Package ‘tramME’

July 5, 2022

Title  Transformation Models with Mixed Effects
Version  1.0.2
Date  2022-07-04

Description  Likelihood-based estimation of mixed-effects transformation models using the Template Model Builder (‘TMB’, Kristensen et al., 2016, <doi:10.18637/jss.v070.i05>). The technical details of transformation models are given in Hothorn et al. (2018, <doi:10.1111/sjos.12291>). Likelihood contributions of exact, randomly censored (left, right, interval) and truncated observations are supported. The random effects are assumed to be normally distributed on the scale of the transformation function, the marginal likelihood is evaluated using the Laplace approximation, and the gradients are calculated with automatic differentiation (Tamasi and Hothorn, 2021, <doi:10.32614/RJ-2021-075>). Penalized smooth shift terms can be defined using ‘mgcv’.

Depends  R (>= 3.6.0), tram (>= 0.3.2), mlt (>= 1.1.0)
Imports  alabama, lme4 (>= 1.1.19), Matrix, methods, mgcv (>= 1.8.34), nlmie, TMB (>= 1.7.15), stats, variables (>= 1.0.2), basefun (>= 1.0.6), numDeriv, MASS, coneproj
Suggests  multcomp, parallel, survival, knitr, coxme, ordinal, ordinalCont, gamm4, gamlss.dist, glmmTMB, xtable

LinkingTo  TMB, RcppEigen
VignetteBuilder  knitr
License  GPL-2
URL  http://ctm.R-forge.R-project.org
Encoding  UTF-8
RoxygenNote  7.2.0
NeedsCompilation  yes

Author  Balint Tamasi [aut, cre] (<https://orcid.org/0000-0002-2629-7362>), Torsten Hothorn [ctb] (<https://orcid.org/0000-0001-8301-0471>)
Maintainer  Balint Tamasi <balint.tamasi@uzh.ch>
Repository  CRAN
Date/Publication  2022-07-05 00:20:02 UTC
R topics documented:

- anova.tramME
- BoxCoxME
- coef.LmME
- coef.SurvregME
- coef.tramME
- coef<-.tramME
- ColrME
- confint.LmME
- confint.tramME
- CoxphME
- duplicate
- duplicate.tramTMB
- edf_smooth.tramME
- LehmannME
- LmME
- logLik.tramME
- model.frame.tramME
- model.matrix.tramME
- optim_control
- plot.smooth.tramME
- plot.tramME
- PolrME
- predict.tramME
- predict.tramTMB
- print.anova.tramME
- print.summary.tramME
- print.tramME
- print.VarCorr.tramME
- ranef.LmME
- ranef.tramME
- residuals.LmME
- residuals.tramME
- sigma.LmME
- smooth_terms.LmME
- smooth_terms.tramME
- summary.tramME
- SurvregME
- tramME
- tramME_model
- tramTMB
- VarCorr.LmME
- VarCorr.tramME
- varcov
- varcov.LmME
- varcov.tramME
- varcov<
Description

Calculates information criteria and LR ratio test for nested tramME models. The calculation of the degrees of freedom is problematic, because the parameter space is restricted.

Usage

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

Arguments

object A tramME object.
object2 A tramME object.
... Optional arguments, for compatibility with the generic. (Ignored)

Details

Currently only supports the comparison of two models. Additional arguments will be ignored. The nestedness of the models is not checked.

Value

A data.frame with the calculated statistics.

Examples

data("sleepstudy", package = "lme4")
mod1 <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
mod2 <- LmME(Reaction ~ Days + (Days || Subject), data = sleepstudy)
anova(mod1, mod2)
BoxCoxME

**Mixed-effects version of BoxCox**

### Description

Mixed-effects version of **BoxCox**

### Usage

```r
BoxCoxME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  resid = FALSE,
  do_update = FALSE,
  estinit = TRUE,
  initpar = NULL,
  fixed = NULL,
 nofit = FALSE,
  control = optim_control(),
  ...
)
```

### Arguments

- **formula**: an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under **tram** and in the package vignette.
- **data**: an optional data frame, list or environment (or object coercible by `as.data.frame`) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`.
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **weights**: an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
- **offset**: this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
- **na.action**: a function which indicates what should happen when the data contain NAs. The default is set to `na.omit`.
- **silent**: Logical. Make **TMB** functionality silent.
Extract the coefficients of the fixed effects terms of an LmME model.

### Description

Extract the coefficients of the fixed effects terms of an LmME model.

### Usage

```r
## S3 method for class 'LmME'
coef(object, as.lm = FALSE, fixed = TRUE, ...)
```

### Arguments

- **object**: An LmME object.
- **as.lm**: If TRUE, return the transformed coefficients as in a `lmerMod` object. If FALSE, return the coefficients from the model.
- **fixed**: If TRUE, also include the fixed parameters.
- **...**: Optional arguments passed to `coef.tramME`.

### Value

A numeric vector of the transformed coefficients.

### Examples

```r
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
coef(fit, as.lm = TRUE)
```
coef.SurvregME

Extract the coefficients of the fixed effects terms of an SurvregME model.

Description

Extract the coefficients of the fixed effects terms of an SurvregME model.

Usage

## S3 method for class 'SurvregME'
coef(object, as.survreg = FALSE, ...)

Arguments

object        An SurvregME object.
as.survreg    If TRUE, return the transformed coefficients as in a survival::survreg object.
...           Optional arguments passed to coef.tramME.

Value

A numeric vector of the transformed coefficients.

Examples

library("survival")
fit <- SurvregME(Surv(time, status) ~ rx + (1 | litter), data = rats)
coef(fit, as.survreg = TRUE)

coef.tramME

Extract the coefficients of the fixed effects terms.

Description

Extract the coefficients of the fixed effects terms.

Usage

## S3 method for class 'tramME'
coef(object, with_baseline = FALSE, fixed = TRUE, ...)

---
Arguments

- **object**: A `tramME` object.
- **with_baseline**: If `TRUE`, also include the baseline parameters and the fixed effects parameters from the smooth terms.
- **fixed**: If `TRUE`, also include the fixed parameters.
- **...**: Optional parameters (ignored).

Value

Numeric vector of parameter values.

Examples

```r
library("survival")
mod <- SurvregME(Surv(time, status) ~ rx + (1 | litter/rx), data = rats,
                 dist = "exponential", nofit = TRUE)
coef(mod, with_baseline = TRUE)
coef(mod, with_baseline = TRUE, fixed = FALSE)
```

---

**Description**

Sets the whole vector of fixed-effects coefficients of a `tramME` model. The parameters of the baseline transformation function should respect the restrictions of the parameter space. This is checked before setting the new parameter values provided that the parameters for the variance components has already been set. If the model contains fixed coefficient parameters, the input should also respect that. When called on a fitted `tram` object, the function sets it to unfitted and removes all parts that come from the estimation.

**Usage**

```r
## S3 replacement method for class 'tramME'
coef(object) <- value
```

**Arguments**

- **object**: A `tramME` object.
- **value**: Numeric vector of new coefficient values.

