Package 'lme4breeding'

February 16, 2026

Version 1.1.1

Date 2026-02-14

Title Breeding-Related Mixed-Effects Models

Maintainer Giovanny Covarrubias-Pazaran <cova_ruber@live.com.mx>

Description Fit relationship-based and customized mixed-effects models with complex variance-covariance structures using the 'lme4' machinery. The core computational algorithms are implemented using the
'Eigen' 'C++' library for numerical linear algebra and 'RcppEigen' 'glue'.

Depends R(>= 3.5.0), lme4 (>= 1.0), Matrix (>= 1.0), methods, crayon, enhancer

LazyLoad yes

License GPL (>= 2)

Author Giovanny Covarrubias-Pazaran [aut, cre] (ORCID: <https://orcid.org/0000-0002-7194-3837>)

Repository CRAN

Suggests rmarkdown, knitr, orthopolynom, RSpectra

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Date/Publication 2026-02-16 01:00:01 UTC

Contents

`lme4breeding-package` **L***inear* **m***ixed* **e***quations* **4 B***reeding*

Description

lme4breeding is nice wrapper of the lme4 package that enables the use of especialized plant and animal breeding models that include relationship matrices among individuals (e.g., genomic relationship matrices) and complex covariance structures between factors (e.g., factor analytic structures) accelerated by the use of the eigen decomposition of relationship matrices. It uses all the lme4 machinery for linear and non-linear models, for different response distributions opening a world of possibilities.

It took me several years to develop a package named sommer that allowed many of the desired models. In May of 2024 I realized that few lines of code (exactly 100 lines) would allow to tweak all the lme4 machinery to fit most plant and animal models popular today at a great speed enabled by the lme4 development team. I will not stop the development of the sommer package since it allows to fit certain models at a greater speed than lme4breeding and other popular packages (in the p > n problem sommer still the best). The major advantage to use lme4breeding will be when you have balanced data for multiple traits or environments (not very likely but please look at the vignettes and make your own opinion) which will make this machinery extremely fast! if you have unbalanced data you may want to stick to the use of mmec until I discover how to adapt the eigen decomposition to unbalanced data. I hope you enjoy it.

The `lmeb` function is the core function of the package which is exactly the same function than lmer or glmer but with few added arguments `relmat` and `addmat` that allow the user to provide relationship matrices and customized incidence matrices respectively. Also the argument `rotation` speeds up highly complex models. The lme4 machinery is designed to deal with a big number of records (r) since it works in the r > c problem and inverts a c x c matrix (being c the number of coefficients). There are `ranef`, `fixef`, `VarCorr` functions to obtain variance-covariance components, BLUPs, BLUEs, residuals, fitted values, variances-covariances for fixed and random effects, etc.

Functions for genetic analysis

The package provides kernels to the estimate additive (`A.matr`) relationship matrix for diploid and polyploid organisms. A good converter from letter code to numeric format is implemented in the function `atcg1234`, which supports higher ploidy levels than diploid. Additional functions for genetic analysis have been included such as build a genotypic hybrid marker matrix (`build.HMM`). If you need to use pedigree you need to convert your pedigree into a relationship matrix (use the 'getA' function from the pedigreemm package).

Functions for trial analysis

Recently, spatial modeling has been added added to lme4breeding using the two-dimensional spline `tps` function.

Keeping lme4breeding updated

The lme4breeding package is updated on CRAN every 4-months due to CRAN policies but you can find the latest source at https://github.com/covaruber/lme4breeding. This can be easily installed typing the following in the R console:

library(devtools)

install_github("covaruber/lme4breeding")

This is recommended if you reported a bug, was fixed and was immediately pushed to GitHub but not in CRAN until the next update.

Tutorials

For tutorials on how to perform different analysis with lme4breeding please look at the vignettes by typing in the terminal:

vignette("lme4breeding.qg")

vignette("lme4breeding.gxe")

Getting started

The package has been equiped with several datasets to learn how to use the lme4breeding package (and almost to learn all sort of quantitative genetic analysis):

* `DT_big` simulated dataset containing 1K individuals in 50 environments to show how to fit big models.

* `DT_btdata` dataset contains an animal (birds) model.

