Package 'sommer'

July 8, 2017

Type Package

Title Solving Mixed Model Equations in R

Version 2.9

Date 2017-08-01

Author Giovanny Covarrubias-Pazaran

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

Description Multivariate linear mixed model solver for estimation of heterogeneous variances and specification of variance covariance structures. Maximum and Restricted Maximum Likelihood (ML/REML) estimates can be obtained using the Direct-Inversion Newton-Raphson (NR), Direct-Inversion Average Information (AI), MME-based Expectation-Maximization (EM), and Efficient Mixed Model Association (EMMA) algorithms. Designed for genomic prediction and genome wide association studies (GWAS) to include additive, dominance and epistatic relationship structures or other covariance structures in R, but also functional as a regular multivariate mixed model software. Multivariate models (multiple responses) can be fitted currently with NR, AI and EMMA algorithms.

Depends R (>= 2.10), Matrix (>= 1.1.1), methods, stats, MASS, parallel

License GPL-3

URL http://www.wisc.edu

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-08 21:18:19 UTC

Suggests knitr, plyr

VignetteBuilder knitr

R topics documented:

sommer-package *Solving Mixed Model Equations in R*

Description

Multivariate-Univariate linear mixed model solver for multiple random effects allowing the specification of variance covariance structures. ML/REML estimates can be obtained using one of the following methods; the **Direct-Inversion Newton-Raphson**, **Direct-Inversion Average Information**, the **MME-based Expectation-Maximization**, **MME-based Average Information**, or the **Efficient Mixed Model Association** algorithms. Designed for genomic prediction and genome wide association studies (GWAS) to include additive, dominance and epistatic relationship structures or other covariance structures, but also functional as a regular mixed model program. Multivariate models (multiple responses) can be fitted currently with any of the previously mentioned direct-inversion algorithms. In addition, **rrBLUP results can be recreated selecting the EMMA method**.

The sommer package has been developed to provide R users with open-source code to understand how most popular likelihood algorithms in mixed model analysis work, related to genetic analysis and other general experiments, but at the same time allowing to perform their real analysis in diploid and polyploid organisms with small and big data sets. The core of the package is the function `mmer` focused in solving multivariate linear mixed models by likelihood methods. **There's 2 ways of fitting a model, specifying all matrices using `mmer` or using the formula-based solver `mmer2`**.This package allows the user to estimate variance components for a mixed model with the advantage of specifying the variance-covariance structure of the random effects and obtain other parameters such as BLUPs, BLUEs, residuals, fitted values, variances-covariances for fixed and random effects, etc. The package is focused on genomic prediction (and genomic selection) and GWAS analysis, although general mixed models can be fitted as well. The package provides kernels to estimate additive (`A.mat`), dominance (`D.mat`), and epistatic (`E.mat`) relationship matrices for diploid and polyploid organisms that have been shown to increase prediction accuracy under certain scenarios. The package provides flexibility to fit other genetic models such as full and half diallel models as well. In addition the `pin` function can be used to estimate standard errors for linear combinations of variance components (i.e. ratios).

The package has been equiped with several datasets to learn how to use the sommer package; the `HDdata` and `FDdata` datasets will introduce users to fit half and full diallel designs respectively. The `h2` dataset shows how to calculate heritability. The `cornHybrid` and `Technow_data` datasets contain data to teach users how to perform genomic prediction in hybrid single crosses which display GCA and SCA effects. The `wheatLines` dataset teaches how to do genomic prediction in single crosses in species displaying only additive effects. The `CPdata` dataset contains data to teach users how to fit genomic prediction models within a biparental population coming from 2 highly heterozygous parents including additive, dominance and epistatic effects. The same data example is used to show how to include the top GWAS hits as fixed effects in the GBLUP model to increase prediction accuracy, the examples can be found in the `hits` documentation. The `PolyData` dataset shows how to fit genomic prediction and GWAS analysis in polyploids. The second example in `Technow_data` data shows how to perform GWAS in single cross hybrids. A good converter from letter code to numeric format is implemented in the function `atcg1234`, which supports higher ploidy levels than diploid. The `RICE` dataset can teach users how to select the best training population using the `TP.prep` function in an applied breeding program, and we show comparisons with other methods.

Traces of selection can be obtained using markers with the eigenGWAS function. A comparison of genomic prediction using univariate versus multivariate models is shown in the example#2 in the CPdata help page. Examples of multivariate models can be found in the example #3 of the CPdata. Furthermore, since v2.3 we have added the nna function which does nearest neighbor to create a new data frame adjusted for spatial variation. The ExpDesigns data contains several datasets to analyze experimental designs relevant to plant breeding and several detailed examples are available. Useful functions for analyzing such designs are included such as the blocker and fill.design. The gryphondata data contains an example of an animal model including pedigree information. The BTdata dataset contains an animal (birds) model to show the use of the pin function. Additional examples for fitting mixed models, such as GWAS and others, can be found in the example section of the mmer and mmer2 functions.

For a short tutorial of how to perform different genetic analysis in sommer **please type**:

vignette("sommer")

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 mmer function which is the core of the sommer package.

Additional functions for genetic analysis have been included such as False Discovery Rate calculation (fdr), plot of genetic maps (map.plot), creation of manhattan plots (manhattan), phasing F1 or CP populations (phase.F1). We have also included the famous transp function to get transparent colors.

Please update 'sommer' every month using:

install.packages("sommer")

If you have good knowledge of residual structures (REML estimates of residual structures) and are willing to help me to implement them in sommer please send me an email.

==========

MODELS ENABLED

The mmer function fits linear mixed models by likelihood methods allowing the used of known covariance structures for random effects. If not provided independence is conferred. This program focuses in the mixed model of the form:

,

$$Y = X\beta + Zu + \epsilon$$

,

with distributions:

,

$$Y\ MVN(X\beta + Zu, var(Zu + \epsilon))$$

where;

,

$$\beta\ N(\beta, 0)$$

$$u\ N(0, G)$$

where G is equal to:
,

K1*σ2(u1)	0	0
0	K2*σ2(u2)	0
...	...	...
0	0	Ki*σ2(ui)

,

for the i.th random effects, allowing the user to specify the variance covariance structures in the K matrices and
,

$$\epsilon\ N(0, R)$$

where: $R = I * \sigma 2\epsilon$
,

and Var(Y) = Var(Zu+e) = ZGZ+R = V which is the phenotypic variance. Estimation of variance components are optimized with any of the 4 methods available; NR, AI, EM, EMMA (rrBLUP method).

==========

GWAS MODELS

The GWAS models in the sommer package are enabled by using the W argument, 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 W 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 "W".

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

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

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 $Var[\varepsilon] = I\sigma_e^2$

For unbalanced designs where phenotypes come from different environments, the environment mean can be modeled using the fixed option (e.g., fixed="env" if the column in the pheno data.frame is called "env"). When principal components are included (P+K model), the loadings are determined from an eigenvalue decomposition of the K matrix.

The argument "P3D" was introduced by Zhang et al. (2010). When P3D=FALSE, this function is equivalent to EMMA with REML (Kang et al. 2008). When P3D=TRUE, it is equivalent to EMMAX (Kang et al. 2010). The P3D=TRUE option is faster but can underestimate significance compared to P3D=FALSE.

Multivariate GWAS are based in Covarrubias-Pazaran et al. (2017, Submitted), 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.

==========

As an example, imagine a mixed model with three random effects G takes the form:

,

$$\begin{matrix} K1*\sigma 2(gca1) & 0 & 0 \\ 0 & K2*\sigma 2(gca2) & 0 \\ 0 & 0 & K3*\sigma 2(sca) \end{matrix} = G$$

,

This mixed model would be specified in the `mmer` function as:

,

X1 <- matrix(1,length(y),1) # incidence matrix for intercept only

ETA <- list(

gca1=list(Z=Z1, K=K1),

gca2=list(Z=Z2, K=K2),

sca=list(Z=Z3, K=K3)

) #for 3 random effects

,

where Z1, Z2, Z3 are incidence matrices for GCA1, GCA2, SCA respectively created using the `model.matrix` function and K1, K2, K3 are their var-cov matrices. Now the fitted model will be typed as:

,

ans <- mmer(Y=Y, X=X1, Z=ETA)

or

,

or if a data frame is available:

,

ans <- mmer2(Y~1, random= ~ g(gca1) + g(gca2) + g(sca), G=list(gca1=K1, gca2=K2, sca=K3), data=yourdata)

==========

The mmer function has been enhanced by adding the argument EIGEND which allows an eigen decomposition of the additive relationship matrix to accelerate genomic prediction models in the order of hundreds of times compared to classical EMMA or AI when dense covariance structures.

Finally, feel free to get in touch with me if you have any questions or suggestions at:

cova_ruber@live.com.mx

I'll be glad to help or answer any question. We have spent a valuable amount of time developing this package. Please cite us 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

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.

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.

Abdollahi Arpanahi R, Morota G, Valente BD, Kranis A, Rosa GJM, Gianola D. 2015. Assessment of bagging GBLUP for whole genome prediction of broiler chicken traits. Journal of Animal Breeding and Genetics 132:218-228.

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

See Also

https://scholar.google.com/citations?user=Ic0qn-kAAAAJ&hl=en

Examples

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

```
#### remove them and run the examples
####============================================####

####============================================####
####============================================####
#### EXAMPLE 1
#### breeding values with 1 variance component
####============================================####
####============================================####

####============================================####
#### simulate genotypic data
#### 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) <- paste("geno",1:dim(M)[2], sep="")
rownames(M) <- paste("mark",1:dim(M)[1], sep="")
####============================================####
#### simulate phenotypes
####============================================####
QTL <- 100*(1:5) #pick 5 QTL
u <- rep(0,1000) #marker effects
u[QTL] <- 1
g <- as.vector(crossprod(M,u))
h2 <- 0.5
y <- g + rnorm(200,mean=0,sd=sqrt((1-h2)/h2*var(g)))
M <- t(M); M[1:5,1:5]

####============================================####
#### fit the model using mmer (matrix based)
####============================================####
Z1 <- diag(length(y))
K1 <-A.mat(M)
ETA <- list( add=list(Z=Z1, K=K1) )
#ans <- mmer(Y=y, Z=ETA)
#summary(ans)

#### run the same but as GWAS
#### just add the marker matrix in the argument W
#### markers are fixed effects

#ans <- mmer(Y=y, Z=ETA, W=M)
#summary(ans)

####============================================####
#### fit the model using mmer2 (formula-based)
####============================================####
K<-A.mat(M) # additive relationship matrix
dat <- data.frame(y=y,id=rownames(M));head(dat)
#ans <- mmer2(y~1, random=~g(id), G=list(id=K), data=dat)
```

```
#summary(ans)

#### run the same but as GWAS
#### just add the marker matrix in the argument W
#### markers are fixed effects

#ans <- mmer2(y~1, random=~g(id), G=list(id=K), W=M, data=dat)
#summary(ans)


##############################################
##############################################
##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####============================================####
####============================================####
#### EXAMPLE 2
#### breeding values with 3 variance components
####============================================####
####============================================####


####============================================####
#### Hybrid prediction using mmer (matrix-based)
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K
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)

colnames(Z1) <- levels(hybrid2$GCA1)
colnames(Z2) <- levels(hybrid2$GCA2)
colnames(Z3) <- levels(hybrid2$SCA)
####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
#K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
#K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####============================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
```

```
####=========================================####
#S <- kronecker(K1, K2) ; dim(S)
#rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

#ETA <- list(GCA1=list(Z=Z1, K=K1),
#            GCA2=list(Z=Z2, K=K2),
#            SCA=list(Z=Z3, K=S)
#            )
#ans <- mmer(Y=y, X=X1, Z=ETA)
#ans$var.comp
#summary(ans)


####=========================================####
#### Hybrid prediction using mmer2 (formula-based)
####=========================================####
# data(cornHybrid)
# hybrid2 <- cornHybrid$hybrid # extract cross data
# A <- cornHybrid$K
# ####=========================================####
# #### Realized IBS relationships for set of parents 1
# ####=========================================####
# K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
# ####=========================================####
# #### Realized IBS relationships for set of parents 2
# ####=========================================####
# K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
# ####=========================================####
# #### Realized IBS relationships for cross
# #### (as the Kronecker product of K1 and K2)
# ####=========================================####
# S <- kronecker(K1, K2) ; dim(S)
# rownames(S) <- colnames(S) <- levels(hybrid2$SCA)
#
# ans <- mmer2(Yield~Location, random=~g(GCA1)+ g(GCA2) + g(SCA),
#              G=list(GCA1=K1, GCA2=K2, SCA=S), data=hybrid2)
# summary(ans)

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####=========================================####
####=========================================####
#### EXAMPLE 3
#### PREDICTING SPECIFIC PERFORMANCE
#### within biparental population
####=========================================####
```

```
####=========================================####

####=========================================####
#### using mmer (matrix-based)
####=========================================####
data(CPdata)
CPpheno <- CPdata$pheno[,-c(1:4)]
CPgeno <- CPdata$geno
#### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
#### fit a model including additive and dominance effects
y <- CPpheno$color
Za <- diag(length(y))
Zd <- diag(length(y))
Ze <- diag(length(y))
#A <- A.mat(CPgeno) # additive relationship matrix
#D <- D.mat(CPgeno) # dominant relationship matrix

####================####
#### ADDITIVE MODEL ####
####================####
#ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- mmer(Y=y, Z=ETA.A)
#summary(ans.A)
####============================####
#### ADDITIVE-DOMINANCE MODEL ####
####============================####
#ETA.AD <- list(add=list(Z=Za,K=A), dom=list(Z=Zd,K=D))
#ans.AD <- mmer(Y=y, Z=ETA.AD)
#summary(ans.AD)
### greater accuracy !!!! 4 percent increment!!
### we run 100 iterations, 4 percent increment in general

####=========================================####
#### same using mmer2 (formula-based)
####=========================================####

# data(CPdata)
# CPpheno <- CPdata$pheno
# CPgeno <- CPdata$geno
# ### look at the data
# head(CPpheno); CPgeno[1:5,1:5]
# ###=========================================####
# ### fit a model including additive and dominance effects
# ###=========================================####
# A <- A.mat(CPgeno)
# D <- D.mat(CPgeno)
# ####=========================================####
# #### ADDITIVE MODEL
# ####=========================================####
# ans.A <- mmer2(Yield~1,random=~ g(id), G=list(id=A),
#                data=CPpheno)
```

```
# summary(ans.A)
# ####=========================================####
# #### ADDITIVE-DOMINANCE MODEL
# ####=========================================####
# CPpheno$idd <- CPpheno$id
# ans.AD <- mmer2(Yield~1,random=~ g(id) + g(idd),
#                 G=list(id=A, idd=D), data=CPpheno)
# summary(ans.AD)

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####=========================================####
####=========================================####
#### EXAMPLE 4
#### PARALLEL RESPONSES
#### accelerating analysis
#### using 'n.cores' argument for parallelization
####=========================================####
####=========================================####

# data(CPdata)
# CPpheno <- CPdata$pheno; CPpheno$idd <- CPpheno$id
# CPgeno <- CPdata$geno
# ###=======================================####
# ### Do incidence and relationship matrices
# ###=======================================####
# A <- A.mat(CPgeno)
# D <- D.mat(CPgeno)
# ###=======================================####
# ### Only additive model using all traits
# ###=======================================####
# head(CPpheno)
# ans.A <- mmer2(cbind(color,Yield,FruitAver)~1, random = ~ g(id),
#               G=list(id=A), n.cores = 3, data=CPpheno)
# summary(ans.A)
# ###=======================================####
# ### Additive + dominance model
# ###=======================================####
# head(CPpheno)
# ans.AD <- mmer2(cbind(color,Yield,FruitAver)~1, random=~g(id)+g(idd),
#                 G=list(id=A, idd=D), n.cores = 3, data=CPpheno)
# summary(ans.AD)

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

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

####======================================####
####======================================####
#### EXAMPLE 5
#### MULTIVARIATE MODEL
####======================================####
####======================================####
# data(CPdata)
# CPpheno <- CPdata$pheno
# CPgeno <- CPdata$geno
# ### look at the data
# head(CPpheno)
# CPgeno[1:5,1:5]
# ## fit a model including additive effects
# A <- A.mat(CPgeno)
# ### MAKE SURE YOU SET 'MVM=TRUE'
# ans.A <- mmer2(cbind(color,Yield,FruitAver)~1, random = ~ g(id),
#               G=list(id=A),MVM=TRUE, data=CPpheno)
# summary(ans.A)

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####======================================####
####======================================####
#### EXAMPLE 6
#### OVERLAY IN MODELS
####======================================####
####======================================####


####======================================####
#### using mmer (matrix-based)
####======================================####
data(HDdata)
head(HDdata)

#### GCA matrix for half diallel using male and female columns

Z1 <- overlay(HDdata[,c("male","female")])

#### SCA matrix

Z2 <- model.matrix(~as.factor(geno)-1, data=HDdata)

#### Define response variable and run
```

```
y <- HDdata$sugar
ETA <- list(GCA=list(Z=Z1), SCA=list(Z=Z2)) # Zu component
modHD <- mmer(Y=y, Z=ETA)
summary(modHD)

####=========================================####
#### using overlay with mmer2 function (formula-based)
####=========================================####
# data(HDdata)
# head(HDdata)
# HDdata$female <- as.factor(HDdata$female)
# HDdata$male <- as.factor(HDdata$male)
# HDdata$geno <- as.factor(HDdata$geno)
# #### model using overlay
# modh <- mmer2(sugar~1, random=~female + and(male) + geno,
#               data=HDdata)
# summary(modh)

# #### model using overlay [and(.)] and covariance structures [g(.)]

# A <- diag(7); A[1,2] <- 0.5; A[2,1] <- 0.5 # fake covariance structure
# colnames(A) <- as.character(1:7); rownames(A) <- colnames(A);A
#
# modh2 <- mmer2(sugar~1, random=~g(female) + and(g(male)) + geno,
#               G=list(female=A, male=A),data=HDdata)
# summary(modh2)

# data(HDdata)
# head(HDdata)
# HDdata$female <- as.factor(HDdata$female)
# HDdata$male <- as.factor(HDdata$male)
# HDdata$geno <- as.factor(HDdata$geno)
# #### model using overlay
# modh <- mmer2(sugar~1, random=~female + and(male) + geno,
#               data=HDdata)
# summary(modh)
```

`A.mat` *Additive relationship matrix*

Description

Calculates the realized additive relationship matrix. Is a wrapper of the A.mat function from the rrBLUP package published by Endelman (2011).

Usage

```
A.mat(X,min.MAF=NULL,max.missing=NULL,impute.method="mean",tol=0.02,
      n.core=1,shrink=FALSE,return.imputed=FALSE, ploidy=2)
```

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.
max.missing	Maximum proportion of missing data; default removes completely missing markers.
impute.method	There are two options. The default is "mean", which imputes with the mean for each marker. The "EM" option imputes with an EM algorithm (see details).
tol	Specifies the convergence criterion for the EM algorithm (see details).
n.core	Specifies the number of cores to use for parallel execution of the EM algorithm (use only at UNIX command line).
shrink	Set shrink=TRUE to use the shrinkage estimation procedure (see Details).
return.imputed	When TRUE, the imputed marker matrix is returned.
ploidy	The ploidy of the organism. The default is 2 which means diploid but higher ploidy levels are supported.

Details

At high marker density, the relationship matrix is estimated as $A = WW'/c$, where $W_{ik} = X_{ik} + 1 - 2p_k$ and p_k is the frequency of the 1 allele at marker k. By using a normalization constant of $c = 2\sum_k p_k(1-p_k)$, the mean of the diagonal elements is $1 + f$ (Endelman and Jannink 2012).

The EM imputation algorithm is based on the multivariate normal distribution and was designed for use with GBS (genotyping-by-sequencing) markers, which tend to be high density but with lots of missing data. Details are given in Poland et al. (2012). The EM algorithm stops at iteration t when the RMS error = $n^{-1}\|A_t - A_{t-1}\|_2$ < tol.

At low marker density (m < n), shrinkage estimation can improve the estimate of the relationship matrix and the accuracy of GEBVs for lines with low accuracy phenotypes (Endelman and Jannink 2012). The shrinkage intensity ranges from 0 (no shrinkage, same estimator as high density formula) to 1 (completely shrunk to $(1 + f)I$). The shrinkage intensity is chosen to minimize the expected mean-squared error and printed to the screen as output.

The shrinkage and EM options are designed for opposite scenarios (low vs. high density) and cannot be used simultaneously.

When the EM algorithm is used, the imputed alleles can lie outside the interval [-1,1]. Polymorphic markers that do not meet the min.MAF and max.missing criteria are not imputed.

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

$imputed 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

Poland, J., J. Endelman et al. 2012. Genomic selection in wheat breeding using genotyping-by-sequencing. Plant Genome 5:103-113. doi: 10.3835/plantgenome2012.06.0006

See Also

mmer and mmer2– the core functions 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)
```

---

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` and `mmer2` – the core functions 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
```

AI *Average Information Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the average information (AI) algorithm.

Usage

```
AI(y, X=NULL, ZETA=NULL, R=NULL, draw=TRUE, REML=TRUE, silent=FALSE,
   iters=50, constraint=TRUE, init=NULL, sherman=FALSE, che=TRUE,
   EIGEND=FALSE, Fishers=FALSE, gss=TRUE, 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 matrix for variance-covariance structures for the 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

`REML` a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`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.

`sherman` a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components in order to perform faster when a mixed model with no covariance structure using the average information algorithm is fitted. The default is FALSE since this software was designed for unreplicated data (altough can fit models with replicated data but slower than lme4).

`che` 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.

`EIGEND` a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

`Fishers`	a TRUE/FALSE value indicating if the program should calculate at the final step and return the inverse of the Fishers Information Matrix.
`gss`	a TRUE/FALSE value indicating if a genomic selection is being fitted just for using certain constraints. When is FALSE (default) the program can make some EM steps to find initial values for variance components when the starting values are to far from the real values causing the likelihood to have a strange behavior and dropping dramatically When TRUE the program does not try EM steps even when far away from the likelihood because in big marker-based models can make the process quite slow.
`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 Lee et al. (2016), it is based on REML. 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 likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

The algorithm can be summarized in the next steps:

.

1) provide initial values for the variance components

2) estimate the phenotypic variance matrix V = ZGZ + R

3) obtain Vinv by inverting V

4) obtain the projection matrix P = Vinv - [Vinv X (X'V-X)- X Vinv]

5) evaluate the logLikelihood as shown above

6) fill the average information matrix (AI) with equation provided in Gilmour et al. (1995)

7) obtain AI.inv by inverting AI (the average information matrix)

8) calculate scores by first derivatives refer as "B" in Gilmour et al. (1995)

9) update the values of variance components by : k(i+1) = k(i) + [B(i) * AI.inv]

10) steps are repeated in a while loop until convergence is reached, the likelihood doesn't increase anymore.

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

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

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

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

See Also

The core functions of the package mmer and mmer2

Examples

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

####============================================####
#### breeding values with 3 variance components
####============================================####

####============================================####
## Import phenotypic data on inbred performance
## Full data
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # extract the var-cov K

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)

####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####============================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####============================================####
S <- kronecker(K1, K2) ; dim(S)
```

```
rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

ETA <- list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=S))
####=========================================####
#### run the next line, it was ommited for CRAN time limitations
####=========================================####
#ans <- AI(y=y, ZETA=ETA)
#ans$var.comp
```

---

ai2help *Average Information Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the average information (ai2help) algorithm.

Usage

```
ai2help(y, X=NULL, ZETA=NULL, R=NULL, draw=TRUE, REML=TRUE,
       silent=FALSE, iters=50, constraint=TRUE, init=NULL,
       sherman=FALSE, che=FALSE, EIGEND=FALSE, Fishers=FALSE,
       gss=TRUE, forced=NULL)
```

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 matrix for variance-covariance structures for the 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
REML	a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`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.

`sherman` a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components in order to perform faster when a mixed model with no covariance structure using the average information algorithm is fitted. The default is FALSE since this software was designed for unreplicated data (altough can fit models with replicated data but slower than lme4).

`che` 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.

`EIGEND` a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

`Fishers` a TRUE/FALSE value indicating if the program should calculate at the final step and return the inverse of the Fishers Information Matrix.

`gss` a TRUE/FALSE value indicating if a genomic selection is being fitted just for using certain constraints. When is FALSE (default) the program can make some EM steps to find initial values for variance components when the starting values are to far from the real values causing the likelihood to have a strange behavior and dropping dramatically When TRUE the program does not try EM steps even when far away from the likelihood because in big marker-based models can make the process quite slow.

`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.

Details

This algorithm is based on Gilmour et al. (1995), it is based on REML. 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 likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

The algorithm can be summarized in the next steps:

.

1) provide initial values for the variance components

2) estimate the phenotypic variance matrix V = ZGZ + R

3) obtain Vinv by inverting V

4) obtain the projection matrix P = Vinv - [Vinv X (X'V-X)- X Vinv]

5) evaluate the logLikelihood as shown above

6) fill the average information matrix (ai2help) with equation provided in Gilmour et al. (1995)

7) obtain ai2help.inv by inverting ai2help (the average information matrix)

8) calculate scores by first derivatives refer as "B" in Gilmour et al. (1995)

9) update the values of variance components by : k(i+1) = k(i) + [B(i) * ai2help.inv]

10) steps are repeated in a while loop until convergence is reached, the likelihood doesn't increase anymore.

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

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

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

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

See Also

The core functions of the package `mmer` and `mmer2`

Examples

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

####=========================================####
#### breeding values with 3 variance components
####=========================================####

####=========================================####
## Import phenotypic data on inbred performance
## Full data
####=========================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # extract the var-cov K

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)

####=========================================####
#### Realized IBS relationships for set of parents 1
####=========================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####=========================================####
#### Realized IBS relationships for set of parents 2
####=========================================####
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####=========================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####=========================================####
S <- kronecker(K1, K2) ; dim(S)
rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

ETA <- list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=S))
####=========================================####
#### run the next line, it was ommited for CRAN time limitations
####=========================================####
#ans <- ai2help(y=y, ZETA=ETA)
#ans$var.comp
```

AImme *Average Information Algorithm MME-based*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the average information (AI) algorithm and sets the argument 'DI=TRUE' to use the MME-based algorithm.

Usage

```
AImme(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.
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.
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.
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 Gilmour et al. (1995), it is based on REML. 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 likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

The algorithm can be summarized in the next steps:

.