**Value**

A `tramME` object with the new coefficient values.
Examples

```r
data("sleepstudy", package = "lme4")
mod <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy, nofit = TRUE)
coef(mod) <- c(-1, 0.5, 1)
```

**ColrME**

Mixed-effects version of Colr

Description

Mixed-effects version of Colr

Usage

```r
ColrME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  resid = FALSE,
  do_update = FALSE,
  estinit = TRUE,
  initpar = NULL,
  fixed = NULL,
  nofit = FALSE,
  control = optim_control(),
  ...
)
```

Arguments

- **formula**: an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under tram and in the package vignette.
- **data**: an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula).
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If present, the weighted log-likelihood is maximised.
offset this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases.

na.action a function which indicates what should happen when the data contain NAs. The default is set to na.omit.

silent Logical. Make TMB functionality silent.

resid Logical. If TRUE, the score residuals are also calculated. This comes with some performance cost.

do_update Logical. If TRUE, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.

estinit logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model

initpar named list of initial parameter values, if NULL, it is ignored

fixed a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix

nofit logical, if TRUE, creates the model object, but does not run the optimization

control list with controls for optimization

... additional arguments to tram.

Value

A ColrME object.

---

### confint.LmME

Confidence intervals for LmME model parameters

Description

Confidence intervals for model parameters on their original scale, optionally consistent with the linear mixed-model specification. When as.lm = TRUE, only Wald CIs are available.

Usage

```r
## S3 method for class 'LmME'
confint(
  object,
  parm = NULL,
  level = 0.95,
  as.lm = FALSE,
  pargroup = c("all", "fixef", "ranef"),
  type = c("Wald", "wald", "profile"),
  estimate = FALSE,
  ...
)
```
Arguments

- `object`: An `LmME` object.
- `parm`: Names of the parameters to extract.
- `level`: Confidence level.
- `as.lm`: Logical. If `TRUE`, return results consistent with the normal linear mixed model parametrization.
- `pargroup`: The name of the parameter group to extract. With `as.lm = FALSE`, the available options are described in `confint.tramME`. When `as.lm = TRUE`, the following options are available:
  - `all`: Fixed effects and variance components parameters.
  - `fixef`: Fixed effects parameters (including FE parameters of the smooth terms).
  - `ranef`: Variance components parameters (including the smoothing parameters of the random effects).
- `type`: Type of the CI: either Wald or profile.
- `estimate`: Logical, add the point estimates in a third column.
- `...`: Optional parameters passed to `confint.tramME`

Value

A matrix with lower and upper bounds.

Examples

data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
confint(fit) ## transformation model parametrization
confint(fit, as.lm = TRUE) ## LMM parametrization
confint(fit, as.lm = TRUE, pargroup = "fixef", estimate = TRUE)
confint(fit, as.lm = TRUE, parm = "(Sigma)") ## error SD

Description

Confidence intervals for model parameters on their original scale. Either Wald CI or profile CI by root finding. Multicore computations are supported in the case of profile confidence intervals, but snow support is yet to be implemented.
Usage

```r
## S3 method for class 'tramME'
confint(
  object,
  parm = NULL,
  level = 0.95,
  pargroup = c("all", "fixef", "shift", "baseline", "ranef", "smooth"),
  type = c("Wald", "wald", "profile"),
  estimate = FALSE,
  pmatch = FALSE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("profile.ncpus", 1L),
  ...)
```

Arguments

- `object`: A `tramME` object.
- `parm`: The indeces or names of the parameters of interest. See in details.
- `level`: Confidence level.
- `pargroup`: The name of the parameter group to return:
  - `all`: All parameters.
  - `fixef`: Fixed effects parameters.
  - `shift`: Shift parameters.
  - `baseline`: Parameters of the baseline transformation function.
  - `ranef`: Variance components parameters.
  - `smooth`: Parameters that belong to the smooth shift terms (both FE and smoothing parameters).
- `type`: Type of the CI: either Wald or profile.
- `estimate`: Logical, add the point estimates in a third column.
- `pmatch`: Logical. If `TRUE`, partial name matching is allowed.
- `parallel`: Method for parallel computation.
- `ncpus`: Number of cores to use for parallel computation.
- `...`: Optional parameters.

Value

A matrix with lower and upper bounds.

Examples

```r
data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
confint(fit)
confint(fit, pargroup = "shift", estimate = TRUE)
exp(confint(fit, 1:2, pargroup = "ranef")) # CIs for the SDs of the REs
```
CoxphME

Mixed-effects version of Coxph

Description

Mixed-effects version of Coxph

Usage

CoxphME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  resid = FALSE,
  do_update = FALSE,
  estinit = TRUE,
  initpar = NULL,
  fixed = NULL,
  nofit = FALSE,
  control = optim_control(),
  ...
)

Arguments

formula an object of class "formula"; a symbolic description of the model structure to be fitted. The details of model specification are given under tram and in the package vignette.
data an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula).
subset an optional vector specifying a subset of observations to be used in the fitting process.
weights an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If present, the weighted log-likelihood is maximised.
offset this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases.
na.action a function which indicates what should happen when the data contain NAs. The default is set to na.omit.
silent Logical. Make TMB functionality silent.
**resid** Logical. If TRUE, the score residuals are also calculated. This comes with some performance cost.

**do_update** Logical. If TRUE, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.

**estinit** logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model

**initpar** named list of initial parameter values, if NULL, it is ignored

**fixed** a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix

**nofit** logical, if TRUE, creates the model object, but does not run the optimization

**control** list with controls for optimization

... additional arguments to `tram`.

**Value**

A CoxphME object.

duplicate

Generic for copying objects that are (partly) modified in place

**Description**

Generic for copying objects that are (partly) modified in place

**Usage**

duplicate(object, ...)

**Arguments**

object An object.

... Optional parameters.
**duplicate.tramTMB**  
*Create a duplicate of the tramTMB object*

**Description**
Create a duplicate of the tramTMB object

**Usage**
```r
## S3 method for class 'tramTMB'
duplicate(object, ...)
```

**Arguments**
- `object`: A tramTMB object.
- `...`: Optional parameters (not used).

**edf_smooth.tramME**  
*EDFs of smooth shift terms*

**Description**
Returns an estimate of effective degrees of freedom associated with each smooth term.

**Usage**
```r
## S3 method for class 'tramME'
edf_smooth(object, ...)
```

**Arguments**
- `object`: A tramME object.
- `...`: Optional arguments passed to the Hessian calculations.

**Details**
The EDFs are calculated by summing up the elements of

\[ \text{diag}(V_\theta I) \]

term-by-term. $V_\theta$ is the joint covariance matrix of fixed and random parameters (the inverse of the joint precision, i.e., Hessian of the negative log-likelihood), and $I$ is the joint precision of the unpenalized negative log-likelihood function. See Wood et al. (2016) or Wood (2017, Chapter 6) for references.
Value

A named vector with the edf values.

References


Examples

data("mcycle", package = "MASS")
fit <- LmME(accel ~ s(times), data = mcycle)
edf_smooth(fit)

LehmannME

*Mixed-effects version of Lehmann*

Description

Mixed-effects version of Lehmann

Usage

LehmannME(
    formula,
    data,
    subset,
    weights,
    offset,
    na.action = na.omit,
    silent = TRUE,
    resid = FALSE,
    do_update = FALSE,
    estinit = TRUE,
    initpar = NULL,
    fixed = NULL,
    nofit = FALSE,
    control = optim_control(),
    ...
)
Arguments

- **formula**: an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.

- **data**: an optional data frame, list or environment (or object coercible by `as.data.frame`) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`.

- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.

- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If present, the weighted log-likelihood is maximised.

- **offset**: this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases.

- **na.action**: a function which indicates what should happen when the data contain NAs. The default is set to `na.omit`.