* `DT_cornhybrids` dataset to perform genomic prediction in hybrid single crosses

* `DT_cpdata` dataset to fit genomic prediction models within a biparental population coming from 2 highly heterozygous parents including additive, dominance and epistatic effects.

* `DT_fulldiallel` dataset with examples to fit full diallel designs.

* `DT_gryphon` data contains an example of an animal model including pedigree information.

* `DT_halfdiallel` dataset with examples to fit half diallel designs.

* `DT_h2` to calculate heritability

* `DT_ige` dataset to show how to fit indirect genetic effect models.

* `DT_legendre` simulated dataset for random regression model.

* `DT_mohring` datasets with examples to fit half diallel designs.

* `DT_polyploid` to fit genomic prediction and GWAS analysis in polyploids.

* `DT_sleepstudy` dataset to know how to translate lme4 models to lme4breeding models.

* `DT_technow` dataset to perform genomic prediction in hybrid single crosses

* `DT_wheat` dataset to do genomic prediction in single crosses in species displaying only additive effects.

Models Enabled

The machinery behind the scenes is lme4.

Bug report and contact

If you have any questions or suggestions please post it in https://stackoverflow.com or https://stats.stackexchange.com

I'll be glad to help or answer any question. I have spent a valuable amount of time developing this package. Please cite this package in your publication. Type 'citation("lme4breeding")' to know how to cite it.

Author(s)

Giovanny Covarrubias-Pazaran

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

Examples

```
data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example

ansMain <- lmeb(Yield ~ Env + (1|Name),
                        relmat = list(Name = A ),
                        data=DT)
vc <- VarCorr(ansMain); print(vc,comp=c("Variance"))
BLUP <- ranef(ansMain, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs
```

---

adjBeta *Adjusting fixed effects for intercept*

Description

This function is a very simple function to add the intercept to all fixed effects except for the first term.

Usage

```
adjBeta(x)
```

Arguments

x	a numeric vector with fixed effects extracted by the fixef function.

Value

$x a numeric vector with the intercept added to all fixed effects except for the first term which corresponds to the intercept.

Author(s)

Giovanny Covarrubias-Pazaran

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

See Also

The core function of the package lmeb

Examples

```
data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example

ansMain <- lmeb(Yield ~ Env + (1|Name),
                    relmat = list(Name = Matrix::chol(A) ),
                    data=DT)

fixef(ansMain)
adjBeta(fixef(ansMain))
```

---

balanceData *Balance a dataset*

Description

Used to balance a dataset for given slope and intercept.

Usage

```
balanceData(data, slope=NULL, intercept=NULL)
```

Arguments

data	Dataset to be balanced.
slope	A character value indicating the column in the dataset that will serve as slope in the random regression model.
intercept	A character vector indicating the column(s) in the dataset that will serve as intercepts in the random regression model. If more than one column name is provided the intercept value is assumed to be the concatenation of the variables provided.

Details

It returns the same original dataset but balanced for the slopes given certain intercept variables.

Value

It returns the new balanced dataset.

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

See Also

lmeb– the core function of the package

Examples

```
data(DT_big, package="enhancer")
DT = DT_big
M = apply(M_big,2,as.numeric)
rownames(M) <- rownames(M_big)

# unbalance the data intentionally to mimic a real scenario with missing records
nas <- sample(1:nrow(DT), round(nrow(DT)*.2))
DTu <- droplevels(DT[setdiff(1:nrow(DT), nas),])

# balance your data to be able to use the eigen decomposition
DTb <- balanceData(data=DTu, slope = "id", intercept = c("envf","repf"))
DTb$value <- imputev(x=DTb$value, by=DTb$env)

# now this dataset can be used with the argument 'rotation=TRUE' in lmeb
```

condVarRotated *Get the rotated conditional variance matrix*

Description

Retrieve the conditional variance matrix from the random effects rotated by the Cholesky factors and Eigen factors if applicable.

Usage

```
condVarRotated(object)
```

Arguments

object	model object of class lmeb

Value

Matrix

Author(s)

Giovanny Covarrubias

See Also

image, lmeb

Examples

```
data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example
head(DT)

## Compound simmetry (CS) model
ans1 <- lmeb(Yield~Env + (1|Name) + (1|Env:Name),
                data=DT)
vc <- VarCorr(ans1); print(vc,comp=c("Variance"))

condVarMat <- condVarRotated(ans1)
# image(condVarMat)
```

Dtable *Get the hypertable for predictions*

Description

For an lmeb object it creates a template for the hypertable indication.