1) provide initial values for the variance components

2) estimate the phenotypic variance matrix V = ZGZ + R

3) obtain Vinv by inverting V

4) obtain the projection matrix P = Vinv - [Vinv X (X'V-X)- X Vinv]

5) evaluate the logLikelihood as shown above

6) fill the average information matrix (AI) with equation provided in Gilmour et al. (1995)

7) obtain AI.inv by inverting AI (the average information matrix)

8) calculate scores by first derivatives refer as "B" in Gilmour et al. (1995)

9) update the values of variance components by : k(i+1) = k(i) + [B(i) * AI.inv]

10) steps are repeated in a while loop until convergence is reached, the likelihood doesn't increase anymore.

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

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

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

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

See Also

The core functions of the package `mmer` and `mmer2`

Examples

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

####=========================================####
#### breeding values with 3 variance components
####=========================================####

####=========================================####
## Import phenotypic data on inbred performance
## Full data
####=========================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # extract the var-cov K

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)

####=========================================####
#### Realized IBS relationships for set of parents 1
####=========================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####=========================================####
#### Realized IBS relationships for set of parents 2
####=========================================####
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####=========================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####=========================================####
S <- kronecker(K1, K2) ; dim(S)
rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

ETA <- list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=S))
####=========================================####
#### run the next line, it was ommited for CRAN time limitations
####=========================================####
#ans <- AImme(y=y, ZETA=ETA)
#ans$var.comp
```

---

and *and functionality*

Description

overlay (add) 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

```
and(x)
```

Arguments

x	the random effect to overlay.

Value

$x the random effect and the signal to mmer2 to do the overlay.

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 mmer2

Examples

```
####==========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####==========================================####
data(HDdata)
head(HDdata)
HDdata$female <- as.factor(HDdata$female)
HDdata$male <- as.factor(HDdata$male)
HDdata$geno <- as.factor(HDdata$geno)
#### model using overlay
modh <- mmer2(sugar~1, random=~female + and(male) + geno,
              data=HDdata)
summary(modh)

#####################################################################
#### model using overlay [and(.)] and covariance structures [g(.)]
```

```
#####################################################################
# A <- diag(7); A[1,2] <- 0.5; A[2,1] <- 0.5 # fake covariance structure
# colnames(A) <- as.character(1:7); rownames(A) <- colnames(A);A
#
# modh2 <- mmer2(sugar~1, random=~g(female) + and(g(male)) + geno,
#               G=list(female=A, male=A),data=HDdata)
# summary(modh2)
```

---

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, ...)
```

Arguments

object	an object of class "mmer"
object2	an object of class "mmer", if NULL the program will return an error specifying that this function was developed for comparing mixed models using likelihood ratio tests and no to provide regular sum of squares results.
...	Further arguments to be passed

Value

vector of anova

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

anova, mmer, and mmer2

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

Description

anova method for class "MMERM".

Usage

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

Arguments

object	an object of class "MMERM"
object2	an object of class "MMERM", if NULL the program will return an error specifying that this function was developed for comparing mixed models using likelihood ratio tests and no to provide regular sum of squares results.
...	Further arguments to be passed

Value

vector of anova

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

anova, mmer, and mmer2

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

Description

anova method for class "mmerM".

Usage

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

Arguments

object	an object of class "mmerM"
object2	an object of class "mmerM", if NULL the program will return an error specifying that this function was developed for comparing mixed models using likelihood ratio tests and no to provide regular sum of squares results.
...	Further arguments to be passed

Value

vector of anova

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

anova, mmer, and mmer2

---

at *at functionality*

Description

This function creates internally the incidence matrices for specific locations when using mmer2.

Usage

```
at(x, levs)
```

Arguments

x	a column of a dataframe.
levs	levels of the columns to keep after using the model.matrix() function.

Value

$dd an incidence matrix with columns for the specific levels specified.

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 mmer2

Examples

```
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
head(hybrid2)
modFD <- mmer2(Yield~1, random=~at(Location,c("3","4")):GCA2, data=hybrid2)
summary(modFD)
```

---

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).

Usage

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

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. 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.

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` and and `mmer2`

Examples

```
data(PolyData)
genotypes <- PolyData$PGeno
genotypes[1:5,1:5] # look the original format

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

####================================================####
#### 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(RICE)
```

```
#X <- RICE$RiceGeno; X[1:5,1:5]; dim(X)
#numo2 <- atcg1234(data=X, ploidy=2)
#numo2[1:5,1:5]; dim(numo2)
```

---

augment *augment design example.*

Description

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

Usage

```
data("augment")
```

Format

The format is: chr "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 and mmer2

Examples

```
####============================================####
#### AUGMENTED DESIGN EXAMPLE
####============================================####
data(augment)
head(augment)
####============================================####
#### fit the mixed model and check summary
####============================================####
mix1 <- mmer2(TSW ~ Check.Gen, random = ~ Block + Genotype:Check, data=augment)
summary(mix1)
blup <- mix1$u.hat$`Genotype:Check`
```

```
####=========================================####
#### get only the blups for unreplicated genotypes
####=========================================####
blup2 <- blup[grep("Check1",rownames(blup)),]
```

`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))
}
```

`big.peaks.col` *Peak search by first derivatives*

Description

This function find all peaks by taking the first derivative based on 'rle' function. Is used in different Fragma functions.

Usage

```
big.peaks.col(x,tre)
```

Arguments

x	A vector of heights or intensities
tre	A scalar value deciding when peaks will be ignored

Details

No major details.

Value

Retuns the biggest peaks for a vector of intensities.

out a vector of positions where the derivative is zero and therefore a peak was found

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.

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

---

blocker *Applying postblocking in a field*

Description

The blocker function takes a dataframe that contains columns indicating x and y coordinates and use them to create blocking with different numbers. For example, a square field with no blocks after applying the function will have several new columns for different blockings that are sequential, i.e. 2 blocks in one direction, then 4 blocks by blocking in the other direction and so on.

Usage

```
blocker(x,rows="ROWS",ranges="RANGES",by, char=FALSE)
```

Arguments

x	a dataframe with 2 obligatory columns; rows and ranges which can have different names and can be matched with the next arguments.
rows	the name of the numeric column that indicates one direction in the field.
ranges	the name of the numeric column that indicates the other direction in the field.
by	optional argument to indicate the name of the column of the dataframe x that indicates the environments so the field is filled by environment.
char	a TRUE/FALSE value indicating if the blocking column should be returned in a character class instead of numeric.

Value

$fin a new dataframe with new columns for different blocking proposals.

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 mmer2

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(CPdata)
CPpheno <- CPdata$pheno#[,-c(1:4)]
CPgeno <- CPdata$geno
#### look at the data
head(CPpheno)
#### fill the design
gg <- fill.design(x=CPpheno, rows="Row",ranges="Col")
head(gg)
#### apply the postblocking
gg2 <- blocker(x=gg, rows="Col",ranges="Row", char = TRUE)
head(gg2)
```

```
#### now you can use them in your mixed models
```

`brewer.pal` *Generate a sequence of colors for groups.*

Description

wrapper of brewer.pal function from RColorBrewer.

Usage

```
brewer.pal(n, name)
```

Arguments

`n`	Number of different colors in the palette, minimum 3, maximum depending on palette.
`name`	A palette name from the lists below: Accent 8 Dark2 8 Paired 12 Pastel1 9 Pastel2 8 Set1 9 Set2 8 Set3 12

Value

A vector of colors.

Examples

```
{
# Plot a colorbar with brewer.pal
mypalette<-brewer.pal(7,"Greens")
}
```

`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("BTdata")
```

Format

The format is: chr "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 and and mmer2

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(BTdata)
head(BTdata)
# mix4 <- mmer2(tarsus ~ sex, random = ~ dam + fosternest,
#               data = BTdata)
# summary(mix4)
#### calculate the ratio and its SE
# pin(mix4, dam.prop ~ V1 / ( V1 + V2 + V3 ) )

#   ####============================================####
#   ####============================================####
#   ####============================================####
#   #### EXAMPLE 2
#   #### more complex multivariate model
#   ####============================================####
#   ####============================================####
#   data(BTdata)
#   mix3 <- mmer2(cbind(tarsus, back) ~ sex,
#                 random = ~ dam + fosternest,
#                 data = BTdata, MVM=TRUE)
#   summary(mix3)
#   #### calculate the genetic correlation
#   pin(mix3, gen.cor ~ V2 / sqrt(V1*V3))
```

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 <covarrubiasp@wisc.edu>

See Also

coef, mmer and and mmer2

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

Description

coef method for class "MMERM".

Usage

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

Arguments

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

Value

vector of coef

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

coef, mmer and mmer2

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

Description

`coef` method for class "`mmerM`".

Usage

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

Arguments

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

Value

vector of coef

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

coef, mmer

cornHybrid *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("cornHybrid")
```

Format

The format is: chr "cornHybrid"

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 mmer2

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(cornHybrid)
####============================================####
#### look at the list structure
```

```
####============================================####
str(cornHybrid)
####============================================####
####============================================####
#### breeding values with 3 variance components
#### hybrid prediction
####============================================####
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K
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)

colnames(Z1) <- levels(hybrid2$GCA1)
colnames(Z2) <- levels(hybrid2$GCA2)
colnames(Z3) <- levels(hybrid2$SCA)
####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####============================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####============================================####
S <- kronecker(K1, K2) ; dim(S)
rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

ETA <- list(GCA1=list(Z=Z1, K=K1),
            GCA2=list(Z=Z2, K=K2),
            SCA=list(Z=Z3, K=S)
            )
#ans <- mmer(Y=y, X=X1, Z=ETA)
#ans$var.comp
#summary(ans)

####============================================####
####============================================####
#### using the 'mmer2' function would be fitted as
####============================================####
####============================================####

#data(cornHybrid)
#hybrid2 <- cornHybrid$hybrid # extract cross data
#A <- cornHybrid$K
#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 <- mmer2(Yield ~ Location, random = ~ g(GCA1) + g(GCA2) + g(SCA),
#       G=list(GCA1=K1, GCA2=K2, SCA=S),data=hybrid2)
#summary(ans)

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####======================================####
####======================================####
#### Example 2
#### covariance structure in interaction effects
####======================================####
####======================================####
# data(cornHybrid)
# hybrid2 <- cornHybrid$hybrid # extract cross data
# A <- cornHybrid$K # additive relationship matrix
# y <- hybrid2$Yield # response
# # subset additive relationship matrix
# K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
# # create new variable
# hybrid2$GCA2.LOC <- paste(hybrid2$GCA2, hybrid2$Location, sep="-")
# L <- diag(4);
# # variance covariance for G:E
# GA <- kronecker(L,K2);
# colnames(GA) <- paste(colnames(K2),sort(rep(1:4,20)), sep="-")
# rownames(GA) <- colnames(GA); GA[1:4,1:4];
#
# # fit the model
# ans <- mmer2(Yield ~ Location, random = ~ g(GCA2) + g(GCA2.LOC),
#              G=list(GCA2=K2, GCA2.LOC=GA),data=hybrid2)
# summary(ans)

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####======================================####
####======================================####
#### Example of multivariate model
####======================================####
####======================================####
```

```
# data(cornHybrid)
# hybrid2 <- cornHybrid$hybrid # extract cross data
# hybrid2 <- hybrid2[which(!is.na(hybrid2$Yield)),]
# names(hybrid2)[5:6] <- c("TY","PH")
# head(hybrid2)
#
# A <- cornHybrid$K
# K1 <- A[levels(hybrid2$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 <- mmer2(cbind(TY,PH) ~ Location, random = ~ g(GCA2) + g(SCA),
#       G=list(GCA2=K2, SCA=S),data=hybrid2, MVM=TRUE)
# summary(ans)
```

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("CPdata")
```

Format

The format is: chr "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 mmer2

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(CPdata)
CPpheno <- CPdata$pheno[,-c(1:4)]
CPgeno <- CPdata$geno
#### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
#### fit a model including additive and dominance effects
y <- CPpheno$color
Za <- diag(length(y))
Zd <- diag(length(y))
Ze <- diag(length(y))
A <- A.mat(CPgeno) # additive relationship matrix
#D <- D.mat(CPgeno) # dominant relationship matrix
#E <- E.mat(CPgeno) # epistatic relationship matrix

y.trn <- y # for prediction accuracy
ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
y.trn[ww] <- NA
####================####
#### ADDITIVE MODEL ####
####================####
#ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- mmer(Y=y.trn, Z=ETA.A)
#cor(ans.A$fitted.y[ww], y[ww], use="pairwise.complete.obs")
####==========================####
#### ADDITIVE-DOMINANCE MODEL ####
####==========================####
#ETA.AD <- list(add=list(Z=Za,K=A), dom=list(Z=Zd,K=D))
#ans.AD <- mmer(Y=y.trn, Z=ETA.AD)
#cor(ans.AD$fitted.y[ww], y[ww], use="pairwise.complete.obs")
### greater accuracy !!!! 4 percent increment!!
### we run 100 iterations, 4 percent increment in general
####====================================####
#### ADDITIVE-DOMINANCE-EPISTASIS MODEL ####
####====================================####
#ETA.ADE <- list(add=list(Z=Za,K=A), dom=list(Z=Zd,K=D), epi=list(Z=Ze,K=E))
#ans.ADE <- mmer(Y=y.trn, Z=ETA.ADE)
#cor(ans.ADE$fitted.y[ww], y[ww], use="pairwise.complete.obs")

#summary(ans.A)
#summary(ans.AD)
```

```
#summary(ans.ADE)
#### adding more effects doesn't necessarily increase prediction accuracy!

####=========================================####
#### RUN AS GWAS MODEL
####=========================================####
#ans.GWAS <- mmer(Y=y, Z=ETA.A, W=CPgeno)
#### or if you have a map
#my.map <- CPdata$map
#ans.GWAS <- mmer(Y=y, Z=ETA.A, W=CPgeno, map=my.map)


##########################################
##########################################
##########################################
##########################################
##########################################
##########################################


####=========================================####
####=========================================####
#### EXAMPLE 2
#### Genomic prediction
#### using multiple responses in parallel
####=========================================####
####=========================================####
#data(CPdata)
#CPpheno <- CPdata$pheno[,-c(1:4)]
#CPgeno <- CPdata$geno
### look at the data
#head(CPpheno)
#CPgeno[1:5,1:5]

#### fit a model including additive effects
#y <- CPpheno$color
#Za <- diag(length(y))
#A <- A.mat(CPgeno) # additive relationship matrix

#y.trn <- CPpheno # for prediction accuracy
#ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
#y.trn[ww,] <- NA
#ETA.A <- list(add=list(Z=Za,K=A))
#*******************************************************
#### WINDOWS
#ans.A <- mmer(Y=y.trn, Z=ETA.A, method="EMMA")
#### MAC
#ans.A <- mmer(Y=y.trn, Z=ETA.A, method="EMMA",n.cores=4)
#*******************************************************

### 1) Extract the fitted values for the simulated data
#fitt <- as.matrix(do.call("cbind",lapply(ans.A, function(x,w){x$fitted.y[w]},w=ww)))
### 2) See the correlation with the original data masked
```

```
#res <- apply(data.frame(1:4),1,function(x,y,z){
#  cor(y[,x],z[,x],use="pairwise.complete.obs")},
#             y=fitt,z=CPpheno[ww,])
#res

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####======================================####
####======================================####
#### EXAMPLE 3
#### Genomic prediction
#### Univariate vs Multivariate models
####======================================####
####======================================####
#data(CPdata)
#CPpheno <- CPdata$pheno[,-c(1:4)]
#CPgeno <- CPdata$geno
### look at the data
#head(CPpheno)
#CPgeno[1:5,1:5]
## fit a model including additive and dominance effects
#Za <- diag(dim(CPpheno)[1])
#A <- A.mat(CPgeno) # additive relationship matrix

#CPpheno2 <- CPpheno
#ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
#CPpheno2[ww,"color"] <- NA
####==================####
#### univariate model ####
####==================####
#ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- mmer(Y=CPpheno2$color, Z=ETA.A)
#cor(ans.A$fitted.y[ww], CPpheno[ww,"color"], use="pairwise.complete.obs")
####======================####
#### multivariate model ####
####     4 traits       ####
####======================####
#### be patient take some time
#ETA.B <- list(add=list(Z=Za,K=A))
#ans.B <- mmer(Y=CPpheno2, Z=ETA.B, MVM=TRUE)
#cor(ans.B$fitted.y[ww,"color"], CPpheno[ww,"color"], use="pairwise.complete.obs")


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

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

####========================================####
####========================================####
#### EXAMPLE 4
#### Cross validation example
####========================================####
####========================================####
## get data
# data(CPdata)
# CPpheno <- CPdata$pheno[,-c(1:4)]
# CPgeno <- CPdata$geno
# head(CPpheno)
# CPgeno[1:5,1:5]
#
# y <- CPpheno$color
# Za <- diag(length(y))
# A <- A.mat(CPgeno)
#
# ## for 5-fold cross validation
# rownames(CPpheno) <- rownames(CPgeno)
#
# cv.cors <- list()
# for(j in 1:50){ # 50 rounds of 5-fold CV
#
#   base <- rownames(CPgeno) # base names of the genotypes
#   ## define the cross validation groups
#   CV.scheme <- list()
#   for(i in 1:5){
#     if(i ==5){totake <- length(base)}else{totake<-round(dim(Za)[1]/5)}
#     CV.scheme[[i]] <- sample(base,totake) # delete data for 1/5 of the population
#     base <- setdiff(base,CV.scheme[[i]])
#   }
#   ## apply the model to all groups
#   coros <- lapply(CV.scheme,function(x){
#     y.trn<- CPpheno
#     y.trn[x,"Yield"] <- NA
#     ETA.A <- list(add=list(Z=Za,K=A))
#     ans.A <- mmer(Y=y.trn[,"Yield"], Z=ETA.A, method="EMMA", silent = TRUE)
#     blup <- ans.A$u.hat$add;rownames(blup) <- rownames(CPpheno)
#     cor(blup[x,1], CPpheno[x,"Yield"], use="pairwise.complete.obs")
#   })
#   print(j)
#   cv.cors[[j]] <- unlist(coros)
# }
#
# boxplot(unlist(cv.cors),ylim=c(0,1))
#
# ans.A <- mmer(Y=CPpheno[,"Yield"], Z=ETA.A, method="EMMA", silent = TRUE)
# (h2<-ans.A$var.comp[1,1]/sum(ans.A$var.comp))
# sqrt(h2)
```

D.mat *Dominance relationship matrix*

Description

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,min.MAF=NULL,max.missing=NULL,impute.method="mean",tol=0.02,
      n.core=1,shrink=FALSE,return.imputed=FALSE, ploidy=2, return.Xd=FALSE,
      method=1)
```

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 D matrix is not sensitive to rare alleles, so by default only monomorphic markers are removed.
max.missing	Maximum proportion of missing data; default removes completely missing markers.
impute.method	There are two options. The default is "mean", which imputes with the mean for each marker. The "EM" option imputes with an EM algorithm (see details).
tol	Specifies the convergence criterion for the EM algorithm (see details).
n.core	Specifies the number of cores to use for parallel execution of the EM algorithm (use only at UNIX command line).
shrink	Set shrink=TRUE to use the shrinkage estimation procedure (see Details).
return.imputed	When TRUE, the imputed marker matrix is returned.
ploidy	The ploidy of the organism. The default is 2 which means diploid but higher ploidy levels are supported.
return.Xd	If TRUE, the function returns the marker matrix converted to a dominant matrix instead of the dominance relationship matrix. The -1 and 1 are converted to 0, and 0's are converted to 1's which is a common incidence matrix for dominance. By default the markers that did not have heterozygote calls are not returned which would mean returning columns of only zeros putting fresh users in a mathematical problem when trying to estimate the variance component. The default for this argument is FALSE, so the function returns a dominant relationship matrix instead of a marker matrix.
method	Method 1 is Endelman and Jannink (2012) and method 2 is Su et al. (2012).

Details

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

At high marker density, the relationship matrix is estimated as $D = WW'/c$, where $W_{ik} = 1 - \|X_{ik}\|$. By using a normalization constant of $c = 2\sum_k p_k q_k (1 - p_k q_k)$.

The EM imputation algorithm is based on the multivariate normal distribution and was designed for use with GBS (genotyping-by-sequencing) markers, which tend to be high density but with lots of missing data. Details are given in Poland et al. (2012). The EM algorithm stops at iteration t when the RMS error = $n^{-1}\|A_t - A_{t-1}\|_2$ < tol.

At low marker density (m < n), shrinkage estimation can improve the estimate of the relationship matrix and the accuracy of GEBVs for lines with low accuracy phenotypes (Endelman and Jannink 2012). The shrinkage intensity ranges from 0 (no shrinkage, same estimator as high density formula) to 1 (completely shrunk to $(1 + f)I$). The shrinkage intensity is chosen to minimize the expected mean-squared error and printed to the screen as output.

The shrinkage and EM options are designed for opposite scenarios (low vs. high density) and cannot be used simultaneously.

When the EM algorithm is used, the imputed alleles can lie outside the interval [-1,1]. Polymorphic markers that do not meet the min.MAF and max.missing criteria are not imputed.

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

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

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

Poland, J., J. Endelman et al. 2012. Genomic selection in wheat breeding using genotyping-by-sequencing. Plant Genome 5:103-113. doi: 10.3835/plantgenome2012.06.0006

See Also

The core functions of the package `mmer` and `mmer2`

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)

#########################################################
#########################################################
#########################################################
#########################################################
#########################################################
#########################################################

####=========================================####
#### EXAMPLE 2
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####=========================================####
data(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno
#### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
#### fit a model including additive and dominance effects
#y <- CPpheno$color
#Za <- diag(length(y))
#Zd <- diag(length(y))
#Ze <- diag(length(y))
#A <- A.mat(CPgeno) # additive relationship matrix
#D <- D.mat(CPgeno) # dominant relationship matrix

#y.trn <- y # for prediction accuracy
#ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
#y.trn[ww] <- NA
####================####
#### ADDITIVE MODEL ####
####================####
#ETA.A <- list(list(Z=Za,K=A))
#ans.A <- mmer(Y=y.trn, Z=ETA.A)
#cor(ans.A$fitted.y[ww], y[ww], use="pairwise.complete.obs")
####==========================####
#### ADDITIVE-DOMINANT MODEL ####
####==========================####
#ETA.AD <- list(list(Z=Za,K=A),list(Z=Zd,K=D))
```

```
#ans.AD <- mmer(Y=y.trn, Z=ETA.AD)
#cor(ans.AD$fitted.y[ww], y[ww], use="pairwise.complete.obs")
### greater accuracy !!!! ~4 percent increment!!
### we run 100 iterations, ~4 percent increment in general
```

`design.score` *design score for the model to be tested*

Description

This function is a wrapper from the GWASpoly package that creates the incidence matrix to be used when calculating the -log10 scores for the markers once the variance components and marker effects are estimated.

Usage

```
design.score(Mi,model,ploidy,min.MAF,max.geno.freq)
```

Arguments

`Mi`	marker to be used
`model`	model to be used for designing the matrix. The default is "additive" which can be extend to any ploidy level but alternative models for polyploids such as "additive","1-dom-alt","1-dom-ref","2-dom-alt","2-dom-ref" are supported based on Rosyara (2016)
`ploidy`	A numeric value indicating the ploidy level of the organism. The default is 2 which means diploid but higher ploidy levels are supported.
`min.MAF`	minimum minor allele frequency
`max.geno.freq`	1 - min.MAF

Value

$S an incidence matrix for the marker based on the model provided.

See Also

The core function of the package `mmer`

Examples

```
# it works internally in the \code{\link{mmer}} function
```

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).

Usage

```
E.mat(X,min.MAF=NULL,max.missing=NULL,impute.method="mean",tol=0.02,
      n.core=1,shrink=FALSE,return.imputed=FALSE, type="A#A", ploidy=2)
```

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.
max.missing	Maximum proportion of missing data; default removes completely missing markers.
impute.method	There are two options. The default is "mean", which imputes with the mean for each marker. The "EM" option imputes with an EM algorithm (see details).
tol	Specifies the convergence criterion for the EM algorithm (see details).
n.core	Specifies the number of cores to use for parallel execution of the EM algorithm (use only at UNIX command line).
shrink	Set shrink=TRUE to use the shrinkage estimation procedure (see Details).
return.imputed	When TRUE, the imputed marker matrix is returned.
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").
ploidy	The ploidy of the organism. The default is 2 which means diploid but higher ploidy levels are supported.

Details

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

Value

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

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

$E the E 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

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

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

Poland, J., J. Endelman et al. 2012. Genomic selection in wheat breeding using genotyping-by-sequencing. Plant Genome 5:103-113. doi: 10.3835/plantgenome2012.06.0006

See Also

The core functions of the package mmer and mmer2

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"
```

---

eigenGWAS *Unraveling selection signatures with eigenGWAS*

Description

This function performs eigenGWAS based on the paper proposed by Chen et al. (2016) to avoid expensive computations with high number of markers. For mathematical details of the derivation please refer to the original publication below in the references.

eigenGWAS is designed to find ancestry informative markers (AIM) and loci under selection. The phenotype of eigenGWAS is an eigenvector generated from marker data.

Usage

```
eigenGWAS(markers, eivec=1, map=NULL)
```

Arguments

markers	a matrix of numeric markers
eivec	eigen vector to be used as a response
map	optional argument to provide a map to sort the signature. Obligatory column names of the map are "Chrom" and "Position".

Value

$result list with signature and Fst metric.

Author(s)

Giovanny Covarrubias-Pazaran

References

Chen, G.B., S.H. Lee, ZX Zhu, B Benyamin, MM Robinson, EigenGWAS: finding loci under selection through genome-wide association studies of eigenvectors in structured populations. Heredity 2016.

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 mmer2

Examples

```
data(CPdata)
markers <- CPdata$geno
markers[1:5,1:5]
#res <- eigenGWAS(markers)
### ==================== ###
### use map information
### ==================== ###
#my.map <- CPdata$map
#head(my.map)
#res <- eigenGWAS(markers, map = my.map)
```

EM *Expectation Maximization Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use 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 and mmer2

Examples

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

##################################################
##################################################
# IMPORT DATA FOR  ESTIMATING 3 VARIANCE COMPONENTS
##################################################
##################################################
## Import phenotypic data on inbred performance
## Full data
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # 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=3)
```

EM2 *Expectation Maximization Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the expectation maximization (EM) algorithm.

Usage

```
EM2(y, X=NULL, ETA=NULL, R=NULL, init=NULL,
   iters=50, REML=TRUE, draw=TRUE, silent=FALSE)
```

Arguments

y	a numeric vector for the response variable
X	an incidence matrix for fixed effects.
ETA	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 matrix for variance-covariance structures for the residuals, i.e. for longitudinal data. if not passed is assumed an identity matrix.
init	Initial values for the variance components if defaults want to be changed.
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.
REML	a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.
draw	a TRUE/FALSE value indicating if a plot of updated values for the variance components 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.

