Package 'tramME'

August 16, 2021

Title Transformation Models with Mixed Effects

Version 0.1.2

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 (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), mvtnorm, numDeriv, MASS, coneproj

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

LinkingTo TMB, RcppEigen

VignetteBuilder knitr

License GPL-2

URL http://ctm.R-forge.R-project.org

Encoding UTF-8

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 2021-08-16 12:30:02 UTC

R topics documented:

`.cctm` *Create a 'conditional' ctm model with random effects as offsets*

Description

Create a 'conditional' ctm model with random effects as offsets

Usage

```
.cctm(mod, coef, negative = FALSE)
```

Arguments

`mod`	A `ctm` model.
`coef`	Coefficient vector for the dummy ctm model.
`negative`	The sign of the random effect term in the corresponding `tramME` model.

`.check_par` *Helper function to check parameter constarints*

Description

Helper function to check parameter constarints

Usage

```
.check_par(obj, par, eps = 1e-07, ...)
```

Arguments

`obj`	A `tramTMB` object
`par`	A parameter vector
`eps`	Tolearnce level
`...`	optional arguments

.combine_formulas *Combine a set of formulas into one (similar to nlme::asOneFormula)*

Description

Combine a set of formulas into one (similar to nlme::asOneFormula)

Usage

```
.combine_formulas(formula, ..., omit = ".")
```

Arguments

formula	the first formula, if it contains a response that will be the response of the resulting formula
...	objects from which a formula can be extracted
omit	parameter vector with variable names to be omitted

.constr_adj *Helper function to adjust the constraints consistently with the parameter restrictions*

Description

Helper function to adjust the constraints consistently with the parameter restrictions

Usage

```
.constr_adj(par, constr, map)
```

Arguments

par	list containing the vector of parameters
constr	list containing the the constraints
map	List defining how to optionally collect and fix parameters - see details.

Value

A list with adjusted constraints.

.ctm2formula *Create a dummy fromula from a ctm object*

Description

Create a dummy fromula from a ctm object

Usage

```
.ctm2formula(ctm)
```

Arguments

ctm	A ctm model from wich the fromula is created

.gen_param *Generator for param closure*

Description

Generator for param closure

Usage

```
.gen_param(par, fe, re, varnames)
```

Arguments

par	Named list of initial parameters (beta and theta).
fe	Necessary information about fixed effects.
re	Necessary infromation about random effects.
varnames	Names of the variables in the model.

.get_cf *Get the coefficient vector*

Description

Get the coefficient vector

Usage

```
.get_cf(obj)
```

Arguments

obj	The tramME object

.get_par *Helper function to extract formatted parameters*

Description

Helper function to extract formatted parameters

Usage

```
.get_par(obj, par = obj$env$par_checked, fixed = TRUE)
```

Arguments

obj	A tramTMB object
par	A parameter vector to be formatted
fixed	Logica; print fixed parameters, too

.idx *Get parameter indices of various structures*

Description

If fixed is logical, it inidcates that the indices refer to the extended parameter vector pargroup = c("fixef", "ranef", "shift", "all")

Usage

```
.idx(
  obj,
  fixed = NULL,
  pargroup = "all",
  which = NULL,
  pmatch = FALSE,
  altpar = NULL
)
```

Arguments

obj	A tramME object
fixed	Logical; should the indices of fixed parameters also be returned?
pargroup	Parameter group
which	Parameter names or indices within groups
pmatch	Is partial matching allowed for which
altpar	Alternative parameterizations (currently only "lm" possible)

.isbars *Check whether formula contains bars (RE parts)*

Description

Check whether formula contains bars (RE parts)

Usage

```
.isbars(f)
```

Arguments

f	formula

.model_name *Generates proper model name for the tramME model*

Description

Generates proper model name for the tramME model

Usage

```
.model_name(obj)
```

Arguments

obj	A tramME object.

.mod_negative *Helper function to figure out if negative = TRUE in a given model*

Description

Helper function to figure out if negative = TRUE in a given model

Usage

```
.mod_negative(ctm, tram = NULL)
```

Arguments

ctm	A ctm model
tram	tram 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.

