Package 'lme4breeding'

September 22, 2025

Version 1.0.80

Date 2025-09-21

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

LazyLoad yes

LazyData 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 2025-09-22 19:00:02 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. The major advantage to use lme4breeding will be when you have balanced data for multiple traits or environments (not very likely) 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 `lmebreed` 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.mat`) 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_halfdiallel`, `DT_fulldiallel` and `DT_mohring` datasets have examples to fit half and full diallel designs.

* `DT_h2` to calculate heritability

* `DT_cornhybrids` and `DT_technow` datasets to perform genomic prediction in hybrid single crosses

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

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

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

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

* `DT_legendre` simulated dataset for random regression model.

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

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)
DT <- DT_example
A <- A_example

ansMain <- lmebreed(Yield ~ Env + (1|Name),
                        relmat = list(Name = A ),
                        data=DT)
vc <- VarCorr(ansMain); print(vc,comp=c("Variance"))
```

A.mat *Additive relationship matrix*

Description

Calculates the realized additive relationship matrix.

Usage

```
A.mat(X,min.MAF=NULL)
```

Arguments

X	Matrix ($n \times m$) of unphased genotypes for n lines and m biallelic markers, coded as {-1,0,1}. Fractional (imputed) and missing values (NA) are allowed.
min.MAF	Minimum minor allele frequency. The A matrix is not sensitive to rare alleles, so by default only monomorphic markers are removed.

Details

For vanraden method: the marker matrix is centered by subtracting column means $M = X - ms$ where ms is the coumn means. Then $A = MM'/c$, where $c = \sum_k d_k/k$, the mean value of the diagonal values of the MM' portion.

Value

If return.imputed = FALSE, the $n \times n$ additive relationship matrix is returned.

If return.imputed = TRUE, the function returns a list containing

$A the A matrix

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

lmebreed – the core function of the package

Examples

```
## random population of 200 lines with 1000 markers
X <- matrix(rep(0,200*1000),200,1000)
for (i in 1:200) {
  X[i,] <- ifelse(runif(1000)<0.5,-1,1)
}

A <- A.mat(X)


## take a look at the Genomic relationship matrix
 colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
 hv <- heatmap(A[1:15,1:15], col = colfunc(100),Colv = "Rowv")
 str(hv)
```

`add.diallel.vars` *add.diallel.vars*

Description

'add.diallel.vars' adds 4 columns to the provided diallel dataset. Specifically, the user provides a dataset with indicator variables for who is the male and female parent and the function returns the same dataset with 4 new dummy variables to allow the model fit of diallel models.

Usage

```
add.diallel.vars(df, par1="Par1", par2="Par2",sep.cross="-")
```

Arguments

`df`	a dataset with the two indicator variables for who is the male and female parent.
`par1`	the name of the column indicating who is the first parent (e.g. male).
`par2`	the name of the column indicating who is the second parent (e.g. female).
`sep.cross`	the character that should be used when creating the column for cross.id. A simple paste of the columns par1 and par2.

Value

A new data set with the following 4 new dummy variables to allow the fit of complex diallel models:

returns a 0 if is a self and a 1 for a cross.

is.cross.self returns a 0 if is a cross and a 1 is is a self.

cross.type returns a -1 for a direct cross, a 0 for a self and a 1 for a reciprocal cross.

cross.id returns a column psting the par1 and par2 columns.

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 `lmebreed` function and the `DT_mohring` example.

Examples

```
data(DT_mohring)
DT <- DT_mohring
head(DT)
DT2 <- add.diallel.vars(DT,par1="Par1", par2="Par2")
head(DT2)
## see ?DT_mohring for an example on how to use the data to fit diallel models.
```

---

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 `lmebreed`

Examples

```
data(DT_example)
DT <- DT_example
A <- A_example

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

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

---

atcg1234 *Letter to number converter*

Description

This function was designed to help users to transform their data in letter format to numeric format. Details in the format are not complex, just a dataframe with markers in columns and individuals in rows. Only markers, NO extra columns of plant names etc (names of plants can be stored as rownames). The function expects a matrix of only polymorphic markers, please make sure you clean your data before using this function. The apply function can help you identify and separate monomorphic from polymorphic markers.

Usage

```
atcg1234(data, ploidy=2, format="ATCG", maf=0, multi=TRUE,
        silent=FALSE, by.allele=FALSE, imp=TRUE, ref.alleles=NULL)
```

Arguments

data	a dataframe with markers in columns and individuals in rows. Preferable the rownames are the ID of the plants so you don't lose track of what is what.
ploidy	a numeric value indicating the ploidy level of the specie. The default is 2 which means diploid.
format	one of the two possible values allowed by the program "ATCG", which means your calls are in base-pair-letter code, i.e. "AT" in a diploid call, "AATT" tetraploid etc (just example). Therefore possible codes can be "A", "T", "C", "G", "-" (deletion), "+" (insertion). Alternatively "AB" format can be used as well. Commonly this depends from the genotyping technologies used, such as GBS or microarrays. In addition, we have enabled also the use of single-letter code used by Cornell, i.e. A=AA, C=CC, T=TT, G=GG, R=AG, Y=CT, S=CG, W=AT, K=GT, M=AC. In the case of GBS code please make sure that you set the N codes to regular NAs handled by R. The "ATCG" format also works for the bi-allelic marker codes from join map such as "lm", "ll","nn", "np","hh","hk","kk"
maf	minor allele frequency used to filter the SNP markers, the default is zero which means all markers are returned in numeric format.
multi	a TRUE/FALSE statement indicating if the function should get rid of the markers with more than 2 alleles. If FALSE, which indicates that if markers with multiple alleles are found, the alternate and reference alleles will be the first 2 alleles found. This could be risky since some alleles will be masked, i.e. AA AG AT would take only A and G as reference and alternate alleles, converting to numeric format 2 1 1, giving the same effect to AG and AT which could be a wrong assumption. The default is TRUE, removes markers with more than two alleles.
silent	a TRUE/FALSE value indicating if a progress bar should be drawn for each step of the conversion. The default is silent=FALSE, which means that we want progress bar to be drawn.
by.allele	a TRUE/FALSE value indicating if the program should transform the data in a zero/one matrix of presence/absense per allele. For example, a marker with 3 alleles A,T,C in a diploid organism will yield 6 possible configurations; AA, AT, AC, TT, TC, CC. Therefore, the program would create 3 columns for this marker indicating the presence/absence of each allele for each genotype.
imp	a TRUE/FALSE value indicating if the function should impute the missing data using the median for each marker. If FALSE, then the program will not impute.
ref.alleles	a matrix with reference alleles to be used for the conversion. The matrix should have as many columns as markers with reference alleles and with 2 rows, being the first row the alternate allele (Alt) and the second row the reference allele (Ref). Rownames should be "Alt" and "Ref" respectively. If not provided the program will decide the reference allele.

Value

$data a new dataframe of markers in numeric format with markers in columns and individuals in rows.

Author(s)

Giovanny Covarrubias-Pazaran

See Also

The core function of the package lmebreed

Examples

```
data(DT_polyploid)
genotypes <- GT_polyploid
genotypes[1:5,1:5] # look the original format


## convert markers to numeric format polyploid potatoes
numo <- atcg1234(data=genotypes, ploidy=4)
numo$M[1:5,1:5]

## convert markers to numeric format diploid rice lines
## single letter code for inbred lines from GBS pipeline
## A=AA, T=TT, C=CC, G=GG

data(DT_rice)
X <- GT_rice; X[1:5,1:5]; dim(X)
numo2 <- atcg1234(data=X, ploidy=2)
numo2$M[1:5,1:5]
```

---

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

lmebreed– the core function of the package

Examples

```
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 lmebreed
```

bbasis *Function for creating B-spline basis functions (Eilers & Marx, 2010)*

Description

Construct a B-spline basis of degree deg with ndx-1 equally-spaced internal knots (ndx segments) on range [x1,xr]. Code copied from Eilers & Marx 2010, WIR: Comp Stat 2, 637-653.

Usage

```
bbasis(x, xl, xr, ndx, deg)
```

Arguments

x	A vector. Data values for spline.
xl	A numeric value. Lower bound for data (lower external knot).
xr	A numeric value. Upper bound for data (upper external knot).
ndx	A numeric value. Number of divisions for x range (equal to number of segments = number of internal knots + 1)
deg	A numeric value. Degree of the polynomial spline.

Details

Not yet amended to coerce values that should be zero to zero!

Value

A matrix with columns holding the P-spline for each value of x. Matrix has ndx+deg columns and length(x) rows.

build.HMM *Build a hybrid marker matrix using parental genotypes from inbred individuals*

Description

Uses the 2 marker matrices from both sets of inbred or partially inbred parents and creates all possible combinations unless the user specifies which hybrid genotypes to build (custom.hyb argument). It returns the additive and dominance marker matrices (-1,0,1; homo,het,homo in additive and 0,1,0; homo,het,homo for dominance).