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 function 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 DATA FOR  ESTIMATING 3 VARIANCE COMPONENTS
```

```
################################################
################################################
## Import phenotypic data on inbred performance
## Full data
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # 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, ETA=ETA, iters=3)
```

EMMA *Efficient Mixed Model Association Algorithm*

Description

This function is used internally in the function mmer when one or more variance component other than the error needs to be estimated through the use of the efficient mixed model association (EMMA) algorithm. When multiple random effects are fitted the function internally create weight for each random effect first and then does all calculations.

Usage

```
EMMA(y, X=NULL, ZETA=NULL, REML=TRUE, silent=FALSE, che=TRUE, forced=NULL, EIGEND=FALSE)
```

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).

`REML` a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`silent` a TRUE/FALSE value indicating if the function should draw the progress bar or iterations performed while working or should not be displayed.

`che` 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.

`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.

`EIGEND` a TRUE/FALSE value indicating if the function should perform the eigen decomposition of the K matrix for square problems to accelerate inversion and estimate parameters faster. Is 2 to 3 times faster than regular EMMA but only works for square problems.

Details

This algorithm is based on Kang et al. (2008), it is based on REML using the ridge regression parameter lambda as the ration of Var(e)/Var(u). 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 for the random effect "u" as Z and its variance-covariance matrix as K.

.

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 and mmer2

Examples

```
####============================================####
#### breeding values with 1 variance component
#### using EMMA algorithm
####============================================####

####============================================####
#### random population of 200 lines with 1000 markers
####============================================####
M <- matrix(rep(0,200*1000),200,1000)
for (i in 1:200) {
  M[i,] <- ifelse(runif(1000)<0.5,-1,1)
}
####============================================####
#### Simulate random phenotypes
####============================================####
u <- rnorm(1000)
g <- as.vector(crossprod(t(M),u))
h2 <- 0.5  #heritability
y <- g + rnorm(200,mean=0,sd=sqrt((1-h2)/h2*var(g)))
A <- A.mat(M)
ETA <- list(add=list(K=A))
ans <- EMMA(y=y, ZETA=ETA)
```

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(ExpDesigns)
data1 <- ExpDesigns$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 <- mmer2(yield~entryc, random=~block+trt, data=data1)
summary(mix1)

# ## compare raw unreplicated measure with adjusted blup
# library(plyr)
# avers <- ddply(data1, .(trt), summarize,mean = round(mean(yield), 2))
# plot(avers$mean[-c(1:4)], mix1$u.hat$trt[-c(1:4),1],ylab="BLUP",xlab="Raw",
#      pch=20, cex=2, col="blue")
# ## if you have row and column information you can see the design
# library(agridat)
# data1$row <- c(sample(1:7),sample(1:6),sample(1:7))
# data1$col <- c(1:3)[data1$block]
# d1 <- desplot(yield ~ row*col, data1, main="ABD1",
#               text=trt,strip.cex=3, cex=1)
# print(d1)

#### ================================== ####
#### ===== Augmented Block Design 2 ==== ####
#### ================================== ####
data(ExpDesigns)
data2 <- ExpDesigns$au2
head(data2)
## response variable: "TSW"
## check indicator: "entryc"
## blocking factor: "Block"
## treatments, replicated and non-replicated: "Entry"
## check no check indicator: "new"
## this is also known as Federer's unreplicated design
mix2<- mmer2(TSW ~ entryc, random=~Block+Entry, data=data2)
```

```
summary(mix2)

#### ================================== ####
#### ===== Incomplete block design  ==== ####
#### ================================== ####
data(ExpDesigns)
data.ibd <- ExpDesigns$ibd$book
head(data.ibd)
ExpDesigns$ibd$sketch
## response variable: "yield"
## 2 replications (r)
## 30 genotypes (trt)
## 10 incomplete blocks (s) with 3 trts each (k)
## design was an alpha design
## agricolae::design.alpha(trt=paste("gen",1:30,sep=""),k=3,r=2,seed=5)$sketch
mix.ibd <- mmer2(yield~Genotype,random=~replication+replication:block,
                 data=data.ibd)
summary(mix.ibd)
# rownames(a)[1] <-"geno1"
#a[-1] <- a[-1]+a[1]
#plot(density(mix.ibd$beta.hat))

## map of the field
# library(agridat)
# data.ibd$block <- as.numeric(as.character(data.ibd$block))
# data.ibd$cols <- as.numeric(as.character(data.ibd$cols))
# d1 <- desplot(yield ~ block*cols, data.ibd, main="IBD",
#               text=Genotype,strip.cex=3, cex=1)
# print(d1)

#### ================================== ####
#### ======= Split Plot Design  ======== ####
#### ================================== ####
data(ExpDesigns)
data.spd <- ExpDesigns$spd
head(data.spd)
## response variable: "yield"
## 3 blocks or reps (r)
## 2 whole plot treatment (A)
## 3 small plot treatments (B)
##
##        i.e BLOCK 1
##[]====================[]
##[] A1(B1) A1(B2) A1(B3) []
##[] A2(B1) A2(B2) A2(B3) []
##[]====================[]
##
## more replication in whole plot treatments (A)
## less replication in sub plot treatments (B)
mix.split <- mmer2(yield ~block + A + B ,random=~ A:B, data=data.spd)
summary(mix.split)

#### ================================== ####
```

```
#### ==== Split-Split Plot Design  ===== ####
#### ================================== ####
data(ExpDesigns)
data.sspd <- ExpDesigns$sspd
head(data.sspd)
## response variable: "yield"
## 5 levels of nitrogen (N) main plot
## 3 levels of management (M) sub-plot
## 3 varieties (B) sub-sub-plot
##
##        i.e BLOCK 1
##[]==================================[]
##[] N1(M1(V1)) N1(M2(V1)) N1(M3(V1)) []
##[] N2(M1(V1)) N2(M2(V1)) N2(M3(V1)) []
##[] N3(M1(V1)) N3(M2(V1)) N3(M3(V1)) []
##[] N4(M1(V1)) N4(M2(V1)) N4(M3(V1)) []
##[] N5(M1(V1)) N5(M2(V1)) N5(M3(V1)) []
##[]==================================[]
##
head(data.sspd)
mix.sspd <- mmer2(yield ~1,random=~ block + nitrogen + management +
                    variety + nitrogen:management + variety:nitrogen +
                    variety:management + variety:nitrogen:management,
                  data=data.sspd)
summary(mix.sspd)

#### ================================== ####
#### ======= Latin Square Design  ====== ####
#### ================================== ####
data(ExpDesigns)
data.lsd <- ExpDesigns$lsd
head(data.lsd)
## response variable: "yield"
## 4 columns (c)
## 4 rows (r)
## 4 varieties (V)
##
##   c1 c2 c3 c4
##[]=============[]
##[] V1 V4 V2 V3 [] row 1
##[] V2 V3 V4 V1 [] row 2
##[] V3 V2 V4 V1 [] row 3
##[] V4 V1 V3 V2 [] row 4
##[]=============[]
##   c1 c2 c3 c4
##
mix.lsd <- mmer2(yield ~ variety ,random=~ row + col, data=data.lsd)
summary(mix.lsd)

# library(agridat)
# desplot(yield ~ row*col, data.lsd, main="LSD",
#               strip.cex=3, cex=1, text=variety)
```

```
#### ================================== ####
#### =====  North Carolina Design I ==== ####
#### ================================== ####
data(ExpDesigns)
data.car1 <- ExpDesigns$car1
head(data.car1)
## response variable: "yield"
## male indicator: "male"
## female indicator: "female"
## replication: "rep"
## set of males: "set"
mix.car1 <- mmer2(yield~set,random=~ set:rep + set:male
                  +set:male:female + set:male:female:rep, data=data.car1)
summary(mix.car1)
(Var.A <- 4*mix.car1$var.comp[2,1])
(Var.D <- 4*mix.car1$var.comp[3,1] - 4*mix.car1$var.comp[2,1])

#### ================================== ####
#### ===== North Carolina Design II ==== ####
#### ================================== ####
data(ExpDesigns)
data.car2 <- ExpDesigns$car2
head(data.car2)
## response variable: "yield"
## male indicator: "male"
## female indicator: "female"
## replication: "rep"
## set of males: "set"
mix.car2 <- mmer2(yield ~ 1, random=~ set + set:rep + set:male
                  + set:female + set:male:female, data=data.car2)
summary(mix.car2)
(Var.Am <- 4*mix.car2$var.comp[3,1])
(Var.Af <- 4*mix.car2$var.comp[4,1])
(Var.D <- 4*mix.car2$var.comp[5,1])

#### ================================== ####
#### ==== North Carolina Design III ==== ####
#### ================================== ####
data(ExpDesigns)
data.car3 <- ExpDesigns$car3
head(data.car3)
## response variable: "yield"
## male indicator: "male"
## female indicator: "female"
## replication: "rep"
## set of males: "set"
mix.car3 <- mmer2(yield ~ set + set:rep, random=~ set:male
                  + set:female + set:male:female, data=data.car3)
summary(mix.car3)
(Var.A <- 4*mix.car3$var.comp[1,1]) # var males
(Var.D <- 2*mix.car3$var.comp[3,1]) # var females in males
```

F1geno *Genotypes from an F1(CP) cross to show phasing*

Description

This dataset contains genotypic data from a cranberry biparental population published in Covarrubias-Pazaran et al. (2016) where a high density map was built using GBS data. This data is a companion to show the use of the phase.F1 function which is used to build parental maps imputing the Aa alleles from AaxAa markers which are usually lost during the phasing process by lack of linkage information. This map allow the user to add the hk markers to the parental maps without loosing information for the Aa alleles as JoinMap v4.1 usually does.

Usage

```
data("F1geno")
```

Format

The format is: chr "F1geno"

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 and mmer2

Examples

```
####==========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####==========================================####
data(F1geno)
F1geno[1:10,1:10]
#maps <- phase.F1(F1geno)
#maps$maternal[1:10,1:10]
#heat <- apply(maps$maternal[,-c(1:6)],2,function(x){as.numeric(as.factor(x))})
#heat[1:10,1:10]
#image(heat)
####==========================================####
#### use variance to remove bad markers
```

```
####=========================================####
#plot(apply(heat,1,var))
#jjj <- which(apply(heat,1,var) > .5)
#image(heat[-jjj,])
####=========================================####
#### do bin mapping using ASMap
####=========================================####
```

FDdata *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("FDdata")
```

Format

The format is: chr "FDdata"

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 mmer2

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####=========================================####
data(FDdata)
head(FDdata)
mix <- mmer2(stems~1, random=~female+male, data=FDdata)
summary(mix)
```

```
####============================================####
####============================================####
#### using mmer function would be like
####============================================####
####============================================####
Z1 <- model.matrix(~female-1, data=FDdata)
Z2 <- model.matrix(~male-1, data=FDdata)
ETA <- list(GCA1=list(Z=Z1), GCA2=list(Z=Z2))
y <- FDdata$stems
mix2 <- mmer(Y=y, Z=ETA, method = "NR")
summary(mix2)


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

# data(FDdata)
# head(FDdata)
#
# mix <- mmer2(cbind(stems,pods,seeds)~1,
#              random=~female+male, MVM=TRUE,data=FDdata)
# summary(mix)
# #### genetic variance covariance
# gvc <- mix$var.comp$female
# #### extract variances (diagonals) and get standard deviations
# sd.gvc <- as.matrix(sqrt(diag(gvc)))
# #### get possible products sd(Vgi) * sd(Vgi')
# prod.sd <- sd.gvc %*% t(sd.gvc)
# #### genetic correlations cov(gi,gi')/[sd(Vgi) * sd(Vgi')]
# (gen.cor <- gvc/prod.sd)
# #### pods and seeds have a strong negative genetic covariance (-.79)
# #### more pods, less seeds
```

`fdr` *False Discovery Rate calculation*

Description

This function calculates the false discovery rate (FDR) at a certain value specified by the user. Is a wrapper of the p.adjust function from the stats package. It has been adjusted to facilitate the user to find the FDR value for a vector of values in scale minus log10. It is used internally of the `mmer` function when the argument W is used to perform GWAS.

Usage

```
fdr(p, fdr.level = 0.05)
```

Arguments

p	a vector of minus log(p values)
fdr.level	the level of false discovery rate desired

Value

$p.ad a vector of new minus log10(p values) adjusted for false discovery rate at a given level specified by the user.

$fdr.value a scalar value indicating where the FDR line should be drawn for your new adjusted minus log10(p values)

$p.or a vector with the initial minus log10(p values) provided by the user.

$fdr.or a scalar value indicating where the FDR line should be drawn for your original minus log10(p values), if the user prefers to plot the original values and wants to draw the FDR line in the original scale.

Author(s)

Giovanny Covarrubias-Pazaran

References

Benjamini, Y., and Yekutieli, D. 2001. The control of the false discovery rate in multiple testing under dependency. Annals of Statistics 29, 1165-1188.

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 mmer2

Examples

```
####============================================####
#### generate your mickey mouse -log10(p-values)
####============================================####
set.seed(1253)
pp <- abs(rnorm(500,0,3));pp[23:34] <- abs(rnorm(12,0,20))

####============================================####
#### see how they look like
####============================================####
plot(pp, col=transp("cadetblue"), pch=20, cex=1.5)

####============================================####
#### adjust the values for FDR and see how they look like
####============================================####
new.pp <- fdr(pp)
plot(new.pp$p.ad, col=transp("cadetblue"), pch=20, cex=1.5)
```

```
####==========================================####
#### or you may want to plot your original values
#### with the FDR line instead
####==========================================####
plot(pp, col=transp("cadetblue"), pch=20, cex=1.5)
abline(h=new.pp$fdr.10, lty=3, col="red", lwd=2)
```

`fdr2` *False Discovery Rate calculation*

Description

This function calculates the false discovery rate (FDR) at a certain value specified by the user. Is a wrapper of the p.adjust function from the stats package. It has been adjusted to facilitate the user to find the FDR value for a vector of values in scale minus log10. It is used internally of the `mmer` function when the argument W is used to perform GWAS.

Usage

```
fdr2(p, fdr.level = 0.05)
```

Arguments

`p`	a vector of minus log(p values)
`fdr.level`	the level of false discovery rate desired

Value

$p.ad a vector of new minus log10(p values) adjusted for false discovery rate at a given level specified by the user.

$fdr.value a scalar value indicating where the FDR line should be drawn for your new adjusted minus log10(p values)

$p.or a vector with the initial minus log10(p values) provided by the user.

$fdr.or a scalar value indicating where the FDR line should be drawn for your original minus log10(p values), if the user prefers to plot the original values and wants to draw the FDR line in the original scale.

Author(s)

Giovanny Covarrubias-Pazaran

References

Benjamini, Y., and Yekutieli, D. 2001. The control of the false discovery rate in multiple testing under dependency. Annals of Statistics 29, 1165-1188.

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 mmer2

Examples

```
####============================================####
#### generate your mickey mouse -log10(p-values)
####============================================####
set.seed(1253)
pp <- abs(rnorm(500,0,3));pp[23:34] <- abs(rnorm(12,0,20))

####============================================####
#### see how they look like
####============================================####
plot(pp, col=transp("cadetblue"), pch=20, cex=1.5)

####============================================####
#### adjust the values for FDR and see how they look like
####============================================####
new.pp <- fdr2(pp)
plot(new.pp$p.ad, col=transp("cadetblue"), pch=20, cex=1.5)

####============================================####
#### or you may want to plot your original values
#### with the FDR line instead
####============================================####
plot(pp, col=transp("cadetblue"), pch=20, cex=1.5)
abline(h=new.pp$fdr.10, lty=3, col="red", lwd=2)
```

`fill.design` *Filling the design of an experiment*

Description

The fill.design function allows the user to fill the missing rows and ranges from an experiment to run specific designs or apply a post blocking. The data frame requires the presence of 2 numeric columns indicating the x and y coordinates, here denominated rows and ranges. This can be effectively used for multi environment trials by using the 'by' argument where you specify the column that indicates the environments.

Usage

```
fill.design(x,rows="ROW",ranges="RANGE",by, extra)
```

Arguments

x	a dataframe with 2 obligatory columns; rows and ranges which can have different names and can be matched with the next arguments.
rows	the name of the numeric column that indicates one direction in the field.

ranges	the name of the numeric column that indicates the other direction in the field.
by	optional argument to indicate the name of the column of the dataframe x that indicates the environments so the field is filled by environment.
extra	name of the extra columns to be filled in the dataset based. This are filled based on the rows, ranges information.

Value

$xnew a new dataframe identical to the one provided but with missing rows and ranges filled in.

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 mmer2

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(CPdata)
CPpheno <- CPdata$pheno#[,-c(1:4)]
CPgeno <- CPdata$geno
#### look at the data
head(CPpheno)
#### fill the design
gg <- fill.design(x=CPpheno, rows="Row",ranges="Col")
head(gg)
```

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

Description

fitted method for class "mmer".

Usage

```
## S3 method for class 'mmer'
fitted(object, type = "complete", ...)
```

Arguments

object	an object of class "mmer"
type	the type of fitted which should be returned. The alternatives are: "complete" (y.hat=Xb+Zu) and "incomplete" (y.hat=Zu).
...	Further arguments to be passed

Value

vector of fitted

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

fitted, mmer and mmer2

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

Description

fitted method for class "MMERM".

Usage

```
## S3 method for class 'MMERM'
fitted(object, type = "complete", ...)
```

Arguments

object	an object of class "MMERM"
type	the type of fitted which should be returned. The alternatives are: "complete" (y.hat=Xb+Zu) and "incomplete" (y.hat=Zu).
...	Further arguments to be passed

Value

vector of fitted

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

fitted, mmer

---

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

Description

fitted method for class "mmerM".

Usage

```
## S3 method for class 'mmerM'
fitted(object, type = "complete", ...)
```

Arguments

object	an object of class "mmerM"
type	the type of fitted which should be returned. The alternatives are: "complete" (y.hat=Xb+Zu) and "incomplete" (y.hat=Zu).
...	Further arguments to be passed

Value

vector of fitted

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

fitted, mmer, and mmer2

g *g functionality*

Description

This function allows mmer2 to recognize the use of a variance-covariance structure for a random effect.

Usage

```
g(x)
```

Arguments

x	random effect with variance-covariance structure to add

Value

$x returns the random effect and mmer2 can include the G 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 mmer2

Examples

```
#### call the data
data(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno
#### create the variance-covariance matrix
A <- A.mat(CPgeno)
#### look at the data and fit the model
head(CPpheno)
mix1 <- mmer2(Yield~1,random=~g(id), G=list(id=A), data=CPpheno)
summary(mix1)
```

gryphondata *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("gryphondata")
```

Format

The format is: chr "gryphondata"

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 mmer2

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(gryphondata)

gryphon <- gryphondata$gryphon
ppedig <- gryphondata$pedi
## obtained using the 'getA()' function from the pedigreemm package
A <- gryphondata$A

#### look at the data
head(gryphon)

#### fit the model with no fixed effects (intercept only)
#mix1 <- mmer2(BWT~1,random=~g(ANIMAL), G=list(ANIMAL=A), data=gryphon)
#summary(mix1)

#### fit the model with SEX as fixed effects
#mix2 <- mmer2(BWT~1+SEX,random=~g(ANIMAL), G=list(ANIMAL=A), data=gryphon)
#summary(mix2)
```

---

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 h2 for the trait.

Usage

```
data("h2")
```

Format

The format is: chr "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 `mmer2`

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(h2)
head(h2)
####============================================####
#### fit the mixed model and extract var.comp
####============================================####
#ans1 <- mmer2(y~1, random=~Name + Env + Name:Env + Block,data=h2, method="NR")
#vc <- ans1$var.comp
#V_E <- vc[2,1];V_GE <- vc[3,1];V_G <- vc[1,1];Ve <- vc[5,1]
####============================================####
#### calculate heritability
####============================================####
#n.env <- length(levels(h2$Env))
#h2c <- V_G/(V_G + V_GE/n.env + Ve/(2*n.env)) #the 2 is a reference for block#
#h2c

####============================================####
####============================================####
#### using the 'mmer' function would be fitted as
####============================================####
####============================================####
#data(h2)
#Y <- h2$y
#Z1 <- model.matrix(~Name-1,data=h2)
#Z2 <- model.matrix(~Env-1,data=h2)
#Z3 <- model.matrix(~Name:Env-1,data=h2)
#Z4 <- model.matrix(~Block-1,data=h2)
#ETA <- list(Name=list(Z=Z1),
#            Env=list(Z=Z2),
#            Name.Env=list(Z=Z3),
#            Block=list(Z=Z4))

#ans1 <- mmer(Y=Y,Z=ETA)
#vc <- ans1$var.comp
#V_E <- vc[2,1];V_GE <- vc[3,1];V_G <- vc[1,1];Ve <- vc[5,1]
#n.env <- length(levels(h2$Env))
#h2c <- V_G/(V_G + V_GE/n.env + Ve/(2*n.env)) #the 2 is a reference for block#
#h2c
```

`hadamard.prod` *Hadamard product of two matrices*

Description

This function returns the Hadamard or Shur product of two matrices, x and y, that have the same row and column dimensions.

Usage

```
hadamard.prod(x, y)
```

Arguments

x	a numeric matrix or vector object
y	a numeric matrix or vector object

Details

The Hadamard product is an element-by-element product of the two matrices. Let $\mathbf{x}$ and $\mathbf{x}$ be two $m \times n$ numeric matrices. The Hadamard product is $\mathbf{x} \circ \mathbf{y} = \begin{bmatrix} x_{1,1}\, y_{1,1} & x_{1,2}\, y_{1,2} & \cdots & x_{1,n}\, y_{1,n} \\ x_{2,1}\, y_{121} & x_{2,2}\, y_{2,2} & \cdots & x_{2,n}\, y_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m,1}\, y_{m,1} & x_{m,2}\, y_{m,2} & \cdots & x_{m,n}\, y_{m,n} \end{bmatrix}$.

It uses the * operation in R.

Value

A matrix.

Note

The function converts vectors to matrices if necessary. The function stops running if x or y is not numeric and an error message is displayed. The function also stops running if x and y do not have the same row and column dimensions and an error mesage is displayed.

References

Hadamard, J (1983). Resolution d'une question relative aux determinants, *Bulletin des Sciences Mathematiques*, 17, 240-246.

Styan, G. P. H. (1973). Hadamard Products and Multivariate Statistical Analysis, *Linear Algebra and Its Applications*, Elsevier, 6, 217-240.

Examples

```
x <- matrix( c( 1, 2, 3, 4 ), nrow=2, byrow=TRUE )
y <- matrix( c( 2, 4, 6, 8 ), nrow=2, byrow=TRUE )
z <- hadamard.prod( x, y )
print( z )
```

HDdata *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. Since there's 2 technical reps the error variance obtained by REML/ML has to be divided over 2.

Usage

```
data("HDdata")
```

Format

The format is: chr "HDdata"

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 mmer2

Examples

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

####============================================####
#### GCA matrix for half diallel using male and female columns
#### use the 'hdm' function to create the half diallel matrix
####============================================####
Z1 <- overlay(HDdata[,c("male","female")])

####============================================####
#### SCA matrix
```

```
####============================================####
Z2 <- model.matrix(~as.factor(geno)-1, data=HDdata)

####============================================####
#### Define response variable and run
####============================================####
y <- HDdata$sugar
ETA <- list(GCA=list(Z=Z1), SCA=list(Z=Z2)) # Zu component
modHD <- mmer(Y=y, Z=ETA)
summary(modHD)

####============================================####
  #### Example 2
  #### using overlay with mmer2 function
  ####============================================####
  # data(HDdata)
  # head(HDdata)
  # HDdata$female <- as.factor(HDdata$female)
  # HDdata$male <- as.factor(HDdata$male)
  # HDdata$geno <- as.factor(HDdata$geno)
  # #### model using overlay
  # modh <- mmer2(sugar~1, random=~female + and(male) + geno,
  #               data=HDdata)
  # summary(modh)
  # #### model using overlay [and(.)] and covariance structures [g(.)]
  # A <- diag(7); A[1,2] <- 0.5; A[2,1] <- 0.5 # fake covariance structure
  # colnames(A) <- as.character(1:7); rownames(A) <- colnames(A);A
  #
  # modh2 <- mmer2(sugar~1, random=~g(female) + and(g(male)) + geno,
  #               G=list(female=A, male=A),data=HDdata)
  # summary(modh2)
```

---

hdm *Half Diallel Matrix*

Description

THIS FUNCTION HAS BEEN OVERPASSED BY THE 'overlay' FUNCTION WHICH IS MORE FLEXIBLE AND FASTER.

This function uses a dataframe with 2 columns named "female" and "male" in numeric format and creates an incidence matrix for a single explanatory variable corresponding to the GCA effect. The resulting incidence matrix can be used in the mmer function as a 'Z' argument in the 2-level list argument for random effects.

Usage

```
hdm(dat)
```

Arguments

dat	a dataframe with 2 columns named 'female' and 'male' with numeric or factor values indicating the male or female used to produce such hybrid.

Value

$Z an incidence matrix with as many columns as parents in the dataframe indicating with ones the parents used for a particular hybrid (in rows).

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 mmer2

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(HDdata)
head(HDdata)
####============================================####
#### GCA matrix for half diallel using male and female columns
#### use the 'hdm' function to create the half diallel matrix
####============================================####
Z1 <- hdm(HDdata[,c(3:4)])
####============================================####
#### Obtain the SCA matrix
####============================================####
Z2 <- model.matrix(~as.factor(geno)-1, data=HDdata)
####============================================####
#### Define the response variable and run
####============================================####
y <- HDdata$sugar
ETA <- list(list(Z=Z1), list(Z=Z2)) # Zu component
modHD <- mmer(Y=y, Z=ETA, draw=FALSE, silent=TRUE)
summary(modHD)
```

`hits` *Creating a fixed effect matrix with significant GWAS markers*

Description

This function was designed to create a design matrix with the significant markers found by a GWAS analysis in order to use it in the GBLUP or genomic prediction analysis to increase the predction accuracy. The method is based on sevral papers suggesting that the use of signifant markers associated to the causal variant may increase the prediction accuracy under the GBLUP framework. For example, Bernardo (2014), Gianola et al. (2014) Abdollahi Arpanahi et al. (2015) papers were using the top 10-20 GWAS hit markers and using them as fixed effect, increasing the prediction accuracy of the genomic prediction model. This phenomena has been explained arguing that the mixed model shrinks too much the effect of markers with big effects and therefore using such markers as fixed effects causes a dramatic increase in the prediction accuracy of a model using them. It has to be noted that for traits of low h2 or with a high number of QTL's with small effects this method doesn't help but actually has a reducing effect in the prediction accuracy.