.nm2mat *Add names to the vc matrix*

Description

Add names to the vc matrix

Usage

```
.nm2mat(m, rnms)
```

Arguments

m	List of covariance matrices without names
rnms	Named list of random effects names

Value

The same list of covariance matrices with proper names

.nm2vec *Add names to the vector of theta parameters*

Description

Add names to the vector of theta parameters

Usage

```
.nm2vec(v, rnms)
```

Arguments

v	theta vector without names
rnms	Named list of random effects names

Value

The same vector with proper names

.nobars *Remove random effects terms from full formulas or calls*

Description

Remove random effects terms from full formulas or calls

Usage

```
.nobars(term)
```

Arguments

term	Call or formula

.parallel_default *Boilerplate parallel-handling function, modified from* glmmTMB

Description

Boilerplate parallel-handling function, modified from glmmTMB

Usage

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

Arguments

parallel	Parallel backend
ncpus	Number of cores/cpus

.re_format *Format random effects*

Description

Format random effects

Usage

```
.re_format(x, rts, rnms, rbs, rlev)
```

Arguments

x	Vector of random effects
rts	Vector of length of random effects terms for each grouping factor
rnms	Named list of random effects names
rbs	Vector of random effects matrix dimensions
rlev	List of the levels of each grouping factor

.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(bs, data)
```

Arguments

bs	Blocksize vector as returned by the function re_terms.
data	Dataset containing the required grouping factors.

.set_cf *Set the coefficient vector*

Description

Set the coefficient vector

Usage

```
.set_cf(obj, val)
```

Arguments

obj	The tramME object
val	The vector of new values

Value

A new list of parameters with updated beta part

.set_vc *Set the parameters of the random effect covariance matrix*

Description

Set the parameters of the random effect covariance matrix

Usage

```
.set_vc(obj, val, as.theta = FALSE)
```

Arguments

obj	The tramME object
val	The vector of new values
as.theta	The input is given according to the reparameterization used by tramTMB

Value

A new list of parameters with updated beta part

.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

.th2vc *Convert from theta vector to vc matrix*

Description

Convert from theta vector to vc matrix

Usage

```
.th2vc(th, rbs)
```

Arguments

th	Vector of theta parameters (reparametrization of the covariance matrices)
rbs	Vector of random effects matrix dimensions

.tramME2ctm *Create a corresponding ctm model for a tramME model*

Description

Takes a tramME formula and generates the FE ctm model (model_only = TRUE)

Usage

```
.tramME2ctm(formula, mname, ...)
```

Arguments

formula	Model formula.
mname	tram(ME) model name.
...	Optional arguments passed to tram

.vc2th *Convert from vc matrix to theta vector*

Description

Convert from vc matrix to theta vector

Usage

```
.vc2th(vc, rbs)
```

Arguments

vc	Covariance matrix of random effects
rbs	Vector of random effects matrix dimensions

AaregME *Mixed-effects version of* Aareg

Description

Mixed-effects version of Aareg

Usage

```
AaregME(
  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 AaregME object.

---

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 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

```
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 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 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, fixed = TRUE, ...)
```

Arguments

object	An LmME object.
as.lm	If TRUE, return the transformed coefficients as in a lmerMod object.
fixed	If TRUE, also include the fixed parameters.
...	Optional arguments 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.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.
fixed	If TRUE, also include the fixed parameters.
...	Optional parameters (ignored).

Value

Numeric vector of parameter values.

Examples

```
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)
```

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 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

```
## 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

```
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

```
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

```
## 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	An 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 `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 *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.tramME *Duplicate a tramME object*

Description

In general, this is not necessary for the usual usage of tramME. It is only written to avoid errors stemming from the fact that some parts of the tramME object are modified in place.

Usage

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

Arguments

object	A tramME object.
...	Optional arguments (currently ignored).

---

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

Description

Create a duplicate of the tramTMB object

Usage

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

Arguments

object	A tramTMB object.
...	Optional parameters (not used).

fe_terms *Create fixed effects data and initial parameters*

Description

Create fixed effects data and initial parameters

Usage

```
fe_terms(mod)
```

Arguments

mod	a mlt model

fitmod *Fit the model.*

Description

Fit the model.

