Package 'enhancer'

May 8, 2026

Version 1.1.1

Date 2026-04-07

Title Mixed-Effects Models Enhancing Functions

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

Description Special functions that enhance other mixed effect model packages by creating overlayed, reduced rank, and reduced model matrices together with multiple data sets to practice the use of these models. For more details see Covarrubias-Pazaran (2016) <doi:10.1371/journal.pone.0156744>.

Depends R(>= 3.5.0), Matrix (>= 1.0), MASS, 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

Enhances rmarkdown, knitr, orthopolynom, RSpectra, evola, lme4breeding, sommer

Config/testthat/edition 3

NeedsCompilation no

Date/Publication 2026-04-12 05:10:21 UTC

Contents

enhancer-package *Mixed Effect Model Enhancing Functions*

Description

enhancer is a collection of functions useful in the fitting of linear mixed effect models and general data manipulation. As more of my packages use the same special functions I decided to have the special functions as its own package to facilitate the maintenance.

Functions for genetic analysis

The package provides kernels to the estimate additive (A.matr) relationship matrix for diploid and polyploid organisms. A good converter from letter code to numeric format is implemented in the function atcg1234 and atcg1234BackTransform, 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).

Also the LD.decay and the neMarker functions can be used to calculate linkage disequilibrium and effective population size.

Functions for trial analysis

Recently, spatial modeling has been added added to enhancer using the two-dimensional spline tps function. Based on bbasis

Functions for special mixed models

Available are add.diallel.vars, leg, overlay, rrm, redmm, simGECorMat function.

Functions for data manipulation and others

Available are adiag1, imputev, propMissing replaceValues, stan, logspace function.

Simple coloring functions available are jet.colors, bathy.colors, transp

Keeping enhancer updated

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

library(devtools)

install_github("covaruber/enhancer")

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

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("enhancer")' to know how to cite it.

Author(s)

Giovanny Covarrubias-Pazaran

`A.matr` *Additive relationship matrix*

Description

Calculates the realized additive relationship matrix (R implementation).

Usage

```
A.matr(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.

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.matr(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.

Examples

```
DT = data.frame(Block=c( rep(1,64), rep(2,64) ),
                Par1=c( sort(rep(1:8,8)), sort(rep(1:8,8)) ),
                Par2=rep(1:8, 16),
                Ftime=rnorm(128)
)
DT2 <- add.diallel.vars(DT,par1="Par1", par2="Par2")
head(DT2)
```

---

adiag1 *Binds arrays corner-to-corner*

Description

Array generalization of blockdiag()

Usage

```
adiag1(... , pad=as.integer(0), do.dimnames=TRUE)
```

Arguments

...	Arrays to be binded together
pad	Value to pad array with; note default keeps integer status of arrays
do.dimnames	Boolean, with default TRUE meaning to return dimnames if possible. Set to FALSE if performance is an issue

Details

Binds any number of arrays together, corner-to-corner. Because the function is associative provided pad is of length 1, this page discusses the two array case.

If x=adiag1(a,b) and dim(a)=c(a_1,...,a_d), dim(b)=c(b_1,...,b_d); then all(dim(x)==dim(a)+dim(b)) and x[1:a_1,...,1:a_d]=a and x[(a_1+1):(a_1+b_1),...,(a_d+1):(a_d+b_d)]=b.

Dimnames are preserved, if both arrays have non-null dimnames, and do.dimnames is TRUE.

Argument pad is usually a length-one vector, but any vector is acceptable; standard recycling is used. Be aware that the output array (of dimension `dim(a)+dim(b)`) is filled with (copies of) `pad` *before* `a` and `b` are copied. This can be confusing.

Value

Returns an array of dimensions `dim(a)+dim(b)` as described above.

Note

In `adiag1(a,b)`, if `a` is a length-one vector, it is coerced to an array of dimensions `rep(1,length(dim(b)))`; likewise `b`. If both `a` and `b` are length-one vectors, return `diag(c(a,b))`.

If `a` and `b` are arrays, function `adiag1()` requires `length(dim(a))==length(dim(b))` (the function does not guess which dimensions have been dropped; see examples section). In particular, note that vectors are not coerced except if of length one.

`adiag1()` is used when padding magic hypercubes in the context of evaluating subarray sums.

Author(s)

Peter Wolf with some additions by Robin Hankin

Examples

```
a <- array( 1,c(2,2))
b <- array(-1,c(2,2))
adiag1(a,b)

## dropped dimensions can count:

b2 <- b1 <- b
dim(a) <- c(2,1,2)
dim(b1) <- c(2,2,1)
dim(b2) <- c(1,2,2)

dim(adiag1(a,b1))
dim(adiag1(a,b2))

## dimnames are preserved if not null:

a <- matrix(1,2,2,dimnames=list(col=c("red","blue"),size=c("big","small")))
b <- 8
dim(b) <- c(1,1)
dimnames(b) <- list(col=c("green"),size=c("tiny"))
adiag1(a,b)   #dimnames preserved
adiag1(a,8)   #dimnames lost because second argument has none.

## non scalar values for pad can be confusing:
q <- matrix(0,3,3)
adiag1(q,q,pad=1:4)

## following example should make the pattern clear:
```

```
adiag1(q,q,pad=1:36)


# Now, a use for arrays with dimensions of zero extent:
z <- array(dim=c(0,3))
colnames(z) <- c("foo","bar","baz")

adiag1(a,z)        # Observe how this has
                  # added no (ie zero) rows to "a" but
                  # three extra columns filled with the pad value

adiag1(a,t(z))
adiag1(z,t(z))     # just the pad value
```

`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

Examples

```
genotypes <- rbind(
  c("AAAG"  ,  "AAGG"  ,  "TTCC"  ,  "TTTT"  ,  "CCCC"),
  c("AAGG"  ,  "AGGG"  ,  "TTCC"  ,  "TTTT"  ,  "TCCC"),
  c("AAGG"  ,  "GGGG"  ,  "TTCC"  ,  "TTTT"  ,  NA),
  c("AAAA"  ,  "GGGG"  ,  "CCCC"  ,  "TTTT"  ,  NA),
  c("AAAA"  ,  "GGGG"  ,  "CCCC"  ,  "TTCC"  ,  NA)
)
rownames(genotypes) <- letters[1:5]
colnames(genotypes) <- paste0("m",1:5)

## convert markers to numeric format polyploid potatoes
numo <- atcg1234(data=genotypes, ploidy=4)
```

```
numo

## convert markers back to letters
Xb <- atcg1234BackTransform(marks= numo$M, refs= numo$ref.alleles)
Xb
```

`atcg1234BackTransform` *Letter to number converter*

Description

This function was designed to help users back transform the numeric marker matrices from the function atcg1234 into letters.

Usage

```
atcg1234BackTransform(marks, refs)
```

Arguments

`marks`	a centered marker matrix coming from atcg1234.
`refs`	a 2 x m matrix for m markers (columns) and 2 rows where the reference and alternate alleles for each marker are indicated.

Value

markers a new marker matrix leter coded according to the reference allele matrix.

Author(s)

Giovanny Covarrubias-Pazaran

Examples

```
genotypes <- rbind(
  c("AAAG"  ,  "AAGG"  ,  "TTCC"  ,  "TTTT"  ,  "CCCC"),
  c("AAGG"  ,  "AGGG"  ,  "TTCC"  ,  "TTTT"  ,  "TCCC"),
  c("AAGG"  ,  "GGGG"  ,  "TTCC"  ,  "TTTT"  ,  NA),
  c("AAAA"  ,  "GGGG"  ,  "CCCC"  ,  "TTTT"  ,  NA),
  c("AAAA"  ,  "GGGG"  ,  "CCCC"  ,  "TTCC"  ,  NA)
)
rownames(genotypes) <- letters[1:5]
colnames(genotypes) <- paste0("m",1:5)

## convert markers to numeric format polyploid potatoes
numo <- atcg1234(data=genotypes, ploidy=4)
numo
```

```
## convert markers back to letters
Xb <- atcg1234BackTransform(marks= numo$M, refs= numo$ref.alleles)
Xb
```

---

bathy.colors *Generate a sequence of colors for plotting bathymetric data.*

Description

bathy.colors(n) generates a sequence of n colors along a linear scale from light grey to pure blue.

Usage