Usage

```
build.HMM(M1,M2, custom.hyb=NULL, return.combos.only=FALSE,
          separator=":",n.batch=1000, verbose=TRUE)
```

Arguments

`M1`	Matrix ($n \times m$) of unphased genotypes for n inbreds and m biallelic markers, coded as {-1,0,1}. Fractional (imputed) and missing values (NA) are not allowed.
`M2`	Matrix ($n \times m$) of unphased genotypes for n inbreds and m biallelic markers, coded as {-1,0,1}. Fractional (imputed) and missing values (NA) are not allowed.
`custom.hyb`	A data frame with columns 'Var1' 'Var2', 'hybrid' which specifies which hybrids should be built using the M1 and M2 matrices provided.
`return.combos.only`	A TRUE/FALSE statement inicating if the function should skip building the geotype matrix for hybrids and only return the data frame with all possible combinations to be build. In case the user wants to subset the hybrids before building the marker matrix.
`separator`	Any desired character to be used when pasting the male and female columns to assign the name to the hybrids.
`n.batch`	An optional integer value to indicate how many hybrids should be constructed at once. When the number of hybrids and number of markers is big it is better to partition the problem into multiple matrix products. By default we assume that no more than 1000 hybrids should be computed at once to use the memory more efficiently.
`verbose`	A logical value indicating if progress and warning messages should be printed in the console.

Details

It returns the marker matrix for hybrids coded as additive (-1,0,1; homo,het,homo) and dominance (0,1,0; homo,het,homo). This function is deviced for building marker matrices for hybrids coming from inbreds. If the parents are close to inbred >F5 you can try deleting the heterozygote calls (0's) and imputing those cells with the most common genotype (1 or -1). The expectation is that for mostly inbred individuals this may not change drastically the result but will make the results more interpretable. For non-inbred parents (F1 to F3) the cross of an F1 x F1 has many possibilities and is not the intention of this function to build genotypes for heterzygote x heterozygote crosses.

Value

It returns the marker matrix for hybrids coded as additive (-1,0,1; homo,het,homo) and dominance (0,1,0; homo,het,homo).

$HMM.add marker matrix for hybrids coded as additive (-1,0,1; homo,het,homo)

$HMM.dom marker matrix for hybrids coded as dominance (0,1,0; homo,het,homo)

$data.used the data frame used to build the hybrid genotypes

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.

Nishio M and Satoh M. 2014. Including Dominance Effects in the Genomic BLUP Method for Genomic Evaluation. Plos One 9(1), doi:10.1371/journal.pone.0085792

Su G, Christensen OF, Ostersen T, Henryon M, Lund MS. 2012. Estimating Additive and Non-Additive Genetic Variances and Predicting Genetic Merits Using Genome-Wide Dense Single Nucleotide Polymorphism Markers. PLoS ONE 7(9): e45293. doi:10.1371/journal.pone.0045293

See Also

lmebreed– the core function of the package

Examples

```
## use Technow data as example
data(DT_technow)
DT <- DT_technow
Md <- (Md_technow * 2) - 1
Mf <- (Mf_technow * 2) - 1

## first get all possible hybrids
res1 <- build.HMM(Md, Mf,
                  return.combos.only = TRUE)
head(res1$data.used)
use <- which(res1$data.used$hybrid %in% DT$hy)
## build the marker matrix for the first 50 hybrids
res2 <- build.HMM(Md, Mf,
                 custom.hyb = res1$data.used[use[1:10],]
                 )
res2$HMM.add[1:5,1:5]
res2$HMM.dom[1:5,1:5]


## now you can use the A.mat()
M <- res2$HMM.add
A <- A.mat(M)
```

DT_augment *DT_augment design example.*

Description

This dataset contains phenotpic data for one trait evaluated in the experimental design known as augmented design. This model allows to obtain BLUPs for genotypes that are unreplicated by dividing the field in blocks and replicating 'check genotypes' in the blocks and unreplicated genotypes randomly within the blocks. The presence of check genotypes (usually cultivars) allows the adjustment of unreplicated genotypes.

Usage

```
data("DT_augment")
```

Format

The format is: chr "DT_augment"

Source

This data was generated by a potato study.

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 functions of the package `lmebreed`

Examples

```
## AUGMENTED DESIGN EXAMPLE
data(DT_augment)
DT <- DT_augment
head(DT)

## fit the mixed model and check summary
mix1 <- lmebreed(TSW ~ Check.Gen + (1|Block) + (1|Genotype:Check),
                 control = lmerControl( # how to control n iterations
                   optCtrl = list(maxfun = 100, maxeval = 100)
                   ),
             data=DT)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance
BLUP <- ranef(mix1, condVar=TRUE)
SEs <- attr(BLUP$`Genotype:Check`, which="postVar")[,,]
```

DT_big *Big GxE Quantitative Genetic Simulation*

Description

A data frame and genetic markers for a population of 1000 individuals evaluated in 25 environments and 2 replicates per environment giving a total of 50 K records to show how to fit big GxE models with genomic information.

Usage

```
data("DT_big")
```

Format

The format is: chr "DT_big"

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 lmebreed

Examples

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



# 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))

# let's fit the GxE model (diagonal first)
DT[,"envf_repf"] = paste(DT[,"envf"],DT[,"repf"],sep="_")
Z <- with(DT, smm(envf_repf))
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
csdiagFormula <- paste0( "value ~ (1|envf) + (0+", paste(colnames(Z), collapse = "+"), "|| id)")
```

```
# fit the models
mixm2 <- lmebreed(formula=as.formula(csdiagFormula), data = DT,
                  relmat = list(id = MMT),
                  control = lmerControl( # how to control n iterations
                   calc.derivs = FALSE,
                   restart_edge = FALSE,
                   optCtrl = list(maxfun = 5000, maxeval = 5000)
                  ),
                  verbose=1L, rotation=TRUE
)

vc <- VarCorr(mixm2); print(vc,comp=c("Variance"))
ran2 = ranef(mixm2)
H0 <- ran2$id
```

DT_btdata *Blue Tit Data for a Quantitative Genetic Experiment*

Description

a data frame with 828 rows and 7 columns, with variables tarsus length (tarsus) and colour (back) measured on 828 individuals (animal). The mother of each is also recorded (dam) together with the foster nest (fosternest) in which the chicks were reared. The date on which the first egg in each nest hatched (hatchdate) is recorded together with the sex (sex) of the individuals.

Usage

```
data("DT_btdata")
```

Format

The format is: chr "DT_btdata"

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 lmebreed

Examples

```
data(DT_btdata)
DT <- DT_btdata
head(DT)



mix4 <- lmebreed(tarsus ~ sex + (1|dam) + (1|fosternest),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
             data = DT)
vc <- VarCorr(mix4); print(vc,comp=c("Variance"))
sigma(mix4)^2 # error variance
BLUP <- ranef(mix4, condVar=TRUE)
SEs <- attr(BLUP$dam, which="postVar")[,,]

### multi-trait model
traits <- c("tarsus" ,"back", "hatchdate")
for(iTrait in traits){DT[,iTrait] <- scale(DT[,iTrait])}
DTL <- reshape(DT[,c("animal", traits)], idvar = "animal", varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- merge(DTL, unique(DT[,c("animal","dam","fosternest","sex")]),
             by="animal", all.x = TRUE)
DTL <- DTL[with(DTL, order(trait)), ]
head(DTL)

system.time(
  mix <- lmebreed(value ~ (0+trait|dam),
                  # relmat = list(geno=A),
                  # rotation = TRUE,
                  data=DTL)
)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
cov2cor(vc$dam)
sigma(mix)^2 # error variance
```

DT_cornhybrids *Corn crosses and markers*

Description

This dataset contains phenotpic data for plant height and grain yield for 100 out of 400 possible hybrids originated from 40 inbred lines belonging to 2 heterotic groups, 20 lines in each, 1600 rows

exist for the 400 possible hybrids evaluated in 4 locations but only 100 crosses have phenotypic information. The purpose of this data is to show how to predict the other 300 crosses.

The data contains 3 elements. The first is the phenotypic data and the parent information for each cross evaluated in the 4 locations. 1200 rows should have missing data but the 100 crosses performed were chosen to be able to estimate the GCA and SCA effects of everything.

The second element of the data set is the phenotypic data and other relevant information for the 40.

The third element is the genomic relationship matrix for the 40 inbred lines originated from 511 SNP markers and calculated using the A.mat function.

Usage

```
data("DT_cornhybrids")
```

Format

The format is: chr "DT_cornhybrids"

Source

This data was generated by a corn study.

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 `lmebreed`

Examples

```
data(DT_cornhybrids)
DT <- DT_cornhybrids
GT <- GT_cornhybrids
A <- GT
K1 <- A[levels(DT$GCA1), levels(DT$GCA1)]; dim(K1)
K2 <- A[levels(DT$GCA2), levels(DT$GCA2)]; dim(K2)


S <- kronecker(K1, K2) ; dim(S)
rownames(S) <- colnames(S) <- levels(DT$SCA)

ans <- lmebreed(Yield ~ Location + (1| GCA1) + (1|GCA2),
                relmat = list(GCA1=K1,
                              GCA2=K2),
                # how to control n iterations
                 # control = lmerControl(
```

```
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                data=DT)
vc <- VarCorr(ans); print(vc,comp=c("Variance"))
sigma(ans)^2 # error variance
BLUP <- ranef(ans, condVar=TRUE)
SEs <- attr(BLUP$GCA1, which="postVar")[,,]
```

DT_cpdata *Genotypic and Phenotypic data for a CP population*

Description

A CP population or F1 cross is the designation for a cross between 2 highly heterozygote individuals; i.e. humans, fruit crops, bredding populations in recurrent selection.

This dataset contains phenotpic data for 363 siblings for an F1 cross. These are averages over 2 environments evaluated for 4 traits; color, yield, fruit average weight, and firmness. The columns in the CPgeno file are the markers whereas the rows are the individuals. The CPpheno data frame contains the measurements for the 363 siblings, and as mentioned before are averages over 2 environments.

Usage

