Package 'sommer'

June 16, 2020

Type Package

Title Solving Mixed Model Equations in R

Version 4.1.0

Date 2020-06-01

Author Giovanny Covarrubias-Pazaran

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

Description Structural multivariate-univariate linear mixed model solver for estimation of multiple random effects and unknown variance-covariance structures (i.e. heterogeneous and unstructured variance models) (Covarrubias-Pazaran, 2016 <doi:10.1371/journal.pone.0156744>; Maier et al., 2015 <doi:10.1016/j.ajhg.2014.12.006>). ML/REML estimates can be obtained using the Direct-Inversion Newton-Raphson and Direct-Inversion Average Information algorithms. Designed for genomic prediction and genome wide association studies (GWAS), particularly focused in the p > n problem (more coefficients than observations) and dense known covariance structures for levels of random effects. Spatial models can also be fitted using i.e. the two-dimensional spline functionality available in sommer.

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

License GPL (>= 2)

Imports Rcpp (>= 0.12.19)

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, plyr, parallel, orthopolynom

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-06-16 16:30:03 UTC

R topics documented:

sommer-package 3
A.mat 9
adiag1 11

anova.mmer . . . 13
AR1 . . . 14
ARMA . . . 15
at . . . 16
atcg1234 . . . 17
bathy.colors . . . 19
bivariateRun . . . 19
build.HMM . . . 21
coef.mmer . . . 23
CS . . . 23
cs . . . 24
D.mat . . . 25
ds . . . 27
DT_augment . . . 28
DT_btdata . . . 29
DT_cornhybrids . . . 30
DT_cpdata . . . 32
DT_example . . . 34
DT_expdesigns . . . 36
DT_fulldiallel . . . 38
DT_gryphon . . . 39
DT_h2 . . . 40
DT_halfdiallel . . . 41
DT_legendre . . . 43
DT_polyploid . . . 44
DT_rice . . . 45
DT_technow . . . 47
DT_wheat . . . 48
DT_yatesoats . . . 50
E.mat . . . 51
EM . . . 53
fcm . . . 56
fill.design . . . 58
fitted.mmer . . . 59
fixm . . . 60
GWAS . . . 61
GWAS2 . . . 68
h2.fun . . . 73
imputev . . . 75
jet.colors . . . 76
leg . . . 77
list2usmat . . . 78
manhattan . . . 79
map.plot . . . 80
MEMMA . . . 82
mmer . . . 84
mmer2 . . . 93
overlay . . . 98

pedtoK 100
pin 101
plot.mmer 103
predict.mmer 104
randef 105
residuals.mmer 105
simGECorMat 106
spatPlots 107
spl2D 109
summary.mmer 112
transformConstraints 112
transp 113
uncm 114
unsBLUP 115
unsm 116
us 117
vs 118

Index **122**

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

Description

Sommer is a structural multivariate-univariate linear mixed model solver for multiple random effects allowing the specification and/or estimation of variance covariance structures. ML/REML estimates can be obtained using the Direct-Inversion Newton-Raphson, Average Information and Efficient Mixed Model Association algorithms coded in C++ using the Armadillo library to optimize dense matrix operations common in genomic selection models. Sommer was designed for genomic prediction and genome wide association studies (GWAS) to include i.e. additive, dominance and epistatic relationship structures or other covariance structures, but also functional as a regular mixed model program.

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, but at the same time allowing to perform their real analysis in diploid and polyploid organisms with small and medium-size data sets (< 10,000 observations for average computers given the computational burden carried by the direct-inversion algorithms). The package is focused in the p > n problem and dense covariance structures when the direct-inversion algorithm becomes faster than MME-based algorithms. **The core of the package is the `mmer` (formula-based) function** that fit the multivariate linear mixed models. This package returns variance-covariance components, BLUPs, BLUEs, residuals, fitted values, variances-covariances for fixed and random effects, etc. The package provides kernels to estimate additive (`A.mat`), dominance (`D.mat`), and epistatic (`E.mat`) relationship matrices for diploid and polyploid organisms. It also provides flexibility to fit other genetic models such as full and half diallel models and random regression models. In addition the `pin` function can be used to estimate standard errors for linear combinations of variance components (i.e. ratios like h2). A good converter from letter code to numeric format is implemented in the function `atcg1234`, which supports

higher ploidy levels than diploid. Recently, spatial modeling has been added added to sommer using the two-dimensional splines (spl2D) to face the lack of AR1 covariance structures in sommer.