Usage

```
Dtable(object)
```

Arguments

object	model object of class lmeb

Value

A data frame with columns; variable, group, type, include, average.

A pure "include" term means that the model matrices for that fixed or random effect is filled with 1's for the positions where column names and row names match.

An "include and average" term means that the model matrices for that fixed or random effect is filled with 1/#1's in that row.

An "average" term alone means that all rows for such fixed or random effect will be filled with 1/#levels in the effect.

If a term is not considered "include" or "average" is then totally ignored in the BLUP and SE calculation.

The default behavior when the user doesn't provide the hyperTable is to include and average any fixed effect that is not part of classify. Include any term making a perfect match with the classify argument and include and average any imperfect match with classify argument (e.g., interactions).

Author(s)

Giovanny Covarrubias

See Also

image, lmeb

Examples

```
data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example
head(DT)

## Compound simmetry (CS) model
ans1 <- lmeb(Yield~Env + (1|Name) + (1|Env:Name),
```

```
                data=DT)
vc <- VarCorr(ans1); print(vc,comp=c("Variance"))
Dtable(ans1)
```

`getMME` *Build the mixed model equations from a lmeb object*

Description

Uses the lmeb object and builds the coefficient matrix (C) and returns its inverse and the solutions to the mixed model equations.

Usage

```
getMME(object, vc=NULL, recordsToUse=NULL)
```

Arguments

`object`	an object of class lmeb.
`vc`	optional variance components to force in the mixed model equations. This this to be the outpur of the `VarCorr` function.
`recordsToUse`	a numeric vector of indices to say which records should be kept for forming the mixed model equations and get solutions. This is particularly useful when we want to predict new individuals.

Details

Uses the lmeb object and builds the coefficient matrix (C) and returns its inverse and the solutions to the mixed model equations. It is internally used by the ranef function when the user wants standard errors for the BLUPs.

Value

$Ci inverse of the coefficient matrix.

$bu solutions to the mixed model equations

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

See Also

`lmeb`– the core function of the package

Examples

```
data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example

ansMain <- lmeb(Yield ~ Env + (1|Name),
                    relmat = list(Name = A ),
                    data=DT)
mme <- getMME(ansMain)



########################################
## showing how to predict the individuals
## that didn't have records in the dataset
########################################
data(DT_cpdata, package="enhancer")
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
#### create the variance-covariance matrix
A <- A.matr(GT) # additive relationship matrix
A <- A + diag(1e-4, ncol(A), ncol(A))
#### look at the data and fit the model
head(DT)
mix1 <- lmeb(Yield~ (1|id) + (1|Rowf) + (1|Colf),
                 relmat=list(id=A),
                 control = lmerControl(
                   check.nobs.vs.nlev = "ignore",
                   check.nobs.vs.rankZ = "ignore",
                   check.nobs.vs.nRE="ignore"
                 ),
                 data=DT)

vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
# the new dataset includes more individuals
DT2 <- DT
DT2$Yield <- imputev(DT2$Yield)
mix1expanded <- lmeb(Yield~ (1|id) + (1|Rowf) + (1|Colf),
                 relmat=list(id=A),
                 control = lmerControl(
                   check.nobs.vs.nlev = "ignore",
                   check.nobs.vs.rankZ = "ignore",
                   check.nobs.vs.nRE="ignore",
                   calc.derivs=TRUE,
                   optCtrl=list(maxeval=1)
                 ),
                 data=DT2)

vc <- VarCorr(mix1expanded); print(vc,comp=c("Variance"))
# predict the individuals that didn't have records in the dataset
res <- getMME(object=mix1expanded, vc=VarCorr(mix1), recordsToUse = which(!is.na(DT$Yield)) )
```

lmeb *Fit breeding-related mixed-effects models*

Description

Fits linear or generalized linear mixed models incorporating known relationships (e.g., genetic relationship matrices) and customized random effects (e.g., overlay models). Big-data genomic models can be fitted using the eigen decomposition proposed by Lee and Van der Werf (2006).

