Package 'sommer'

February 3, 2025

Type Package

Title Solving Mixed Model Equations in R

Version 4.3.7

Date 2025-02-01

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

Description Structural multivariate-univariate linear mixed model solver for estimation of multiple random effects with unknown variance-covariance structures (e.g., heterogeneous and unstructured) and known covariance among levels of random effects (e.g., pedigree and genomic relationship matrices) (Covarrubias-Pazaran, 2016 <doi:10.1371/journal.pone.0156744>; Maier et al., 2015 <doi:10.1016/j.ajhg.2014.12.006>; Jensen et al., 199 timates can be obtained using the Direct-Inversion Newton-Raphson and Direct-Inversion Average Information algorithms for the problems r x r (r being the number of records) or using the Henderson-based average information algorithm for the problem c x c (c being the number of coefficients to estimate). Spatial models can also be fitted using the two-dimensional spline functionality available.

Depends R (>= 3.5.0), Matrix (>= 1.1.1), methods, stats, MASS, crayon

License GPL (>= 2)

Imports Rcpp (>= 0.12.19)

BugReports https://github.com/covaruber/sommer/issues

URL https://github.com/covaruber/sommer

LinkingTo Rcpp, RcppArmadillo, RcppProgress

Suggests rmarkdown, knitr, plyr, parallel, orthopolynom, RSpectra, lattice, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2025-02-03 17:20:02 UTC

Contents

`sommer-package` **So***lving* **M***ixed* **M***odel* **E***quations in* **R**

Description

Sommer is a structural multivariate-univariate linear mixed model solver for multiple random effects allowing the specification and/or estimation of variance covariance structures. REML estimates can be obtained using two major methods

Direct-Inversion (Newton-Raphson and Average Information) » `mmer` function

Henderson's mixed model equations (Average Information) » `mmec` function

The algorithms are coded in C++ using the Armadillo library to optimize dense matrix operations common in genomic models. Sommer was designed to include complex covariance structures, e.g., unstructured, reduced-rank, diagonal. And also to model relationships between levels of a random effect, e.g., additive, dominance and epistatic relationship structures.

The `mmer` function can deal well with small and medium-size data sets (< 10,000 observations/records for average computers given the computational burden carried by the direct-inversion algorithms) since it works in the c > r problem and inverts an r x r matrix (being r the number of records and c the number of coefficients). On the other hand, the `mmec` function can deal with greater number of records (r>250K) as long as the number of coefficients to estimate is < 10,000 coefficients (c) since it works in the r > c problem and inverts a c x c matrix (being c the number of coefficients). The `predict.mmer` and `predict.mmec` functions can be used to obtain adjusted means. This package returns variance-covariance components, BLUPs, BLUEs, residuals, fitted values, variances-covariances for fixed and random effects, etc.

Functions for genetic analysis

The package provides kernels to estimate additive (`A.mat`), dominance (`D.mat`), epistatic (`E.mat`), single-step (`H.mat`) relationship matrices for diploid and polyploid organisms. It also provides flexibility to fit other genetic models such as full and half diallel models and random regression models.

A good converter from letter code to numeric format is implemented in the function `atcg1234`, which supports higher ploidy levels than diploid. Additional functions for genetic analysis have been included such as build a genotypic hybrid marker matrix (`build.HMM`), plot of genetic maps

(map.plot), creation of manhattan plots (manhattan). If you need to use pedigree you need to convert your pedigree into a relationship matrix (use the 'getA' function from the pedigreemm package).

Functions for statistical analysis and S3 methods

The vpredict function can be used to estimate standard errors for linear combinations of variance components (e.g. ratios like h2). The r2 function calculates reliability. S3 methods are available for some parameter extraction such as:

+ predict.mmer, predict.mmec,

+ fitted.mmer, fitted.mmec,

+ residuals.mmer, residuals.mmec,

+ summary.mmer, summary.mmec,

+ coef.mmer, coef.mmec,

+ anova.mmer, anova.mmec,

+ plot.mmer, plot.mmec.

Functions for trial analysis

Recently, spatial modeling has been added added to sommer using the two-dimensional spline (spl2Da and spl2Db for mmer and spl2Dc for mmec) functions.

Keeping sommer updated

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

library(devtools)

install_github("covaruber/sommer")

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

Tutorials

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

vignette("v1.sommer.quick.start")

vignette("v2.sommer.changes.and.faqs")

vignette("v3.sommer.qg")

vignette("v4.sommer.gxe")

or visit **https://covaruber.github.io**

Getting started

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

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

* DT_h2 to calculate heritability

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

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

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

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

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

* DT_btdata dataset contains an animal (birds) model.

* DT_legendre simulated dataset for random regression model.

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

Differences of sommer >= 4.1.7 with previous versions

Since version 4.1.7 I have introduced the mme-based average information function 'mmec' which is much faster when dealing with the r > c problem (more records than coefficients to estimate). This introduces its own covariance structure functons such as vsc(), usc(), dsc(), atc(), csc(). Please give it a try, although is in early phase of development.

Differences of sommer >= 3.7.0 with previous versions

Since version 3.7 I have completly redefined the specification of the variance-covariance structures to provide more flexibility to the user. This has particularly helped the residual covariance structures and the easier combination of custom random effects and overlay models. I think that although this will bring some uncomfortable situations at the beggining, in the long term this will help users to fit better models. In esence, I have abandoned the asreml formulation (not the structures available) given it's limitations to combine some of the sommer structures but all covariance structures can now be fitted using the 'vsr' functions.

Differences of sommer >= 3.0.0 with previous versions

Since version 3.0 I have decided to focus in developing the multivariate solver and for doing this I have decided to remove the M argument (for GWAS analysis) from the mmer function and move it to it's own function GWAS.

Before the mmer solver had implemented the usr(trait), diag(trait), at(trait) asreml formulation for multivariate models that allow to specify the structure of the trait in multivariate models. Therefore the MVM argument was no longer needed. After version 3.7 now the multi-trait structures can be specified in the Gt and Gtc arguments of the vsr function.

The Average Information algorithm had been removed in the past from the package because of its instability to deal with very complex models without good initial values. Now after 3.7 I have

brought it back after I noticed that starting with NR the first three iterations gives enough flexibility to the AI algorithm.

Keep in mind that sommer uses direct inversion (DI) algorithm which can be very slow for datasets with many observations (big 'n'). The package is focused in problems of the type p > n (more random effect(s) levels than observations) and models with dense covariance structures. For example, for experiment with dense covariance structures with low-replication (i.e. 2000 records from 1000 individuals replicated twice with a covariance structure of 1000x1000) sommer will be faster than MME-based software. Also for genomic problems with large number of random effect levels, i.e. 300 individuals (n) with 100,000 genetic markers (p). On the other hand, for highly replicated trials with small covariance structures or n > p (i.e. 2000 records from 200 individuals replicated 10 times with covariance structure of 200x200) asreml or other MME-based algorithms will be much faster and I recommend you to use that software.

Models Enabled

The core of the package are the `mmer` and `mmec` (formula-based) functions which solve the mixed model equations. The functions are an interface to call the 'NR' Direct-Inversion Newton-Raphson, 'AI' Direct-Inversion Average Information or the mme-based Average Information (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2016). Since version 2.0 sommer can handle multivariate models. Following Maier et al. (2015), the multivariate (and by extension the univariate) mixed model implemented has the form:

where y_i is a vector of trait phenotypes, β_i is a vector of fixed effects, u_i is a vector of random effects for individuals and e_i are residuals for trait i (i = 1,..., t). The random effects (u_1 ... u_i and e_i) are assumed to be normally distributed with mean zero. X and Z are incidence matrices for fixed and random effects respectively. The distribution of the multivariate response and the phenotypic variance covariance (V) are:

where K is the relationship or covariance matrix for the kth random effect (u=1,...,k), and R=I is an identity matrix for the residual term. The terms $\sigma^2_{g_i}$ and $\sigma^2_{\epsilon_i}$ denote the genetic (or any of the kth random terms) and residual variance of trait i, respectively and $\sigma_{g_{ij}}$ and $\sigma_{\epsilon_{ij}}$ the genetic (or any of the kth random terms) and residual covariance between traits i and j (i=1,...,t, and j=1,...,t). The algorithm implemented optimizes the log likelihood:

where || is the determinant of a matrix. And the REML estimates are updated using a Newton optimization algorithm of the form:

Where, theta is the vector of variance components for random effects and covariance components among traits, H^-1 is the inverse of the Hessian matrix of second derivatives for the kth cycle, dL/dsigma^2_i is the vector of first derivatives of the likelihood with respect to the variance-covariance components. The Eigen decomposition of the relationship matrix proposed by Lee and Van Der Werf (2016) was included in the Newton-Raphson algorithm to improve time efficiency. Additionally, the popular vpredict function to estimate standard errors for linear combinations of variance components (i.e. heritabilities and genetic correlations) was added to the package as well.

GWAS Models

The GWAS models in the sommer package are enabled by using the M argument in the functions `GWAS`, which is expected to be a numeric marker matrix. Markers are treated as fixed effects according to the model proposed by Yu et al. (2006) for diploids, and Rosyara et al. (2016) (for polyploids). The matrices X and M are both fixed effects, but they are separated by 2 different

arguments to distinguish factors such as environmental and design factors for the argument "X" and markers with "M".

The genome-wide association analysis is based on the mixed model:

$$y = X\beta + Zg + M\tau + e$$

where β is a vector of fixed effects that can model both environmental factors and population structure. The variable g models the genetic background of each line as a random effect with $Var[g] = K\sigma^2$. The variable τ models the additive SNP effect as a fixed effect. The residual variance is $Var[\varepsilon] = I\sigma_e^2$

When principal components are included (P+K model), the loadings are determined from an eigenvalue decomposition of the K matrix and are used in the fixed effect part.

The argument "P3D" introduced by Zhang et al. (2010) can be used with the P3D argument. When P3D=FALSE, this function is equivalent to AI/NR with REML where the variance components are estimated for each SNP or marker tested (Kang et al. 2008). When P3D=TRUE, it is equivalent to NR (Kang et al. 2010) where the assumption is that variance components for all SNP/markers are the same and therefore the variance components are estimated only once (and markers are tested in a WLS framework being the the weight matrix (M) the inverse of the phenotypic variance matrix (V)). Therefore, P3D=TRUE option is faster but can underestimate significance compared to P3D=FALSE.

Multivariate GWAS are based in Covarrubias-Pazaran et al. (2018, In preparation), which adjusts betas for all response variables and then does the regular GWAS with such adjusted betas or marker effects.

For extra details about the methods please read the canonical papers listed in the References section.

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

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

Covarrubias-Pazaran G. 2018. Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

Bernardo Rex. 2010. Breeding for quantitative traits in plants. Second edition. Stemma Press. 390 pp.

Gilmour et al. 1995. Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. Biometrics 51(4):1440-1450.

Henderson C.R. 1975. Best Linear Unbiased Estimation and Prediction under a Selection Model. Biometrics vol. 31(2):423-447.

Kang et al. 2008. Efficient control of population structure in model organism association mapping. Genetics 178:1709-1723.

Lee et al. 2015. MTG2: An efficient algorithm for multivariate linear mixed model analysis based on genomic information. Cold Spring Harbor. doi: http://dx.doi.org/10.1101/027201.

Maier et al. 2015. Joint analysis of psychiatric disorders increases accuracy of risk prediction for schizophrenia, bipolar disorder, and major depressive disorder. Am J Hum Genet; 96(2):283-294.

Searle. 1993. Applying the EM algorithm to calculating ML and REML estimates of variance components. Paper invited for the 1993 American Statistical Association Meeting, San Francisco.

Yu et al. 2006. A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Genetics 38:203-208.

Tunnicliffe W. 1989. On the use of marginal likelihood in time series model estimation. JRSS 51(1):15-27.

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

####============================================####
#### EXAMPLES
#### Different models with sommer
####============================================####

data(DT_example)

# DT <- DT_example
# DT=DT[with(DT, order(Env)), ]
# head(DT)
#
# ####============================================####
# #### Univariate homogeneous variance models  ####
# ####============================================####
#
# ## Compound simmetry (CS) model
# ans1 <- mmer(Yield~Env,
#              random= ~ Name + Env:Name,
#              rcov= ~ units,
#              data=DT)
# summary(ans1)
#
# ans2 <- mmec(Yield~Env,
#              random= ~ Name + Env:Name,
#              rcov= ~ units,
#              data=DT)
# summary(ans2)
```

```
#
# ####============================================####
# #### Univariate heterogeneous variance models  ####
# ####============================================####
# ## Compound simmetry (CS) + Diagonal (DIAG) model
# ans3 <- mmer(Yield~Env,
#              random= ~Name + vsr(dsr(Env),Name),
#              rcov= ~ vsr(dsr(Env),units),
#              data=DT)
# summary(ans3)
#
# ans4 <- mmec(Yield~Env,
#              random= ~Name + vsc(dsc(Env),isc(Name)),
#              rcov= ~ vsc(dsc(Env),isc(units)),
#              data=DT)
# summary(ans4)
```

`A.mat` *Additive relationship matrix*

Description

Calculates the realized additive relationship matrix. Currently is the C++ implementation of van Raden (2008).

Usage

```
A.mat(X,min.MAF=0,return.imputed=FALSE)
```

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.
`return.imputed`	When TRUE, the imputed marker matrix is returned.

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

$X the imputed marker matrix

References

Endelman, J.B., and J.-L. Jannink. 2012. Shrinkage estimation of the realized relationship matrix. G3:Genes, Genomes, Genetics. 2:1405-1413. doi: 10.1534/g3.112.004259

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

See Also

`mmer` – the core function of the package

Examples

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

A <- A.mat(X)

####============================================####
#### take a look at the Genomic relationship matrix
#### (just a small part)
####============================================####
# 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

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

See Also

The core functions of the package mmer, the overlay function and the DT_mohring example.

Examples

```
####==========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####==========================================####
data(DT_mohring)
DT <- DT_mohring
head(DT)
DT2 <- add.diallel.vars(DT,par1="Par1", par2="Par2")
head(DT2)
## see ?DT_mohring for an example on how to use the data to fit diallel models.
```

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

See Also

mmer – the core function of the package

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
```

Description

Univariate version of the average information (AI) algorithm.

Usage

```
AI(X=NULL,Z=NULL, Zind=NULL, Ai=NULL,y=NULL,S=NULL, H=NULL,
   nIters=80, tolParConvLL=1e-4, tolParConvNorm=.05, tolParInv=1e-6, theta=NULL,
    thetaC=NULL, thetaF=NULL, addScaleParam=NULL,
    weightInfEMv=NULL, weightInfMat=NULL)
```

Arguments

X	an incidence matrix for fixed effects.
Z	Z is a list of lists each element contains the Z matrices required for the covariance structure specified for a random effect.
Zind	vector specifying to which random effect each Z matrix belongs to.
Ai	is a list with the inverses of the relationship matrix for each random effect.
y	is the response variable
S	is the list of residual matrices.
H	is the matrix of weights. This will be squared via the cholesky decomposition and apply to the residual matrices.
nIters	number of REML iterations .
tolParConvLL	rule for stoping the optimization problem, difference in log-likelihood between the current and past iteration.
tolParConvNorm	rule for stoping the optimization problem, difference in norms.
tolParInv	value to add to the diagonals of a matrix that cannot be inverted because is not positive-definite.
theta	is the initial values for the vc (matrices should be symmetric).
thetaC	is the constraints for vc: 1 positive, 2 unconstrained, 3 fixed.
thetaF	is the dataframe indicating the fixed constraints as x times another vc, rows indicate the variance components, columns the scale parameters (other VC plus additional ones preferred).
addScaleParam	any additional scale parameter to be included when applying constraints in thetaF.
weightInfEMv	is the vector to be put in a diagonal matrix (a list with as many matrices as iterations) representing the weight assigned to the EM information matrix.
weightInfMat	is a vector of weights to the information matrix for the operation delta = I- * dLu/dLx # unstructured models may require less weight to the information matrix.

Details

This algorithm is based on Jensen, Madsen and Thompson (1997)

Value

If all parameters are correctly indicated the program will return a list with the following information:

res a list of different outputs

References

Jensen, J., Mantysaari, E. A., Madsen, P., and Thompson, R. (1997). Residual maximum likelihood estimation of (co) variance components in multivariate mixed linear models using average information. Journal of the Indian Society of Agricultural Statistics, 49, 215-236.

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

data("DT_example")
DT <- DT_example
K <- A_example
#### look at the data and fit the model
head(DT)

zz <- with(DT, vsr(dsr(Env),Name))

Z <- c(list(model.matrix(~Name-1, data=DT)),zz$Z)

Zind <- c(1,2,2,2)

A <- list(diag(41), diag(41))#rep(list(diag(41)),4)

Ai <- lapply(A, function(x){solve(x)})

theta <- list(
  matrix(10,1,1),
  diag(10,3,3),
  diag(10,3,3)
);theta

thetaC <- list(
  matrix(1,1,1),
  diag(1,3,3),
  diag(1,3,3)
```

```
);thetaC

X <- model.matrix(~Env, data=DT)

y <- as.matrix(DT$Yield)

DTx <- DT; DTx$units <- as.factor(1:nrow(DTx))
ss <- with(DTx, vsr(dsr(Env),units) )

S <- ss$Z

H <- diag(length(y))

addScaleParam <- 0
nn <- unlist(lapply(thetaC, function(x){length(which(x > 0))}))
nn2 <- sum(nn[1:max(Zind)])
ff <- diag(nn2)
thetaF <- cbind(ff,matrix(0,nn2,1))

## apply the function
weightInfMat=rep(1,40); # weights for the information matrix
weightInfEMv=c(seq(.9,.1,-.1),rep(0,36)); # weights for the EM information matrix

# expr = res3<-AI(X=X,Z=Z, Zind=Zind,
#                 Ai=Ai,y=y,
#                 S=S,
#                 H=H,
#                 nIters=20, tolParConvLL=1e-5,
#                 tolParConvNorm=0.05,
#                 tolParInv=1e-6,theta=theta,
#                 thetaC=thetaC,thetaF=thetaF,
#                 addScaleParam=addScaleParam, weightInfEMv = weightInfEMv,
#                 weightInfMat = weightInfMat
#
# )
# # compare results
# res3$monitor
```

---

anova.mmec *anova form a GLMM fitted with mmec*

Description

anova method for class "mmec".

Usage

```
## S3 method for class 'mmec'
anova(object, object2=NULL, ...)
```

Arguments

object	an object of class "mmec"
object2	an object of class "mmec", if NULL the program will provide regular sum of squares results.
...	Further arguments to be passed

Value

vector of anova

Author(s)

Giovanny Covarrubias

See Also

anova, mmec

---

anova.mmer *anova form a GLMM fitted with mmer*

Description

anova method for class "mmer".

Usage

```
## S3 method for class 'mmer'
anova(object, object2=NULL, type=1, ...)
```

Arguments

object	an object of class "mmer"
object2	an object of class "mmer", if NULL the program will provide regular sum of squares results.
type	anova type, I or II
...	Further arguments to be passed

Value

vector of anova

Author(s)

Giovanny Covarrubias

See Also

anova, mmer

AR1 *Autocorrelation matrix of order 1.*

Description

Creates an autocorrelation matrix of order one with parameters specified.

Usage

```
AR1(x,rho=0.25)
```

Arguments

x	vector of the variable to define the factor levels for the AR1 covariance structure.
rho	rho value for the matrix.

Details

Specially useful for constructing covariance structures for rows and ranges to capture better the spatial variation trends in the field. The rho value is assumed fixed and values of the variance component will be optimized through REML.

Value

If everything is defined correctly the function returns:

$nn 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

See Also

The core function of the package mmer

Examples

```
x <- 1:4
R1 <- AR1(x,rho=.25)
image(R1)
```

ARMA *Autocorrelation Moving average.*

Description

Creates an ARMA matrix of order one with parameters specified.

Usage

```
ARMA(x, rho=0.25, lambda=0.25)
```

Arguments

x	vector of the variable to define the factor levels for the ARMA covariance structure.
rho	rho value for the matrix.
lambda	dimensions of the square matrix.

Details

Specially useful for constructing covariance structures for rows and ranges to capture better the spatial variation trends in the field. The rho value is assumed fixed and values of the variance component will be optimized through REML.

Value

If everything is defined correctly the function returns:

$nn 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

See Also

The core function of the package mmer

Examples

```
x <- 1:4
R1 <- ARMA(x,rho=.25,lambda=0.2)
image(R1)
```

atc *atc covariance structure*

Description

atc creates a diagonal covariance structure for specific levels of the random effect to be used with the mmec solver.

Usage

```
atc(x, levs, thetaC, theta)
```

Arguments

x	vector of observations for the random effect.
levs	levels of the random effect to use for building the incidence matrices.
thetaC	an optional symmetric matrix for constraints in the variance-covariance components. The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define how the variance-covariance components should be estimated: 0: component will not be estimated 1: component will be estimated and constrained to be positive 2: component will be estimated and unconstrained 3: component will be fixed to the value provided in the theta argument
theta	an optional symmetric matrix for initial values of the variance-covariance components. The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define the initial values of the variance-covariance components that will be subject to the constraints provided in thetaC. If not provided, initial values will be calculated as: theta* = diag(ncol(mm))*.05 + matrix(.1,ncol(mm),ncol(mm)) where mm is the incidence matrix for the factor 'x'. The values provided should be scaled by the variance of the response variable. theta = theta*/var(y)

Value

$res a list with the provided vector and the variance covariance structure expected.

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

See Also

The function vsc to know how to use atc in the mmec solver.

Examples

```
x <- as.factor(c(1:5,1:5,1:5));x
atc(x, c("1","2"))
## how to use the theta and thetaC arguments:
# data(DT_example)
# DT <- DT_example
# theta <- diag(2)*2; theta # initial VCs
# thetaC <- diag(2)*3; thetaC # fixed VCs
# ans1 <- mmec(Yield~Env,
#              random= ~ vsc( atc(Env, levs=c("CA.2013", "CA.2011"),
#                                 theta = theta,thetaC = thetaC),isc(Name) ),
#              rcov= ~ units, nIters = 1,
#              data=DT)
# summary(ans1)$varcomp
```

---

atcg1234 *Letter to number converter*

Description

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

Usage

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

Arguments

data	a dataframe with markers in columns and individuals in rows. Preferable the rownames are the ID of the plants so you don't lose track of what is what.
ploidy	a numeric value indicating the ploidy level of the specie. The default is 2 which means diploid.
format	one of the two possible values allowed by the program "ATCG", which means your calls are in base-pair-letter code, i.e. "AT" in a diploid call, "AATT" tetraploid etc (just example). Therefore possible codes can be "A", "T", "C", "G", "-" (deletion), "+" (insertion). Alternatively "AB" format can be used as

well. Commonly this depends from the genotyping technologies used, such as GBS or microarrays. In addition, we have enabled also the use of single-letter code used by Cornell, i.e. A=AA, C=CC, T=TT, G=GG, R=AG, Y=CT, S=CG, W=AT, K=GT, M=AC. In the case of GBS code please make sure that you set the N codes to regular NAs handled by R. The "ATCG" format also works for the bi-allelic marker codes from join map such as "lm", "ll","nn", "np","hh","hk","kk"

`maf` minor allele frequency used to filter the SNP markers, the default is zero which means all markers are returned in numeric format.

`multi` a TRUE/FALSE statement indicating if the function should get rid of the markers with more than 2 alleles. If FALSE, which indicates that if markers with multiple alleles are found, the alternate and reference alleles will be the first 2 alleles found. This could be risky since some alleles will be masked, i.e. AA AG AT would take only A and G as reference and alternate alleles, converting to numeric format 2 1 1, giving the same effect to AG and AT which could be a wrong assumption. The default is TRUE, removes markers with more than two alleles.

`silent` a TRUE/FALSE value indicating if a progress bar should be drawn for each step of the conversion. The default is silent=FALSE, which means that we want progress bar to be drawn.

`by.allele` a TRUE/FALSE value indicating if the program should transform the data in a zero/one matrix of presence/absense per allele. For example, a marker with 3 alleles A,T,C in a diploid organism will yield 6 possible configurations; AA, AT, AC, TT, TC, CC. Therefore, the program would create 3 columns for this marker indicating the presence/absence of each allele for each genotype.

`imp` a TRUE/FALSE value indicating if the function should impute the missing data using the median for each marker. If FALSE, then the program will not impute.

`ref.alleles` a matrix with reference alleles to be used for the conversion. The matrix should have as many columns as markers with reference alleles and with 2 rows, being the first row the alternate allele (Alt) and the second row the reference allele (Ref). Rownames should be "Alt" and "Ref" respectively. If not provided the program will decide the reference allele.

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

See Also

The core function of the package `mmer`

Examples

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

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

####================================================####
#### convert markers to numeric format diploid rice lines
#### single letter code for inbred lines from GBS pipeline
#### A=AA, T=TT, C=CC, G=GG
####================================================####
# data(DT_rice)
# X <- GT_rice; X[1:5,1:5]; dim(X)
# numo2 <- atcg1234(data=X, ploidy=2)
# numo2$M[1:5,1:5]
```

---

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

See Also

The core function of the package mmer

Examples

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

# ####================================================####
# #### convert markers to numeric format polyploid potatoes
# ####================================================####
# numo <- atcg1234(data=genotypes, ploidy=4)
# numo$M[1:5,1:5]
# numob <- atcg1234BackTransform(marks =  numo$M, refs =  numo$ref.alleles)
# numob[1:4,1:4]
#
# ####================================================####
# #### convert markers to numeric format diploid rice lines
# #### single letter code for inbred lines from GBS pipeline
# #### A=AA, T=TT, C=CC, G=GG
# ####================================================####
# data(DT_rice)
# X <- GT_rice; X[1:5,1:5]; dim(X)
# numo2 <- atcg1234(data=X, ploidy=2)
# numo2$M[1:5,1:5]
# Xb <- atcg1234BackTransform(marks= numo2$M, refs= numo2$ref.alleles)
# Xb[1:4,1:4]
```

---

atr *atr covariance structure*

Description

`atr` creates a diagonal covariance structure for specific levels of the random effect to be used with the mmer solver.

Usage

```
atr(x, levs)
```

Arguments

`x`	vector of observations for the random effect.
`levs`	levels of the random effect to use for building the incidence matrices.

Value

$res a list with the provided vector and the variance covariance structure expected.

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

See Also

The function `vsr` to know how to use `atr` in the `mmer` solver.

Examples

```
x <- as.factor(c(1:5,1:5,1:5));x
atr(x)
atr(x, c("1","2"))
```

`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.

bivariateRun *bivariateRun functionality*

Description

Sometimes multi-trait models can present many singularities making the model hard to estimate with many traits. One of the most effective strategies is to estimate all possible variance and covariances splitting in multiple bivariate models. This function takes a model that has t traits and splits the model in as many bivariate models as needed to estimate all the variance and covariances to provide the initial values for the model with all traits.

Usage

```
bivariateRun(model, n.core)
```

Arguments

model	a model fitted with the mmer function with argument return.param=TRUE.
n.core	number of cores to use in the mclapply function to parallelize the models to be run to avoid increase in computational time. Please keep in mind that this is only available in Linux and macOS systems. Please check the details in the mclapply documentation of the parallel package.

Value

$sigmas the list with the variance covariance parameters for all traits together.

$sigmascor the list with the correlation for the variance components for all traits together.

$model the results from the bivariate models.

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

See Also

The core function of the package mmer

Examples

```
# ####==========================================####
# ####==========================================####
# #### EXAMPLE 1
# #### simple example with univariate models
# ####==========================================####
# ####==========================================####
# data("DT_cpdata")
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT)
# #### look at the data and fit the model
# head(DT)
# ans.m <- mmer(cbind(Yield,color,FruitAver, Firmness)~1,
#               random=~ vsr(id, Gu=A, Gtc=unsm(4))
#               + vsr(Rowf,Gtc=diag(4))
#               + vsr(Colf,Gtc=diag(4)), na.method.Y="include",
#               rcov=~ vsr(units,Gtc=unsm(4)), return.param = TRUE,
#               data=DT)
#
# # define the number of cores (number of bivariate models) as (nt*(nt-1))/2
```

```
# nt=4
# (nt*(nt-1))/2
# res <- bivariateRun(ans.m,n.core = 6)
# # now use the variance componets to fit a join model
# mm <- transformConstraints(ans.m[[8]],3)
#
# ans.m.final <- mmer(cbind(Yield,color,FruitAver, Firmness)~1,
#               random=~ vsr(id, Gu=A, Gtc=unsm(4))
#               + vsr(Rowf,Gtc=diag(4))
#               + vsr(Colf,Gtc=diag(4)), na.method.Y="include",
#               rcov=~ vsr(units,Gtc=unsm(4)),
#               init = res$sigmas_scaled, constraints = mm,
#               data=DT, iters=1)
#
# summary(ans.m.final)
```

`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=":")
```

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.