```
data("DT_cpdata")
```

Format

The format is: chr "DT_cpdata"

Source

This data was simulated for fruit breeding applications.

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 lmebreed

Examples

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
## create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix
A <- A + diag(1e-4, ncol(A), ncol(A))
## look at the data and fit the model
head(DT)



mix1 <- lmebreed(Yield~ (1|id) + (1|Rowf) + (1|Colf),
                 relmat=list(id=A),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                 data=DT)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance

# run one last iteration with imputed data
# to make sure you get predictions for every level
DT2 <- DT
DT2$Yield <- imputev(DT2$Yield)
mix2 <- update(mix1, returnMod = TRUE,
               start=getME(mix1, "theta"),
               data=DT2)
predsMix2 <- ranef(mix2)
# if you don't want the imputed vector to have an effect in
# the predictions you can use the getMME function to use
# the extended model and get predictions without including the
# imputed data (I know is a bit messy)
preds <- getMME(object=mix2, # extended model
                vc=VarCorr(mix1), # variance components
                recordsToUse = which(!is.na(DT$Yield)) # records to use for MME
                )
# now you could compare between both types of predictions, the last ones are in
# theory the correct ones.
plot(preds$bu[2:364,], predsMix2$id[,1])
```

DT_example *Broad sense heritability calculation.*

Description

This dataset contains phenotpic data for 41 potato lines evaluated in 3 environments in an RCBD design. The phenotypic trait is tuber quality and we show how to obtain an estimate of DT_example for the trait.

Usage

```
data("DT_example")
```

Format

The format is: chr "DT_example"

Source

This data was generated by a potato study.

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 functions of the package lmebreed

Examples

```
data(DT_example)
DT <- DT_example
A <- A_example
head(DT)

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

BLUP <- ranef(ans1, condVar=TRUE)$Name
SEs <- attr(BLUP, which="postVar")[,,]



## Compound simmetry (CS) + Diagonal (DIAG) model
## with relationship matrix
Z <- with(DT, smm(Env))
csdiagFormula <- paste0( "Yield ~ Env + (", paste(colnames(Z), collapse = "+"), "|| Name)")
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
ansCSDG <- lmebreed(as.formula(csdiagFormula),
```

```
                    relmat = list(Name = A ),
                   # how to control n iterations
                   # control = lmerControl(
                   #  optCtrl = list(maxfun = 100, maxeval = 100)
                   #   ),
                    data=DT)
vc <- VarCorr(ansCSDG); print(vc,comp=c("Variance"))

## Compound simmetry (CS) + Diagonal (DIAG) model
## with diagonal residuals
## with relationship matrix
Z <- with(DT, smm(Env))
DT$units <- as.factor(1:nrow(DT))
csdiagFormula <- paste0( "Yield ~ Env",
                         "+(", paste(colnames(Z), collapse = "+"), "|| Name)",
                         "+(0+ ",paste(colnames(Z), collapse = "+"), "|| units)")
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
ansCSDG <- lmebreed(as.formula(csdiagFormula),
                    relmat = list(Name = A ),
                    # how to control n iterations
                    # control = lmerControl(
                    #  optCtrl = list(maxfun = 100, maxeval = 100)
                    #   ),
                    data=DT)
vc <- VarCorr(ansCSDG); print(vc,comp=c("Variance"))
sigma(ansCSDG)^2 # error variance
```

`DT_expdesigns` *Data for different experimental designs*

Description

The following data is a list containing data frames for different type of experimental designs relevant in plant breeding:

1) Augmented designs (2 examples)

2) Incomplete block designs (1 example)

3) Split plot design (2 examples)

4) Latin square designs (1 example)

5) North Carolina designs I,II and III

How to fit each is shown at the Examples section. This may help you get introduced to experimental designs relevant to plant breeding. Good luck.

Format

Different based on the design.

Source

Datasets and more detail about them can be found in the agricolae package. Here we just show the datasets and how to analyze them using the lme4breeding package.

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_expdesigns)
DT <- DT_expdesigns
names(DT)
data1 <- DT$au1
head(data1)
## response variable: "yield"
## check indicator: "entryc" ('nc' for all unreplicated, but personal.name for checks)
## blocking factor: "block"
## treatments, personal names for replicated and non-replicated: "trt"
## check no check indicator: "new"
mix1 <- lmebreed(yield~entryc + (1|block)+(1|trt),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
             data=data1)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance
BLUP <- ranef(mix1, condVar=TRUE)
SEs <- attr(BLUP$trt, which="postVar")[,,]
```

DT_fulldiallel *Full diallel data for corn hybrids*

Description

This dataset contains phenotpic data for 36 winter bean hybrids, coming from a full diallel design and evaluated for 9 traits. The column male and female origin columns are included as well.

Usage

```
data("DT_fulldiallel")
```

Format

The format is: chr "DT_fulldiallel"

Source

This data was generated by a winter bean study and originally included in the agridat package.

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 lmebreed

Examples

```
data(DT_fulldiallel)
DT <- DT_fulldiallel
head(DT)
mix <- lmebreed(stems~1 + (1|female)+(1|male),
                # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                data=DT)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
sigma(mix)^2 # error variance
BLUP <- ranef(mix, condVar=TRUE)
SEs <- attr(BLUP$female, which="postVar")[,,]
```

DT_gryphon *Gryphon data from the Journal of Animal Ecology*

Description

This is a dataset that was included in the Journal of animal ecology by Wilson et al. (2010; see references) to help users understand how to use mixed models with animal datasets with pedigree data.

The dataset contains 3 elements:

gryphon; variables indicating the animal, the mother of the animal, sex of the animal, and two quantitative traits named 'BWT' and 'TARSUS'.

pedi; dataset with 2 columns indicating the sire and the dam of the animals contained in the gryphon dataset.

A; additive relationship matrix formed using the 'getA()' function used over the pedi dataframe.

Usage

```
data("DT_gryphon")
```

Format

The format is: chr "DT_gryphon"

Source

This data comes from the Journal of Animal Ecology. Please, if using this data cite Wilson et al. publication. If using our mixed model solver please cite Covarrubias' publication.

References

Wilson AJ, et al. (2010) An ecologist's guide to the animal model. Journal of Animal Ecology 79(1): 13-26.

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 `lmebreed`

Examples

```
data(DT_gryphon)
DT <- DT_gryphon
A <- A_gryphon
P <- P_gryphon
#### look at the data
head(DT)



## fit the model with no fixed effects (intercept only)
mix1 <- lmebreed(BWT~ (1|ANIMAL),
                 relmat = list(ANIMAL=A),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                 data=DT)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
```

```
sigma(mix1)^2 # error variance
BLUP <- ranef(mix1, condVar=TRUE)
SEs <- attr(BLUP$ANIMAL, which="postVar")[,,]


### multi-trait model
traits <- c("BWT","TARSUS")
for(iTrait in traits){DT[,iTrait] <- scale(imputev(DT[,iTrait]))}
DTL <- reshape(DT[,c("ANIMAL", traits)], idvar = "ANIMAL", varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait)), ]
head(DTL)

system.time(
  mix <- lmebreed(value ~ (0+trait|ANIMAL),
                  relmat = list(ANIMAL=A),
                  # rotation = TRUE,
                  data=DTL)
)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
cov2cor(vc$ANIMAL)
sigma(mix)^2 # error variance
```

DT_h2 *Broad sense heritability calculation.*

Description

This dataset contains phenotpic data for 41 potato lines evaluated in 5 locations across 3 years in an RCBD design. The phenotypic trait is tuber quality and we show how to obtain an estimate of DT_h2 for the trait.

Usage

```
data("DT_h2")
```

Format

The format is: chr "DT_h2"

Source

This data was generated by a potato study.

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 lmebreed

Examples

```
data(DT_h2)
DT <- DT_h2
head(DT)



DT=DT[with(DT, order(Env)), ]

Z <- with(DT, smm(Env))
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
csdiagFormula <- paste0( "y ~ Env + (", paste(colnames(Z), collapse = "+"), "|| Name)")
blockFormula <- paste0( "(0+", paste(colnames(Z), collapse = "+"), "|| Block)")
form <- paste(csdiagFormula , blockFormula, sep="+")

ans1b <- lmebreed(as.formula(form),
                  # how to control n iterations
                  # control = lmerControl(
                  #  optCtrl = list(maxfun = 100, maxeval = 100)
                  #   ),
                  data=DT)
vc <- VarCorr(ans1b); print(vc,comp=c("Variance"))
sigma(ans1b)^2 # error variance
```

DT_halfdiallel *half diallel data for corn hybrids*

Description

This dataset contains phenotpic data for 21 corn hybrids, with 2 technical repetitions, coming from a half diallel design and evaluated for sugar content. The column geno indicates the hybrid and male and female origin columns are included as well.

Usage

```
data("DT_halfdiallel")
```

Format

The format is: chr "DT_halfdiallel"

Source

This data was generated by a corn study.

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 `lmebreed`

Examples