```
bathy.colors(n, alpha = 1)
```

Arguments

n	The number of colors to return.
alpha	Alpha values to be passed to rgb().

Value

A vector of blue scale colors.

Examples

```
{
# Plot a colorbar using bathy.colors
image(matrix(seq(100), 100), col=bathy.colors(100))
}
```

---

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

Examples

```
nInds=3
nMarks=5
# set of lines 1
M1 <- matrix(rep(0,nInds*nMarks),nInds,nMarks)
for (i in 1:nInds) {
  M1[i,] <- ifelse(runif(nMarks)<0.5,-1,1)
}
rownames(M1) <- letters[1:nInds]
# set of lines 2
M2 <- matrix(rep(0,nInds*nMarks),nInds,nMarks)
for (i in 1:nInds) {
  M2[i,] <- ifelse(runif(nMarks)<0.5,-1,1)
}
rownames(M2) <- letters[(nInds+1):(nInds*2)]
# build additive and dominance matrices for single crosses
build.HMM(M1,M2)
```

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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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

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



############## sommer #################
if(requireNamespace("sommer")){
library(sommer)
mix1 <- mmes(TSW ~ Check.Gen,
             random = ~ Block + Genotype:Check,
             data=DT)
summary(mix1)$varcomp

}

############## lme4breeding #################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
## fit the mixed model and check summary
mix <- lmeb(TSW ~ Check.Gen + (1|Block) + (1|Genotype:Check),
             data=DT)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
sigma(mix)^2 # error variance
BLUP <- ranef(mix, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs
}

############## end #################
```

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.

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="_")
DT=DT[with(DT, order(envf_repf)), ]




if(requireNamespace("lme4breeding")){
library(lme4breeding)
# fit the models
mixm2 <- lmeb(value ~ (1|envf_repf) + (0+envf_repf||id),
                  data = DT,
                  relmat = list(id = MMT),
                  control = lmerControl(
                    restart_edge = FALSE
                  ), nIters=1000,
                  verbose=1L, rotation=TRUE
)

vc <- VarCorr(mixm2); print(vc,comp=c("Variance"))
ran2 = ranef(mixm2, condVar=FALSE)
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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_btdata)
DT <- DT_btdata
head(DT)



############# sommer ################

if(requireNamespace("sommer")){
library(sommer)
mix4 <- mmes(tarsus ~ sex,
             random = ~ dam + fosternest,
             rcov=~units,
             data = DT)
summary(mix4)$varcomp

# MULTI-TRAIT EXAMPLE
```

```
traits <- c("tarsus","back","hatchdate")
DT[,traits] <- apply(DT[,traits],2,scale)
DTL <- reshape(DT[,c("animal","sex","dam","fosternest", traits)],
               idvar = c("animal","sex","dam","fosternest"),
               varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait,animal)), ]
head(DTL)

mix3 <- mmes(value ~ trait:sex - 1, henderson=TRUE,
                random = ~ vsm(usm(trait),ism(dam)) +
                           vsm(usm(trait), ism(fosternest)),
                rcov= ~ vsm(dsm(trait),ism(units)),
                data = DTL)


summary(mix3)$varcomp
#### calculate the genetic correlation
cov2cor(mix3$theta[[1]])
cov2cor(mix3$theta[[2]])

}

############# lme4breeding ################

if(requireNamespace("lme4breeding")){
library(lme4breeding)
mix4 <- lmeb(tarsus ~ sex + (1|dam) + (1|fosternest),
             data = DT)
vc <- VarCorr(mix4); print(vc,comp=c("Variance"))
sigma(mix4)^2 # error variance
BLUP <- ranef(mix4, condVar=TRUE)
PEV <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

### 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 <- lmeb(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

}

############# end ################
```

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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_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)




################ sommer ##################

if(requireNamespace("sommer")){

library(sommer)
ans <- mmes(Yield ~ Location,
          random = ~ vsm(ism(GCA1),Gu=K1) + vsm(ism(GCA2),Gu=K2), # + vsm(ism(SCA),Gu=S),
             rcov=~units,
             data=DT)
summary(ans)$varcomp

## mmec uses the inverse of the relationship matrix
K1i <- solve(K1 + diag(1e-4,ncol(K1),ncol(K1)))
K1i <- as(as(as( K1i,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(K1i, 'inverse')=TRUE
K2i <- solve(K2 + diag(1e-4,ncol(K2),ncol(K2)))
K2i <- as(as(as( K2i,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(K2i, 'inverse')=TRUE
Si <- solve(S + diag(1e-4,ncol(S),ncol(S)))
Si <- as(as(as( Si,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Si, 'inverse')=TRUE
ans2 <- mmes(Yield ~ Location,
         random = ~ vsm(ism(GCA1),Gu=K1i) + vsm(ism(GCA2),Gu=K2i), # + vsm(ism(SCA),Gu=Si),
             henderson=TRUE,
             rcov=~units,
             data=DT)
summary(ans2)$varcomp

}

################ lme4breeding ##################

if(requireNamespace("lme4breeding")){
library(lme4breeding)
```

```
ans <- lmeb(Yield ~ Location + (1| GCA1) + (1|GCA2),
               relmat = list(GCA1=K1,
                             GCA2=K2),
               data=DT)
vc <- VarCorr(ans); print(vc,comp=c("Variance"))
sigma(ans)^2 # error variance
BLUP <- ranef(ans, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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.

Giovanny Covarrubias-Pazaran (2024). evola: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. G3 Gene|Genomes|Genetics 11(2):jkaa017. https://doi.org/10.1093/g3journal/jkaa017.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. Genome Research, 19, 136-142. http://genome.cshlp.org/content/19/1/136.

Examples

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



######## sommer #########

if(requireNamespace("sommer")){
library(sommer)
mix1 <- mmes(Yield~1, henderson=FALSE,
              random=~vsm(ism(id),Gu=A)
                      + Rowf + Colf,
              rcov=~units,
              data=DT)
summary(mix1)$varcomp

## mmec uses the inverse of the relationship matrix
Ai <- solve(A + diag(1e-4,ncol(A),ncol(A)))
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE
mix2 <- mmes(Yield~1, henderson=TRUE,
              random=~vsm(ism(id),Gu=Ai)
                      + Rowf + Colf,
              rcov=~units,
              data=DT)
summary(mix2)$varcomp

vg <- summary(mix2)$varcomp[1,1] # genetic variance
G <- A*vg # genetic variance-covariance
Ci <- mix2$Ci # coefficient matrix
ind <- as.vector(mix2$partitions$`vsm(ism(id), Gu = Ai)`)
ind <- seq(ind[1],ind[2])
Ctt <- Ci[ind,ind] # portion of Ci for genotypes
R2 <- (G - Ctt)/G # reliability matrix
mean(diag(R2)) # average reliability of the trial
####======================####
```

```
#### multivariate model ####
####     2 traits       ####
#########################
traits <- c("color","Yield")
DT[,traits] <- apply(DT[,traits],2,scale)
DTL <- reshape(DT[,c("id", traits)],
               idvar = c("id"),
               varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait)), ]
head(DTL)

# if using mmes=TRUE you need to provide the inverse
Ai <- solve(A + diag(1e-4,ncol(A),ncol(A)))
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE
#### be patient take some time
ansm <- mmes( value ~ trait, # henderson=TRUE,
               random=~ vsm(usm(trait), ism(id), Gu=A), # Ai if henderson
               rcov=~ vsm(dsm(trait), ism(units)),
               data=DTL)
cov2cor(ansm$theta[[1]])

}

######## lme4breeding #########

if(requireNamespace("lme4breeding")){
library(lme4breeding)
mix1 <- lmeb(Yield~ (1|id) + (1|Rowf) + (1|Colf),
                 relmat=list(id=A),
                 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, suppressOpt = TRUE,
               start=getME(mix1, "theta"),
               data=DT2)
predsMix2 <- ranef(mix2)
condVAR <- lapply(predsMix2, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

# 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])

}

######## evola #########

if(requireNamespace("evola")){
library(evola)
DT$occ <- 1
DT$Yield <- imputev(DT$Yield)
A <- A[DT$id,DT$id]
# get best 20 individuals weighting variance by ~0.5=(30*pi)/180
res<-evolafit(formula=cbind(Yield, occ)~id, dt= DT,
              # constraints: if sum is greater than this ignore
              constraintsUB = c(Inf,20),
              # constraints: if sum is smaller than this ignore
              constraintsLB= c(-Inf,-Inf),
              # weight the traits for the selection
              b = c(1,0),
              # population parameters
              nCrosses = 100, nProgeny = 10,
              recombGens=1, nChr=1, mutRateAllele=0,
              # coancestry parameters
              D=A, lambda= (30*pi)/180 , nQtlStart = 20,
              # selection parameters
              propSelBetween = 0.5, propSelWithin =0.5,
              nGenerations = 40)

Q <- pullQtlGeno(res$pop, simParam = res$simParam, trait=1); Q <- Q/2
best = bestSol(res$pop)[,"fitness"];best
qa = (Q %*% DT$Yield)[best,]; qa
qDq = Q[best,] %*% A %*% Q[best,]; qDq
sum(Q[best,]) # total # of inds selected

evolmonitor(res)

plot(DT$Yield, col=as.factor(Q[best,]),
     pch=(Q[best,]*19)+1)

pareto(res)

}

############ end ############
```