Usage

```
lmeb(formula, data, REML = TRUE, control = list(), start = NULL,
         verbose = 1L, subset, weights, na.action, offset, contrasts = NULL,
         calc.derivs = FALSE, nIters=100,
         # lmeb special params
         family = NULL, relmat = list(), addmat = list(), trace=1L,
        dateWarning=TRUE, rotation=FALSE,  rotationK=NULL, coefOutRotation=Inf,
         returnFormula=FALSE, suppressOpt=FALSE, ...)
```

Arguments

formula	as in lmer
data	as in lmer
REML	as in lmer
control	as in lmer
start	as in lmer
verbose	as in lmer
subset	as in lmer
weights	as in lmer
na.action	as in lmer
offset	as in lmer
contrasts	as in lmer
calc.derivs	as in lmerControl. The intention of having this argument at the level of main argument is to set it as FALSE as default to speed up computation. This means that in lme4breeding if you pass calc.derivs inside the lmerControl object it won't have any effect. You need to set it using the argument in the lmeb function.

nIters — the number of iterations used by the optimizeGlmer and optimizeLmer functions. Internally, it replaces the values of the maxfun and maxeval arguments in the lmerControl optCtrl passed.This means that in lme4breeding if you pass calc.derivs inside the lmerControl object it won't have any effect. You need to set it using the argument in the lmeb function.

family — as in glmer

relmat — an optional named list of relationship matrices between levels of a given random effect (not the inverse). Internally the Cholesky decomposition of those matrices is computed to adjust the incidence matrices. The names of the elements in the list must correspond to the names of slope factors for random-effects terms in the formula argument.

addmat — an optional named list of customized incidence matrices. The names of the elements must correspond to the names of grouping factors for random-effects terms in the formula argument. Depending on the use-case the element in the list may be a single matrix or a list of matrices. Please see examples and vignettes to learn how to use it.

trace — integer scalar. If > 0 verbose output is generated during the progress of the model fit.

dateWarning — a logical value indicating if you want to be warned when a new version of lme4breeding is available on CRAN. Default is TRUE.

rotation — a logical value indicating if you want to compute the eigen decomposition of the relationship matrix to rotate the response vector y and the fixed effect matrix X in order to accelerate the computation (Lee and Vander Werf, 2016). This argument requires the dataset to be balanced and without missing data for the slope variable, intercept variables, and the response involved in the model. Also make sure that your dataset is sorted by intercepts and slopes so the rotation that is applied consequtively makes sense (e.g., `dt = dt[with(dt, order(intercept, slope)), ]`). See details to understand more about this argument and vignettes for examples.

rotationK — an integer value indicating the number of eigen vectors to compute when the rotation argument is set to TRUE. By default all eigen vectors are computed.

coefOutRotation — a numeric value denoting the inter-quantile outlier coefficient to be used in the rotation of the response when using the eigen decomposition. Experimental. Currently is set to Inf value so no outliers are removed.

returnFormula — a logical value indicating if you want to only get the results from the lFormula after the relmat and addmat arguments have been applied. This is normally just used for debugging.

suppressOpt — a logical value indicating if you want to force the model to use your customized variance components without estimating them. It skips the optimize(g)Lmer step to force the customized variance components. The variance components should be provided in the start argument. If you want to provide the variance components from a previous model you can get the initial values by running:

```
getME(mix1, 'sigma')
```

which returns a vector with theta values.

... — as in lmer

Details

All arguments to this function are the same as those to the function lmer except relmat and addmat which must be named lists. Each name must correspond to the name of a grouping factor in a random-effects term in the formula. The observed levels of that factor must be contained in the rownames and columnames of the relmat. Each relmat is the relationship matrix restricted to the observed levels and applied to the model matrix for that term. The incidence matrices in the addmat argument must match the dimensions of the final fit (pay attention to missing data in responses).

When you use the relmat argument the square root of the relationship matrix will be computed internally. Therefore, to recover the correct BLUPs for those effects you need to use the ranef function which internally multiple the obtained BLUPs by the square root of the relationship matrix one more time to recover the correct BLUPs.