```
data("DT_halfdiallel")
DT <- DT_halfdiallel
head(DT)
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)
# overlay matrix to be added to the addmat argument
Z <- with(DT, overlay(femalef,malef) )
## inital values for incidence matrix but irrelevant
## since these will be replaced by admat argument
fema <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]
## model using overlay without relationship matrix
modh <- lmebreed(sugar ~ (1|genof) + (1|fema),
                 addmat = list(fema=Z),
             data=DT)
vc <- VarCorr(modh); print(vc,comp=c("Variance"))



## model using overlay with relationship matrix
## relationship matrix to be added to the relmat argument
A <- diag(7); colnames(A) <- rownames(A) <- 1:7;A
modh <- lmebreed(sugar ~ (1|genof) + (1|fema),
                 addmat = list(fema=Z),
                 relmat = list(fema=A),
                 # how to control n iterations
                 # control = lmerControl(
```

```
                #  optCtrl = list(maxfun = 100, maxeval = 100)
                #   ),
                data=DT)
vc <- VarCorr(modh); print(vc,comp=c("Variance"))
sigma(modh)^2 # error variance

BLUP <- ranef(modh, condVar=TRUE)
SEs <- attr(BLUP$genof, which="postVar")[,,]
```

DT_ige *Data to fit indirect genetic effects.*

Description

This dataset contains phenotpic data for 98 individuals where they are measured with the purpose of identifying the effect of the neighbour in a focal individual.

Usage

```
data("DT_ige")
```

Format

The format is: chr "DT_ige"

Source

This data was masked from a shared study.

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 lmebreed

Examples

```
data(DT_ige)
DT <- DT_ige
# Indirect genetic effects model without covariance between DGE and IGE
modIGE <- lmebreed(trait ~ block + (1|focal) + (1|neighbour),
              data = DT)
vc <- VarCorr(modIGE); print(vc,comp=c("Variance"))




## Add relationship matrices
A_ige <- A_ige + diag(1e-4, ncol(A_ige), ncol(A_ige) )
modIGE <- lmebreed(trait ~ block + (1|focal) + (1|neighbour),
                   relmat = list(focal=A_ige,
                                 neighbour=A_ige),
                   # how to control n iterations
                   # control = lmerControl(
                   #  optCtrl = list(maxfun = 100, maxeval = 100)
                   #   ),
                   data = DT)
vc <- VarCorr(modIGE); print(vc,comp=c("Variance"))

## Indirect genetic effects model with covariance between DGE and IGE using relationship matrices
## Relationship matrix
A_ige <- A_ige + diag(1e-4, ncol(A_ige), ncol(A_ige) )
## Define 2 dummy variables to make a fake covariance
## for two different random effects
DT$fn <- DT$nn <- 1
## Create the incidence matrix for the first random effect
Zf <- Matrix::sparse.model.matrix( ~ focal-1, data=DT )
colnames(Zf) <- gsub("focal","", colnames(Zf))
## Create the incidence matrix for the second random effect
Zn <- Matrix::sparse.model.matrix( ~ neighbour-1, data=DT )
colnames(Zn) <- gsub("neighbour","", colnames(Zn))
## Make inital values for incidence matrix but irrelevant
## since these will be replaced by the addmat argument
both <- (rep(colnames(Zf), nrow(DT)))[1:nrow(DT)]
## Fit the model
modIGE <- lmebreed(trait ~ block + (0+fn+nn|both),
                   addmat = list(both=list(Zf,Zn)),
                   relmat = list(both=A_ige),
                   data = DT)
vc <- VarCorr(modIGE); print(vc,comp=c("Variance"))
sigma(modIGE)^2 # error variance

blups <- ranef(modIGE)
pairs(blups$both)
cov2cor(vc$both)
```

DT_legendre *Simulated data for random regression*

Description

A data frame with 4 columns; SUBJECT, X, Xf and Y to show how to use the Legendre polynomials in the lmebreed function using a numeric variable X and a response variable Y.

Usage

```
data("DT_legendre")
```

Format

The format is: chr "DT_legendre"

Source

This data was simulated for fruit breeding applications.

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 lmebreed

Examples

```
data(DT_legendre)
DT <- DT_legendre
head(DT)



library(orthopolynom)
Z <- with(DT, smm(leg(X,1)) )
## diagonal random regression
form <- paste0( "Y ~ Xf + (0+", paste(colnames(Z), collapse = "+"), "|| SUBJECT)")
## unstructured random regression
form <- paste0( "Y ~ Xf + (0+", paste(colnames(Z), collapse = "+"), "| SUBJECT)")
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
mRR2b<-lmebreed(as.formula(form),
```

```
                # how to control n iterations
                # control = lmerControl(
                #  optCtrl = list(maxfun = 100, maxeval = 100)
                #   ),
           , data=DT)
vc <- VarCorr(mRR2b); print(vc,comp=c("Variance"))
sigma(mRR2b)^2 # error variance
```

DT_mohring *Full diallel data for corn hybrids*

Description

This dataset contains phenotpic data for 36 winter bean hybrids, coming from a full diallel design and evaluated for 9 traits. The column male and female origin columns are included as well.

Usage

```
data("DT_mohring")
```

Format

The format is: chr "DT_mohring"

Source

This data was generated by a winter bean study and originally included in the agridat package.

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 lmebreed

Examples

```
data(DT_mohring)
DT <- DT_mohring
head(DT)
DT2 <- add.diallel.vars(DT,par1="Par1", par2="Par2")
head(DT2)
## is.cross denotes a hybrid (1)
## is.self denotes an inbred (1)
## cross.type denotes one way (-1, e.g. AxB) and reciprocal (1, e.g., BxA) and no cross (0)
## cross.id denotes the name of the cross (same name for direct & reciprocal)

## GRIFFING MODEL 2 with reciprocal effects ############################
## overlay matrix to be added to the addmat argument
Z <- with(DT, overlay(Par1, Par2) )
fema <- (rep(colnames(Z), nrow(Z)))[1:nrow(Z)]
mod1h <- lmebreed(Ftime ~ 1 + (1|Block) +
                    ## GCA male & female overlayed
                    (1|fema) +
                    ## SCA effects (includes cross and selfs)
                    (cross.type||cross.id),
                  addmat=list(fema=Z),
                  data=DT2 )
vc <- VarCorr(mod1h); print(vc,comp=c("Variance"))

##                                    VarComp VarCompSE   Zratio
## Block.Ftime-Ftime                  0.00000   9.32181 0.000000
## overlay(Par1, Par2).Ftime-Ftime 1276.73089 750.17269 1.701916
## cross.id.Ftime-Ftime            1110.99090 330.16921 3.364914
## cross.id:cross.type.Ftime-Ftime   66.02295  49.26876 1.340057
## units.Ftime-Ftime                418.47949  74.56442 5.612321
##



## GRIFFING MODEL 2, no reciprocal effects ##############################
mod1h <- lmebreed(Ftime ~ Block + is.cross +
                    ## GCA male & female overlayed main and rr
                    ## GCA effect (calculated only in hybrids; remaining variance)
                    (is.cross||fema) +
                    ## SCA effect (calculated in hybrids only)
                    (0+is.cross||cross.id),
                  addmat=list(fema=Z),
                  data=DT2 )
vc <- VarCorr(mod1h); print(vc,comp=c("Variance"))

##                                           VarComp  VarCompSE   Zratio
## overlay(Par1, Par2).Ftime-Ftime          2304.1781 1261.63193 1.826347
## overlay(Par1, Par2):is.cross.Ftime-Ftime  613.6040  402.74347 1.523560
## cross.id:is.cross.Ftime-Ftime             340.7030  148.56225 2.293335
## units.Ftime-Ftime                         501.6275   74.36075 6.745864
##
```

```
## GRIFFING MODEL 3, no reciprocal effects ################################
mod1h <- lmebreed(Ftime ~ Block + is.cross +
                    ## GCAC (only for hybrids)
                    (0+is.cross||fema) +
                    ## male GCA (only for inbreds)
                    (0+is.self||Par1) +
                    ## SCA (for hybrids only)
                    (0+is.cross||cross.id),
                  addmat=list(fema=Z),
                  data=DT2 )
vc <- VarCorr(mod1h); print(vc,comp=c("Variance"))

##                                          VarComp  VarCompSE   Zratio
## overlay(Par1, Par2):is.cross.Ftime-Ftime  927.7895  537.91218 1.724797
## Par1:is.self.Ftime-Ftime                 9960.9247 5456.58188 1.825488
## cross.id:is.cross.Ftime-Ftime             341.4567  148.53667 2.298804
## units.Ftime-Ftime                         498.5974   73.92066 6.745035
##
## GRIFFING MODEL 2, with reciprocal effects ##############################
## In Mohring: mixed model 3 reduced
mod1h <- lmebreed(Ftime ~ Block + is.cross +
                    ## GCAC (for hybrids only)
                    (0+is.cross||fema) +
                    ## male GCA (for selfs only)
                    (0+is.self||Par1) +
                    ## SCA (for hybrids only)
                    (0+is.cross||cross.id)+
                    ## SCAR reciprocal effects (remaning SCA variance)
                    (0+cross.type||cross.id),
                  addmat=list(fema=Z),
                  data=DT2 )
vc <- VarCorr(mod1h); print(vc,comp=c("Variance"))