Usage

```
hits(gwasm, nmar=10, threshold=1, pick=FALSE, method="cluster", only.mark=FALSE,
    plotting=TRUE)
```

Arguments

`gwasm`	a GWAS model fitted using mmer
`nmar`	the number of GWAS hits (markers) to be used for designing the incidence matrix. It finds the markers with maximum significance value and uses them to create the design matrix. The default is the top 10 markers.
`threshold`	a numeric value indicating the minimum significance value to be used for finding the significant markers. the dedault is 1.
`pick`	a TRUE/FALSE value indicating if the user prefers to pick the peaks by himself. The default is FALSE leaving the peak selection to one of the two methods available. If set to TRUE R will allow the user to pick the peaks by cliking over the peaks and typing 'Esc' when the user is done selecting the peaks.
`method`	one of the two methods available; "cluster" performs peak selection by making clusters using k-means (random clusters), whereas "maximum" takes the markers with highest log p.values and select those for designing the model matrix.
`only.mark`	a TRUE/FALSE statement indicating if the only output should be the marker names or the incidence matrix. By default it returns the incidence matrix but if turned to TRUE will return only the marker names. Useful when want to identify significant markers in different populations.
`plotting`	a TRUE/FALSE statement indicating if the program should plot the GWAS showing which markers were selected.

Value

If all parameters are correctly indicated the program will return:

$X2 a design matrix with individuals in rows and markers in columns to be used for the GBLUP model based on the GWAS model provided.

Author(s)

Giovanny Covarrubias-Pazaran

References

Bernardo, Rex. 2014. Genomewide selection when major genes are known. Crop Science 54.1:68-75.

Gianola, Daniel, et al. 2014. Enhancing genome-enabled prediction by bagging genomic BLUP. PLoS One 9.4: e91693.

Abdollahi Arpanahi R, Morota G, Valente BD, Kranis A, Rosa GJM, Gianola D. 2015. Assessment of hitsging GBLUP for whole genome prediction of broiler chicken traits. Journal of Animal Breeding and Genetics 132:218-228.

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 `mmer2`

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with a single '#' mark,
#### remove them and run the examples
####============================================####
data(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno

####============================================####
#### convert markers to numeric format
####============================================####
## fit a model including additive and dominance effects
y <- CPpheno$color
Za <- diag(length(y))
A <- A.mat(CPgeno) # additive relationship matrix

####============================================####
#### compare prediction accuracies between
#### GBLUP and hits GBLUP
####============================================####
set.seed(1234)
```

```
y.trn <- y # for prediction accuracy
ww <- sample(c(1:dim(Za)[1]),72) # delete data for one fifth of the population
y.trn[ww] <- NA

####=========================================####
#### identify major genes and create the hit matrix
####=========================================####

#ETA.A <- list(list(Z=Za,K=A))
#ans.GWAS <- mmer(Y=y.trn, Z=ETA.A, W=CPgeno)
#summary(ans.GWAS)

####=========================================####
#### run the hits function to design the matrix
#### for top GWAS hits
####=========================================####

#X1 <- hits(ans.GWAS);head(X1); dim(X1)

#ETA.A <- list(list(Z=Za,K=A))
#ans.A <- mmer(Y=y.trn, Z=ETA.A) # GBLUP
#ans.AF <- mmer(Y=y.trn, X=X1, Z=ETA.A) # hitsging-GBLUP
#cor(ans.A$fitted.y[ww], y[ww], use="pairwise.complete.obs") # GBLUP
#cor(ans.AF$fitted.y[ww], y[ww], use="pairwise.complete.obs") # hitsging-GBLUP
#### little increase in prediction accuracy


##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####=========================================####
####=========================================####
#### EXAMPLE 2
#### simulate genotypic data
#### 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)
}
####=========================================####
#### simulate phenotypes with h2=0.5
####=========================================####
QTL <- 100*(1:5) #pick 5 QTL
u <- rep(0,1000) #marker effects
u[QTL] <- 1
g <- as.vector(crossprod(M,u))
h2 <- 0.5
y <- g + rnorm(200,mean=0,sd=sqrt((1-h2)/h2*var(g)))
```

```
M <- t(M)
####============================================####
#### Define random effects
####============================================####
Z1 <- diag(length(y))
ETA <- list( list(Z=Z1, K=A.mat(M)))
####============================================####
#### GWAS to extract hits matrix
####============================================####
#ans <- mmer(Y=y, Z=ETA, method="EMMA", W=M)
#X1 <- hits(gwasm=ans, nmar=10)

#iters=20
#res <- numeric(iters) # store results from GBLUP
#res2 <- numeric(iters) # store results from hitsGBLUP
#for(i in 1:iters){
#  y.trn <- y
#  ww <- sample(1:length(y),round(length(y)/5)) # delete 1/5 of the pop to predict
#  y.trn[ww] <- NA
#  ans <- mmer(Y=y.trn, Z=ETA, method="EMMA", silent=TRUE)
#  res[i] <- cor(ans$fitted.y[ww],y[ww])
#  ans2 <- mmer(Y=y.trn, X=X1, Z=ETA, method="EMMA", silent=TRUE)
#  res2[i] <- cor(ans2$fitted.y[ww],y[ww])
#}
#### compare predictive ability of the 2 methods
#boxplot(data.frame(GBLUP=res,hitsGBLUP=res2), col=2:3)
#### given the simulated h2 this should be the mean accuracy
#sqrt(.5)
#### but we got:
#apply(data.frame(GBLUP=res,hitsGBLUP=res2),2,mean)

#### The extreme difference in this example is because the QTLs
#### simulated have no linkage disequilibrium with the neighboring
#### markers. In the reality this wouldn't occur, and altough an
#### increase is expected will not be higher than 0-10 percent)
```

`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 functions of the package mmer and mmer2

Examples

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

---

is.diagonal.matrix *Test for diagonal square matrix*

Description

This function returns TRUE if the given matrix argument x is a square numeric matrix and that the off-diagonal elements are close to zero in absolute value to within the given tolerance level. Otherwise, a FALSE value is returned.

Usage

```
is.diagonal.matrix(x, tol = 1e-08)
```

Arguments

x	a numeric square matrix
tol	a numeric tolerance level usually left out

Value

A TRUE or FALSE value.

References

Bellman, R. (1987). *Matrix Analysis*, Second edition, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics.

Horn, R. A. and C. R. Johnson (1990). *Matrix Analysis*, Cambridge University Press.

Examples

```
A <- diag( 1, 3 )
is.diagonal.matrix( A )
B <- matrix( c( 1, 2, 3, 4 ), nrow=2, byrow=TRUE )
is.diagonal.matrix( B )
C <- matrix( c( 1, 0, 0, 0 ), nrow=2, byrow=TRUE )
is.diagonal.matrix( C )
```

---

is.square.matrix *Test for square matrix*

Description

The function returns TRUE if the argument is a square matrix and FALSE otherwise.

Usage

```
is.square.matrix(x)
```

Arguments

x	a matrix

Value

TRUE or FALSE

References

Bellman, R. (1987). *Matrix Analysis*, Second edition, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics.

Examples

```
A <- matrix( seq( 1, 12, 1 ), nrow=3, byrow=TRUE )
is.square.matrix( A )
B <- matrix( seq( 1, 16, 1 ), nrow=4, byrow=TRUE )
is.square.matrix( B )
```

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 functions of the package mmer and mmer2

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 2 elements; "by.LG" and "all.LG". 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.

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 mmer2

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(CPdata)
#### get the marker matrix
CPgeno <- CPdata$geno; CPgeno[1:5,1:5]
#### get the map
mapCP <- CPdata$map; 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")
```

MAI *Multivariate Average Information Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the average information (MAI) algorithm and the MVM argument is set to TRUE.

Usage

```
MAI(Y, X=NULL, ZETA=NULL, draw=TRUE, REML=TRUE, silent=FALSE, iters=20,
    init=NULL, tol=1e-3, che=TRUE, EIGEND=FALSE, forced=NULL, IMP=FALSE)
```

Arguments

Y	a matrix or data frame of response variables
X	an incidence matrix for fixed effects.
ZETA	incidence matrices and var-cov matrices for random effects. This works for ONE OR MORE random effects. **This needs to be provided as a 2-level list structure**. For example: ' ETA <- list(A=list(Z=Z1, K=K1),

B=list(Z=Z2, K=K2),
C=list(Z=Z3, K=K3)
)
'

makes a 2 level list for 3 the random effects A, B and C, stored in a variable we call ETA. The general idea is that each random effect is a list, i.e. A=list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix for the random effect, **if K is not provided is assumed an identity matrix** conferring independence.
'

PLEASE remember to **use the names Z and K for all random effects** when you provide your matrices, **that's the only way the program distinguishes between a Z or a K matrix**.
'

To provide extra detail, I'll rephrase it; when moving to situations of more than one random effect, you need to build a list for each random effect, and at the end everything gets joined in a list as well (BGLR type of format). Is called a 2-level list, i.e. A=list(Z=Z1, K=K1) and B=list(Z=Z2, K=K2) refers to 2 random effects and they should be put together in a list:
'

ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z1, K=K1))
'

Now you can fit your model as:
'

mod1 <- mmer(y=y, Z=ETA)
'

You can see the examples at the bottom to have a clearer idea how to fit your models.

`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

`REML` a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`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.

`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.

`tol` Convergence criteria. If the change in residual log likelihood for one cycle is less than `10 x tol` the algorithm finishes. If each component of the change proposed by the Newton-Raphson is lower in magnitude than `tol` the algorithm finishes. Default value is `1e-4`.

che — 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.

EIGEND — a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

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.

IMP — a TRUE/FALSE statement if the function should impute the Y matrix/dataframe with the median value or should get rid of missing values for the estimation of variance components.

Details

.

The likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

The algorithm can be summarized in the next steps:

.

1) provide initial values for the variance components

2) estimate the phenotypic variance matrix V = ZGZ + R

3) obtain Vinv by inverting V

4) obtain the projection matrix P = Vinv - [Vinv X (X'V-X)- X Vinv]

5) evaluate the logLikelihood as shown above

6) fill the average information matrix (MAI) with equation provided in Gilmour et al. (1995)

7) obtain MAI.inv by inverting MAI (the average information matrix)

8) calculate scores by first derivatives refer as "B" in Gilmour et al. (1995)

9) update the values of variance components by : k(i+1) = k(i) + [B(i) * MAI.inv]

10) steps are repeated in a while loop until convergence is reached, the likelihood doesn't increase anymore.

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

$ZETA random effect incidence and variace-covariance matrices

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

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

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

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

See Also

The core functions of the package mmer and mmer2

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(CPdata)
CPpheno <- CPdata$pheno[,-c(1:4)]
CPgeno <- CPdata$geno
### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
## fit a model including additive and dominance effects
Y <- CPpheno
Za <- diag(dim(Y)[1])
A <- A.mat(CPgeno) # additive relationship matrix
```

```
####================####
#### ADDITIVE MODEL ####
####================####
ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- MAI(Y=Y, ZETA=ETA.A)
#ans.A$var.comp
```

MAI2 *Multivariate Average Information Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the average information (MAI) algorithm and the MVM argument is set to TRUE.

Usage

```
MAI2(Y,X=NULL,ZETA=NULL,init=NULL,maxcyc=20,tol=1e-3, tolparinv=1e-6, draw=TRUE,
     silent=FALSE, constraint=FALSE,EIGEND=FALSE,forced=NULL,IMP=FALSE)
```

Arguments

Y	a matrix or data frame of response variables
X	an incidence matrix for fixed effects.
ZETA	incidence matrices and var-cov matrices for random effects. This works for ONE OR MORE random effects. **This needs to be provided as a 2-level list structure**. For example: ' ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z2, K=K2), C=list(Z=Z3, K=K3)) ' makes a 2 level list for 3 the random effects A, B and C, stored in a variable we call ETA. The general idea is that each random effect is a list, i.e. A=list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix for the random effect, **if K is not provided is assumed an identity matrix** conferring independence. ' PLEASE remember to **use the names Z and K for all random effects** when you provide your matrices, **that's the only way the program distinguishes between a Z or a K matrix**. '

To provide extra detail, I'll rephrase it; when moving to situations of more than one random effect, you need to build a list for each random effect, and at the end everything gets joined in a list as well (BGLR type of format). Is called a 2-level list, i.e. A=list(Z=Z1, K=K1) and B=list(Z=Z2, K=K2) refers to 2 random effects and they should be put together in a list:
'

ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z1, K=K1))
'

Now you can fit your model as:
'

mod1 <- mmer(y=y, Z=ETA)
'

You can see the examples at the bottom to have a clearer idea how to fit your models.

`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.

`maxcyc` Maximum number of cycles allowed. Default value is 50. A warning is output to the screen if this is reached before convergence.

`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

`tol` Convergence criteria. If the change in residual log likelihood for one cycle is less than `10 x tol` the algorithm finishes. If each component of the change proposed by the Newton-Raphson is lower in magnitude than `tol` the algorithm finishes. Default value is `1e-4`.

`tolparinv` tolerance parameter for matrix inverse

`silent` a TRUE/FALSE value indicating if the function should draw the progress bar while working or should not be displayed. The default is FALSE, which means is not silent and will display the progress bar.

`constraint` a TRUE/FALSE value indicating if the function should apply the boundary constraint indicating that variance components that are zero should be removed from the analysis and variance components recalculated.

`EIGEND` a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this method are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

`forced` a list of values for variance-covariance components to be used if the user wants to force those values.

`IMP` a TRUE/FALSE statement if the function should impute the Y matrix/dataframe with the median value or should get rid of missing values for the estimation of variance components.

Details

.

The likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

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

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

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 `mmer2`

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(CPdata)
CPpheno <- CPdata$pheno[,-c(1:4)]
CPgeno <- CPdata$geno
### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
## fit a model including additive and dominance effects
Y <- CPpheno
Za <- diag(dim(Y)[1])
A <- A.mat(CPgeno) # additive relationship matrix
####================####
#### ADDITIVE MODEL ####
####================####
ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- MAI2(Y=Y, ZETA=ETA.A)
#ans.A$var.comp
```

`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.

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 and mmer2

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 and mmer2

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")
```

---

matrix.trace *The trace of a matrix*

Description

This function returns the trace of a given square numeric matrix.

Usage

```
matrix.trace(x)
```

Arguments

x	a matrix

Value

A numeric value which is the sum of the values on the diagonal.

Note

If the argument x is not numeric, the function presents and error message and terminates. If the argument x is not a square matrix, the function presents an error message and terminates.

References

Bellman, R. (1987). *Matrix Analysis*, Second edition, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics.

Examples

```
A <- matrix( seq( 1, 16, 1 ), nrow=4, byrow=TRUE )
matrix.trace( A )
```

`maxi.qtl` *Peak search by first derivatives*

Description

This function find maximum peaks in a data frame from QTL mapping.

Usage

```
maxi.qtl(sma, distan=10, no.qtl=5, q95=2.5, LOD.int=FALSE, LODdrop=2, allCI=TRUE)
```

Arguments

`sma`	A data frame with 3 basic columns; "chr", "pos", "lod". Make sure your data frame has this 3 column names. Even though not all the times the scale is measure as LOD this will not affect the peak search, is just to help the program recognize which column has the signal intensity measurement.
`distan`	A numeric value indicating how far in distance (e.g. cM) we will consider a close QTL a different QTL since several points can form a QTL peak. The column "pos" is used to find such distances.
`no.qtl`	A numeric value indicating the maximum number of QTLs to look for in a chromosome.
`q95`	A numeric value indicating the threshold where we want to declare a peak significant in your scale (e.g. LOD, -log10(p.val), etc.).
`LOD.int`	A TRUE/FALSE statement indicating if the program should return all markers in the X-LOD interval drop. By default will construct a 2-LOD confidence interval but can be modified with the LOD.drop argument.
`LODdrop`	A numeric value indicating the LOD interval to construct.
`allCI`	A TRUE/FALSE statement indicating if the program should return all the markers within the X-LOD interval or only the edge markers along with the marker with the maximum LOD peak.

Details

No major details. Is just a maximum peak searcher controlling for distance.

Value

Retuns the biggest peaks for a data frame with chromosomes, position and intensities.

out a data frame with the positions of the highest peaks.

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 and mmer2

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, che=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

che	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` and `mmer2`

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno
### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
## fit a model including additive and dominance effects
Y <- CPpheno
Za <- diag(dim(Y)[1])
A <- A.mat(CPgeno) # 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
```

`mmer` *Mixed Model Equations in R*

Description

This function solves univariate and multivariate linear mixed models by likelihood methods. It has been **implemented to work directly with incidence and variance covariance matrices** for each random effect. Current optimization **methods** are; **efficient mixed model association (EMMA)** (Kang et al. 2008), **Direct-Inversion Average Information (AI)** (Lee et al. 2015), **MME-based expectation maximization (EM)** (Searle 1993; Bernardo 2010), **MME-based Average Information (AI)**(Gilmour et al. 1995). and the default **Direct-Inversion Newton-Raphson (NR)** (Tunnicliffe 1989). All algorithms are able to deal with multiple variance components. The package provides kernels to estimate additive (`A.mat`), dominance (`D.mat`), and epistatic (`E.mat`) relationship

matrices. The package provides flexibility to fit other genetic models such as full and half diallel models as well (see how to use the and functionality), and heterogeneous variances (at function). The core algorithms in sommer based in **Direct-Inversion surpasses in performance the MME-based only when dense covariance structures are present** (GBLUP and GWAS case). **With sparse matrices** (meaning when dense covariance structures don't exist), the MME-based algorithms are faster and we recommend to **shift to the use of method="EM" or method="AI" combined with DI="FALSE"** to change from direct-inversion to MME-based algorithms. **rrBLUP results can be recreated using the EMMA method**.

Finally, feel free to get in touch with me if you have any **questions, bug reports and suggestions** at: covarrubiasp@wisc.edu

For a short tutorial of how to perform different genetic analysis with sommer please type:

vignette("sommer")

Usage

```
mmer(Y, X=NULL, Z=NULL, R=NULL, W=NULL, method="NR", REML=TRUE, DI=TRUE,
     MVM=FALSE, iters=20, draw=FALSE, init=NULL, n.PC=0, P3D=TRUE,
     models="additive", ploidy=2, min.MAF = 0.05, silent=FALSE,
     family=NULL, constraint=TRUE, sherman=FALSE, EIGEND=FALSE,
     forced=NULL, map=NULL,fdr.level=0.05, manh.col=NULL,
     gwas.plots=TRUE, n.cores=1,tolpar = 1e-06,tolparinv = 1e-06,
     che=TRUE, IMP=TRUE)
```

Arguments

Y	a numeric **vector or matrix/dataframe** for the response variable(s) (it can run multivariate and also runs univariate in parallel by using the 'n.cores' argument). The **multivariate version is only enabled when the argument 'MVM' is set to TRUE** (not the default).
X	an incidence matrix or a list of incidence matrices (in the case of parallel models) for **fixed effects** related to environmental effects or experimental design. These can be constructed using the model.matrix function.
Z	incidence matrices and var-cov matrices for **random effects**. This works for ONE OR MORE random effects. **This needs to be provided as a 2-level list structure**. For example: ' ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z2, K=K2), C=list(Z=Z3, K=K3)) ' mod1 <- mmer(y=y, Z=ETA) ' makes a 2 level list for modeling 3 the random effects A, B and C, stored in a variable we named ETA (can have any name). The general idea is that each

random effect is a list, i.e. A=list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix for the random effect, **if K is not provided is assumed an identity matrix** confering independence.
'

PLEASE remember to **use only the names Z and K for all random effects** when you provide your matrices, **that's the only way the program distinguishes between a Z (incidence) or a K (covariance) matrix**. You can see the examples at the bottom to have a clearer idea how to fit your models.

`R` a list with R matrices used when the user wants heterogeneous residual variances, i.e. if user wants different residual variance components for each environment in a multienvironment analysis.

`W` an incidence matrix for extra fixed effects and **only to be used if GWAS is pursued** to fit a model of the form

$$y = X\beta + Zg + W\tau + \varepsilon$$

. Markers will be treated as fixed effects according to Yu et al. (2006) **for diploids**, and Rosyara et al (2016) **for polyploids**. For details about the methods please go to the Details section. The dashed line in the Manhattan plots corresponds to an FDR rate of 0.05 and is calculated using the p.adjust function included in the stats package.

`method` this refers to the method or algorithm to be used for estimating variance components. The package currently is supported by 4 algorithms; **EMMA** efficient mixed model association (Kang et al. 2008), **AI** average information (Gilmour et al. 1995; Lee et al. 2015), **EM** expectation maximization (Searle 1993; Bernardo 2010), and the default Newton-Raphson **NR** (Tunnicliffe 1989).

`REML` a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`DI` a TRUE/FALSE value indicating if the method (i.e. AI, NR, etc.) to estimate variance components should use direct inversion (DI=TRUE) or MME-based (DI=FALSE) techniques if available for such method.

`MVM` a TRUE/FALSE value **indicating if the model should be run as multivariate** or as parallel univariate models. the default is FALSE which will indicate to run parallel univariate models. The 'n.cores' argument decides how many cores use in such parallelization. MVM=TRUE will run a multivariate model but only for a single random effect.

`iters` a scalar value indicating **how many iterations** have to be performed if the optimization algorithms. There is no rule of tumb for the number of iterations but less than8 is usually enough. The default value is 20 iterations but usually will take less than that stopping before reaching the maximum number of iterations.

`draw` a TRUE/FALSE value indicating if a plot of **updated values for the variance components** and the log-likelihood **should be drawn or not** during the optimization process. The default is FALSE. It's been set to FALSE because is less the computation time when the computer doesn't have to draw plots.

`init` an vector of initial values for the EM, NR or AI algorithms. If not provided the program uses a starting values the variance(y)/#random.eff which are usually good starting values.

n.PC — when the user performs GWAS this refers to the number of principal components to include as fixed effects for Q + K model. Default is 0 (equals K model). See Details section.

P3D — when the user performs GWAS, P3D=TRUE means that the variance components are estimated by REML only once, without any markers in the model. When P3D=FALSE, variance components are estimated by REML for each marker separately. The default is the first case. See Details section.

models — **The model to be used in GWAS based on ploidy**. The default is the additive model which applies for diploids and polyploids but the model can be a vector with all possible models, i.e. "additive","1-dom-alt","1-dom-ref","2-dom-alt","2-dom-ref" models are supported for polyploids based on Rosyara (2016). See Details section.

ploidy — A numeric value indicating the ploidy level of the organism. The default is 2 which means diploid but higher ploidy levels are supported. This **is only relevant if you are performing GWAS in polyploids**.

min.MAF — when the user performs GWAS min.MAF is a scalar value between 0-1 indicating what is theminor allele frequency to be allowed for a marker provided in the argument "W". In general is known that results for markers with alleles with MAF < 0.05 are not reliable unless sample size is big enough.

silent — a TRUE/FALSE value indicating if the function should or shouldn't draw the **progress bar**. The default is FALSE, which means is not silent and will display the progress bar.

family — a family object to specify the distribution of the response variable. For details see family help page. The argument would look something like this; family=poisson(), or family=Gamma(), etc. For more sophisticated models please look at lme4 package from Douglas Bates. **NOT IMPLEMENTED YET**. Planned for ~ v2.5

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.

sherman — a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components.

EIGEND — a TRUE/FALSE value indicating if an **eigen decomposition for the additive relationship matrix** should be performed or not. This is based on Lee (2015). The limitations of this method are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

forced — a vector (univariate models) or list (multivariate models) 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. The default is NULL, meaning that variance components will be estimated by REML/ML.

`map` a data frame with 2 columns, 'Chrom', and 'Locus' to **create a nice manhattan plot when performing GWAS**. The map does not need to have same dimensions than the marker matrix. The function will look for markers in common among the W matrix and the map provided. Although, the association tests are performed for all markers, only the markers in common will be plotted.

`fdr.level` a level of **false discovery rate** to calculate and plot the line **in** the **GWAS** plot. Default is fdr.level=0.05. If there's not a single significant marker nothing is returned.

`manh.col` a vector with **colors desired for the manhattan plot**. Default are cadetblue and red alternated for each chromosome.

`gwas.plots` a TRUE/FALSE statement indicating if the **manhattan and qq plot should be drawn or not**. The default is TRUE but you may want to avoid it.

`n.cores` number of cores to use when the user passes multiple responses in the model for a faster execution of univariate models. The default is 1. It relies on forking and hence is **not available on Windows** unless mc.cores = 1.

`tolpar` tolerance parameter for convergence in the models.

`tolparinv` tolerance parameter for matrix inversion in the models.

`che` a TRUE/FALSE value statement indicating if the program should spend time **checking the matrices** of random effects provided by the user **for possible errors** in dimensions.

`IMP` a TRUE/FALSE statement if the function should impute the Y matrix/dataframe with the median value or should get rid of missing values for the estimation of variance components. **Only for multivariate mixed models**.

Details

The package has been equiped with several datasets to learn how to use the sommer package; the `HDdata` and `FDdata` datasets will introduce users to fit half and full diallel designs respectively. The `h2` dataset shows how to calculate heritability. The `cornHybrid` and `Technow_data` datasets contain data to teach users how to perform genomic prediction in hybrid single crosses which display GCA and SCA effects. The `wheatLines` dataset teaches how to do genomic prediction in single crosses in species displaying only additive effects. The `CPdata` dataset contains data to teach users how to fit genomic prediction models within a biparental population coming from 2 highly heterozygous parents including additive, dominance and epistatic effects. The same data example is used to show how to include the top GWAS hits as fixed effects in the GBLUP model to increase prediction accuracy, the examples can be found in the `hits` documentation. The `PolyData` dataset shows how to fit genomic prediction and GWAS analysis in polyploids. The second example in `Technow_data` data shows how to perform GWAS in single cross hybrids. A good converter from letter code to numeric format is implemented in the function `atcg1234`, which supports higher ploidy levels than diploid. The `RICE` dataset can teach users how to select the best training population using the `TP.prep` function in an applied breeding program, and we show comparison some comparison with other methods. Traces of selection can be obtained using markers with the `eigenGWAS` function. Additional examples for fitting mixed models, such as GWAS and others, can be found in the

example section of the mmer and mmer2 functions. Examples of multivariate models can be found in the example #3 of the CPdata. In addition, since v2.3 we have added the nna function which does nearest neighbor to create a new data frame adjusted for spatial variation. The ExpDesigns data contains several datasets to analyze experimental designs relevant to plant breeding and several detailed examples are available. Useful functions for analyzing such designs are included such as the blocker and fill.design.

