing information on the model fit:

- `ResponseInfo` A vector containing response names with (family) reported.

- `Resps` A vector containing response names only.
- `family` A list of families fit.
- `K` An integer specifying the number of longitudinal sub-models.
- `Pcounts` A list containing informations about the number of parameters/random effects:
  - `P` A vector of length K containing the number of fixed effects for each response (in order).
  - `Pd` A vector of length K containing the number of dispersion parameters for each response (in order) 0 denotes no parameter for that response.
  - `q` An integer denoting the number of random effects.
  - `vD` An integer denoting the number of unique variance-covariance parameters estimated.
- `long.formulas` A list of `long.formulas` (i.e. from `joint` call).
- `disp.formulas` A list of `disp.formulas` (i.e. from `joint` call). If no `disp.formulas` are supplied to `joint`, then this is populated by a list of K `"~1"`. The environment is set to `parent.frame` in this case to avoid memory overheads in returned objects.
- `surv.formula` Formula object from `joint` call.
- `survtime` The name of the event time used in `surv.formula`.
- `status` The name of the event indicator used in `surv.formula`.
- `control` List of control parameters used, see `joint`.
- `convergence.criteria` List of parameters relating to the stopping rule.
- `inds` A list of length two, named `R` and `Cpp`, each of which contains the indices for fixed effects β for each response, or the random effects b for the named platform.
- `n` Number of subjects.
- `nobs` A vector containing total number of observations for each response.
- `mi` A K x n matrix containing the number of observations for subject i for the kth response.
- `nev` Number of events.
- `id.assign` A list containing the original ids of subjects in the `data` supplied to `joint`, and the id assigned to them for use in subsequent functions.

`Hessian` The (approximated) Hessian found at MLEs. Only returned if control argument `post.process=TRUE`.

`vcov` The full variance-covariance matrix between parameters. Only returned if control argument `post.process=TRUE`.

`SE` A named vector of approximated standard error for each estimated parameter. Only returned if control argument `post.process=TRUE`.

`logLik` log-likelihood evaluated at parameter estimates. Only returned if control argument `post.process=TRUE`.

`REs` The random effects, with subject-specific variance matrices attributed. If control argumnet `post.process=TRUE` then these are found at MLEs (i.e. are posterior estimates), otherwise they are taken from the final EM iteration.

`elapsed.time` Named numeric containing breakdown of elapsed time for `joint` fit.

`dmats` A list of data matrices on each of the longitudinal and survival processes for each subject.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`joint`.

logLik.joint *Log-likelihood for joint model.*

Description

Calculate joint log-likelihood, degrees of freedom, AIC and BIC of joint model fit.

Usage

```
## S3 method for class 'joint'
logLik(object, conditional = FALSE, ...)
```

Arguments

`object`	a `joint` object.
`conditional`	Logical. Should the conditional or observed data log-likelihood be returned? See **details**.
`...`	additional arguments (none used).

Details

Calculate the log-likelihood of a joint model of survival and multivariate longitudinal data (i.e. a `joint` object). The argument `conditional` manages whether or not the log-likelihood *conditional* on the random effects, or simply the observed data log-likelihood is returned (the default, `conditional = FALSE`).

If `conditional = TRUE`, then the log-likelihood conditional on the random effects is returned. That is

$$\log f(T_i, \Delta_i, Y_i|b_i;\Omega) = \log f(Y_i|b_i;\Omega) + \log f(T_i, \Delta_i|b_i;\Omega) + \log f(b_i|\Omega)$$

If `conditional = FALSE`, then the observed data log-likelihood is returned i.e.

$$\log\int f(Y_i|b_i;\Omega)f(T_i,\Delta_i|b_i;\Omega)f(b_i|\Omega)db_i.$$

Additionally, the degrees of freedom, ν is given by

$$\nu = \texttt{length(vech(D))} + \sum_{k=1}^{K}\{P_k + P_{\sigma_k}\} + P_s,$$

where P_k denotes the number of coefficients estimated for the kth response, and P_{σ_k} the number of dispersion parameters estimated. P_s denotes the number of survival coefficients, i.e. the length of `c(zeta, gamma)`. Finally, all covariance parameters are captured in `length(vech(D))`.