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

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

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

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

See Also

mmer– the core function of the package

Examples

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


## first get all possible hybrids
res1 <- build.HMM(Md, Mf,
                  return.combos.only = TRUE)
head(res1$data.used)

## build the marker matrix for the first 50 hybrids
```

```
res2 <- build.HMM(Md, Mf,
                  custom.hyb = res1$data.used[1:50,]
                  )
res2$HMM.add[1:5,1:5]
res2$HMM.dom[1:5,1:5]

## now you can use the A.mat(), D.mat() and E.mat() functions
# M <- res2$HMM.add
# A <- A.mat(M)
# D <- D.mat(M)
```

`coef.mmec` *coef form a GLMM fitted with mmec*

Description

`coef` method for class "`mmec`".

Usage

```
## S3 method for class 'mmec'
coef(object, ...)
```

Arguments

`object`	an object of class "`mmec`"
`...`	Further arguments to be passed

Value

vector of coef

Author(s)

Giovanny Covarrubias

See Also

`coef`, `mmec`

coef.mmer *coef form a GLMM fitted with mmer*

Description

coef method for class "mmer".

Usage

```
## S3 method for class 'mmer'
coef(object, ...)
```

Arguments

object	an object of class "mmer"
...	Further arguments to be passed

Value

vector of coef

Author(s)

Giovanny Covarrubias

See Also

coef, mmer

corImputation *Imputing a matrix using correlations*

Description

corImputation imputes missing data based on the correlation that exists between row levels.

Usage

```
corImputation(wide, Gu=NULL, nearest=10, roundR=FALSE)
```

Arguments

wide	numeric matrix with individuals in rows and time variable in columns (e.g., environments, genetic markers, etc.).
Gu	optional correlation matrix between the individuals or row levels. If NULL it will be computed as the correlation of t(wide).
nearest	integer value describing how many nearest neighbours (the ones showing the highest correlation) should be used to average and return the imputed value.
roundR	a TRUE/FALSE statement describing if the average result should be rounded or not. This may be specifically useful for categorical data in the form of numbers (e.g., -1,0,1).

Value

$res a list with the imputed matrix and the original matrix.

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

See Also

The core functions of the package mmer and mmec

Examples

```
###################################
### imputing genotype data example
###################################
# data(DT_cpdata)
# X <- GT_cpdata
# # add missing data
# v <- sample(1:length(X), 500)
# Xna <- X
# Xna[v]<- NA
# ## impute (can take some time)
# Y <- corImputation(wide=Xna, Gu=NULL, nearest=20, roundR=TRUE)
# cm <- table(Y$imputed[v],X[v])
# ## calculate accuracy
# sum(diag(cm))/length(v)
###################################
### imputing phenotypic data example
###################################
# data(DT_h2)
# X <- reshape(DT_h2[,c("Name","Env","y")], direction = "wide", idvar = "Name",
#                 timevar = "Env", v.names = "y", sep= "_")
```

```
# rownames(X) <- X$Name
# X <- as.matrix(X[,-1])
# head(X)
# # add missing data
# v <- sample(1:length(X), 50)
# Xna <- X
# Xna[v]<- NA
# ## impute
# Y <- corImputation(wide=Xna, Gu=NULL, nearest=20, roundR=TRUE)
# plot(y=Y$imputed[v],x=X[v], xlab="true",ylab="predicted")
# cor(Y$imputed[v],X[v], use = "complete.obs")
```

---

covc *covariance between random effects*

Description

covc merges the incidence matrices and covariance matrices of two random effects to fit an unstructured model between 2 different random effects to be fitted with the mmec solver.

Usage

```
covc(ran1, ran2, thetaC=NULL, theta=NULL)
```

Arguments

ran1	the random call of the first random effect.
ran2	the random call of the first random effect.
thetaC	an optional matrix for constraints in the variance components.
theta	an optional symmetric matrix for initial values of the variance-covariance components. When providing customized values, these values should be scaled with respect to the original variance. For example, to provide an initial value of 1 to a given variance component, theta would be built as: theta = matrix(1 / var(response)) The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define the initial values of the variance-covariance components that will be subject to the constraints provided in thetaC. If not provided, initial values will be calculated as: theta = diag(ncol(mm))*.05 + matrix(.1,ncol(mm),ncol(mm)) where mm is the incidence matrix for the factor 'x'.

Details

This implementation aims to fit models where covariance between random variables is expected to exist. For example, indirect genetic effects.

Value

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

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

Bijma, P. (2014). The quantitative genetics of indirect genetic effects: a selective review of modelling issues. Heredity, 112(1), 61-69.

See Also

The function `vsc` to know how to use `covc` in the `mmec` solver.

Examples

```
data(DT_ige)
DT <- DT_ige
covRes <- with(DT, covc( vsc(isc(focal)) , vsc(isc(neighbour)) ) )
str(covRes)
# look at DT_ige help page to see how to fit an actual model
```

CS *Compound symmetry matrix*

Description

Creates a compound symmetry matrix with parameters specified.

Usage

```
CS(x, rho=0.25)
```

Arguments

x	vector of the variable to define the factor levels for the ARMA covariance structure.
rho	rho value for the matrix.

Details

Specially useful for constructing covariance structures for rows and ranges to capture better the spatial variation trends in the field. The rho value is assumed fixed and values of the variance component will be optimized through REML.

Value

If everything is defined correctly the function returns:

$nn 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

See Also

The core function of the package mmer

Examples

```
x <- 1:4
R1 <- CS(x,rho=.25)
image(R1)
```

---

csc *customized covariance structure*

Description

csc creates a customized covariance structure for specific levels of the random effect to be used with the mmec solver.

Usage

```
csc(x, mm, thetaC, theta)
```

Arguments

x	vector of observations for the random effect.
mm	customized variance-covariance structure for the levels of the random effect.
thetaC	an optional symmetric matrix for constraints in the variance-covariance components. The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define how the variance-covariance components should be estimated: 0: component will not be estimated

1: component will be estimated and constrained to be positive

2: component will be estimated and unconstrained

3: component will be fixed to the value provided in the theta argument

`theta` an optional symmetric matrix for initial values of the variance-covariance components. When providing customized values, these values should be scaled with respect to the original variance. For example, to provide an initial value of 1 to a given variance component, theta would be built as:

theta = matrix(1 / var(response))

The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define the initial values of the variance-covariance components that will be subject to the constraints provided in thetaC. If not provided, initial values will be calculated as:

theta = diag(ncol(mm))*.05 + matrix(.1,ncol(mm),ncol(mm))

where mm is the incidence matrix for the factor 'x'.

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

The function `vsc` to know how to use `csc` in the `mmec` solver.

Examples

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

---

csr *customized covariance structure*

Description

`csr` creates a customized covariance structure for specific levels of the random effect to be used with the `mmer` solver.

Usage

```
csr(x, mm)
```

Arguments

x	vector of observations for the random effect.
mm	customized variance-covariance structure for the levels of the random effect.

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

The function vsr to know how to use csr in the mmer solver.

Examples

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

D.mat *Dominance relationship matrix*

Description

C++ implementation of the dominance matrix. Calculates the realized dominance relationship matrix. Can help to increase the prediction accuracy when 2 conditions are met; 1) The trait has intermediate to high heritability, 2) The population contains a big number of individuals that are half or full sibs (HS & FS).

Usage

```
D.mat(X,nishio=TRUE,min.MAF=0,return.imputed=FALSE)
```

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.
nishio	If TRUE Nishio ans Satoh. (2014), otherwise Su et al. (2012). See references.
min.MAF	Minimum minor allele frequency. The D matrix is not sensitive to rare alleles, so by default only monomorphic markers are removed.
return.imputed	When TRUE, the imputed marker matrix is returned.

Details

The additive marker coefficients will be used to compute dominance coefficients as: Xd = 1-abs(X) for diploids.

For nishio method: the marker matrix is centered by subtracting column means $M = Xd - ms$ where ms is the column means. Then $A = MM'/c$, where $c = 2\sum_k p_k(1-p_k)$.

For su method: the marker matrix is normalized by subtracting row means $M = Xd - 2pq$ where 2pq is the product of allele frequencies times 2. Then $A = MM'/c$, where $c = 2\sum_k 2pq_k(1-2pq_k)$.

Value

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

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

$D the D matrix

$imputed the imputed marker 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

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

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

See Also

The core functions of the package mmer

Examples

```
####=========================================####
#### EXAMPLE 1
####=========================================####
####random population of 200 lines with 1000 markers
X <- matrix(rep(0,200*1000),200,1000)
for (i in 1:200) {
```

```
  X[i,] <- sample(c(-1,0,0,1), size=1000, replace=TRUE)
}

D <- D.mat(X)
```

---

dfToMatrix *data frame to matrix*

Description

This function takes a matrix that is in data frame format and transforms it into a matrix. Other packages that allows you to obtain an additive relationship matrix from a pedigree is the 'pedigreemm' package.

Usage

```
dfToMatrix(x, row="Row",column="Column",
           value="Ainverse", returnInverse=FALSE,
           bend=1e-6)
```

Arguments

x	ginv element, output from the Ainverse function.
row	name of the column in x that indicates the row in the original relationship matrix.
column	name of the column in x that indicates the column in the original relationship matrix.
value	name of the column in x that indicates the value for a given row and column in the original relationship matrix.
returnInverse	a TRUE/FALSE value indicating if the inverse of the x matrix should be computed once the data frame x is converted into a matrix.
bend	a numeric value to add to the diagonal matrix in case matrix is singular for inversion.

Value

K	pedigree transformed in a relationship matrix.
Kinv	inverse of the pedigree transformed in a relationship matrix.

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

See Also

The core functions of the package mmer

Examples

```
library(Matrix)
m <- matrix(1:9,3,3)
m <- tcrossprod(m)

mdf <- as.data.frame(as.table(m))
mdf

dfToMatrix(mdf, row = "Var1", column = "Var2",
            value = "Freq",returnInverse=FALSE )
```

---

dsc *diagonal covariance structure*

Description

dsc creates a diagonal covariance structure for the levels of the random effect to be used with the mmec solver.

Usage

```
dsc(x, thetaC=NULL, theta=NULL)
```

Arguments

x	vector of observations for the random effect.
thetaC	an optional symmetric matrix for constraints in the variance-covariance components. The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define how the variance-covariance components should be estimated: 0: component will not be estimated 1: component will be estimated and constrained to be positive 2: component will be estimated and unconstrained 3: component will be fixed to the value provided in the theta argument
theta	an optional symmetric matrix for initial values of the variance-covariance components. When providing customized values, these values should be scaled with respect to the original variance. For example, to provide an initial value of 1 to a given variance component, theta would be built as: theta = matrix(1 / var(response))

The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define the initial values of the variance-covariance components that will be subject to the constraints provided in thetaC. If not provided, initial values will be calculated as:

diag(ncol(mm))*.05 + matrix(.1,ncol(mm),ncol(mm))

where mm is the incidence matrix for the factor 'x'.

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

See the function `vsc` to know how to use `dsc` in the `mmec` solver.

Examples

```
x <- as.factor(c(1:5,1:5,1:5));x
dsc(x)
## how to use the theta and thetaC arguments:
# data(DT_example)
# DT <- DT_example
# theta <- diag(3)*2; theta # initial VCs
# thetaC <- diag(3)*3; thetaC # fixed VCs
# ans1 <- mmec(Yield~Env,
#              random= ~ vsc( dsc(Env,theta = theta,thetaC = thetaC),isc(Name) ),
#              rcov= ~ units,
#              data=DT)
# summary(ans1)$varcomp
```

---

dsr *diagonal covariance structure*

Description

`dsr` creates a diagonal covariance structure for the levels of the random effect to be used with the `mmer` solver.

Usage

```
dsr(x)
```

Arguments

x	vector of observations for the random effect.

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

See the function `vsr` to know how to use `dsr` in the `mmer` solver.

Examples

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

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.

The column 'Plot' indicates the number of plot in the field The column 'Entry' is a numeric value for each entry The colum 'Genotype' is the name of the individual The column 'Block' is the replicate or big block The column TSW is the response variable The column check is an indicator column for checks (0) and non-checks (1) The column Check.Gen is an indicator column for checks (89,90,91) and non-checks (999)

The dataset has 3 unique checks (Ross=89, MF183=90, Starlight=91) and 50 entries.

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

See Also

The core functions of the package `mmer` and `mmec`

Examples

```
# ###=========================================####
# ### AUGMENTED DESIGN EXAMPLE
# ###=========================================####
# data(DT_augment)
# DT <- DT_augment
# head(DT)
# ####=========================================####
# #### fit the mixed model and check summary
# ####=========================================####
# mix1 <- mmer(TSW ~ Check.Gen,
#              random = ~ Block + Genotype:Check,
#              data=DT)
# summary(mix1)$varcomp
#
# mix2 <- mmec(TSW ~ Check.Gen,
#              random = ~ Block + Genotype:Check,
#              data=DT)
# summary(mix2)$varcomp
```

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

See Also

The core functions of the package `mmer`, `mmec`

Examples

```
# ####========================================####
# #### For CRAN time limitations most lines in the
# #### examples are silenced with one '#' mark,
# #### remove them and run the examples
# ####========================================####
# ####========================================####
# ####========================================####
# #### EXAMPLE 1
# #### simple example
# ####========================================####
# ####========================================####
# data(DT_btdata)
# DT <- DT_btdata
# head(DT)
# mix4 <- mmer(tarsus ~ sex,
#              random = ~ dam + fosternest,
#              rcov=~units,
#              data = DT)
# summary(mix4)$varcomp
#
# mix5 <- mmec(tarsus ~ sex,
#              random = ~ dam + fosternest,
#              rcov=~units,
#              data = DT)
# summary(mix5)$varcomp
#
```

```
# ####========================================####
# ####========================================####
# ####========================================####
# #### EXAMPLE 2
# #### more complex multivariate model
# ####========================================####
# ####========================================####
# data(DT_btdata)
# DT <- DT_btdata
# mix3 <- mmer(cbind(tarsus, back) ~ sex,
#                 random = ~ vsr(dam) + vsr(fosternest),
#                 rcov= ~ vsr(units, Gtc=diag(2)),
#                 data = DT)
# summary(mix3)
# #### calculate the genetic correlation
# cov2cor(mix3$sigma$`u:dam`)
# cov2cor(mix3$sigma$`u:fosternest`)
```

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

See Also

The core functions of the package mmer and mmec

Examples

```
# ####========================================####
# #### For CRAN time limitations most lines in the
# #### examples are silenced with one '#' mark,
# #### remove them and run the examples using
# #### command + shift + C |OR| control + shift + C
# ####========================================####
#
# data(DT_cornhybrids)
# DT <- DT_cornhybrids
# DTi <- DTi_cornhybrids
# GT <- GT_cornhybrids
# hybrid2 <- DT # extract cross data
# A <- GT
# K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
# K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
# S <- kronecker(K1, K2) ; dim(S)
# rownames(S) <- colnames(S) <- levels(hybrid2$SCA)
#
# ans <- mmer(Yield ~ Location,
#              random = ~ vsr(GCA1,Gu=K1) + vsr(GCA2,Gu=K2), # + vsr(SCA,Gu=S),
#              rcov=~units,
#              data=hybrid2)
# summary(ans)$varcomp
#
# ## mmec uses the inverse of the relationship matrix
# K1i <- as(solve(K1 + diag(1e-4,ncol(K1),ncol(K1))), Class="dgCMatrix")
# K2i <- as(solve(K2 + diag(1e-4,ncol(K2),ncol(K2))), Class="dgCMatrix")
# Si <- as(solve(S + diag(1e-4,ncol(S),ncol(S))), Class="dgCMatrix")
# ans2 <- mmec(Yield ~ Location,
#         random = ~ vsc(isc(GCA1),Gu=K1i) + vsc(isc(GCA2),Gu=K2i), # + vsc(isc(SCA),Gu=Si),
#             rcov=~units,
#             data=hybrid2)
# summary(ans2)$varcomp
```

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

See Also

The core functions of the package `mmer` and `mmec`

Examples

```
# ####========================================####
# #### For CRAN time limitations most lines in the
# #### examples are silenced with one '#' mark,
# #### remove them and run the examples using
# #### command + shift + C |OR| control + shift + C
# ####========================================####
#
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT) # additive relationship matrix
# #### look at the data and fit the model
# head(DT)
# mix1 <- mmer(Yield~1,
#               random=~vsr(id,Gu=A)
#                       + Rowf + Colf,
```

```
#               rcov=~units,
#               data=DT)
# summary(mix1)$varcomp
#
# ## mmec uses the inverse of the relationship matrix
# Ai <- as(solve(A + diag(1e-4,ncol(A),ncol(A))), Class="dgCMatrix")
# mix2 <- mmec(Yield~1,
#              random=~vsc(isc(id),Gu=Ai)
#              + Rowf + Colf,
#              rcov=~units,
#              data=DT)
# summary(mix2)$varcomp
#
# vg <- summary(mix2)$varcomp[1,1] # genetic variance
# G <- A*vg # genetic variance-covariance
# Ci <- mix2$Ci # coefficient matrix
# ind <- as.vector(mix2$partitions$`vsc(isc(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       ####
# ####====================####
# #### be patient take some time
# ans.m <- mmer(cbind(Yield,color)~1,
#               random=~ vsr(id, Gu=A)
#               + vsr(Rowf,Gtc=diag(2))
#               + vsr(Colf,Gtc=diag(2)),
#               rcov=~ vsr(units),
#               data=DT)
# cov2cor(ans.m$sigma$`u:id`)
```

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

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
####============================================####
#### EXAMPLES
#### Different models with sommer
####============================================####

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

####============================================####
#### Univariate homogeneous variance models  ####
####============================================####

## Compound simmetry (CS) model
ans1 <- mmer(Yield~Env,
             random= ~ Name + Env:Name,
             rcov= ~ units,
             data=DT)
summary(ans1)$varcomp

ans1b <- mmec(Yield~Env,
             random= ~ Name + Env:Name,
             rcov= ~ units,
             data=DT)
summary(ans1b)$varcomp

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

```
# ####============================================####
#
# ## Compound simmetry (CS) + Diagonal (DIAG) model
# ans2 <- mmer(Yield~Env,
#              random= ~Name + vsr(dsr(Env),Name),
#              rcov= ~ vsr(dsr(Env),units),
#              data=DT)
# summary(ans2)
#
# DT=DT[with(DT, order(Env)), ]
# ans2b <- mmec(Yield~Env,
#              random= ~Name + vsc(dsc(Env),isc(Name)) +
#                       vsc(atc(Env, c("CA.2011") ),isc(Block)) ,
#              rcov= ~ vsc(dsc(Env),isc(units)),
#              data=DT)
# summary(ans2b)
#
# ####============================================####
# #### Multivariate homogeneous variance models ####
# ####============================================####
#
# ## Multivariate Compound simmetry (CS) model
# DT$EnvName <- paste(DT$Env,DT$Name)
# ans4 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vsr(Name) + vsr(EnvName),
#               rcov= ~ vsr(units),
#               data=DT)
# summary(ans4)
```

DT_expdesigns *Data for different experimental designs*

Description

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

1) Augmented designs (2 examples)

2) Incomplete block designs (1 example)

3) Split plot design (2 examples)

4) Latin square designs (1 example)

5) North Carolina designs I,II and III

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

Format

Different based on the design.

Source

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

Examples

```
# #### ================================== ####
# #### ===== Augmented Block Design 1 ==== ####
# #### ================================== ####
# data(DT_expdesigns)
# DT <- DT_expdesigns
# names(DT)
# data1 <- DT$au1
# head(data1)
# ## response variable: "yield"
# ## check indicator: "entryc" ('nc' for all unreplicated, but personal.name for checks)
# ## blocking factor: "block"
# ## treatments, personal names for replicated and non-replicated: "trt"
# ## check no check indicator: "new"
# mix1 <- mmer(yield~entryc,
#              random=~block+trt,
#              rcov=~units, tolpar = 1e-6,
#              data=data1)
# summary(mix1)$varcomp
#
# mix1b <- mmec(yield~entryc,
#              random=~block+trt,
#              rcov=~units, tolParConv = 1e-6,
#              data=data1)
# summary(mix1b)$varcomp
```

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

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(DT_fulldiallel)
DT <- DT_fulldiallel
head(DT)
mix <- mmer(stems~1, random=~female+male, data=DT)
summary(mix)

mixb <- mmec(stems~1, random=~female+male, data=DT)
summary(mixb)$varcomp

####============================================####
####============================================####
#### Multivariate model example
####============================================####
####============================================####

data(DT_fulldiallel)
DT <- DT_fulldiallel
head(DT)

mix <- mmer(cbind(stems,pods,seeds)~1,
            random=~vsr(female) + vsr(male),
            rcov=~vsr(units),
            data=DT)
summary(mix)
#### genetic variance covariance
cov2cor(mix$sigma$`u:female`)
cov2cor(mix$sigma$`u:male`)
cov2cor(mix$sigma$`u:units`)
```

DT_gryphon *Gryphon data from the Journal of Animal Ecology*

Description

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

The dataset contains 3 elements:

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

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

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

Usage

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

Format

The format is: chr "DT_gryphon"

Source

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

References

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

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

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####
```

```
# data(DT_gryphon)
# DT <- DT_gryphon
# A <- A_gryphon
# P <- P_gryphon
# #### look at the data
# head(DT)
# #### fit the model with no fixed effects (intercept only)
# mix1 <- mmer(BWT~1,
#              random=~vsr(ANIMAL,Gu=A),
#              rcov=~units,
#              data=DT)
# summary(mix1)$varcomp
#
# ## mmec uses the inverse of the relationship matrix
# Ai <- as(solve(A + diag(1e-4,ncol(A),ncol(A))), Class="dgCMatrix")
# mix1b <- mmec(BWT~1,
#              random=~vsc(isc(ANIMAL),Gu=Ai),
#              rcov=~units, tolParConv = 1e-5,
#              data=DT)
# summary(mix1b)$varcomp
#
# #### fit the multivariate model with no fixed effects (intercept only)
# mix2 <- mmer(cbind(BWT,TARSUS)~1,
#              random=~vsr(ANIMAL,Gu=A),
#              rcov=~vsr(units),
#              na.method.Y = "include2",
#              data=DT)
# summary(mix2)
# cov2cor(mix2$sigma$`u:ANIMAL`)
# cov2cor(mix2$sigma$`u:units`)
```

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

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(DT_h2)
DT <- DT_h2
head(DT)
####============================================####
#### fit the mixed model (very heavy model)
####============================================####
# ans1 <- mmer(y~Env,
#               random=~vsr(dsr(Env),Name) + vsr(dsr(Env),Block),
#               rcov=~vsr(dsr(Env),units),
#               data=DT)
# summary(ans1)$varcomp
#
# DT=DT[with(DT, order(Env)), ]
# ans1b <- mmec(y~Env,
#              random=~vsc(dsc(Env),isc(Name)) + vsc(dsc(Env),isc(Block)),
#              rcov=~vsc(dsc(Env),isc(units)),
#              data=DT)
# summary(ans1b)$varcomp
```