Usage

```
fitmod(object, ...)
```

Arguments

object	An object.
...	Optional parameters.

fitmod.tramME *Call the optimizer on a tramME object*

Description

Call the optimizer on a tramME object

Usage

```
## S3 method for class 'tramME'
fitmod(object, initpar = NULL, control = optim_control(), ...)
```

Arguments

object	A tramME object.
initpar	named list of initial parameter values, if NULL, it is ignored
control	list with controls for optimization
...	additional arguments to tram.

---

is.pd *Check positive definiteness*

Description

Check positive definiteness

Usage

```
is.pd(m)
```

Arguments

m	a matrix

Value

logical

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 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 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,
  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 model*

Description

Get the log-likelihood of the model

Usage

```
## S3 method for class 'tramME'
logLik(
  object,
 param = c(coef(object, with_baseline = TRUE, fixed = FALSE), varcov(object, as.theta
    = TRUE)),
  newdata = NULL,
  ...
)
```

Arguments

object	A tramME object.
param	An optional vector of parameter values in the structure (beta, theta).
newdata	An optional data.frame to calculate the out-of-sample log-likelihood.
...	Optional argument (for consistency with generic).

Value

A numeric value of the log-likelihood.

Examples

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

`lpterms` *Generic method for extracting terms of the linear predictor*

Description

Generic method for extracting terms of the linear predictor

Usage

```
lpterms(object, ...)
```

Arguments

`object`	A model object
`...`	Optional parameters

Value

The value of the baseline transfromation function at certain points.

`lpterms.tramME` *Get inidvidual terms of the linear predictor and their confidence intervals*

Description

`term` can be variable groups (baseline/interacting or shift) or names of variables.

Usage

```
## S3 method for class 'tramME'
lpterms(
  object,
  newdata = model.frame(object)[, -1L],
  term = c("baseline", "shift"),
  type = c("trafo", "distribution", "survivor", "cumhazard"),
  confidence = c("none", "interval", "band"),
  level = 0.95,
  K = 50,
  ...
)
```

Arguments

object	A tramME object.
newdata	A data.frame containing the values at which the functions are evaluated.
term	The names or identifiers of the terms we want to evaluate.
type	The scale on which the functions are evaluated.
confidence	Pointwise confidence interval or confidence band.
level	Confidence level.
K	Integer, number of points of 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.

Note

Currently it only takes the fixed effects into account when calculating intervals (either pointwise confidence intervals or confidence bands).

Examples

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

---

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

Description

Extract model frame from a tramME model

Usage

```
## S3 method for class 'tramME'
model.frame(formula, ...)
```

Arguments

formula	A tramME object
...	Optional arguments (currently ignored)

model.matrix.tramME *Model matrix for tramME mdoels*

Description

Creates the model matrix of fixed and random effects corresponding a `tramME` model from a `data.frame` of response and covariate values.

Usage

```
## S3 method for class 'tramME'
model.matrix(
  object,
  data = model.frame(object),
  type = c("fixef", "ranef"),
  with_baseline = TRUE,
  ...
)
```

Arguments

`object`	A `tramME` object.
`data`	A `data.frame` containing the variable values.
`type`	Either `"fixef"` or `"ranef"`.
`with_baseline`	Logical; indicating whether the returned fixed effects model matrix should contain the columns corresponding to the baseline transfromation. (ignored when `type = "ranef"`)
`...`	Additional arguments.

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.

offset *Generic method for* `"offset"`

Description

Generic method for `"offset"`

Usage

```
offset(object)
```

Arguments

object	An object.

---

offset.default *Default method for "offset"*

Description

Overloads the original offset function.

Usage

```
## Default S3 method:
offset(object)
```

Arguments

object	An offset to be included in a model frame

---

offset.tramME *Get the offset vector of a tramME object.*

Description

Get the offset vector of a tramME object.