With the degrees of freedom, we can additionally compute AIC and BIC, which are defined in no special way; and are calculated using the observed data log-likelihood.

Value

Returns an object of class `logLik`, a number which is the log-likelihood of the fitted model `object`. This has multiple attributes: `df` which is the degrees of freedom, `df.residual`; the number of residual degrees of freedom; `AIC` and `BIC` which are the Akaike or Bayes information criterion evaluated at either the conditional or observed log-likelihood (as requested by argument `conditional`).

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>)

References

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

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

See Also

`extractAIC.joint` and `anova.joint`

Examples

```
# Bivariate simulated data (2x Gaussian)
data <- simData(n = 100,
   D = diag(c(.25, .04, .2, .02)),
   gamma = c(0.4, -0.2), theta = c(-2, .2))$data
fit <- joint(list(
    Y.1 ~ time + cont + bin + (1 + time|id),
    Y.2 ~ time + cont + bin + (1 + time|id)
  ), Surv(survtime, status) ~ cont + bin,
  data = data,
  family = list('gaussian', 'gaussian'))

logLik(fit)
```

`parseCoxph` *Parsing the survival formula and constructing all survival-related data objects.*

Description

Creates a set of survival data and fits a coxph model using a survival formula and a data set.

Usage

```
parseCoxph(surv.formula, data, center = TRUE)
```

Arguments

`surv.formula`	A formula readable by ‘coxph‘.
`data`	a set of data containing covariate information for variables named by ‘surv.formula‘. Can be of any ’completeness’, as the function returns a reduced set.
`center`	Should the covariate matrices be mean-centered before being returned? defaults to `center = TRUE`.

Value

A list with class `parseCoxph` containing:

`survdata` reduced version of `data`, with only one row per subject, with covariates specified by `surv.formula` along with survival time and failure status.

`Smat` matrix containing all requisite survival covariates (one row per subject).

`ph` the model fit from `coxph`.

`Delta` list of failure indicators for each of the unique subjects.

`n` number of unique subjects.

`ft` vector of unique failure times.

`nev` vector containing number of failures at each failure time `ft`.

`survtime` the name of the `time` variable in `surv.formula`.

`status` the name of the `event` variable in `surv.formula`.

Examples

```
data = simData()$data
parseCoxph(Surv(survtime, status) ~ bin, data = data)
```

`PBC` *Primary biliary cirrhosis data*

Description

Primary biliary cirrhosis (PBC) data. PBC is a chronic liver disease which affects the bile ducts of the liver, complications of which can ultimately lead to death. The longitudinal profile of numerous biomarkers were observed for 312 patients at the Mayo Clinic between 1974 and 1984 with patients assigned to either the active (D-penicillamine, n=154 (50.6 placebo treatment arm (Murtaugh 1994). The data is publicly available in numerous places, including `joineRML` and `survival`. The presence of many longitudinal biomarkers of clinical interest as well as an event-time has lead to the PBC data becoming widely used in literature.

Usage

```
data('PBC')
```

Format

`data.frame` with 312 patients and 19 variables:

`id` Subject identifier

`survtime` Survival time in years

`drug` Binary indicator covariate: was the patient assigned active (`drug=1`) or placebo?

`sex` Binary indicator covariate: Takes value 1 if the subject is female, and zero if male.

`time` Time of visit (0=baseline).

`ascites` Binary *response* variable. Takes value 1 if accumulation of fluid in abdomen ("ascites") present.

`hepatomegaly` Binary *response* variable. Takes value 1 if enlarged liver ("hepatomegaly") present.

`spiders` Binary *response* variable. Takes value 1 if malformed blood vessels in skin ("hepatomegaly") present.

`edema` Factor variable describing edema therapy, see `pbcseq`.

`serBilir` Serum bilirubin (measured in mg/dl).

`serChol` Serum cholesterol (measured in mg/dl).

`album` Serum albumin (measured in mg/dl).

`alkaline` Alkaline phosphotase (measured in U/liter).

`SGOT` Aspartate aminotransferase (measured in U/liter).

