Package 'tramME'

July 31, 2020

Title Transformation Models with Mixed Effects

Version 0.0.3

Description Likelihood-based estimation of mixed-effects transformation models using the Template Model Builder (TMB). The technical details of transformation models are given in Hothorn et al. (2018) <doi:10.1111/sjos.12291>. 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 (AD).

Depends R (>= 3.6.0), tram (>= 0.3.2), mlt (>= 1.1.0)

Imports alabama, lme4 (>= 1.1.19), Matrix, methods, nlme, TMB (>= 1.7.15), stats, variables (>= 1.0.2), basefun (>= 1.0.6)

Suggests multcomp, parallel, survival, knitr, coxme, ordinal, ordinalCont, ggplot2

LinkingTo TMB, RcppEigen

VignetteBuilder knitr

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

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 2020-07-31 08:30:03 UTC

R topics documented:

`.check_coef` *Check the validity of a vector of coefficients*

Description

Check the validity of a vector of coefficients

Usage

```
.check_coef(x, mst)
```

Arguments

`x`	Vector of coefficients
`mst`	List describing the model structure

`.check_varcov` *Check the validity of a vector of coefficients*

Description

Check the validity of a vector of coefficients

Usage

```
.check_varcov(x, vc)
```

Arguments

`x`	List of covariance matrices
`vc`	varcov from the tramME object

.dummy_ctm *Dummy ctm model with random effects as offsets*

Description

Dummy ctm model with random effects as offsets

Usage

```
.dummy_ctm(mst, coef)
```

Arguments

mst	List describing the model structure
coef	Coefficient vector for the dummy ctm model

.mlt_data *Extract information from an mlt model*

Description

Extract information from an mlt model

Usage

```
.mlt_data(mod)
```

Arguments

mod	mlt model

.model_name *Create name for a specific tramME model*

Description

Create name for a specific tramME model

Usage

```
.model_name(mst)
```

Arguments

mst	model structure list

.nobars *Remove random effects terms*

Description

Remove random effects terms

Usage

```
.nobars(term)
```

Arguments

term	Call or formula

.parallel_default *Boilerplate parallel-handling function, copied from* lme4

Description

Boilerplate parallel-handling function, copied from lme4

Usage

```
.parallel_default(parallel = c("no", "multicore", "snow"), ncpus = 1L)
```

Arguments

parallel	Parallel backend
ncpus	Number of cores/cpus

.paridx *Match parameters with parameter groups and indeces*

Description

Match parameters with parameter groups and indeces

Usage

```
.paridx(mst, which = NULL, pargroup = "all", pmatch = FALSE, altpar = NULL)
```

Arguments

mst	model structure as in an obj$model
which	parameter name or index within group
pargroup	parameter group
pmatch	partrial matching allowed
altpar	optional alternative parametrization such as LMM (altpar = "lm")

.renames *Return names for random effects parameters*

Description

Return names for random effects parameters

Usage

```
.renames(rst)
```

Arguments

rst	the random effect structure as saved in obj$model

.re_data *Create RE data*

Description

Create RE data

Usage

```
.re_data(rhsterm, data, negative = FALSE)
```

Arguments

rhsterm	Right-hand-side term of the formula
data	data
negative	should the random effects terms be multiplied with -1?

.re_format *Reshape unformatted random effects vectors*

Description

Reshape unformatted random effects vectors

Usage

```
.re_format(rst, x)
```

Arguments

rst	the random effect structure as saved in obj$model
x	Unformatted random effects vector

.re_size *Calculates the size of the random effect vector implied by the model and the data*

Description

Calculates the size of the random effect vector implied by the model and the data

Usage

```
.re_size(mst, data)
```

Arguments

mst	list describing the model structure
data	Dataset containing the required grouping factors

.sim_re *Simulates random effects vector from a tramME object*

Description

Simulates random effects vector from a tramME object

Usage

```
.sim_re(vc, n)
```

Arguments

vc	list of RE variance-covariances
n	list of number of values to be simulated for each grouping factor

.subbars *My own version of lme4::subbar*

Description

My own version of lme4::subbar

Usage

```
.subbars(term)
```

Arguments

term	Call or formula

.tramME *General ME tram model*

Description

General ME tram model

Usage

```
.tramME(formula, data, na.action, silent, nofit, 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 Details 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).
na.action	a function which indicates what should happen when the data contain NAs. The default is set to na.omit.
silent	Logical, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.

---

anova.tramME *Comparison of nested tramME models.*

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 fitted tramME model.
object2	A fitted tramME model.
...	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 *ME version of tram::BoxCox*

Description

ME version of tram::BoxCox

Usage

```
BoxCoxME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	additional arguments to tram.

Value

A BoxCoxME object.