==========

MODELS ENABLED

The mmer function fits linear mixed models by likelihood methods allowing the used of known covariance structures for random effects. If not provided independence is conferred. This program focuses in the mixed model of the form:

,

$$Y = X\beta + Zu + \epsilon$$

,

with distributions:

,

$$Y \ MVN(X\beta + Zu, var(Zu + \epsilon))$$

where;

,

$$\beta \ N(\beta, 0)$$

$$u \ N(0, G)$$

where G is equal to:

,

$$\begin{matrix} \text{K1*}\sigma\text{2(u1)} & 0 & 0 \\ 0 & \text{K2*}\sigma\text{2(u2)} & 0 \\ \ldots & \ldots & \ldots \\ 0 & 0 & \text{Ki*}\sigma\text{2(ui)} \end{matrix}$$

,

for the i.th random effects, allowing the user to specify the variance covariance structures in the K matrices and

,

$$\epsilon \ N(0, R)$$

where: $R = I * \sigma 2\epsilon$
,

also Var(Y) = Var(Zu+e) = ZGZ+R = V which is the phenotypic variance. Estimation of variance components are optimized with any of the 4 methods available; NR, AI, EM, EMMA (rrBLUP method).

==========

GWAS MODELS

The GWAS models in the sommer package are enabled by using the W argument, 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 W 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 "W". Multivariate GWAS models are based on Covarrubias-Pazaran et al. (2017) where covariance among traits are exploited as well.

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

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

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 $Var[\varepsilon] = I\sigma_e^2$

For unbalanced designs where phenotypes come from different environments, the environment mean can be modeled using the fixed option (e.g., fixed="env" if the column in the pheno data.frame is called "env"). When principal components are included (P+K model), the loadings are determined from an eigenvalue decomposition of the K matrix.

The argument "P3D" was introduced by Zhang et al. (2010). When P3D=FALSE, this function is equivalent to EMMA with REML (Kang et al. 2008). When P3D=TRUE, it is equivalent to EMMAX (Kang et al. 2010). The P3D=TRUE option is faster but can underestimate significance compared to P3D=FALSE.

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

==========

Additional functions for genetic analysis have been included such as False Discovery Rate calculation (`fdr`), plot of genetic maps (`map.plot`), creation of manhattan plots (`manhattan`), phasing F1 or CP populations (`phase.F1`). We have also included the famous `transp` function to get transparent colors.

Value

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

$var.comp 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

$LL LogLikelihood

$AIC Akaike information criterion

$BIC Bayesian information criterion

$X incidence matrix for fixed effects

$fitted.y Fitted values y.hat=XB+Zu

$fitted.u Fitted values only across random effects u.hat=Zu.1+....+Zu.i

$residuals Residual values e = y - XB or y - y.hat.fixed

$cond.residuals Conditional residual values e = y - (XB + Zu) or y - y.hat

$fitted.y.good Fitted values y.hat=XB+Zu only for data that had no missing data originally. Only used for my checks.

$Z incidence matrix for random effects. If more than one random effect this will be the column binding of individual Z matrices.

$K variance-covariance matrix for random effects. If more than one random effect this will be the diagonal binding of individual K matrices.

$fish.inv If was set to TRUE the Fishers information matrix will be in this slot.

$method The method for extimation of variance components specified by the user.

$maxim Maximization used. An argument for the program to know if REML or ML was used. If TRUE means that REML was used instead of ML.

$score the -log10(p-value) for each marker if a GWAS model is fitted by specifying the W parameter in the model.

$map if GWAS is performed and a map is provided the program will return a new map of the markers in common among the map and the W matrix and the -log10(p.values) for such marker tests.

Please share your ideas and code, future generations of scientists can be better trained if we are not greedy sharing our knowledge. Feel free to use my code for your own software! good luck with your analysis.

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 et al. Validating multivariate genomic selection and genome wide association in American cranberry. Submitted to the Plant Genome (2016).

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 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.

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
####============================================####

####============================================####
####============================================####
#### EXAMPLE 1
#### breeding values with 1 variance component
####============================================####
####============================================####

####============================================####
#### simulate genotypic data
#### 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)
}
####============================================####
#### simulate phenotypes
####============================================####
QTL <- 100*(1:5) #pick 5 QTL
u <- rep(0,1000) #marker effects
u[QTL] <- 1
g <- as.vector(crossprod(M,u))
h2 <- 0.5
y <- g + rnorm(200,mean=0,sd=sqrt((1-h2)/h2*var(g)))
M <- t(M)
```

```
####============================================####
#### fit the model
####============================================####
Z1 <- diag(length(y))
ETA <- list( add=list(Z=Z1, K=A.mat(M)) )
ans <- mmer(Y=y, Z=ETA)
summary(ans)

####============================================####
#### run the same but as GWAS
#### just add the marker matrix in the argument W
#### markers are fixed effects
####============================================####

#ans <- mmer(Y=y, Z=ETA, W=M)
#summary(ans)

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####============================================####
####============================================####
#### EXAMPLE 2
#### breeding values with 3 variance components
#### Hybrid prediction
####============================================####
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K
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)

colnames(Z1) <- levels(hybrid2$GCA1)
colnames(Z2) <- levels(hybrid2$GCA2)
colnames(Z3) <- levels(hybrid2$SCA)
####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
#K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
#K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####============================================####
#### Realized IBS relationships for cross
```

```
#### (as the Kronecker product of K1 and K2)
####=========================================####
#S <- kronecker(K1, K2) ; dim(S)
#rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

#ETA <- list(GCA1=list(Z=Z1, K=K1),
#            GCA2=list(Z=Z2, K=K2),
#            SCA=list(Z=Z3, K=S)
#            )
#ans <- mmer(Y=y, X=X1, Z=ETA)
#ans$var.comp
#summary(ans)

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####=========================================####
####=========================================####
#### EXAMPLE 3
#### use data from example 2
#### COMPARE WITH MCMCglmm
####=========================================####
####=========================================####

####=========================================####
#### the same model run in MCMCglmm:
####=========================================####
#library(MCMCglmm)
# pro <- list(GCA1 = as(solve(K1), "sparseMatrix"), GCA2 = as(solve(K2),
#      + "sparseMatrix"), SCA = as(solve(S), "sparseMatrix") )
#system.time(mox <- MCMCglmm(Yield ~ Location, random = ~ GCA1 + GCA2 + SCA,
#      + data = hybrid2, verbose = T, ginverse=pro))
## Takes 7:13 minutes in MCMCglmm, in sommer only takes 7 seconds

####=========================================####
#### it is also possible to do GWAS for hybrids, separatting
#### and accounting for effects of GCA1, GCA2, SCA
####=========================================####

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####=========================================####
####=========================================####
#### EXAMPLE 4
```

```
#### COMPARE WITH cpgen
####============================================####
####============================================####

#Z_list = list(Z1,Z2,Z3)
#G_list = list(solve(K1), solve(K2), solve(S))
#fit <- clmm(y = y, Z = Z_list, ginverse=G_list, niter=15000, burnin=5000)
####============================================####
#### inspect results and notice that variance
#### components were NOT estimated correctly!!
#### also takes longer and no user-friendly
####============================================####
#str(fit)

###########################################
###########################################
###########################################
###########################################
###########################################
###########################################

####============================================####
####============================================####
#### EXAMPLE 5
#### COMPARE WITH pedigreemm example
####============================================####
####============================================####

#library(pedigreemm)
#A <- as.matrix(getA(pedCowsR))
#y <- milk$milk
#Z1 <- model.matrix(~id-1, data=milk); dim(Z1)
#vv <- match(unique(milk$id), gsub("id","",colnames(Z1)))
#K1<- A[vv,vv]; dim(K1)
#Z2 <- model.matrix(~as.factor(herd)-1, data=milk); dim(Z2)
#ETA<- list(list(Z=Z1, K=K1),list(Z=Z2))
#fm3 <- mmer(Y=y, Z=ETA)
####============================================####
##### or using mmer2 would look:
####============================================####
#fm3 <- mmer2(fixed=milk ~ 1, random = ~ id + herd,
#             G=list(id=K1), data=milk)
#summary(fm3)
####============================================####
#### Try pedigreemm but takes longer,
#### is an extension of lme4
####============================================####
#fm2 <- pedigreemm(milk ~ (1 | id) + (1 | herd),data = milk, pedigree = list(id= pedCowsR))
#plot(fm3$u.hat[[1]], ranef(fm2)$id[,1])
#plot(fm3$u.hat[[2]], ranef(fm2)$herd[,1])
####============================================####
#### a big data frame with 3397 rows and 1359 animals analyzed
#### pedigreemm takes 4 min, sommer takes 1 minute
```

```
####=========================================####

##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####=========================================####
####=========================================####
#### EXAMPLE 6
#### PREDICTING SPECIFIC PERFORMANCE
#### within biparental population
####=========================================####
####=========================================####

#data(CPdata)
#CPpheno <- CPdata$pheno[,-c(1:4)]
#CPgeno <- CPdata$geno
## look at the data
#head(CPpheno)
#CPgeno[1:5,1:5]
####=========================================####
#### fit a model including additive and dominance effects
####=========================================####
#y <- CPpheno$color
#Za <- diag(length(y))
#Zd <- diag(length(y))
#A <- A.mat(CPgeno)
#D <- D.mat(CPgeno)

#y.trn <- y # for prediction accuracy
#ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
#y.trn[ww] <- NA

####================####
#### ADDITIVE MODEL ####
####================####
#ETA.A <- list(list(Z=Za,K=A))
#ans.A <- mmer(Y=y.trn, Z=ETA.A)
#cor(ans.A$fitted.y[ww], y[ww], use="pairwise.complete.obs")
####==========================####
#### ADDITIVE-DOMINANT MODEL ####
####==========================####
#ETA.AD <- list(list(Z=Za,K=A),list(Z=Zd,K=D))
#ans.AD <- mmer(Y=y.trn, Z=ETA.AD)
#cor(ans.AD$fitted.y[ww], y[ww], use="pairwise.complete.obs")
### greater accuracy !!!! 4 percent increment!!
### we run 100 iterations, 4 percent increment in general
####====================================####
#### ADDITIVE-DOMINANT-EPISTATIC MODEL ####
####====================================####
```

```
#ETA.ADE <- list(list(Z=Za,K=A),list(Z=Zd,K=D),list(Z=Ze,K=E))
#ans.ADE <- mmer(Y=y.trn, Z=ETA.ADE)
#cor(ans.ADE$fitted.y[ww], y[ww], use="pairwise.complete.obs")
#### adding more effects doesn't necessarily increase prediction accuracy!

########## NOTE
## nesting in R is indicated as
## assume blocks nested in locations
## Loc + Block/Loc
## is the same than
## Loc + Block + Loc:Block

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####============================================####
####============================================####
#### EXAMPLE 7
#### MULTIVARIATE MODEL
####============================================####
####============================================####
#data(CPdata)
#CPpheno <- CPdata$pheno[,-c(1:4)]
#CPgeno <- CPdata$geno
#### look at the data
#head(CPpheno)
#CPgeno[1:5,1:5]
#### fit a model including additive and dominance effects
#Y <- CPpheno
#Za <- diag(dim(Y)[1])
#A <- A.mat(CPgeno) # additive relationship matrix
####=================####
#### ADDITIVE MODEL ####
####=================####
#ETA.A <- list(add=list(Z=Za,K=A))
#### MAKE SURE YOU SET 'MVM=TRUE'
#ans.A <- mmer(Y=Y, Z=ETA.A, MVM=TRUE)
#summary(ans.A)

#### genetic variance covariance
#gvc <- ans.A$var.comp$Vu
#### extract variances (diagonals) and get standard deviations
#sd.gvc <- as.matrix(sqrt(diag(gvc)))
#### get possible products sd(Vgi) * sd(Vgi')
#prod.sd <- sd.gvc %*% t(sd.gvc) ###remove the \
#### genetic correlations cov(gi,gi')/[sd(Vgi) * sd(Vgi')]
#(gen.cor <- gvc/prod.sd)
#### heritabilities
#(h2 <- diag(gvc) / diag(cov(Y, use = "complete.obs")))
```

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

####======================================####
####======================================####
#### EXAMPLE 8
#### Genomic prediction
#### Univariate vs Multivariate models
####======================================####
####======================================####
data(CPdata)
CPpheno <- CPdata$pheno[,-c(1:4)]
CPgeno <- CPdata$geno
#### look at the data
#head(CPpheno)
#CPgeno[1:5,1:5]
#### fit a model including additive and dominance effects
#Za <- diag(dim(CPpheno)[1])
#A <- A.mat(CPgeno) # additive relationship matrix

#CPpheno2 <- CPpheno
#ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
#CPpheno2[ww,"Yield"] <- NA
####==================####
#### univariate model ####
####==================####
#ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- mmer(Y=CPpheno2$Yield, Z=ETA.A, method="NR")
#ans.A$var.comp
#cor(ans.A$fitted.y[ww], CPpheno[ww,"Yield"], use="pairwise.complete.obs")
####======================####
#### multivariate model ####
####======================####
#### estimate var.comp for subset
#ETA.B <- list(add=list(Z=diag(dim(A[-ww,-ww])[1]),K=A[-ww,-ww]))
#ans.B <- mmer(Y=CPpheno2[-ww,c(2,1,3)], Z=ETA.B,MVM=TRUE)
#ans.B$var.comp
#### now force the variance components for getting the plants with no data
#ETA.C <- list(add=list(Z=Za,K=A))
#ans.C <- mmer(Y=CPpheno2[,c(2,1,3)], Z=ETA.C, forced=ans.B$var.comp,MVM=TRUE)
#cor(ans.C$u.hat$add[ww,1], CPpheno[ww,"Yield"], use="pairwise.complete.obs")
```

---

mmer2 *Mixed Model Equations in R (formula-based mmer)*

Description

This function solves univariate and multivariate linear mixed models by likelihood methods. It has been **implemented to work with a data frame in a formula-based fashion** which will create the incidence and variance covariance matrices for the user and call the `mmer` function in the background. Current optimization **methods** are; **efficient mixed model association (EMMA)** (Kang et al. 2008), **Direct-Inversion Average Information (AI)** (Lee et al. 2015), **MME-based expectation maximization (EM)** (Searle 1993; Bernardo 2010), **MME-based Average Information (AI)**(Gilmour et al. 1995). and the default **Direct-Inversion Newton-Raphson (NR)** (Tunnicliffe 1989). All algorithms are able to deal with multiple variance components. The package provides kernels to estimate additive (`A.mat`), dominance (`D.mat`), and epistatic (`E.mat`) relationship matrices. The package provides flexibility to fit other genetic models such as full and half diallel models as well (see how to use the `and` functionality), and heterogeneous variances (`at` function). The core algorithms in sommer based in **Direct-Inversion surpasses in performance the MME-based only when dense covariance structures are present** (GBLUP and GWAS case). **With sparse matrices** (meaning when dense covariance structures don't exist), the MME-based algorithms are faster and we recommend to **shift to the use of method="EM" or method="AI" combined with DI="FALSE"** to change from direct-inversion to MME-based algorithms. **rrBLUP results can be recreated using the EMMA method**.

Finally, feel free to get in touch with me if you have any **questions, bug reports and suggestions** at: cova_ruber@live.com.mx

For a short tutorial of how to perform different genetic analysis with sommer please **type**:

vignette("sommer")

Usage

```
mmer2(fixed, random, rcov, data, G=NULL, W=NULL, method="NR", REML=TRUE, DI=TRUE,
      MVM=FALSE, iters=20, draw=FALSE, init=NULL, family=gaussian,
      silent=FALSE, constraint=TRUE, sherman=FALSE, EIGEND=FALSE,
      forced=NULL, map=NULL, fdr.level=0.05, manh.col=NULL, min.n=FALSE,
      gwas.plots=TRUE,n.cores=1,tolpar = 1e-06,tolparinv = 1e-06, IMP=TRUE,
      n.PC=0, P3D=TRUE, models="additive", ploidy=2, min.MAF=0.05)
```

Arguments

`fixed`	a formula specifying the **response** variable(s) **and fixed effects**, i.e. Yield ~ Location for **univariate** models, or cbind(color,Yield) ~ Location for **multivariate** models **or univariate in parallel**. For running multivariate the **'MVM' argument** needs to be set to TRUE.
`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 others: **at(.)** can be used to specify heterogeneous variance, i.e. random=~at(Location,c("3","4")):ID **diag(.)** can be used to specify a diagonal variance structure, i.e. random=~diag(Location):ID **us(.)** can be used to specify a unstructured variance, i.e. random=~us(Location):ID **and(.)** can be used to specify overlay of matrices, i.e. random=~male + and(female)

g(.) can be used to specify that a covariance structure for the random effect should be included, i.e. random=~g(male), random=~ g(female) + and(g(male)), etc.

`rcov` a formula specifying the name of the **error term**, i.e. rcov= ~ units or rcov=~at(.):units.

The functions that can be used to fit heterogeneous residual variances are:

at(.) can be used to specify heterogeneous variance, i.e. rcov=~at(Location):units

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

`G` a list containing the **variance-covariance matrices for the random effects**, i.e. if the user specifies:

random=~ g(genotype) + g(year)

then the G argument should be used as:

G=list(genotype=M1, year=M2)

where M1 and M2 are the variance-covariance matrices for random effects 'genotype' and 'year' respectively.

`W` an incidence matrix for extra fixed effects and **only to be used if GWAS is pursued** to fit a model of the form

$$y = X\beta + Zg + W\tau + \varepsilon$$

Markers will be treated as fixed effects according to Yu et al. (2006) **for diploids**, and Rosyara et al (2016) **for polyploids**. For details about the methods please go to the Details section. The dashed line in the Manhattan plots corresponds to an FDR rate of 0.05 and is calculated using the p.adjust function included in the stats package.

`method` this refers to the method or algorithm to be used for estimating variance components. The package currently is supported by 4 algorithms; **EMMA** efficient mixed model association (Kang et al. 2008), **AI** Direct-inversion average information (Gilmour et al. 1995; Lee et al. 2015), **EM** expectation maximization (Searle 1993; Bernardo 2010), and the default Direct-inversion Newton-Raphson **NR** (Tunnicliffe 1989).

`REML` a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`DI` a TRUE/FALSE value indicating if the method (i.e. AI, NR, etc.) to estimate variance components should use direct inversion (DI=TRUE) or MME-based (DI=FALSE) techniques if available for such method.

`MVM` a TRUE/FALSE value **indicating if the model should be run as multivariate** or as parallel univariate models. the default is FALSE which will indicate to run parallel univariate models. The 'n.cores' argument decides how many cores use in such parallelization. MVM=TRUE will run a multivariate model but only for a single random effect.

`iters` a scalar value indicating **how many iterations** have to be performed if the optimization algorithms. There is no rule of tumb for the number of iterations but less than8 is usually enough. The default value is 20 iterations but usually will take less than that stopping before reaching the maximum number of iterations.

`draw` a TRUE/FALSE value indicating if a plot of **updated values for the variance components** and the log-likelihood **should be drawn or not** during the optimization process. The default is FALSE. It's been set to FALSE because is less the computation time when the computer doesn't have to draw plots.

`init` an vector of initial values for the EM, NR or AI algorithms. If not provided the program uses a starting values the variance(y)/#random.eff which are usually good starting values.

`family` a family object to specify the distribution of the response variable. For details see `family` help page. The argument would look something like this; family=poisson(), or family=Gamma(), etc. For more sophisticated models please look at lme4 package from Douglas Bates. **NOT IMPLEMENTED YET**.

`silent` a TRUE/FALSE value indicating if the function should or shouldn't draw the **progress bar**. The default is FALSE, which means is not silent and will display the progress bar.

`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.

`sherman` a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components.

`EIGEND` a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

`forced` a vector (univariate models) or list of matrices (multivariate models) of numeric values for (co)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.

`map` a data frame with 2 columns, 'Chrom', and 'Locus' to **create a nice manhattan plot when performing GWAS**. The map does not need to have same dimensions that the marker matrix. The function will look for markers in common among the W matrix and the map provided. Although, the association tests are performed for all markers, only the markers in common will be plotted.

`fdr.level` a level of **false discovery rate** to calculate and plot the line in the GWAS plot. Default is fdr.level=0.05. If there's not a single significant marker nothing is returned.

`manh.col` a vector with **colors desired for the manhattan plot**. Default are cadetblue and red alternated for each chromosome. A function for drawing manhattan plots is available(`manhattan`).

`min.n` a TRUE/FALSE statement indicating if the constraint of number of levels of each grouping factor must be < number of observations. This is a constrained usually find in lme4 and has been added and set to TRUE but can be set to FALSE when only one measure per plant is available and the user wants to perform GWAS or genomic selection with limited data.

`gwas.plots` a TRUE/FALSE statement indicating if the **manhattan and qq plot should be drawn or not**. The default is TRUE but you may want to avoid it. A function for drawing manhattan plots is available(`manhattan`).

`n.cores` number of cores to use when the user passes multiple responses in the model for a faster execution. The default is 1. It relies on forking and hence is **not available on Windows** unless mc.cores = 1. Only useful for univariate models. For multivariate models operations cannot be parallelized.

`tolpar` tolerance parameter for convergence in the multivariate models.

`tolparinv` tolerance parameter for matrix inverse in the multivariate models.

`IMP` a TRUE/FALSE statement if the function should impute the Y matrix/dataframe of responses with the median value or should ignore rows with missing values for the estimation of variance-covariance components. Only for multivariate mixed models, not used in univariate.

`n.PC` when the user performs GWAS this refers to the number of principal components to include as fixed effects for Q + K model. Default is 0 (equals K model). See Details section.

`P3D` when the user performs GWAS, P3D=TRUE means that the variance components are estimated by REML only once, without any markers in the model. When P3D=FALSE, variance components are estimated by REML for each marker separately. The default is the first case. See Details section.

`models` **The model to be used in GWAS based on ploidy**. The default is the additive model which applies for diploids and polyploids but the model can be a vector with all possible models, i.e. "additive","1-dom-alt","1-dom-ref","2-dom-alt","2-dom-ref" models are supported for polyploids based on Rosyara (2016). See Details section.

`ploidy` A numeric value indicating the ploidy level of the organism to be used in GWAS for creating the incidence matrices when doing the marker tests. The default is 2 which means diploid, but higher ploidy levels are supported. This **is only relevant if you are performing GWAS in polyploids**.

`min.MAF` when the user performs GWAS min.MAF is a scalar value between 0-1 indicating what is theminor allele frequency to be allowed for a marker provided in the argument "W". In general is known that results for markers with alleles with MAF < 0.05 are not reliable unless sample size is big enough. Markers below this value will be ignored.

Details

The package has been equiped with several datasets to learn how to use the sommer package; the `HDdata` and `FDdata` datasets will introduce users to fit half and full diallel designs respectively. The `h2` dataset shows how to calculate heritability. The `cornHybrid` and `Technow_data` datasets contain data to teach users how to perform genomic prediction in hybrid single crosses which display GCA

and SCA effects. The wheatLines dataset teaches how to do genomic prediction in single crosses in species displaying only additive effects. The CPdata dataset contains data to teach users how to fit genomic prediction models within a biparental population coming from 2 highly heterozygous parents including additive, dominance and epistatic effects. The same data example is used to show how to include the top GWAS hits as fixed effects in the GBLUP model to increase prediction accuracy, the examples can be found in the hits documentation. The PolyData dataset shows how to fit genomic prediction and GWAS analysis in polyploids. The second example in Technow_data data shows how to perform GWAS in single cross hybrids. A good converter from letter code to numeric format is implemented in the function atcg1234, which supports higher ploidy levels than diploid. The RICE dataset can teach users how to select the best training population using the TP.prep function in an applied breeding program, and we show comparison some comparison with other methods. Traces of selection can be obtained using markers with the eigenGWAS function. Additional examples for fitting mixed models, such as GWAS and others, can be found in the example section of the mmer and mmer2 functions. Examples of multivariate models can be found in the example #3 of the CPdata. In addition, since v2.3 we have added the nna function which does nearest neighbor to create a new data frame adjusted for spatial variation. The ExpDesigns data contains several datasets to analyze experimental designs relevant to plant breeding and several detailed examples are available. Useful functions for analyzing such designs are included such as the blocker and fill.design.

==========

MODELS ENABLED

The mmer2 calls in the background the mmer function which fits linear mixed models by likelihood methods allowing the used of known covariance structures for random effects. If not provided independence is conferred. This program focuses in the mixed model of the form:

,

$$Y = X\beta + Zu + \epsilon$$

,

with distributions:

,

$$Y\ MVN(X\beta + Zu, var(Zu + \epsilon))$$

where;

,

$$\beta\ N(\beta, 0)$$

$$u\ N(0, G)$$

where G is equal to:

,

K1*σ2(u1)	0	0
0	K2*σ2(u2)	0
...	...	...
0	0	Ki*σ2(ui)

,

for the i.th random effects, allowing the user to specify the variance covariance structures in the K matrices and

,

$$\epsilon \, N(0, R)$$

where: $R = I * \sigma 2\epsilon$

,

also Var(Y) = Var(Zu+e) = ZGZ+R = V which is the phenotypic variance. Estimation of variance components are optimized with any of the 4 methods available; NR, AI, EM, EMMA (rrBLUP method).

GWAS MODELS

The GWAS models in the sommer package are enabled by using the W argument, 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 W 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 "W".

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

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

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 $Var[\varepsilon] = I\sigma_e^2$

For unbalanced designs where phenotypes come from different environments, the environment mean can be modeled using the fixed option (e.g., fixed="env" if the column in the pheno data.frame is called "env"). When principal components are included (P+K model), the loadings are determined from an eigenvalue decomposition of the K matrix.

The argument "P3D" was introduced by Zhang et al. (2010). When P3D=FALSE, this function is equivalent to EMMA with REML (Kang et al. 2008). When P3D=TRUE, it is equivalent to EMMAX (Kang et al. 2010). The P3D=TRUE option is faster but can underestimate significance compared to P3D=FALSE.

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

Additional functions for genetic analysis have been included such as False Discovery Rate calculation (fdr), plot of genetic maps (map.plot), creation of manhattan plots (manhattan), phasing F1 or CP populations (phase.F1). We have also included the famous transp function to get transparent colors.

Value

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

$var.comp 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

$LL LogLikelihood

$AIC Akaike information criterion

$BIC Bayesian information criterion

$X incidence matrix for fixed effects

$fitted.y Fitted values y.hat=XB+Zu

$fitted.u Fitted values only across random effects u.hat=Zu.1+....+Zu.i

$residuals Residual values e = y - XB or y - y.hat.fixed

$cond.residuals Conditional residual values e = y - (XB + Zu) or y - y.hat

$fitted.y.good Fitted values y.hat=XB+Zu only for data that had no missing data originally. Only used for my checks.