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

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

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

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

Ai <- as(solve(A + diag(1e-4,ncol(A),ncol(A))), Class="dgCMatrix")
modhb <- mmec(sugar~1,
             random=~vsc(isc(overlay(femalef,malef, sparse = TRUE)), Gu=Ai)
             + genof,
             data=DT)
summary(modhb)$varcomp
```

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

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

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
####============================================####
#### EXAMPLES
#### Different models with sommer
####============================================####

data(DT_ige)
DT <- DT_ige
# # Indirect genetic effects model without covariance between DGE and IGE
# modIGE <- mmec(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 <- mmec(trait ~ block, dateWarning = FALSE,
#                random = ~ covc( vsc(isc(focal)), vsc(isc(neighbour)) ),
#                rcov = ~ units, nIters=100,
#               data = DT)
# summary(modIGE)$varcomp
# pmonitor(modIGE)
#
# # form relationship matrix
# Ai <- as( solve(A_ige + diag(1e-5, nrow(A_ige),nrow(A_ige) )), Class="dgCMatrix")
# # Indirect genetic effects model with covariance between DGE and IGE using relationship matrices
# modIGE <- mmec(trait ~ block, dateWarning = FALSE,
#                random = ~ covc( vsc(isc(focal), Gu=Ai), vsc(isc(neighbour), Gu=Ai) ),
#                rcov = ~ units, nIters=100,
#               data = DT)
# summary(modIGE)$varcomp
# pmonitor(modIGE)
```

`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 mmer 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

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

See Also

The core functions of the package `mmer`

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####=========================================####
# you need to install the orthopolynom library to do random regression models
# library(orthopolynom)
# data(DT_legendre)
# DT <- DT_legendre
# mRR2<-mmer(Y~ 1 + Xf
#            , random=~ vsr(usr(leg(X,1)),SUBJECT)
#            , rcov=~vsr(units)
#            , data=DT)
# summary(mRR2)$varcomp
#
# mRR2b<-mmec(Y~ 1 + Xf
#            , random=~ vsc(usc(leg(X,1)),isc(SUBJECT))
#            , rcov=~vsc(isc(units))
#            , data=DT)
# summary(mRR2b)$varcomp
```

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

See Also

The core functions of the package mmer and mmec

Examples

```
# ####========================================####
# #### For CRAN time limitations most lines in the
# #### examples are silenced with one '#' mark,
# #### remove them and run the examples
# ####========================================####
# data(DT_mohring)
# DT <- DT_mohring
# head(DT)
# DT2 <- add.diallel.vars(DT,par1="Par1", par2="Par2")
# head(DT2)
# # is.cross denotes a hybrid (1)
# # is.self denotes an inbred (1)
# # cross.type denotes one way (-1, e.g. AxB) and reciprocal (1, e.g., BxA) and no cross (0)
# # cross.id denotes the name of the cross (same name for direct & reciprocal)
#
# # GRIFFING MODEL 2 with reciprocal effects ############################
# mod1h <- mmer(Ftime ~ 1, data=DT2,
#               random = ~ Block
#               # GCA male & female overlayed
#               + 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)
#               + cross.id:cross.type)
# summary(mod1h)$varcomp
#
# mod1hb <- mmec(Ftime ~ 1, data=DT2,
#                random = ~ Block
#                # GCA male & female overlayed
#                + vsc(isc(overlay(Par1, Par2)))
#                # SCA effects (includes cross and selfs)
#                + cross.id
#                # SCAR reciprocal effects (remaining variance in crosses;
#                # if zero there's no reciprocal effects)
#                + vsc(dsc(cross.type), isc(cross.id)) )
# summary(mod1hb)$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 <- mmer(Ftime ~ Block + is.cross, data=DT2,
```

```
#               random = ~
#                # GCA effect for all (hybrids and inbreds)
#                 overlay(Par1, Par2)
#               # GCA effect (calculated only in hybrids; remaining variance)
#               + overlay(Par1, Par2):is.cross
#               # SCA effect (calculated in hybrids only)
#               + cross.id:is.cross)
# summary(mod1h)$varcomp
#
# mod1h <- mmec(Ftime ~ Block + is.cross, data=DT2, nIters = 50,
#               random = ~
#                 # GCA effects for all (hybrids and inbreds)
#                 vsc(isc(overlay(Par1, Par2)))
#               # GCA effect (calculated only in hybrids; remaining variance)
#               + vsc(isc(is.cross),isc(overlay(Par1, Par2)))
#               # SCA effect (calculated in hybrids only)
#               + vsc(isc(is.cross), isc(cross.id))
#               )
# summary(mod1h)$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 <- mmer(Ftime ~ Block + is.cross, data=DT2,
#               random = ~
#                 # GCAC (only for hybrids)
#                 overlay(Par1, Par2):is.cross
#               # male GCA (only for inbreds)
#               + Par1:is.self
#               # SCA (for hybrids only)
#               + cross.id:is.cross)
# summary(mod1h)$varcomp
#
# mod1h <- mmec(Ftime ~ Block + is.cross, data=DT2, nIters = 100,
#               random = ~
#                 # GCAC (only for hybrids)
#                 vsc(isc(is.cross),isc(overlay(Par1, Par2)))
#               # male GCA (only for inbreds)
#               + vsc(isc(is.self),isc(Par1))
#               # SCA (for hybrids only)
#               + vsc(isc(is.cross), isc(cross.id))
#               )
# summary(mod1h)$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 <- mmer(Ftime ~ Block + is.cross, data=DT2,
#               random = ~
#                 # GCAC (for hybrids only)
#                 overlay(Par1, Par2):is.cross
#               # male GCA (for selfs only)
#               + Par1:is.self
#               # SCA (for hybrids only)
#               + cross.id:is.cross
#               # SCAR reciprocal effects (remaning SCA variance)
#               + cross.id:cross.type)
# summary(mod1h)$varcomp
#
# mod1h <- mmec(Ftime ~ Block + is.cross, data=DT2, nIters = 100,
#               random = ~
#                 # GCAC (for hybrids only)
#                 vsc(isc(is.cross),isc(overlay(Par1, Par2)))
#               # male GCA (for selfs only)
#               + vsc(isc(is.self),isc(Par1))
#               # SCA (for hybrids only)
#               + vsc(isc(is.cross), isc(cross.id))
#               # SCAR reciprocal effects (remaning SCA variance)
#               + vsc(isc(cross.type), isc(cross.id))
#               )
# summary(mod1h)$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 <- mmer(Ftime ~ Block + is.cross, data=DT2,
#               random = ~
#                 # GCAC (for hybrids only)
#                 overlay(Par1,Par2):is.cross
#               # RGCA: exclude selfs (to identify reciprocal GCA effects)
#               + overlay(Par1,Par2):cross.type
#               # male GCA (for selfs only)
#               + Par1:is.self
#               # SCA (for hybrids only)
#               + cross.id:is.cross
#               # SCAR: exclude selfs (if zero there's no reciprocal SCA effects)
#               + cross.id:cross.type)
# summary(mod1h)$varcomp
#
# mod1h <- mmec(Ftime ~ Block + is.cross, data=DT2,nIters = 100,
#               random = ~
#                 # GCAC (for hybrids only)
```

```
#                 vsc(isc(is.cross),isc(overlay(Par1, Par2)))
#               # RGCA: exclude selfs (to identify reciprocal GCA effects)
#               + vsc(isc(cross.type),isc(overlay(Par1, Par2)))
#               # male GCA (for selfs only)
#               + vsc(isc(is.self),isc(Par1))
#               # SCA (for hybrids only)
#               + vsc(isc(is.cross), isc(cross.id))
#               # SCAR: exclude selfs (if zero there's no reciprocal SCA effects)
#               + vsc(isc(cross.type), isc(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                         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.

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####

data(DT_polyploid)
# DT <- DT_polyploid
# GT <- GT_polyploid
# MP <- MP_polyploid
# ####============================================####
# ####### convert markers to numeric format
# ####============================================####
# numo <- atcg1234(data=GT, ploidy=4);
# numo$M[1:5,1:5];
# numo$ref.allele[,1:5]
#
# ###============================================####
# ###### plants with both genotypes and phenotypes
# ###============================================####
# common <- intersect(DT$Name,rownames(numo$M))
#
# ###============================================####
# ### get the markers and phenotypes for such inds
# ###============================================####
# marks <- numo$M[common,]; marks[1:5,1:5]
# DT2 <- DT[match(common,DT$Name),];
# DT2 <- as.data.frame(DT2)
# DT2[1:5,]
#
# ###============================================####
# ###### Additive relationship matrix, specify ploidy
# ###============================================####
# A <- A.mat(marks)
# D <- D.mat(marks)
# ###============================================####
```

```
# ### run as mixed model
# ###========================================####
# ans <- mmer(tuber_shape~1,
#             random=~vsr(Name, Gu=A),
#             data=DT2)
# summary(ans)$varcomp
#
# Ai <- as(solve(A + diag(1e-4,ncol(A),ncol(A))), Class="dgCMatrix")
# ansb <- mmec(tuber_shape~1,
#             random=~vsc(isc(Name), Gu=Ai),
#             data=DT2)
# summary(ansb)$varcomp
#
```

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.

See Also

The core functions of the package mmer

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####=========================================####
data(DT_rice)
# DT <- DT_rice
# GT <- GT_rice
# GTn <- GTn_rice
# head(DT)
# M <- atcg1234(GT)
# A <- A.mat(M$M)
# mix <- mmer(Protein.content~1,
#             random = ~vsr(geno, Gu=A) + geno,
#             rcov=~units,
#             data=DT)
# summary(mix)$varcomp
#
# Ai <- as(solve(A + diag(1e-6,ncol(A),ncol(A))), Class="dgCMatrix")
# mixb <- mmec(Protein.content~1,
#             random = ~vsc(isc(geno), Gu=Ai) + geno,
#             rcov=~units,
#             data=DT)
# summary(mixb)$varcomp
```

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

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
# library(lme4)
data(DT_sleepstudy)
DT <- DT_sleepstudy
head(DT)
##################################
## lme4
# fm1 <- lmer(Reaction ~ Days + (1 | Subject), data=DT)
# vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
## sommer
fm2 <- mmer(Reaction ~ Days,
            random= ~ Subject,
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

##################################
## lme4
# fm1 <- lmer(Reaction ~ Days + (Days || Subject), data=DT)
# vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
## sommer
fm2 <- mmer(Reaction ~ Days,
            random= ~ Subject + vsr(Days, Subject),
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

##################################
## lme4
# fm1 <- lmer(Reaction ~ Days + (Days | Subject), data=DT)
# vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
## sommer
## no equivalence in sommer to find the correlation between the 2 vc
## this is the most similar which is equivalent to (intercept || slope)
fm2 <- mmer(Reaction ~ Days,
            random= ~ Subject + vsr(Days, Subject),
```

```
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp

##################################
## lme4
# fm1 <- lmer(Reaction ~ Days + (0 + Days | Subject), data=DT)
# vc <- VarCorr(fm1); print(vc,comp=c("Variance"))
## sommer
fm2 <- mmer(Reaction ~ Days,
            random= ~ vsr(Days, Subject),
            data=DT, tolParInv = 1e-6, verbose = FALSE)
summary(fm2)$varcomp
```

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

Description

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

Usage

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

Format

The format is: chr "DT_technow"

Source

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

References

If using this data for your own research please cite:

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

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

See Also

The core functions of the package mmer

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####=========================================####
data(DT_technow)
DT <- DT_technow
Md <- Md_technow
Mf <- Mf_technow
# Md <- (Md*2) - 1
# Mf <- (Mf*2) - 1
# Ad <- A.mat(Md)
# Af <- A.mat(Mf)
# ###=========================================####
# ###=========================================####
# ans2 <- mmer(GY~1,
#              random=~vsr(dent,Gu=Ad) + vsr(flint,Gu=Af),
#              rcov=~units,
#              data=DT)
# summary(ans2)$varcomp
#
# Adi <- as(solve(Ad + diag(1e-4,ncol(Ad),ncol(Ad))), Class="dgCMatrix")
# Afi <- as(solve(Af + diag(1e-4,ncol(Af),ncol(Af))), Class="dgCMatrix")
# ans2b <- mmec(GY~1,
#              random=~vsc(isc(dent),Gu=Adi) + vsc(isc(flint),Gu=Afi),
#              rcov=~units,
#              data=DT)
# summary(ans2b)$varcomp
# ####=========================================####
# #### multivariate overlayed model
# ####=========================================####
# M <- rbind(Md,Mf)
# A <- A.mat(M)
# ans3 <- mmer(cbind(GY,GM)~1,
#              random=~vsr(overlay(dent,flint),Gu=A),
#              rcov=~vsr(units,Gtc=diag(2)),
#              data=DT)
# summary(ans2)
# cov2cor(ans3$sigma[[1]])
```

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

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

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

See Also

The core functions of the package mmer

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####=========================================####
# data(DT_wheat)
# DT <- DT_wheat
# GT <- GT_wheat
# DTlong <- data.frame(pheno=as.vector(DT),
#                      env=sort(rep(1:4,nrow(DT))),
#                      id=rep(rownames(DT),4))
# DT <- as.data.frame(DT);colnames(DT) <- paste0("x",1:4);DT$line <- rownames(DT);
# rownames(GT) <- DT$line
# K <- A.mat(GT) # additive relationship matrix
# K[1:4,1:4]
# ###=========================================####
# ###=========================================####
# ### using formula based 'mmer'
# ###=========================================####
# ###=========================================####
# head(DT)
# #### univariate
# mix0 <- mmer(x1~1,
#              random = ~vsr(line,Gu=K),
#              rcov=~units,
#              data=DT)
# summary(mix0)$varcomp
#
# Ki <- as(solve(K + diag(1e-4,ncol(K),ncol(K))), Class="dgCMatrix")
# mix0b <- mmec(x1~1,
#              random = ~vsc(isc(line),Gu=Ki),
#              rcov=~units,
#              data=DT)
# summary(mix0b)$varcomp
```

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

Description

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

Format

`block` block factor with 6 levels

`nitro` nitrogen treatment in hundredweight per acre

`Variety` genotype factor, 3 levels

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

`row` row location

`column` column location

Source

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

References

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

Examples

```
### ========================== ###
### ========================== ###
data(DT_yatesoats)
DT <- DT_yatesoats
head(DT)
# m3 <- mmer(fixed=Y ~ V + N + V:N,
#            random = ~ B + B:MP,
#            rcov=~units,
#            data = DT)
# summary(m3)$varcomp
#
# m3b <- mmec(fixed=Y ~ V + N + V:N,
#            random = ~ B + B:MP,
#            rcov=~units,
#            data = DT)
# summary(m3b)$varcomp
```

E.mat *Epistatic relationship matrix*

Description

Calculates the realized epistatic relationship matrix of second order (additive x additive, additive x dominance, or dominance x dominance) using hadamard products with the C++ Armadillo library.

Usage

```
E.mat(X,nishio=TRUE,type="A#A",min.MAF=0.02)
```

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.
nishio	If TRUE Nishio ans Satoh. (2014), otherwise Su et al. (2012) (see Details in the D.mat help page).
type	An argument specifying the type of epistatic relationship matrix desired. The default is the second order epistasis (additive x additive) type="A#A". Other options are additive x dominant (type="A#D"), or dominant by dominant (type="D#D").
min.MAF	Minimum minor allele frequency. The A matrix is not sensitive to rare alleles, so by default only monomorphic markers are removed.

Details

it is computed as the Hadamard product of the epistatic relationship matrix; E=A#A, E=A#D, E=D#D.

Value

The epistatic relationship matrix is returned.

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

Endelman, J.B., and J.-L. Jannink. 2012. Shrinkage estimation of the realized relationship matrix. G3:Genes, Genomes, Genetics. 2:1405-1413. doi: 10.1534/g3.112.004259

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

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

See Also

The core functions of the package mmer

Examples

```
####=========================================####
####random population of 200 lines with 1000 markers
####=========================================####
X <- matrix(rep(0,200*1000),200,1000)
for (i in 1:200) {
  X[i,] <- sample(c(-1,0,0,1), size=1000, replace=TRUE)
}

E <- E.mat(X, type="A#A")
# if heterozygote markers are present can be used "A#D" or "D#D"
```

EM *Expectation Maximization Algorithm*

Description

Univariate version of the expectation maximization (EM) algorithm.

Usage

```
EM(y,X=NULL,ZETA=NULL,R=NULL,iters=30,draw=TRUE,silent=FALSE,
   constraint=TRUE, init=NULL, forced=NULL, tolpar = 1e-04,
   tolparinv = 1e-06)
```

Arguments

y	a numeric vector for the response variable
X	an incidence matrix for fixed effects.
ZETA	an incidence matrix for random effects. This can be for one or more random effects. This NEEDS TO BE PROVIDED AS A LIST STRUCTURE. For example Z=list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=K3)) makes a 2 level list for 3 random effects. The general idea is that each random effect with or without its variance-covariance structure is a list, i.e. list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix. When moving to more than one random effect we need to make several lists that need to be inside another list. What we call a 2-level list, i.e. list(Z=Z1, K=K1) and list(Z=Z2, K=K2) would need to be put in the form; list(list(Z=Z1, K=K1),list(Z=Z1, K=K1)), which as can be seen, is a list of lists (2-level list).
R	a list of matrices for residuals, i.e. for longitudinal data. if not passed is assumed an identity matrix.
draw	a TRUE/FALSE value indicating if a plot of updated values for the variance components and the likelihood should be drawn or not. The default is TRUE. COMPUTATION TIME IS SMALLER IF YOU DON'T PLOT SETTING draw=FALSE
silent	a TRUE/FALSE value indicating if the function should draw the progress bar or iterations performed while working or should not be displayed.
iters	a scalar value indicating how many iterations have to be performed if the EM is performed. There is no rule of tumb for the number of iterations. The default value is 100 iterations or EM steps.
constraint	a TRUE/FALSE value indicating if the program should use the boundary constraint when one or more variance component is close to the zero boundary. The default is TRUE but needs to be used carefully. It works ideally when few variance components are close to the boundary but when there are too many variance components close to zero we highly recommend setting this parameter to FALSE since is more likely to get the right value of the variance components in this way.

`init`	vector of initial values for the variance components. By default this is NULL and variance components are estimated by the method selected, but in case the user want to provide initial values this argument is functional.
`forced`	a vector of numeric values for variance components including error if the user wants to force the values of the variance components. On the meantime only works for forcing all of them and not a subset of them. The default is NULL, meaning that variance components will be estimated by REML/ML.
`tolpar`	tolerance parameter for convergence in the models.
`tolparinv`	tolerance parameter for matrix inversion in the models.

Details

This algorithm is based on Searle (1993) and Bernanrdo (2010). This handles models of the form:

y = Xb + Zu + e

b ~ N[b.hat, 0] .....zero variance because is a fixed term

u ~ N[0, K*sigma(u)] .....where: K*sigma(u) = G

e ~ N[0, I*sigma(e)] .....where: I*sigma(e) = R

y ~ N[Xb, var(Zu+e)] .....where;

var(y) = var(Zu+e) = ZGZ+R = V which is the phenotypic variance

.

The function allows the user to specify the incidence matrices with their respective variance-covariance matrix in a 2 level list structure. For example imagine a mixed model with the following design:

.

fixed = only intercept.....b ~ N[b.hat, 0]

random = GCA1 + GCA2 + SCA.....u ~ N[0, G]

.

where G is:

.

|K*sigma(gca1).....0.....0.....|

|.....0.....S*sigma(gca2).....0.....| = G |.....0.....0.....W*sigma(sca)..|

.

The function is based on useing initial values for variance components, i.e.:

.

var(e) <- 100 var(u1) <- 100 with incidence matrix Z1 var(u2) <- 100 with incidence matrix Z2 var(u3) <- 100 with incidence matrix Z3

.

and estimates the lambda(vx) values in the mixed model equations (MME) developed by Henderson (1975), i.e. consider the 3 random effects stated above, the MME are:

.

|.....X'*R*X.....X'*R*Z1.....X'*R*Z2.....X'*R*Z3 .....| |...X'Ry...|

|.....Z1'*R*X.....Z1'*R*Z1+K1*v1.....Z1'*R*Z2.....Z1'*R*Z3.....| |...Z1'Ry...|
|.....Z2'*R*X.....Z2'*R*Z1.....Z2'*R*Z2+K2*v2.....Z2'*R*Z3.....| |...Z2'Ry...|

|.....Z3'*R*X.....Z3'*R*Z1.....Z3'*R*Z2.....Z3'*R*Z3+K3*v3.....| |...Z3'Ry...|
. .....C.inv.....RHS

.

where "*"" is a matrix product, R is the inverse of the var-cov matrix for the errors, Z1, Z2, Z3 are incidence matrices for random effects, X is the incidence matrix for fixed effects, K1,K2, K3 are the var-cov matrices for random effects and v1,v2,v3 are the estimates of variance components. . The algorithm can be summarized in the next steps: . 1) provide initial values for the variance components 2) estimate the coefficient matrix from MME known as "C" 3) solve the mixed equations as theta = RHS * C.inv 4) obtain new estimates of fixed (b's) and random effects (u's) called theta 5) update values for variance components according to formulas 6) steps are repeated for a number of iterations specified by the user, ideally is enough when no more variations in the estimates is found, in several problems that could take thousands of iterations, whereas in other 10 iterations could be enough.

Value

If all parameters are correctly indicated the program will return a list with the following information:

$var.com a vector with the values of the variance components estimated

$V.inv a matrix with the inverse of the phenotypic variance V = ZGZ+R, V^-1

$u.hat a vector with BLUPs for random effects

$Var.u.hat a vector with variances for BLUPs

$PEV.u.hat a vector with predicted error variance for BLUPs

$beta.hat a vector for BLUEs of fixed effects

$Var.beta.hat a vector with variances for BLUEs

$X incidence matrix for fixed effects

$Z incidence matrix for random effects, if not passed is assumed to be a diagonal matrix

$K the var-cov matrix for the random effect fitted in Z

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

Bernardo Rex. 2010. Breeding for quantitative traits in plants. Second edition. Stemma Press. 390 pp. Searle. 1993. Applying the EM algorithm to calculating ML and REML estimates of variance components. Paper invited for the 1993 American Statistical Association Meeting, San Francisco.

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

# ## Import phenotypic data on inbred performance
# ## Full data
# data("DT_cornhybrids")
# hybrid2 <- DT_cornhybrids # extract cross data
# A <- GT_cornhybrids # extract the var-cov K
# ##########################################
# ##########################################
# ## breeding values with 3 variance components
# ##########################################
# ##########################################
# y <- hybrid2$Yield
# X1 <- model.matrix(~ Location, data = hybrid2);dim(X1)
# Z1 <- model.matrix(~ GCA1 -1, data = hybrid2);dim(Z1)
# Z2 <- model.matrix(~ GCA2 -1, data = hybrid2);dim(Z2)
# Z3 <- model.matrix(~ SCA -1, data = hybrid2);dim(Z3)
#
# K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
# ## Realized IBS relationships for set of parents 1
# K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
# ## Realized IBS relationships for set of parents 2
# S <- kronecker(K1, K2) ; dim(S)
# ## Realized IBS relationships for cross (as the Kronecker product of K1 and K2)
# rownames(S) <- colnames(S) <- levels(hybrid2$SCA)
#
# ETA <- list(list(Z=Z1, K=K1), list(Z=Z2, K=K2))#, list(Z=Z3, K=S))
# ans <- EM(y=y, ZETA=ETA, iters=50)
# ans$var.comp
#
# # compare with NR method
# mix1 <- mmer(Yield~1, random=~vs(GCA1,Gu=K1)+vs(GCA2,Gu=K2), data=hybrid2)
# summary(mix1)$varcomp
#
```

`fcm` *fixed effect constraint indication matrix*

Description

`fcm` creates a matrix with the correct number of columns to specify a constraint in the fixed effects using the Gtc argument of the `vsr` function.

Usage

```
fcm(x, reps=NULL)
```

Arguments

x	vector of 1's and 0's corresponding to the traits for which this fixed effect should be fitted. For example, for a trivariate model if the fixed effect "x" wants to be fitted only for trait 1 and 2 but not for the 3rd trait then you would use fcm(c(1,1,0)) in the Gtc argument of the vsr() function.
reps	integer specifying the number of times the matrix should be repeated in a list format to provide easily the constraints in complex models that use the ds(), us() or cs() structures.

Value

$res a matrix or a list of matrices with the constraints to be provided in the Gtc argument of the vsr function.

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

See Also

The function vsr to know how to use fcm in the mmer solver.

Examples

```
fcm(c(1,1,0))
fcm(c(0,1,1))
fcm(c(1,1,1))

fcm(c(1,1,1),2)

# ## model with Env estimated for both traits
# data(DT_example)
# DT <- DT_example
# A <- A_example
# ans4 <- mmer(cbind(Yield, Weight) ~ Env,
#              random= ~ vsr(Name) + vsr(Env:Name),
#              rcov= ~ vsr(units),
#              data=DT)
# summary(ans4)$betas
# ## model with Env only estimated for Yield
# ans4b <- mmer(cbind(Yield, Weight) ~ vsr(Env, Gtc=fcm(c(1,0))),
```

```
#              random= ~ vsr(Name) + vsr(Env:Name),
#              rcov= ~ vsr(units),
#              data=DT)
# summary(ans4b)$betas
```

`fitted.mmec` *fitted form a LMM fitted with mmec*

Description

`fitted` method for class "`mmec`".