DT_envs *Simulated phenotypic records for a set of 100 environments and 200 entries.*

Description

This is a simulated dataset that aims to show how to pick the most representative locations and how to optimize sparse testing allocation.

Usage

```
data("DT_envs")
```

Format

The format is: chr "DT_envs"

Source

This data was simulated for experimental design applications.

References

Giovanny Covarrubias-Pazaran (2024). evola: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. G3 Gene|Genomes|Genetics 11(2):jkaa017. https://doi.org/10.1093/g3journal/jkaa017.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. Genome Research, 19, 136-142. http://genome.cshlp.org/content/19/1/136.

Examples

```
data(DT_envs)
# scale passport data for environments
S=scale(W)
# build a relationship matrix between environments
G=tcrossprod(S)/ncol(S)
G[1:4,1:4]
# build the dataset for the genetic algorithm
DT <- data.frame(cov=rnorm(nrow(G)), occ=1, id=rownames(G))
nLocs=10 # desired number of representative locations



if(requireNamespace("evola")){
library(evola)
res<-evolafit(formula=cbind(cov, occ)~id, dt= DT,
              # constraints: if sum is greater than this ignore
```

```
                constraintsUB = c(Inf, Inf),
                # constraints: if sum is smaller than this ignore
                constraintsLB= c(-Inf, nLocs),
                # weight the traits for the selection
                b = c(1,0),
                # population parameters
                nCrosses = 100, nProgeny = 20,
                recombGens=1, nChr=1, mutRateAllele=0,
                # coancestry parameters
                D=G, lambda= 0 , nQtlStart = nLocs,
                # selection parameters
                fitnessf = inbFun, selectTop = FALSE,
                # fixNumQtlPerInd = TRUE,
                # propSelBetween = 0.5, propSelWithin =0.5,
                nGenerations = 50, verbose = TRUE)

Q <- pullQtlGeno(res$pop, simParam = res$simParam, trait=1); Q <- Q/2
best <- bestSol(res$pop, selectTop = FALSE)[,"fitness"]
solution <- Q[best,]
sum(solution) # total # of inds selected
names(solution[which(solution>0)])

evolmonitor(res)
pareto(res)

svdW <- svd(G, nu = 2, nv = 2)
PCW <- G %*% svdW$v
rownames(PCW) <- rownames(G)
plot(PCW[,1], PCW[,2], col = (solution*2)+2,
     pch=(solution*14)+4,  xlab = "pc1", ylab = "pc2")
labs <- rownames(PCW)
labs[which(solution==0)]=""
text(x=PCW[,1], y=PCW[,2], labels=labs, cex=0.5, pos=3)

}
```

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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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
head(DT)



################## sommer ####################

if(requireNamespace("sommer")){
library(sommer)
## Compound simmetry (CS) model
ans1 <- mmes(Yield~Env,
             random= ~ Name + Env:Name,
             rcov= ~ units,
             data=DT)
summary(ans1)$varcomp

####============================================####
#### Univariate heterogeneous variance models  ####
####============================================####

## Compound simmetry (CS) + Diagonal (DIAG) model
DT=DT[with(DT, order(Env)), ]
ans2 <- mmes(Yield~Env,
             random= ~Name + vsm(dsm(Env),ism(Name)),
             rcov= ~ vsm(dsm(Env),ism(units)),
             data=DT)
summary(ans2)

####============================================####
#### Multi-trait variance models               ####
####============================================####
```

```
# stack traits
traits <- c("Yield","Weight")
DT[,traits] <- apply(DT[,traits],2,scale)
DTL <- reshape(DT[,c("Name","Env","Block", traits)],
               idvar = c("Name","Env","Block"),
               varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait,Env)), ]
head(DTL)

## model
ans1 <- mmes(value~ trait,
             random= ~ vsm(usm(trait), ism(Name)),
             rcov= ~ vsm(dsm(trait), ism(units)),
             data=DTL)
summary(ans1)$varcomp
cov2cor(ans1$theta$`vsm(usm(trait), ism(Name`)

}

################## lme4breeding ####################

if(requireNamespace("lme4breeding")){
library(lme4breeding)
## Compound simmetry (CS) model
ans1 <- lmeb(Yield~Env + (1|Name) + (1|Env:Name),
             data=DT)
vc <- VarCorr(ans1); print(vc,comp=c("Variance"))

BLUP <- ranef(ans1, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

## Main (M) + Diagonal (DIAG) model
## with relationship matrix
ansCSDG <- lmeb(Yield ~ Env + (Env||Name),
                    relmat = list(Name = A ),
                    data=DT)
vc <- VarCorr(ansCSDG); print(vc,comp=c("Variance"))

## Main (M) + Diagonal (DIAG) model
## with diagonal residuals
## with relationship matrix
ansCSDG <- lmeb(Yield ~ Env + (Env||Name) + (0+Env||unitsR),
                    relmat = list(Name = A ),
                    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 mixed model packages.

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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"



############ sommer ##############
if(requireNamespace("sommer")){
library(sommer)
mix1 <- mmes(yield~entryc,
             random=~block+trt,
             rcov=~units,
             data=data1)
summary(mix1)$varcomp

}

############ lme4breeding ##############
if(requireNamespace("lme4breeding")){
library(lme4breeding)
mix1 <- lmeb(yield~entryc + (1|block)+(1|trt),
             data=data1)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance
BLUP <- ranef(mix1, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

`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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_fulldiallel)
DT <- DT_fulldiallel
head(DT)



############## sommer ################

if(requireNamespace("sommer")){
library(sommer)
mix <- mmes(stems~1, random=~female+male, data=DT)
summary(mix)

}

############## lme4breeding ################

if(requireNamespace("lme4breeding")){
library(lme4breeding)
mix <- lmeb(stems~1 + (1|female)+(1|male),
               data=DT)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
sigma(mix)^2 # error variance
BLUP <- ranef(mix, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

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.

Examples

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



############## sommer #################

if(requireNamespace("sommer")){
library(sommer)
```

```
mix1 <- mmes(BWT~1,
             random=~vsm(ism(ANIMAL),Gu=A),
             rcov=~units,
             data=DT)
summary(mix1)$varcomp

## mmes algorithm uses the inverse of the relationship matrix
## if you select henderson=TRUE
Ai <- solve(A + diag(1e-4,ncol(A),ncol(A)))
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE
####======================####
#### multivariate model ####
####     2 traits       ####
####======================####
head(DT)

traits <- c("BWT","TARSUS")
DT[,traits] <- apply(DT[,traits],2,scale)
DTL <- reshape(DT[,c("ANIMAL","MOTHER","BYEAR","SEX", traits)],
               idvar = c("ANIMAL","MOTHER","BYEAR","SEX"),
               varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait,ANIMAL)), ]
head(DTL)

# #### fit the multivariate model with no fixed effects (intercept only)
mix2 <- mmes(value~trait, # henderson=TRUE,
             random=~vsm(usm(trait),ism(ANIMAL),Gu=A),
             rcov=~vsm(dsm(trait),ism(units)),
             data=DTL)
summary(mix2)$varcomp
cov2cor(mix2$theta[[1]])