`platelets` Platelet count per cubic ml/1000.

`histologic` Histologic stage of disease, see `pbcseq`.

`status` Survival status, `status=1` if the subject experienced mortality and `=0` if censored.

`age` Standardised age at baseline visit.

Details

Nine longitudinal biomarkers exist with varying degrees of completeness in the data.

Source

`pbcseq`

References

Murtaugh PA, Dickson ER, Van Dam GM, Malinchoc M, Grambsch PM, Langworthy AL, Gips CH. Primary biliary cirrhosis: Prediction of short-term survival based on repeated patient visits. *Hepatology* 1994; **20(1)**; 126-134.

`plot.residuals.joint` *Plot joint model residuals*

Description

Plot residuals obtained by a joint model (obtained by `joint`). If the `residuals.joint` object represents the longitudinal process, a simple (paneled) plot is produced (one for each response). If the residual object contains the Cox-Snell residuals then several plots are produced (interactively): The KM estimate of survival function of said residuals and then repeated for each survival covariate in the model call to `joint` (if requested).

Usage

```
## S3 method for class 'residuals.joint'
plot(x, strata = FALSE, ...)
```

Arguments

`x`	an object with class `residuals.joint`.
`strata`	logical, should strata (for the survival sub-model only). Defaults to `strata = FALSE` which produces only one plot of Cox-Snell residuals.
`...`	additional arguments (none used).

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`residuals.joint`

`ranef.joint` *Extract random effects from a `joint` object.*

Description

Return the random effects $\hat{\boldsymbol{b}}$ which maximises the complete data log-likelihood at the MLEs $\hat{\Omega}$.

Usage

```
## S3 method for class 'joint'
ranef(object, Var = FALSE, ...)
```

Arguments

`object`	a joint model fit by the `joint` function.
`Var`	logical, should the estimated variance of the random effects at $\hat{\Omega}$ be returned? Defaults to `Var=FALSE`.
`...`	additional arguments (none used).

Value

A `matrix` containing required random effects effects. If `Var=TRUE`, instead a list is returned with first element the `matrix` of random effects and second a `matrix` of the variances $\hat{\Sigma}$. Note that these are *posterior modes* of the random effects. Conditional distribution can be found by `cond.ranefs`.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`fixef.joint` `cond.ranefs`

Examples

```
# Univariate fit on PBC data -----------------------------------------
data(PBC)

# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'albumin'))
PBC <- na.omit(PBC)

# Specify univariate fit
long.formulas <- list(
  albumin ~ time*drug + (1 + time|id)
)
surv.formula <- Surv(survtime, status) ~ drug

fit <- joint(long.formulas, surv.formula, PBC, family = list('gaussian'))
b <- ranef(fit, FALSE)
```

`residuals.joint` *Obtain joint model residuals*

Description

returns the Pearson residuals values from a `joint` object.

Usage

```
## S3 method for class 'joint'
residuals(
  object,
  what = c("longit", "surv"),
  type = c("response", "pearson"),
  ...
)
```

Arguments

`object`	a joint model fit by `joint` function.
`what`	character string. Should the `"long"`itudinal process(es) be extracted, or the `"surv"`ival ones?
`type`	character. The residual type for `what = "long"` residuals only. Choices are on the `"response"` scale or `"pearson"` residuals. Cox-Snell residuals are returend if `what = "surv"`.
`...`	Additional arguments (none used).

Value

a named list of length K of class `residuals.joint` containing residuals produced by the joint model for each of the $k = 1, \ldots, K$ responses, along with the fitted values as an attribute.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`fitted.joint` `plot.residuals.joint`

Examples

```
# Trivariate fit on PBC data -----------------------------------------
data(PBC)

# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'albumin', 'hepatomegaly', 'platelets'))
PBC <- na.omit(PBC)

# Specify trivariate fit
long.formulas <- list(
  albumin ~ time*drug + (1 + time|id),
  platelets ~ time * drug + (1 + time|id),
  hepatomegaly ~ time * drug + (1|id)
)
surv.formula <- Surv(survtime, status) ~ drug
```

```
fit <- joint(long.formulas, surv.formula, PBC,
             family = list('gaussian', 'poisson', 'binomial'))
R <- residuals(fit, type = 'pearson')
plot(R)
plot(residuals(fit, what = "surv"))
```

rgenpois *Simulate realisations from a generalised poisson distribution*

Description

Simulate realisations from a generalised poisson distribution

Usage

```
rgenpois(mu, phi)
```

Arguments

mu	A numeric vector of rates $\exp \eta$, with η the linear predictor.
phi	A numeric specifying the dispersion φ. If $\varphi < 0$ the response will be underdispersed and overdispersed if $\varphi > 0$.