Usage

```
## S3 method for class 'mmec'
fitted(object, ...)
```

Arguments

`object`	an object of class "`mmec`"
`...`	Further arguments to be passed to the mmec function

Value

vector of fitted values of the form y.hat = Xb + Zu including all terms of the model.

Author(s)

Giovanny Covarrubias

See Also

`fitted`, `mmec`

Examples

```
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT) # additive relationship matrix
# #### look at the data and fit the model
# head(DT)
# mix1 <- mmer(Yield~1,
#              random=~vsr(id,Gu=A)
#                     + Rowf + Colf + spl2Da(Row,Col),
#              rcov=~units,
```

```
#               data=DT)
#
# ff=fitted(mix1)
#
# colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
# lattice::wireframe(`u:Row.fitted`~Row*Col, data=ff$dataWithFitted,
#           aspect=c(61/87,0.4), drape=TRUE,# col.regions = colfunc,
#           light.source=c(10,0,10))
# lattice::levelplot(`u:Row.fitted`~Row*Col, data=ff$dataWithFitted, col.regions = colfunc)
```

`fitted.mmer` *fitted form a LMM fitted with mmer*

Description

`fitted` method for class `"mmer"`.

Usage

```
## S3 method for class 'mmer'
fitted(object, ...)
```

Arguments

`object`	an object of class `"mmer"`
`...`	Further arguments to be passed to the mmer function

Value

vector of fitted values of the form y.hat = Xb + Zu including all terms of the model.

Author(s)

Giovanny Covarrubias

See Also

`fitted`, `mmer`

Examples

```
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT) # additive relationship matrix
# #### look at the data and fit the model
```

```
# head(DT)
# mix1 <- mmer(Yield~1,
#              random=~vsr(id,Gu=A)
#                      + Rowf + Colf + spl2Da(Row,Col),
#              rcov=~units,
#              data=DT)
#
# ff=fitted(mix1)
#
# colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
# lattice::wireframe(`u:Row.fitted`~Row*Col, data=ff$dataWithFitted,
#           aspect=c(61/87,0.4), drape=TRUE,# col.regions = colfunc,
#           light.source=c(10,0,10))
# lattice::levelplot(`u:Row.fitted`~Row*Col, data=ff$dataWithFitted, col.regions = colfunc)
```

`fixm` *fixed indication matrix*

Description

`fixm` creates a square matrix with 3's in the diagnals and off-diagonals to quickly specify a fixed constraint in the Gtc argument of the `vsr` function.

Usage

```
fixm(x, reps=NULL)
```

Arguments

`x`	integer specifying the number of traits to be fitted for a given random effect.
`reps`	integer specifying the number of times the matrix should be repeated in a list format to provide easily the constraints in complex models that use the ds(), us() or cs() structures.

Value

$res a matrix or a list of matrices with the constraints to be provided in the Gtc argument of the `vsr` function.

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

See Also

The function vsr to know how to use fixm in the mmer solver.

Examples

```
fixm(4)
fixm(4,2)
```

---

gvsr *general variance structure specification*

Description

gvsr function to build general variance-covariance structures for combination of random effects to be fitted in the mmer solver.

Usage

```
gvsr(..., Gu=NULL, Guc=NULL, Gti=NULL, Gtc=NULL, form=NULL)
```

Arguments

...	names of the random effects (variables in the dataset) to be used to create a general variance structure. For example, for 2 random effects (variables); mom and progeny, a model specified as: gvsr(mom, progeny) will create a variance structure of the form: | sigma2.m sigma.pm | | sigma.pm sigma2.p | where not only variance components for each random effect will be estimated but also the covariance component between the 2 random effects is estimated. The user can also provide a numeric vector or matrix to be considered the design matrix for the ith random effect. More than two random effects can be provided.
Gu	list of matrices with the known variance-covariance values for the levels of the different random effects provided in the "..." argument (i.e. relationship matrix among individuals or any other known covariance matrix). If NULL, then an identity matrix is assumed. For example, a model with 2 random effects with covariance structure should be provided as: gvsr(mom, progeny, Gu=list(Am,Ap)) where Am and Ap are the relationship matrices for the random effects for mom and progeny respectively.
Guc	matrix with the constraints for the u random effects. This is used to specify which variance and covariance parameters between the 1 to 1 combinations of random effects should be estimated. For example, for 2 random effects the expected variance-covariance matrix expected to be estimated (when the default Guc=NULL) is and unstructured model:

| sigma2.m sigma.pm |

| sigma.pm sigma2.p |

but the user can constrain which parameters should be estimated. Providing:

Guc=diag(2) would fit:

| sigma2.m ...0... |

| ...0... sigma2.p |

Gti — matrix with dimensions t x t (t equal to number of traits) with initial values of the variance-covariance components for the random effect specified in the .... argument. If the value is NULL the program will provide the initial values.

Gtc — matrix with dimensions t x t (t equal to number of traits) of constraints for the variance-covariance components for the random effect specified in the ... argument according to the following rules:

0: not to be estimated

1: estimated and constrained to be positive (i.e. variance component)

2: estimated and unconstrained (can be negative or positive, i.e. covariance component)

3: not to be estimated but fixed (value has to be provided in the Gti argument)

In the multi-response scenario if the user doesn't specify this argument the default is to build an unstructured matrix (using the unsm() function). This argument needs to be used wisely since some covariance among responses may not make sense. Useful functions to specify constraints are; diag(), unsm(), fixm().

form — an additional structure to specify a kronecker product on top of the general covariance structure defined in the ... argument.

Value

$res a list with all neccesary elements (incidence matrices, known var-cov structures, unknown covariance structures to be estimated and constraints) to be used in the mmer solver.

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

Covarrubias-Pazaran G (2018) Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

See Also

The core function of the package: mmer

Examples

```
data(DT_ige)
DT <- DT_ige
Af <- A_ige
An <- A_ige
### Direct genetic effects model
# modDGE <- mmer(trait ~ block,
#                random = ~ focal,
#                rcov = ~ units,
#                data = DT)
# summary(modDGE)$varcomp
#
### Indirect genetic effects model without covariance between DGE and IGE
# modDGE <- mmer(trait ~ block,
#                random = ~focal + neighbour,
#                rcov = ~ units,
#                data = DT)
# summary(modDGE)$varcomp
#
### Indirect genetic effects model with covariance between DGE and IGE
# modIGE <- mmer(trait ~ block,
#               random = ~ gvsr(focal, neighbour),
#               rcov = ~ units, iters=4,
#               data = DT)
# summary(modIGE)$varcomp
#
### Indirect genetic effects model with covariance between DGE and IGE using relatioship matrices
# modIGEb <- mmer(trait ~ block,
#                random = ~ gvsr(focal, neighbour, Gu=list(Af,An)),
#                rcov = ~ units,
#                data = DT)
# summary(modIGEb)$varcomp
```

GWAS *Genome wide association study analysis*

Description

Fits a multivariate/univariate linear mixed model GWAS by likelihood methods (REML), see the Details section below. It uses the mmer function and its core coded in C++ using the Armadillo library to optimize dense matrix operations common in the derect-inversion algorithms. After the model fit extracts the inverse of the phenotypic variance matrix to perform the association test for the "p" markers. Please check the Details section (Model enabled) if you have any issue with making the function run.

The package also provides functions to estimate additive (A.mat), dominance (D.mat), epistatic (E.mat) and single step (H.mat) relationship matrices to model known covariances among genotypes typical in plant and animal breeding problems. Other functions to build known covariance

structures among levels of random effects are autoregresive (AR1), compound symmetry (CS) and autoregressive moving average (ARMA) where the user needs to fix the correlation value for such models (this is different to estimating unknown covariance structures). Additionally, overlayed models can be implemented as well (overlay function). Spatial modeling can be done through the two dimensional splines (spl2Da and spl2Db). Random regression models can also be fitted through the (leg) function (orthopolynom package installation is needed for using the leg function).

The sommer package is updated on CRAN every 3-months due to CRAN policies but you can find the latest source at https://github.com/covaruber/sommer . This can be easily installed typing the following in the R console:

library(devtools)

install_github("covaruber/sommer")

This is recommended since bugs fixes will be immediately available in the GitHub source. **For tutorials** on how to perform different analysis with sommer please look at the vignettes by typing in the terminal:

vignette("v1.sommer.quick.start")

vignette("v2.sommer.changes.and.faqs")

vignette("v3.sommer.qg")

vignette("v4.sommer.gxe")

or visit **https://covaruber.github.io**

Usage

```
GWAS(fixed, random, rcov, data, weights, W,
    nIters=20, tolParConvLL = 1e-03, tolParInv = 1e-06,
    init=NULL, constraints=NULL,method="NR",
    getPEV=TRUE,naMethodX="exclude",
    naMethodY="exclude",returnParam=FALSE,
    dateWarning=TRUE,date.warning=TRUE,verbose=FALSE,
    stepWeight=NULL, emWeight=NULL,
    M=NULL, gTerm=NULL, n.PC = 0, min.MAF = 0.05,
    P3D = TRUE)
```

Arguments

fixed	A formula specifying the **response variable(s) and fixed effects**, i.e: *response ~ covariate* for univariate models *cbind(response.i,response.j) ~ covariate* for multivariate models The fcm() function can be used to constrain fixed effects in multi-response models.
random	a formula specifying the name of the **random effects**, i.e. *random= ~ genotype + year*. Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*): vsr(...,Gu,Gt,Gtc) is the main function to specify variance models and special structures for random effects. On the ... argument you provide the unknown

variance-covariance structures (i.e. usr,dsr,at,csr) and the random effect where such covariance structure will be used (the random effect of interest). Gu is used to provide known covariance matrices among the levels of the random effect, Gt initial values and Gtc for constraints. Auxiliar functions for building the variance models are:

** `dsr(x)`, `usr(x)`, `csr(x)` and `atr(x,levs)` can be used to specify unknown diagonal, unstructured and customized unstructured and diagonal covariance structures to be estimated by REML.

** `unsm(x)`, `fixm(x)` and `diag(x)` can be used to build easily matrices to specify constraints in the Gtc argument of the `vsr()` function.

** `overlay()`, `spl2Da()`, `spl2Db()`, and `leg()` functions can be used to specify overlayed of design matrices of random effects, two dimensional spline and random regression models within the `vsr()` function.

`rcov` a formula specifying the name of the **error term**, i.e. *rcov= ~ units*.

The functions that can be used to fit heterogeneous residual variances are the same used on the random term but the random effect is always "units", i.e. *rcov=~vsr(dsr(Location),units)*

`data` a data frame containing the variables specified in the formulas for response, fixed, and random effects.

`weights` name of the covariate for weights. To be used for the product R = Wsi*R*Wsi, where * is the matrix product, Wsi is the square root of the inverse of W and R is the residual matrix.

`W` Alternatively, instead of providing a vector of weights the user can specify an entire W matrix (e.g., when covariances exist). To be used first to produce Wis = solve(chol(W)), and then calculate R = Wsi*R*Wsi.t(), where * is the matrix product, and R is the residual matrix. Only one of the arguments weights or W should be used. If both are indicated W will be given the preference.

`nIters` Maximum number of iterations allowed. Default value is 15.

`tolParConvLL` Convergence criteria.

`tolParInv` tolerance parameter for matrix inverse used when singularities are encountered.

`init` initial values for the variance components. By default this is NULL and variance components are estimated by the method selected, but in case the user want to provide initial values for ALL var-cov components this argument is functional. It has to be provided as a list or an array, where each list element is one variance component and if multitrait model is pursued each element of the list is a matrix of variance covariance components among traits. Initial values can also be provided in the Gt argument of the vsr function.Is highly encouraged to use the Gt and Gtc arguments of the vsr function instead of this argument

`constraints` when initial values are provided these have to be accompanied by their constraints. See the vsr function for more details on the constraints. Is highly encouraged to use the Gt and Gtc arguments of the vsr function instead of this argument.

`method` this refers to the method or algorithm to be used for estimating variance components. Direct-inversion Newton-Raphson **NR** and Average Information **AI** (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2015).

`getPEV` a TRUE/FALSE value indicating if the program should return the predicted error variance and variance for random effects. This option is provided since this can take a long time for certain models where p > n by a big extent.

`naMethodX` one of the two possible values; "include" or "exclude". If "include" is selected then the function will impute the X matrices for fixed effects with the median value. If "exclude" is selected it will get rid of all rows with missing values for the X (fixed) covariates. The default is "exclude". The "include" option should be used carefully.

`naMethodY` one of the three possible values; "include", "include2" or "exclude". If "include" is selected then the function will impute the response variables with the median value. The difference between "include" and "include2" is only available in the multitrait models when the imputation can happen for the entire matrix of responses or only for complete cases ("include2"). If "exclude" is selected it will get rid of rows in responses where missing values are present for the estimation of variance components. The default is "exclude".

`returnParam` a TRUE/FALSE value to indicate if the program should return the parameters used for modeling without fitting the model.

`dateWarning` a TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package.

`date.warning` a TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package.

`verbose` a TRUE/FALSE value to indicate if the program should return the progress of the iterative algorithm.

`stepWeight` A vector of values (of length equal to the number of iterations) indicating the weight used to multiply the update (delta) for variance components at each iteration. If NULL the 1st iteration will be multiplied by 0.5, the 2nd by 0.7, and the rest by 0.9. This argument can help to avoid that variance components go outside the parameter space in the initial iterations which doesn't happen very often with the NR method but it can be detected by looking at the behavior of the likelihood. In that case you may want to give a smaller weight to the initial 8-10 iterations.

`emWeight` A vector of values (of length equal to the number of iterations) indicating with values between 0 and 1 the weight assigned to the EM information matrix. And the values 1 - emWeight will be applied to the NR/AI information matrix to produce a joint information matrix. If NULL weights for EM information matrix are zero and 1 for the NR/AI information matrix.

`M` The marker matrix containing the marker scores for each level of the random effect selected in the gTerm argument, coded as numeric based on the number of reference alleles in the genotype call, e.g. (-1,0,1) = (aa,Aa,AA), levels in diploid individuals. Individuals in rows and markers in columns. No additional columns should be provided, is a purely numerical matrix. Similar logic applies to polyploid individuals, e.g. (-3,-2,-1,0,1,2,3) = (aaaa,aaaA,aaAA,Aaaa,AAaa,AAAa,AAAA).

`gTerm` a character vector indicating the random effect linked to the marker matrix M (i.e. the genetic term) in the model. The random effect selected should have the same number of levels than the number of rows of M. When fitting only a random effect without a special covariance structure (e.g., dsr, usr, etc.) you

will need to add the call 'u:' to the name of the random effect given the behavior of the naming rules of the solver when having a simple random effect without covariance structure.

`n.PC` Number of principal components to include as fixed effects. Default is 0 (equals K model).

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`P3D` When P3D=TRUE, variance components are estimated by REML only once, without any markers in the model and then a for loop for hypothesis testing is performed. When P3D=FALSE, variance components are estimated by REML for each marker separately. The latter can be quite time consuming. As many models will be run as number of marker.

Details

Citation

Type *citation("sommer")* to know how to cite the sommer package in your publications.

Models Enabled

For details about the models enabled and more information about the covariance structures please check the help page of the package (`sommer`). In general the GWAS model implemented in sommer to obtain marker effect is a generalized linear model of the form:

b = (X'V-X)X'V-y

with X = ZMi

where: b is the marker effect (dimensions 1 x mt) y is the response variable (univariate or multivariate) (dimensions 1 x nt) V- is the inverse of the phenotypic variance matrix (dimensions nt x nt) Z is the incidence matrix for the random effect selected (gTerm argument) to perform the GWAS (dimensions nt x ut) Mi is the ith column of the marker matrix (M argument) (dimensions u x m)

for t traits, n observations, m markers and u levels of the random effect. Depending if P3D is TRUE or FALSE the V- matrix will be calculated once and used for all marker tests (P3D=TRUE) or estimated through REML for each marker (P3D=FALSE).

Special Functions

`vsr(atr(x,levels),y)`

can be used to specify heterogeneous variance for the "y"" factor covariate at specific levels of the factor covariate "x", i.e. *random=~vsr(at(Location,c("A","B")),ID)* fits a variance component for ID at levels A and B of the factor covariate Location.

`vsr(dsr(x),y)`

can be used to specify a diagonal covariance structure for the "y"" covariate for all levels of the factor covariate "x", i.e. *random=~vsr(dsr(Location,ID)* fits a variance component for ID at all levels of the factor covariate Location.

`vsr(usr(x),y)`

can be used to specify an unstructured covariance structure for the "y"" covariate for all levels of the factor covariate "x", i.e. *random=~vsr(usr(Location),ID)* fits variance and covariance components for ID at all levels of the factor covariate Location.

`vsr(overlay(...,rlist=NULL,prefix=NULL))`

can be used to specify overlay of design matrices between consecutive random effects specified, i.e. *random=~overlay(male,female)* overlays (overlaps) the incidence matrices for the male and female random effects to obtain a single variance component for both effects. The 'rlist' argument is a list with each element being a numeric value that multiplies the incidence matrix to be overlayed. See `overlay` for details.Can be combined with vsr().

`spl2Da(x.coord, y.coord, at.var, at.levels))`

can be used to fit a 2-dimensional spline (i.e. spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class) assuming a single variance component. The 2D spline can be fitted at specific levels using the `at` and `at.levels` arguments. For example *random=~spl2Da(x.coord=Row,y.coord=Range,at.var=*

`spl2Db(x.coord, y.coord, at.var, at.levels))`

can be used to fit a 2-dimensional spline (i.e. spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class) assuming multiple variance components. The 2D spline can be fitted at specific levels using the `at` and `at.levels` arguments. For example *random=~spl2Db(x.coord=Row,y.coord=Range,at.var=*

For a short tutorial on how to use this special functions you can look at the vignettes by typing in the terminal:

vignette('sommer.start')

Bug report and contact

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

If you have any bug report please go to https://github.com/covaruber/sommer or send me an email to address it asap.

Example Datasets

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

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

* `DT_h2` to calculate heritability

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

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

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

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

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

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

Additional Functions

Other functions such as `summary`, `fitted`, `randef` (notice here is randef not ranef), `anova`, `residuals`, `coef` and `plot` applicable to typical linear models can also be applied to models fitted using the GWAS-type of functions.

Additional functions for genetic analysis have been included such as build a genotypic hybrid marker matrix (`build.HMM`), plot of genetic maps (`map.plot`), creation of manhattan plots (`manhattan`).

If you need to use pedigree you need to convert your pedigree into a relationship matrix (i.e. use the `getA` function from the pedigreemm package).

Useful functions for analyzing field trials are included such as the `spl2Da` and `spl2Db`.

Value

If all parameters are correctly indicated the program will return a list with the following information:

`Vi`	the inverse of the phenotypic variance matrix V^- = (ZGZ+R)^-1
`sigma`	a list with the values of the variance-covariance components with one list element for each random effect.
`sigma_scaled`	a list with the values of the scaled variance-covariance components with one list element for each random effect.
`sigmaSE`	Hessian matrix containing the variance-covariance for the variance components. SE's can be obtained taking the square root of the diagonal values of the Hessian.
`Beta`	a data frame for trait BLUEs (fixed effects).
`VarBeta`	a variance-covariance matrix for trait BLUEs
`U`	a list (one element for each random effect) with a data frame for trait BLUPs.
`VarU`	a list (one element for each random effect) with the variance-covariance matrix for trait BLUPs.
`PevU`	a list (one element for each random effect) with the predicted error variance matrix for trait BLUPs.
`fitted`	Fitted values y.hat=XB
`residuals`	Residual values e = Y - XB
`AIC`	Akaike information criterion
`BIC`	Bayesian information criterion
`convergence`	a TRUE/FALSE statement indicating if the model converged.
`monitor`	The values of log-likelihood and variance-covariance components across iterations during the REML estimation.
`scores`	marker scores (-log_(10)p) for the traits
`method`	The method for extimation of variance components specified by the user.
`constraints`	contraints used in the mixed models for the random effects.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

Covarrubias-Pazaran G. 2018. Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

Bernardo Rex. 2010. Breeding for quantitative traits in plants. Second edition. Stemma Press. 390 pp.

Gilmour et al. 1995. Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. Biometrics 51(4):1440-1450.

Kang et al. 2008. Efficient control of population structure in model organism association mapping. Genetics 178:1709-1723.

Lee, D.-J., Durban, M., and Eilers, P.H.C. (2013). Efficient two-dimensional smoothing with P-spline ANOVA mixed models and nested bases. Computational Statistics and Data Analysis, 61, 22 - 37.

Lee et al. 2015. MTG2: An efficient algorithm for multivariate linear mixed model analysis based on genomic information. Cold Spring Harbor. doi: http://dx.doi.org/10.1101/027201.

Maier et al. 2015. Joint analysis of psychiatric disorders increases accuracy of risk prediction for schizophrenia, bipolar disorder, and major depressive disorder. Am J Hum Genet; 96(2):283-294.

Rodriguez-Alvarez, Maria Xose, et al. Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics 23 (2018): 52-71.

Searle. 1993. Applying the EM algorithm to calculating ML and REML estimates of variance components. Paper invited for the 1993 American Statistical Association Meeting, San Francisco.

Yu et al. 2006. A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Genetics 38:203-208.

Tunnicliffe W. 1989. On the use of marginal likelihood in time series model estimation. JRSS 51(1):15-27.

Zhang et al. 2010. Mixed linear model approach adapted for genome-wide association studies. Nat. Genet. 42:355-360.

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####
#####===========================================####
##### potato example
#####===========================================####
#
# data(DT_polyploid)
# DT <- DT_polyploid
# GT <- GT_polyploid
# MP <- MP_polyploid
# ####=========================================####
# ####### convert markers to numeric format
# ####=========================================####
# numo <- atcg1234(data=GT, ploidy=4);
# numo$M[1:5,1:5];
# numo$ref.allele[,1:5]
#
```

```
# ###=========================================####
# ###### plants with both genotypes and phenotypes
# ###=========================================####
# common <- intersect(DT$Name,rownames(numo$M))
#
# ###=========================================####
# ### get the markers and phenotypes for such inds
# ###=========================================####
# marks <- numo$M[common,]; marks[1:5,1:5]
# DT2 <- DT[match(common,DT$Name),];
# DT2 <- as.data.frame(DT2)
# DT2[1:5,]
#
# ###=========================================####
# ###### Additive relationship matrix, specify ploidy
# ###=========================================####
# A <- A.mat(marks)
# ###=========================================####
# ### run it as GWAS model
# ###=========================================####
# ans2 <- GWAS(tuber_shape~1,
#              random=~vsr(Name,Gu=A),
#              rcov=~units,
#              gTerm = "u:Name",
#              M=marks, data=DT2)
# plot(ans2$scores[,1])
```

H *Two-way id by features table*

Description

H creates a two way id by features table that can be used as the H argument in the rrc function that extracts latent covariates.

Usage

```
H(timevar=NULL, idvar=NULL, response=NULL, Gu=NULL)
```

Arguments

timevar	vector of the dataset containing the variable to be used to form columns in the wide table.
idvar	vector of the dataset containing the variable to be used to form rows in the wide table.
response	vector of the dataset containing the response variable to be used to fill the cells of the wide table.
Gu	an optional covariance matrix (**not the inverse**) between levels of the idvar in case a sparse (unbalanced) design between timevar and idvar exist.

Details

This is just an aggregate, reshape and imputation of a long format table to a wide format table.

Value

$H two way table of id by features effects.

Author(s)

Giovanny Covarrubias-Pazaran

See Also

The function `vsc` to know how to use `H` in the `rrc` function.

Examples

```
# data(DT_h2)
# DT <- DT_h2
# DT=DT[with(DT, order(Env)), ]
# H0 <- with(DT, H(Env, Name, y) )
# Z <- with(DT, rrc(Env, H0, 2))
```

H.mat *Combined relationship matrix H*

Description

Given a matrix A and a matrix G returns a H matrix with the C++ Armadillo library.

Usage

```
H.mat(A, G, tau = 1, omega = 1, tolparinv=1e-6)
```

Arguments

`A`	Additive relationship matrix based on pedigree.
`G`	Additive relationship matrix based on marker data.
`tau`	As described by Martini et al. (2018).
`omega`	As described by Martini et al. (2018).
`tolparinv`	Tolerance parameter for matrix inverse used when singularities are encountered in the estimation procedure.

Details

See references

Value

H Matrix with the relationship between the individuals based on pedigree and corrected by molecular information

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

Martini, J. W., Schrauf, M. F., Garcia-Baccino, C. A., Pimentel, E. C., Munilla, S., Rogberg-Munoz, A., ... & Simianer, H. (2018). The effect of the H-1 scaling factors tau and omega on the structure of H in the single-step procedure. Genetics Selection Evolution, 50(1), 16.

See Also

The core functions of the package mmer

Examples

```
####=========================================####
####random population of 200 lines with 1000 markers
####=========================================####
M <- matrix(rep(0,200*1000),200,1000)
for (i in 1:200) {
  M[i,] <- sample(c(-1,0,0,1), size=1000, replace=TRUE)
}
rownames(M) <- 1:nrow(M)
v <- sample(1:nrow(M),100)
M2 <- M[v,]

A <- A.mat(M) # assume this is a pedigree-based matrix for the sake of example
G <- A.mat(M2)

H <- H.mat(A,G)
# colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
# hv <- heatmap(H[1:15,1:15], col = colfunc(100),Colv = "Rowv")
```

---

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")
```

Arguments

x	a numeric or character vector.
method	the method to choose between mean or median.

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

See Also

The core function of the package mmer

Examples

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

---

isc *identity covariance structure*

Description

isc creates an identity covariance structure for the levels of the random effect to be used with the mmec solver. Any random effect with a special covariance structure should end with an isc() structure.