##                                           VarComp  VarCompSE   Zratio
## overlay(Par1, Par2):is.cross.Ftime-Ftime   927.78742  537.89981 1.724833
## Par1:is.self.Ftime-Ftime                 10001.78854 5456.47578 1.833013
## cross.id:is.cross.Ftime-Ftime              361.89712  148.54264 2.436318
## cross.id:cross.type.Ftime-Ftime             66.43695   49.24492 1.349113
## units.Ftime-Ftime                          416.82960   74.27202 5.612203
##
## GRIFFING MODEL 3, with RGCA + RSCA #####################################
## In Mohring: mixed model 3
mod1h <- lmebreed(Ftime ~ Block + is.cross +
                    ## GCAC (for hybrids only)
                    (0+is.cross||fema) +
                    ## RGCA: exclude selfs (to identify reciprocal GCA effects)
                    (0+cross.type||fema) +
                    ## male GCA (for selfs only)
                    (0+is.self||Par1) +
                    ## SCA (for hybrids only)
                    (0+is.cross||cross.id)+
                    ## SCAR reciprocal effects (remaning SCA variance)
                    (0+cross.type||cross.id),
```

```
                  addmat=list(fema=Z),
                  data=DT2 )
vc <- VarCorr(mod1h); print(vc,comp=c("Variance"))
```

```
##                                           VarComp  VarCompSE    Zratio
## overlay(Par1, Par2):is.cross.Ftime-Ftime   927.7843  537.88164 1.7248857
## Par1:is.self.Ftime-Ftime                 10001.7570 5456.30125 1.8330654
## cross.id:is.cross.Ftime-Ftime              361.8958  148.53670 2.4364068
## overlay(Par1, Par2):cross.type.Ftime-Ftime  17.9799   19.92428 0.9024114
## cross.id:cross.type.Ftime-Ftime             30.9519   46.43908 0.6665054
## units.Ftime-Ftime                          416.09922  447.2101 0.93043333
```

`DT_polyploid` *Genotypic and Phenotypic data for a potato polyploid population*

Description

This dataset contains phenotpic data for 18 traits measured in 187 individuals from a potato diversity panel. In addition contains genotypic data for 221 individuals genotyped with 3522 SNP markers. Please if using this data for your own research make sure you cite Rosyara's (2015) publication (see References).

Usage

```
data("DT_polyploid")
```

Format

The format is: chr "DT_polyploid"

Source

This data was extracted from Rosyara (2016).

References

If using this data for your own research please cite:

Rosyara Umesh R., Walter S. De Jong, David S. Douches, Jeffrey B. Endelman. Software for genome-wide association studies in autopolyploids and its application to potato. The Plant Genome 2015.

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 functions of the package lmebreed

Examples

```
data(DT_polyploid)
DT <- DT_polyploid
GT <- GT_polyploid
MP <- MP_polyploid
## convert markers to numeric format
numo <- atcg1234(data=GT[,1:100], ploidy=4);
numo$M[1:5,1:5];
numo$ref.allele[,1:5]



## plants with both genotypes and phenotypes
common <- intersect(DT$Name,rownames(numo$M))

## get the markers and phenotypes for such inds
marks <- numo$M[common,]; marks[1:5,1:5]
DT2 <- DT[match(common,DT$Name),];
DT2 <- as.data.frame(DT2)
DT2[1:5,]

## Additive relationship matrix, specify ploidy
A <- A.mat(marks)

## run as mixed model
A <- A + diag(1e-4,ncol(A),ncol(A))
ans <- lmebreed(tuber_shape~ (1|Name),
                relmat = list(Name=A),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                data=DT2)
vc <- VarCorr(ans); print(vc,comp=c("Variance"))
sigma(ans)^2 # error variance
```

DT_rice *Rice lines dataset*

Description

Information from a collection of 413 rice lines. The DT_rice data set is from Rice Diversity Org. Program. The lines are genotyped with 36,901 SNP markers and phenotyped for more than 30

traits. This data set was included in the package to play with it. If using it for your research make sure you cite the original publication from Zhao et al.(2011).

Usage

```
data(DT_rice)
```

Format

RicePheno contains the phenotypes RiceGeno contains genotypes letter code RiceGenoN contains the genotypes in numerical code using atcg1234 converter function

Source

Rice Diversity Organization http://www.ricediversity.org/data/index.cfm.

References

Keyan Zhao, Chih-Wei Tung, Georgia C. Eizenga, Mark H. Wright, M. Liakat Ali, Adam H. Price, Gareth J. Norton, M. Rafiqul Islam, Andy Reynolds, Jason Mezey, Anna M. McClung, Carlos D. Bustamante & Susan R. McCouch (2011). Genome-wide association mapping reveals a rich genetic architecture of complex traits in Oryza sativa. Nat Comm 2:467 DOI: 10.1038/ncomms1467, Published Online 13 Sep 2011.

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 `lmebreed`

Examples

```
data(DT_rice)
DT <- DT_rice
GT <- GT_rice
GTn <- GTn_rice
head(DT)
M <- atcg1234(GT)



### univariate model
A <- A.mat(M$M)
A <- A + diag(1e-4, ncol(A), ncol(A))
mix <- lmebreed(Protein.content ~ (1|geno),
                relmat = list(geno=A),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
```

```
                #   ),
               data=DT)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
sigma(mix)^2 # error variance

### multi-trait model
traits <- c("Flowering.time.at.Arkansas" ,"Seed.volume", "Protein.content")
for(iTrait in traits){DT[,iTrait] <- scale(DT[,iTrait])}
DTL <- reshape(DT[,c("geno", traits)], idvar = "geno", varying = traits,
        v.names = "value", direction = "long",
        timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait)), ]
head(DTL)

system.time(
  mix <- lmebreed(value ~ (0+trait|geno),
                  relmat = list(geno=A),
                  # how to control n iterations
                  # control = lmerControl(
                  #  optCtrl = list(maxfun = 100, maxeval = 100)
                  #   ),
                  rotation = TRUE,
                  data=DTL)
)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
vc$geno
sigma(mix)^2 # error variance
```

`DT_sleepstudy` *Reaction times in a sleep deprivation study*

Description

The average reaction time per day for subjects in a sleep deprivation study. On day 0 the subjects had their normal amount of sleep. Starting that night they were restricted to 3 hours of sleep per night. The observations represent the average reaction time on a series of tests given each day to each subject. Data from sleepstudy to see how lme4 models can be translated in sommer.

Usage

```
data("DT_sleepstudy")
```

Format

The format is: chr "DT_sleepstudy"

Source

These data are from the study described in Belenky et al. (2003), for the sleep deprived group and for the first 10 days of the study, up to the recovery period.

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.

Gregory Belenky et al. (2003) Patterns of performance degradation and restoration during sleep restrictions and subsequent recovery: a sleep dose-response study. Journal of Sleep Research 12, 1-12.

See Also

The core function of the package lmebreed

Examples

```
data(DT_sleepstudy)
DT <- DT_sleepstudy
head(DT)
##################################
fm1 <- lmebreed(Reaction ~ Days + (1 | Subject),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                data=DT)
vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
sigma(fm1)^2 # error variance



##################################
fm1 <- lmebreed(Reaction ~ Days + (Days || Subject), data=DT)
vc <- VarCorr(fm1); print(vc,comp=c("Variance"))

##################################
fm1 <- lmebreed(Reaction ~ Days + (Days | Subject), data=DT)
vc <- VarCorr(fm1); print(vc,comp=c("Variance"))

##################################
fm1 <- lmebreed(Reaction ~ Days + (0 + Days | Subject), data=DT)
vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
```

DT_technow *Genotypic and Phenotypic data from single cross hybrids (Technow et al.,2014)*

Description

This dataset contains phenotpic data for 2 traits measured in 1254 single cross hybrids coming from the cross of Flint x Dent heterotic groups. In addition contains the genotipic data (35,478 markers) for each of the 123 Dent lines and 86 Flint lines. The purpose of this data is to demosntrate the prediction of unrealized crosses (9324 unrealized crosses, 1254 evaluated, total 10578 single crosses). We have added the additive relationship matrix (A) but can be easily obtained using the A.mat function on the marker data. Please if using this data for your own research cite Technow et al. (2014) publication (see References).

Usage

```
data("DT_technow")
```

Format

The format is: chr "DT_technow"

Source

This data was extracted from Technow et al. (2014).

References

If using this data for your own research please cite:

Technow et al. 2014. Genome properties and prospects of genomic predictions of hybrid performance in a Breeding program of maize. Genetics 197:1343-1355.

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 `lmebreed`

Examples

```
data(DT_technow)
DT <- DT_technow
Md <- (Md_technow*2) - 1
Mf <- (Mf_technow*2) - 1
Ad <- A.mat(Md)
```

```
Af <- A.mat(Mf)
Ad <- Ad + diag(1e-4, ncol(Ad), ncol(Ad))
Af <- Af + diag(1e-4, ncol(Af), ncol(Af))
## simple model
ans2 <- lmebreed(GY ~ (1|dent) + (1|flint),
                 data=DT)
vc <- VarCorr(ans2); print(vc,comp=c("Variance"))
BLUP <- ranef(ans2, condVar=TRUE)
SEs <- attr(BLUP$dent, which="postVar")[,,]



### with relationship matrices
ans2 <- lmebreed(GY ~ (1|dent) + (1|flint),
             relmat = list(dent=Ad,
                           flint=Af),
             # how to control n iterations
             # control = lmerControl(
             #  optCtrl = list(maxfun = 100, maxeval = 100)
             #   ),
             data=DT)
vc <- VarCorr(ans2); print(vc,comp=c("Variance"))