Details

Follows the "GP-1" implementation of the generalised Poisson distribution outlined in Zamani & Ismail (2012). The variance of produced Y is $(1 + \varphi)^2\mu$. As such the dispersion parameter is bounded (i.e. not in positive reals as with CMP distribution).

Value

An appropriately-dimensioned vector of count data.

References

Zamani H and Ismail N. Functional Form for the Generalized Poisson Regression Model, *Communications in Statistics - Theory and Methods* 2012; **41(20)**; 3666-3675.

ROC *Receiver Operator Characteristics (ROC) for a* `joint` *model.*

Description

Using longitudinal information available up to a time, establish diagnostic capabilities (ROC, AUC and Brier score) of a fitted joint model.

Usage

```
ROC(fit, data, Tstart, delta, control = list(), progress = TRUE, boot = FALSE)
```

Arguments

`fit`	a joint model fit by the `joint` function.
`data`	the data to which the original `joint` model was fit.
`Tstart`	The start of the time window of interest, `Tstart` denotes the time point up to which longitudinal process(es) is used in calculation of survival probabilities.
`delta`	scalar denoting the length of time interval to check for failure times.
`control`	list of control arguments to be passed to `dynPred`, which acts as the main workhorse function for `ROC`. Takes default arguments of `dynPred` if not supplied.
`progress`	should a progress bar be shown, showing the current progress of the ROC function (to `progress` = `TRUE`.
`boot`	logical. Not currently used, legacy argument.

Value

A list of class `ROC.joint` consisting of:

`Tstart` numeric denoting the start of the time window of interest; all dynamic predictions generated used longitudinal information up-to time T_{start}.

`delta` scalar which denotes length of interval to check, such that the window is defined by $[T_{\text{start}}, T_{\text{start}}, +\delta]$.

`candidate.u` candidate vector of failure times to calculate dynamic probability of surviving for each subject alive in `data` at time T_{start}.

`window.failures` numeric denoting the number of observed failures in $[T_{\text{start}}, T_{\text{start}}, +\delta]$.

`Tstart.alive` numeric denoting the risk set at `Tstart`.

`metrics` a `data.frame` containing probabilistic `thresholds` with: `TP` true positives; `FN` false negatives; `FP` false positives; `TN` true negatives; `TPR` true positive rate (sensitivity); `FPR` false positive rate (1-specificity); `Acc` accuracy; `PPV` positive predictive value (precision); `NPV` negative predictive value; `F1s` F1 score and `J` Youden's J statistic.

AUC the area under the curve.

BrierScore The Brier score.

PE The predicted error (taking into account censoring), loss function: square.

MH.acceptance Raw acceptance percentages for each subject sampled.

MH.acceptance.bar mean acceptance of M-H scheme across all subjects.

simulation.info list containing information about call to dynPred.

Author(s)

James Murray (<j.murray7@ncl.ac.uk>).

See Also

dynPred, and plot.ROC.joint.

Examples

```
data(PBC)
PBC$serBilir <- log(PBC$serBilir)
long.formulas <- list(serBilir ~ drug * time + (1 + time|id))
surv.formula <- Surv(survtime, status) ~ drug
family <- list('gaussian')
fit <- joint(long.formulas, surv.formula, PBC, family)
(roc <- ROC(fit, PBC, Tstart = 8, delta = 2, control = list(nsim = 25)))
plot(roc)
```

---

simData *Simulate data from a multivariate joint model*

Description

Simulate multivariate longitudinal and survival data from a joint model specification, with potential mixture of response families. Implementation is similar to existing packages (e.g. joineR, joineRML).