Usage

```
isc(x, thetaC=NULL, theta=NULL)
```

Arguments

x	vector of observations for the random effect.
thetaC	an optional 1 x 1 matrix for constraints in the variance-covariance components. The values in the matrix define how the variance-covariance components should be estimated: 0: component will not be estimated 1: component will be estimated and constrained to be positive (default) 2: component will be estimated and unconstrained 3: component will be fixed to the value provided in the theta argument
theta	an optional 1 x 1 matrix for initial values of the variance-covariance component. When providing customized values, these values should be scaled with respect to the original variance. For example, to provide an initial value of 1 to a given variance component, theta would be built as: theta = matrix(1 / var(response)) The values in the matrix define the initial values of the variance-covariance components that will be subject to the constraints provided in thetaC. If not provided, initial values (theta) will be 0.15

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

See the function vsc to know how to use isc in the mmec solver.

Examples

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

# data(DT_example)
# ans1 <- mmec(Yield~Env,
#              random= ~ vsc( isc( Name ) ),
#              data=DT_example)
# summary(ans1)$varcomp
```

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.

See Also

The core function of the package mmer

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.

See Also

The core functions of the package mmer and mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####
data(DT_cpdata)
#### get the marker matrix
CPgeno <- GT_cpdata; CPgeno[1:5,1:5]
#### get the map
mapCP <- MP_cpdata; head(mapCP)
names(mapCP) <- c("Locus","Position","LG")
#### with example purposes we only do 3 chromosomes
mapCP <- mapCP[which(mapCP$LG <= 3),]
#### run the function
# res <- LD.decay(CPgeno, mapCP)
```

```
# names(res)
#### 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$all.LG)
# par(new=T)
# lilo <- predict(mod,data.frame(d=1:55))
# plot(lilo, bty="n", xaxt="n", yaxt="n", col="green",
#      type="l", ylim=c(0,1),ylab="",xlab="",lwd=2)
# res3 <- LD.decay(markers=CPgeno, map=mapCP,
#                 unlinked = TRUE,gamma = .95)
# abline(h=res3$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

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

See Also

The core functions of the package mmer

Examples

```
x <- sort(rep(1:3,100))
# you need to install the orthopolynom library
# leg(x, n=1)
# leg(x, n=2)

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

---

list2usmat *list or vector to unstructured matrix*

Description

list2usmat creates an unstructured square matrix taking a vector or list to fill the diagonal and upper triangular with the values provided.

Usage

```
list2usmat(sigmaL)
```

Arguments

sigmaL	vector or list of values to put on the matrix.

Value

$res a matrix with the values provided.

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

See Also

The function `vsr` to know how to use `list2usmat` in the `mmer` solver.

Examples

```
list2usmat(as.list(1:3))
list2usmat(as.list(1:10))
```

`logspace` *Decreasing logarithmic trend*

Description

logspace creates a vector with decreasing logaritmic trend.

Usage

```
logspace(n, start, end)
```

Arguments

`n`	number of values to generate.
`start`	initial value.
`end`	last value.

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

See Also

The core functions of the package `mmer` and `mmec`

Examples

```
logspace(5, 1, .05)
```

`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

See Also

The core functions of the package mmer

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

See Also

The core functions of the package mmer

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)
```

MEMMA *Multivariate Efficient Mixed Model Association Algorithm*

Description

This function is used internally in the function mmer when multiple responses are selected for a single variance component other than the error. It uses the efficient mixed model association (MEMMA) algorithm.

Usage

```
MEMMA(Y, X=NULL, ZETA=NULL, tolpar = 1e-06, tolparinv = 1e-06, check.model=TRUE,
      silent=TRUE)
```

Arguments

Y	a numeric vector for the response variable
X	an incidence matrix for fixed effects.
ZETA	an incidence matrix for random effects. This can be for one or more random effects. This NEEDS TO BE PROVIDED AS A LIST STRUCTURE. For example Z=list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=K3)) makes a 2 level list for 3 random effects. The general idea is that each random effect with or without its variance-covariance structure is a list, i.e. list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix. When moving to more than one random effect we need to make several lists that need to be inside another list. What we call a 2-level list, i.e. list(Z=Z1, K=K1) and list(Z=Z2, K=K2) would need to be put in the form; list(list(Z=Z1, K=K1),list(Z=Z1, K=K1)), which as can be seen, is a list of lists (2-level list).
tolpar	tolerance parameter for convergence
tolparinv	tolerance parameter for matrix inverse
check.model	a TRUE/FALSE value indicating if list structure provided by the user is correct to fix it. The default is TRUE but is turned off to FALSE within the mmer function which would imply a double check.
silent	a TRUE/FALSE value indicating if the function should draw the progress bar or iterations performed while working or should not be displayed.

Details

.

The likelihood function optimized in this algorithm is:

.

logL = (n - p) * log(sum(eta^2/(lambda + delta)) + sum(log(lambda + delta))

.

where: (n-p) refers to the degrees of freedom lambda are the eigenvalues mentioned by Kang et al.(2008) delta is the REML estimator of the ridge parameter

.

The algorithm can be summarized in the next steps:

.

1) provide initial value for the ridge parameter

2) estimate S = I - X(X'X)-X'

3) obtain the phenotypic variance V = ZKZ' + delta.prov*I

4) perform an eigen decomposition of SVS

5) create "lambda"" as the eigenvalues of SVS and "U"" as the eigenvectors

6) estimate eta=U'y

7) optimize the likelihood shown above providing "eta", "lambdas" and optimize with respect to "delta" which is the ridge parameter and contains Ve/Vu

Value

If all parameters are correctly indicated the program will return a list with the following information:

$Vu a scalar value for the variance component estimated

$Ve a scalar value for the error variance estimated

$V.inv a matrix with the inverse of the phenotypic variance V = ZGZ+R, V^-1

$u.hat a vector with BLUPs for random effects

$Var.u.hat a vector with variances for BLUPs

$PEV.u.hat a vector with predicted error variance for BLUPs

$beta.hat a vector for BLUEs of fixed effects

$Var.beta.hat a vector with variances for BLUEs

$X incidence matrix for fixed effects, if not passed is assumed to only include the intercept

$Z incidence matrix for random effects, if not passed is assumed to be a diagonal matrix

$K the var-cov matrix for the random effect fitted in Z

$ll the log-likelihood value for obtained when optimizing the likelihood function when using ML or REML

References

Kang et al. 2008. Efficient control of population structure in model organism association mapping. Genetics 178:1709-1723.

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
# data(CPdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# ### look at the data
# head(DT)
# GT[1:5,1:5]
# ## fit a model including additive and dominance effects
# Y <- DT[,c("color","Yield")]
# Za <- diag(dim(Y)[1])
# A <- A.mat(GT) # additive relationship matrix
# ####================####
# #### ADDITIVE MODEL ####
# ####================####
# ETA.A <- list(add=list(Z=Za,K=A))
# #ans.A <- MEMMA(Y=Y, ZETA=ETA.A)
# #ans.A$var.comp
```

---

mmec **m***ixed* **m***odel* **e***quations for* **c** *coefficients*

Description

The mmec function uses the Henderson mixed model equations and the Average Information algorithm coded in C++ using the Armadillo library to optimize matrix operations common in problems with sparse data (e.g., genotype by environment models). This algorithm is **intended to be used for problems of the type r > c (more records in the data than coefficients to estimate)**. For problems with of the type c > r (more coefficients to estimate than records available), the direct inversion algorithms are faster and we recommend to shift to the use of the mmer function.

Usage

```
mmec(fixed, random, rcov, data, W, nIters=25, tolParConvLL = 1e-03,
     tolParConvNorm = 1e-04, tolParInv = 1e-06, naMethodX="exclude",
     naMethodY="exclude", returnParam=FALSE, dateWarning=TRUE,
     verbose=TRUE,addScaleParam=NULL, stepWeight=NULL, emWeight=NULL,
     contrasts=NULL)
```

Arguments

`fixed`	A formula specifying the **response variable(s) and fixed effects**, i.e: *response ~ covariate*
`random`	A formula specifying the name of the **random effects**, e.g., *random= ~ genotype + year*. Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*): `vsc(...,Gu)` is the main function to specify variance models and special structures for random effects. On the ... argument you provide the unknown variance-covariance structures (e.g., usc,dsc,at,csc) and the random effect where such covariance structure will be used (the random effect of interest). Gu is used to provide known covariance matrices among the levels of the random effect. Auxiliar functions for building the variance models are: ** `dsc(x)`, `usc(x)`, `rrc(x,y,z)` , `isc(x)`,`csc(x)`, and `atc(x,levs)` can be used to specify unknown diagonal, unstructured, reduced-rank, identity, and customized unstructured and diagonal covariance structures respectively to be estimated by REML. ** `unsm(x)`, `fixm(x)` and `diag(x)` can be used to build easily matrices to specify constraints in the Gtc argument of the `vsc()` function. ** `overlay()`, `spl2Dc()`, and `leg()`, `redmm()` functions can be used to specify overlayed of design matrices of random effects, two dimensional spline, random regression, and dimensionality-reduction models within the `vsc()` function.
`rcov`	A formula specifying the name of the **error term**, e.g., *rcov= ~ units*. Special heterogeneous and special variance models and constraints for the residual part are the same used on the random term but the name of the random effect is always "units" which can be thought as a column with as many levels as rows in the data, e.g., *rcov=~vsc(dsc(covariate),isc(units))* When fitting structures at the level of residuals please make sure that your data is sorted based on the factors defining the structure. For example, for *rcov= ~ vsc(dsc(xx), isc(units))* sort the datatset by the variable xx.
`data`	A data frame containing the variables specified in the formulas for response, fixed, and random effects.
`W`	Weights matrix (e.g., when covariance among plots exist). Internally W is squared and inverted as Wsi = solve(chol(W)), then the residual matrix is calculated as R = Wsi*O*Wsi.t(), where * is the matrix product, and O is the original residual matrix.
`nIters`	Maximum number of iterations allowed.

`tolParConvLL` Convergence criteria based in the change of log-likelihood between iteration i and i-1.

`tolParConvNorm` Convergence criteria based in the norm proposed by Jensen, Madsen and Thompson (1997):

e1 = || InfMatInv.diag()/sqrt(N) * dLu ||

where InfMatInv.diag() is the diagonal of the inverse of the information matrix, N is the total number of variance components, and dLu is the vector of first derivatives.

`tolParInv` Tolerance parameter for matrix inverse used when singularities are encountered in the estimation procedure.

`naMethodX` One of the two possible values; "include" or "exclude". If "include" is selected then the function will impute the X matrices for fixed effects with the median value. If "exclude" is selected it will get rid of all rows with missing values for the X (fixed) covariates. The default is "exclude". The "include" option should be used carefully.

`naMethodY` One of the three possible values; "include", "include2" or "exclude" (default) to treat the observations in response variable to be used in the estimation of variance components. The first option "include" will impute the response variables for all rows with the median value, whereas "include2" imputes the responses only for rows where there is observation(s) for at least one of the responses (only available in the multi-response models). If "exclude" is selected (default) it will get rid of rows in response(s) where missing values are present for at least one of the responses.

`returnParam` A TRUE/FALSE value to indicate if the program should return the parameters to be used for fitting the model instead of fitting the model.

`dateWarning` A TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package.

`verbose` A TRUE/FALSE value to indicate if the program should return the progress of the iterative algorithm.

`addScaleParam` additional scale parameters for the thetaF matrix.

`stepWeight` A vector of values (of length equal to the number of iterations) indicating the weight used to multiply the update (delta) for variance components at each iteration. If NULL the 1st iteration will be multiplied by 0.5, the 2nd by 0.7, and the rest by 0.9. This argument can help to avoid that variance components go outside the parameter space in the initial iterations which happens very often with the AI method but it can be detected by looking at the behavior of the likelihood. In that case you may want to give a smaller weight.

`emWeight` A vector of values (of length equal to the number of iterations) indicating with values between 0 and 1 the weight assigned to the EM information matrix. And the values 1 - emWeight will be applied to the AI information matrix to produce a joint information matrix. By default the function gives a weight to the EM algorithm of a logarithmic decrease rate using the following code `logspace(nIters,1,0.05)`.

`contrasts` an optional list. See the contrasts.arg of model.matrix.default.

Details

The use of this function requires a good understanding of mixed models. Please review the 'sommer.quick.start' vignette and pay attention to details like format of your random and fixed variables (e.g. character and factor variables have different properties when returning BLUEs or BLUPs, please see the 'sommer.changes.and.faqs' vignette).

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

vignette("v1.sommer.quick.start")

vignette("v2.sommer.changes.and.faqs")

vignette("v3.sommer.qg")

vignette("v4.sommer.gxe")

Citation

Type *citation("sommer")* to know how to cite the sommer package in your publications.

Special variance structures

`vsc(atc(x,levels),isc(y))`

can be used to specify heterogeneous variance for the "y" covariate at specific levels of the covariate "x", e.g., *random=~vsc(at(Location,c("A","B")),isc(ID))* fits a variance component for ID at levels A and B of the covariate Location.

`vsc(dsc(x),isc(y))`

can be used to specify a diagonal covariance structure for the "y" covariate for all levels of the covariate "x", e.g., *random=~vsc(dsc(Location),isc(ID))* fits a variance component for ID at all levels of the covariate Location.

`vsc(usc(x),isc(y))`

can be used to specify an unstructured covariance structure for the "y" covariate for all levels of the covariate "x", e.g., *random=~vsc(usc(Location),isc(ID))* fits variance and covariance components for ID at all levels of the covariate Location.

`vsc(usc(rrc(x,y,z,nPC)),isc(y))`

can be used to specify an unstructured covariance structure for the "y" effect for all levels of the covariate "x", and a response variable "z", e.g., *random=~vsc(rrc(Location,ID,response, nPC=2),isc(ID))* fits a reduced-rank factor analytic covariance for ID at 2 principal components of the covariate Location.

`vsc(isc(overlay(...,rlist=NULL,prefix=NULL)))`

can be used to specify overlay of design matrices between consecutive random effects specified, e.g., *random=~vsc(isc(overlay(male,female)))* overlays (overlaps) the incidence matrices for the male and female random effects to obtain a single variance component for both effects. The 'rlist' argument is a list with each element being a numeric value that multiplies the incidence matrix to be overlayed. See `overlay` for details.Can be combined with vsc().

`vsc(isc(redmm(x,M,nPC)))`

can be used to create a reduced model matrix of an effect (x) assumed to be a linear function of some feature matrix (M), e.g., *random=~vsc(isc(redmm(x,M)))* creates an incidence matrix from a very large set of features (M) that belong to the levels of x to create a reduced model matrix. See `redmm` for details.Can be combined with vsc().

`vsc(leg(x,n),isc(y))`

can be used to fit a random regression model using a numerical variable `x` that marks the trayectory for the random effect `y`. The leg function can be combined with the special functions `dsc`, `usc` `at` and `csc`. For example *random=~vsc(leg(x,1),isc(y))* or *random=~vsc(usc(leg(x,1)),isc(y))*.

`spl2Dc(x.coord, y.coord, at.var, at.levels))`

can be used to fit a 2-dimensional spline (e.g., spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class) assuming multiple variance components. The 2D spline can be fitted at specific levels using the `at.var` and `at.levels` arguments. For example *random=~spl2Dc(x.coord=Row,y.coord=Range,at.v*

Covariance between random effects

`covc( vsc(isc(ran1)), vsc(isc(ran2)) )`

can be used to specify covariance between two different random effects, e.g., *random=~covc(vsc(isc(x1)), vsc(isc(x2)))* where two random effects in their own vsc() structure are encapsulated. Only applies for simple random effects.

S3 methods

S3 methods are available for some parameter extraction such as `fitted.mmec`, `residuals.mmec`, `summary.mmec`, `randef`, `coef.mmec`, `anova.mmec`, `plot.mmec`, and `predict.mmec` to obtain adjusted means. In addition, the `vpredict` function (replacement of the pin function) can be used to estimate standard errors for linear combinations of variance components (e.g., ratios like h2). The `r2` function calculates reliability.

Additional Functions

Additional functions for genetic analysis have been included such as relationship matrix building (`A.mat`, `D.mat`, `E.mat`, `H.mat`), build a genotypic hybrid marker matrix (`build.HMM`), plot of genetic maps (`map.plot`), and manhattan plots (`manhattan`). If you need to build a pedigree-based relationship matrix use the `getA` function from the pedigreemm package.

Bug report and contact

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

If you have any bug report please go to https://github.com/covaruber/sommer or send me an email to address it asap, just make sure you have read the vignettes carefully before sending your question.

Example Datasets

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

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

* `DT_h2` to calculate heritability

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

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

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

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

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

* DT_btdata dataset contains an animal (birds) model.

* DT_legendre simulated dataset for random regression model.

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

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

Models Enabled

For details about the models enabled and more information about the covariance structures please check the help page of the package (sommer).

Value

If all parameters are correctly indicated the program will return a list with the following information:

llik	the vector of log-likelihoods across iterations
M	the coeficient matrix extended by the response vector y]
W	the column binded matrix W = [X Z y]
b	the vector of fixed effect.
u	the vector of random effect.
bu	the vector of fixed and random effects together.
Ci	the inverse of the coefficient matrix.
avInf	The matrix of second derivatives of the likelihood with respect to the i,j th variance-covariance component.
monitor	The values of the variance-covariance components across iterations during the REML estimation.
constraints	The vector of constraints.
AIC	Akaike information criterion
BIC	Bayesian information criterion
convergence	a TRUE/FALSE statement indicating if the model converged.
partitions	a list where each element contains a matrix indicating where each random effect starts and ends.
percDelta	the matrix of percentage change in deltas (see tolParConvNorm argument).
normMonitor	the matrix of the three norms calculated (see tolParConvNorm argument).
toBoundary	the matrix of variance components that were forced to the boundary across iterations.
Cchol	the Cholesky decomposition of the coefficient matrix.
theta	a list of estimated variance covariance matrices. Each element of the list corresponds to the different random and residual components
sigma	the vector form of the variance-covariance parameters.
data	the dataset used in the model fitting.
y	the response vector.
partitionsX	a list where each element contains a matrix indicating where each fixed effect starts and ends.

uList	a list containing the BLUPs in data frame format where rows are levels of the random effects and column the different factors at which the random effect is fitted. This is specially useful for diagonal and unstructured models.
uPevList	a list containing the BLUPs in data frame format where rows are levels of the random effects and column the different factors at which the random effect is fitted. This is specially useful for diagonal and unstructured models.
Dtable	the table to be used for the predict function to help the program recognize the factors available.
args	the fixed, random and residual formulas from the mmec model.

Author(s)

Coded by Christelle Fernandez Camacho & Giovanny Covarrubias-Pazaran

References

Jensen, J., Mantysaari, E. A., Madsen, P., and Thompson, R. (1997). Residual maximum likelihood estimation of (co) variance components in multivariate mixed linear models using average information. Journal of the Indian Society of Agricultural Statistics, 49, 215-236.

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

Gilmour et al. 1995. Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. Biometrics 51(4):1440-1450.

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

####============================================####
#### EXAMPLES
#### Different models with sommer
####============================================####

data(DT_example)
DT <- DT_example
head(DT)

####============================================####
#### Univariate homogeneous variance models  ####
####============================================####

## Compound simmetry (CS) model
ans1 <- mmec(Yield~Env,
             random= ~ Name + Env:Name,
             rcov= ~ units,
             data=DT)
```

```
summary(ans1)

# ####============================================####
# #### Univariate heterogeneous variance models  ####
# ####============================================####
# DT=DT[with(DT, order(Env)), ]
# ## Compound simmetry (CS) + Diagonal (DIAG) model
# ans2 <- mmec(Yield~Env,
#              random= ~Name + vsc(dsc(Env),isc(Name)),
#              rcov= ~ vsc(dsc(Env),isc(units)),
#              data=DT)
# summary(ans2)
#
# ####============================================####
# ####  Univariate unstructured variance models  ####
# ####============================================####
#
# ans3 <- mmec(Yield~Env,
#              random=~ vsc(usc(Env),isc(Name)),
#              rcov=~vsc(dsc(Env),isc(units)),
#              data=DT)
# summary(ans3)
```

---

mmer **m***ixed* **m***odel* **e***quations for* **r** *records*

Description

The mmer function uses the Direct-Inversion Newton-Raphson or Average Information coded in C++ using the Armadillo library to optimize dense matrix operations common in genomic selection models. These algorithms are **intended to be used for problems of the type c > r (more coefficients to estimate than records in the dataset) and/or dense matrices**. For problems with sparse data, or problems of the type r > c (more records in the dataset than coefficients to estimate), the MME-based algorithm in the mmec function is faster and we recommend to shift to use that function.

Usage