- **silent**: Logical. Make TMB functionality silent.

- **resid**: Logical. If TRUE, the score residuals are also calculated. This comes with some performance cost.

- **do_update**: Logical. If TRUE, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.

- **estinit**: logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model

- **initpar**: named list of initial parameter values, if NULL, it is ignored

- **fixed**: a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix

- **nofit**: logical, if TRUE, creates the model object, but does not run the optimization

- **control**: list with controls for optimization

- **...**: additional arguments to `tram`.

Value

A LehmannME object.

---

**LmME**

Mixed-effects version of Lm

---

Description

Mixed-effects version of Lm
Usage

LmME(
    formula,
    data,
    subset,
    weights,
    offset,
    na.action = na.omit,
    silent = TRUE,
    resid = FALSE,
    do_update = FALSE,
    estinit = TRUE,
    initpar = NULL,
    fixed = NULL,
    nofit = FALSE,
    control = optim_control(),
    ...
)

Arguments

formula an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under tram and in the package vignette.
data an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula).
subset an optional vector specifying a subset of observations to be used in the fitting process.
weights an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If present, the weighted log-likelihood is maximised.
offset this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases.
na.action a function which indicates what should happen when the data contain NAs. The default is set to na.omit.
silent Logical. Make TMB functionality silent.
resid Logical. If TRUE, the score residuals are also calculated. This comes with some performance cost.
do_update Logical. If TRUE, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.
estinit logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model
initpar named list of initial parameter values, if NULL, it is ignored
fixed a named vector of fixed regression coefficients; the names need to correspond to 
column names of the design matrix

nofit logical, if TRUE, creates the model object, but does not run the optimization

control list with controls for optimization

... additional arguments to tram.

Value

A LmME object.

logLik.tramME

Get the log-likelihood of the tramME model

Description

Evaluates the log-likelihood function. New parameter values and data can optionally be supplied. 
In the latter case, the function returns the out-of-sample log-likelihood.

Usage

## S3 method for class 'tramME'
logLik(
  object,
  param = NULL,
  newdata = NULL,
  type = c("integrated", "fix_smooth", "penalized"),
  ...
)

Arguments

object A tramME object.

param An optional named list of parameter values (beta and theta). See details. Optional-

newdata An optional data.frame to calculate the out-of-sample log-likelihood.

type The type of the likelihood to be calculated:

• integrated (default when newdata = NULL): The marginal log-likelihood, 
calculated by integrating out the random effects.

• fix_smooth (default when newdata is supplied): Treating the penalized pa-

  rameters of the smooth terms as fixed at their posterior mode predictions 
  and only integrating out the 'true' random effects. (Consistent with the 
  functionality of ranef.tramME and residuals.tramME when fix_smooth 
  = TRUE.)
- penalized: Treat all parameters as fixed, return the penalized log-likelihood (conditional log-likelihood + penalty for smooth terms and random effects). This is equivalent to fixing all random effect values.

See details.

Optional argument (for consistency with generic).

Details

By default, param is set to the estimated (or previously set) parameters. If the parameter vectors in the model are incomplete (contain NA elements), the returned log-likelihood will also be NA, unless the user provides new values.

Setting type = "fix_smooth" fixes the random effects terms that correspond to penalized smooths at their estimated values, so that they are not refitted when newdata is supplied. This is consistent with treating these parameter regularized fixed terms, i.e. as 'new-style' random effects described by Hodges (2014, Chapter 13).

The "fix_smooth" and "penalized" options for type are just for convenience. The same functionality can be achieved by setting param$gamma to the desired values. "penalized" respects the values of param$gamma if both are supplied, while "fix_smooth" overwrites them with the fitted values if there are ambiguities.

Value

A numeric value of the log-likelihood.

Type of the log-likelihood

By default, logLik calculates the _integrated_ (or marginal) log-likelihood by integrating over the random effects. By fixing the random effects, the value of the log-likelihood changes, because TMB won't integrate over these random effects. This will result in the _penalized_ log-likelihood (conditional log-likelihood + penalty for smooth terms and random effects, see example).

By setting type = "penalized", the function will 'fix' all random effects and penalized parameters of the smooth terms at their predicted levels, and calculate the penalized log-likelihood. In this sense, setting type = "fix_smooth" will result in a hybrid log-likelihood value, where the 'true' random effects (c.f. Hodges 2014, Ch. 13) are integrated out, while it includes the penalty values for the penalized parameters of the smooths terms.

In general, it is not clear which type of log-likelihood we should calculate when we want to evaluate models based on their out-of-sample log-likelihood values. The context and the model setup are key in these cases. Please make sure you know what you want to calculate to avoid misunderstandings.

References

Examples

```r
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
logLik(fit)

data("mcycle", package = "MASS")
fit <- LmME(accel ~ s(times), data = mcycle)
logLik(fit) < logLik(fit, type = "penalized")
```

---

**model.frame.tramME**  
*Extract model frame from a tramME model*

---

**Description**

Extract model frame from a tramME model

**Usage**

```r
## S3 method for class 'tramME'
model.frame(
  formula,
  data = NULL,
  group_as_factor = FALSE,
  ignore_response = FALSE,
  ...
)
```

**Arguments**

- **formula**  
  A tramME object.

- **data**  
  a data.frame, list or environment (or object coercible by `as.data.frame` to a data.frame), containing the variables in `formula`. Neither a matrix nor an array will be accepted.

- **group_as_factor**  
  Logical; If TRUE, automatically convert the grouping variables of the random effects to factors. (not used, might not be needed)  

- **ignore_response**  
  Logical; If TRUE, the response is not added to the result. In this case the function won't look for it in data.

- **...**  
  Optional arguments, passed to `model.frame`. 
Details

In mlt, the basis functions expect the response variables in the data to be evaluated, i.e. instead of x and y columns we should have a `Surv(x, y)` column when the response is a Surv object. model.frame.tramME builds the model frame accordingly, assigning to the resulting object the class tramME_data to indicate this structure to other functions that use its results. If the input data is a tramME_data is also expects this structure.

Value

A tramME_data object, which is also a data.frame.

Examples

data("sleepstudy", package = "lme4")
mod <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy, nofit = TRUE)
model.frame(mod)

Description

Model matrix for fixed effects, random effects, and baseline transformations (with interacting terms if present).

Usage

## S3 method for class 'tramME'
model.matrix(
  object,
  data = model.frame(object),
  type = c("Y", "X", "Zt"),
  drop_unused_groups = FALSE,
  keep_sign = TRUE,
  simplify = FALSE,
  ...
)

Arguments

object A tramME object.
data A data.frame containing the variable values.
type "X": Fixed effects model matrix. "Zt": Random effects model matrix (transposed). "Y": Model matrices for the baseline transformation.
drop_unused_groups Logical; remove unused levels of the random effects. (see drop.unused.levels argument of mkReTrms)
Details

Creates model matrices for fixed effects (type = "X") and random effects (type = "Zt") and baseline transformation (type = "Y"), by evaluating the respective basis functions given a new dataset.

The response values may be exact, censored (left, right, interval) and truncated (left, right, interval), and the function returns several, potentially empty, model matrices:

- Ye: Exact observations.
- Yeprime: The model matrix corresponding to the first derivative of the baseline transformation, evaluated at exact observations.
- Yl: Left-censored observations.
- Yr: Right-censored observations.
- Yil and Yir: Interval-censored observations evaluated at the left and right bounds of the interval.
- Ytl: Left-truncated observations.
- Ytr: Right-truncated observations.
- Ytil and Ytir: Interval-truncated observations evaluated at the left and right bounds of the interval.

for the baseline transformations (unless simplify = TRUE).