Usage

```
simData(
  n = 250,
  ntms = 10,
  fup = 5,
  family = list("gaussian", "gaussian"),
  sigma = list(0.16, 0.16),
  beta = rbind(c(1, 0.1, 0.33, -0.5), c(1, 0.1, 0.33, -0.5)),
  D = NULL,
  gamma = c(0.5, -0.5),
  zeta = c(0.05, -0.3),
  theta = c(-4, 0.2),
  cens.rate = exp(-3.5),
  regular.times = TRUE,
  dof = Inf,
  random.formulas = NULL,
  disp.formulas = NULL,
  return.ranefs = FALSE
)
```

Arguments

`n`	the number of subjects
`ntms`	the number of time points
`fup`	the maximum follow-up time, such that t = [0, ..., fup] with length `ntms`. In instances where subject i *doesn't* fail before `fup`, their censoring time is set as `fup + 0.1`.
`family`	a K-`list` of families, see **details**.
`sigma`	a K-`list` of dispersion parameters corresponding to the order of `family`, and matching `disp.formulas` specification; see **details**.
`beta`	a $K \times 4$ matrix specifying fixed effects for each K parameter, in the order (Intercept), time, continuous, binary.
`D`	a positive-definite matrix specifying the variance-covariance matrix for the random effects. If not supplied an identity matrix is assumed.
`gamma`	a K-vector specifying the association parameters for each longitudinal outcome.
`zeta`	a vector of length 2 specifying the coefficients for the baseline covariates in the survival sub-model, in the order of continuous and binary.
`theta`	parameters to control the failure rate, see **baseline hazard**.
`cens.rate`	parameter for `rexp` to generate censoring times for each subject.
`regular.times`	logical, if `regular.times = TRUE` (the default), then *every* subject will have the same follow-up times defined by `seq(0, fup, length.out = ntms)`. If `regular.times = FALSE` then follow-up times are set as random draws from a uniform distribution with maximum `fup`.
`dof`	integer, specifies the degrees of freedom of the multivariate t-distribution used to generate the random effects. If specified, this t-distribution is used. If left at the default `dof=Inf` then the random effects are drawn from a multivariate normal distribution.
`random.formulas`	allows user to specify if an intercept-and-slope (`~ time`) or intercept-only (`~1`) random effects structure should be used on a response-by-response basis. Defaults to an intercept-and-slope for all responses.
`disp.formulas`	allows user to specify the dispersion model simulated. Intended use is to allow swapping between intercept only (the default) and a time-varying one (`~ time`). Note that this should be a K-`list` of formula objects, so if only one dispersion model is wanted, then an intercept-only should be specified for remaining sub-models. The corresponding item in list of `sigma` parameters should be of appropriate size. Defaults to an intercept-only model.
`return.ranefs`	a logical determining whether the *true* random effects should be returned. This is largely for internal/simulation use. Default `return.ranefs = FALSE`.

Details

`simData` simulates data from a multivariate joint model with a mixture of families for each $k = 1, \ldots, K$ response. The specification of `family` changes requisite dispersion parameter `sigma`, if applicable. The `family` list can (currently) contain:

`"gaussian"` Simulated with identity link, corresponding item in `sigma` will be the **variance**.

`"poisson"` Simulated with log link, corresponding dispersion in `sigma` can be anything, as it doesn't impact simulation.

`"binomial"` Simulated with logit link, corresponding dispersion in `sigma` can be anything, as it doesn't impact simulation.

`"negbin"` Simulated with a log link, corresponding item in `sigma` will be the **overdispersion** defined on the log scale. Simulated variance is $\mu + \mu^2/\varphi$.

`"genpois"` Simulated with a log link, corresponding item in `sigma` will be the **dispersion**. Values < 0 correspond to under-dispersion, and values > 0 over- dispersion. See `rgenpois` for more information. Simulated variance is $(1 + \varphi)^2\mu$.

`"Gamma"` Simulated with a log link, corresponding item in `sigma` will be the **shape** parameter, defined on the log-scale.