$Z incidence matrix for random effects. If more than one random effect this will be the column binding of individual Z matrices.

$K variance-covariance matrix for random effects. If more than one random effect this will be the diagonal binding of individual K matrices.

$fish.inv If was set to TRUE the Fishers information matrix will be in this slot.

$method The method for extimation of variance components specified by the user.

$maxim Maximization used. An argument for the program to know if REML or ML was used. If TRUE means that REML was used instead of ML.

$score the -log10(p-value) for each marker if a GWAS model is fitted by specifying the W parameter in the model.

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

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 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.

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
####============================================####

####============================================####
####============================================####
#### EXAMPLE 1
#### breeding values with 3 variance components
#### for hybrid prediction
####============================================####
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K
y <- hybrid2$Yield

####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
```

```
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####=========================================####
#### SCA relationship matrix (kronecker)
####=========================================####
S <- kronecker(K1, K2, make.dimnames=TRUE) ; dim(S)

#head(hybrid2)
#ans <- mmer2(Yield ~ Location, random = ~ g(GCA1) + g(GCA2) + g(SCA),
#       G=list(GCA1=K1, GCA2=K2, SCA=S),data=hybrid2)
#ans$var.comp
#summary(ans)

#### you can specify heterogeneus variances
# ans <- mmer2(fixed=Yield ~ 1, random = ~ GCA2 + at(Location):GCA2,
# data=hybrid2)
# summary(ans)

############################################################
############################################################
############################################################
############################################################
############################################################
############################################################

####=========================================####
####=========================================####
#### EXAMPLE 2
#### breeding values for 1 variance component
#### with variance covariance structure
####=========================================####
####=========================================####
data(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno
#### create the variance-covariance matrix
A <- A.mat(CPgeno)
#### look at the data and fit the model
head(CPpheno)
mix1 <- mmer2(Yield~1,random=~g(id), G=list(id=A), data=CPpheno)
summary(mix1)

############################################################
############################################################
############################################################
############################################################
############################################################
############################################################

####=========================================####
#### EXAMPLE 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(cornHybrid)
#hybrid2 <- cornHybrid$hybrid
#fm1 <- lmer(Yield ~ Location + (1|GCA1) + (1|GCA2) + (1|SCA),
#        data=hybrid2 )
#out <- mmer2(Yield ~ Location, random = ~ GCA1 + GCA2 + SCA,
#        data=hybrid2)
#summary(fm1)
#summary(out)
#### same BLUPs for GCA1, GCA2, SCA than lme4
#plot(out$cond.residuals, residuals(fm1))
#plot(out$u.hat[[1]], ranef(fm1)$GCA1[,1])
#plot(out$u.hat[[2]], ranef(fm1)$GCA2[,1])
#vv=which(abs(out$u.hat[[3]]) > 0)
#plot(out$u.hat[[3]][vv,], ranef(fm1)$SCA[,1])

#### a more complex model specifying which locations
#out2 <- mmer2(Yield ~ 1, random = ~ at(Location,c("3","4")):GCA2
#        + at(Location,c("3","4")):SCA,
#        data=hybrid2)
#summary(out2)

############################################################
############################################################
############################################################
############################################################
############################################################
############################################################

####==========================================####
#### EXAMPLE 4
#### comparison with lmer, install 'lme4'
#### and run the code below
####==========================================####

#library(lme4)
#data(sleepstudy)
#fm1 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
#summary(fm1)

#out <- mmer2(Reaction ~ Days, random = ~ Subject, data=sleepstudy)
#summary(out)

# plot(out$cond.residuals, residuals(fm1))
# plot(out$u.hat[[1]], ranef(fm1)[[1]][,1])

### specific variances for Subjects at Days
# sleepstudy$Days <- as.factor(sleepstudy$Days)
```

```
# out <- mmer2(Reaction ~ 1, random = ~ Subject + at(Days):Subject,
#              data=sleepstudy)
# summary(out)
##########################################################
##########################################################
##########################################################
##########################################################
##########################################################
##########################################################

####=========================================####
#### EXAMPLE 5
#### Multivariate model example
####=========================================####

# data(FDdata)
# head(FDdata)
#
# mix <- mmer2(cbind(stems,pods,seeds)~1,
#              random=~female+male, MVM=TRUE,data=FDdata)
# summary(mix)
# #### genetic variance covariance
# gvc <- mix$var.comp$female
# #### extract variances (diagonals) and get standard deviations
# sd.gvc <- as.matrix(sqrt(diag(gvc)))
# #### get possible products sd(Vgi) * sd(Vgi')
# prod.sd <- sd.gvc %*% t(sd.gvc)
# #### genetic correlations cov(gi,gi')/[sd(Vgi) * sd(Vgi')]
# (gen.cor <- gvc/prod.sd)
# #### pods and seeds have a strong negative genetic covariance (-.79)
# #### more pods, less seeds

##########################################################
##########################################################
##########################################################
##########################################################
##########################################################
##########################################################

####=========================================####
#### EXAMPLE 6
#### More on modeling
####=========================================####

# library(lme4)
# data(sleepstudy)
# head(sleepstudy)
# # try lme4
# (fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy))
# summary(fm1)
# # try sommer
# out <- mmer2(Reaction ~ Days, random = ~ Subject + at(Days):Subject, data=sleepstudy)
# summary(out)
```

```
# plot(out$cond.residuals, residuals(fm1))
# plot(out$u.hat$Subject, ranef(fm1)[[1]][,1])
# # now consider Days as factor by creating a new indicator variable
# sleepstudy$Days2 <- as.factor(sleepstudy$Days)
# out2 <- mmer2(Reaction ~ Days, random = ~ Subject + at(Days2):Subject, data=sleepstudy)
# summary(out2)
# plot(out2$cond.residuals, residuals(fm1))
# plot(out2$u.hat$Subject, ranef(fm1)[[1]][,1])
```

MMERM *Multivariate mixed model solver to be called inside mmer*

Description

This function estimates parameters for a model of the form

$$Y = BX + GZ + E$$

where Y is the $n.obsxr.responses$ matrix of response variable, X is a $n.obsxq.fix.effects$ known design matrix of fixed effects, Z is a $n.obsxl.ran.effects$ known design matrix of random effects, B is $q.fix.effectsxn.reponses$ matrix of fixed effects coefficients and G and E are independent matrix variate variables with $N_{dxl}(0, V_G, K)$ and $N_{dxn}(0, V_E, I_n)$ correspondingly. It also produces the BLUPs for the random effects G and some other statistics.

Usage

```
MMERM(Y, X=NULL, Z=NULL, W=NULL, method="NR", tolpar = 1e-06, tolparinv = 1e-06,
      draw=TRUE, REML=TRUE, silent=FALSE, iters=15, init=NULL, che=TRUE,
      EIGEND=FALSE, forced=NULL, P3D=TRUE, models="additive", ploidy=2,
      min.MAF=0.05, gwas.plots=TRUE, map=NULL,manh.col=NULL,fdr.level=0.05,
      constraint=TRUE, IMP=TRUE, n.PC=0)
```

Arguments

Y	a numeric vector for the response variable or a data frame of multiple responses (it can run multivariate and also runs univariate in parallel by using the 'n.cores' argument). The multivariate version is only enabled for a single random effect.
X	an incidence matrix for fixed effects related to environmental effects or experimental design. This has to be provided as a matrix, NOT in a list structure.
Z	incidence matrices and var-cov matrices for random effects. This works for ONE OR MORE random effects. THIS NEEDS TO BE PROVIDED AS A 2-LEVEL LIST STRUCTURE. For example: ' ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z2, K=K2), C=list(Z=Z3, K=K3)

)
'

makes a 2 level list for 3 the random effects A, B and C, stored in a variable we call ETA. The general idea is that each random effect is a list, i.e. A=list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix for the random effect, if K is not provided is assumed an identity matrix conferring independence.
'

PLEASE remember to use the names Z and K FOR ALL RANDOM EFFECTS when you provide your matrices, that's the only way the program distinguishes between a Z or a K matrix.
'

To provide extra detail, I'll rephrase it; when moving to situations of more than one random effect, you need to build a list for each random effect, and at the end everything gets joined in a list as well (BGLR type of format). Is called a 2-level list, i.e. A=list(Z=Z1, K=K1) and B=list(Z=Z2, K=K2) refers to 2 random effects and they should be put together in a list:
'

ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z1, K=K1))
'

Now you can fit your model as:
'

mod1 <- mmer(Y=y, Z=ETA)
'

You can see the examples at the bottom to have a clearer idea how to fit your models.

W an incidence matrix for extra fixed effects and only to be used if GWAS is desired and markers will be treated as fixed effects according to Yu et al. (2006) for diploids, and Rosyara et al (2016) for polyploids. Theoretically X and W are both fixed effects, but they are separated to perform GWAS in a model y = Xb + Zu + Wg, allowing the program to recognize the markers from other fixed factors such as environmental factors. This has to be provided as a matrix same than X.

Performs genome-wide association analysis based on the mixed model (Yu et al. 2006):

$$y = X\beta + Zg + W\tau + \varepsilon$$

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 $Var[\varepsilon] = I\sigma_e^2$

For unbalanced designs where phenotypes come from different environments, the environment mean can be modeled using the fixed option (e.g., fixed="env" if the column in the pheno data.frame is called "env"). When principal components are included (P+K model), the loadings are determined from an eigenvalue decomposition of the K matrix.

The terminology "P3D" (population parameters previously determined) was introduced by Zhang et al. (2010). When P3D=FALSE, this function is equivalent to EMMA with REML (Kang et al. 2008). When P3D=TRUE, it is equivalent to EMMAX (Kang et al. 2010). The P3D=TRUE option is faster but can underestimate significance compared to P3D=FALSE.

The dashed line in the Manhattan plots corresponds to an FDR rate of 0.05 and is calculated using the p.adjust function included in the stats package.

`method` multivariate method to apply. Currently only "EMMA-M" is available for a single random effect.

`tolpar` tolerance parameter for convergence

`tolparinv` tolerance parameter for matrix inverse

`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

`REML` a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

`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.

`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.

`che` 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.

`EIGEND` a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

`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.

`P3D` when the user performs GWAS, P3D=TRUE means that the variance components are estimated by REML only once, without any markers in the model. When P3D=FALSE, variance components are estimated by REML for each marker separately. The default is the first case.

models	The model to be used in GWAS. The default is the additive model which applies for diploids and polyploids but the model can be a vector with all possible models, i.e. "additive","1-dom-alt","1-dom-ref","2-dom-alt","2-dom-ref" models are supported for polyploids based on Rosyara (2016).
ploidy	A numeric value indicating the ploidy level of the organism. The default is 2 which means diploid but higher ploidy levels are supported. This should only be modified if you are performing GWAS in polyploids.
min.MAF	when the user performs GWAS min.MAF is a scalar value between 0-1 indicating what is theminor allele frequency to be allowed for a marker during a GWAS analysis when providing the matrix of markers W. In general is known that results for markers with alleles with MAF < 0.05 are not reliable unless sample size is big enough.
gwas.plots	a TRUE/FALSE statement indicating if the GWAS and qq plot should be drawn or not. The default is TRUE but you may want to avoid it.
map	a data frame with 2 columns, 'Chrom', and 'Locus' not neccesarily with same dimensions that markers. The program will look for markers in common among the W matrix and the map provided. Although, the association tests are performed for all markers, only the markers in common will be plotted.
manh.col	a vector with colors desired for the manhattan plot. Default are cadetblue and red alternated.
fdr.level	a level of FDR to calculate and plot the line in the GWAS plot. Default is fdr.level=0.05
constraint	a TRUE/FALSE value indicating if the function should apply the boundary constraint indicating that variance components that are zero should be removed from the analysis and variance components recalculated.
IMP	a TRUE/FALSE statement if the function should impute the Y matrix/dataframe with the median value or should get rid of missing values for the estimation of variance components. Only for multivariate mixed models.
n.PC	when the user performs GWAS this refers to the number of principal components to include as fixed effects for Q + K model. Default is 0 (equals K model).

Value

Vg	Estimate of V_G
Ve	Estimate of V_E
Bhat	BLUEs for B
Gpred	BLUPs for G
XsqtestB	χ^2 test statistics for testing whether the fixed effect coefficients are equal to zero.
pvalB	pvalues obtained from large sample theory for the fixed effects. We report the pvalues adjusted by the "padjust" function for all fixed effect coefficients.
XsqtestG	χ^2 test statistic values for testing whether the BLUPs are equal to zero.
pvalG	pvalues obtained from large sample theory for the BLUPs. We report the pvalues adjusted by the "padjust" function.
varGhat	Large sample variance for BLUPs.

varBhat	Large sample variance for the elements of B.
PEVGhat	Prediction error variance estimates for the BLUPs.

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####=========================================####
data(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno
### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
## fit a model including additive and dominance effects
Y <- CPpheno
Za <- diag(dim(Y)[1])
A <- A.mat(CPgeno) # additive relationship matrix
####================####
#### ADDITIVE MODEL ####
####================####
ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- MMERM(Y=Y, Z=ETA.A)
#ans.A$var.comp
```

---

mmerSNOW *Mixed Model Equations in R univariate*

Description

This function is used internally for the mmer function when univariate models are run in parallel. Please refer to the mmer function help page for more detail.

Finally, feel free to get in touch with me if you have any questions or suggestion at:

covarrubiasp@wisc.edu

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

Please report bugs and provide data to recreate the problem. The only way to improve open-source software is with the scientific community help.

Usage

```
mmerSNOW(y, X=NULL, Z=NULL, W=NULL, R=NULL, method="NR", REML=TRUE,DI=TRUE,
     iters=20, draw=FALSE, init=NULL, n.PC=0, P3D=TRUE,
     models="additive", ploidy=2, min.MAF = 0.05, silent=FALSE,
     family=NULL, constraint=TRUE, sherman=FALSE, EIGEND=FALSE,
```

```
Fishers=FALSE, gss=TRUE, forced=NULL, full.rank=TRUE,
map=NULL,fdr.level=0.05, manh.col=NULL, gwas.plots=TRUE,
tolpar = 1e-04,tolparinv = 1e-06)
```

Arguments

y	a numeric vector for the response variable
X	an incidence matrix for fixed effects related to environmental effects or experimental design. This has to be provided as a matrix, NOT in a list structure.
Z	incidence matrices and var-cov matrices for random effects. This works for ONE OR MORE random effects. THIS NEEDS TO BE PROVIDED AS A 2-LEVEL LIST STRUCTURE. For example: . ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z2, K=K2), C=list(Z=Z3, K=K3)) . makes a 2 level list for 3 the random effects A, B and C, stored in a variable we call ETA. The general idea is that each random effect is a list, i.e. A=list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix for the random effect, if K is not provided is assumed an identity matrix conferring independence. . PLEASE remember to use the names Z and K FOR ALL RANDOM EFFECTS when you provide your matrices, that's the only way the program distinguishes between a Z or a K matrix. . To provide extra detail, I'll rephrase it; when moving to situations of more than one random effect, you need to build a list for each random effect, and at the end everything gets joined in a list as well (BGLR type of format). Is called a 2-level list, i.e. A=list(Z=Z1, K=K1) and B=list(Z=Z2, K=K2) refers to 2 random effects and they should be put together in a list: . ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z1, K=K1)) . Now you can fit your model as: . mod1 <- mmerSNOW(y=y, Z=ETA) . You can see the examples at the bottom to have a clearer idea how to fit your models.

W — an incidence matrix for extra fixed effects and only to be used if GWAS is desired and markers will be treated as fixed effects according to Yu et al. (2006) for diploids, and Rosyara et al (2016) for polyploids. Theoretically X and W are both fixed effects, but they are separated to perform GWAS in a model y = Xb + Zu + Wg, allowing the program to recognize the markers from other fixed factors such as environmental factors. This has to be provided as a matrix same than X.

R — a matrix for variance-covariance structures for the residuals, i.e. for longitudinal data. if not passed is assumed an identity matrix. THIS PART STILLS IN DEVELOPMENT, NOT FUNCTIONAL YET, it is plan to be implemented in version 1.6.

method — this refers to the method or algorithm to be used for estimating variance components. The package currently is supported by 4 algorithms; "EMMA" efficient mixed model association (Kang et al. 2008), "AI" average information (Gilmour et al. 1995; Lee et al. 2015), "EM" expectation maximization (Searle 1993; Bernardo 2010), and Newton-Raphson "NR" (Tunnicliffe 1989). The default method is average information "AI" because of its ability to handle multiple random effects and its greater speed compared to "EM", "NR" and "EMMA" which can handle multiple random effects but are slower in dense models.

REML — a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.

DI — a TRUE/FALSE value indicating if the method to estimate variance components should be direct inversion (DI=TRUE) or MME-based (DI=FALSE) if available for such method.

iters — a scalar value indicating how many iterations have to be performed if the EM or AI algorithms are selected. There is no rule of tumb for the number of iterations. The default value is 50 iterations or EM steps, but usually will take less than that stopping before reaching the maximum number of iterations. For the AI algorithm usually takes just a few iterations.

draw — a TRUE/FALSE value indicating if a plot of updated values for the variance components and the log-likelihood should be drawn or not during the optimization process. The default is FALSE. It's been set to FALSE because is less the computation time when the computer doesn't have to draw plots.

init — an vector of initial values for the EM, NR or AI algorithms. If not provided the program uses a starting values the variance(y)/#random.eff which are usually good starting values.

n.PC — when the user performs GWAS this refers to the number of principal components to include as fixed effects for Q + K model. Default is 0 (equals K model).

P3D — when the user performs GWAS, P3D=TRUE means that the variance components are estimated by REML only once, without any markers in the model. When P3D=FALSE, variance components are estimated by REML for each marker separately. The default is the first case.

models — The model to be used in GWAS. The default is the additive model which applies for diploids and polyploids but the model can be a vector with all possible models, i.e. "additive","1-dom-alt","1-dom-ref","2-dom-alt","2-dom-ref" models are supported for polyploids based on Rosyara (2016).

ploidy — A numeric value indicating the ploidy level of the organism. The default is 2 which means diploid but higher ploidy levels are supported. This should only be modified if you are performing GWAS in polyploids.

min.MAF — when the user performs GWAS min.MAF is a scalar value between 0-1 indicating what is theminor allele frequency to be allowed for a marker during a GWAS analysis when providing the matrix of markers W. In general is known that results for markers with alleles with MAF < 0.05 are not reliable unless sample size is big enough.

silent — a TRUE/FALSE value indicating if the function should draw the progress bar and poems (see poe function) while working or should not be displayed. The default is FALSE, which means is not silent and will display the progress bar and a short poem to help the scientist (and me haha) remember that life is more than analyzing data.

family — a family object to specify the distribution of the response variable. The program will only use the link function to transform the response. For details see family help page. The argument would look something like this; family=poisson(), or family=Gamma(), etc. For more sophisticated models please look at lme4 package from Douglas Bates. NOT IMPLEMENTED YET. Planned for ~ v1.8

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.

sherman — a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components. This will perform faster when a mixed model with no covariance structures is fitted (only AI algorithm). The default is FALSE since this software was designed for unreplicated data (altough can fit models with replicated data but slower than lme4).

EIGEND — a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this method are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

Fishers — a TRUE/FALSE value indicating if the program should calculate at the final step and return the inverse of the Fishers Information Matrix.

gss — a TRUE/FALSE value indicating if a genomic selection is being fitted just for using certain constraints. When is FALSE the program can make some EM steps to find initial values for variance components when the starting values are to far from the real values causing the likelihood to have a strange behavior and dropping dramatically When TRUE (default) the program does not try EM steps

even when far away from the likelihood because in big marker-based models can make the process quite slow.

`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.

`full.rank` a TRUE/FALSE value indicating if the program should investigate X'X to be full rank to avoid problems when solving the linear system. By default this is TRUE which will display a message in the console to le the user know if the X is full rank or not. and will remove extra columns until full rank condition is met. This could not like some users so it can be desactivated but will return an error anyways at some point due to the fact that X'X is not be invertible. This condition is analyzed once missing data in the response variable 'y' has been removed, not in the original X matrix.

`map` a data frame with 2 columns, 'Chrom', and 'Locus' not neccesarily with same dimensions that markers. The program will look for markers in common among the W matrix and the map provided. Although, the association tests are performed for all markers, only the markers in common will be plotted.

`fdr.level` a level of FDR to calculate and plot the line in the GWAS plot. Default is fdr.level=0.05

`manh.col` a vector with colors desired for the manhattan plot. Default are cadetblue and red alternated.

`gwas.plots` a TRUE/FALSE statement indicating if the GWAS and qq plot should be drawn or not. The default is TRUE but you may want to avoid it.

`tolpar` tolerance parameter for convergence in the multivariate models.

`tolparinv` tolerance parameter for matrix inverse in the multivariate models.

Details

The package has been developed to provide R users with code to understand how most common algorithms in mixed model analysis work related to genetics field, but also allowing to perform their real analysis. This package allows the user to calculate the variance components for a mixed model with the advantage of specifying the variance-covariance structure of the random effects. This program focuses in the mixed model of the form:
,

$$Y = X\beta + Zu + \epsilon$$

,

with distributions:
,

$$Y\ MVN(X\beta + Zu, var(Zu + \epsilon))$$

where;

,

$$\beta \; N(\beta, 0)$$

$$u \; N(0, G)$$

where G is equal to:
,

K1*σ2(u1)	0	0
0	K2*σ2(u2)	0
...	...	...
0	0	Ki*σ2(ui)

,

for the i.th random effects, allowing the user to specify the variance covariance structures in the K matrices and
,

$$\epsilon \; N(0, R)$$

where: $R = I * \sigma 2\epsilon$
,

This mixed model would be specified in the mmerSNOW function as:

.

X1 <- matrix(1,length(y),1) incidence matrix for intercept only

ETA <- list(gca1=list(Z=Z1, K=K1), gca2=list(Z=Z2, K=K2), sca=list(Z=Z3, K=K3)) for 3 random effects

.

where Z1, Z2, Z3 are incidence matrices for GCA1, GCA2, SCA respectively created using the model.matrix function and K1, K2, K3 are their var-cov matrices. Now the fitted model will be typed as:

.

ans <- mmerSNOW(y=y, X=X1, Z=ETA)

or

ans <- mmerSNOW2(y~1, random= ~ gca1 + gca2 + sca, G=list(gca1=K1, gca2=K2, sca=K3), data=yourdata)

FOR DETAILS ON HOW THE "AI", EM" AND "EMMA" ALGORITHMS WORK PLEASE REFER TO AI , EM AND EMMA

In addition, the package contains a very nice function to plot genetic maps with numeric variable or traits next to the LGs, see the map.plot function to see how easy can be done. The package contains other functions:

transp function transform a vector of colors in transparent colors.

fdr calculates the false discovery rate for a vector of p-values.

A.mat is a wrapper of the A.mat function from the rrBLUP package.

D.mat calculates the dominant relationship matrix.

E.mat calculates de epistatic relationship matrix.

score.calc is a function that can be used to calculate a -log10 p-value for a vector of BLUEs for marker effects.

Other functions such as summary, fitted, randef (notice sommer uses randef not ranef), anova, residuals, coef and plot applicable to typical linear models can also be applied to models fitted using this function which is the core of the sommer package.

Value

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

$var.comp 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

$LL LogLikelihood

$AIC Akaike information criterion

$BIC Bayesian information criterion

$X incidence matrix for fixed effects

$fitted.y Fitted values y.hat=XB+Zu

$fitted.u Fitted values only across random effects u.hat=Zu.1+....+Zu.i

$residuals Residual values e = y - XB or y - y.hat.fixed

$cond.residuals Conditional residual values e = y - (XB + Zu) or y - y.hat

$fitted.y.good Fitted values y.hat=XB+Zu only for data that had no missing data originally. Only used for my checks.

$Z incidence matrix for random effects. If more than one random effect this will be the column binding of individual Z matrices.

$K variance-covariance matrix for random effects. If more than one random effect this will be the diagonal binding of individual K matrices.

$fish.inv If was set to TRUE the Fishers information matrix will be in this slot.

$method The method for extimation of variance components specified by the user.

$maxim Maximization used. An argument for the program to know if REML or ML was used. If TRUE means that REML was used instead of ML.

$score the -log10(p-value) for each marker if a GWAS model is fitted by specifying the W parameter in the model.

$map if GWAS is performed and a map is provided the program will return a new map of the markers in common among the map and the W matrix and the -log10(p.values) for such marker tests.

In addition, we have included a couple of random poems from Latin American writers to help the scientist (an me haha) remember from time to time that life is more than analyzing data. You can always silence this feature by setting the argument silent=TRUE, which will avoid the program to display the poems. If you want to contribute with a poem, phrase or short citation for future versions of sommer, feel free to send it to me to:

covarrubiasp@wisc.edu

Please share your ideas and code, future generations of scientists can be better if we are not greedy sharing our knowledge. Feel free to use my code for your own software! good luck with your analysis.