Value

List of requested model matrices.

Note

The model matrix of the random effects is a sparse matrix and it is transposed to be directly used with Matrix::crossprod which is faster than transposing and multiplying ("Zt" instead of "Z").

Examples

library("survival")
rats$litter <- factor(rats$litter)
m <- CoxphME(Surv(time, status) ~ rx + (1 | litter), data = rats,
            log_first = TRUE, nofit = TRUE)
mm <- model.matrix(m)
nd <- model.frame(m)[rep(1, 100), ]
nd[[1]] <- seq(1, 120, length.out = 100)
mm2 <- model.matrix(m, data = nd, simplify = TRUE)
mm3 <- model.matrix(m, data = nd, simplify = TRUE, drop_unused_groups = TRUE)
## compare mm2$Zt & mm3$Zt
**optim_control**  
*Set up and control optimization parameters*

### Description

Set up and control optimization parameters

### Usage

```r
optim_control(
  method = c("nlminb", "BFGS", "CG", "L-BFGS-B"),
  scale = TRUE,
  trace = FALSE,
  ntry = 5,
  ...
)
```

### Arguments

- **method**  
  Optimization procedure.
- **scale**  
  Logical; if TRUE rescale the fixed effects design matrix to improve convergence.
- **trace**  
  Logical; print trace of the optimization.
- **ntry**  
  Number of restarts with new random initialization if optimization fails to converge.
- **...**  
  Optional arguments passed to `auglag`, `nlminb` or `optim` as a list of control parameters.

---

**plot.smooth.tramME**  
*Plot smooth terms of a tramME model.*

### Description

Plot smooth terms of a tramME model.

### Usage

```r
## S3 method for class 'smooth.tramME'
plot(
  x,
  which = seq_along(x),
  col = 1,
  fill = grey(0.5, 0.25),
  add = FALSE,
  ...
)
```
Arguments

- **x** A `smooth.tramME` object.
- **which** Select terms to be printed by their indices.
- **col** Line color for the point estimates.
- **fill** Fill color for the confidence intervals.
- **add** Add the plot to an existing figure.
- **...** Optional parameters passed to the plotting functions.

Examples

```r
data("mcycle", package = "MASS")
fit <- LmME(accel ~ s(times), data = mcycle)
plot(smooth_terms(fit, as.lm = TRUE))
```

---

**plot.tramME**

*Plotting method for tramME objects*

Description

Plot the conditional distribution evaluated at a grid of possible response values and a set of covariate and random effects values on a specified scale.

Usage

```r
## S3 method for class 'tramME'
plot(
  x, 
  newdata = model.frame(x), 
  ranef = NULL, 
  fix_smooth = TRUE, 
  type = c("trafo", "distribution", "survivor", "density", "logdensity", "hazard", "loghazard", "cumhazard", "odds", "logodds", "quantile"), 
  ... 
)
```

Arguments

- **x** A `tramME` object.
- **newdata** an optional data frame of observations
- **ranef** Random effects (either in named list format or a numeric vector) or the word "zero". See Details.
- **fix_smooth** If FALSE, the random effects coefficients of the smooth terms are refitted to `newdata`. It's probably not what you want to do.
- **type** The scale on which the predictions are evaluated:
- **trafo**: The prediction evaluated on the scale of the transformation function.
- **distribution**: The prediction evaluated on the scale of the conditional CDF.
- **survivor**: The prediction evaluated on the scale of the (conditional) survivor function.
- **density, logdensity**: The prediction evaluated on the scale of the conditional (log-)PDF.
- **hazard, loghazard, cumhazard**: The prediction evaluated on the hazard/log-hazard/cumulative hazard scale.
- **odds, logodds**: The prediction evaluated on the (log-)odds scale.
- **quantile**: Return the quantiles of the conditional outcome distribution corresponding to `newdata`. For more information, see Details.

Additional arguments, passed to `plot.mlt`.

**Details**
When `ranef` is equal to "zero", a vector of zeros with the right size is substituted. For more details, see `predict.tramME`.

For more information on how to control the grid on which the functions are evaluated, see the documentation of `predict.mlt`.

**Value**
A numeric matrix of the predicted values invisibly.

**Examples**
```r
data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
plot(fit, K = 100, type = "density")
```

---

**PolrME**

*Mixed-effects version of Polr*

**Description**

Mixed-effects version of Polr

**Usage**

`PolrME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,`
method = c("logistic", "probit", "loglog", "cloglog"),
silent = TRUE,
resid = FALSE,
do_update = FALSE,
estinit = TRUE,
initpar = NULL,
fixed = NULL,
nofit = FALSE,
control = optim_control(),
"
)

**Arguments**

- **formula**: an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under tram and in the package vignette.
- **data**: an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula).
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If present, the weighted log-likelihood is maximised.
- **offset**: this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases.
- **na.action**: a function which indicates what should happen when the data contain NAs. The default is set to na.omit.
- **method**: a character describing the link function.
- **silent**: Logical. Make TMB functionality silent.
- **resid**: Logical. If TRUE, the score residuals are also calculated. This comes with some performance cost.
- **do_update**: Logical. If TRUE, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.
- **estinit**: logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model
- **initpar**: named list of initial parameter values, if NULL, it is ignored
- **fixed**: a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix
- **noffit**: logical, if TRUE, creates the model object, but does not run the optimization
- **control**: list with controls for optimization
- **...**: additional arguments to tram.
**predict.tramME**

**Value**

A PolrME object.

**predict.tramME**

*Predict method for tramME objects*

**Description**

Evaluates the _conditional_ distribution implied by a tramME model, given by a set of covariates and random effects on a selected scale.

**Usage**

```r
## S3 method for class 'tramME'
predict(
  object,
  newdata = model.frame(object),
  ranef = NULL,
  fix_smooth = TRUE,
  type = c("lp", "trafo", "distribution", "survivor", "density", "logdensity",
           "hazard", "loghazard", "cumhazard", "odds", "logodds", "quantile"),
  ...
)
```

**Arguments**

- `object`: A tramME object.
- `newdata`: an optional data frame of observations
- `ranef`: Random effects it can be a ranef.tramME object, a named list, an unnamed list, NULL or the word "zero". See Details.
- `fix_smooth`: If FALSE, the random effects coefficients of the smooth terms are refitted to newdata. It’s probably not what you want to do.
- `type`: The scale on which the predictions are evaluated:
  - *lp*: Linear predictor (Xb + Zg). For more information, see Details.
  - *trafo*: The prediction evaluated on the scale of the transformation function.
  - *distribution*: The prediction evaluated on the scale of the (conditional) CDF.
  - *survivor*: The prediction evaluated on the scale of the (conditional) survivor function.
  - *density, logdensity*: The prediction evaluated on the scale of the conditional (log-)PDF.
  - *hazard, loghazard, cumhazard*: The prediction evaluated on the hazard/log-hazard/cumulative hazard scale.
  - *odds, logodds*: The prediction evaluated on the (log-)odds scale.
  - *quantile*: Return the quantiles of the conditional outcome distribution corresponding to newdata. For more information, see Details.
- `...`: Additional arguments, passed to predict.mlt.
Details

When newdata contains values of the response variable, prediction is only done for those values. In this case, if random effects vector (ranef) is not supplied by the user, the function predicts the random effects from the model using newdata.