}

############## lme4breeding #################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
## fit the model with no fixed effects (intercept only)
mix1 <- lmeb(BWT~ (1|ANIMAL),
                relmat = list(ANIMAL=A),
                data=DT)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance
BLUP <- ranef(mix1, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs


### 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 <- lmeb(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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_h2)
DT <- DT_h2
head(DT)



############### sommer ################
if(requireNamespace("sommer")){
library(sommer)
DT=DT[with(DT, order(Env)), ]
ans1 <- mmes(y~Env, henderson=TRUE,
             random=~vsm(dsm(Env),ism(Name)) + vsm(dsm(Env),ism(Block)),
             rcov=~vsm(dsm(Env),ism(units)),
             data=DT)
summary(ans1)$varcomp

}

############### lme4breeding ################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
# model with:
# Main+Diagonal for Name
# Diagonal for Block
# Diagonal for residuals
DT=DT[with(DT, order(Env)), ]
ans1b <- lmeb(y ~ Env + (Env||Name) + (0+Env||Block) + (0+Env||unitsR),
                 verbose = 1L,
                 data=DT)
vc <- VarCorr(ans1b); print(vc,comp=c("Variance"))
sigma(ans1b)^2 # error variance

BLUP <- ranef(ans1b, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
############### end ################
```

`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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_halfdiallel")
DT <- DT_halfdiallel
head(DT)
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)



############# sommer #############
if(requireNamespace("sommer")){
library(sommer)
A <- diag(7); colnames(A) <- rownames(A) <- 1:7;A # if you want to provide a covariance matrix
#### model using overlay
modh <- mmes(sugar~1,
             random=~vsm(ism(overlay(femalef,malef, sparse = FALSE)), Gu=A)
             + genof,
             data=DT)
summary(modh)$varcomp

# if ussing mmes=TRUE provide Gu with inverses and give more iterations
# Ai <- solve(A + diag(1e-4,ncol(A),ncol(A)))
# Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
# attr(Ai, 'inverse')=TRUE

}

############# lme4breeding #############
if(requireNamespace("lme4breeding")){
```

```
library(lme4breeding)
# overlay matrix to be added to the addmat argument
Z <- with(DT, overlay(femalef,malef) )
## model using overlay without relationship matrix
modh <- lmeb(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 <- lmeb(sugar ~ (1|genof) + (1|fema),
                addmat = list(fema=Z),
                relmat = list(fema=A),
                data=DT)
vc <- VarCorr(modh); print(vc,comp=c("Variance"))
sigma(modh)^2 # error variance

BLUP <- ranef(modh, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}

############## end #############
```

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.

Examples

```
data(DT_ige)
DT <- DT_ige


########################################
############### sommer #################
########################################
if(requireNamespace("sommer")){
library(sommer)
# Indirect genetic effects model without covariance between DGE and IGE
modIGE <- mmes(trait ~ block, dateWarning = FALSE,
               random = ~ focal + neighbour,
               rcov = ~ units, nIters=100,
              data = DT)
summary(modIGE)$varcomp
pmonitor(modIGE)

# Indirect genetic effects model with covariance between DGE and IGE using relationship matrices
modIGE <- mmes(trait ~ block, dateWarning = FALSE,
               random = ~ covm( vsm(ism(focal)), vsm(ism(neighbour)) ),
               rcov = ~ units, nIters=100,
              data = DT)
summary(modIGE)$varcomp
pmonitor(modIGE)

# form relationship matrix
Ai <- solve(A_ige + diag(1e-5, nrow(A_ige),nrow(A_ige) ))
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE
# Indirect genetic effects model with covariance between DGE and IGE using relationship matrices
modIGE <- mmes(trait ~ block, dateWarning = FALSE,
               random = ~ covm( vsm(ism(focal), Gu=Ai), vsm(ism(neighbour), Gu=Ai) ),
               rcov = ~ units, nIters=100,
              data = DT)
summary(modIGE)$varcomp
pmonitor(modIGE)

}
############################################
############### lme4breeding ###############
############################################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
```

```
# Indirect genetic effects model without covariance between DGE and IGE
modIGE <- lmeb(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 <- lmeb(trait ~ block + (1|focal) + (1|neighbour),
                   relmat = list(focal=A_ige,
                                 neighbour=A_ige),
                   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))
## Fit the model
modIGE <- lmeb(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)
condVAR <- lapply(blups, function(x){attr(x, which="postVar")}) # take sqrt() for SEs
pairs(blups$both)
cov2cor(vc$both)

}

########## end ###########
```

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 lmeb 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.

Examples

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



########################################
############### sommer #################
########################################
if(requireNamespace("sommer")){
library(sommer)
library(orthopolynom)
mRR2<-mmes(Y~ 1 + Xf
           , random=~ vsm(usm(leg(X,1)),ism(SUBJECT))
           , rcov=~units
           , data=DT)
summary(mRR2)$varcomp

}

############################################
############### lme4breeding #################
############################################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
library(orthopolynom)
```

```
Z <- with(DT, smm(leg(X,1)) )
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
## diagonal random regression Y ~ Xf + (0+leg0+leg1|| SUBJECT)
## unstructured random regression Y ~ Xf + (0+leg0+leg1| SUBJECT)
mRR2b<-lmeb(Y ~ Xf + (0+leg0+leg1| SUBJECT),
               , data=DT)
vc <- VarCorr(mRR2b); print(vc,comp=c("Variance"))
sigma(mRR2b)^2 # error variance

BLUP <- ranef(mRR2b, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_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)




################################################
###############  sommer.     #################
################################################

if(requireNamespace("sommer")){
library(sommer)
# GRIFFING MODEL 2 with reciprocal effects ############################

mod1hb <- mmes(Ftime ~ 1, data=DT2,
               random = ~ Block
               # GCA male & female overlayed
               + vsm(ism(overlay(Par1, Par2)))
               # SCA effects (includes cross and selfs)
               + cross.id
               # SCAR reciprocal effects (remaining variance in crosses;
               # if zero there's no reciprocal effects)
               + vsm(dsm(cross.type), ism(cross.id)) )
summary(mod1hb)$varcomp

##                                    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 <- mmes(Ftime ~ Block + is.cross, data=DT2, nIters = 50,
              random = ~
                # GCA effects for all (hybrids and inbreds)
                vsm(ism(overlay(Par1, Par2)))
              # GCA effect (calculated only in hybrids; remaining variance)
              + vsm(ism(is.cross),ism(overlay(Par1, Par2)))
              # SCA effect (calculated in hybrids only)
              + vsm(ism(is.cross), ism(cross.id))
              )
```

```
summary(mod1h)$varcomp

##                                          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 <- mmes(Ftime ~ Block + is.cross, data=DT2, nIters = 100,
              random = ~
                # GCAC (only for hybrids)
                vsm(ism(is.cross),ism(overlay(Par1, Par2)))
              # male GCA (only for inbreds)
              + vsm(ism(is.self),ism(Par1))
              # SCA (for hybrids only)
              + vsm(ism(is.cross), ism(cross.id))
              )
summary(mod1h)$varcomp
##                                          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 <- mmes(Ftime ~ Block + is.cross, data=DT2, nIters = 100,
              random = ~
                # GCAC (for hybrids only)
                vsm(ism(is.cross),ism(overlay(Par1, Par2)))
              # male GCA (for selfs only)
              + vsm(ism(is.self),ism(Par1))
              # SCA (for hybrids only)
              + vsm(ism(is.cross), ism(cross.id))
              # SCAR reciprocal effects (remaning SCA variance)
              + vsm(ism(cross.type), ism(cross.id))
              )
summary(mod1h)$varcomp

##                                            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 <- mmes(Ftime ~ Block + is.cross, data=DT2,nIters = 100,
```

```
               random = ~
                 # GCAC (for hybrids only)
                 vsm(ism(is.cross),ism(overlay(Par1, Par2)))
               # RGCA: exclude selfs (to identify reciprocal GCA effects)
               + vsm(ism(cross.type),ism(overlay(Par1, Par2)))
               # male GCA (for selfs only)
               + vsm(ism(is.self),ism(Par1))
               # SCA (for hybrids only)
               + vsm(ism(is.cross), ism(cross.id))
               # SCAR: exclude selfs (if zero there's no reciprocal SCA effects)
               + vsm(ism(cross.type), ism(cross.id))
               )
summary(mod1h)$varcomp

##                                           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

}

##############################################
############### lme4breeding #################
##############################################

if(requireNamespace("lme4breeding")){
library(lme4breeding)
## 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 <- lmeb(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"))

BLUP <- ranef(mod1h, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

##                                    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 <- lmeb(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 <- lmeb(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 <- lmeb(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 <- lmeb(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.