Starting with version 3.0 the GWAS models that use the M argument have their own function GWAS and is not part of the mmer function.

Keeping updated

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

library(devtools)

install_github("covaruber/sommer")

This is recommended since bugs fixes will be immediately available in the GitHub source but not in CRAN until the next update.

Tutorials

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

vignette("sommer.start")

vignette("sommer")

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

Getting started

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

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

* DT_h2 to calculate heritability

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

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

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

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

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

* DT_btdata dataset contains an animal (birds) model.

* DT_legendre simulated dataset for random regression model.

Other functions such as summary, fitted, randef (notice here is randef not ranef), anova, variogram, residuals, coef and plot applicable to typical linear models can also be applied to models fitted using the mmer-type of functions. Additional functions for genetic analysis have been included such as heritability (h2.fun), build a genotypic hybrid marker matrix (build.HMM), plot of genetic maps (map.plot), creation of manhattan plots (manhattan). If you need to use pedigree you need to convert your pedigree into a relationship matrix (use the ‘getA‘ function from the pedigreemm package).

Useful functions for analyzing field trials are included such as the spl2D, and fill.design.

Differences of sommer >= 3.7 with previous versions

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

Differences of sommer >= 3.0 with previous versions

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

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

The Average Information algorithm had been removed in the past from the package because of its instability to deal with very complex models without good initial values. Now after 3.7 I have brought it back after I noticed that starting with NR the first three iterations gives enough flexibility to the AI algorithm.

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

Models Enabled

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

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

where K is the relationship or covariance matrix for the kth random effect (u=1,...,k), and R=I is an identity matrix for the residual term. The terms $\sigma^2_{g_i}$ and $\sigma^2_{\epsilon_i}$ denote the genetic (or any of the kth

random terms) and residual variance of trait i, respectively and $\sigma_{g_{ij}}$ and $\sigma_{\epsilon_{ij}}$ the genetic (or any of the kth random terms) and residual covariance between traits i and j (i=1,...,t, and j=1,...,t). The algorithm implemented optimizes the log likelihood:

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

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

GWAS Models

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

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

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

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

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

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

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

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

Bug report and contact

If you have any questions or suggestions please post it in https://stackoverflow.com or https://stats.stackexchange.com and send me an email with the link at cova_ruber@live.com.mx

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

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

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

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

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

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

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

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

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

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

Examples

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

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

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

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

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

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

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

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

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

ans3 <- mmer(Yield~Env,
             random=~ vs(us(Env),Name),
             rcov=~vs(us(Env),units),
             data=DT)
summary(ans3)

# ####==========================================####
# #### Multivariate homogeneous variance models ####
# ####==========================================####
#
# ## Multivariate Compound simmetry (CS) model
# DT$EnvName <- paste(DT$Env,DT$Name)
# ans4 <- mmer(cbind(Yield, Weight) ~ Env,
#              random= ~ vs(Name, Gtc = unsm(2)) + vs(EnvName,Gtc = unsm(2)),
#              rcov= ~ vs(units, Gtc = unsm(2)),
#              data=DT)
# summary(ans4)
#
# ####==========================================####
```

```
# #### Multivariate heterogeneous variance models  ####
# ####=============================================####
#
# ## Multivariate Compound simmetry (CS) + Diagonal (DIAG) model
# ans5 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(Name, Gtc = unsm(2)) + vs(ds(Env),Name, Gtc = unsm(2)),
#               rcov= ~ vs(ds(Env),units, Gtc = unsm(2)),
#               data=DT)
# summary(ans5)
#
# ####=========================================####
# #### Multivariate unstructured variance models ####
# ####=========================================####
#
# ans6 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(us(Env),Name, Gtc = unsm(2)),
#               rcov= ~ vs(ds(Env),units, Gtc = unsm(2)),
#               data=DT)
# summary(ans6)
```

`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). If using this function please cite 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` – the core function of the package

Examples

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

A <- A.mat(X)

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

---

adiag1 *Binds arrays corner-to-corner*

Description

Array generalization of blockdiag()

Usage

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

Arguments

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

Details

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

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

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

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

Value

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

Note

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

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

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

Author(s)

Peter Wolf with some additions by Robin Hankin

See Also

`mmer` – the core function of the package

Examples

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

## dropped dimensions can count:

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

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

## dimnames are preserved if not null:

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

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

## following example should make the pattern clear:
adiag1(q,q,pad=1:36)
```

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

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

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

---

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

Description

anova method for class "mmer".