### overlayed model
M <- rbind(Md,Mf)
A <- A.mat(M)
A <- A + diag(1e-4,ncol(A), ncol(A))
Z <- with(DT, overlay(dent,flint) )
Z = Z[which(!is.na(DT$GY)),]
# inital values for incidence matrix but irrelevant
# since these will be replaced by admat argument
fema <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]
#### model using overlay without relationship matrix
ans2 <- lmebreed(GY ~ (1|fema),
                 addmat = list(fema=Z),
                 relmat = list(fema=A),
                 data=DT)
vc <- VarCorr(ans2); print(vc,comp=c("Variance"))
sigma(ans2)^2 # error variance
```

DT_wheat *wheat lines dataset*

Description

Information from a collection of 599 historical CIMMYT wheat lines. The wheat data set is from CIMMYT's Global Wheat Program. Historically, this program has conducted numerous international trials across a wide variety of wheat-producing environments. The environments represented

in these trials were grouped into four basic target sets of environments comprising four main agro-climatic regions previously defined and widely used by CIMMYT's Global Wheat Breeding Program. The phenotypic trait considered here was the average grain yield (GY) of the 599 wheat lines evaluated in each of these four mega-environments.

A pedigree tracing back many generations was available, and the Browse application of the International Crop Information System (ICIS), as described in (McLaren *et al.* 2000, 2005) was used for deriving the relationship matrix A among the 599 lines; it accounts for selection and inbreeding.

Wheat lines were recently genotyped using 1447 Diversity Array Technology (DArT) generated by Triticarte Pty. Ltd. (Canberra, Australia; http://www.triticarte.com.au). The DArT markers may take on two values, denoted by their presence or absence. Markers with a minor allele frequency lower than 0.05 were removed, and missing genotypes were imputed with samples from the marginal distribution of marker genotypes, that is, $x_{ij} = Bernoulli(\hat{p}_j)$, where $\hat{p}_j$ is the estimated allele frequency computed from the non-missing genotypes. The number of DArT MMs after edition was 1279.

Usage

```
data(DT_wheat)
```

Format

Matrix Y contains the average grain yield, column 1: Grain yield for environment 1 and so on.

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

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.

McLaren, C. G., L. Ramos, C. Lopez, and W. Eusebio. 2000. "Applications of the geneaology manegment system." In *International Crop Information System. Technical Development Manual, version VI*, edited by McLaren, C. G., J.W. White and P.N. Fox. pp. 5.8-5.13. CIMMyT, Mexico: CIMMyT and IRRI.

McLaren, C. G., R. Bruskiewich, A.M. Portugal, and A.B. Cosico. 2005. The International Rice Information System. A platform for meta-analysis of rice crop data. *Plant Physiology* **139**: 637-642.

See Also

The core function of the package `lmebreed`

Examples

```
data(DT_wheat)
DT <- DT_wheat
GT <- GT_wheat
DT <- data.frame(pheno=as.vector(DT),
                 env=as.factor(paste0("e", sort(rep(1:4,nrow(DT))))),
                 id=rep(rownames(DT),4))

rownames(GT) <-  rownames(DT_wheat)
K <- A.mat(GT) # additive relationship matrix
K[1:4,1:4]
##
head(DT)



#### main effect model
system.time(
mix0 <- lmebreed(pheno ~ (1|id),
                 relmat = list(id=K),
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                 data=DT)
                 )
vc <- VarCorr(mix0); print(vc,comp=c("Variance"))
sigma(mix0)^2 # error variance
BLUP <- ranef(mix0, condVar=TRUE)
SEs <- attr(BLUP$id, which="postVar")[,,]

#### unstructured model
Z <- with(DT, smm(env))
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
system.time(
mix1 <- lmebreed(pheno ~ (0 + e1 + e2 + e3 + e4 | id),
                 relmat = list(id=K),
                 rotation = TRUE,
                 # how to control n iterations
                 # control = lmerControl(
                 #  optCtrl = list(maxfun = 100, maxeval = 100)
                 #   ),
                 data=DT)
)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance
```

DT_yatesoats *Yield of oats in a split-block experiment*

Description

The yield of oats from a split-plot field trial using three varieties and four levels of manurial treatment. The experiment was laid out in 6 blocks of 3 main plots, each split into 4 sub-plots. The varieties were applied to the main plots and the manurial (nitrogen) treatments to the sub-plots.

Format

`block` block factor with 6 levels

`nitro` nitrogen treatment in hundredweight per acre

`Variety` genotype factor, 3 levels

`yield` yield in 1/4 lbs per sub-plot, each 1/80 acre.

`row` row location

`column` column location

Source

Yates, Frank (1935) Complex experiments, *Journal of the Royal Statistical Society Suppl.* 2, 181–247.

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

Examples

```
data(DT_yatesoats)
DT <- DT_yatesoats
head(DT)
m3 <- lmebreed(Y ~ V + N + V:N +
             (1|B) + (1|B:MP),
             # how to control n iterations
             # control = lmerControl(
             #  optCtrl = list(maxfun = 100, maxeval = 100)
             #   ),
           data = DT)
vc <- VarCorr(m3); print(vc,comp=c("Variance"))
sigma(m3)^2 # error variance
```

`fillData` *Filling gaps for a dataset to balance*

Description

`fillData` creates a balanced dataset in the long format to be used with the `lmebreed` solver for multi-trait models.

Usage

```
fillData(data, toBalanceSplit=NULL, toBalanceFill=NULL)
```

Arguments

`data`	a data frame with traits in wide format.
`toBalanceSplit`	variable to split the dataset for balancing.
`toBalanceFill`	variable of the factor to balance across the levels of the toBalanceSplit variable.

Value

$res a data frame with traits in long format.

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 function for the `lmebreed` solver.

Examples

```
DT <- DT_example
A <- A_example
DT2 <- fillData(DT , toBalanceSplit="Env", toBalanceFill="Name" )
head(DT2)



## Compound simmetry (CS) model
ans2 <- lmebreed(Yield~Env+ (0+Env|Name),
```

```
                relmat = list(Name=A),
                rotation = TRUE,
                data=DT2)
vc <- VarCorr(ans2); print(vc,comp=c("Variance"))

BLUP <- ranef(ans2, condVar=TRUE)$Name
SEs <- attr(BLUP, which="postVar")[,,]
```

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

Description

Uses the lmebreed 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 lmebreed.
`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 lmebreed 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

lmebreed– the core function of the package

Examples

```
data(DT_example)
DT <- DT_example
A <- A_example

ansMain <- lmebreed(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)
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
#### create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix
A <- A + diag(1e-4, ncol(A), ncol(A))
#### look at the data and fit the model
head(DT)
mix1 <- lmebreed(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 <- lmebreed(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)) )
```

`I.mat` *Identity relationship matrix*

Description

Shortcut to get an identity relationship matrix.

Usage

```
I.mat(x)
```

Arguments

`x`	Vector of values to identify the unique values and form an iende.

Details

Nothing but a shortcut to avoid few lines of code.

Value

If return.imputed = FALSE, the $n \times n$ identity matrix is returned.

$I the I matrix

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

`lmebreed` – the core function of the package

Examples

```
## random population of 200 lines with 1000 markers
```

`imputev` *Imputing a numeric or character vector*

Description

This function is a very simple function to impute a numeric or character vector with the mean or median value of the vector.

Usage

```
imputev(x, method="median",by=NULL)
```

Arguments

`x`	a numeric or character vector.
`method`	the method to choose between mean or median.
`by`	the level at which imputation will be done.

Value

$x a numeric or character vector imputed with the method selected.

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 `lmebreed`

Examples

```
set.seed(1253)
x <- rnorm(100)
x[sample(1:100,10)] <- NA
imputev(x)
```

leg *Legendre polynomial matrix*

Description

Legendre polynomials of order 'n' are created given a vector 'x' and normalized to lay between values u and v.

Usage

```
leg(x,n=1,u=-1,v=1, intercept=TRUE, intercept1=FALSE)
```

Arguments

x	numeric vector to be used for the polynomial.
n	order of the Legendre polynomials.
u	lower bound for the polynomial.
v	upper bound for the polynomial.
intercept	a TRUE/FALSE value indicating if the intercept should be included.
intercept1	a TRUE/FALSE value indicating if the intercept should have value 1 (is multiplied by sqrt(2)).

Value

$S3 an Legendre polynomial matrix of order n.

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 lmebreed

Examples

```
x <- sort(rep(1:3,100))



library(orthopolynom)
 leg(x, n=1)
 leg(x, n=2)

## see dataset data(DT_legendre) for a random regression modeling example
```

`lmebreed` *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

```
lmebreed(formula, data, REML = TRUE, control = list(), start = NULL,
         verbose = 0L, subset, weights, na.action, offset, contrasts = NULL,
         # lmebreed special params
         family = NULL, relmat = list(), addmat = list(), trace=1L,
         dateWarning=TRUE, rotation=FALSE,  rotationK=NULL, coefOutRotation=8,
         returnParams=FALSE, returnMod=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`
`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. 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. 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 to avoid overshooting.

`returnParams` 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.

`returnMod` a logical value indicating if you want to force the model to use your customized variance components without estimating them. 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, fillData for balancing datasets for two variables.