The argument rotation applies the eigen decomposition proposed by Lee and Van der Werf in 2016 and makes the genetic evaluation totally sparse leading to incredible gains in speed compared to the classical approach. Internally, the eigen decomposition UDU' is carried in the relationship matrix. The U matrix is then taken to the n x n level (n being the number of records), and post-multiplied by a matrix of records presence (n x n) using the element-wise multiplication of two matrices (Schur product). By default is not activated since this may not provide the exact same variance components than other software due to numerical reasons. If you would like to obtain the exact same variance components than other software you will have to keep rotation=FALSE. This will slow down considerably the speed. Normally when the rotation is activated and variance components differ slightly with other software they will still provide extremely similar estimates at the speed of hundreds or thousands of times faster. Please consider this.

Additional useful functions are; tps for spatial kernels, rrm for reduced rank matrices, atcg1234 for conversion of genetic markers, overlay for overlay matrices, reshape for moving wide format multi-trait datasets into long format.

Normally, using the optimizer argument inside the lmerControl keep in mind that the number of iterations is called differently depending on the optimizer. For Nelder_Mead, bobyqa and nlminbwrap is called "maxfun" whereas for nloptwrap is called "maxeval". This should be passed inside a list in the optCtrl argument. For example:

```
lmeb(... , control = lmerControl( optimizer="Nelder_Mead", optCtrl=list(maxfun=100)
), ... )
```

But here, we have created the nIters argument to control the number of iterations in the optimizer. To predict values for unobserved levels you will need to impute the data and update your model with the new dataset and the initial starting values:

```
newModel <- update(oldModel, suppressOpt = TRUE, start=getME(oldModel, 'sigma'), data=imputedData)
```

It is also worth mentioning that when the user does not provide the (g)lmerControl we take the freedom to specify it to avoid some common warning or error messages such as calculating the second derivatives for big models or not allowing to have a single record per treatment level:

```
control <- (g)lmerControl( calc.derivs = FALSE, restart_edge = FALSE, check.nobs.vs.nlev
= "ignore", check.nobs.vs.rankZ = "ignore", check.nobs.vs.nRE="ignore" )
```

This is important to notice because if the user decides to specify its own controls then the user should also consider some of these arguments that are the defaults in lmeb.

Methods

Some important methods to keep in mind are:

fixef: allows you to extract fixed coefficients (BLUEs)

vcov: allows you to extract the standard errors of fixed effects

ranef: allows you to extract random coefficients (BLUPs) and their standard errors

condVarRotated: allows you to extract the predicted error variance (PEV) matrix from a model

mkMmeIndex: alllows you to extract the indices of the different fixed and random coefficients

predict: allows you to extract fixed coefficients (BLUEs)

Example Datasets

The package has been equiped with several datasets to learn how to use the lme4breeding package:

* DT_big simulated dataset containing 1K individuals in 50 environments to show how to fit big models.

* DT_btdata dataset contains an animal (birds) model.

* DT_cornhybrids dataset to perform genomic prediction in hybrid single crosses

* DT_cpdata dataset to fit genomic prediction models within a biparental population coming from 2 highly heterozygous parents including additive, dominance and epistatic effects.

* DT_fulldiallel dataset with examples to fit full diallel designs.

* DT_gryphon data contains an example of an animal model including pedigree information.

* DT_halfdiallel dataset with examples to fit half diallel designs.

* DT_h2 to calculate heritability

* DT_ige dataset to show how to fit indirect genetic effect models.

* DT_legendre simulated dataset for random regression model.

* DT_mohring datasets with examples to fit half diallel designs.

* DT_polyploid to fit genomic prediction and GWAS analysis in polyploids.

* DT_sleepstudy dataset to know how to translate lme4 models to lme4breeding models.

* DT_technow dataset to perform genomic prediction in hybrid single crosses

* DT_wheat dataset to do genomic prediction in single crosses in species displaying only additive effects.

Value

a lmeb object.

Author(s)

Giovanny Covarrubias-Pazaran

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

Lee & Van der Werf (2016). MTG2: an efficient algorithm for multivariate linear mixed model analysis based on genomic information. Bioinformatics, 32(9), 1420-1422.