Other references:

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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_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.matr(marks)



##############################################
###############  sommer.     #################
##############################################
```

```
if(requireNamespace("sommer")){
library(sommer)
A <- A.mat(marks)
D <- D.mat(marks)
###========================================####
### run as mixed model
###========================================####
ans <- mmes(tuber_shape~1,
            random=~vsm(ism(Name), Gu=A),
            data=DT2)
summary(ans)$varcomp
# if using mmes=TRUE provide Gu as inverse
Ai <- solve(A + diag(1e-4,ncol(A),ncol(A)))
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE

}

##############################################
############### lme4breeding #################
##############################################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
## run as mixed model
A <- A + diag(1e-4,ncol(A),ncol(A))
ans <- lmeb(tuber_shape~ (1|Name),
                relmat = list(Name=A),
                data=DT2)
vc <- VarCorr(ans); print(vc,comp=c("Variance"))
sigma(ans)^2 # error variance

BLUP <- ranef(ans, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

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.

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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



################################################
############### sommer.      #################
################################################
if(requireNamespace("sommer")){
library(sommer)
A <- A.mat(M$M)
mix <- mmes(Protein.content~1,
            random = ~vsm(ism(geno), Gu=A) + geno,
            rcov=~units,
            data=DT)
summary(mix)$varcomp
# if using henderson=TRUE provide Gu as inverse
Ai <- solve(A + diag(1e-6,ncol(A),ncol(A)))
```

```
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE



## MULTI-TRAIT MODEL
## reshape in long format the dataset
traits <- c("Protein.content","Flag.leaf.length")
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)

M <- DTLM <- atcg1234(GT)
A <- A.mat(M$M)
mix <- mmes(value~trait,
            random = ~vsm(usm(trait),ism(geno), Gu=A) ,
            rcov=~vsm(dsm(trait), ism(units)),
            data=DTL)
summary(mix)$varcomp
cov2cor(mix$theta$`vsm(usm(trait), ism(geno), Gu = A`)

}

##############################################
############### lme4breeding #################
##############################################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
### univariate model
A <- A.matr(M$M)
A <- A + diag(1e-4, ncol(A), ncol(A))
mix <- lmeb(Protein.content ~ (1|geno),
                relmat = list(geno=A),
                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, geno)), ]
head(DTL)

system.time(
  mix <- lmeb(value ~ (0+trait|geno),
                  relmat = list(geno=A),
                  rotation = TRUE,
                  data=DTL)
```

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

BLUP <- ranef(mix, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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.

Examples

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



##############################################
###############  sommer.     #################
##############################################
if(requireNamespace("sommer")){
library(sommer)
#################################
fm2 <- mmes(Reaction ~ Days,
            random= ~ Subject,
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

#################################
fm2 <- mmes(Reaction ~ Days,
            random= ~ Subject + vsm(ism(Days), ism(Subject)),
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

#################################
fm2 <- mmes(Reaction ~ Days,
            random= ~ Subject + vsm(ism(Days), ism(Subject)),
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

#################################
fm2 <- mmes(Reaction ~ Days,
            random= ~ vsm(ism(Days), ism(Subject)),
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

}

##############################################
############### lme4breeding #################
##############################################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
#################################
fm1 <- lmeb(Reaction ~ Days + (1 | Subject),
               data=DT)
vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
sigma(fm1)^2 # error variance

#################################
fm1 <- lmeb(Reaction ~ Days + (Days || Subject), data=DT)
```

```
vc <- VarCorr(fm1); print(vc,comp=c("Variance"))

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

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

BLUP <- ranef(fm1, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
```

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.matr 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.

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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.

Giovanny Covarrubias-Pazaran (2024). evola: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. G3 Gene|Genomes|Genetics 11(2):jkaa017. https://doi.org/10.1093/g3journal/jkaa017.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. Genome Research, 19, 136-142. http://genome.cshlp.org/content/19/1/136.

Examples

```
data(DT_technow)
DT <- DT_technow

Md <- apply(Md_technow,2,as.numeric)
rownames(Md) <- rownames(Md_technow)
Mf <- apply(Mf_technow,2,as.numeric)
rownames(Mf) <- rownames(Mf_technow)

Md <- (Md*2) - 1
Mf <- (Mf*2) - 1

Ad <- A.matr(Md)
Af <- A.matr(Mf)
Ad <- Ad + diag(1e-4, ncol(Ad), ncol(Ad))
Af <- Af + diag(1e-4, ncol(Af), ncol(Af))



################# sommer ###################
if(requireNamespace("sommer")){
library(sommer)
ans2 <- mmes(GY~1,
             random=~vsm(ism(dent),Gu=Ad) + vsm(ism(flint),Gu=Af),
             rcov=~units,
             data=DT)
summary(ans2)$varcomp

Adi <- solve(Ad + diag(1e-4,ncol(Ad),ncol(Ad)))
Adi <- as(as(as( Adi,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Adi, 'inverse')=TRUE
Afi <- solve(Af + diag(1e-4,ncol(Af),ncol(Af)))
Afi <- as(as(as( Afi,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Afi, 'inverse')=TRUE
####====================####
#### multivariate model ####
```

```
####     2 traits       ####
####======================####
head(DT)

traits <- c("GY","GM")
DT[,traits] <- apply(DT[,traits],2,scale)
DTL <- reshape(DT[,c("hybrid","dent","flint", traits)],
               idvar = c("hybrid","dent","flint"),
               varying = traits,
               v.names = "value", direction = "long",
               timevar = "trait", times = traits )
DTL <- DTL[with(DTL, order(trait,hybrid)), ]
head(DTL)

M <- rbind(Md,Mf)
A <- A.mat(M)
Ai <- solve(A + diag(1e-4,ncol(A),ncol(A)))
Ai <- as(as(as( Ai,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ai, 'inverse')=TRUE

ans3 <- mmes(value~trait, henderson=TRUE,
             random=~vsm(usm(trait),ism(overlay(dent,flint)),Gu=Ai),
             rcov=~ vsm(dsm(trait), ism(units)),
             data=DTL)
summary(ans3)
cov2cor(ans3$theta[[1]])

}

################## lme4breeding ####################

if(requireNamespace("lme4breeding")){
library(lme4breeding)
## simple model
ans2 <- lmeb(GY ~ (1|dent) + (1|flint),
                data=DT)
vc <- VarCorr(ans2); print(vc,comp=c("Variance"))
BLUP <- ranef(ans2, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

### with relationship matrices
ans2 <- lmeb(GY ~ (1|dent) + (1|flint),
             relmat = list(dent=Ad,
                           flint=Af),
             data=DT)
vc <- VarCorr(ans2); print(vc,comp=c("Variance"))

### overlayed model
M <- rbind(Md,Mf)
A <- A.matr(M)
A <- A + diag(1e-4,ncol(A), ncol(A))
Z <- with(DT, overlay(dent,flint) )
Z = Z[which(!is.na(DT$GY)),]
```

```
#### model using overlay without relationship matrix
ans2 <- lmeb(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
BLUP <- ranef(ans2, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

### rotated model for hybrids
H <- kronecker(Ad,Af, make.dimnames = TRUE)
H <- H[which(colnames(H)%in% DT$hy),which(colnames(H)%in% DT$hy)]

ans3 <- lmeb(GY ~ (1|hy),
             relmat = list(hy=H),
             rotation=TRUE,
             data=DT)
vc <- VarCorr(ans3); print(vc,comp=c("Variance"))
BLUP <- ranef(ans3, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

}
################## evola ####################

if(requireNamespace("evola")){
library(evola)
DT <- DT_technow
  DT$occ <- 1; DT$occ[1]=0
  combos <- build.HMM(Md,Mf, return.combos.only = TRUE)
  combos <- combos$data.used[which(combos$data.used$hybrid %in%DT$hy),]
  M <- build.HMM(Md,Mf, custom.hyb = combos)
  A <- A.matr(M$HMM.add)
  A <- A[DT$hy,DT$hy]
  # run the genetic algorithm
  # we assig a weight to x'Dx of (20*pi)/180=0.34
  res<-evolafit(formula = c(GY, occ)~hy,
                dt= DT,
                # constraints: if sum is greater than this ignore
                constraintsUB = c(Inf,100),
                # constraints: if sum is smaller than this ignore
                constraintsLB= c(-Inf,-Inf),
                # weight the traits for the selection
                b = c(1,0),
                # population parameters
                nCrosses = 100, nProgeny = 10,
                recombGens=1, nChr=1, mutRateAllele=0,
                # coancestry parameters
                D=A, lambda= (20*pi)/180 , nQtlStart = 90,
                # selection parameters
                propSelBetween = 0.5, propSelWithin =0.5,
                nGenerations = 20)
```

```
  Q <- pullQtlGeno(res$pop, simParam = res$simParam, trait=1); Q <- Q/2
  best = bestSol(res$pop)[,"fitness"]
  qa = (Q %*% DT$GY)[best,]; qa
  qAq = Q[best,] %*% A %*% Q[best,]; qAq
  sum(Q[best,]) # total # of inds selected

  evolmonitor(res)
  plot(DT$GY, col=as.factor(Q[best,]),
       pch=(Q[best,]*19)+1)

  pareto(res)

}
################## end ##################
```