Usage

```
## S3 method for class 'tramME'
offset(object)
```

Arguments

object	A tramME object.

offset<- *Generic method for* "offset<-"

Description

Generic method for "offset<-"

Usage

```
offset(object) <- value
```

Arguments

object	A model object.
value	The new vector of the offsets.

Value

An object with the same class as object, with updated offset vector.

offset<-.tramME *Set the values of the offsets of a tramME model.*

Description

This method updates the internal tramTMB object, the model.frame of the tramME object and the function call to propagate the change.

Usage

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

Arguments

object	A tramME object defined with do_update = TRUE.
value	A vector of new offset values.

Value

A tramME object with the new offset values.

Note

It works only when the tramME model is defined with do_update = TRUE.

optim_control *Set up and control optimization parameters*

Description

Set up and control optimization parameters

Usage

```
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.

optim_tramTMB *Optimize the tramTMB object*

Description

Currently only with alabama::auglag with either nlminb or optim in the case of constrained optimization and nlminb if there are no constraints.

Usage

```
optim_tramTMB(
  obj,
  par = NULL,
  method = "nlminb",
  control = list(),
  trace = FALSE,
  ntry = 5,
  scale = TRUE,
  ...
)
```

Arguments

obj	a tramTMB object
par	optional vector of initial parameter values
method	the method used by alabama::auglag
control	a list of control parameters
trace	logical, whether the trace should be printed during the optimization
ntry	number of restarts with perturbed initial values when not converged
scale	Logical, if TRUE, the fixed effects design matrices are scaled to improve convergence
...	optional arguments, currently not in use

---

parboot.tramME *Do parametric bootsrap using a tarmME model*

Description

Do parametric bootsrap using a tarmME model

Usage

```
## S3 method for class 'tramME'
parboot(
  object,
  statistic,
  nsim = 1,
  conditional = FALSE,
  seed = NULL,
  ...,
  simplify = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("profile.ncpus", 1L)
)
```

Arguments

object	A tramME object.
statistic	A function that calculates the statistic of interest.
nsim	Number of draws.
conditional	Logical, if TRUE, the resampling is conditional on the fitted vector of random effects.
seed	optional seed for the random number generator
...	Optional arguments passed to statistic.

simplify	logical or character string; should the result be simplified to a vector, matrix or higher dimensional array if possible? For sapply it must be named and not abbreviated. The default value, TRUE, returns a vector or matrix if appropriate, whereas if simplify = "array" the result may be an array of "rank" (=length(dim(.))) one higher than the result of FUN(X[[i]]).
parallel	Method for parallel computation.
ncpus	Number of cores to use for parallel computation.

Value

A list/vector/array (whichever is consistent with simplify) of bootstrapped values returned by statistic.

`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 = model.frame(x),
  ranef = NULL,
 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.
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 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

```
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
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 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

```
## S3 method for class 'tramME'
predict(
  object,
  newdata = model.frame(object),
  ranef = NULL,
  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 (either in named list format or a numeric vector) or the word "zero". See Details.
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 transfromation 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 conditonal 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.

When 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 conditonal 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
nd <- sleepstudy
nd$Reaction <- NULL
pr <- predict(fit, newdata = nd, ranef = ranef(fit), 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,
  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

```
## 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

```
## 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,
 param = c(coef(object, with_baseline = TRUE, fixed = FALSE), varcov(object, as.theta
    = TRUE)),
  newdata = NULL,
  condVar = FALSE,
  raw = FALSE,
  ...
)
```

Arguments

`object`	A `tramME` object.
`param`	An optional vector of parameter values in the structure (beta, theta).
`newdata`	An optional `data.frame` of new observations for which the new random effects values are predicted.
`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)
```

`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 (currently ignored).

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

```
## S3 method for class 'tramME'
residuals(
  object,
 param = c(coef(object, with_baseline = TRUE, fixed = FALSE), varcov(object, as.theta
    = TRUE)),
  newdata = NULL,
  ...
)
```

Arguments

object	A tramME object.
param	An optional vector of parameter values in the structure (beta, theta).
newdata	An optional data.frame.
...	Optional arguments (currently ignored).

Examples

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

---

re_terms *Create random effects data and initial paramaters*

Description

Create random effects data and initial paramaters

Usage