When no response values are supplied in newdata, the prediction is done on a grid of values for each line of the dataset (see predict.mlt for information on how to control the setup of this grid). In this case, the user has to specify the vector of random effects to avoid ambiguities.

The linear predictor (type = "lp") equals to the shift terms plus the random effects terms _without_ the baseline transformation function_.

The linear predictor (type = "lp") and the conditional quantile function (type = "quantile") are special in that they do not return results evaluated on a grid, even when the response variable in newdata is missing. The probabilities for the evaluation of the quantile function can be supplied with the prob argument of predict.mlt.

In the case of type = "quantile", when the some of the requested conditional quantiles fall outside of the support of the response distribution (specified when the model was set up), the inversion of the CDF cannot be done exactly and tramME returns censored values.

ranef can be different objects based on what we want to calculate and what the other inputs are. If ranef is a ranef.tramME, we assume that it contains the full set of random effects, but not the penalized coefficients of the smooth terms. In this case fix_smooth must be TRUE. If ranef is a named vector, we are fixing the supplied random effects (and penalized coefficients) and predict the rest from newdata (fix_smooth may also be used in this case). In this case, the random effects are identified with the same naming convention as in object$param$gamma.

If ranef is an unnamed vector, the function expects the full set of necessary random effects (with or without penalized coefficients, depending on fix_smooth). If ranef = NULL (the default), all random effects and optionally penalized parameters (although this is not recommended) are predicted from newdata. Finally, if ranef is equal to "zero", a vector of zeros with the right size is used.

Value

A numeric vector/matrix of the predicted values (depending on the inputs) or a response object, when the some of the requested conditional quantiles fall outside of the support of the response distribution specified when the model was set up (only can occur with type = "quantile").

Examples

data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
predict(fit, type = "trafo") ## evaluate on the transformation function scale
d <- sleepstudy
d$Reaction <- NULL
pr <- predict(fit, newdata = d, ranef = ranef(fit), type = "distribution",
 K = 100)
predict.tramTMB

Post-estimation calculations in a tramTMB model

Description

Post-estimation calculations in a tramTMB model

Usage

## S3 method for class 'tramTMB'
predict(
  object,
  newdata,
  parameters = .get_par(object, full = TRUE),
  scale = c("lp", "trafo"),
  cov = FALSE,
  as.lm = FALSE,
  ...
)

Arguments

object A tramTMB object
newdata A named list with elements Y, X and Z (not all necessary)
parameters A named list of parameter values
scale The scale on which the post-estimation calculations are done
cov Logical; If TRUE, calculate the full covariance matrix of the calculated values
as.lm Logical; reparameterize as a LMM
... Optional arguments (ignored).

print.anova.tramME

Printing anova.tramME table

Description

Printing anova.tramME table

Usage

## S3 method for class 'anova.tramME'
print(
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

Arguments

- **x**: A `anova.tramME` object.
- **digits**: Minimum number of significant digits to be used for most numbers.
- **signif.stars**: Logical; if `TRUE`, P-values are additionally encoded visually as 'significance stars' in order to help scanning of long coefficient tables. It defaults to the `show.signif.stars` slot of `options`.
- **...**: Optional arguments passed to `printCoefmat`.

Value

Invisibly returns the `anova.tramME` object.

---

**print.summary.tramME**

*Print method for tramME model summary*

**Description**

Print method for tramME model summary

**Usage**

```r
## S3 method for class 'summary.tramME'
print(
  x,
  fancy = !isTRUE(getOption("knitr.in.progress")) & interactive(),
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

**Arguments**

- **x**: A `summary.tramME` object.
- **fancy**: Logical; if `TRUE`, use color in outputs.
- **digits**: Minimum number of significant digits to be used for most numbers.
- **signif.stars**: Logical; if `TRUE`, P-values are additionally encoded visually as 'significance stars' in order to help scanning of long coefficient tables. It defaults to the `show.signif.stars` slot of `options`.
- **...**: Optional arguments passed to `printCoefmat`.

**Value**

The input `summary.tramME` object, invisibly.
**print.tramME**

*Print tramME model*

---

**Description**

Print tramME model

**Usage**

```r
## S3 method for class 'tramME'
print(x, digits = max(getOption("digits") - 2L, 3L), ...)
```

**Arguments**

- `x`: A `tramME` object.
- `digits`: Number of significant digits
- `...`: Optional arguments (for consistency with the generic)

**Value**

The original `tramME` object invisibly

---

**print.VarCorr.tramME**

*Print method for the variance-correlation parameters of a tramME object*

---

**Description**

Print method for the variance-correlation parameters of a tramME object

**Usage**

```r
## S3 method for class 'VarCorr.tramME'
print(x, sd = TRUE, digits = max(getOption("digits") - 2L, 3L), ...)
```

**Arguments**

- `x`: A `VarCorr.tramME` object.
- `sd`: Logical. Print standard deviations instead of variances.
- `digits`: Number of digits
- `...`: Optional arguments

**Value**

Invisibly returns the input `VarCorr.tramME` object.
\textbf{ranef.LmME} \hfill \textit{Extract the conditional modes of random effects of an LmME model}

\section*{Description}

The \texttt{condVar} option is not implemented for \texttt{ranef.LmME}. Setting \texttt{raw=TURE} will return the raw random effects estimates from the transformation model parametrization.

\section*{Usage}

\begin{verbatim}
## S3 method for class 'LmME'
ranef(object, as.lm = FALSE, ...)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{object} \hfill A fitted LmME object.
  \item \texttt{as.lm} \hfill If \texttt{TRUE}, return the transformed conditional modes as in a normal linear mixed effects model.
  \item \texttt{...} \hfill Optional parameters passed to \texttt{ranef.tramME}.
\end{itemize}

\section*{Value}

A numeric vector or a \texttt{ranef.tramME} object depending on the inputs.

\section*{Examples}

\begin{verbatim}
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
ranef(fit, raw = TRUE) ## transformation model parametrization!
ranef(fit, as.lm = TRUE)
\end{verbatim}

\textbf{ranef.tramME} \hfill \textit{Point estimates and conditional variances of random effects.}

\section*{Description}

Extract the conditional modes and conditional variances of random effects in a formatted or unformatted way.
Usage