Examples

```
nInds=300
nMarks=1000
nEnvs=2

#random population of nInds lines with nMarks markers
M <- matrix(rep(0,nInds*nMarks),nInds,nMarks)
for (i in 1:nInds) {
  M[i,] <- ifelse(runif(nMarks)<0.5,-1,1)
}
# compute the relationship matrix
MMT <- tcrossprod(M)
m <- sum(diag(MMT))/nrow(MMT)
MMT <- MMT/m
MMT <- MMT + diag(1e-05, ncol(MMT), ncol(MMT))
# get phenotypes
yl <- el <- il <- list()
for(iEnv in 1:nEnvs){
  #random phenotypes
  u <- rnorm(nMarks)
  g <- as.vector(crossprod(t(M),u))
  h2 <- 0.5  #heritability
  envEffect <- rnorm(1, mean = abs(rnorm(1)))
  yl[[iEnv]] <- g + rnorm(nInds,mean=0,sd=sqrt((1-h2)/h2*var(g))) + envEffect
  el[[iEnv]] <- rep(paste0("e",iEnv), length(g))
  il[[iEnv]] <-  paste0("i",1:length(g))
}
y <- unlist(yl)
env <- unlist(el)
ind <- unlist(il)
colnames(MMT) <- rownames(MMT) <- paste0("i",1:length(g))
# fit the mixed model with rotation
mix <- lmeb(y~env + (0+env|ind), relmat=list(ind=MMT), rotation = TRUE )
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
# lattice::levelplot(cov2cor(vc$ind))
# retrieve parameters
BLUP <- ranef(mix, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs
```

`lmeb-class` *Relationship-based mixed-effects model fits*

Description

A mixed-effects model fit by `lmeb`. This class extends class "`merMod`" class and includes one additional slot, `relfac`, which is a list of (left) Cholesky factors of the relationship matrices derived from "`lmeb`" objects.

Objects from the Class

Objects are created by calls to the `lmeb` function.

Slots

`relfac`**:** A list of relationship matrix factors. All other slots are inherited from class "`merMod`".

`udu`**:** A list of eigen decomposition elements. All other slots are inherited from class "`merMod`".

Extends

Class "`merMod`", directly.

Methods

fitted `signature(object = "lmeb")`: actually a non-method in that `fitted` doesn't apply to such objects because of the pre-whitening.

ranef `signature(object = "lmeb")`: back-transforms BLUPs and their conditional variances when models include the relationship between levels of random effects as returned for the object viewed as a "`merMod`)" object.

residuals `signature(object = "lmeb")`: also a non-method for the same reason as `fitted`

See Also

`lmeb`

Examples

```
showClass("lmeb")

data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example

## Compound simmetry (CS) model
ans1 <- lmeb(Yield~Env + (1|Name) + (1|Env:Name),
                  data=DT)

fitted(ans1)
```

```
residuals(ans1)
rr <- ranef(ans1)
```

---

mkMmeIndex *Table to track effects*

Description

For an lmeb object it creates a table to track the indices of each effect for a given slope and intercept.

Usage

```
mkMmeIndex(object)
```

Arguments

object	model object of class lmeb

Value

data frame with 4 columns:

1) index: indicates the position in the effects and PEV matrix (condVarRotated)

2) level: the name of the level for a given effect

3) variable: the intercept effect in the model

4) group: the slope effect in the model

Author(s)

Giovanny Covarrubias

See Also

condVarRotated, lmeb

Examples

```
data(DT_example, package="enhancer")
DT <- DT_example
A <- A_example
head(DT)

## Compound simmetry (CS) model
ans1 <- lmeb(Yield~Env + (1|Name) + (1|Env:Name),
                 data=DT)
vc <- VarCorr(ans1); print(vc,comp=c("Variance"))

head(mkMmeIndex(ans1))
```

predict.lmeb *Predict form of a LMM fitted with lmeb/lmebreed*

Description

predict method for class "lmeb".

Usage

```
## S3 method for class 'lmeb'
predict(object, hyperTable=NULL, classify, usePEV=FALSE, ...)
```

Arguments

object	a mixed model of class "lmeb" fitted with lmeb
hyperTable	a data frame with columns; variable, group, type, include, average. See the Dtable function to understand the format. A pure **include** term means that the model matrices for that fixed or random effect is filled with 1s for the positions where column names and row names match. An **include and average** term means that the model matrices for that fixed or random effect is filled with 1/#1s in that row. A pure **average** term alone means that all rows for such fixed or random effect will be filled with 1/#levels in the effect. If a term is not considered **include** or **average** is then totally ignored in the BLUP and SE calculation. The default behavior when the user doesn't provide the hyperTable is to **include and average** any fixed effect that is not part of classify. Include any term making a perfect match with the classify argument and **include and average** any imperfect match with classify argument (e.g., interactions).
classify	is a character value indicating which term we are computing the predictions for.
usePEV	is a logical value indicating whether we should use the conditional variance or the PEV for the computation of standard errors for the linear combination.
...	Further arguments to be passed.