Therefore, for families `"negbin"`, `"Gamma"`, `"genpois"`, the user can define the dispersion model desired in `disp.formulas`, which creates a data matrix W. For the `"negbin"` and `"Gamma"` cases, we define $\varphi_i = \exp\{W_i\sigma_i\}$ (i.e. the exponent of the linear predictor of the dispersion model); and for `"genpois"` the identity of the linear is used.

Value

A list of two `data.frames`: One with the full longitudinal data, and another with only survival data. If `return.ranefs=TRUE`, a matrix of the true b values is also returned. By default (i.e. no arguments provided), a bivariate Gaussian set of joint data is returned.

Baseline hazard

When simulating the survival time, the baseline hazard is a Gompertz distribution controlled by `theta=c(x,y)`:

$$\lambda_0(t) = \exp x + yt$$

where y is the shape parameter, and the scale parameter is $\exp x$.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

References

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

See Also

joint

Examples

```
# 1) A set of univariate data ------------------------------------------
beta <- c(2.0, 0.33, -0.25, 0.15)
# Note that by default arguments are bivariate, so need to specify the univariate case
univ.data <- simData(beta = beta,
                     gamma = 0.15, sigma = list(0.2), family = list("gaussian"),
                     D = diag(c(0.25, 0.05)))

# 2) Univariate data, with failure rate controlled ---------------------
# In reality, many facets contribute to the simulated failure rate, in
# this example, we'll just atler the baseline hazard via 'theta'.
univ.data.highfail <- simData(beta = beta,
                              gamma = 0.15, sigma = list(0.0), family = list("poisson"),
                              D = diag(c(0.40, 0.08)), theta = c(-2, 0.1))

# 3) Trivariate (K = 3) mixture of families with dispersion parameters -
beta <- do.call(rbind, replicate(3, c(2, -0.1, 0.1, -0.2), simplify = FALSE))
gamma <- c(0.3, -0.3, 0.3)
D <- diag(c(0.25, 0.09, 0.25, 0.05, 0.25, 0.09))
family <- list('gaussian', 'genpois', 'negbin')
sigma <- list(.16, 1.5, log(1.5))
triv.data <- simData(ntms=15, family = family, sigma = sigma, beta = beta, D = D,
                     gamma = gamma, theta = c(-3, 0.2), zeta = c(0,-.2))

# 4) K = 4 mixture of families with/out dispersion ---------------------
beta <- do.call(rbind, replicate(4, c(2, -0.1, 0.1, -0.2), simplify = FALSE))
gamma <- c(-0.75, 0.3, -0.6, 0.5)
D <- diag(c(0.25, 0.09, 0.25, 0.05, 0.25, 0.09, 0.16, 0.02))
family <- list('gaussian', 'poisson', 'binomial', 'gaussian')
sigma <- list(.16, 0, 0, .05) # 0 can be anything here, as it is ignored internally.
mix.data <- simData(ntms=15, family = family, sigma = sigma, beta = beta, D = D, gamma = gamma,
                    theta = c(-3, 0.2), zeta = c(0,-.2))

# 5) Bivariate joint model with two dispersion models. -----------------
disp.formulas <- list(~time, ~time)          # Two time-varying dispersion models
sigma <- list(c(0.00, -0.10), c(0.10, 0.15)) # specified in form of intercept, slope
D <- diag(c(.25, 0.04, 0.50, 0.10))
disp.data <- simData(family = list("genpois", "negbin"), sigma = sigma, D = D,
                     beta = rbind(c(0, 0.05, -0.15, 0.00), 1 + c(0, 0.25, 0.15, -0.20)),
                     gamma = c(1.5, 1.5),
                     disp.formulas = disp.formulas, fup = 5)

# 6) Trivariate joint model with mixture of random effects models ------
# It can be hard to e.g. fit a binomial model on an intercept and slope, since e.g.
# glmmTMB might struggle to accurately fit it (singular fits, etc.). To that end, could
# swap the corresponding random effects specification to be an intercept-only.
family <- list("gaussian", "binomial", "gaussian")
# A list of formulae, even though we want to change the second sub-model's specification
# we need to specify the rest of the items, too (same as disp.formulas, sigma).
random.formulas <- list(~time, ~1, ~time)
beta <- rbind(c(2, -0.2, 0.5, -0.25), c(0, 0.5, 1, -1), c(-2, 0.2, -0.5, 0.25))
# NOTE that the specification of RE matrix will need to match.
D <- diag(c(0.25, 0.09, 1, 0.33, 0.05))
# Simulate data, and return REs as a sanity check...
mix.REspec.data <- simData(beta = beta, D = D, family = family,
                           gamma = c(-0.5, 1, 0.5), sigma = list(0.15, 0, 0.15),
                           random.formulas = random.formulas, return.ranefs = TRUE)
```