```
mmer(fixed, random, rcov, data, weights, W, nIters=20, tolParConvLL = 1e-03,
     tolParInv = 1e-06, init=NULL, constraints=NULL,method="NR", getPEV=TRUE,
     naMethodX="exclude", naMethodY="exclude",returnParam=FALSE,
   dateWarning=TRUE,date.warning=TRUE,verbose=TRUE, reshapeOutput=TRUE, stepWeight=NULL,
     emWeight=NULL, contrasts=NULL)
```

Arguments

fixed	A formula specifying the **response variable(s) and fixed effects**, e.g.: *response ~ covariate* for univariate models *cbind(response.i,response.j) ~ covariate* for multivariate models

The fcm() function can be used to constrain fixed effects in multi-response models.

random — A formula specifying the name of the **random effects**, e.g. *random= ~ genotype + year*.

Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*):

vsr(...,Gu,Gti,Gtc) is the main function to specify variance models and special structures for random effects. On the ... argument you provide the unknown variance-covariance structures (e.g., usr,dsr,atr,csr) and the random effect where such covariance structure will be used (the random effect of interest). Gu is used to provide known covariance matrices among the levels of the random effect, Gti initial values and Gtc for constraints. Auxiliar functions for building the variance models are:

** dsr(x), usr(x), csr(x) and atr(x,levs) can be used to specify unknown diagonal, unstructured and customized unstructured and diagonal covariance structures to be estimated by REML.

** unsm(x), fixm(x) and diag(x) can be used to build easily matrices to specify constraints in the Gtc argument of the vsr() function.

** overlay(), spl2Da(), spl2Db(), and leg() functions can be used to specify overlayed of design matrices of random effects, two dimensional spline and random regression models within the vsr() function.

gvsr(...,Gu,Guc,Gti,Gtc) is an alternative function to specify general variance structures between different random effects. An special case in the indirect genetic effect models. Is similar to the vsr function but in the ... argument the different random effects are provided.

rcov — A formula specifying the name of the **error term**, e.g., *rcov= ~ units*.

Special heterogeneous and special variance models and constraints for the residual part are the same used on the random term but the name of the random effect is always "units" which can be thought as a column with as many levels as rows in the data, e.g., *rcov=~vsr(dsr(covariate),units)*

data — A data frame containing the variables specified in the formulas for response, fixed, and random effects.

weights — Name of the covariate for weights. To be used for the product R = Wsi*R*Wsi, where * is the matrix product, Wsi is the square root of the inverse of W and R is the residual matrix.

W — Alternatively, instead of providing a vector of weights the user can specify an entire W matrix (e.g., when covariances exist). To be used first to produce Wis = solve(chol(W)), and then calculate R = Wsi*R*Wsi.t(), where * is the matrix product, and R is the residual matrix. Only one of the arguments weights or W should be used. If both are indicated W will be given the preference.

nIters — Maximum number of iterations allowed.

tolParConvLL — Convergence criteria for the change in log-likelihood.

tolParInv — Tolerance parameter for matrix inverse used when singularities are encountered in the estimation procedure.

`init` Initial values for the variance components. By default this is NULL and initial values for the variance components are provided by the algorithm, but in case the user want to provide initial values for ALL var-cov components this argument is functional. It has to be provided as a list, where each list element corresponds to one random effect (1x1 matrix) and if multitrait model is pursued each element of the list is a matrix of variance covariance components among traits for such random effect. Initial values can also be provided in the Gti argument of the vsr function. Is highly encouraged to use the Gti and Gtc arguments of the vsr function instead of this argument, but these argument can be used to provide all initial values at once

`constraints` When initial values are provided these have to be accompanied by their constraints. See the vsr function for more details on the constraints. Is highly encouraged to use the Gti and Gtc arguments of the vsr function instead of this argument but these argument can be used to provide all constraints at once.

`method` This refers to the method or algorithm to be used for estimating variance components. Direct-inversion Newton-Raphson **NR** and Average Information **AI** (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2015).

`getPEV` A TRUE/FALSE value indicating if the program should return the predicted error variance and variance for random effects. This option is provided since this can take a long time for certain models where p is > n by a big extent.

`naMethodX` One of the two possible values; "include" or "exclude". If "include" is selected then the function will impute the X matrices for fixed effects with the median value. If "exclude" is selected it will get rid of all rows with missing values for the X (fixed) covariates. The default is "exclude". The "include" option should be used carefully.

`naMethodY` One of the three possible values; "include", "include2" or "exclude" (default) to treat the observations in response variable to be used in the estimation of variance components. The first option "include" will impute the response variables for all rows with the median value, whereas "include2" imputes the responses only for rows where there is observation(s) for at least one of the responses (only available in the multi-response models). If "exclude" is selected (default) it will get rid of rows in response(s) where missing values are present for at least one of the responses.

`returnParam` A TRUE/FALSE value to indicate if the program should return the parameters to be used for fitting the model instead of fitting the model.

`dateWarning` A TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package.

`date.warning` A TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package. This argument will be removed soon, just left for backcompatibility.

`verbose` A TRUE/FALSE value to indicate if the program should return the progress of the iterative algorithm.

`reshapeOutput` A TRUE/FALSE value to indicate if the output should be reshaped to be easier to interpret for the user, some information is missing from the multivariate models for an easy interpretation.

`stepWeight` A vector of values (of length equal to the number of iterations) indicating the weight used to multiply the update (delta) for variance components at each iteration. If NULL the 1st iteration will be multiplied by 0.5, the 2nd by 0.7, and the rest by 0.9. This argument can help to avoid that variance components go outside the parameter space in the initial iterations which doesn't happen very often with the NR method but it can be detected by looking at the behavior of the likelihood. In that case you may want to give a smaller weight to the initial 8-10 iterations.

`emWeight` A vector of values (of length equal to the number of iterations) indicating with values between 0 and 1 the weight assigned to the EM information matrix. And the values 1 - emWeight will be applied to the NR/AI information matrix to produce a joint information matrix.

`contrasts` an optional list. See the contrasts.arg of model.matrix.default.

Details

The use of this function requires a good understanding of mixed models. Please review the 'sommer.quick.start' vignette and pay attention to details like format of your random and fixed variables (e.g. character and factor variables have different properties when returning BLUEs or BLUPs, please see the 'sommer.changes.and.faqs' vignette).

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

vignette("v1.sommer.quick.start")

vignette("v2.sommer.changes.and.faqs")

vignette("v3.sommer.qg")

vignette("v4.sommer.gxe")

Citation

Type *citation("sommer")* to know how to cite the sommer package in your publications.

Special variance structures

`vsr(atr(x,levels),y)`

can be used to specify heterogeneous variance for the "y" covariate at specific levels of the covariate "x", e.g., *random=~vsr(at(Location,c("A","B")),ID)* fits a variance component for ID at levels A and B of the covariate Location.

`vsr(dsr(x),y)`

can be used to specify a diagonal covariance structure for the "y" covariate for all levels of the covariate "x", e.g., *random=~vsr(dsr(Location),ID)* fits a variance component for ID at all levels of the covariate Location.

`vsr(usr(x),y)`

can be used to specify an unstructured covariance structure for the "y" covariate for all levels of the covariate "x", e.g., *random=~vsr(usr(Location),ID)* fits variance and covariance components for ID at all levels of the covariate Location.

`vsr(overlay(...,rlist=NULL,prefix=NULL))`

can be used to specify overlay of design matrices between consecutive random effects specified, e.g., *random=~vsr(overlay(male,female))* overlays (overlaps) the incidence matrices for the male and female random effects to obtain a single variance component for both effects. The 'rlist' argument is a list with each element being a numeric value that multiplies the incidence matrix to be overlayed. See `overlay` for details.Can be combined with vsr().

`vsr(leg(x,n),y)`

can be used to fit a random regression model using a numerical variable `x` that marks the trayectory for the random effect `y`. The leg function can be combined with the special functions `dsr`, `usr at` and `csr`. For example *random=~vsr(leg(x,1),y)* or *random=~vsr(usr(leg(x,1)),y)*.

`vsr(x,Gtc=fcm(v))`

can be used to constrain fixed effects in the multi-response mixed models. This is a vector that specifies if the fixed effect is to be estimated for such trait. For example *fixed=cbind(response.i, response.j)~vsr(Rowf, Gtc=fcm(c(1,0)))* means that the fixed effect Rowf should only be estimated for the first response and the second should only have the intercept.

`gvsr(x,y)`

can be used to fit variance and covariance parameters between two or more random effects. For example, indirect genetic effect models.

`spl2Da(x.coord, y.coord, at.var, at.levels))`

can be used to fit a 2-dimensional spline (e.g., spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class) assuming a single variance component. The 2D spline can be fitted at specific levels using the `at.var` and `at.levels` arguments. For example *random=~spl2Da(x.coord=Row,y.coord=Range,at.*

`spl2Db(x.coord, y.coord, at.var, at.levels))`

can be used to fit a 2-dimensional spline (e.g., spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class) assuming multiple variance components. The 2D spline can be fitted at specific levels using the `at.var` and `at.levels` arguments. For example *random=~spl2Db(x.coord=Row,y.coord=Range,at.*

S3 methods

S3 methods are available for some parameter extraction such as `fitted.mmer`, `residuals.mmer`, `summary.mmer`, `randef`, `coef.mmer`, `anova.mmer`, `plot.mmer`, and `predict.mmer` to obtain adjusted means. In addition, the `vpredict` function (replacement of the pin function) can be used to estimate standard errors for linear combinations of variance components (e.g., ratios like h2).

Additional Functions

Additional functions for genetic analysis have been included such as relationship matrix building (`A.mat`, `D.mat`, `E.mat`, `H.mat`), build a genotypic hybrid marker matrix (`build.HMM`), plot of genetic maps (`map.plot`), and manhattan plots (`manhattan`). If you need to build a pedigree-based relationship matrix use the `getA` function from the pedigreemm package.

Bug report and contact

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

If you have any bug report please go to https://github.com/covaruber/sommer or send me an email to address it asap, just make sure you have read the vignettes carefully before sending your question.

Example Datasets

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

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

* DT_h2 to calculate heritability

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

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

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

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

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

* DT_btdata dataset contains an animal (birds) model.

* DT_legendre simulated dataset for random regression model.

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

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

Models Enabled

For details about the models enabled and more information about the covariance structures please check the help page of the package (sommer).

Value

If all parameters are correctly indicated the program will return a list with the following information:

Vi	the inverse of the phenotypic variance matrix V^- = (ZGZ+R)^-1
P	the projection matrix Vi - [Vi*(X*Vi*X)^-*Vi]
sigma	a list with the values of the variance-covariance components with one list element for each random effect.
sigma_scaled	a list with the values of the scaled variance-covariance components with one list element for each random effect.
sigmaSE	Hessian matrix containing the variance-covariance for the variance components. SE's can be obtained taking the square root of the diagonal values of the Hessian.
Beta	a data frame for trait BLUEs (fixed effects).
VarBeta	a variance-covariance matrix for trait BLUEs
U	a list (one element for each random effect) with a data frame for trait BLUPs.
VarU	a list (one element for each random effect) with the variance-covariance matrix for trait BLUPs.
PevU	a list (one element for each random effect) with the predicted error variance matrix for trait BLUPs.
fitted	Fitted values y.hat=XB
residuals	Residual values e = Y - XB
AIC	Akaike information criterion
BIC	Bayesian information criterion

convergence	a TRUE/FALSE statement indicating if the model converged.
monitor	The values of log-likelihood and variance-covariance components across iterations during the REML estimation.
percChange	The percent change of variance components across iterations. There should be one column less than the number of iterations. Calculated as percChange = ((x_i/x_i-1) - 1) * 100 where i is the ith iteration.
dL	The vector of first derivatives of the likelihood with respect to the ith variance-covariance component.
dL2	The matrix of second derivatives of the likelihood with respect to the i.j th variance-covariance component.
method	The method for extimation of variance components specified by the user.
call	Formula for fixed, random and rcov used.
constraints	contraints used in the mixed models for the random effects.
constraintsF	contraints used in the mixed models for the fixed effects.
data	The dataset used in the model after removing missing records for the response variable.
dataOriginal	The original dataset used in the model.
terms	The name of terms for responses, fixed, random and residual effects in the model.
termsN	The number of effects associated to fixed, random and residual effects in the model.
sigmaVector	a vectorized version of the sigma element (variance-covariance components) to match easily the standard errors of the var-cov components stored in the element sigmaSE.
reshapeOutput	The value provided to the mmer function for the argument with the same name.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

Covarrubias-Pazaran G. 2018. Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

Bernardo Rex. 2010. Breeding for quantitative traits in plants. Second edition. Stemma Press. 390 pp.

Gilmour et al. 1995. Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. Biometrics 51(4):1440-1450.

Kang et al. 2008. Efficient control of population structure in model organism association mapping. Genetics 178:1709-1723.

Lee, D.-J., Durban, M., and Eilers, P.H.C. (2013). Efficient two-dimensional smoothing with P-spline ANOVA mixed models and nested bases. Computational Statistics and Data Analysis, 61, 22 - 37.

Lee et al. 2015. MTG2: An efficient algorithm for multivariate linear mixed model analysis based on genomic information. Cold Spring Harbor. doi: http://dx.doi.org/10.1101/027201.

Maier et al. 2015. Joint analysis of psychiatric disorders increases accuracy of risk prediction for schizophrenia, bipolar disorder, and major depressive disorder. Am J Hum Genet; 96(2):283-294.

Rodriguez-Alvarez, Maria Xose, et al. Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics 23 (2018): 52-71.

Searle. 1993. Applying the EM algorithm to calculating ML and REML estimates of variance components. Paper invited for the 1993 American Statistical Association Meeting, San Francisco.

Yu et al. 2006. A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Genetics 38:203-208.

Tunnicliffe W. 1989. On the use of marginal likelihood in time series model estimation. JRSS 51(1):15-27.

Zhang et al. 2010. Mixed linear model approach adapted for genome-wide association studies. Nat. Genet. 42:355-360.

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

####============================================####
#### EXAMPLES
#### Different models with sommer
####============================================####

data(DT_example)
DT <- DT_example
head(DT)

####============================================####
#### Univariate homogeneous variance models  ####
####============================================####

## Compound simmetry (CS) model
ans1 <- mmer(Yield~Env,
             random= ~ Name + Env:Name,
             rcov= ~ units,
             data=DT)
summary(ans1)

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

```
## Compound simmetry (CS) + Diagonal (DIAG) model
ans2 <- mmer(Yield~Env,
             random= ~Name + vsr(dsr(Env),Name),
             rcov= ~ vsr(dsr(Env),units),
             data=DT)
summary(ans2)

####============================================####
####  Univariate unstructured variance models  ####
####============================================####

ans3 <- mmer(Yield~Env,
             random=~ vsr(usr(Env),Name),
             rcov=~vsr(dsr(Env),units),
             data=DT)
summary(ans3)

# ####============================================####
# #### Multivariate homogeneous variance models ####
# ####============================================####
#
# ## Multivariate Compound simmetry (CS) model
# DT$EnvName <- paste(DT$Env,DT$Name)
# ans4 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vsr(Name, Gtc = unsm(2)) + vsr(EnvName,Gtc = unsm(2)),
#               rcov= ~ vsr(units, Gtc = unsm(2)),
#               data=DT)
# summary(ans4)
#
# ####================================================####
# #### Multivariate heterogeneous variance models  ####
# ####================================================####
#
# ## Multivariate Compound simmetry (CS) + Diagonal (DIAG) model
# ans5 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vsr(Name, Gtc = unsm(2)) + vsr(dsr(Env),Name, Gtc = unsm(2)),
#               rcov= ~ vsr(dsr(Env),units, Gtc = unsm(2)),
#               data=DT)
# summary(ans5)
#
# ####============================================####
# #### Multivariate unstructured variance models ####
# ####============================================####
#
# ans6 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vsr(usr(Env),Name, Gtc = unsm(2)),
#               rcov= ~ vsr(dsr(Env),units, Gtc = unsm(2)),
#               data=DT)
# summary(ans6)
#
# ####========================================####
# ####========================================####
```

```
# #### EXAMPLE SET 2
# #### 2 variance components
# #### one random effect with variance covariance structure
# ####=========================================####
# ####=========================================####
#
# data("DT_cpdata")
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# head(DT)
# GT[1:4,1:4]
# #### create the variance-covariance matrix
# A <- A.mat(GT)
# #### look at the data and fit the model
# mix1 <- mmer(Yield~1,
#              random=~vsr(id, Gu=A) + Rowf,
#              rcov=~units,
#              data=DT)
# summary(mix1)$varcomp
# #### calculate heritability
# vpredict(mix1, h1 ~ V1/(V1+V3) )
# #### multi trait example
# mix2 <- mmer(cbind(Yield,color)~1,
#               random = ~ vsr(id, Gu=A, Gtc = unsm(2)) + # unstructured at trait level
#                          vsr(Rowf, Gtc=diag(2)) + # diagonal structure at trait level
#                            vsr(Colf, Gtc=diag(2)), # diagonal structure at trait level
#               rcov = ~ vsr(units, Gtc = unsm(2)), # unstructured at trait level
#               data=DT)
# summary(mix2)
#
# ####=========================================####
# #### EXAMPLE SET 3
# #### comparison with lmer, install 'lme4'
# #### and run the code below
# ####=========================================####
#
# #### lmer cannot use var-cov matrices so we will not
# #### use them in this comparison example
#
# library(lme4)
# library(sommer)
# data("DT_cornhybrids")
# DT <- DT_cornhybrids
# DTi <- DTi_cornhybrids
# GT <- GT_cornhybrids
#
# fm1 <- lmer(Yield ~ Location + (1|GCA1) + (1|GCA2) + (1|SCA),
#             data=DT )
# out <- mmer(Yield ~ Location,
#              random = ~ GCA1 + GCA2 + SCA,
#              rcov = ~ units,
#              data=DT)
```

```
# summary(fm1)
# summary(out)
# ### same BLUPs for GCA1, GCA2, SCA than lme4
# plot(out$U$GCA1$Yield, ranef(fm1)$GCA1[,1])
# plot(out$U$GCA2$Yield, ranef(fm1)$GCA2[,1])
# vv=which(abs(out$U$SCA$Yield) > 0)
# plot(out$U$SCA$Yield[vv], ranef(fm1)$SCA[,1])
#
# ### a more complex model specifying which locations
# head(DT)
# out2 <- mmer(Yield ~ Location,
#              random = ~ vsr(atr(Location,c("3","4")),GCA2) +
#                         vsr(atr(Location,c("3","4")),SCA),
#              rcov = ~ vsr(dsr(Location),units),
#              data=DT)
# summary(out2)
```

---

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

See Also

The core functions of the package mmec

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####=========================================####

# data(DT_cpdata) # Madison cranberries
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# M <- GT
# # run the function
# ne <- neMarker(M, neExplore = seq(2,30,2), nSamples = 10)
# #################
# data(DT_technow) # maize
# M <- Md_technow # dent
# M <- (M*2) - 1
# M <- M + 1
# # run the function
# ne <- neMarker(M, neExplore = seq(5,100,5), nSamples = 10)
# ##
# M <- Mf_technow # flint
# M <- (M*2) - 1
# M <- M + 1
# # run the function
# ne <- neMarker(M, neExplore = seq(5,100,5), nSamples = 10)
# #################
# data(DT_wheat) # cimmyt wheat
# M <- GT_wheat + 1
# # run the function
# ne <- neMarker(M, neExplore = seq(5,60,5), nSamples = 10)
# ###############
# data(DT_rice) # Zhao rice
# M <- atcg1234(GT_rice)$M
# # run the function
# ne <- neMarker(M,  neExplore = seq(10,100,10), nSamples = 10)
# ###############
# data(DT_polyploid) # endelman potatoes
# M <- atcg1234(data=GT_polyploid, ploidy=4)$M
# # run the function
# ne <- neMarker(M,  neExplore = seq(10,100,10), nSamples = 10)
#
```

```
# library(ggplot2) #For making plots
# ggplot(ne,aes(x=Ne,y=allelesCovered))+
#   geom_ribbon(aes(x=Ne,ymin=allelesCovered-allelesCoveredSe,
#                   ymax=allelesCovered+allelesCoveredSe),
#                   alpha=0.2,linetype=0)+
#   geom_line(linewidth=1)+
#   guides(alpha=FALSE)+
#   theme_bw()+
#   scale_x_continuous("Individual number")+
#   scale_y_continuous("Allele coverage")  +
#               geom_hline(yintercept = 0.95) +
#               geom_hline(yintercept = 0.975)
```

`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.

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

See Also

The core functions of the package `mmer` and a function for creating dummy variables for diallel models named `add.diallel.vars`.

Examples

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

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

`plot.mmec` *plot form a LMM plot with mmec*

Description

`plot` method for class `"mmec"`.

Usage

```
## S3 method for class 'mmec'
plot(x,stnd=TRUE, ...)
```

Arguments

`x`	an object of class `"mmec"`
`stnd`	argument for ploting the residuals to know if they should be standarized.
`...`	Further arguments to be passed

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

plot, mmec

Examples

```
data(DT_yatesoats)
DT <- DT_yatesoats
head(DT)
m3 <- mmec(fixed=Y ~ V + N + V:N,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)
plot(m3)
```

---

plot.mmer *plot form a LMM plot with mmer*

Description

plot method for class "mmer".

Usage

```
## S3 method for class 'mmer'
plot(x,stnd=TRUE, ...)
```

Arguments

x	an object of class "mmer"
stnd	argument for ploting the residuals to know if they should be standarized.
...	Further arguments to be passed

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

plot, mmer

Examples

```
data(DT_yatesoats)
DT <- DT_yatesoats
head(DT)
m3 <- mmer(fixed=Y ~ V + N + V:N,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)
plot(m3)
```

---

pmonitor *plot the change of VC across iterations*

Description

plot for monitoring.

Usage

```
pmonitor(object, ...)
```

Arguments

object	model object of class "mmec"
...	Further arguments to be passed to the plot function.

Value

vector of plot

Author(s)

Giovanny Covarrubias

See Also

plot, mmec

Examples

```
data(DT_yatesoats)
DT <- DT_yatesoats
head(DT)
m3 <- mmec(fixed=Y ~ V + N + V:N,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)
pmonitor(m3)
```

`predict.mmec` *Predict form of a LMM fitted with mmec*

Description

`predict` method for class "`mmec`".

Usage

```
## S3 method for class 'mmec'
predict(object, Dtable=NULL, D, ...)
```

Arguments

`object`	a mixed model of class "`mmec`"
`Dtable`	a table specifying the terms to be included or averaged. An "include" term means that the model matrices for that fixed or random effect is filled with 1's for the positions where column names and row names match. An "include and average" term means that the model matrices for that fixed or random effect is filled with 1/#1's in that row. An "average" term alone means that all rows for such fixed or random effect will be filled with 1/#levels in the effect. If a term is not considered "include" or "average" is then totally ignored in the BLUP and SE calculation. The default rule to invoke when the user doesn't provide the Dtable is to include and average all terms that match the argument D.
`D`	a character string specifying the variable used to extract levels for the rows of the D matrix and its construction. Alternatively, the D matrix (of class dgCMatrix) specifying the matrix to be used for the predictions directly.
`...`	Further arguments to be passed.

Details

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

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

predicted.value equal Xb + Zu.1 + ... + Zu.n

For computing standard errors for predictions the parts of the coefficient matrix:

C11 equal (X.t() V.inv() X).inv()

C12 equal 0 - [(X.t() V.inv() X).inv() X.t() V.inv() G Z]

C22 equal PEV equal G - [Z.t() G[V.inv() - (V.inv() X X.t() V.inv() X V.inv() X)]G Z.t()]

In practive C equals (W.t() V.inv() W).inv()

when both fixed and random effects are present in the inclusion set. If only fixed and random effects are included, only the respective terms from the SE for fixed or random effects are calculated.

Value

`pvals`	the table of predictions according to the specified arguments.
`vcov`	the variance covariance for the predictions.
`D`	the model matrix for predictions as defined in Welham et al.(2004).
`Dtable`	the table specifying the terms to include and terms to be averaged.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

`predict`, `mmec`

Examples

```
data(DT_yatesoats)
DT <- DT_yatesoats
m3 <- mmec(fixed=Y ~ V + N + V:N ,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)

############################
```

```
## predict means for nitrogen
############################
Dt <- m3$Dtable; Dt
# first fixed effect just average
Dt[1,"average"] = TRUE
# second fixed effect include
Dt[2,"include"] = TRUE
# third fixed effect include and average
Dt[3,"include"] = TRUE
Dt[3,"average"] = TRUE
Dt

pp=predict(object=m3, Dtable=Dt, D="N")
pp$pvals

############################
## predict means for variety
############################

Dt <- m3$Dtable; Dt
# first fixed effect include
Dt[1,"include"] = TRUE
# second fixed effect just average
Dt[2,"average"] = TRUE
# third fixed effect include and average
Dt[3,"include"] = TRUE
Dt[3,"average"] = TRUE
Dt

pp=predict(object=m3, Dtable=Dt, D="V")
pp$pvals

############################
## predict means for nitrogen:variety
############################
# prediction matrix D based on (equivalent to classify in asreml)
Dt <- m3$Dtable; Dt
# first fixed effect include and average
Dt[1,"include"] = TRUE
Dt[1,"average"] = TRUE
# second fixed effect include and average
Dt[2,"include"] = TRUE
Dt[2,"average"] = TRUE
# third fixed effect include and average
Dt[3,"include"] = TRUE
Dt[3,"average"] = TRUE
Dt

pp=predict(object=m3, Dtable=Dt, D="N:V")
pp$pvals
```

`predict.mmer` *Predict form of a LMM fitted with mmer*

Description

`predict` method for class `"mmer"`.

Usage

```
## S3 method for class 'mmer'
predict(object, Dtable=NULL, D, ...)
```

Arguments

`object`	a mixed model of class `"mmer"`
`Dtable`	a table specifying the terms to be included or averaged. An "include" term means that the model matrices for that fixed or random effect is filled with 1's for the positions where column names and row names match. An "include and average" term means that the model matrices for that fixed or random effect is filled with 1/#1's in that row. An "average" term alone means that all rows for such fixed or random effect will be filled with 1/#levels in the effect. If a term is not considered "include" or "average" is then totally ignored in the BLUP and SE calculation. The default rule to invoke when the user doesn't provide the Dtable is to include and average all terms that match the argument D.
`D`	a character string specifying the variable used to extract levels for the rows of the D matrix and its construction. Alternatively, the D matrix (of class dgCMatrix) specifying the matrix to be used for the predictions directly.
`...`	Further arguments to be passed.

Details

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

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

predicted.value equal Xb + Zu.1 + ... + Zu.n

For computing standard errors for predictions the parts of the coefficient matrix:

C11 equal (X.t() V.inv() X).inv()

C12 equal 0 - [(X.t() V.inv() X).inv() X.t() V.inv() G Z]

C22 equal PEV equal G - [Z.t() G[V.inv() - (V.inv() X X.t() V.inv() X V.inv() X)]G Z.t()]

In practive C equals (W.t() V.inv() W).inv()

when both fixed and random effects are present in the inclusion set. If only fixed and random effects are included, only the respective terms from the SE for fixed or random effects are calculated.

Value

pvals	the table of predictions according to the specified arguments.
vcov	the variance covariance for the predictions.
D	the model matrix for predictions as defined in Welham et al.(2004).
Dtable	the table specifying the terms to include and terms to be averaged.

Author(s)

Giovanny Covarrubias

References

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

See Also

predict, mmer

Examples

```
data(DT_yatesoats)
DT <- DT_yatesoats
m3 <- mmer(fixed=Y ~ V + N + V:N ,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)

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

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

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 sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

See Also

See the function vsc to know how to use propMissing in the mmec solver.

Examples

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

r2 *Reliability*

Description

Calculates the reliability of BLUPs in a sommer model.

Usage

```
r2(object, object2=NULL)
```

Arguments

object	Model fitted with the mmec function.
object2	An optional model identical to object in the first argument but fitted with the argument returnParam set to TRUE to access the relationship matrices from the fitted model.

Details

The reliability method calculated is the classical animal model: R2=(G-PEV)/G

Value

result a list with as many elements as random effects fitted containing reliabilities for individual BLUPs.

References

Mrode, R. A. (2014). Linear models for the prediction of animal breeding values. Cabi.

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

See Also

mmec – the core function of the package

Examples

```
####============================================####
#### Example population
####============================================####
data(DT_example)
DT <- DT_example
head(DT)
ans1 <- mmec(Yield~Env,
             random= ~ Name + Env:Name,
             rcov= ~ units,
```

```
              data=DT)
rel=r2(ans1)
```

---

randef *extracting random effects*

Description

This function is extracts the random effects from a mixed model fitted by mmer.

Usage

```
randef(object)
```

Arguments

object	an mmer object

Value

$randef a list structure with the random effects or BLUPs.

Examples

```
# randef(model)
```

---

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

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

See Also

The core functions of the package mmec

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####

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

M <- tcrossprod(Md)
xx = with(DT, redmm(x=dent, M=M, nPC=10))
```

```
# ans <- mmec(GY~1,
#             # new model matrix instead of dent
#             random=~vsc(isc(xx$Z)),
#             rcov=~units,
#             data=DT)
# summary(ans)$varcomp
# u = xx$Lam * ans$uList[[1]] # change * for matrix product
```

`residuals.mmec` *Residuals form a GLMM fitted with mmec*

Description

`residuals` method for class "`mmec`".

Usage

```
## S3 method for class 'mmec'
residuals(object, ...)
```

Arguments

`object`	an object of class "`mmec`"
`...`	Further arguments to be passed

Value

vector of residuals of the form e = y - Xb - Zu, the so called conditional residuals.

Author(s)

Giovanny Covarrubias

See Also

`residuals`, `mmec`

residuals.mmer *Residuals form a GLMM fitted with mmer*

Description

`residuals` method for class "`mmer`".

Usage

```
## S3 method for class 'mmer'
residuals(object, ...)
```

Arguments

`object`	an object of class "`mmer`"
`...`	Further arguments to be passed

Value

vector of residuals of the form e = y - Xb - Zu, the so called conditional residuals.

Author(s)

Giovanny Covarrubias

See Also

`residuals`, `mmer`

rrc *reduced rank covariance structure*

Description

`rrc` creates a reduced rank factor analytic covariance structure 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 `mmec` solver to fit random regressions on the latent covariates.