When 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:

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

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, returnMod = TRUE, start=getME(oldModel, 'sigma'), data=imputedData)
```

Example Datasets

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

* DT_halfdiallel, DT_fulldiallel and DT_mohring datasets have examples to fit half and full diallel designs.

* DT_h2 to calculate heritability

* DT_cornhybrids and DT_technow datasets to perform genomic prediction in hybrid single crosses

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

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

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

* DT_btdata dataset contains an animal (birds) model.

* DT_legendre simulated dataset for random regression model.

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

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

Value

a lmebreed object.

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.

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

```
data(DT_example)
DT <- DT_example
A <- A_example

ansMain <- lmebreed(Yield ~ Env + (1|Name),
                        relmat = list(Name = A ),
                        data=DT)
vc <- VarCorr(ansMain); print(vc,comp=c("Variance"))
sigma(ansMain)^2 # error variance

BLUP <- ranef(ansMain, condVar=TRUE)$Name
SEs <- attr(BLUP, which="postVar")[,,]
```

---

lmebreed-class *Relationship-based mixed-effects model fits*

Description

A mixed-effects model fit by lmebreed. 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 "lmebreed" objects.

Objects from the Class

Objects are created by calls to the lmebreed function.

Slots

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

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

Extends

Class "merMod", directly.

Methods

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

ranef signature(object = "lmebreed"): incorporates the relationship into the random effects as returned for the object viewed as a "merMod)" object.

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

See Also

lmebreed

Examples

```
showClass("lmebreed")
```

---

overlay *Overlay Matrix*

Description

'overlay' adds r times the design matrix for model term t to the existing design matrix. Specifically, if the model up to this point has p effects and t has a effects, the a columns of the design matrix for t are multiplied by the scalar r (default value 1.0). This can be used to force a correlation of 1 between two terms as in a diallel analysis.

Usage

```
overlay(..., rlist=NULL, prefix=NULL, sparse=FALSE)
```

Arguments

...	as many vectors as desired to overlay.
rlist	a list of scalar values indicating the times that each incidence matrix overlayed should be multiplied by. By default r=1.
prefix	a character name to be added before the column names of the final overlay matrix. This may be useful if you have entries with names starting with numbers which programs such as asreml doesn't like, or for posterior extraction of parameters, that way 'grep'ing is easier.
sparse	a TRUE/FALSE statement specifying if the matrices should be built as sparse or regular matrices.

Value

$S3 an incidence matrix with as many columns levels in the vectors provided to build the incidence matrix.

Author(s)

Giovanny Covarrubias-Pazaran

References

Fikret Isik. 2009. Analysis of Diallel Mating Designs. North Carolina State University, Raleigh, USA.

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 `lmebreed` and a function for creating dummy variables for diallel models named `add.diallel.vars`.

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data("DT_halfdiallel")
DT <- DT_halfdiallel
head(DT)
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)

with(DT, overlay(femalef,malef, sparse = TRUE))
with(DT, overlay(femalef,malef, sparse = FALSE))
```

---

redmm *Reduced Model Matrix*

Description

'redmm' reduces a model matrix by performing a singular value decomposition or Cholesky on an incidence matrix.

Usage

```
redmm(x, M = NULL, Lam=NULL, nPC=50, cholD=FALSE, returnLam=FALSE)
```

Arguments

x	as vector with values to form a model matrix or the complete incidence matrix itself for an effect of interest.
M	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.
Lam	a matrix of loadings in case is already available to avoid recomputing it.
nPC	number of principal components to keep from the matrix of loadings to form the model matrix.
cholD	should a Cholesky or a Singular value decomposition should be used. The default is the SVD.
returnLam	should the function return the loading matrix in addition to the incidence matrix. Default is FALSE.

Value

$S3 A list with 3 elements:

1) The model matrix to be used in the mixed modeling.
2) The reduced matrix of loadings (nPC columns).
3) The full matrix of loadings.

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 lmebreed

Examples

```
data(DT_technow)
DT <- DT_technow
Md <- Md_technow

M <- tcrossprod(Md)
```

```
Z = with(DT, redmm(x=dent, M=M, nPC=10))
custom <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]
ans <- lmebreed(GY ~ (1|custom),
                addmat = list(custom=Z),
            data=DT)
vc <- VarCorr(ans); print(vc,comp=c("Variance"))

xx <- with(DT, redmm(x=dent, M=M, nPC=10, returnLam = TRUE))
u = tcrossprod(xx$Lam, t(as.matrix( ranef(ans)[[1]] ) ))
```

`rrm` *reduced rank matrix*

Description

`rrm` creates a reduced rank factor analytic matrix by selecting the n vectors of the L matrix of the Cholesky decomposition or the U vectors of the SVD decomposition (loadings or latent covariates) to create a new incidence matrix of latent covariates that can be used with the `lmebreed` solver to fit random regressions on the latent covariates.

Usage

```
rrm(x=NULL, H=NULL, nPC=2, returnGamma=FALSE, cholD=TRUE)
```

Arguments

`x`	vector of the dataset containing the variable to be used to form the incidence matrix.
`H`	two-way table of identifiers (rows; e.g., genotypes) by features (columns; e.g., environments) effects. Row names and column names are required. No missing data is allowed.
`nPC`	number of principal components to keep from the loadings matrix.
`returnGamma`	a TRUE/FALSE argument specifying if the function should return the matrix of loadings used to build the incidence matrix for the model. The default is FALSE so it returns only the incidence matrix.
`cholD`	a TRUE/FALSE argument specifying if the Cholesky decomposition should be calculated or the singular value decomposition should be used instead.

Details

This implementation of a version of the reduced rank factor analytic models uses the so-called principal component (PC) models (Meyer, 2009) which assumes specific effects (psi) are equal to 0. The model is as follows:

$y = Xb + Zu + e$

where the variance of u ~ MVN(0, Sigma)

Sigma = (Gamma_t Gamma) + Psi

Extended factor analytic model:

y = Xb + Z(I Gamma)c + Zs + e = Xb + Z*c + Zs + e

where `y` is the response variable, `X` and `Z` are incidence matrices for fixed and random effects respectively, `I` is a diagonal matrix, `Gamma` are the factor loadings for `c` common factor scores, and `s` are the specific effects, `e` is the vector of residuals.

Reduced rank model:

y = Xb + Z(I Gamma)c + e = Xb + Z*c + e

which is equal to the one above but assumes specific effects = 0.

The algorithm in rrm is the following:

1) uses a wide-format table of timevar (m columns) by idvar (q rows) named H to form the initial variance-covariance matrix (Sigma) which is calculated as Sigma = H'H of dimensions m x m (column dimensions, e.g., environments x environments).

2) The Sigma matrix is then center and scaled.

3) A Cholesky (L matrix) or SVD decomposition (U D V') is performed in the Sigma matrix.

4) n vectors from L (when Cholesky is used) or U sqrt(D) (when SVD is used) are kept to form Gamma. Gamma = L[,1:nPc] or Gamma = U[,1:nPC]. These are the so-called loadings (L for all loadings, Gamma for the subset of loadings).

4) Gamma is used to form a new incidence matrix as Z* = Z Gamma

5) This matrix is later used for the REML machinery to be used with the usc (unstructured) or smm (diagonal) structures to estimate variance components and factor scores. The resulting BLUPs from the mixed model are the optimized factor scores. Pretty much as a random regression over latent covariates.

This implementation does not update the loadings (latent covariates) during the REML process, only estimates the REML factor scores for fixed loadings. This is different to other software (e.g., asreml) where the loadings are updated during the REML process as well.

BLUPs for genotypes in all locations can be recovered as:

u = Gamma * u_scores

The resulting loadings (Gamma) and factor scores can be thought as an equivalent to the classical factor analysis.

As an additional information, notice that we calculate the factor loadings from BLUPs and the mixed model only calculates the factor scores. This is different to the asreml software where loadings are calculated as variance components through REML. Despite the difference we have run multiple examples and simulations and the BLUPs from both approaches are on average >0.98 correlated so you can be confident that our approach is robust.

Value

$Z a incidence matrix Z* = Z Gamma which is the original incidence matrix for the timevar multiplied by the loadings.

$Gamma a matrix of loadings or latent covariates.

$Sigma the covariance matrix used to calculate Gamma.

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.

Meyer K (2009) Factor analytic models for genotype by environment type problems and structured covariance matrices. Genetics Selection Evolution, 41:21

See Also

The `lmebreed` solver.

Examples

```
data(DT_h2)
DT <- DT_h2
DT=DT[with(DT, order(Env)), ]
indNames <- na.omit(unique(DT$Name))
A <- diag(length(indNames))
rownames(A) <- colnames(A) <- indNames



# fit diagonal model first to produce H matrix
Z <- with(DT, smm(Env))
diagFormula <- paste0( "y ~ Env + (0+", paste(colnames(Z), collapse = "+"),
                       "|| Name)")
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
print(as.formula(diagFormula))
ans1a <- lmebreed(as.formula(diagFormula),
                  relmat = list(Name=A),
                  data=DT)
vc <- VarCorr(ans1a); print(vc,comp=c("Variance"))
H0 <- ranef(ans1a)$Name # GxE table

# reduced rank model
Z <- with(DT,  rrm(Env, H = H0, nPC = 3))
Zd <- with(DT, smm(Env))
faFormula <- paste0( "y ~ Env + (0+", paste(colnames(Z), collapse = "+"),
                     "| Name) + (0+",paste(colnames(Zd), collapse = "+"),
                     "|| Name)")
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
print(as.formula(faFormula))
ansFA <- lmebreed(as.formula(faFormula),
                  relmat = list(Name=A),
                  data=DT)
vc <- VarCorr(ansFA); print(vc,comp=c("Variance"))
```

```
ve <- attr(vc, "sc")^2; ve

loadings=with(DT, rrm(Env, nPC = 3, H = H0, returnGamma = TRUE) )$Gamma
Gint <- loadings %*% vc$Name %*% t(loadings)
Gspec <- diag( unlist(lapply(vc[2:16], function(x){x[[1]]})) )
G <- Gint + Gspec
# lattice::levelplot(cov2cor(G))
# colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
# hv <- heatmap(cov2cor(G), col = colfunc(100), symm = TRUE)

u <- ranef(ansFA)$Name
uInter <- as.matrix(u[,1:3]) %*% t(as.matrix(loadings))
uSpec <- as.matrix(u[,-c(1:3)])
u <- uSpec + uInter
```

`simage` *Image of sparsity between two variables*

Description

`image` for sparsity.