Usage

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

Arguments

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

Value

vector of anova

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

anova, mmer

AR1 *Autocorrelation matrix of order 1.*

Description

Creates an autocorrelation matrix of order one with parameters specified.

Usage

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

Arguments

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

Details

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

Value

If everything is defined correctly the function returns:

$nn the correlation matrix

References

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

See Also

The core function of the package mmer

Examples

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

ARMA *Autocorrelation Moving average.*

Description

Creates an ARMA matrix of order one with parameters specified.

Usage

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

Arguments

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

Details

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

Value

If everything is defined correctly the function returns:

$nn the correlation matrix

References

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

See Also

The core function of the package mmer

Examples

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

at *at covariance structure*

Description

at creates a diagonal covariance structure for specific levels of the random effect.

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function vs to know how to use at in the mmer solver.

Examples

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

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, ref.alleles=NULL)
```

Arguments

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

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

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

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

See Also

The core function of the package mmer

Examples

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

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

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

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))
}
```

bivariateRun *bivariateRun functionality*

Description

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

Usage

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

Arguments

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

Value

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

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

$model the results from the bivariate models.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The core function of the package mmer

Examples

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

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

`build.HMM` *Build a hybrid marker matrix using parental genotypes*

Description

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

Usage

```
build.HMM(M1,M2, custom.hyb=NULL, return.combos.only=FALSE)
```

Arguments

`M1`	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 not allowed.
`M2`	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 not allowed.
`custom.hyb`	A data frame with columns 'Var1' 'Var2', 'hybrid' which specifies which hybrids should be built using the M1 and M2 matrices provided.
`return.combos.only`	A TRUE/FALSE statement inicating if the function should skip building the geotype matrix for hybrids and only return the data frame with all possible combinations to be build. In case the user wants to subset the hybrids before building the marker matrix.

Details

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

Value

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

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

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

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

References

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

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

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

See Also

mmer– the core function of the package

Examples

```
####============================================####
#### use Technow data as example
####============================================####
data(DT_technow)
DT <- DT_technow
Md <- Md_technow
Mf <- Mf_technow
Ad <- Ad_technow
Af <- Af_technow

M.flint <- Mf # Marker matrix Flint
M.dent <- Md # Marker matrix Dent

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

## build the marker matrix for the first 50 hybrids
res2 <- build.HMM(M.dent, M.flint,
                 custom.hyb = res1$data.used[1:50,]
                 )
res2$HMM.add[1:5,1:5]
res2$HMM.dom[1:5,1:5]

## now you can use the A.mat(), D.mat() and E.mat() functions
```

```
# M <- res2$HMM.add
# A <- A.mat(M)
# D <- D.mat(M)
```

`coef.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`

`CS` *Compound symmetry matrix*

Description

Creates a compound symmetry matrix with parameters specified.

Usage

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

Arguments

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

Details

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

Value

If everything is defined correctly the function returns:

$nn the correlation matrix

References

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

See Also

The core function of the package mmer

Examples

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

---

cs *customized covariance structure*

Description

cs creates a customized covariance structure for specific levels of the random effect.

Usage

```
cs(x, mm)
```

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function `vs` to know how to use `cs` in the `mmer` solver.

Examples

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

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=3)
```

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) used for A.mat, method 2 is Nishio ans Satoh. (2014), method 3 is Su et al. (2012). See references.

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

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

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

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### 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)
```

---

ds *diagonal covariance structure*

Description

ds creates a diagonal covariance structure for the levels of the random effect.

Usage

```
ds(x)
```

Arguments

x	vector of observations for the random effect.

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function vs to know how to use ds in the mmer solver.