Usage

```
rrc(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 rrc 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 dsc (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.

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

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

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

See Also

The function `vsc` to know how to use `rrc` in the `mmec` solver.

Examples

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



# fit diagonal model first to produce H matrix
ansDG <- mmec(y~Env,
              random=~ vsc(dsc(Env), isc(Name)),
              rcov=~units, nIters = 100,
              # we recommend giving more EM iterations at the beggining
```

```
              emWeight = c(rep(1,10),logspace(10,1,.05), rep(.05,80)),
              data=DT)

H0 <- ansDG$uList$`vsc(dsc(Env), isc(Name))` # GxE table

# reduced rank model
ansFA <- mmec(y~Env,
              random=~vsc( usc(rrc(Env, H = H0, nPC = 3)) , isc(Name)) + # rr
                vsc(dsc(Env), isc(Name)), # diag
              rcov=~units,
              # we recommend giving more iterations to these models
              nIters = 100,
              # we recommend giving more EM iterations at the beggining
              emWeight = c(rep(1,10),logspace(10,1,.05), rep(.05,80)),
              data=DT)

vcFA <- ansFA$theta[[1]]
vcDG <- ansFA$theta[[2]]

loadings=with(DT, rrc(Env, nPC = 3, H = H0, returnGamma = TRUE) )$Gamma
scores <- ansFA$uList[[1]]

vcUS <- loadings %*% vcFA %*% t(loadings)
G <- vcUS + vcDG
# colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
# hv <- heatmap(cov2cor(G), col = colfunc(100), symm = TRUE)

uFA <- scores %*% t(loadings)
uDG <- ansFA$uList[[2]]
u <- uFA + uDG
```

`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

See Also

mmer – the core function of the package

Examples

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

---

spl2Da *Two-dimensional penalised tensor-product of marginal B-Spline basis.*

Description

Auxiliary function used for modelling the spatial or environmental effect as a two-dimensional penalised tensor-product (isotropic approach) based on Lee et al. (2013) and Rodriguez-Alvarez et al. (2018). This is a modified wrapper of some portions of the SpATS package to build a single incidence matrix containing all the columns from tensor products of the x and y coordinates and it fits such matrix as a single random effect. Then the heterogeneous covariances structure capabilities of sommer can be used to enhance the model fit. You may be interested in reading and citing not only sommer but also Wageningen publications if using this 2D spline methodology.

Usage

```
spl2Da(x.coord,y.coord,at.var=NULL,at.levels=NULL, type="PSANOVA",
      nsegments = c(10,10), penaltyord = c(2,2), degree = c(3,3),
      nestorder = c(1,1))
```

Arguments

x.coord	vector of coordinates on the x-axis direction (i.e. row) to use in the 2 dimensional spline.
y.coord	vector of coordinates on the y-axis direction (i.e. range or column) to use in the 2 dimensional spline.
at.var	vector of indication variable where heterogeneous variance is required (e.g., a different spl2D for each field).
at.levels	character vector with the names of the leves for the at term that should be used, if missing all levels are used.
type	one of the two methods "PSANOVA" or "SAP". See details below.
nsegments	numerical vector of length 2 containing the number of segments for each marginal (strictly nsegments - 1 is the number of internal knots in the domain of the covariate). Atomic values are also valid, being recycled. Default set to 10.
penaltyord	numerical vector of length 2 containing the penalty order for each marginal. Atomic values are also valid, being recycled. Default set to 2 (second order). Currently, only second order penalties are allowed.
degree	numerical vector of length 2 containing the order of the polynomial of the B-spline basis for each marginal. Atomic values are also valid, being recycled. Default set to 3 (cubic B-splines).
nestorder	numerical vector of length 2 containing the divisor of the number of segments (nsegments) to be used for the construction of the nested B-spline basis for the smooth-by-smooth interaction component. In this case, the nested B-spline basis will be constructed assuming a total of nsegments/nestorder segments. Default set to 1, which implies that nested basis are not used. See SAP for more details.

Details

The following documentation is taken from the SpATS package. Please refer to this package and associated publications if you are interested in going deeper on this technique:

Within the P-spline framework, anisotropic low-rank tensor-product smoothers have become the general approach for modelling multidimensional surfaces (Eilers and Marx 2003; Wood 2006). In the original SpATS package, was proposed to model the spatial or environmental effect by means of the tensor-product of B-splines basis functions. In other words, was proposed to model the spatial trend as a smooth bivariate surface jointly defined over the the spatial coordinates. Accordingly, the current function has been designed to allow the user to specify the spatial coordinates that the spatial trend is a function of. There is no restriction about how the spatial coordinates shall be specified: these can be the longitude and latitude of the position of the plot on the field or the column and row numbers. The only restriction is that the variables defining the spatial coordinates should be numeric (in contrast to factors).

As far as estimation is concerned, we have used in this package the equivalence between P-splines and linear mixed models (Currie and Durban, 2002). Under this approach, the smoothing parameters are expressed as the ratio between variance components. Moreover, the smooth components are decomposed in two parts: one which is not penalised (and treated as fixed) and one with is penalised (and treated as random). For the two-dimensional case, the mixed model representation leads also to a very interesting decomposition of the penalised part of the bivariate surface in three different components (Lee and Durban, 2011): (a) a component that contains the smooth main effect (smooth trend) along one of the covariates that the surface is a function of (as, e.g, the x-spatial coordinate or column position of the plot in the field), (b) a component that contains the smooth main effect (smooth trend) along the other covariate (i.e., the y-spatial coordinate or row position); and (c) a smooth interaction component (sum of the linear-by-smooth interaction components and the smooth-by-smooth interaction component).

The original implementation of SpATS assumes two different smoothing parameters, i.e., one for each covariate in the smooth component. Accordingly, the same smoothing parameters are used for both, the main effects and the smooth interaction. However, this approach can be extended to deal with the ANOVA-type decomposition presented in Lee and Durban (2011). In their approach, four different smoothing parameters are considered for the smooth surface, that are in concordance with the aforementioned decomposition: (a) two smoothing parameter, one for each of the main effects; and (b) two smoothing parameter for the smooth interaction component.

It should be noted that, the computational burden associated with the estimation of the two-dimensional tensor-product smoother might be prohibitive if the dimension of the marginal bases is large. In these cases, Lee et al. (2013) propose to reduce the computational cost by using nested bases. The idea is to reduce the dimension of the marginal bases (and therefore the associated number of parameters to be estimated), but only for the smooth-by-smooth interaction component. As pointed out by the authors, this simplification can be justified by the fact that the main effects would in fact explain most of the structure (or spatial trend) presented in the data, and so a less rich representation of the smooth-by-smooth interaction component could be needed. In order to ensure that the reduced bivariate surface is in fact nested to the model including only the main effects, Lee et al. (2013) show that the number of segments used for the nested basis should be a divisor of the number of segments used in the original basis (`nsegments` argument). In the present function, the divisor of the number of segments is specified through the argument `nestorder`. For a more detailed review on this topic, see Lee (2010) and Lee et al. (2013). The "PSANOVA" approach represents an alternative method. In this case, the smooth bivariate surface (or spatial trend) is decomposed in five different components each of them depending on a single smoothing parameter (see Lee et al., 2013).

As mentioned at the beginning, the piece of documentation stated above was taken completely from the SpATS package in order to provide a deeper explanation. In practice, sommer uses some pieces of code from SpATS to build the design matrix containing all the columns from tensor products of the x and y coordinates and it fits such matrix as a single random effect. As a result the same variance component is assumed for the linear, linear by linear, linear by spline, and spline by spline interactions. This results in a less flexible approach than the one proposed by Rodriguez-Alvarez et al. (2018) but still makes a pretty good job to model the spatial variation. Use under your own risk.

References

Rodriguez-Alvarez, M.X, Boer, M.P., van Eeuwijk, F.A., and Eilers, P.H.C. (2018). SpATS: Spatial Analysis of Field Trials with Splines. R package version 1.0-9. https://CRAN.R-project.org/package=SpATS.

Rodriguez-Alvarez, M.X., et al. (2015) Fast smoothng parameter separaton n multdmensonal generalzed P-splnes: the SAP algorthm. Statistics and Computing 25.5: 941-957.

Lee, D.-J., Durban, M., and Eilers, P.H.C. (2013). Efficient two-dimensional smoothing with P-spline ANOVA mixed models and nested bases. Computational Statistics and Data Analysis, 61, 22 - 37.

Gilmour, A.R., Cullis, B.R., and Verbyla, A.P. (1997). Accounting for Natural and Extraneous Variation in the Analysis of Field Experiments. Journal of Agricultural, Biological, and Environmental Statistics, 2, 269 - 293.

See Also

mmer, spl2Db

Examples

```
## ============================ ##
## example to use spl2Da()
## ============================ ##
data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# A <- A.mat(GT)
# mix <- mmer(Yield~1,
#             random=~vsr(id, Gu=A) +
#               vsr(Rowf) +
#               vsr(Colf) +
#               spl2Da(Row,Col),
#             rcov=~units,
#             data=DT)
# summary(mix)$varcomp
## ============================ ##
## mimic 2 fields
## ============================ ##
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
# A <- A.mat(GT)
# mix <- mmer(Yield~1,
#             random=~vsr(dsr(FIELD),id, Gu=A) +
#               vsr(dsr(FIELD),Rowf) +
#               vsr(dsr(FIELD),Colf) +
#                 spl2Da(Row,Col,at.var=FIELD),
#             rcov=~vsr(dsr(FIELD),units),
#             data=DT2)
```

spl2Db *Two-dimensional penalised tensor-product of marginal B-Spline basis.*

Description

Auxiliary function used for modelling the spatial or environmental effect as a two-dimensional penalised tensor-product (isotropic approach) based on Lee et al. (2013) and Rodriguez-Alvarez et al. (2018). spl2Db gets Tensor-Product P-Spline Mixed Model Incidence Matrices for use with sommer and its main function mmer. We thank Sue Welham for making the TPSbits package available to the community. If you're using this function for your research please cite her TPSbits package :) this is mostly a wrapper of her tpsmmb function to enable the use in sommer.

Usage

```
spl2Db(x.coord,y.coord,at.var=NULL,at.levels=NULL,nsegments = c(10,10),
       degree = c(3,3), penaltyord = c(2,2), nestorder = c(1,1),
       minbound=NULL, maxbound=NULL, method="Lee", what="bits")
```

Arguments

x.coord	vector of coordinates on the x-axis direction (i.e. row) to use in the 2 dimensional spline.
y.coord	vector of coordinates on the y-axis direction (i.e. range or column) to use in the 2 dimensional spline.
at.var	vector of indication variable where heterogeneous variance is required (e.g., a different spl2D for each field).
at.levels	character vector with the names of the leves for the at term that should be used, if missing all levels are used.
nsegments	numerical vector of length 2 containing the number of segments for each marginal (strictly nsegments - 1 is the number of internal knots in the domain of the covariate). Atomic values are also valid, being recycled. Default set to 10.
degree	numerical vector of length 2 containing the order of the polynomial of the B-spline basis for each marginal. Atomic values are also valid, being recycled. Default set to 3 (cubic B-splines).
penaltyord	numerical vector of length 2 containing the penalty order for each marginal. Atomic values are also valid, being recycled. Default set to 2 (second order). Currently, only second order penalties are allowed.
nestorder	numerical vector of length 2 containing the divisor of the number of segments (nsegments) to be used for the construction of the nested B-spline basis for the smooth-by-smooth interaction component. In this case, the nested B-spline basis will be constructed assuming a total of nsegments/nestorder segments. Default set to 1, which implies that nested basis are not used. See SAP for more details.
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.
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.
what	one of two values; 'base' or 'bits' to return: base = matrix for columns cbind(TP.col,TP.row,TP.C.n,TP.R.n,TP.CR.n). To be used in the fixed part. bits = matrices for the tensor products. To be used in the random part.

Details

The following documentation is taken from the SpATS package. Please refer to this package and associated publications if you are interested in going deeper on this technique:

Within the P-spline framework, anisotropic low-rank tensor-product smoothers have become the general approach for modelling multidimensional surfaces (Eilers and Marx 2003; Wood 2006). In the original SpATS package, was proposed to model the spatial or environmental effect by means of the tensor-product of B-splines basis functions. In other words, was proposed to model the spatial trend as a smooth bivariate surface jointly defined over the the spatial coordinates. Accordingly, the current function has been designed to allow the user to specify the spatial coordinates that the spatial trend is a function of. There is no restriction about how the spatial coordinates shall be specified: these can be the longitude and latitude of the position of the plot on the field or the column and row numbers. The only restriction is that the variables defining the spatial coordinates should be numeric (in contrast to factors).

As far as estimation is concerned, we have used in this package the equivalence between P-splines and linear mixed models (Currie and Durban, 2002). Under this approach, the smoothing parameters are expressed as the ratio between variance components. Moreover, the smooth components are decomposed in two parts: one which is not penalised (and treated as fixed) and one with is penalised (and treated as random). For the two-dimensional case, the mixed model representation leads also to a very interesting decomposition of the penalised part of the bivariate surface in three different components (Lee and Durban, 2011): (a) a component that contains the smooth main effect (smooth trend) along one of the covariates that the surface is a function of (as, e.g, the x-spatial coordinate or column position of the plot in the field), (b) a component that contains the smooth main effect (smooth trend) along the other covariate (i.e., the y-spatial coordinate or row position); and (c) a smooth interaction component (sum of the linear-by-smooth interaction components and the smooth-by-smooth interaction component).

The default implementation assumes two different smoothing parameters, i.e., one for each covariate in the smooth component. Accordingly, the same smoothing parameters are used for both, the main effects and the smooth interaction. However, this approach can be extended to deal with the ANOVA-type decomposition presented in Lee and Durban (2011). In their approach, four different smoothing parameters are considered for the smooth surface, that are in concordance with the aforementioned decomposition: (a) two smoothing parameter, one for each of the main effects; and (b) two smoothing parameter for the smooth interaction component.

It should be noted that, the computational burden associated with the estimation of the two-dimensional tensor-product smoother might be prohibitive if the dimension of the marginal bases is large. In these cases, Lee et al. (2013) propose to reduce the computational cost by using nested bases. The

idea is to reduce the dimension of the marginal bases (and therefore the associated number of parameters to be estimated), but only for the smooth-by-smooth interaction component. As pointed out by the authors, this simplification can be justified by the fact that the main effects would in fact explain most of the structure (or spatial trend) presented in the data, and so a less rich representation of the smooth-by-smooth interaction component could be needed. In order to ensure that the reduced bivariate surface is in fact nested to the model including only the main effects, Lee et al. (2013) show that the number of segments used for the nested basis should be a divisor of the number of segments used in the original basis (`nsegments` argument). In the present function, the divisor of the number of segments is specified through the argument `nestorder`. For a more detailed review on this topic, see Lee (2010) and Lee et al. (2013). The "PSANOVA" approach represents an alternative method. In this case, the smooth bivariate surface (or spatial trend) is decomposed in five different components each of them depending on a single smoothing parameter (see Lee et al., 2013).

Value

List of length 7 elements:

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. `fR` = Xr1:Zc
3. `fC` = Xr2:Zc
4. `fR.C` = Zr:Xc1
5. `R.fC` = Zr:Xc2
6. `fR.fC` = Zc:Zr
7. `all` = Xr1:Zc | Xr2:Zc | Zr:Xc1 | Zr:Xc2 | Zc:Zr

References

Sue Welham (2021). TPSbits: Creates Structures to Enable Fitting and Examination of 2D Tensor-Product Splines using ASReml-R. R package version 1.0.0.

Rodriguez-Alvarez, M.X, Boer, M.P., van Eeuwijk, F.A., and Eilers, P.H.C. (2018). SpATS: Spatial Analysis of Field Trials with Splines. R package version 1.0-9. https://CRAN.R-project.org/package=SpATS.

Rodriguez-Alvarez, M.X., et al. (2015) Fast smoothng parameter separaton n multdmensonal generalzed P-splnes: the SAP algorthm. Statistics and Computing 25.5: 941-957.

Lee, D.-J., Durban, M., and Eilers, P.H.C. (2013). Efficient two-dimensional smoothing with P-spline ANOVA mixed models and nested bases. Computational Statistics and Data Analysis, 61, 22 - 37.

Gilmour, A.R., Cullis, B.R., and Verbyla, A.P. (1997). Accounting for Natural and Extraneous Variation in the Analysis of Field Experiments. Journal of Agricultural, Biological, and Environmental Statistics, 2, 269 - 293.

See Also

mmer, spl2Da

Examples

```
## ============================ ##
## example to use spl2Db()
## ============================ ##
data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# A <- A.mat(GT)
# mix <- mmer(Yield~1,
#             random=~vsr(id, Gu=A) +
#               vsr(Rowf) +
#               vsr(Colf) +
#               spl2Db(Row,Col),
#             rcov=~units,
#             data=DT)
# summary(mix)$varcomp
## ============================ ##
## mimic 2 fields
## ============================ ##
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
# A <- A.mat(GT)
# mix <- mmer(Yield~1,
#             random=~vsr(dsr(FIELD),id, Gu=A) +
#               vsr(dsr(FIELD),Rowf) +
#               vsr(dsr(FIELD),Colf) +
#                 spl2Db(Row,Col,at.var=FIELD),
#             rcov=~vsr(dsr(FIELD),units),
#             data=DT2)
```

spl2Dc *Two-dimensional penalised tensor-product of marginal B-Spline basis.*

Description

Auxiliary function used for modelling the spatial or environmental effect as a two-dimensional penalised tensor-product (isotropic approach) based on Lee et al. (2013) and Rodriguez-Alvarez et al. (2018). This is a modified wrapper of some portions of the SpATS package to build a single incidence matrix containing all the columns from tensor products of the x and y coordinates and it fits such matrix as a single random effect. Then the heterogeneous covariances structure capabilities of sommec can be used to enhance the model fit. You may be interested in reading and citing not only sommec but also Wageningen publications if using this 2D spline methodology.

Usage

```
spl2Dc(x.coord,y.coord,at.var=NULL,at.levels=NULL, type="PSANOVA",
      nsegments = c(10,10), penaltyord = c(2,2), degree = c(3,3),
      nestorder = c(1,1), thetaC=NULL, theta=NULL, sp=FALSE)
```

Arguments

x.coord	vector of coordinates on the x-axis direction (i.e. row) to use in the 2 dimensional spline.
y.coord	vector of coordinates on the y-axis direction (i.e. range or column) to use in the 2 dimensional spline.
at.var	vector of indication variable where heterogeneous variance is required (e.g., a different spl2D for each field).
at.levels	character vector with the names of the leves for the at term that should be used, if missing all levels are used.
type	one of the two methods "PSANOVA" or "SAP". See details below.
nsegments	numerical vector of length 2 containing the number of segments for each marginal (strictly nsegments - 1 is the number of internal knots in the domain of the covariate). Atomic values are also valid, being recycled. Default set to 10.
penaltyord	numerical vector of length 2 containing the penalty order for each marginal. Atomic values are also valid, being recycled. Default set to 2 (second order). Currently, only second order penalties are allowed.
degree	numerical vector of length 2 containing the order of the polynomial of the B-spline basis for each marginal. Atomic values are also valid, being recycled. Default set to 3 (cubic B-splines).
nestorder	numerical vector of length 2 containing the divisor of the number of segments (nsegments) to be used for the construction of the nested B-spline basis for the smooth-by-smooth interaction component. In this case, the nested B-spline basis will be constructed assuming a total of nsegments/nestorder segments. Default set to 1, which implies that nested basis are not used. See SAP for more details.

thetaC	an optional matrix for constraints in the variance components.
theta	an optional matrix for initial values of the variance components.
sp	a TRUE/FALSE statement to indicate if the VC from this structure should be multiplied by the scale parameter added in the mmec function through the addScaleParam argument in the mmec function .

Details

The following documentation is taken from the SpATS package. Please refer to this package and associated publications if you are interested in going deeper on this technique:

Within the P-spline framework, anisotropic low-rank tensor-product smoothers have become the general approach for modelling multidimensional surfaces (Eilers and Marx 2003; Wood 2006). In the original SpATS package, was proposed to model the spatial or environmental effect by means of the tensor-product of B-splines basis functions. In other words, was proposed to model the spatial trend as a smooth bivariate surface jointly defined over the the spatial coordinates. Accordingly, the current function has been designed to allow the user to specify the spatial coordinates that the spatial trend is a function of. There is no restriction about how the spatial coordinates shall be specified: these can be the longitude and latitude of the position of the plot on the field or the column and row numbers. The only restriction is that the variables defining the spatial coordinates should be numeric (in contrast to factors).

As far as estimation is concerned, we have used in this package the equivalence between P-splines and linear mixed models (Currie and Durban, 2002). Under this approach, the smoothing parameters are expressed as the ratio between variance components. Moreover, the smooth components are decomposed in two parts: one which is not penalised (and treated as fixed) and one with is penalised (and treated as random). For the two-dimensional case, the mixed model representation leads also to a very interesting decomposition of the penalised part of the bivariate surface in three different components (Lee and Durban, 2011): (a) a component that contains the smooth main effect (smooth trend) along one of the covariates that the surface is a function of (as, e.g, the x-spatial coordinate or column position of the plot in the field), (b) a component that contains the smooth main effect (smooth trend) along the other covariate (i.e., the y-spatial coordinate or row position); and (c) a smooth interaction component (sum of the linear-by-smooth interaction components and the smooth-by-smooth interaction component).

The original implementation of SpATS assumes two different smoothing parameters, i.e., one for each covariate in the smooth component. Accordingly, the same smoothing parameters are used for both, the main effects and the smooth interaction. However, this approach can be extended to deal with the ANOVA-type decomposition presented in Lee and Durban (2011). In their approach, four different smoothing parameters are considered for the smooth surface, that are in concordance with the aforementioned decomposition: (a) two smoothing parameter, one for each of the main effects; and (b) two smoothing parameter for the smooth interaction component.

It should be noted that, the computational burden associated with the estimation of the two-dimensional tensor-product smoother might be prohibitive if the dimension of the marginal bases is large. In these cases, Lee et al. (2013) propose to reduce the computational cost by using nested bases. The idea is to reduce the dimension of the marginal bases (and therefore the associated number of parameters to be estimated), but only for the smooth-by-smooth interaction component. As pointed out by the authors, this simplification can be justified by the fact that the main effects would in fact explain most of the structure (or spatial trend) presented in the data, and so a less rich representation of the smooth-by-smooth interaction component could be needed. In order to ensure that the

reduced bivariate surface is in fact nested to the model including only the main effects, Lee et al. (2013) show that the number of segments used for the nested basis should be a divisor of the number of segments used in the original basis (`nsegments` argument). In the present function, the divisor of the number of segments is specified through the argument `nestorder`. For a more detailed review on this topic, see Lee (2010) and Lee et al. (2013). The "PSANOVA" approach represents an alternative method. In this case, the smooth bivariate surface (or spatial trend) is decomposed in five different components each of them depending on a single smoothing parameter (see Lee et al., 2013).

As mentioned at the beginning, the piece of documentation stated above was taken completely from the SpATS package in order to provide a deeper explanation. In practice, sommec uses some pieces of code from SpATS to build the design matrix containing all the columns from tensor products of the x and y coordinates and it fits such matrix as a single random effect. As a result the same variance component is assumed for the linear, linear by linear, linear by spline, and spline by spline interactions. This results in a less flexible approach than the one proposed by Rodriguez-Alvarez et al. (2018) but still makes a pretty good job to model the spatial variation. Use under your own risk.

References

Rodriguez-Alvarez, M.X, Boer, M.P., van Eeuwijk, F.A., and Eilers, P.H.C. (2018). SpATS: Spatial Analysis of Field Trials with Splines. R package version 1.0-9. https://CRAN.R-project.org/package=SpATS.

Rodriguez-Alvarez, M.X., et al. (2015) Fast smoothng parameter separaton n multdmensonal generalzed P-splnes: the SAP algorthm. Statistics and Computing 25.5: 941-957.

Lee, D.-J., Durban, M., and Eilers, P.H.C. (2013). Efficient two-dimensional smoothing with P-spline ANOVA mixed models and nested bases. Computational Statistics and Data Analysis, 61, 22 - 37.

Gilmour, A.R., Cullis, B.R., and Verbyla, A.P. (1997). Accounting for Natural and Extraneous Variation in the Analysis of Field Experiments. Journal of Agricultural, Biological, and Environmental Statistics, 2, 269 - 293.

See Also

`mmec`, `spl2Db`

Examples

```
## ============================ ##
## example to use spl2Dc()
## ============================ ##
data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# A <- A.mat(GT)
# Ai <- as(solve(A + diag(1e-4,ncol(A),ncol(A))), Class="dgCMatrix")
# mix <- mmec(Yield~1,
#             random=~vsc(isc(id), Gu=Ai) +
#               Rowf + Colf +
#               spl2Dc(Row,Col),
```

```
#             rcov=~units,
#             data=DT)
# summary(mix)$varcomp
## ============================ ##
## mimic 2 fields
## ============================ ##
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
# mix <- mmec(Yield~1,
#             random=~vsc(dsc(FIELD),isc(id), Gu=Ai) +
#               vsc(dsc(FIELD),isc(Rowf)) +
#               vsc(dsc(FIELD),isc(Colf)) +
#                 spl2Dc(Row,Col,at.var=FIELD),
#             rcov=~vsc(dsc(FIELD),isc(units)),
#             data=DT2)
```

---

spl2Dmats *Get Tensor Product Spline Mixed Model Incidence Matrices*

Description

spl2Dmats gets Tensor-Product P-Spline Mixed Model Incidence Matrices for use with sommer and its main function mmer. We thank Sue Welham for making the TPSbits package available to the community. If you're using this function for your research please cite her TPSbits package :) this is mostly a wrapper of her tpsmmb function to enable the use in sommer.

Usage

```
spl2Dmats(
  x.coord.name,
  y.coord.name,
  data,
  at.name,
  at.levels,
  nsegments=NULL,
  minbound=NULL,
  maxbound=NULL,
  degree = c(3, 3),
  penaltyord = c(2,2),
  nestorder = c(1,1),
  method = "Lee"
)
```

Arguments

x.coord.name	A string. Gives the name of data element holding column locations.
y.coord.name	A string. Gives the name of data element holding row locations.
data	A dataframe. Holds the dataset to be used for fitting.
at.name	name of a variable defining if the 2D spline matrices should be created at different units (e.g., at different environments).
at.levels	a vector of names indicating which levels of the at.name variable should be used for fitting the 2D spline function.
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.
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.