```r
## S3 method for class 'tramME'
ranef(
  object,
  param = NULL,
  newdata = NULL,
  fix_smooth = !is.null(newdata),
  condVar = FALSE,
  raw = FALSE,
  ...
)
```

Arguments

- `object`: A `tramME` object.
- `param`: An optional named list of parameter values (beta and theta). See details. Optionally, gamma elements can also be added, which leads to 'fixing' those random effects terms at the supplied values.
- `newdata`: An optional `data.frame` of new observations for which the new random effects values are predicted.
- `fix_smooth`: Logical; it is set to `TRUE` by default, if `newdata` is supplied. The random effects parameters corresponding the smooth terms are fixed and not fitted (posterior mode) to `newdata` instead they are treated just like fixed effects parameters. See details.
- `condVar`: If `TRUE`, include the conditional variances as attributes. Only works with `raw = FALSE`.
- `raw`: Return the unformatted RE estimates as fitted by the model.
- `...`: Optional arguments (for consistency with generic)

Details

- `raw = TRUE` returns the whole vector of random effects (i.e. with parameters of smooth shift terms), while `raw = FALSE` only returns the formatted list of actual random effects (i.e. for grouped observations) values. For the conceptual differences between the two types of random effects, see Hodges (2014, Chapter 13).
- The conditional variances of the fixed random effects are set to `NA`.

Value

Depending on the value of `raw`, either a numeric vector or a `ranef.tramME` object which contains the conditional mode and variance estimates by grouping factors.

Warning

The function has several optional arguments that allow great flexibility beyond its most basic usage. The user should be careful with setting these, because some combinations might not return sensible results. Only limited sanity checks are performed.
References


Examples

data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy, order = 5)
ranef(fit, raw = TRUE)
ranef(fit)

residuals.LmME  Residuals of a LmME model

Description

Calculates the score residuals of an intercept term fixed at 0. In the case of an LmME model, this is equal to the residual of an LMM.

Usage

## S3 method for class 'LmME'
residuals(object, as.lm = FALSE, ...)

Arguments

object       An LmME object.

as.lm       If TRUE, return the residuals as in a normal linear mixed effects model.

...          Optional arguments (for consistency with generic)

Examples

data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
resid(fit)
residuals.tramME

Residuals of a tramME model

Description

Calculates the score residuals of an intercept term fixed at 0.

Usage

```r
## S3 method for class 'tramME'
residuals(
  object,
  param = NULL,
  newdata = NULL,
  fix_smooth = !is.null(newdata),
  ...
)
```

Arguments

- `object`: A tramME object.
- `param`: An optional named list of parameter values (beta and theta). See details. Optionally, gamma elements can also be added, which leads to 'fixing' those random effects terms at the supplied values.
- `newdata`: An optional data.frame of observations for which we want to calculate the residuals.
- `fix_smooth`: Logical; it is set to TRUE by default, if newdata is supplied. The random effects parameters corresponding the smooth terms are fixed and not fitted (posterior mode) to newdata instead they are treated just like fixed effects parameters. See details.
- `...`: Optional arguments (for consistency with generic)

Examples

```r
library("survival")
fit <- SurvregME(Surv(time, status) ~ rx + (1 | litter), data = rats)
resid(fit)
```
### sigma.LmME

*Extract the SD of the error term of an LmME model.*

**Description**

Extract the SD of the error term of an LmME model.

**Usage**

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

**Arguments**

- `object`: An LmME object.
- `...`: Optional argument (for consistency with generic).

**Value**

A numeric value of the transformed sigma parameter.

**Examples**

```r
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
sigma(fit)
```

---

### smooth_terms.LmME

*Evaluate smooth terms of a LmME model.*

**Description**

Evaluate smooth terms of a LmME model.

**Usage**

```r
## S3 method for class 'LmME'
smooth_terms(object, as.lm = FALSE, k = 100, newdata = NULL, ...) 
```
Arguments

- **object**: A `tramME` object.
- **as.lm**: Logical; if `TRUE` return the rescaled values according to a LMM parametrization.
- **k**: Integer, the number of points to be used to evaluate the smooth terms. Ignored when `newdata` is supplied.
- **newdata**: A `data.frame` with new values for the smooth terms. If `NULL`, the new data is set up based on the `model.frame` and `k`. Smooths for which the supplied information in this input is incomplete will be ignored.
- **...**: Optional arguments. `as.lm` is passed through this when it is necessary.

Value

A list of results from evaluating the smooth terms of the model.

Examples

```r
data("mcycle", package = "MASS")
fit <- LmME(accel ~ s(times), data = mcycle)
plot(smooth_terms(fit, as.lm = TRUE))
```

smooth_terms.tramME  
*Extract and evaluate the smooth terms of a tramME model*

Description

Extract and evaluate the smooth terms of a tramME model.

Usage

```r
## S3 method for class 'tramME'
smooth_terms(object, k = 100, newdata = NULL, ...)
```

Arguments

- **object**: A `tramME` object.
- **k**: Integer, the number of points to be used to evaluate the smooth terms. Ignored when `newdata` is supplied.
- **newdata**: A `data.frame` with new values for the smooth terms. If `NULL`, the new data is set up based on the `model.frame` and `k`. Smooths for which the supplied information in this input is incomplete will be ignored.
- **...**: Optional arguments. `as.lm` is passed through this when it is necessary.

Value

A list of results from evaluating the smooth terms of the model.
Examples

```r
data("mcycle", package = "MASS")
fit <- LmME(accel ~ s(times), data = mcycle)
plot(smooth_terms(fit))
```

summary.tramME

Summary method for tramME model

Description

Summary method for tramME model

Usage

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

Arguments

- `object` A tramME object
- `...` Optional arguments (for consistency with the generic)

Value

A summary.tramME object.

SurvregME

Mixed-effects version of Survreg

Description

Mixed-effects version of Survreg

Usage

```r
SurvregME(
    formula,
    data, 
    subset, 
    weights, 
    offset, 
    na.action = na.omit,
    dist = c("weibull", "logistic", "gaussian", "exponential", "rayleigh", "loggaussian", 
        "lognormal", "loglogistic"), 
    scale = 0,
```
SurvregME

```r
silent = TRUE,
resid = FALSE,
do_update = FALSE,
estinit = TRUE,
initpar = NULL,
fixed = NULL,
nofit = FALSE,
control = optim_control(),
...
)
```

Arguments

- `formula`: an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
- `data`: an optional data frame, list or environment (or object coercible by `as.data.frame`) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
- `subset`: an optional vector specifying a subset of observations to be used in the fitting process.
- `weights`: an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
- `offset`: this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
- `na.action`: a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset.
- `dist`: character defining the conditional distribution of the (not necessarily positive) response, current choices include Weibull, logistic, normal, exponential, Rayleigh, log-normal (same as log-gaussian), or log-logistic.
- `scale`: a fixed value for the scale parameter(s).
- `silent`: logical, make TMB functionality silent
- `resid`: logical, Should the score residuals also be calculated?
- `do_update`: Logical. If `TRUE`, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.
- `estinit`: logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model
- `initpar`: named list of initial parameter values, if `NULL`, it is ignored
- `fixed`: a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix
- `nofit`: logical, if `TRUE`, creates the model object, but does not run the optimization
- `control`: list with controls for optimization
- `...`: additional arguments to `tram`. 
Value

A SurvregME object.

Description

The specific model types (LmME, BoxCoxME, ColrME, etc.) are wrappers around this function.

Usage

tramME(
  formula,
  tram,
  call,
  data,
  subset,
  weights,
  offset,
  na.action,
  silent = TRUE,
  resid = FALSE,
  do_update = FALSE,
  estinit = TRUE,
  initpar = NULL,
  fixed = NULL,
  nofit = FALSE,
  control = optim_control(),
  ...
)

Arguments

formula an object of class "formula": a symbolic description of the model structure to be fitted. The details of model specification are given under tram and in the package vignette.

tram Parameter vector for the tram model type.

call The original function call (to be passed from the wrapper).

data an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula).

subset an optional vector specifying a subset of observations to be used in the fitting process.
weights an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If present, the weighted log-likelihood is maximised.

offset this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases.

na.action a function which indicates what should happen when the data contain NAs. The default is set to na.omit.

silent Logical. Make TMB functionality silent.

resid Logical. If TRUE, the score residuals are also calculated. This comes with some performance cost.

do_update Logical. If TRUE, the model is set up so that the weights and the offsets are updateable. This comes with some performance cost.

estinit logical, estimate a vector of initial values for the fixed effects parameters from a (fixed effects only) mlt model

initpar named list of initial parameter values, if NULL, it is ignored

fixed a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix

nofit logical, if TRUE, creates the model object, but does not run the optimization

control list with controls for optimization

... additional arguments to tram.

**Warning**

You should not call directly this function. Only exported for technical reasons.

---

**tramME_model**  
*Create an object that defines a tramME_model*

---

**Description**

There are two ways of defining tramME models:

1. A ctm model and a formula defining the random effects and smooth terms.
2. A formula combining the notation of tram, lme4 and mgcv, a tram function name, and a dataset to set up the bases.

**Usage**