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 agroclimatic 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

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

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.

Giovanny Covarrubias-Pazaran (2024). evola: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. G3 Gene|Genomes|Genetics 11(2):jkaa017. https://doi.org/10.1093/g3journal/jkaa017.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. Genome Research, 19, 136-142. http://genome.cshlp.org/content/19/1/136.

Examples

```
data(DT_wheat)
DT <- DT_wheat
GT <- apply(GT_wheat,2,as.numeric)
rownames(GT) <- rownames(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))

K <- tcrossprod(GT-1)
m <- sum(diag(K))/nrow(K)
K <- K/m
K <- K + diag(1e-05, ncol(K), ncol(K))
K[1:4,1:4]
##
head(DT)



############## sommer ###############
if(requireNamespace("sommer")){
library(sommer)
mix0 <- mmes(pheno~1,
             random = ~vsm(ism(id),Gu=K),
             rcov=~units,
             data=DT[which(DT$env == "e1"),])
```

```
summary(mix0)$varcomp
# if using mmes=TRUE provide Gu as inverse
Ki <- solve(K + diag(1e-4,ncol(K),ncol(K)))
Ki <- as(as(as( Ki,  "dMatrix"), "generalMatrix"), "CsparseMatrix")
attr(Ki, 'inverse')=TRUE

}

############## lme4breeding ###############

if(requireNamespace("lme4breeding")){
library(lme4breeding)
#### main effect model
system.time(
mix0 <- lmeb(pheno ~ (1|id),
                 relmat = list(id=K),
                 data=DT)
                 )
vc <- VarCorr(mix0); print(vc,comp=c("Variance"))
sigma(mix0)^2 # error variance
BLUP <- ranef(mix0, condVar=TRUE)
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs

#### unstructured model
DT <- DT[with(DT, order(env, id)), ] # sort for rotation
system.time(
  mix1 <- lmeb(pheno ~ (0 + env | id),
                   relmat = list(id=K),
                   rotation = TRUE,
                   data=DT)
)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
sigma(mix1)^2 # error variance

}

############## evola ###############

if(requireNamespace("evola")){
library(evola)
DT <- as.data.frame(DT_wheat)
DT$id <- rownames(DT) # IDs
DT$occ <- 1; DT$occ[1]=0 # to track occurrences
DT$dummy <- 1; DT$dummy[1]=0 # dummy trait

# if genomic
G <- K
# if pedigree
A <- A_wheat
A[1:4,1:4]
##Perform eigenvalue decomposition for clustering
##And select cluster 5 as target set to predict
pcNum=25
```

```
svdWheat <- svd(A, nu = pcNum, nv = pcNum)
PCWheat <- A %*% svdWheat$v
rownames(PCWheat) <- rownames(A)
DistWheat <- dist(PCWheat)
TreeWheat <- cutree(hclust(DistWheat), k = 5 )
plot(PCWheat[,1], PCWheat[,2], col = TreeWheat,
     pch = as.character(TreeWheat), xlab = "pc1", ylab = "pc2")
vp <- rownames(PCWheat)[TreeWheat == 3]; length(vp)
tp <- setdiff(rownames(PCWheat),vp)

As <- A[tp,tp]
DT2 <- DT[rownames(As),]
DT2$cov <- apply(A[tp,vp],1,mean)

# we assign a weight to x'Dx of (60*pi)/180=1
res<-evolafit(formula=cbind(cov, occ)~id, dt= DT2,
              # constraints: if sum is greater than this ignore
              constraintsUB = c(Inf, 100),
              # constraints: if sum is smaller than this ignore
              constraintsLB= c(-Inf, -Inf),
              # weight the traits for the selection
              b = c(1,0),
              # population parameters
              nCrosses = 100, nProgeny = 10,
              recombGens=1, nChr=1, mutRateAllele=0,
              # coancestry parameters
              D=As, lambda= (60*pi)/180 , nQtlStart = 90,
              # selection parameters
              propSelBetween = 0.5, propSelWithin =0.5,
              nGenerations = 30, verbose = TRUE)

Q <- pullQtlGeno(res$pop, simParam = res$simParam, trait=1); Q <- Q/2
best <- bestSol(res$pop)[,"fitness"]
sum(Q[best,]) # total # of inds selected
pareto(res)

}

################ end ##################
```

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)



##############################################
############### sommer.      #################
##############################################
if(requireNamespace("sommer")){
library(sommer)
m3 <- mmes(fixed=Y ~ V + N + V:N,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)
summary(m3)$varcomp

}

##############################################
############### lme4breeding #################
##############################################
if(requireNamespace("lme4breeding")){
library(lme4breeding)
mix <- lmeb(Y ~ V + N + V:N +
             (1|B) + (1|B:MP),
           data = DT)
vc <- VarCorr(mix); print(vc,comp=c("Variance"))
sigma(mix)^2 # error variance

BLUP <- ranef(mix, condVar=TRUE)
```

```
condVAR <- lapply(BLUP, function(x){attr(x, which="postVar")}) # take sqrt() for SEs
}
```

`I.matr` *Identity relationship matrix*

Description

Shortcut to get an identity relationship matrix.

Usage

```
I.matr(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.

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	an alternative vector with values declaring the method to be applied by.

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
####============================================####
#### generate your mickey mouse -log10(p-values)
####============================================####
set.seed(1253)
x <- rnorm(100)
x[sample(1:100,10)] <- NA
imputev(x)
```

jet.colors *Generate a sequence of colors alog the jet colormap.*

Description

`jet.colors(n)` generates a sequence of n colors from dark blue to cyan to yellow to dark red. It is similar to the default color schemes in Python's matplotlib or MATLAB.

Usage

```
jet.colors(n, alpha = 1)
```

Arguments

`n`	The number of colors to return.
`alpha`	The transparency value of the colors. See `?rgb` for details.

Value

A vector of colors along the jet colorramp.

Examples

```
{
# Plot a colorbar with jet.colors
image(matrix(seq(100), 100), col=jet.colors(100))
}
```

LD.decay *Calculation of linkage disequilibrium decay*

Description

This function calculates the LD decay based on a marker matrix and a map with distances between markers in cM or base pairs.

Usage

```
LD.decay(markers,map,silent=FALSE,unlinked=FALSE,gamma=0.95)
```

Arguments

markers	a numeric matrix of markers (columns) by individuals (rows) in -1, 0, 1 format.
map	a data frame with 3 columns "Locus" (name of markers), "LG" (linkage group or chromosome), and "Position" (in cM or base pairs).
silent	a TRUE/FALSE value statement indicating if the program should or should not display the progress bar. silent=TRUE means that will not be displayed.
unlinked	a TRUE/FALSE value statement indicating if the program should or should not calculate the alpha(see next argument) percentile of interchromosomal LD.
gamma	a percentile value for LD breakage to be used in the calculation of interchromosomal LD extent.

Value

$resp a list with 3 elements; "by.LG", "all.LG", "LDM". The first element (by.LG) is a list with as many elements as chromosomes where each contains a matrix with 3 columns, the distance, the r2 value, and the p-value associated to the chi-square test for disequilibrium. The second element (all.LG) has a big matrix with distance, r2 values and p-values, for each point from all chromosomes in a single data.frame. The third element (LDM) is the matrix of linkage disequilibrium between pairs of markers.

If unlinked is selected the program should return the gamma percentile interchromosomal LD (r2) for each chromosome and average.

References

Laido, Giovanni, et al. Linkage disequilibrium and genome-wide association mapping in tetraploid wheat (Triticum turgidum L.). PloS one 9.4 (2014): e95211.

Examples