Value

List of length 7 elements:

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. fR = Xr1:Zc
3. fC = Xr2:Zc
4. fR.C = Zr:Xc1
5. R.fC = Zr:Xc2
6. fR.fC = Zc:Zr
7. all = Xr1:Zc | Xr2:Zc | Zr:Xc1 | Zr:Xc2 | Zc:Zr

Examples

```
data("DT_cpdata")
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
#### create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix

M <- spl2Dmats(x.coord.name = "Col", y.coord.name = "Row", data=DT, nseg =c(14,21))
head(M$data)
# m1g <- mmer(Yield~1+TP.CR.2+TP.CR.3+TP.CR.4,
#             random=~Rowf+Colf+vsr(M$fC)+vsr(M$fR)+
#               vsr(M$fC.R)+vsr(M$C.fR)+vsr(M$fC.fR)+
#               vsr(id,Gu=A),
#             data=M$data, tolpar = 1e-6,
#             iters=30)
#
# summary(m1g)$varcomp
```

---

stackTrait *Stacking traits in a dataset*

Description

stackTrait creates a dataset stacking traits in the long format to be used with the mmec solver for multi-trait models.

Usage

```
stackTrait(data, traits)
```

Arguments

data	a data frame with traits in wide format.
traits	variable names corresponding to the traits that should be in the long format.

Value

$res a data frame with traits in long format.

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

See Also

The function vsc to know how to use stackTrait in the mmec solver.

Examples

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

DT2 <- stackTrait(DT, traits = c("Yield","Weight"))
head(DT2)
```

---

summary.mmec *summary form a GLMM fitted with mmec*

Description

summary method for class "mmec".

Usage

```
## S3 method for class 'mmec'
summary(object, ...)
```

Arguments

object	an object of class "mmec"
...	Further arguments to be passed

Value

vector of summary

Author(s)

Giovanny Covarrubias-Pazaran

See Also

summary, mmec

---

summary.mmer *summary form a GLMM fitted with mmer*

Description

summary method for class "mmer".

Usage

```
## S3 method for class 'mmer'
summary(object, ...)
```

Arguments

object	an object of class "mmer"
...	Further arguments to be passed

Value

vector of summary

Author(s)

Giovanny Covarrubias-Pazaran

See Also

summary, mmer

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 its main function mmer.

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).

---

tpsmmbwrapper *Get Tensor Product Spline Mixed Model Incidence Matrices*

Description

tpsmmbwrapper 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 its main function mmer.

Usage

```
tpsmmbwrapper(
  columncoordinates,
  rowcoordinates,
  data,
  nsegments=NULL,
  minbound=NULL,
  maxbound=NULL,
  degree = c(3, 3),
  penaltyord = c(2, 2),
  nestorder = c(1, 1),
  asreml = "mbf",
  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.
data	A dataframe. Holds the dataset to be used for fitting.
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).

---

transformConstraints *transformConstraints*

Description

transformConstraints takes a list of matrices with constraints and transforms all the non-zero values to the value desired. The purpose of this function is to make easy the transformation of initial constraints to a fixed-constraint list to be provided to a mixed model fitted with the mmer function.

Usage

```
transformConstraints(list0,value=1)
```

Arguments

list0	a list of matrices with constraints according to the rules specified in the vsr function (0: not to be estimated, 1: positive, 2:unconstrained, 3:fixed).
value	value to be used to replace all the non-zero values in the constraint matrices.

Value

$res a list with the modified constraint matrices.

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

See Also

The function vsr to know how to use transformConstraints in the mmer solver.

Examples

```
(a <- list(unsm(4), diag(4)))
transformConstraints(a, value=3)
```

---

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.

See Also

The core functions of the package mmer

Examples

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

---

unsm *unstructured indication matrix*

Description

unsm creates a square matrix with ones in the diagonals and 2's in the off-diagonals to quickly specify an unstructured constraint in the Gtc argument of the vsr function.

Usage

```
unsm(x, reps=NULL)
```

Arguments

x	integer specifying the number of traits to be fitted for a given random effect.
reps	integer specifying the number of times the matrix should be repeated in a list format to provide easily the constraints in complex models that use the ds(), us() or cs() structures.

Value

$res a matrix or a list of matrices with the constraints to be provided in the Gtc argument of the vsr function.

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

See Also

The function `vsr` to know how to use `unsm` in the `mmer` solver.

Examples

```
unsm(3)
unsm(3,2)
```

---

usc *unstructured covariance structure*

Description

`usc` creates an unstructured covariance structure for specific levels of the random effect to be used with the `mmec` solver.

Usage

```
usc(x, thetaC, theta)
```

Arguments

`x`	vector of observations for the random effect.
`thetaC`	an optional symmetric matrix for constraints in the variance-covariance components. The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define how the variance-covariance components should be estimated: 0: component will not be estimated 1: component will be estimated and constrained to be positive 2: component will be estimated and unconstrained 3: component will be fixed to the value provided in the theta argument
`theta`	an optional symmetric matrix for initial values of the variance-covariance components. When providing customized values, these values should be scaled with respect to the original variance. For example, to provide an initial value of 1 to a given variance component, theta would be built as: theta = matrix(1 / var(response))

The symmetric matrix should have as many rows and columns as the number of levels in the factor 'x'. The values in the matrix define the initial values of the variance-covariance components that will be subject to the constraints provided in thetaC. If not provided, initial values will be calculated as:

diag(ncol(mm))*.05 + matrix(.1,ncol(mm),ncol(mm))

where mm is the incidence matrix for the factor 'x'.

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

The function `vsc` to know how to use `usc` in the `mmec` solver.

Examples

```
x <- as.factor(c(1:5,1:5,1:5));x
usc(x)
## how to use the theta and thetaC arguments:
# data(DT_example)
# DT <- DT_example
# theta <- matrix(9:1,3,3);
# theta[lower.tri(theta)] <- t(theta)[lower.tri(theta)]
# theta # initial VCs
# thetaC <- fixm(3); thetaC # fixed VCs
# ans1 <- mmec(Yield~Env,
#              random= ~ vsc( usc(Env,theta = theta,thetaC = thetaC),isc(Name) ),
#              rcov= ~ units, nIters = 1,
#              data=DT)
# summary(ans1)$varcomp
```

usr *unstructured covariance structure*

Description

usr creates an unstructured covariance structure for specific levels of the random effect to be used with the mmer solver.

Usage

```
usr(x)
```

Arguments

x	vector of observations for the random effect.

Value

$res a list with the provided vector and the variance covariance structure expected for the levels of the random effect.

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

See Also

The function vsr to know how to use usr in the mmer solver.

Examples

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

vpredict *vpredict form of a LMM fitted with mmer*

Description

vpredict method for class "mmer".

Post-analysis procedure to calculate linear combinations of variance components. Its intended use is when the variance components are either simple variances or are variances and covariances in an unstructured matrix. The functions covered are linear combinations of the variance components (for example, phenotypic variance), a ratio of two components (for example, heritabilities) and the correlation based on three components (for example, genetic correlation).

The calculations are based on the estimated variance parameters and their variance matrix as represented by the inverse of the Fisher or Average information matrix. Note that this matrix has zero values for fixed variance parameters including those near the parameter space boundary.

The transform is specified with a formula. On the left side of the formula is a name for the transformation. On the right side of the formula is a transformation specified with shortcut names like 'V1', 'V2', etc. The easiest way to identify these shortcut names is to use 'summary(object)$varcomp'. The rows of this object can referred to with shortcuts 'V1', 'V2', etc. See the example below.

Usage

```
vpredict(object, transform)
## S3 method for class 'mmer'
vpredict(object, transform)
```

Arguments

object	a model fitted with the mmer function.
transform	a formula to calculate the function.

Details

The delta method (e.g., Lynch and Walsh 1998, Appendix 1; Ver Hoef 2012) uses a Taylor series expansion to approximate the moments of a function of parameters. Here, a second-order Taylor series expansion is implemented to approximate the standard error for a function of (co)variance parameters. Partial first derivatives of the function are calculated by algorithmic differentiation with deriv.

Though vpredict can calculate standard errors for non-linear functions of (co)variance parameters from a fitted mmer model, it is limited to non-linear functions constructed by mathematical operations such as the arithmetic operators +, -, *, / and ^, and single-variable functions such as exp and log. See deriv for more information.

Value

dd	the parameter and its standard error.

Author(s)

Giovanny Covarrubias

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

Lynch, M. and B. Walsh 1998. Genetics and Analysis of Quantitative Traits. Sinauer Associates, Inc., Sunderland, MA, USA.

Ver Hoef, J.M. 2012. Who invented the delta method? The American Statistician 66:124-127. DOI: 10.1080/00031305.2012.687494

See Also

vpredict, mmer

Examples

```
####============================================####
####============================================####
#### EXAMPLE 1
#### simple example with univariate models
####============================================####
####============================================####
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT)
# #### look at the data and fit the model
# head(DT)
# mix1 <- mmer(Yield~1,
#              random=~vsr(id,Gu=A),
#              data=DT)
# summary(mix1)$varcomp
# #### run the vpredict function
# vpredict(mix1, h2 ~ V1 / ( V1 + V2 ) )
#
# ####============================================####
# ####============================================####
# #### EXAMPLE 2
# #### simple example with multivariate models
# ####============================================####
# ####============================================####
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT)
```

```
# #### look at the data and fit the model
# head(DT)
# mix2 <- mmer(cbind(Yield,color)~1,
#               random=~vsr(id,Gu=A, Gt=unsm(2)),
#               rcov=~vsr(units, Gt=unsm(2)),
#              data=DT)
# summary(mix2)$varcomp
# ## genetic correlation
# vpredict(mix2, gen.cor ~ V2 / sqrt(V1*V3))
#
# ####==========================================####
# ####==========================================####
# #### EXAMPLE 3
# #### more complex multivariate model
# ####==========================================####
# ####==========================================####
# data(DT_btdata)
# DT <- DT_btdata
# mix3 <- mmer(cbind(tarsus, back) ~ sex,
#               random = ~ vsr(dam, Gtc=unsm(2)) + vsr(fosternest,Gtc=diag(2)),
#               rcov=~vsr(units,Gtc=unsm(2)),
#               data = DT)
# summary(mix3)$varcomp
# #### calculate the genetic correlation
# vpredict(mix3, gen.cor ~ V2 / sqrt(V1*V3))
#
# ####==========================================####
# ####==========================================####
# #### EXAMPLE 4
# #### going back to simple examples
# ####==========================================####
# ####==========================================####
# data(DT_btdata)
# DT <- DT_btdata
# mix4 <- mmer(tarsus ~ sex, random = ~ dam + fosternest,
#               data = DT)
# summary(mix4)$varcomp
# #### calculate the ratio and its SE
# vpredict(mix4, dam.prop ~ V1 / ( V1 + V2 + V3 ) )
```

---

vs *variance structure specification*

Description

vs is the main function to build the variance-covariance structure for the random effects to be fitted in the mmer solver.

Usage

```
vs(..., Gu=NULL, Gti=NULL, Gtc=NULL, reorderGu=TRUE, buildGu=TRUE)
```

Arguments

...	variance structure to be specified following the logic desired in the internal kronecker product. For example, if user wants to define a diagonal variance structure for the random effect 'genotypes'(g) with respect to a random effect 'environments'(e), this is:
	`var(g) = G.e @ I.g`
	being `G.e` a matrix containing the variance covariance components for g (genotypes) in each level of e (environments), `I.g` is the covariance among levels of g (genotypes; i.e. relationship matrix), and @ is the kronecker product. This would be specified in the mmer solver as:
	`random=~vs(dsr(e),g)`
	One strength of sommer is the ability to specify very complex structures with as many kronecker products as desired. For example:
	`var(g) = G.e @ G.f @ G.h @ I.g`
	is equivalent to
	`random=~vs(e,f,h,g)`
	where different covariance structures can be applied to the levels of `e,f,h` (i.e. `dsr`, `usr`, `csr`, `atr` or a combination of these). For more examples please see the vignettes 'sommer.start' available in the package.

Gu	matrix with the known variance-covariance values for the levels of the u.th random effect (i.e. relationship matrix among individuals or any other known covariance matrix). If `NULL`, then an identity matrix is assumed. The Gu matrix can have more levels than the ones present in the random effect linked to it but not the other way around. Otherwise, an error message of missing level in Gu will be returned.
Gti	matrix with dimensions t x t (t equal to number of traits) with initial values of the variance-covariance components for the random effect specified in the .... argument. If `NULL` the program will provide the initial values. The values need to be scaled, see Details section.
Gtc	matrix with dimensions t x t (t equal to number of traits) of constraints for the variance-covariance components for the random effect specified in the ... argument according to the following rules: `0`: not to be estimated `1`: estimated and constrained to be positive (i.e. variance component) `2`: estimated and unconstrained (can be negative or positive, i.e. covariance component) `3`: not to be estimated but fixed (value has to be provided in the `Gti` argument) In the multi-response scenario if the user doesn't specify this argument the default is to build an unstructured matrix (using the `unsm`() function). This argument needs to be used wisely since some covariance among responses may not make sense. Useful functions to specify constraints are; `diag`(), `unsm`(), `fixm`().

reorderGu — a TRUE/FALSE statement if the Gu matrix should be reordered based on the names of the design matrix of the random effect or passed with the custom order of the user. This may be important when fitting covariance components in a customized fashion. Only for advanced users.

buildGu — a TRUE/FALSE statement to indicate if the Gu matrix should be built in R when the value for the argument Gu=NULL. Repeat, only when when the value for the argument Gu is equal to NULL. In some cases when the incidence matrix is wide (e.g. rrBLUP models) the covariance structure is a huge p x p matrix that can be avoided when performing matrix operations. By setting this argument to FALSE it allows to skip forming this covariance matrix.

Details

When providing initial values in the Gti argument the user has to provide scaled variance component values. The user can provide values from a previous model by accessing the sigma_scaled output from an mmer model or if an specific value is desired the user can obtain the scaled value as:

```
m = x/var(y)
```

where x is the desired initial value and y is the response variable. You can find an example in the DT_cpdata dataset.

Value

$res a list with all neccesary elements (incidence matrices, known var-cov structures, unknown covariance structures to be estimated and constraints) to be used in the mmer solver.

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

Covarrubias-Pazaran G (2018) Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

See Also

The core function of the package: mmer

Examples

```
# Please use the function vsr() for mmer() and vsc() for mmec.
```

vsc *variance structure specification*

Description

vsc is the main function to build the variance-covariance structure for the random effects to be fitted in the mmec solver.

Usage

```
vsc(..., Gu=NULL, buildGu=TRUE, meN=1, meTheta=NULL, meThetaC=NULL,
    sp=FALSE, isFixed=FALSE, verbose=TRUE)
```

Arguments

...	variance structure to be specified following the logic desired in the internal kronecker product. For example, if user wants to define a diagonal variance structure for the random effect 'genotypes'(g) with respect to a random effect 'environments'(e), this is:
	`var(g) = G.e @ I.g`
	being `G.e` a matrix containing the variance covariance components for g (genotypes) in each level of e (environments), `I.g` is the covariance among levels of g (genotypes; i.e. relationship matrix), and @ is the kronecker product. This would be specified in the mmec solver as:
	`random=~vsc(dsc(e),g)`
	One strength of sommer is the ability to specify very complex structures with as many kronecker products as desired. For example:
	`var(g) = G.e @ G.f @ G.h @ I.g`
	is equivalent to
	`random=~vsc(e,f,h,g)`
	where different covariance structures can be applied to the levels of `e,f,h` (i.e. dsc, usc, csc, atr or a combination of these). For more examples please see the vignettes 'sommer.start' available in the package.

Gu	matrix with the inverse of a known variance-covariance values for the levels of the u.th random effect (e.g., the inverse of a relationship matrix among individuals or any other known inverse covariance matrix). If NULL, then an identity matrix is assumed. The Gu matrix can have more levels than the ones present in the random effect linked to it but not the other way around. Otherwise, an error message of missing level in Gu will be returned.
buildGu	a TRUE/FALSE statement to indicate if the Gu matrix should be built in R when the value for the argument Gu=NULL. Repeat, only when when the value for the argument Gu is equal to NULL. In some cases when the incidence matrix is wide (e.g. rrBLUP models) the covariance structure is a huge p x p matrix that can be avoided when performing matrix operations. By setting this argument to FALSE it allows to skip forming this covariance matrix.

meN	number of main effects in the variance structure. Is always counted from last to first.
meTheta	variance covariance matrix between the main effects desired. Just to be modified if the number of main effects is greater of 1 (e.g., indirect genetic effects).
meThetaC	constraints for the variance covariance matrix between the main effects desired. Just to be modified if the number of main effects is greater of 1 (e.g., indirect genetic effects).
sp	a TRUE/FALSE statement to indicate if the VC from this structure should be multiplied by the scale parameter added in the mmec function through the addScaleParam argument in the mmec function .
isFixed	a TRUE/FALSE statement to indicate if the vsc function is being used in the fixed part of the model. When TRUE, the function only returns the model matrix to avoid any error messages associated to returning all elements for a random effect. FALSE is the default since it is assumed to be used for a variance structure in a random effect.
verbose	a TRUE/FALSE statement to indicate if messages should be printed when special situations accur. For example, adding unphenotyped individuals to the incidence matrices when present in the relationship matrices.

Details

...

Value

$res a list with all neccesary elements (incidence matrices, known var-cov structures, unknown covariance structures to be estimated and constraints) to be used in the mmec solver.

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

Covarrubias-Pazaran G (2018) Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

See Also

The core function of the package: mmec

Examples

```
data(DT_example)
DT <- DT_example
DT=DT[with(DT, order(Env)), ]
```

```
A <- A_example

x <- as.character(unique(DT$Name))
DT <- droplevels(DT[which(!is.na(match(DT$Name, x[1:5]))),])
## ============================ ##
## example without structure
## ============================ ##
isc(DT$Name)
mix <- mmec(Yield~Env,
            random= ~ vsc(isc(Name)),
            rcov=~ units,
            nIters=3,
            data=DT)

## ============================ ##
## example to without structure but
## using covariance among levels in the
## random effect Name
## ============================ ##
Ai <- as(solve(A + diag(1e-4,ncol(A),ncol(A))), Class="dgCMatrix")
mix <- mmec(Yield~Env,
            random= ~ vsc(isc(Name), Gu=Ai),
            rcov=~ units,
            nIters=3,
            data=DT)
summary(mix)$varcomp
## ============================ ##
## example to use dsc() structure (DIAGONAL)
## ============================ ##
dsc(DT$Year)
mix <- mmec(Yield~Env,
            random= ~ vsc(dsc(Year),isc(Name)),
            rcov=~ vsc(dsc(Year),isc(units)),
            nIters=3,
            data=DT)
summary(mix)$varcomp
## ============================ ##
## example to use atc() structure (level-specific)
## ============================ ##
# unique(DT$Year)
# mix <- mmec(Yield~Env,
#             random= ~ vsc(atc(Year,c("2011","2012")),isc(Name)),
#             rcov=~ vsc(dsc(Year),isc(units)),
#             data=DT)
## ============================ ##
## example to use usc() structure (UNSTRUCTURED)
## ============================ ##
usc(DT$Year)
mix <- mmec(Yield~Env,
            random= ~ vsc(usc(Year),isc(Name)),
            rcov=~ vsc(dsc(Year),isc(units)),
            nIters = 3,
            data=DT)
```

```
## ============================ ##
## example using structure in fixed effect
## (notice the isFixed argument)
## ============================ ##
mix <- mmec(Yield~ vsc(atc(Env,"CA.2011"), isc(Name), isFixed = TRUE),
            rcov=~ units,
            nIters=3,
            data=DT)
```

---

vsr *variance structure specification*

Description

vsr is the main function to build the variance-covariance structure for the random effects to be fitted in the mmer solver.

Usage

```
vsr(..., Gu=NULL, Gti=NULL, Gtc=NULL, reorderGu=TRUE, buildGu=TRUE)
```

Arguments

...	variance structure to be specified following the logic desired in the internal kronecker product. For example, if user wants to define a diagonal variance structure for the random effect 'genotypes'(g) with respect to a random effect 'environments'(e), this is:
	`var(g) = G.e @ I.g`
	being G.e a matrix containing the variance covariance components for g (genotypes) in each level of e (environments), I.g is the covariance among levels of g (genotypes; i.e. relationship matrix), and @ is the kronecker product. This would be specified in the mmer solver as:
	`random=~vsr(dsr(e),g)`
	One strength of sommer is the ability to specify very complex structures with as many kronecker products as desired. For example:
	`var(g) = G.e @ G.f @ G.h @ I.g`
	is equivalent to
	`random=~vsr(e,f,h,g)`
	where different covariance structures can be applied to the levels of e,f,h (i.e. dsr, usr, csr, atr or a combination of these). For more examples please see the vignettes 'sommer.start' available in the package.

Gu	matrix with the known variance-covariance values for the levels of the u.th random effect (i.e. relationship matrix among individuals or any other known covariance matrix). If NULL, then an identity matrix is assumed. The Gu matrix can have more levels than the ones present in the random effect linked to it but not the other way around. Otherwise, an error message of missing level in Gu will be returned.

Gti — matrix with dimensions t x t (t equal to number of traits) with initial values of the variance-covariance components for the random effect specified in the .... argument. If NULL the program will provide the initial values. The values need to be scaled, see Details section.

Gtc — matrix with dimensions t x t (t equal to number of traits) of constraints for the variance-covariance components for the random effect specified in the ... argument according to the following rules:

0: not to be estimated

1: estimated and constrained to be positive (i.e. variance component)

2: estimated and unconstrained (can be negative or positive, i.e. covariance component)

3: not to be estimated but fixed (value has to be provided in the Gti argument)

In the multi-response scenario if the user doesn't specify this argument the default is to build an unstructured matrix (using the unsm() function). This argument needs to be used wisely since some covariance among responses may not make sense. Useful functions to specify constraints are; diag(), unsm(), fixm().

reorderGu — a TRUE/FALSE statement if the Gu matrix should be reordered based on the names of the design matrix of the random effect or passed with the custom order of the user. This may be important when fitting covariance components in a customized fashion. Only for advanced users.

buildGu — a TRUE/FALSE statement to indicate if the Gu matrix should be built in R when the value for the argument Gu=NULL. Repeat, only when when the value for the argument Gu is equal to NULL. In some cases when the incidence matrix is wide (e.g. rrBLUP models) the covariance structure is a huge p x p matrix that can be avoided when performing matrix operations. By setting this argument to FALSE it allows to skip forming this covariance matrix.

Details

When providing initial values in the Gti argument the user has to provide scaled variance component values. The user can provide values from a previous model by accessing the sigma_scaled output from an mmer model or if an specific value is desired the user can obtain the scaled value as:

```
m = x/var(y)
```

where x is the desired initial value and y is the response variable. You can find an example in the DT_cpdata dataset.

Value

$res a list with all neccesary elements (incidence matrices, known var-cov structures, unknown covariance structures to be estimated and constraints) to be used in the mmer solver.

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

Covarrubias-Pazaran G (2018) Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: https://doi.org/10.1101/354639

See Also

The core function of the package: mmer

Examples

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

## ============================ ##
## example to without structure
## ============================ ##
dsr(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vsr(Name),
            rcov=~ vsr(units),
            data=DT)

## ============================ ##
## example to without structure but
## using covariance among levels in the
## random effect Name
## ============================ ##
dsr(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vsr(Name, Gu=A),
            rcov=~ vsr(units),
            data=DT)

## ============================ ##
## example to use dsr() structure (DIAGONAL)
## ============================ ##
dsr(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vsr(dsr(Year),Name),
            rcov=~ vsr(dsr(Year),units),
            data=DT)

## ============================ ##
## example to use atr() structure (level-specific)
## ============================ ##
unique(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vsr(atr(Year,c("2011","2012")),Name),
```

```
            rcov=~ vsr(dsr(Year),units),
            data=DT)

## ============================ ##
## example to use usr() structure (UNSTRUCTURED)
## ============================ ##
usr(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vsr(usr(Year),Name),
            rcov=~ vsr(dsr(Year),units),
            data=DT)

## ============================ ##
## example to use csr() structure (CUSTOMIZED)
## ============================ ##
unique(DT$Year)
mm <- matrix(1,3,3); mm[1,3] <- mm[3,1] <- 0;mm #don't estimate cov 2011-2013
mix <- mmer(Yield~Env,
            random= ~ vsr(csr(Year,mm),Name),
            rcov=~ vsr(dsr(Year),units),
            data=DT)

## ============================ ##
## example to use overlay() + vsr() structure
## ============================ ##
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)
A <- diag(7); colnames(A) <- rownames(A) <- 1:7;A # if you want to provide a covariance matrix
#### model using overlay
modh <- mmer(sugar~1,
             random=~vsr(overlay(femalef,malef,sparse=FALSE), Gu=A)
                    + genof,
              data=DT)

## ============================ ##
## example to use vsr() + dsr() + spl2D() structure
## ============================ ##
# ### mimic two fields
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
#
```

```
# mix <- mmer(Yield~1,
#             random=~vsr(dsr(FIELD),id, Gu=A) +
#               vsr(dsr(FIELD),Rowf) +
#               vsr(dsr(FIELD),Colf) +
#                 vsr(dsr(FIELD),spl2D(Row,Col)),
#             rcov=~vsr(dsr(FIELD),units),
#             data=DT2)
```

`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(DT_yatesoats)
DT <- DT_yatesoats

m3 <- mmer(fixed=Y ~ V + N + V:N-1,
           random = ~ B + B:MP,
           rcov=~units,
           data = DT)

wald.test(b = m3$Beta$Estimate, Sigma = m3$VarBeta, Terms = 2)

LL <- matrix(0,nrow=1, ncol=12)
LL[1,2] <- 1
LL[1,3] <- -1
LL

wald.test(b = m3$Beta$Estimate, Sigma = m3$VarBeta, L=LL)
```

Index