`coef.LmME` *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

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

Arguments

`object`	An LmME object (fitted or unfitted).
`as.lm`	If TRUE, return the transformed coefficients as in a `lmerMod` object.
`...`	optional parameters passed to `coef.tramME`

Value

A numeric vector of the transformed coefficients.

Examples

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

Arguments

object	A tramME object (fitted or unfitted)
with_baseline	If TRUE, include the baseline parameters, too.
...	Optional parameters (ignored).

Value

Numeric vector of parameter values.

Examples

```
data("sleepstudy", package = "lme4")
mod <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy, nofit = TRUE)
coef(mod, with_baseline = TRUE)
```

---

coef<-.tramME *Set coefficients of a tramME model.*

Description

Sets the whole vector of 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. When called on a fitted tram object, the function sets it to unfitted and removes all parts that come from the estimation.

Usage

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

Arguments

object	A tramME object (fitted or unfitted).
value	Numeric vector of new coefficient values.

Value

An unfitted tramME object with the new coefficient values.

Examples

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

ColrME *ME version of tram::Colr*

Description

ME version of tram::Colr

Usage

```
ColrME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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 by the na.action setting of options, and is na.fail if that is unset.
silent	Logical, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	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

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

Arguments

`object`	A fitted LmME object.
`parm`	The indeces or names of the parameters of interest. See in details.
`level`	Confidence level.
`as.lm`	Logical. If `TRUE`, return results consistent with the normal linear mixed model parametrization.
`pargroup`	fixef: fixed-effects, shift: shift parameters, all: fixed effects and variance component parameters, baseline: parameters of the baseline transformation function, ranef: variance components parameters.
`type`	Type of the CI: either Wald or profile.
`estimate`	Logical, add the point estimates in a thrid column
`pmatch`	Logical. If `TRUE`, partial name matching is allowed.
`...`	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
```

`confint.tramME` *Confidence intervals for tramME model parameters*

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

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

Arguments

`object`	A fitted tramME object.
`parm`	The indeces or names of the parameters of interest. See in details.
`level`	Confidence level.
`pargroup`	fixef: fixed-effects, shift: shift parameters, all: fixed effects and variance component parameters, baseline: parameters of the baseline transformation function, ranef: variance components parameters.
`type`	Type of the CI: either Wald or profile.
`estimate`	Logical, add the point estimates in a thrid 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

```
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 *ME version of tram::Coxph*

Description

ME version of tram::Coxph

Usage

```
CoxphME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	additional arguments to tram.

Value

A CoxphME object.

LehmannME *ME version of tram::Lehmann*

Description

ME version of tram::Lehmann

Usage

```
LehmannME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	additional arguments to tram.

Value

A LehmannME object.

LmME *ME version of tram::Lm*

Description

ME version of tram::Lm

Usage

```
LmME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	additional arguments to tram.

Value

A LmME object.

`logLik.tramME` *Get the log-likelihood of the model*

Description

Get the log-likelihood of the model

Usage

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

Arguments

object	A fitted tramME model
...	Optional argument (for consistency with generic)

Value

A numeric value of the log-likelihood at its optimum.

Examples

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

plot.trafo.tramME *Plotting method for* trafo.tramME *objects*

Description

Plotting method for trafo.tramME objects

Usage

```
## S3 method for class 'trafo.tramME'
plot(x, col = 1, fill = "lightgrey", lty = 1, add = FALSE, ...)
```

Arguments

x	A trafo.tramME object.
col	Line colors, recycled if shorter than the size of the trafo.tramME object.
fill	Fill color for the confidence intervals.
lty	Line types.
add	If TRUE add to an existing plot.
...	Additional arguments, passed to plot or lines.

Value

The original trafo.tramME object, invisibly.

Examples

```
data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
tr <- trafo(fit, type = "trafo", confidence = "interval", K = 100)
plot(tr, col = 2, main = "Trafo")
```

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

```
## S3 method for class 'tramME'
plot(x, newdata = NULL, ranef = NULL, ...)
```

Arguments

x	A tramME object
newdata	an optional data frame of observations
ranef	Vector of random effects or the word "zero". 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.