```
nInds=300
nMarks=500

#random population of nInds lines with nMarks markers
M <- matrix(rep(0,nInds*nMarks),nInds,nMarks)
for (i in 1:nInds) {
  M[i,] <- ifelse(runif(nMarks)<0.5,-1,1)
}
rownames(M) <- paste0("ind",1:nInds)
colnames(M) <- paste0("mark",1:nMarks)

mapM = data.frame(Locus=colnames(M),Position=1:nMarks,LG=1)

res <- LD.decay(M, mapM)


 #### subset only markers with significant LD
res$all.LG <- res$all.LG[which(res$all.LG$p < .001),]
#### plot the LD decay
with(res$all.LG, plot(r2~d,col=transp("cadetblue"),
```

```
                    xlim=c(0,55), ylim=c(0,1),
                    pch=20,cex=0.5,yaxt="n",
                    xaxt="n",ylab=expression(r^2),
                    xlab="Distance in cM")
                    )
axis(1, at=seq(0,55,5), labels=seq(0,55,5))
axis(2,at=seq(0,1,.1), labels=seq(0,1,.1), las=1)

#### if you want to add the loess regression lines
#### this could take a long time!!!
# mod <- loess(r2 ~ d, data=res$all.LG)
# par(new=TRUE)
# lilo <- predict(mod,data.frame(d=1:55))
# plot(lilo, bty="n", xaxt="n", yaxt="n", col="green",
#     type="l", ylim=c(0,1),ylab="",xlab="",lwd=2)
# res3 <- LD.decay(markers=CPgeno, map=mapCP,
#                unlinked = TRUE,gamma = .95)
# abline(h=res3$all.LG, col="red")
```

---

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.

Examples

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



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

---

logspace *Decreasing logarithmic trend*

Description

logspace creates a vector with decreasing logaritmic trend.

Usage

```
logspace(x, p=2)
```

Arguments

x	sequence of values to pass through the function.
p	power to be applied to the values.

Value

$res a vector of length n with logarithmic decrease trend.

Author(s)

Giovanny Covarrubias-Pazaran

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
plot(logspace(1:100,p=1))
plot(logspace(1:100,p=2))
plot(logspace(1:100,p=3))
```

`manhattan` *Creating a manhattan plot*

Description

This function was designed to create a manhattan plot using a data frame with columns "Chrom" (Chromosome), "Position" and "p.val" (significance for the test).

Usage

```
manhattan(map, col=NULL, fdr.level=0.05, show.fdr=TRUE, PVCN=NULL, ylim=NULL, ...)
```

Arguments

`map`	the data frame with 3 columns with names; "Chrom" (Chromosome), "Position" and "p.val" (significance for the test).
`col`	colors prefered by the user to be used in the manhattan plot. The default is NULL which will use the red-blue palette.
`fdr.level`	false discovery rate to be drawn in the plot.
`show.fdr`	a TRUE/FALSE value indicating if the FDR value should be shown in the manhattan plot or not. By default is TRUE meaning that will be displayed.
`PVCN`	In case the user wants to provide the name of the column that should be treated as the "p.val" column expected by the program in the 'map' argument.
`ylim`	the y axis limits for the manhattan plot if the user wants to customize it. By default the plot will reflect the minimum and maximum values found.
`...`	additional arguments to be passed to the plot function such as pch, cex, etc.

Value

If all parameters are correctly indicated the program will return:

$plot.data a manhattan plot

Author(s)

Giovanny Covarrubias-Pazaran

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
#random population of 200 lines with 1000 markers
M <- matrix(rep(0,200*1000),1000,200)
for (i in 1:200) {
  M[,i] <- ifelse(runif(1000)<0.5,-1,1)
}
colnames(M) <- 1:200
set.seed(1234)
pp <- abs(rnorm(500,0,3));pp[23:34] <- abs(rnorm(12,0,20))
geno <- data.frame(Locus=paste("m",1:500, sep="."),Chrom=sort(rep(c(1:5),100)),
                   Position=rep(seq(1,100,1),5),
                   p.val=pp, check.names=FALSE)
geno$Locus <- as.character(geno$Locus)
## look at the data, 5LGs, 100 markers in each
## -log(p.val) value for simulated trait
head(geno)
tail(geno)
manhattan(geno)
```

---

map.plot *Creating a genetic map plot*

Description

This function was designed to create a genetic map plot using a data frame indicating the Linkage Group (LG), Position and marker names (Locus).

Usage

```
map.plot(data, trait = NULL, trait.scale = "same",
        col.chr = NULL, col.trait = NULL, type = "hist", cex = 0.4,
        lwd = 1, cex.axis = 0.4, cex.trait=0.8, jump = 5)
```

Arguments

data	the data frame with 3 columns with names; Locus, LG and Position
trait	if something wants to be plotted next the linkage groups the user must indicate the name of the column containing the values to be ploted, i.e. p-values, LOD scores, X2 segregation distortion values, etc.

trait.scale	is trait is not NULL, this is a character value indicating if the y axis limits for the trait plotted next to the chromosomes should be the same or different for each linkage group. The default value is "same", which means that the same y axis limit is conserved across linkage groups. For giving an individual y axis limit for each linkage group write "diff".
col.chr	a vector with color names for the chromosomes. If NULL they will be drawn in gray-black scale.
col.trait	a vector with color names for the dots, lines or histogram for the trait plotted next to the LG's
type	a character value indicating if the trait should be plotted as scatterplot 'dot', histogram 'hist', line 'line' next to the chromosomes.
cex	the cex value determining the size of the cM position labels in the LGs
lwd	the width of the lines in the plot
cex.axis	the cex value for sizing the labels of LGs and traits plotted (top labels)
cex.trait	the cex value for sizing the dots or lines of the trait plotted
jump	a scalar value indicating how often should be drawn a number next to the LG indicating the position. The default is 5 which means every 5 cM a label will be drawn, i.e. 0,5,10,15,... cM.

Value

If all parameters are correctly indicated the program will return:

$plot.data a plot with the LGs and the information used to create a plot

Author(s)

Giovanny Covarrubias-Pazaran

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
#random population of 200 lines with 1000 markers
M <- matrix(rep(0,200*1000),1000,200)
for (i in 1:200) {
  M[,i] <- ifelse(runif(1000)<0.5,-1,1)
}
colnames(M) <- 1:200
set.seed(1234)
geno <- data.frame(Locus=paste("m",1:500, sep="."),LG=sort(rep(c(1:5),100)),
                   Position=rep(seq(1,100,1),5),
                   X2=rnorm(500,10,4), check.names=FALSE)
geno$Locus <- as.character(geno$Locus)
## look at the data, 5LGs, 100 markers in each
```

```
## X2 value for segregation distortion simulated
head(geno)
tail(geno)
table(geno$LG) # 5 LGs, 100 marks
map.plot(geno, trait="X2", type="line")
map.plot(geno, trait="X2", type="hist")
map.plot(geno, trait="X2", type="dot")



data("DT_cpdata")
MP <- MP_cpdata
colnames(MP)[3] <- c("LG")
head(MP)
map.plot(MP, type="line", cex=0.6)
```

---

neMarker *Effective population size based on marker matrix*

Description

'neMarker' uses a marker matrix to approximate the effective population size (Ne) by discovering how many individuals are needed to sample all possible alleles in a population.

Usage

```
neMarker(M, neExplore=NULL, maxMarker=1000, nSamples=5)
```

Arguments

M	marker matrix coded in a numerical faashion (any allele dosage is fine).
neExplore	a vector of numbers with the effective population sizes to be explored.
maxMarker	maximum number of markers to use for the analysis.
nSamples	number of individuals to sample for the Ne calculation.