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

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

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
####=========================================####

####=========================================####
####=========================================####
#### EXAMPLE 1
#### breeding values with 1 variance component
####=========================================####
####=========================================####

####=========================================####
#### simulate genotypic data
#### 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)
}
####=========================================####
#### simulate phenotypes
####=========================================####
QTL <- 100*(1:5) #pick 5 QTL
u <- rep(0,1000) #marker effects
u[QTL] <- 1
g <- as.vector(crossprod(M,u))
h2 <- 0.5
y <- g + rnorm(200,mean=0,sd=sqrt((1-h2)/h2*var(g)))
M <- t(M)
####=========================================####
#### fit the model
####=========================================####
Z1 <- diag(length(y))
ETA <- list( list(Z=Z1, K=A.mat(M)))
ans <- mmerSNOW(y=y, Z=ETA, method="EMMA")
summary(ans)

####=========================================####
#### run the same but as GWAS
#### just add the marker matrix in the argument W
#### markers are fixed effects
####=========================================####

#ans <- mmerSNOW(y=y, Z=ETA, W=M, method="EMMA")
#summary(ans)
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####=========================================####
####=========================================####
```

```
#### EXAMPLE 2
#### breeding values with 3 variance components
#### Hybrid prediction
####============================================####
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K
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)

####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
#K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
#K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####============================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####============================================####
#S <- kronecker(K1, K2) ; dim(S)
#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 <- mmerSNOW(y=y, X=X1, Z=ETA)
#ans$var.comp
#summary(ans)

############################################
############################################
############################################
############################################
############################################
############################################

####============================================####
####============================================####
#### EXAMPLE 3
#### COMPARE WITH MCMCglmm
####============================================####
####============================================####

####============================================####
#### the same model run in MCMCglmm:
####============================================####
#library(MCMCglmm)
# pro <- list(GCA1 = as(solve(K1), "sparseMatrix"), GCA2 = as(solve(K2),
```

```
#      + "sparseMatrix"), SCA = as(solve(S), "sparseMatrix") )
#system.time(mox <- MCMCglmm(Yield ~ Location, random = ~ GCA1 + GCA2 + SCA,
#      + data = hybrid2, verbose = T, ginverse=pro))
## Takes 7:13 minutes in MCMCglmm, in sommer only takes 7 seconds

####=========================================####
#### it is also possible to do GWAS for hybrids, separatting
#### and accounting for effects of GCA1, GCA2, SCA
####=========================================####

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####=========================================####
####=========================================####
#### EXAMPLE 4
#### COMPARE WITH cpgen
####=========================================####
####=========================================####

#Z_list = list(Z1,Z2,Z3)
#G_list = list(solve(K1), solve(K2), solve(S))
#fit <- clmm(y = y, Z = Z_list, ginverse=G_list, niter=15000, burnin=5000)
####=========================================####
#### inspect results and notice that variance
#### components were NOT estimated correctly!!
#### also takes longer and no user-friendly
####=========================================####
#str(fit)

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####=========================================####
####=========================================####
#### EXAMPLE 5
#### COMPARE WITH pedigreemm example
####=========================================####
####=========================================####

#library(pedigreemm)
#A <- as.matrix(getA(pedCowsR))
#y <- milk$milk
#Z1 <- model.matrix(~id-1, data=milk); dim(Z1)
#vv <- match(unique(milk$id), gsub("id","",colnames(Z1)))
```

```
#K1<- A[vv,vv]; dim(K1)
#Z2 <- model.matrix(~as.factor(herd)-1, data=milk); dim(Z2)
#ETA<- list(list(Z=Z1, K=K1),list(Z=Z2))
#fm3 <- mmerSNOW(y=y, Z=ETA)

####=========================================####
#### Try pedigreemm but takes longer,
#### is an extension of lme4
####=========================================####
#fm2 <- pedigreemm(milk ~ (1 | id) + (1 | herd),data = milk, pedigree = list(id= pedCowsR))
#plot(fm3$u.hat[[1]], ranef(fm2)$id[,1])
#plot(fm3$u.hat[[2]], ranef(fm2)$herd[,1])
####=========================================####
#### a big data frame with 3397 rows and 1359 animals analyzed
#### pedigreemm takes 4 min, sommer takes 1 minute
####=========================================####

##############################################
##############################################
##############################################
##############################################
##############################################
##############################################

####=========================================####
####=========================================####
#### EXAMPLE 6
#### PREDICTING SPECIFIC PERFORMANCE
#### within biparental population
####=========================================####
####=========================================####

#data(CPdata)
#CPpheno <- CPdata$pheno
#CPgeno <- CPdata$geno
## look at the data
#head(CPpheno)
#CPgeno[1:5,1:5]
####=========================================####
#### fit a model including additive and dominance effects
####=========================================####
#y <- CPpheno$color
#Za <- diag(length(y))
#Zd <- diag(length(y))
#A <- A.mat(CPgeno)
#D <- D.mat(CPgeno)

#y.trn <- y # for prediction accuracy
#ww <- sample(c(1:dim(Za)[1]),72) # delete data for 1/5 of the population
#y.trn[ww] <- NA

####================####
#### ADDITIVE MODEL ####
```

```
####================####
#ETA.A <- list(list(Z=Za,K=A))
#ans.A <- mmerSNOW(y=y.trn, Z=ETA.A)
#cor(ans.A$fitted.y[ww], y[ww], use="pairwise.complete.obs")
####============================####
#### ADDITIVE-DOMINANT MODEL ####
####============================####
#ETA.AD <- list(list(Z=Za,K=A),list(Z=Zd,K=D))
#ans.AD <- mmerSNOW(y=y.trn, Z=ETA.AD)
#cor(ans.AD$fitted.y[ww], y[ww], use="pairwise.complete.obs")
### greater accuracy !!!! 4 percent increment!!
### we run 100 iterations, 4 percent increment in general
####====================================####
#### ADDITIVE-DOMINANT-EPISTATIC MODEL ####
####====================================####
#ETA.ADE <- list(list(Z=Za,K=A),list(Z=Zd,K=D),list(Z=Ze,K=E))
#ans.ADE <- mmerSNOW(y=y.trn, Z=ETA.ADE)
#cor(ans.ADE$fitted.y[ww], y[ww], use="pairwise.complete.obs")
#### adding more effects doesn't necessarily increase prediction accuracy!

########## NOTE
## nesting in R is indicated as
## assume blocks nested in locations
## Loc + Block/Loc
## is the same than
## Loc + Block + Loc:Block
```

MNR *Multivariate Newton-Raphson Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the Newton-Raphson (MNR) algorithm for multivariate models (multiple responses).

Usage

```
MNR(Y,X=NULL,ZETA=NULL,R=NULL,init=NULL,maxcyc=20,tol=1e-3,
     tolparinv=1e-6,draw=FALSE, silent=FALSE, constraint=FALSE,
     EIGEND=FALSE,forced=NULL, IMP=FALSE)
```

Arguments

Y	a matrix or data frame of response variables
X	an incidence matrix for fixed effects.

ZETA

incidence matrices and var-cov matrices for random effects. This works for ONE OR MORE random effects. **This needs to be provided as a 2-level list structure**. For example:
'

ETA <- list(
A=list(Z=Z1, K=K1),
B=list(Z=Z2, K=K2),
C=list(Z=Z3, K=K3)
)
'

makes a 2 level list for 3 the random effects A, B and C, stored in a variable we call ETA. The general idea is that each random effect is a list, i.e. A=list(Z=Z1, K=K1) where Z is the incidence matrix and K the var-cov matrix for the random effect, **if K is not provided is assumed an identity matrix** conferring independence.
'

PLEASE remember to **use the names Z and K for all random effects** when you provide your matrices, **that's the only way the program distinguishes between a Z or a K matrix**.
'

To provide extra detail, I'll rephrase it; when moving to situations of more than one random effect, you need to build a list for each random effect, and at the end everything gets joined in a list as well (BGLR type of format). Is called a 2-level list, i.e. A=list(Z=Z1, K=K1) and B=list(Z=Z2, K=K2) refers to 2 random effects and they should be put together in a list:
'

ETA <- list(A=list(Z=Z1, K=K1), B=list(Z=Z1, K=K1))
'

Now you can fit your model as:
'

mod1 <- mmer(Y=y, Z=ETA)
'

You can see the examples at the bottom to have a clearer idea how to fit your models.

R

list of R matrices to correct for residual. Internally the program will do the kronecker product of such matrices to create R.

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.

maxcyc

Maximum number of cycles allowed. Default value is 50. A warning is output to the screen if this is reached before convergence.

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

`tol`	Convergence criteria. If the change in residual log likelihood for one cycle is less than `10 x tol` the algorithm finishes. If each component of the change proposed by the Newton-Raphson is lower in magnitude than `tol` the algorithm finishes. Default value is `1e-4`.
`tolparinv`	tolerance parameter for matrix inverse
`silent`	a TRUE/FALSE value indicating if the function should draw the progress bar while working or should not be displayed. The default is FALSE, which means is not silent and will display the progress bar.
`constraint`	a TRUE/FALSE value indicating if the function should apply the boundary constraint indicating that variance components that are zero should be removed from the analysis and variance components recalculated.
`EIGEND`	a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this method are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.
`forced`	a list of values for variance-covariance components to be used if the user wants to force those values.
`IMP`	a TRUE/FALSE statement if the function should impute the Y matrix/dataframe with the median value or should get rid of missing values for the estimation of variance components.

Details

.

The likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

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

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

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 mmer2

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(CPdata)
CPpheno <- CPdata$pheno[,-c(1:4)]
CPgeno <- CPdata$geno
### look at the data
head(CPpheno)
CPgeno[1:5,1:5]
## fit a model including additive and dominance effects
Y <- CPpheno
Za <- diag(dim(Y)[1])
A <- A.mat(CPgeno) # additive relationship matrix
####================####
#### ADDITIVE MODEL ####
####================####
ETA.A <- list(add=list(Z=Za,K=A))
#ans.A <- MNR(Y=Y, ZETA=ETA.A)
#ans.A$var.comp
```

`my.colors` *All typical colors in R easy to access.*

Description

This dataset is just a vector of the different colors in R from the pdf available at CRAN. Just to make easier the access to them.

Usage

```
data("my.colors")
```

Format

The format is: chr "my.colors"

Source

This data is from CRAN.

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. https://cran.rstudio.com/web/packages/sommer/

See Also

The core function of the package `mmer`

Examples

```
####============================================####
####============================================####
data(my.colors)
set.seed(1234)
palette <- sample(my.colors, 16) # sample some
palette2 <- transp(palette) # make them transparent
ma <- matrix(1:16,4,4)
layout(matrix(1:2,1,2))
image(ma, col=palette)
image(ma, col=palette2)
layout(matrix(1,1,1))
####============================================####
####============================================####
```

`name.change` *renaming a vector by adding zeros*

Description

This function is a very simple function to rename a vector that was named wrong and zeros were not added by mistake, i.e.:

x <- c("P1","P9","P10","P99","P187")

but the user desires the a vector of the form:

x <- c("P001","P009","P010","P099","P187")

this function does that simple task.

Usage

```
name.change(x, separ="P", maxn=999)
```

Arguments

`x`	a character vector with the names.
`separ`	a character argument indicating what comes before the numbers that need to be corrected.
`maxn`	maximum number of individual-label found in the vector. This will be important for the function to know how many zeros should be added.

Value

$x a character vector with the names corrected by the added zeros.

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

```
x <- c("P1","P9","P10","P99","P187")
## try adding zeros by indicating that 999 could be the max number
name.change(x, sep="P",maxn=999)
name.change(x, sep="P",maxn=9999)
```

nna *Nearest neighbour adjustment*

Description

This function takes a dataframe that contains the variables 'row', 'col', and 'y' to create a new variable 'nnx' which is a new covariable to be used in the linear model based on the neighboring rows and columns. By default it uses the formula used by Lado et al. (2013) where the new variable is defined as:

nnx_i = yi - mean(y1,y2,...,y6)

where spatially such plants will look like:

[...] [....] [y2] [...] [...]

[y5] [y1] [yi] [y3] [y6]

[...] [....] [y4] [...] [...]

Corners and edge plots are treated by only using the neighbor plots available, i.e.:

[y2] [...] [...]

[yi] [y3] [y4]

[y1] [...] [...]

The number of plants to use in the adjustment can be modified with the additional arguments 'nrows' and 'ncols'. Once each new 'y' is computed a new variable named 'nnx' is added to the original dataframe which can be used as an extra covariate in the model.

Usage

```
nna(pheno,trait="y",rown="row",coln="col",nrows=1,ncols=2)
```

Arguments

pheno	a dataframe that contains the variables 'row', 'col', and 'y'.
trait	an additional name for the response variable in case the user wants to avoid to change the names of the dataframe.
rown	an additional name for the row variable in case the user wants to avoid to change the names of the dataframe.
coln	an additional name for the column variable in case the user wants to avoid to change the names of the dataframe.
nrows	the number of row neighbors to be used in the adjustment for the plants.
ncols	the number of column neighbors to be used in the adjustment for the plants.

Details

The function assumes unreplicated data, one measure per plot and a single experiment. If the user has a dataframe with the same data replicated in different environments or blocks this should be separated so the function do not adjust for neighbouring plots in other years, repetitions or environments.

Value

If everything is defined correctly the function returns:

$pheno the original dataframe with a new variable named 'nnx' which contains the nearest neighbor adjustment.

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

Lado et al. (2013) Increased genomic prediction accuracy in wheat breeding through spatial adjustment of field trial data. G3, 3:2105:2114

See Also

The core functions of the package mmer and mmer2

Examples

```
data(yates.oats)

head(yates.oats)
newyates <- nna(yates.oats, trait="Y")
head(newyates)

plot(newyates$Y, newyates$nnx)
cor(newyates$Y, newyates$nnx)

#### now fit the models and compare #####

m3 <- mmer2(fixed=Y ~ V*N, random = ~ B + B:MP,
            data = yates.oats)
yates.oats$res <- residuals(m3)

m4 <- mmer2(fixed=Y ~ V*N + nnx, random = ~ B + B:MP,
            data = newyates)
newyates$res <- residuals(m4)

# library(lattice)
# wireframe(res~row*col,yates.oats)
# wireframe(res~row*col,newyates)
```

NR *Newton-Raphson Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the Newton-Raphson (NR) algorithm.

Usage

```
NR(y, X=NULL, ZETA=NULL, R=NULL, draw=TRUE, REML=TRUE, silent=FALSE,
   iters=15, constraint=TRUE, init=NULL, sherman=FALSE, che=TRUE,
   EIGEND=FALSE, Fishers=FALSE, gss=TRUE, forced=NULL, identity=TRUE,
   kernel=NULL, start=NULL, taper=NULL, verbose=0, gamVals=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 matrix for variance-covariance structures for the 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
REML	a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.
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.

`sherman` a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components in order to perform faster when a mixed model with no covariance structure using the average information algorithm is fitted. The default is FALSE since this software was designed for unreplicated data (altough can fit models with replicated data but slower than lme4).

`che` 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.

`EIGEND` a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

`Fishers` a TRUE/FALSE value indicating if the program should calculate at the final step and return the inverse of the Fishers Information Matrix.

`gss` a TRUE/FALSE value indicating if a genomic selection is being fitted just for using certain constraints. When is FALSE (default) the program can make some EM steps to find initial values for variance components when the starting values are to far from the real values causing the likelihood to have a strange behavior and dropping dramatically When TRUE the program does not try EM steps even when far away from the likelihood because in big marker-based models can make the process quite slow.

`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.

`identity` Logical variable, includes the identity as the final matrix of the covariance structure. Default is TRUE

`kernel` Compute the log likelihood based on a reduced observation TY where T has this kernel. Default value of NULL assumes that the kernal matches the fixed effects model matrix X corresponding to REML. Setting kernel=0 gives the ordinary likelihood and kernel=1 gives the one dimensional subspace of constant vectors. See examples for more details.

`start`	Specify the variance components at which the Newton-Raphson algorithm starts. Default value is `rep(var(y),k)`.
`taper`	The proportion of each step to take. A vector of values from 0 to 1 of length maxcyc. Default value takes smaller steps initially.
`verbose`	Controls level of time output, takes values 0, 1 or 2, Default is 0, level 1 gives parameter estimates and value of log likelihood at each stage.
`gamVals`	When k=2, the marginal log likelihood based on the residual configuration statistic (see Tunnicliffe Wilson(1989)), is evaluated first at `(1-gam) V1 + gam V2` for each value of `gam` in `gamVals`, a set of values from the unit interval. Subsequently the Newton-Raphson algorithm is started at variance components corresponding the the value of `gam` that has the highest marginal log likelihood. This is overridden if `start` is specified.
`tolpar`	tolerance parameter for convergence in the models.
`tolparinv`	tolerance parameter for matrix inversion in the models.

Details

This algorithm is based on Tunnicliffe (1989), it is based on REML. 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 likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

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

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

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 mmer2

Examples

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

####============================================####
```

```
#### breeding values with 3 variance components
####============================================####

####============================================####
## Import phenotypic data on inbred performance
## Full data
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # extract the var-cov K

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)

####============================================####
#### Realized IBS relationships for set of parents 1
####============================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####============================================####
#### Realized IBS relationships for set of parents 2
####============================================####
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####============================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####============================================####
S <- kronecker(K1, K2) ; dim(S)
rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

ETA <- list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=S))
####============================================####
#### run the next line, it was ommited for CRAN time limitations
####============================================####
#ans <- NR(y=y, ZETA=ETA)
#ans$var.comp
```

NR22 *Newton-Raphson Algorithm*

Description

This function is used internally in the function mmer when MORE than 1 variance component needs to be estimated through the use of the Newton-Raphson (NR22) algorithm.

Usage

```
NR22(y, X=NULL, ZETA=NULL, R=NULL, draw=TRUE, REML=TRUE, silent=FALSE,
```

```
iters=15, constraint=TRUE, init=NULL, sherman=FALSE, che=TRUE,
EIGEND=FALSE, Fishers=FALSE, gss=TRUE, forced=NULL, identity=TRUE,
kernel=NULL, start=NULL, taper=NULL, verbose=0, gamVals=NULL,
maxcyc=15, tol=1e-4)
```

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 matrix for variance-covariance structures for the 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
REML	a TRUE/FALSE value indicating if restricted maximum likelihood should be used instead of ML. The default is TRUE.
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.
sherman	a TRUE/FALSE value indicating if Sherman-Morrison-Woodbury formula (Seber, 2003, p. 467) should be used when estimating variance components in order to perform faster when a mixed model with no covariance structure using the average information algorithm is fitted. The default is FALSE since this software was designed for unreplicated data (altough can fit models with replicated data but slower than lme4).

che — 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.

EIGEND — a TRUE/FALSE value indicating if an eigen decomposition for the additive relationship matrix should be performed or not. This is based on Lee (2015). The limitations of this methos are: 1) can only be applied to one relationship matrix 2) The system needs to be squared and no missing data is allowed (then missing data is imputed with the median). The default is FALSE to avoid the user get into trouble but experimented users can take advantage from this feature to fit big models, i.e. 5000 individuals in 555 seconds = 9 minutes in a MacBook 4GB RAM.

Fishers — a TRUE/FALSE value indicating if the program should calculate at the final step and return the inverse of the Fishers Information Matrix.

gss — a TRUE/FALSE value indicating if a genomic selection is being fitted just for using certain constraints. When is FALSE (default) the program can make some EM steps to find initial values for variance components when the starting values are to far from the real values causing the likelihood to have a strange behavior and dropping dramatically When TRUE the program does not try EM steps even when far away from the likelihood because in big marker-based models can make the process quite slow.

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.

identity — Logical variable, includes the identity as the final matrix of the covariance structure. Default is TRUE

kernel — Compute the log likelihood based on a reduced observation TY where T has this kernel. Default value of NULL assumes that the kernal matches the fixed effects model matrix X corresponding to REML. Setting kernel=0 gives the ordinary likelihood and kernel=1 gives the one dimensional subspace of constant vectors. See examples for more details.

start — Specify the variance components at which the Newton-Raphson algorithm starts. Default value is `rep(var(y),k)`.

taper — The proportion of each step to take. A vector of values from 0 to 1 of length maxcyc. Default value takes smaller steps initially.

verbose — Controls level of time output, takes values 0, 1 or 2, Default is 0, level 1 gives parameter estimates and value of log likelihood at each stage.

gamVals — When k=2, the marginal log likelihood based on the residual configuration statistic (see Tunnicliffe Wilson(1989)), is evaluated first at `(1-gam) V1 + gam V2` for each value of `gam` in `gamVals`, a set of values from the unit interval. Subsequently the Newton-Raphson algorithm is started at variance components corresponding the the value of `gam` that has the highest marginal log likelihood. This is overridden if `start` is specified.

maxcyc — Maximum number of cycles allowed. Default value is 50. A warning is output to the screen if this is reached before convergence.

`tol` Convergence criteria. If the change in residual log likelihood for one cycle is less than `10 x tol` the algorithm finishes. If each component of the change proposed by the Newton-Raphson is lower in magnitude than `tol` the algorithm finishes. Default value is `1e-4`.

Details

This algorithm is based on Tunnicliffe (1989), it is based on REML. 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 likelihood function optimized in this algorithm is:

.

logL = -0.5 * (log(| V |) + log(| X'VX |) + y'Py

.

where: | | refers to the derminant of a matrix

.

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

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

See Also

The core functions of the package mmer and mmer2

Examples

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

####============================================####
#### breeding values with 3 variance components
####============================================####

####============================================####
## Import phenotypic data on inbred performance
## Full data
####============================================####
data(cornHybrid)
hybrid2 <- cornHybrid$hybrid # extract cross data
A <- cornHybrid$K # extract the var-cov K

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)

####==========================================####
#### Realized IBS relationships for set of parents 1
####==========================================####
K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
####==========================================####
#### Realized IBS relationships for set of parents 2
####==========================================####
K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
####==========================================####
#### Realized IBS relationships for cross
#### (as the Kronecker product of K1 and K2)
####==========================================####
S <- kronecker(K1, K2) ; dim(S)
rownames(S) <- colnames(S) <- levels(hybrid2$SCA)

ETA <- list(list(Z=Z1, K=K1), list(Z=Z2, K=K2), list(Z=Z3, K=S))
####==========================================####
#### run the next line, it was ommited for CRAN time limitations
####==========================================####
#ans <- NR22(y=y, ZETA=ETA)
#ans$var.comp
```

`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(dat, rlist=NULL, prefix=NULL)
```

Arguments

`dat`	a dataframe with as many columns 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.

Value

$S3 an incidence matrix with as many columns as parents in the dataframe indicating with ones the parents used for a particular hybrid (in rows).

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 mmer2

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(HDdata)
head(HDdata)
####============================================####
#### GCA matrix for half diallel using male and female columns
#### use the 'overlay' function to create the half diallel matrix
####============================================####
Z1 <- overlay(HDdata[,c(3:4)])
####============================================####
#### Obtain the SCA matrix
####============================================####
Z2 <- model.matrix(~as.factor(geno)-1, data=HDdata)
####============================================####
#### Define the response variable and run
####============================================####
y <- HDdata$sugar
ETA <- list(list(Z=Z1), list(Z=Z2)) # Zu component
modHD <- mmer(Y=y, Z=ETA, draw=FALSE, silent=TRUE)
summary(modHD)

####============================================####
#### Example 2
#### using overlay with mmer2 function
####============================================####
# data(HDdata)
```

```
# head(HDdata)
# HDdata$female <- as.factor(HDdata$female)
# HDdata$male <- as.factor(HDdata$male)
# HDdata$geno <- as.factor(HDdata$geno)
# #### model using overlay
# modh <- mmer2(sugar~1, random=~female + and(male) + geno,
#               data=HDdata)
# summary(modh)
# #### model using overlay [and(.)] and covariance structures [g(.)]
# A <- diag(7); A[1,2] <- 0.5; A[2,1] <- 0.5 # fake covariance structure
# colnames(A) <- as.character(1:7); rownames(A) <- colnames(A);A
#
# modh2 <- mmer2(sugar~1, random=~g(female) + and(g(male)) + geno,
#               G=list(female=A, male=A),data=HDdata)
# summary(modh2)
```

PEV *Selecting the best training population for genomic selection*

Description

This function is a wrapper from the STPGA package to obtain the best subset of individuals to predict a group of individuals minimizing the predicted error variance (PEV). Is used internally in the TP.prep function.

Usage

```
PEV(PCAs,candidates,Test,ntoselect, npop, nelite, mutprob, niterations, lambda)
```

Arguments

PCAs	nxk PCA matrix from the predictor variables.
candidates	vector of names for the population without the test set.
Test	name of the individuals in the test or VP set.
ntoselect	number of individuals to select in the training population.
npop	number of solutions at each iteration.
nelite	number of elite solutions for TP pops.
mutprob	probability of mutation for each solution
niterations	number of iterations
lambda	scalar shrinkage in PEV.

Value

If all parameters are correctly indicated the program will return:

$tp.list a list with the names of the TP populations. Each element of the list corresponds to each population size specified in the 'tp.size' argument.

Author(s)

Giovanny Covarrubias-Pazaran

References

Akdemir, Deniz. "Training population selection for breeding value" prediction. 2014.

See Also

The core functions of the package mmer and mmer2

`phase.F1` *Phasing F1 (CP) data in biparental populations*

Description

This function was designed to provide a useful tool for building dense genetic maps in F1 (sometimes referred as CP populations) crosses. This tool alone doesn't build genetic maps but is an important intermediate step researchers will find difficult to deal with.

It takes CP data in a preliminary order (produced by JoinMap4.1 or OneMap) and produces parental maps phased including hk markers translated to parental format without missing the information from hk markers (JoinMap usually looses the information from hk alleles from hk markers, so much missing data makes difficult the bin mapping).

Ideally mapping in F1s would include the following steps:

1) Do initial map with any algorithm able to deal with CP crosses (i.e. JoinMap v4.1 ML algorithm can do maps with all markers lmxll, nnxnp, hkxhk, efxeg, abxcd simultaneously)

2) Develop parental maps (this is the step this function performs, including phasing and imputation of hk markers)

3) Do bin mapping in the parental maps (ASMap in R is a good choice for doing the bin maps)

4) Do an integrated map using the parental bin maps (LP merge is a good option)

Usage

```
phase.F1(genos, silent=FALSE, start=7)
```

Arguments

`genos`	a dataframe with joinmap format as shown in the `F1geno` data which is explained with more detail in the example below.
`silent`	a TRUE/FALSE statement indicating if the program should print the progrss bar as the function works.
`start`	the marker where the program should start to look for the first hk marker.

Value

$ff a list with 6 data frames, consensus and parental maps in joinmap and Rqtl format.

Author(s)

Brandon Schlautman

See Also

The core functions of the package mmer and mmer2

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####=========================================####
data(F1geno)
F1geno[1:10,1:10]
#maps <- phase.F1(F1geno)
#maps$maternal[1:10,1:10]
#heat <- apply(maps$maternal[,-c(1:6)],2,function(x){as.numeric(as.factor(x))})
#heat[1:10,1:10]
#image(heat)
####=========================================####
#### use variance to remove bad markers
####=========================================####
#plot(apply(heat,1,var))
#jjj <- which(apply(heat,1,var) > .5)
#image(heat[-jjj,])
####=========================================####
#### do bin mapping using ASMap
####=========================================####
```

`pin` *pin functionality*

Description

Post-analysis procedure to calculate functions 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 pin file specifies the functions to be calculated.

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.

Usage

```
pin(object, transform)
```

Arguments

object	a model fitted with the mmer or mmer2 functions.
transform	formula to calculate the function.

Value

$dd the parameter and its standard error.