Usage

```
simage(data, Var1=NULL, Var2=NULL, ...)
```

Arguments

`data`	model data of class `"data.frame"`
`Var1`	variable to set in the x axis
`Var2`	variable to set in the y axis
`...`	Further arguments to be passed to the image function.

Value

vector of image

Author(s)

Giovanny Covarrubias

See Also

`image`, `lmebreed`

Examples

```
# row x column combinations
data(DT_cpdata)
DT <- DT_cpdata
head(DT)
simage(data=DT, Var1 = "Rowf", Var2 = "Colf")

# dent x flint combinations
data(DT_technow)
DT <- DT_technow
head(DT)
simage(data=DT, Var1 = "dent", Var2 = "flint")

# male x female combinations
data(DT_halfdiallel)
DT <- DT_halfdiallel
head(DT)
simage(data=DT, Var1 = "male", Var2 = "female")
```

`simage2` *Shortcut to sparse matrix image*

Description

`image` for sparsity.

Usage

```
simage2(X, ...)
```

Arguments

`X`	object of class `"matrix"`
`...`	Further arguments to be passed to the image function.

Value

vector of image

Author(s)

Giovanny Covarrubias

See Also

`image`, `lmebreed`

Examples

```
simage2(matrix(0:8,3,3))
```

---

smm *sparse model matrix*

Description

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

Examples

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

tps *Get Tensor Product Spline Mixed Model Incidence Matrices*

Description

tps is a wrapper of tpsmmb function from the TPSbits package to avoid version dependencies but if you're using this function for your research please cite the TPSbits package. To be used with lme4breeding and its main function lmebreed.

Usage

```
tps(
  columncoordinates,
  rowcoordinates,
  nsegments=NULL,
  minbound=NULL,
  maxbound=NULL,
  degree = c(3, 3),
  penaltyord = c(2, 2),
  nestorder = c(1, 1),
  asreml = "grp",
  eigenvalues = "include",
  method = "Lee",
  stub = NULL
)
```

Arguments

columncoordinates	A string. Gives the name of data element holding column locations.
rowcoordinates	A string. Gives the name of data element holding row locations.
nsegments	A list of length 2. Number of segments to split column and row ranges into, respectively (= number of internal knots + 1). If only one number is specified, that value is used in both dimensions. If not specified, (number of unique values - 1) is used in each dimension; for a grid layout (equal spacing) this gives a knot at each data value.
minbound	A list of length 2. The lower bound to be used for column and row dimensions respectively; default calculated as the minimum value for each dimension.
maxbound	A list of length 2. The upper bound to be used for column and row dimensions respectively; default calculated as the maximum value for each dimension.
degree	A list of length 2. The degree of polynomial spline to be used for column and row dimensions respectively; default=3.
penaltyord	A list of length 2. The order of differencing for column and row dimensions, respectively; default=2.

`nestorder` A list of length 2. The order of nesting for column and row dimensions, respectively; default=1 (no nesting). A value of 2 generates a spline with half the number of segments in that dimension, etc. The number of segments in each direction must be a multiple of the order of nesting.

`asreml` A string. Indicates the types of structures to be generated for use in asreml models; default `"mbf"`. The appropriate eigenvalue scaling is included within the Z matrices unless setting `scaling="none"` is used, and then the scaling factors are supplied separately in the returned object.

- `asreml="mbf"` indicates the function should put the spline design matrices into structures for use with `"mbf"`;
- `asreml="grp"` indicates the function should add the composite spline design matrices (eg. for second-order differencing, matrices Xr1:Zc, Xr2:Zc, Zr:Xc1, Zr:Xc2 and Zc:Zr) into the data frame and provide a group list structure for each term;
- `asreml="sepgrp"` indicates the function should generate the individual X and Z spline design matrices separately (ie. Xc, Xr, Zc and Zr), plus the smooth x smooth interaction term as a whole (ie. Zc:Zr), and provide a group list structure for each term.
- `asreml="own"` indicates the function should generate the composite matrix (Xr:Zc | Zr:Xc | Zc:Zr) as a single set of columns.

`eigenvalues` A string. Indicates whether eigenvalues should be included within the Z design matrices `eigenvalues="include"`, or whether this scaling should be omitted (`eigenvalues="omit"`); default `eigenvalues="include"`. If the eigenvalue scaling is omitted from the Z design matrices, then it should instead be included in the model as a variance structure to obtain the correct TPspline model.

`method` A string. Method for forming the penalty; default=`"Lee"` ie the penalty from Lee, Durban & Eilers (2013, CSDA 61, 22-37). The alternative method is `"Wood"` ie. the method from Wood et al (2012, Stat Comp 23, 341-360). This option is a research tool and requires further investigation.

`stub` A string. Stub to be attached to names in the `mbf` list to avoid over-writing structures and general confusion.

Value

List of length 7, 8 or 9 (according to the `asreml` and `eigenvalues` parameter settings).

1. `data` = the input data frame augmented with structures required to fit tensor product splines in `asreml-R`. This data frame can be used to fit the TPS model.

   Added columns:

   - `TP.col`, `TP.row` = column and row coordinates
   - `TP.CxR` = combined index for use with smooth x smooth term
   - `TP.C.n` for n=1:(diff.c) = X parts of column spline for use in random model (where diff.c is the order of column differencing)
   - `TP.R.n` for n=1:(diff.r) = X parts of row spline for use in random model (where diff.r is the order of row differencing)

- TP.CR.n for n=1:((diff.c*diff.r)) = interaction between the two X parts for use in fixed model. The first variate is a constant term which should be omitted from the model when the constant (1) is present. If all elements are included in the model then the constant term should be omitted, eg. y ~ -1 + TP.CR.1 + TP.CR.2 + TP.CR.3 + TP.CR.4 + other terms...
- when asreml="grp" or "sepgrp", the spline basis functions are also added into the data frame. Column numbers for each term are given in the grp list structure.

2. mbflist = list that can be used in call to asreml (so long as Z matrix data frames extracted with right names, eg BcZ<stub>.df)
3. BcZ.df = mbf data frame mapping onto smooth part of column spline, last column (labelled TP.col) gives column index
4. BrZ.df = mbf data frame mapping onto smooth part of row spline, last column (labelled TP.row) gives row index
5. BcrZ.df = mbf data frame mapping onto smooth x smooth term, last column (labelled TP.CxR) maps onto col x row combined index
6. dim = list structure, holding dimension values relating to the model:
   (a) "diff.c" = order of differencing used in column dimension
   (b) "nbc" = number of random basis functions in column dimension
   (c) "nbcn" = number of nested random basis functions in column dimension used in smooth x smooth term
   (d) "diff.r" = order of differencing used in column dimension
   (e) "nbr" = number of random basis functions in column dimension
   (f) "nbrn" = number of nested random basis functions in column dimension used in smooth x smooth term
7. trace = list of trace values for ZGZ' for the random TPspline terms, where Z is the design matrix and G is the known diagonal variance matrix derived from eigenvalues. This can be used to rescale the spline design matrix (or equivalently variance components).
8. grp = list structure, only added for settings asreml="grp", asreml="sepgrp" or asreml="own". For asreml="grp", provides column indexes for each of the 5 random components of the 2D splines. For asreml="sepgrp", provides column indexes for each of the X and Z component matrices for the 1D splines, plus the composite smooth x smooth interaction term. For asreml="own", provides column indexes for the composite random model. Dimensions of the components can be derived from the values in the dim item. The Z terms are scaled by the associated eigenvalues when eigenvalues="include", but not when eigenvalues="omit".
9. eigen = list structure, only added for option setting eigenvalues="omit". Holds the diagonal elements of the inverse variance matrix for the terms Xc:Zr (called diagr), Zc:Xr (called diagc) and Zc:Zr (called diagcr).

Examples

```
data(DT_cpdata)
DT <- DT_cpdata
# add the units column
DT$units <- as.factor(1:nrow(DT))
# get spatial incidence matrix
Zs <- with(DT, tps(Row, Col))$All
```

```
rownames(Zs) <- DT$units
# reduce the matrix to its PCs
Z = with(DT, redmm(x=units, M=Zs, nPC=100))
# create dummy variable
spatial <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]



# fit model
mix1 <- lmebreed(Yield~ (1|Rowf) + (1|Colf) + (1|spatial),
                 addmat =list(spatial=Z),
                 data=DT)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
```

---

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 `lmebreed`

Examples

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata
## create the variance-covariance matrix
A <- A.mat(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