Examples

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

DT_augment *DT_augment design example.*

Description

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

Usage

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

Format

The format is: chr "DT_augment"

Source

This data was generated by a potato study.

References

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

See Also

The core functions of the package mmer

Examples

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

DT_btdata *Blue Tit Data for a Quantitative Genetic Experiment*

Description

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

Usage

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

Format

The format is: chr "DT_btdata"

References

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

See Also

The core functions of the package mmer

Examples

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

# ####=========================================####
# ####=========================================####
# #### EXAMPLE 1
# #### simple example
# ####=========================================####
# ####=========================================####
# data(DT_btdata)
# DT <- DT_btdata
# head(DT)
# mix4 <- mmer(tarsus ~ sex,
#              random = ~ dam + fosternest,
#              rcov=~units,
#              data = DT)
# summary(mix4)
#
# ####=========================================####
# ####=========================================####
# ####=========================================####
# #### EXAMPLE 2
# #### more complex multivariate model
# ####=========================================####
# ####=========================================####
# data(DT_btdata)
# DT <- DT_btdata
# mix3 <- mmer(cbind(tarsus, back) ~ sex,
#                random = ~ vs(dam) + vs(fosternest),
#                rcov= ~ vs(units, Gtc=diag(2)),
#                data = DT)
# summary(mix3)
# #### calculate the genetic correlation
# cov2cor(mix3$sigma$`u:dam`)
# cov2cor(mix3$sigma$`u:fosternest`)
```

`DT_cornhybrids` *Corn crosses and markers*

Description

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

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

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

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

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

Usage

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

Format

The format is: chr "DT_cornhybrids"

Source

This data was generated by a corn study.

References

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####
#
# data(DT_cornhybrids)
# DT <- DT_cornhybrids
# DTi <- DTi_cornhybrids
# GT <- GT_cornhybrids
# hybrid2 <- DT # extract cross data
# A <- GT
# K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
# K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
# S <- kronecker(K1, K2) ; dim(S)
# rownames(S) <- colnames(S) <- levels(hybrid2$SCA)
#
```

```
# ans <- mmer(Yield ~ Location,
#              random = ~ vs(GCA1,Gu=K1) + vs(GCA2,Gu=K2) + vs(SCA,Gu=S),
#              rcov=~units,
#              data=hybrid2)
#
#
# ####==========================================####
# ####==========================================####
# #### Example of multivariate model
# ####==========================================####
# ####==========================================####
#
# data(DT_cornhybrids)
# hybrid2 <- DT_cornhybrids # extract cross data
# DTi <- DTi_cornhybrids
# GT <- GT_cornhybrids
# hybrid2 <- hybrid2[which(!is.na(hybrid2$Yield)),]
# names(hybrid2)[5:6] <- c("TY","PH")
# head(hybrid2)
#
# A <- GT
# K1 <- A[levels(hybrid2$GCA1), levels(hybrid2$GCA1)]; dim(K1)
# K2 <- A[levels(hybrid2$GCA2), levels(hybrid2$GCA2)]; dim(K2)
# S <- kronecker(K1, K2) ; dim(S)
# rownames(S) <- colnames(S) <- levels(hybrid2$SCA)
#
# ans <- mmer(cbind(TY,PH) ~ Location,
#              random = ~ vs(GCA2,Gu=K2) + vs(SCA,Gu=S),
#              rcov = ~ vs(units,Gtc=diag(2)),
#              data=hybrid2)
```

DT_cpdata *Genotypic and Phenotypic data for a CP population*

Description

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

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

Usage

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

Format

The format is: chr "DT_cpdata"

Source

This data was simulated for fruit breeding applications.