Value

A numeric matrix of the predicted values invisibly

Examples

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

PolrME *ME version of tram::Polr*

Description

ME version of tram::Polr

Usage

```
PolrME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  method = c("logistic", "probit", "loglog", "cloglog"),
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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 by the na.action setting of options, and is na.fail if that is unset.
method	a character describing the link function.
silent	Logical, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	additional arguments to tram.

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 desired scale. When newdata contains values of the response variable, prediction is only done for those values. When no response values are supplied, prediction is done on a grid of values. Unfitted tramME models can also be used for prediction as long as the coefficent parameter are set manually (with coef<-).

Usage

```
## S3 method for class 'tramME'
predict(object, newdata = NULL, ranef = NULL, ...)
```

Arguments

object	A tramME object
newdata	an optional data frame of observations
ranef	Vector of random effects or the word "zero". See details.
...	Additional arguments, passed to predict.mlt.

Details

When ranef is equal to "zero", a vector of zeros with the right size is substituted.

Value

A numeric matrix of the predicted values invisibly

Examples

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

---

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 retrurns the anova.tramME object.

---

print.simulate.tramME *Print method for* simulate.tramME *objects*

Description

Automatically hides the seed attribute of the object.

Usage

```
## S3 method for class 'simulate.tramME'
print(x, suppress_seed = TRUE, ...)
```

Arguments

x	A simulate.tramME object
suppress_seed	Logical, suppress seed if true.
...	Additional parameters passed to various print methods.

Value

The input simulate.tramME object, invisibly.

print.summary.tramME *Print method for tramME model summary*

Description

Print method for tramME model summary

Usage

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

Arguments

x	a summary.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

The input summary.tramME object, invisibly.

print.tramME *Print tramME model*

Description

Print tramME model

Usage

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

Arguments

x	A fitted or unfitted tramME model
digits	Number of significant digits
...	Optional arguments (for consistency with the generic)

Value

The original tramME obejct 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

```
## 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.

ranef.LmME *Extract the conditional modes of random effects of an LmME model*

Description

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

Usage

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

Arguments

object	A fitted LmME object.
as.lm	If TRUE, return the transformed conditional modes as in a normal linear mixed effects model.
...	Optional parameters passed to ranef.tramME.

Value

A numeric vector or a ranef.tramME object depending on the inputs.

Examples

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

ranef.tramME *Extract the conditional modes and conditional variances of random effects*

Description

Extract the conditional modes and conditional variances of random effects

Usage

```
## S3 method for class 'tramME'
ranef(object, condVar = FALSE, raw = FALSE, ...)
```

Arguments

object	A fitted tramME object.
condVar	If TRUE, include the conditional variances as attributes.
raw	Return the unformatted RE estimates as fitted by the model.
...	Optional arguments (for consistency with generic)

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.

Examples

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

---

refit.tramME *Refit the model with a new response vector*

Description

Useful for parametric bootstrap.

Usage

```
## S3 method for class 'tramME'
refit(object, newresp, ...)
```

Arguments

object	A tramME object.
newresp	A vector of new response values.
...	optional arguments for compatibility

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

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

Arguments

object	An LmME object (fitted or unfitted).
...	Optional argument (for consistency with generic)

Value

A numeric value of the transformed sigma parameter.

Examples

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

simulate.tramME *Simulate outcome variable from an estimated model*

Description

Utilizes the simulation method of mlt. When the vector of random effects is supplied, the simulation is conditional on it.

Usage

```
## S3 method for class 'tramME'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata = NULL,
  ranef = NULL,
  what = c("response", "ranef", "joint"),
  bysim = TRUE,
  ...
)
```

Arguments

object	A fitted tramME object.
nsim	number of samples to generate
seed	optional seed for the random number generator
newdata	an optional data frame of observations
ranef	If NULL, random effects are simulated from their estimated distribution for each draw in nsim, i.e. the simulation is from the marginal/joint distribution of the response (and random effects). Otherwise the simulation is conditional on the supplied random effects. When ranef = "zero", a vector of zeros with the right size is substituted.
what	Defaults to 'response'. 'ranef' returns draws from the random effects distribution, 'joint' results in simulated data from the joint distribution of random effects and responses. When it is set to other than 'response', ranef=NULL and bysim=TRUE must be set.
bysim	logical, if TRUE a list with nsim elements is returned, each element is of length nrow(newdata) and contains one sample from the conditional distribution for each row of newdata. If FALSE, a list of length nrow(newdata) is returned, its ith element of length nsim contains nsim samples from the conditional distribution given newdata[i,].
...	Additional arguments, passed to simulate.mlt.

Value

A simulate.tramME object with the structure defined by the inputs.