```
re_terms(ranef, data, negative)
```

Arguments

ranef	a list of random effects formulas from findbars
data	data.frame containing the variables of the model
negative	logical value that indicates whether the random effects have a negative sign

Value

A list containing data and parameter values to be used in the TMB model.

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.
...	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 from a tramME 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 = model.frame(object),
  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 of the nsim draws, 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". what = "ranef" returns draws from the random effects distribution, what = "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.

Details

In certain settings, the conditional CDF of the outcome cannot be inverted on some some intervals. In these cases, simulate.mlt returns censored observations.

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 *Mixed-effects version of* Survreg

Description

Mixed-effects version of 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,
  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 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.

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"),
  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)
```

---

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.
2. A formula combining the notation of **tram** and **lme4**, a tram function name, and a dataset to set up the bases.

Usage

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

Arguments

formula	formula that either describes the whole model or the random effects specification. If the model contains random effects, formula has to contain their definition in lme4-style notation.
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	tram 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
negative	an optional parameter that defines whether the random effects have a positive or a negative sign in the model when the fixed effecst 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 transfromation 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.

---

tramTMB *Create a tramTMB object*

Description

Create a tramTMB object

Usage

```
tramTMB(
  data,
  parameters,
  constraint,
  negative,
  map = list(),
  resid = FALSE,
  do_update = FALSE,
  ...
)
```

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 constarints 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
...	optional parameters passed to TMB::MakeADFun

Value

A tramTMB object.

---

tramTMB_inputs *Create inputs of the tramTMB model*

Description

The function generates random values for NA values in the list of initial parameter values.

Usage

```
tramTMB_inputs(model, ft, rt, data, param = NULL)
```

Arguments

model	a list describing the structure of the model as returned by tramME_model
ft	fixed effects terms as returned by the function fe_terms
rt	random effects terms as returned by the function re_terms
data	model frame containing offsets and weights
param	optional named list of initial parameter values

Value

A list with data matrices and initial parameter values

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 that 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 inputs.

Value

A variance-covariance matrix.

varcov.LmME — *Extract the variance-covariance matrix of the random effects of an LmME model*

Description

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

Usage

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

Arguments

object	A LmME object.
as.lm	If TRUE, the returned values correspond to the LMM parametrization.
as.theta	Logical value, if TRUE, the values are returned in their reparameterized form.
...	Optional arguments (unused).

Value

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

Examples

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

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, as.theta = FALSE, ...)
```

Arguments

object	A tramME object.
as.theta	Logical value, if TRUE, the values are returned in their reparameterized form.
...	Optional arguments (unused).

Value

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

Examples

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

---

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

Description

Generic method for "varcov<-"

Usage

```
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 infromation about the optimization.

Usage

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

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).

Value

A 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 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.

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`	1. all: all variables, 2. response: response variable, 3. grouping: grouping factors for random effects, 4. shifting: shifting variables, 5. 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" with the option as.survreg = TRUE 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 same parametrization as used by lmer.
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 passed to vcov.tramTMB

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 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

```
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
```

vcov.tramTMB *Variance-covariance matrix of the parameters*

Description

Variance-covariance matrix of the parameters

Usage

```
## 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)

weights.tramME *Get the observation weight vector of a tramME object.*

Description

Get the observation weight vector of a tramME object.

Usage

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

Arguments

object	A tramME object.
...	Optional arguments (ignored).

weights<- *Generic method for "weights<-"*

Description

Generic method for "weights<-"

Usage

```
weights(object) <- value
```

Arguments

object	A model object.
value	The new vector of the weights.

Value

An object with the same class as object, with updated weight vector.

weights<-.tramME *Set the values of the observation weights of a tramME model.*

Description

This method updates the internal tramTMB object, the model.frame of the tramME object and the function call to propagate the change.

Usage

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

Arguments

object	A tramME object defined with do_update = TRUE.
value	A vector of new weight values.

Value

A tramME object with the new weight values.

Note

It works only when the tramME model is defined with do_update = TRUE.

Index