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(CPdata)
CPpheno <- CPdata$pheno
CPgeno <- CPdata$geno
#### create the variance-covariance matrix
A <- A.mat(CPgeno)
#### look at the data and fit the model
head(CPpheno)
mix1 <- mmer2(Yield~1,random=~g(id), G=list(id=A), data=CPpheno)
summary(mix1)
#### run the pin function
pin(mix1, h2 ~ V1 / ( V1 + V2 ) )

# ####============================================####
# ####============================================####
# #### EXAMPLE 2
# #### simple example with multivariate models
# ####============================================####
```

```
# ####========================================####
# data(CPdata)
# CPpheno <- CPdata$pheno
# CPgeno <- CPdata$geno
# #### create the variance-covariance matrix
# A <- A.mat(CPgeno)
# #### look at the data and fit the model
# head(CPpheno)
# mix2 <- mmer2(cbind(Yield,color)~1,random=~g(id), G=list(id=A),
#               data=CPpheno, MVM=TRUE)
# summary(mix2)
# ## genetic correlation
# pin(mix2, gen.cor ~ V2 / sqrt(V1*V3))
#
# ####========================================####
# ####========================================####
# #### EXAMPLE 3
# #### more complex multivariate model
# ####========================================####
# ####========================================####
# data(BTdata)
# mix3 <- mmer2(cbind(tarsus, back) ~ sex,
#                random = ~ dam + fosternest,
#                data = BTdata, MVM=TRUE)
# summary(mix3)
# #### calculate the genetic correlation
# pin(mix3, gen.cor ~ V2 / sqrt(V1*V3))
#
# ####========================================####
# ####========================================####
# #### EXAMPLE 4
# #### going back to simple examples
# ####========================================####
# ####========================================####
# data(BTdata)
# mix4 <- mmer2(tarsus ~ sex, random = ~ dam + fosternest,
#                data = BTdata)
# summary(mix4)
# #### calculate the ratio and its SE
# pin(mix4, dam.prop ~ V1 / ( V1 + V2 + V3 ) )
```

`plot.mmer` *plot form a GLMM plot with mmer*

Description

`plot` method for class `"mmer"`.

Usage

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

Arguments

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

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

plot, mmer and mmer2

---

plot.MMERM *plot form a GLMM plot with mmer*

Description

plot method for class "MMERM".

Usage

```
## S3 method for class 'MMERM'
plot(x, ...)
```

Arguments

x	an object of class "MMERM"
...	Further arguments to be passed

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

plot, mmer

plot.mmerM *plot form a GLMM plot with mmer*

Description

plot method for class "mmerM".

Usage

```
## S3 method for class 'mmerM'
plot(x, ...)
```

Arguments

x	an object of class "mmerM"
...	Further arguments to be passed

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

plot, mmer

poe *Short poems from Latin America, and other places why not?*

Description

This function was designed for fun to remember the scientist (and me) that life is more than analyzing data. If you want to contribute with a poem you like for the following versions of sommer sent it to me to:

covarrubiasp@wisc.edu

Usage

```
poe(h)
```

Arguments

h	a numeric value between 1 to 8 to select the poem. In the function a random sample is taken

Value

$poe.value a poem to remember the scientist (and me) that life is more than analyzing data.

Author(s)

Giovanny Covarrubias-Pazaran

References

Many

See Also

The core functions of the package mmer and mmer2

Examples

```
poe(sample(1:8,1))
```

PolyData *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 cite Rosyara's (2015) publication (see References).

Usage

```
data("PolyData")
```

Format

The format is: chr "PolyData"

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 and mmer2

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(PolyData)
genotypes <- PolyData$PGeno
phenotypes <- PolyData$PPheno

# ####============================================####
# ####### convert markers to numeric format
# ####============================================####
# numo <- atcg1234(data=genotypes, ploidy=4); numo[1:5,1:5]; dim(numo)
#
# ###============================================####
# ###### plants with both genotypes and phenotypes
# ###============================================####
# common <- intersect(phenotypes$Name,rownames(numo))
#
# ###============================================####
# ### get the markers and phenotypes for such inds
# ###============================================####
# marks <- numo[common,]; marks[1:5,1:5]
# phenotypes2 <- phenotypes[match(common,phenotypes$Name),];
# phenotypes2[1:5,1:5]
#
# ###============================================####
# ### response variable
# ###============================================####
# yy.trn <- phenotypes2
# set.seed(1234)
# ww <- sample(1:187,38)
# yy.trn[ww,"tuber_shape"] <- NA
#
# ###============================================####
# ###### Additive relationship matrix, specify ploidy
```

```
# ###=========================================####
# A <- A.mat(marks, ploidy=4); dim(K1);K1[1:5,1:5]
# D <- D.mat(marks, ploidy=4)
# E <- E.mat(marks, ploidy=4)
# ###=========================================####
# ### run the genomic selection model
# ###=========================================####
# ans <- mmer2(tuber_shape~1, random=~g(Name),
#              G=list(Name=A), data=yy.trn)
# cor(phenotypes2[ww,"tuber_shape"],ans$fitted.y[ww])
# summary(ans)
#
# ###=========================================####
# ### run it as GWAS model
# ###=========================================####
# my.map <- PolyData$map
# models <- c("additive","1-dom-alt","1-dom-ref","2-dom-alt","2-dom-ref")
# ans2 <- mmer2(tuber_shape~1, random=~g(Name), models = "additive",
#              G=list(Name=A), W=marks, data=phenotypes2)
# summary(ans2)
#
# ###=========================================####
# ### compare to GWAS including dominance
# ###=========================================####
# phenotypes2$Named <- phenotypes2$Name
# ans3 <- mmer2(tuber_shape~1, random=~g(Name) + g(Named), models = "additive",
#               G=list(Name=A, Named=D), W=marks, data=phenotypes2)
# summary(ans3)
```

---

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)
```

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

Description

residuals method for class "mmer".

Usage

```
## S3 method for class 'mmer'
residuals(object, type = "conditional", ...)
```

Arguments

object	an object of class "mmer"
type	the type of residuals which should be returned. The alternatives are: "conditional" (e=y-(Xb+Zu)) and "regular" (e=y-Xb).
...	Further arguments to be passed

Value

vector of residuals

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

residuals, mmer, and mmer2

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

Description

residuals method for class "MMERM".

Usage

```
## S3 method for class 'MMERM'
residuals(object, type = "conditional", ...)
```

Arguments

object	an object of class "MMERM"
type	the type of residuals which should be returned. The alternatives are: "conditional" (e=y-(Xb+Zu)) and "regular" (e=y-Xb).
...	Further arguments to be passed

Value

vector of residuals

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

residuals, mmer

---

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

Description

residuals method for class "mmerM".

Usage

```
## S3 method for class 'mmerM'
residuals(object, type = "conditional", ...)
```

Arguments

object	an object of class "mmerM"
type	the type of residuals which should be returned. The alternatives are: "conditional" (e=y-(Xb+Zu)) and "regular" (e=y-Xb).
...	Further arguments to be passed

Value

vector of residuals

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

residuals, mmer

RICE *Rice lines dataset*

Description

Information from a collection of 413 rice lines. The 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 illustrate the selection of training populations for obtainin maximum prediction accuracies. We have found that the best way to select a training population is either 1) taking a random sample from the entire population, 2) do PCA and selecting very similar and very different genotypes (quite similar to random sampling), or 3) performing a GWAS with the data available to identify regions of significance and do PCA but using only markers that are in significant regions, (selecting from plants close to the VP and also far, to make sure we sample from different haplotypes for the QTLs). All versions are shown in the examples with rice data using the `TP.prep` function.

Usage

```
data(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` and `mmer2`

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(RICE)
W <- RICE$RiceGenoN; W[1:5,1:5]; dim(W)
Y <- RICE$RicePheno; Y[1:5,1:5]; dim(Y)
#(vp.str <- RICE$vp.str[1:25]) # validation population with structure
#bases <- seq(50,200,25) # TP sizes required

####=========================================####
#### a) Random sample based
####=========================================####
#rest <- (1:dim(W)[1])[-which(rownames(W) %in% vp.str)]
#tp.plant.strR <- TP.prep(markers=W, vp.names = vp.str,
#                         tp.size = bases, method="random")

####=========================================####
#### b) Genetic simmilarity-dissimilarity based
####=========================================####
#tp.plant.strG1 <- TP.prep(markers=W, vp.names = vp.str,
#                         tp.size = bases, method="sim-dissim")

#tp.plant.strG2 <- TP.prep(markers=W, vp.names = vp.str,
#                         tp.size = bases, method="sim")

####=========================================####
#### c) QTL based
#### we assume real situation that VP
#### wouldn't be phenotyped for GWAS
####=========================================####
#K1 <- A.mat(W)
#Z1 <- diag(dim(K1)[1])
#image(K1)
#ETA1 <- list(list(Z=Z1,K=K1))
#YY <- Y; YY[vp.str,] <- NA
#ans.GWAS <- mmer(Y=YY$Protein.content, Z=ETA1, W=W)
#(h2 <- sqrt(ans.GWAS$var.comp[1,]/sum(ans.GWAS$var.comp)))
#X <- bag(ans.GWAS, nmar = 25); head(X); dim(X)

#tp.plant.strG3 <- TP.prep(markers=X, vp.names = vp.str,
#                        tp.size = bases, method="sim-dissim")

####=========================================####
#### COMPARE METHODS with 50 plants
####=========================================####
#metho <- list(a=tp.plant.strR[[1]], b=tp.plant.strG1[[1]],
#      c=tp.plant.strG2[[1]], d=tp.plant.strG3[[1]])

#resos <- numeric()
#for(i in 1:4){ # for each method

#  tpgi <- metho[[i]] # training pop genetic-based
#  Wpgi <- W[c(vp.str,tpgi),] # provisional markers based on genes
#  Ypgi <- Y[c(vp.str,tpgi),] # provisional phenos based on genes
```

```
#  Xpgi <- X[c(vp.str,tpgi),] # provisional bag-mat based on genes

#  Kpgi <- A.mat(Wpgi)
#  Zpgi <- diag(dim(Kpgi)[1])
#  Ypgi[vp.str,] <- NA
#  yt <- Ypgi$Protein.content
#  ETAg <- list(list(Z=Zpgi, K=Kpgi))
#  mixg <- mmer(y=yt, X=Xpgi, Z=ETAg, method="EMMA") #X=Xpgi,
#  resos[i] <- cor(Y[vp.str,"Protein.content"],
#                  fitted(mixg)[(1:length(vp.str)),],
#                  use="complete")
#}
#resos
### the random method needs more iterations to have a real idea what
### would be the predictive ability of using plants at random,
### this is just to give an idea to users.
```

`score.calc` *Score calculation for markers*

Description

This function is a wrapper from the rrBLUP package to be used when a mixed model including markers to perform GWAS is specified and once the variance components have been estimated the fixed effects are obtained as B= (X'V-X)-X'V-y and the score calculation is obtained with the F statistic as F = Beta^2 / Var(Beta) where Var(Beta) = SSe/(n-p) * [XH-X']-, and quantile value for the beta distribution is calculated as q = (n-p) / (n-p + 1 * F) which once obtained, the -log10 for such value is the score value.

Usage

```
score.calc(marks,y,Z,X,K,ZZ,M,Hinv,ploidy,model,min.MAF,
           max.geno.freq,silent=FALSE,P3D=TRUE, method="NR")
```

Arguments

`marks`	marker names
`y`	response variable
`Z`	incidence matrix of random effects
`X`	incidence matrix X as full rank from eigen decomposition
`K`	covariance structure for random effects
`ZZ`	incidence matrix of random effects
`M`	marker matrix
`Hinv`	inverse of the phenotypic variance matrix
`ploidy`	numeric value of ploidy level, i.e. 2
`model`	model for GWAS

min.MAF	minimum minor allele frequency
max.geno.freq	1 - min.MAF
silent	a TRUE/FALSE value indicating if the progress bar should be drawn or not
P3D	when the user performs GWAS, P3D=TRUE means that the variance components are estimated by REML only once, without any markers in the model. When P3D=FALSE, variance components are estimated by REML for each marker separately. The default is the first case.
method	one of the four methods to estimate variance components.

Value

$score a vector with the -log10(p-values) for the marker effects in the trait under study

See Also

The core functions of the package mmer and mmer2

Examples

```
# it works internally in the \code{\link{mmer}} function
```

---

score.calcMV *Score calculation for markers*

Description

This function is a wrapper from the rrBLUP package to be used when a mixed model including markers to perform GWAS is specified and once the variance components have been estimated the fixed effects are obtained as B= (X'V-X)-X'V-y and the score calculation is obtained with the F statistic as F = Beta^2 / Var(Beta) where Var(Beta) = SSe/(n-p) * [XH-X']-, and quantile value for the beta distribution is calculated as q = (n-p) / (n-p + 1 * F) which once obtained, the -log10 for such value is the score value.

Usage

```
score.calcMV(marks,Y,Z,X,K,ZZ,M,Hinv,ploidy,model,min.MAF,
          max.geno.freq,silent=FALSE,P3D=TRUE)
```

Arguments

marks	marker names
Y	response variable
Z	incidence matrix of random effects
X	incidence matrix X as full rank from eigen decomposition
K	covariance structure for random effects

ZZ	incidence matrix of random effects
M	marker matrix
Hinv	inverse of the phenotypic variance matrix
ploidy	numeric value of ploidy level, i.e. 2
model	model for GWAS
min.MAF	minimum minor allele frequency
max.geno.freq	1 - min.MAF
silent	a TRUE/FALSE value indicating if the progress bar should be drawn or not
P3D	when the user performs GWAS, P3D=TRUE means that the variance components are estimated by REML only once, without any markers in the model. When P3D=FALSE, variance components are estimated by REML for each marker separately. The default is the first case.

Value

$score a vector with the -log10(p-values) for the marker effects in the trait under study

See Also

The core functions of the package mmer and mmer2

Examples

```
# it works internally in the \code{\link{mmer}} function
```

---

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 <covarrubiasp@wisc.edu>

See Also

summary, mmer and mmer2

---

summary.MMERM *summary form a GLMM fitted with mmer*

Description

summary method for class "MMERM".

Usage

```
## S3 method for class 'MMERM'
summary(object, ...)
```

Arguments

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

Value

vector of summary

Author(s)

Giovanny Covarrubias-Pazaran <covarrubiasp@wisc.edu>

See Also

summary, mmer

summary.mmerM *summary form a GLMM fitted with mmer*

Description

summary method for class "mmerM".

Usage

```
## S3 method for class 'mmerM'
summary(object, ...)
```

Arguments

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

Value

vector of summary

Author(s)

Giovanny Covarrubias-Pazaran <covarrubiasp@wisc.edu>

See Also

summary, mmer

Technow_data *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("Technow_data")
```

Format

The format is: chr "Technow_data"

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 and mmer2

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(Technow_data)

A.flint <- Technow_data$AF # Additive relationship matrix Flint
A.dent <- Technow_data$AD # Additive relationship matrix Dent
M.flint <- Technow_data$MF # Marker matrix Flint
M.dent <- Technow_data$MD # Marker matrix Dent

pheno <- Technow_data$pheno # phenotypes for 1254 single cross hybrids
pheno$hy <- paste(pheno$dent, pheno$flint, sep=":");head(pheno);dim(pheno)

####============================================####
#### CREATE A DATA FRAME WITH ALL POSSIBLE HYBRIDS
####============================================####
# DD <- kronecker(A.dent,A.flint,make.dimnames=TRUE)
# hybs <- data.frame(sca=rownames(DD),yield=NA,matter=NA,gcad=NA, gcaf=NA)
# hybs$yield[match(pheno$hy, hybs$sca)] <- pheno$GY
# hybs$matter[match(pheno$hy, hybs$sca)] <- pheno$GM
# hybs$gcad <- as.factor(gsub(":.*","",hybs$sca))
# hybs$gcaf <- as.factor(gsub(".*:","",hybs$sca))
# head(hybs)
####============================================####
## RUN THE PREDICTION MODEL
####============================================####
# y.trn <- hybs
```

```
# vv1 <- which(!is.na(hybs$yield))
# vv2 <- sample(vv1, 100)
# y.trn[vv2,"yield"] <- NA
# anss2 <- mmer2(yield~1, random=~g(gcad) + g(gcaf), G=list(gcad=A.dent, gcaf=A.flint),
#                method="EM", data=y.trn)
# summary(anss2)
# cor(anss2$fitted.y[vv2], hybs$yield[vv2])

#### try EM algorithm as well if only 2 variance components
#### you can try adding SCA effects by adding SCA=list(Z=Z3, K=DD)
#### to the random effects but will slow down the model a lot


##########################################
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################
##########################################

####========================================####
####========================================####
#### EXAMPLE 2
#### GWAS in single cross hybrids
####========================================####
####========================================####
# data(Technow_data)
# pheno <- Technow_data$pheno # phenotypes for 1254 single cross hybrids
# pheno$hy <- paste(pheno$dent, pheno$flint, sep=":");head(pheno);dim(pheno)
# pheno$dent <- as.factor(pheno$dent)
# pheno$flint <- as.factor(pheno$flint)
# ###========================================####
# ### RUN GWAS combining markers
# ###========================================####
# M.dent <- Technow_data$MD # Marker matrix Dent
# M.flint <- Technow_data$MF # Marker matrix Flint
# Mall <- rbind(M.dent,M.flint); dim(Mall)
# Adf <- A.mat(Mall) # Additive relationship matrix
# # marker matrix for GWAS is the cbind of dent and flint,
# # parental genomes analyzed by separate but in same framework
# MGF <- cbind(M.dent[pheno$dent,],M.flint[pheno$flint,]); dim(MGF)
#
# mox <- mmer2(GY~1, random=~g(dent) + and(g(flint)),
#              G=list(dent=Adf, flint=Adf), method="EM",
#              W=MGF, data=pheno)
# summary(mox)
```

TP.prep *Selecting the best training population for genomic selection*

Description

This function was designed to help users build the best training population for real genomic selection applications. The RICE dataset from Zhao et al. (2011) has been incorporated to the package to highlight this new feature of sommer implemented in v.1.5. The TP.prep function is just a function that uses genetic markers or any kind of numeric matrix to perform principal component analysis, identify the names of the validation population the user provide, and find the most suitable training population for performing genomic selection based on similarities or similarities-dissimilarities.

My recomendation would be to randomly choose the TP and predict the performance of the VP. Do this a 100 times and get the average GEBV for the VP and take decisions using those.

Usage

```
TP.prep(markers=NULL, vp.names=NULL, tp.size=NULL, method="random",
        npop=300, nelite=1, mutprob=.5, niterations=100, lambda=NULL)
```

Arguments

markers	a marker matrix in numeric format but any numeric matrix can be used in this argument. No extra columns should be in this matrix. ROWNAMES SHOOULD BE THE NAMES OF THE INDIVIDUALSIN THE POPULATION.
vp.names	a character vector with the names of the validaton population. Make sure that the marker matrix has rownames with the individuals so the program can find them and identify the most suitable training population.
tp.size	a numeric vector indicating the population sizes desired for the training population. By default is NULL which will make the program select in steps of 20 until reach the population size (number of rows in the marker matrix).
method	a method among "similar", "sim-dissim", "random", "PEV", or "CDmean" indicating if the program should look for similar plants (based on the PCA of the marker matrix) or a mixture of similar and constrasting individuals, select plants at random, minimize the PEV, or calculate the CD mean. The default is "sim-dissim", meaning will make a mixture.
npop	genetic algorithm parameter, number of solutions at each iteration. Only for the CDmean algorithm.
nelite	genetic algorithm parameter, number of solutions selected as elite parents which will generate the next set of solutions. Only for the CDmean algorithm.
mutprob	genetic algorithm parameter, probability of mutation for each generated solution. Only for the CDmean algorithm.
niterations	genetic algorithm parameter, number of iterations.Only for the CDmean algorithm.
lambda	a value for the CDmean algorithm where lambda=Var(e)/Var(g).

Value

If all parameters are correctly indicated the program will return:

$tp.list a list with the names of the TP populations. Each element of the list corresponds to each population size specified in the 'tp.size' argument.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

See Also

The core functions of the package mmer and mmer2

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(RICE)
W <- RICE$RiceGenoN; W[1:5,1:5]; dim(W)
#### lets use only 5000 random markers across the genome
#### for demonstration purposes
ww <- sample(1:dim(W)[2],5000)
W <- W[,ww]; dim(W)

Y <- RICE$RicePheno; Y[1:5,1:5]; dim(Y)
(vp.str <- RICE$vp.str[1:25]) # validation population with structure
bases <- seq(50,200,25) # TP sizes required

####============================================####
####============================================####
#### a) Random sample based
####============================================####
####============================================####
#### rest of the plants not in the validation population (VP)
rest <- (1:dim(W)[1])[-which(rownames(W) %in% vp.str)]
#### select the best TP for the VP, provide:
```

```
#### markers, names of VP, size of the TP desired, method
#tp.plant.strR <- TP.prep(markers=W, vp.names = vp.str,
#                         tp.size = bases, method="random")

####=========================================####
#### b) Genetic simmilarity-dissimilarity based
####=========================================####
#tp.plant.strG1 <- TP.prep(markers=W, vp.names = vp.str,
#                         tp.size = bases, method="sim-dissim")

#tp.plant.strG2 <- TP.prep(markers=W, vp.names = vp.str,
#                         tp.size = bases, method="sim")

####=========================================####
####=========================================####
#### c) QTL based
#### we assume real situation that VP
#### wouldn't be phenotyped for GWAS
####=========================================####
####=========================================####
#K1 <- A.mat(W)
#Z1 <- diag(dim(K1)[1])
#image(K1)
#ETA1 <- list(list(Z=Z1,K=K1))
#YY <- Y; YY[vp.str,] <- NA
####=========================================####
#### be patient, the GWAS is for >36,000 SNPs
####=========================================####
#ans.GWAS <- mmer(Y=YY$Protein.content, Z=ETA1, W=W)
#(h2 <- sqrt(ans.GWAS$var.comp[1,]/sum(ans.GWAS$var.comp)))
####=========================================####
#### get marker matrix for the most significant QTLs
#### and do the TP selection using only those markers
####=========================================####
#X <- hits(ans.GWAS, nmar = 25); head(X); dim(X)
#tp.plant.strG3 <- TP.prep(markers=X, vp.names = vp.str,
#                      tp.size = bases, method="sim-dissim")

####=========================================####
####=========================================####
#### COMPARE METHODS with 50 plants
####=========================================####
####=========================================####
#metho <- list(a=tp.plant.strR[[1]], b=tp.plant.strG1[[1]],
#      c=tp.plant.strG2[[1]], d=tp.plant.strG3[[1]])

#resos <- numeric()
#for(i in 1:4){ # for each method

#  tpgi <- metho[[i]] # training pop genetic-based
#  Wpgi <- W[c(vp.str,tpgi),] # provisional markers based on genes
#  Ypgi <- Y[c(vp.str,tpgi),] # provisional phenos based on genes
#  Xpgi <- X[c(vp.str,tpgi),] # provisional bag-mat based on genes
```

```
#  Kpgi <- A.mat(Wpgi)
#  Zpgi <- diag(dim(Kpgi)[1])
#  Ypgi[vp.str,] <- NA
#  yt <- Ypgi$Protein.content
#  ETAg <- list(list(Z=Zpgi, K=Kpgi))
#  mixg <- mmer(Y=yt, X=Xpgi, Z=ETAg, method="EMMA") #X=Xpgi,
#  resos[i] <- cor(Y[vp.str,"Protein.content"],
#                  fitted(mixg)[(1:length(vp.str)),],
#                  use="complete")
#}
#resos
### the random method needs more iterations to have a real idea what
### would be the predictive ability of using plants at random,
### this is just to give an idea to users.
```

`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 and mmer2

Examples

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

---

us *us functionality*

Description

This function creates internally the incidence matrices for an unstructure model when using mmer2.

Usage

```
us(x)
```

Arguments

x	a column of a dusaframe.

Value

$dd an incidence musrix with columns for the specific levels specified.

Author(s)

Giovanny Covarrubias-Pazaran

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitusive 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 mmer2

`wheatLines` *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(wheatLines)
```

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 and mmer2

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(wheatLines)
X <- wheatLines$wheatGeno; X[1:5,1:5]; dim(X)
Y <- wheatLines$wheatPheno
rownames(X) <- rownames(Y)

####============================================####
#### select environment 1
####============================================####
#y <- Y[,1] # response grain yield
#Z1 <- diag(length(y)) # incidence matrix
#K <- A.mat(X) # additive relationship matrix

####============================================####
#### GBLUP pedigree-based approach
####============================================####
#ETA <- list(line=list(Z=Z1, K=K))
#ans <- mmer(Y=y, Z=ETA, method="EMMA") # kinship based
#summary(ans)

####============================================####
#### GBLUP marker based approach
####============================================####
#ETA2 <- list(mar=list(Z=X))
#ans2 <- mmer(Y=y, Z=ETA2, method="EMMA") # marker based
#summary(ans2)

####============================================####
#### compare and check that is the same result
####============================================####
#plot(ans$u.hat$line, (X%*%ans2$u.hat$mar), xaxt="n", yaxt="n",
#     ylab="Marker-based GBLUP", xlab="Pedigree-based GBLUP")

####============================================####
#### PREDICT PROGENY
####============================================####
#GEBV.pb <- ans$u.hat$line # this are the BV
#rownames(GEBV.pb) <- rownames(Y)

####============================================####
#### all possible crosses = 179,101
####============================================####
```

```
#crosses <- do.call(expand.grid, list(rownames(Y),rownames(Y))); dim(crosses)
#cross2 <- duplicated(t(apply(crosses, 1, sort)))
#crosses2 <- crosses[cross2,]; head(crosses2); dim(crosses2)

####=========================================####
#### match the possible crosses with the parental BV
####=========================================####
#GCA1 = GEBV.pb[match(crosses2[,1], rownames(GEBV.pb))] # get GCA1 BLUP of each hybrid
#GCA2 = GEBV.pb[match(crosses2[,2], rownames(GEBV.pb))] # get GCA1 BLUP of each hybrid

####=========================================####
#### join everything
####=========================================####
#BV <- data.frame(crosses2,GCA1,GCA2); head(BV)
#BV$BVcross <- apply(BV[,c(3:4)],1,mean); head(BV)
#plot(BV$BVcross)
```

`yates.oats` *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

```
### ========================== ###
### using the mmer2 function
### ========================== ###

data(yates.oats)
head(yates.oats)
m3 <- mmer2(fixed=Y ~ V*N, random = ~ B + B:MP,
            data = yates.oats)
summary(m3)
m3$var.comp

### ========================== ###
### using the mmer function
### ========================== ###

###response
y<-yates.oats$Y
###fixed effects
X1 <- model.matrix(~V*N,yates.oats)
###random effects
Z1 <- model.matrix(~B-1, yates.oats)
Z2 <- model.matrix(~B:MP-1, yates.oats)
ETA <- list(B=list(Z=Z1),BMP=list(Z=Z2))
###run the model
m4 <- mmer(Y=y,X=X1,Z=ETA)
summary(m4)
```

Index