Details

This function allows to produce predictions specifying those variables that define the margins of the hypertable to be predicted (argument classify). Predictions are obtained for each combination of values of the specified variables that is present in the data set used to fit the model. See vignettes for more details.

For predicted values the pertinent design matrices X and Z together with BLUEs (b) and BLUPs (u) are multiplied and added together.

`predicted.value` = `Db`

`standard.error` = `D Cinv Dt`

Value

`pvals`	the table of predictions according to the specified arguments.
`mapCondVar`	the map between the hypertable and the effects.
`hyperTable`	the table specifying the terms to include and terms to be averaged.
`b`	the fixed and random effects in a vector form.
`condVarMat`	the variance covariance for the predictions.
`D`	the model matrix for predictions as defined in Welham et al.(2004).
`classify`	the character value used to indicate which term we are computing the predictions for.

Author(s)

Giovanny Covarrubias-Pazaran

References

Welham, S., Cullis, B., Gogel, B., Gilmour, A., and Thompson, R. (2004). Prediction in linear mixed models. Australian and New Zealand Journal of Statistics, 46, 325 - 347.

See Also

`predict`, `lmeb`

Examples

```
data(DT_yatesoats, package="enhancer")
DT <- DT_yatesoats
m3 <- lmeb(Y ~ V + N + V:N +
                 (1|B) + (1|B:MP),
               data = DT)

############################
## predict means for nitrogen
############################
pp=predict(object=m3, classify="N")
pp$pvals
pp$hyperTable

############################
## predict means for variety
############################
pp=predict(object=m3, classify="V")
pp$pvals
```

```
pp$hyperTable
############################
## predict means for nitrogen:variety
############################
pp=predict(object=m3, classify="N:V")
pp$pvals
ht <- pp$hyperTable
ht[4,"include"]=1
ht[4,"average"]=0

pp2=predict(object=m3, classify="N:V", hyperTable=ht)
pp2$pvals
```

---

smm *sparse model matrix*

Description

smm creates a sparse model matrix for the levels of the random effect to be used with the lmeb solver.

Usage

```
smm(x)
```

Arguments

x	vector of observations for the random effect.

Value

$res a model matrix for a given factor.

Author(s)

Giovanny Covarrubias-Pazaran

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

See Also

The lmeb solver.

Examples

```
x <- as.factor(c(1:5,1:5,1:5));x
smm(x)
```

---

umat *Rotation matrix UDU' decomposition*

Description

'umat' creates the equivalent of a U matrix from the eigen decomposition of a relationship matrix of dimensions equal to the number of records equivalent to Lee and Van der Werf (2016).

Usage

```
umat(formula, relmat, data, addmat, k=NULL)
```

Arguments

formula	a formula expressing the factor to decompose.
relmat	an optional matrix of features explaining the levels of x. If not provided is assumed that the entire incidence matrix has been provided in x. But if provided, the decomposition occurs in the matrix M.
data	a dataset to be used for modeling.
addmat	additional matrices.
k	an integer value indicating the number of eigen vectors to compute when the rotation argument is set to TRUE. By default all eigen vectors are computed.

Value

$S3 A list with 3 elements:

1) The U' matrix of dimensions n x n (eigen vectors), n being the number of records.

2) The original U matrix of dimensions m x m (eigen vectors), m being the number of coefficients or levels in relmat.

3) The D matrix of dimensions m x m (eigen values), m being the number of coefficients or levels in relmat.

Author(s)

Giovanny Covarrubias-Pazaran

References

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

See Also

The core function of the package lmeb

Examples

```
data(DT_cpdata,package="enhancer")
DT <- DT_cpdata
GT <- GT_cpdata
## create the variance-covariance matrix
A <- A.matr(GT) # additive relationship matrix
A <- A + diag(1e-4, ncol(A), ncol(A))
## look at the data and fit the model
head(DT)
xx <- umat(~id, relmat = list(id=A), data=DT)
```

Index