Value

$S3 A vector with allele coverage based on different number of individuals

Author(s)

Giovanny Covarrubias-Pazaran

References

Not based on any theory published yet but in a solid intuition on what is really important for a breeding program when we ask what is the effective population size

Examples

```
nInds=300
nMarks=1000

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

# run the function
ne <- neMarker(M, neExplore = seq(2,300,100), nSamples = 5)
# plot(ne$Ne)
```

---

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.

Examples

```
femalef <- as.factor(c("a","b","c"))
malef <- as.factor(c("d","e","a"))
overlay(femalef,malef, sparse = TRUE)
overlay(femalef,malef, sparse = FALSE)
```

---

propMissing *Proportion of missing data*

Description

`propMissing` quick calculation of the proportion of missing data in a vector.

Usage

```
propMissing(x)
```

Arguments

x	vector of observations.

Value

$res a numeric value with the proportion of missing data.

Author(s)

Giovanny Covarrubias-Pazaran

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommec. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
x <- c(1:10, NA)
propMissing(x)
```

`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.

Examples

```
# data set including the covariate to be reduced
DT <- data.frame(Env=sort(rep(paste("e",1:5,sep="_"), 2)))
DT
# GxE matrix
H0 <- matrix(rnorm(30), nrow=5, ncol=6) # 5 envs, 6 features
rownames(H0) <- paste("e",1:5,sep="_")
colnames(H0) <- paste("x",1:6,sep="_")
H0
# get compression

if(requireNamespace("RSpectra")){

library(RSpectra)
Z <- with(DT,  redmm(Env, M = H0, nPC = 3))
Z

}
```

---

replaceValues *Replace the values of a vector*

Description

Simple function to map a vector of values to another.

Usage

```
replaceValues(Source, Search, Replace)
```

Arguments

Source	A source of values to change.
Search	The values to search.
Replace	The equivalent values to replace.

Details

Simple function to map a vector of values another.

Value

$res new values from a second vector.

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
x <- letters[1:5]
y <- 1:5
z <- rep(letters[1:5],3)

replaceValues(Source=z, Search=x, Replace=y)
```

---

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 `lmeb` 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

Examples

```
# data set including the covariate to be reduced
DT <- data.frame(Env=sort(rep(paste("e",1:5,sep="_"), 2)))
DT
# GxE matrix
H0 <- matrix(rnorm(30), nrow=6, ncol=5) # 5 envs, 6 inds
colnames(H0) <- paste("e",1:5,sep="_")
rownames(H0) <- paste("x",1:6,sep="_")
H0
# get latent factors
Z <- with(DT,  rrm(Env, H = H0, nPC = 3))
Z
```

---

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

Examples

```
# row x column combinations existing

DT <- data.frame(Row=sort(rep(1:10,5)), Col=rep(1:5,10))
DT$Rowf <- as.factor(DT$Row)
DT$Colf <- as.factor(DT$Col)
DT <- DT[-1,] # remove one intentionally
simage(data=DT, Var1 = "Rowf", Var2 = "Colf")

# you can also inpect which cross combinations exist
```

---

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

Examples

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

---

simGECorMat *Create a GE correlation matrix for simulation purposes.*

Description

Makes a simple correlation matrix based on the number of environments and megaenvironments desired.

Usage

```
simGECorMat(nEnv,nMegaEnv,mu=0.7, v=0.2, mu2=0, v2=0.3)
```

Arguments

nEnv	Number of environments to simulate. Needs to be divisible by the nMegaEnv argument.
nMegaEnv	Number of megaenvironments to simulate.
mu	Mean value of the genetic correlation within megaenvironments.
v	variance in the genetic correlation within megaenvironments.
mu2	Mean value of the genetic correlation between megaenvironments.
v2	variance in the genetic correlation between megaenvironments.

Details

Simple simulation of a correlation matrix for environments and megaenvironments.

Value

G the correlation matrix

$G the correlation matrix

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
simGECorMat(9,3)
```

`stan` *Standardize a vector of values in range 0 to 1*

Description

Simple function to map a vector of values to the range of 0 and 1 values to have a better behavior of the algorithm.

Usage

```
stan(x, lb=0, ub=1)
```

Arguments

`x`	A vector of numeric values.
`lb`	Lower bound value to map the x values.
`ub`	Upper bound value to map the x values.

Details

Simple function to map a vector of values to the range of 0 and 1 values to have a better behavior of the algorithm.

Value

$res new values in range 0 to 1

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

Examples

```
x <- rnorm(20, 10, 3);x
stan(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. This function is internally used by the spl2Dmatrices function to get Tensor-Product P-Spline Mixed Model Bits (design matrices) for use with sommer and lme4breeding.

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).

---

transp *Creating color with transparency*

Description

This function takes a color and returns the same with a certain alpha grade transparency.

Usage

```
transp(col, alpha=0.5)
```

Arguments

col	Color to be used for transparency
alpha	Grade of transparency desired

Details

No major details.

Value

If arguments are correctly specified the function returns:

$res A new color with certain grade of transparency

References

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
transp("red", alpha=0.5)
```

---

wald.test *Wald Test for Model Coefficients*

Description

Computes a Wald χ^2 test for 1 or more coefficients, given their variance-covariance matrix.

Usage

```
  wald.test(Sigma, b, Terms = NULL, L = NULL, H0 = NULL,
            df = NULL, verbose = FALSE)
  ## S3 method for class 'wald.test'
print(x, digits = 2, ...)
```

Arguments

Sigma	A var-cov matrix, usually extracted from one of the fitting functions (e.g., lm, glm, ...).
b	A vector of coefficients with var-cov matrix Sigma. These coefficients are usually extracted from one of the fitting functions available in R (e.g., lm, glm,...).
Terms	An optional integer vector specifying which coefficients should be *jointly* tested, using a Wald χ^2 or F test. Its elements correspond to the columns or rows of the var-cov matrix given in Sigma. Default is NULL.
L	An optional matrix conformable to b, such as its product with b i.e., L %*% b gives the linear combinations of the coefficients to be tested. Default is NULL.
H0	A numeric vector giving the null hypothesis for the test. It must be as long as Terms or must have the same number of columns as L. Default to 0 for all the coefficients to be tested.
df	A numeric vector giving the degrees of freedom to be used in an F test, i.e. the degrees of freedom of the residuals of the model from which b and Sigma were fitted. Default to NULL, for no F test. See the section **Details** for more information.
verbose	A logical scalar controlling the amount of output information. The default is FALSE, providing minimum output.
x	Object of class "wald.test"
digits	Number of decimal places for displaying test results. Default to 2.
...	Additional arguments to print.

Details

The key assumption is that the coefficients asymptotically follow a (multivariate) normal distribution with mean = model coefficients and variance = their var-cov matrix.
One (and only one) of Terms or L must be given. When L is given, it must have the same number of columns as the length of b, and the same number of rows as the number of linear combinations of coefficients. When df is given, the χ^2 Wald statistic is divided by m = the number of linear combinations of coefficients to be tested (i.e., length(Terms) or nrow(L)). Under the null hypothesis H0, this new statistic follows an $F(m, df)$ distribution.

Value

An object of class wald.test, printed with print.wald.test.

References

Diggle, P.J., Liang, K.-Y., Zeger, S.L., 1994. Analysis of longitudinal data. Oxford, Clarendon Press, 253 p.
Draper, N.R., Smith, H., 1998. Applied Regression Analysis. New York, John Wiley & Sons, Inc., 706 p.

Examples

```
data("mtcars")
DT <- mtcars
head(DT)
m3 <- lm(mpg ~ as.factor(carb),
          data = DT)

# extrac coefficients and variance of coefficients

wald.test(b = m3$coefficients, Sigma = vcov(m3), Terms = 2)

# make a test for a specific linear combination
LL <- matrix(0,nrow=1, ncol=6)
LL[1,2] <- 1
LL[1,3] <- -1
LL

wald.test(b = m3$coefficients, Sigma = vcov(m3), L=LL)

# sommer has 'Ci' slot for Sigma and 'bu' for coefficients
# lme4breeding has condVar() function to extract Sigma for random effects and
#  vcov() for fixed effect and has fixef() and ranef() to extract fixed and
#   random coefficients.
```

Index