---

summary.joint   *Summary of an* `joint` *object.*

Description

Generate summary of a fitted multivariate joint model.

Usage

```
## S3 method for class 'joint'
summary(object, ...)
```

Arguments

`object`	a joint model fit by the `joint` function.
`...`	additional arguments (none used).

Value

Object of class `summary.joint`.

Author(s)

James Murray <`j.murray7@ncl.ac.uk`>

See Also

`joint` and `joint.object`

Examples

```
# Simple univariate on log(serum bilirubin) ----------------------------
data(PBC)
long.formulas <-  list(
  log(serBilir) ~ drug * (time + I(time^2)) + (1 + time + I(time^2)|id)
)
surv.formula <- Surv(survtime, status) ~ sex + drug
fit <- joint(long.formulas = long.formulas,
             surv.formula = surv.formula,
             data = PBC, family = list("gaussian"))
summary(fit)

# Bivariate with a dispersion model -----------------------------------
PBC <- na.omit(PBC[,c('id', 'survtime', 'status', 'sex',
                      'drug', 'platelets', 'albumin', 'time')])
long.formula <- list(
  platelets ~ time * drug + (1 + time|id),
  albumin ~ time * drug + (1 + time|id)
)
surv.formula <- Surv(survtime, status) ~ sex + drug
fit <- joint(long.formula, surv.formula, PBC,
             family = list("negbin", "gaussian"),
             disp.formula = list(~time, ~1))
summary(fit)
```

`vcov.joint` *Extract the variance-covariance matrix from a* `joint` *fit.*

Description

Extract the variance-covariance matrix from a `joint` fit.

Usage

```
## S3 method for class 'joint'
vcov(object, corr = FALSE, ...)
```

Arguments

`object`	a joint model fit by the `joint` function.
`corr`	should the correlation matrix be returned instead of the variance-covariance?
`...`	extra arguments (none used).

Details

Uses the observed-empirical **approximation** of information matrix (Mclachlan & Krishnan, 2008). The standard errors for the baseline hazard are not estimated.

Value

A variance-covariance matrix for the joint model object.

Methodology

Many competing ways exist for obtaining the observed information matrix in an EM algorithm. In the context of joint modelling, the observed empirical approximation of the information matrix has been used previously (`joineRML`, Hickey et al. 2018). Elsewhere, estimation of the observed information in a semi-parametric setting is outlined neatly in Xu et al. (2014). Here, they advocate for approximation of this information matrix by numerical differentiation of the profile Fisher Score vector. We do not consider this methodology owing to its computational expense. That is, for each element of Ω which is perturbed by some small amount $\tilde{\Omega}^p$, we must re-calculate $\hat{b}_i$ and $\hat{\Sigma}_i$.

Author(s)

James Murray <`j.murray7@ncl.ac.uk`>

References

Hickey GL, Philipson P, Jorgensen A, Kolamunnage-Dona R. `joineRML`: a joint model and software package for time-to-event and multivariate longitudinal outcomes. *BMC Med. Res. Methodol.* 2018; **50**

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

Xu C, Baines PD, Wang J. Standard error estimation using the EM algorithm for the joint modeling of survival and longitudinal data. *Biostatistics* 2014; **15**(4).