```r
tramME_model(
  formula = NULL,
  data = NULL,
  tram = NULL,
  ctm = NULL,
  smooth = NULL,
  negative = NULL,
  ...)
```


**Arguments**

- **formula**: formula that either describes the whole model or the random effects specification. If the model contains random effects or smooth terms, `formula` has to contain their definition in `lme4`-style and `mgcv`-style notation, respectively.
- **data**: an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
- **tram**: trm model name: Lm, BoxCox, Colr, Polr, Coxph, Survreg, Lehmann, Aareg, or the suffixed versions of these (e.g. ColrME). Ignored when a ctm model is also supplied.
- **ctm**: A ctm model
- **smooth**: Optional pre-defined smooth specification of the class `tramME_smooth`. If present, the smooth terms in the formula are ignored.
- **negative**: an optional parameter that defines whether the random effects have a positive or a negative sign in the model when the fixed effects part is defined through a ctm.
- **...**: optional arguments passed to `tram` when the model is defined by the formula.

**Value**

A `tramME_model` object that defines the mixed effects transformation model.

**Note**

Similarly to `mlt`, the offsets and the weights are not part of the model, but they are data and they are not saved in the returned object.

---

**Description**

Create a `tramTMB` object

**Usage**

```r
tramTMB(
  data,
  parameters,
  constraint,
  negative,
  map = list(),
  resid = FALSE,
  do_update = FALSE,
  check_const = TRUE,
  no_int = FALSE,
  ...
)
```
VarCorr.LmME

Arguments

- **data**: List of data objects (vectors, matrices, arrays, factors, sparse matrices) required by the user template (order does not matter and un-used components are allowed).
- **parameters**: List of all parameter objects required by the user template (both random and fixed effects).
- **constraint**: List describing the constraints on the parameters.
- **negative**: Logical, whether the model is parameterized with negative values.
- **map**: Same as map argument of TMB::MakeADFun.
- **resid**: Logical, indicating whether the score residuals are calculated from the resulting object.
- **do_update**: Logical, indicating whether the model should be set up with updateable offsets and weights.
- **check_const**: Logical; if TRUE check the parameter constraints before evaluating the returned functions.
- **no_int**: Logical; if FALSE skip the numerical integration step.
- **...**: Optional parameters passed to TMB::MakeADFun.

Value

A tramTMB object.

Note

The post-estimation parameters are supplied as a part of data.

---

VarCorr.LmME

Variance and correlation matrices of random effects of an LmME object

Description

The returned parameters are the transformed versions of the original parameters that correspond to the normal linear mixed model parametrization.

Usage

```r
## S3 method for class 'LmME'
VarCorr(x, sigma = 1, as.lm = FALSE, ...)
```
Arguments

x  An LmME object.
sigma  Standard deviation of the error term in the LMM parametrization (should not be set manually; only for consistency with the generic method)
as.lm  If TRUE, return the variances and correlations that correspond to a normal linear mixed model (i.e. lmerMod).
...
Optional arguments (for consistency with generic)

Details

The function only returns the correlation matrices that belong to actual random effects (defined for groups in the data) and ignores the random effects parameters of the smooth shift terms. To extract these, the user should use varcov with full = TRUE.

Value

A list of vectors with variances and correlation matrices corresponding to the various grouping variables.

Examples

data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
VarCorr(fit) # tranformation model parametrization
VarCorr(fit, as.lm = TRUE) # LMM parametrization

---

VarCorr.tramME  Variances and correlation matrices of random effects

Description

This function calculates the variances and correlations from varcov.tramME.

Usage

## S3 method for class 'tramME'
VarCorr(x, ...)

Arguments

x  A tramME object
...
optional arguments (for consistency with the generic method)

Details

The function only returns the correlation matrices that belong to actual random effects (defined for groups in the data) and ignores the random effects parameters of the smooth shift terms. To extract these, the user should use varcov with full = TRUE.
A list of vectors with variances and correlation matrices corresponding to the various grouping variables.

```r
data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
VarCorr(fit)
```

Generic method for `varcov`

```r
varcov(object, ...)  
```

- **object**: A model object.
- **...**: Optional inputs.

A variance-covariance matrix.

Extract the variance-covariance matrix of the random effects of an LmME model

```r
## S3 method for class 'LmME'
varcov(object, as.lm = FALSE, as.theta = FALSE, full = FALSE, ...)
```

Extract the variance-covariance matrix of the random effects of an LmME model

```r
## S3 method for class 'LmME'
varcov(object, as.lm = FALSE, as.theta = FALSE, full = FALSE, ...)
```
Arguments

object A \texttt{LmME} object.

as.lm If \texttt{TRUE}, the returned values correspond to the \texttt{LMM} parametrization.

as.theta Logical value, if \texttt{TRUE}, the values are returned in their reparameterized form.

full Logical value; if \texttt{TRUE}, return all random effects elements, if \texttt{FALSE}, do not return the random effects parameters of the smooth terms.

... Optional arguments (unused).

Value

A list of the covariance matrices or a vector of theta values.

Examples

data("sleepstudy", package = "lme4")
fit <- \texttt{LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)}
\texttt{varcov(fit, as.lm = TRUE)}
\texttt{varcov(fit, as.theta = TRUE, as.lm = TRUE)}

varcov.tramME

\textit{Extract the variance-covariance matrix of the random effects}

Description

Returns the covariance matrix of the random effects as saved in the \texttt{tramME} object. The returned values correspond to the transformation model parametrization.

Usage

\#
\texttt{S3 method for class \textquotesingle tramME\textquotesingle}
\texttt{varcov(object, as.theta = FALSE, full = FALSE, \ldots)}

Arguments

object A \texttt{tramME} object.

as.theta Logical value, if \texttt{TRUE}, the values are returned in their reparameterized form.

full Logical value; if \texttt{TRUE}, return all random effects elements, if \texttt{FALSE}, do not return the random effects parameters of the smooth terms.

... Optional arguments (unused).

Value

A list of the covariance matrices or a vector of theta values.
Examples

```r
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
varcov(fit)
varcov(fit, as.theta = TRUE)
```

---

**Description**

Generic method for "varcov<-"

**Usage**

```r
varcov(object, ...) <- value
```

**Arguments**

- `object` A model object.
- `...` Optional inputs.
- `value` The new value of the covariance matrix.

**Value**

An object with the same class as `object`, with updated variance-covariance matrix of random effects.

---

**varcov<-.tramME**

Set the values of the random effects covariance matrices of a tramME model.

---

**Description**

Sets the list containing the covariance matrices of a tramME model. The matrices have to be positive definite. Just as in "coef<-", when the function is called on a fitted object, the function will remove the information about the optimization.

**Usage**