Examples

```
data("sleepstudy", package = "lme4")
fit <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
sim <- simulate(fit, nsim = 10, seed = 123)
```

---

summary.tramME *Summary method for tramME model*

Description

Summary method for tramME model

Usage

```
## 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 *ME version of tram::Survreg*

Description

ME version of tram::Survreg

Usage

```
SurvregME(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
 dist = c("weibull", "logistic", "gaussian", "exponential", "rayleigh", "loggaussian",
    "lognormal", "loglogistic"),
  scale = 0,
  silent = TRUE,
  nofit = FALSE,
  optim_control = list(outer = list(), optim = list()),
  ...
)
```

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 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, if TRUE, prints all tracing information.
nofit	Logical, if TRUE, creates the model objects, but does not run the optimization.
optim_control	List of optional arguments for the optimizer.
...	additional arguments to tram.

Value

A SurvregME object.

Warning

Fixing the scale parameter is currently not available.

---

trafo *Generic method for extracting baseline transformations*

Description

Generic method for extracting baseline transformations

Usage

```
trafo(object, ...)
```

Arguments

object	A model object
...	Optional parameters

Value

The value of the baseline transfromation function at certain points.

`trafo.tramME` *Get the baseline transformation function and its confidence interval*

Description

For stratified models, it returns a list of data frames for each stratum.

Usage

```
## S3 method for class 'tramME'
trafo(
  object,
  newdata = NULL,
  type = c("trafo", "distribution", "survivor", "cumhazard"),
  confidence = c("none", "interval", "band", "asymptotic"),
  level = 0.95,
  K = 50,
  ...
)
```

Arguments

`object`	A fitted tramME object.
`newdata`	Values of the interacting terms to be used.
`type`	The scale on which the transformation function is evaluated.
`confidence`	Pointwise confidence interval or confidence band.
`level`	Confidence level.
`K`	Integer, number of points in the grid the function is evaluated on.
`...`	Additional parameters (for consistency with generic)

Value

Matrix or list of matrices containing the point estimates and the confidence intervals.

Examples

```
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
tr <- trafo(fit, type = "distribution", confidence = "interval", K = 100)
```

VarCorr.LmME — *Variances and correlation matrices of random effects of an LmME object*

Description

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

Usage

```
## 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)

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)

Value

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

Examples

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

---

varcov *Generic method for* varcov

Description

Generic method for varcov

Usage

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

Arguments

object	A model object
...	Optional parameters

Value

A variance-covariance matrix.

varcov.tramME *Extract the variance-covariance matrix of the random effects*

Description

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

Usage

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

Arguments

object	A tramME object (fitted or unfitted).
...	Optional arguments (unused)

Value

A list of the covariance matrices.

Examples

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

varcov<- *Generic method for "varcov<-"*

Description

Generic method for "varcov<-"

Usage

```
varcov(object) <- value
```

Arguments

object	A model object
value	The new value of the covariance matrix

Value

An object with the same class as object, with updated variance-covariance matrix.

`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, it will be set to unfitted.

Usage

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

Arguments

`object`	A tramME object (fitted or unfitted).
`value`	A list of positive definite covariance matrices.

Details

The supplied list does not have to be named, and the names will be ignored. When multiple grouping factors are present, the function assumes the same order as in the object to be modified. 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.

Value

An unfitted 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 the selected group. The returned names are derived names as tramME uses them. For example, when the response is a Surv object, `variable.names` returns the name of that object, and the names of the variables used to create it.

Usage

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

Arguments

`object`	a tramME object (fitted or unfitted)
`which`	all: all non-eliminated variable names, response: response variable, grouping: grouping factors for random effects, shifting: shifting variables, interacting: interacting variables.
`...`	optional parameters

Value

A vector of variable names.

Examples

```
data("sleepstudy", package = "lme4")
mod <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy, nofit = TRUE)
variable.names(mod)
variable.names(mod, "response")
```

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

Description

`pargroup = "baseline"` is not available for `LmME` objects.

Usage

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

Arguments

object	A fitted LmME object.
as.lm	If TRUE, return the covariance matrix of the transformed parameters as in a lmerMod object.
parm	The indeces or names of the parameters of interest. See in details.
pargroup	fixef: fixed-effects, shift: shift parameters, all: fixed effects and variance component parameters, baseline: parameters of the baseline transformation function, ranef: variance components parameters.
pmatch	Logical. If TRUE, partial name matching is allowed.
...	Optional arguments

Value

A numeric covariance matrix.

Examples

```
data("sleepstudy", package = "lme4")
fit <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
vcov(fit) ## transformation model parametrization
vcov(fit, as.lm = TRUE) ## LMM parametrization
## cov of coefficient AND other terms with 'Days' in names
vcov(fit, as.lm = TRUE, parm = "Days", pmatch = TRUE)
vcov(fit, as.lm = TRUE, parm = "^Days", pmatch = TRUE) ## var of coefficient only
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

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

Arguments

object	A fitted tramME object.
parm	The indeces or names of the parameters of interest. See in details.
pargroup	fixef: fixed-effects, shift: shift parameters, all: fixed effects and variance component parameters, baseline: parameters of the baseline transformation function, ranef: variance components parameters.
pmatch	Logical. If TRUE, partial name matching is allowed.
...	Optional arguments

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

```
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
```

Index