Examples

```
# Univariate fit on PBC data -------------------------------------------
data(PBC)

# Subset data and remove NAs
PBC <- subset(PBC, select = c('id', 'survtime', 'status', 'drug', 'time',
                              'albumin'))
PBC <- na.omit(PBC)

# Specify univariate fit
long.formulas <- list(
  albumin ~ time + (1 + time|id)
)
surv.formula <- Surv(survtime, status) ~ drug

fit <- joint(long.formulas, surv.formula, PBC, family = list('gaussian'))

vcov(fit)
```

xtable.joint	*Print an LaTeX-ready* xtable *for a* joint *object.*

Description

Prints an xtable output for a fitted joint object to the console, or to a specified save location

Usage

```
## S3 method for class 'joint'
xtable(
  x,
  caption = NULL,
  label = NULL,
  align = NULL,
  digits = NULL,
  display = NULL,
  auto = FALSE,
  p.val = FALSE,
  max.row = NULL,
  dp = 3,
  vcov = FALSE,
  capture = FALSE,
  capture.location = "",
  hlines = "middle-bottom",
  booktabs = TRUE,
  size = "footnotesize",
  ...
)
```

Arguments

x	a joint model fit by the joint function.
caption	character, specifies the caption argument of xtable. By default this takes value NULL, which results in a generic caption being generated.
label	character, specifies the label argument of xtable.
align	character, specifies the align argument of xtable. Note by default this is NULL, as alignment is done internally.
digits	integer, specifies the digits argument of xtable. Note by default this is NULL, as argument dp controls this (but can be specified through this, too).
display	character, specifies the display argument of xtable.
auto	logical, specifies the auto argument of xtable. Defaults to FALSE. Not recommended to change.
p.val	logical, should p-values be returned? Defaults to p.val = FALSE.

`max.row`	integer, the number of rows after which the table is 'broken' vertically and merged horizontally; useful for long tables. Defaults to `max.row = NULL` which results in one long table. Note that this can be quite finicky, so trial and error may be required.
`dp`	integer, the number of decimal places to round the estimate, standard error and confidence intervals to; defaults to `dp = 3`.
`vcov`	logical, should the half-vectorisation of the block diagonal of covariance matrix be reported? Default is `vcov = FALSE`.
`capture`	logical, should the printed `xtable` output be saved anywhere instead of just printed to the console? Defaults to `capture = FALSE`.
`capture.location`	character, if `capture = TRUE`, this should specify what *file* it should be saved in. Defaults to `capture.location = ""`.
`hlines`	character, specifies which horizontal lines are used in the outputted LaTeX table. Supply a character string which contains `"top"`, `"middle"` and/or `"bottom"` (in any order) to specify a `toprule`; `midrule` and `bottomrule` in the table. If `booktabs = FALSE`, then these will simply be `hlines`. For instance `hlines = "top-middle-bottom"` prints all three; whilst `hlines = "middle-bottom"` skips the `toprule`.
`booktabs`	logical, if `booktabs = TRUE` (the default) then `toprule`; `midrule` and `bottomrule` replace the usual `hlines`.
`size`	character, LaTeX size to be placed before the tabular environment, defaults to `size = "footnotesize"`; replace with `"normalsize"` if wanted.
`...`	additional arguments, none used.

Value

A LaTeX-ready `xtable` print-out of the joint model. A list containing constituent tables is also returned invisibly, along with the final `xtable` output.

Author(s)

James Murray (<`j.murray7@ncl.ac.uk`>).

See Also

`joint`

Examples

```
# Bivariate joint model -------------------------------------------------
require(xtable)
data <- simData(n = 100)$data
long.formula <- list(
  Y.1 ~ time + cont + bin + (1 + time|id),
  Y.2 ~ time + cont + bin + (1 + time|id)
)
surv.formula <- Surv(survtime, status) ~ cont + bin
```

```
family <- list("gaussian", "gaussian")
fit <- joint(long.formula, surv.formula, data, family)
xtable(fit)
# Example of arguments: add dummy caption, add p-values.
xtable(fit, p.val = TRUE, dp = 4, caption = "This is a caption")
# Change size, place horizontal lines everywhere
xtable(fit, size = "normalsize", hlines = c("top-middle-bottom"))
# Make a wider table without booktabs
xtable(fit, booktabs = FALSE, max.row = 6)
```

Index