```r
## S3 replacement method for class 'tramME'
varcov(object, as.theta = FALSE, ...) <- value
```
variable.names.tramME

Arguments

object A tramME object.
as.theta Logical value, if TRUE, indicating that the new values are supplied in their reparameterized form.
... Optional arguments (ignored).
value A list of positive definite covariance matrices.

Details

The supplied list has to be named with the same names as implied by the model. Hence, it might be a good idea to call varcov first, and modify this list to make sure that the input has the right structure.

The new values can also be supplied in a form that corresponds to the reparametrization used by the tramTMB model (see the option as.theta = TRUE).

All random effects variance parameters must be supplied. When there are penalized smooth terms in the model variance parameters corresponding to these should also be part of the input list.

Value

A new tramME object with the new coefficient values.

Examples

data("sleepstudy", package = "lme4")
mod <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy, nofit = TRUE)
vc <- varcov(mod)
vc[[1]] <- matrix(c(1, 0, 0, 2), ncol = 2)
varcov(mod) <- vc

variable.names.tramME  Return variable names.

Description

Returns the variable names corresponding to different variable groups in a tramME model.

Usage

## S3 method for class 'tramME'
variable.names(
  object,
  which = c("all", "response", "grouping", "shifting", "interacting", "smooth", "ranef"),
  ...
)
Arguments

- object: a tramME object (fitted or unfitted)
- which (optional parameters)
  - all: all variables,
  - response: response variable,
  - grouping: grouping factors for random effects,
  - shifting: shifting variables,
  - interacting: interacting variables,
  - smooth: variables in smooth terms,
  - ranef: all random effects variables (covariates with random slopes and grouping factors).

Details

The returned names are the names as they are used by tramME. For example, when the response is a Surv object, variable.names returns the name of that object, and not the names of the variables used to create it.

Value

A vector of variable names.

Examples

```r
library(lme4)

# Load data
data("sleepstudy", package = "lme4")

# Fit LmME model
mod <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy, nofit = TRUE)

# Get variable names
variable.names(mod)
variable.names(mod, "response")
```

Description

Get the variance-covariance matrix of the parameters of an LmME model.

Usage

```r
## S3 method for class 'LmME'
vcov(
  object,
  as.lm = FALSE,
  parm = NULL,
  pargroup = c("all", "fixef", "ranef"),
  ...
)
```
Arguments

- **object**: A fitted LmME object.
- **as.lm**: If TRUE, return the covariance matrix of the same parametrization as used by `lmer`.
- **parm**: Names of the parameters to extract.
- **pargroup**: The name of the parameter group to extract. With `as.lm = FALSE`, the available options are described in `confint.tramME`. When `as.lm = TRUE`, the following options are available:
  - **all**: Fixed effects and variance components parameters.
  - **fixef**: Fixed effects parameters (including FE parameters of the smooth terms).
  - **ranef**: Variance components parameters (including the smoothing parameters of the random effects).
- ... Optional parameters passed to `confint.tramME`

Value

A numeric covariance matrix.

Examples

```r
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
vcov(fit) ## transformation model parametrization
vcov(fit, as.lm = TRUE) ## LMM parametrization
vcov(fit, as.lm = TRUE, pargroup = "fixef") ## cov of fixed effects
```

---

**vcov.tramME**

*Calculate the variance-covariance matrix of the parameters*

Description

Extracts the covariance matrix of the selected parameters. The returned values are on the same scale as the estimated parameter values, i.e. the standard deviations of the random effect terms are on log scale.

Usage

```r
## S3 method for class 'tramME'
vcov(
  object,
  parm = NULL,
  pargroup = c("all", "fixef", "shift", "baseline", "ranef", "smooth"),
  pmatch = FALSE,
  ...
)
```
vcov.tramTMB

Variance-covariance matrix of the parameters

Description

Variance-covariance matrix of the parameters

Arguments

- **object**: A fitted tramME object.
- **parm**: The indeces or names of the parameters of interest. See in details.
- **pargroup**: The name of the parameter group to return:
  - all: All parameters.
  - fixef: Fixed effects parameters.
  - shift: Shift parameters.
  - baseline: Parameters of the baseline transformation function.
  - ranef: Variance components parameters.
  - smooth: Parameters that belong to the smooth shift terms (both FE and smoothing parameters).
- **pmatch**: Logical. If TRUE, partial name matching is allowed.
- **...**: Optional arguments passed to vcov.tramTMB

Details

The argument parm defines the indices or the names of the parameters of interest within the selected pargroup. When pmatch = TRUE, partial matching of parameter names is allowed.

Value

A numeric covariance matrix.

Examples

```r
data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy, order = 10)
vcov(fit)
vcov(fit, pargroup = "ranef")
vcov(fit, pargroup = "baseline")
vcov(fit, parm = "Reaction") ## same as previous
```
Usage

```r
## S3 method for class 'tramTMB'
vcov(
  object,
  par = object$env$par_checked,
  method = c("optimHess", "numDeriv", "analytical"),
  control = list(),
  ...
)
```

Arguments

- **object** A `tramTMB` object.
- **par** An optional vector of parameter values.
- **method** Method for calculating the covariance matrix.
- **control** Optional named list of controls to be passed to the specific methods.
- **...** Optional arguments (ignored)
Index

anova.tramME, 3
as.data.frame, 20
auglag, 23
BoxCox, 4
BoxCoxME, 4, 40
coeff.LmME, 5
coeff.SurvregME, 6
coeff.tramME, 6
coeff<-.tramME, 7
Colr, 8
ColrME, 8, 40
confint.LmME, 9
confint.tramME, 10
Coxph, 12
CoxphME, 12
duplicate, 13
duplicate.tramTMB, 14
edf_smooth (edf_smooth.tramME), 14
edf_smooth.tramME, 14
Lehmann, 15
LehmannME, 15
Lm, 16
lmer, 50
LmME, 16, 40
logLik.tramME, 18
mkReTrms, 21
mlt, 21
model.frame, 20
model.frame.tramME, 20
model.matrix.tramME, 21
nlminb, 23
optim, 23
optim_control, 23
options, 30
plot.mlt, 25
plot.smooth.tramME, 23
plot.tramME, 24
Polr, 25
PolrME, 25
predict.mlt, 25, 27, 28
predict.tramME, 25, 27
predict.tramTMB, 29
print.anova.tramME, 29
print.summary.tramME, 30
print.tramME, 31
print.VarCorr.tramME, 31
printCoefmat, 30
ranef (ranef.tramME), 32
ranef.LmME, 32
ranef.tramME, 18, 32
residuals.LmME, 34
residuals.tramME, 18, 35
sigma.LmME, 36
smooth_terms (smooth_terms.tramME), 37
smooth_terms.LmME, 36
smooth_terms.tramME, 37
summary.tramME, 38
Surv, 21
Survreg, 38
SurvregME, 38
tram, 4, 5, 8, 9, 12, 13, 16–18, 26, 39–41
tramME, 40
tramME_model, 41
tramTMB, 42
VarCorr (VarCorr.tramME), 44
VarCorr.LmME, 43
VarCorr.tramME, 44
varcov, 45
varcov.LmME, 45
varcov.tramME, 46
varcov<-, 47
varcov<-.tramME, 47
variable.names.tramME, 48
vcov.LmME, 49
vcov.tramME, 50
vcov.tramTMB, 51