References

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####
#
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT) # additive relationship matrix
# #### look at the data and fit the model
# head(DT)
# mix1 <- mmer(Yield~1,
#               random=~vs(id,Gu=A)
#                       + Rowf + Colf,
#               rcov=~units,
#               data=DT)
# summary(mix1)
#
# ####============================================####
# #### adding dominance and forcing the other VC's
# ####============================================####
#
# DT$idd <- DT$id;
# A <- A.mat(GT) # additive relationship matrix
# D <- D.mat(GT) # dominance relationship matrix
# mm <- matrix(3,1,1);mm ## matrix to fix the var comp
# mix2 <- mmer(Yield~1,
#               random=~vs(id, Gu=A, Gt=mix1$sigma_scaled$id, Gtc=mm)
#                       + vs(Rowf,Gt=mix1$sigma_scaled$Rowf, Gtc=mm)
#                       + vs(Colf,Gt=mix1$sigma_scaled$Colf, Gtc=mm)
```

```
#                       + vs(idd, Gu=D, Gtc=unsm(1)),
#               rcov=~vs(units,Gt=mix1$sigma_scaled$units, Gtc=mm),
#               data=DT)
# summary(mix2)
#
# ####====================####
# #### multivariate model ####
# ####     2 traits       ####
# ####====================####
# #### be patient take some time
# ans.m <- mmer(cbind(Yield,color)~1,
#                random=~ vs(id, Gu=A)
#                + vs(Rowf,Gtc=diag(2))
#                + vs(Colf,Gtc=diag(2)),
#                rcov=~ vs(units),
#                data=DT)
# cov2cor(ans.m$sigma$`u:id`)
#
```

DT_example *Broad sense heritability calculation.*

Description

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

Usage

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

Format

The format is: chr "DT_example"

Source

This data was generated by a potato study.

References

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

See Also

The core functions of the package mmer

Examples

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

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

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

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

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

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

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

ans3 <- mmer(Yield~Env,
            random=~ vs(us(Env),Name),
            rcov=~vs(us(Env),units),
            data=DT)
summary(ans3)

# ####=========================================####
# #### Multivariate homogeneous variance models ####
# ####=========================================####
#
```

```
# ## Multivariate Compound simmetry (CS) model
# DT$EnvName <- paste(DT$Env,DT$Name)
# ans4 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(Name) + vs(EnvName),
#               rcov= ~ vs(units),
#               data=DT)
# summary(ans4)
#
# ####============================================####
# #### Multivariate heterogeneous variance models  ####
# ####============================================####
#
# ## Multivariate Compound simmetry (CS) + Diagonal (DIAG) model
# ans5 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(Name) + vs(ds(Env),Name),
#               rcov= ~ vs(ds(Env),units),
#               data=DT)
# summary(ans5)
#
# ####============================================####
# #### Multivariate unstructured variance models ####
# ####============================================####
#
# ans6 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(us(Env),Name),
#               rcov= ~ vs(ds(Env),units),
#               data=DT)
# summary(ans6)
```

DT_expdesigns *Data for different experimental designs*

Description

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

1) Augmented designs (2 examples)

2) Incomplete block designs (1 example)

3) Split plot design (2 examples)

4) Latin square designs (1 example)

5) North Carolina designs I,II and III

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

Format

Different based on the design.

Source

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

References

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

Examples

```
# #### ================================== ####
# #### ===== Augmented Block Design 1 ==== ####
# #### ================================== ####
# data(DT_expdesigns)
# DT <- DT_expdesigns
# names(DT)
# data1 <- DT$au1
# head(data1)
# ## response variable: "yield"
# ## check indicator: "entryc" ('nc' for all unreplicated, but personal.name for checks)
# ## blocking factor: "block"
# ## treatments, personal names for replicated and non-replicated: "trt"
# ## check no check indicator: "new"
# mix1 <- mmer(yield~entryc,
#              random=~block+trt,
#              rcov=~units,
#              data=data1)
# summary(mix1)
#
# #### ================================== ####
# #### ==== North Carolina Design III ==== ####
# #### ================================== ####
#
# data.car3 <- DT$car3
# data.car3$setrep <- paste(data.car3$set,data.car3$rep,sep=":")
# head(data.car3)
# ## response variable: "yield"
# ## male indicator: "male"
# ## female indicator: "female"
# ## replication: "rep"
# ## set of males: "set"
# mix.car3 <- mmer(yield ~ set,
#                  random=~ male
#                         + female ,
#                  rcov=~units,
#                  data=data.car3)
# (suma <- summary(mix.car3))
```

DT_fulldiallel *Full diallel data for corn hybrids*

Description

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

Usage

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

Format

The format is: chr "DT_fulldiallel"

Source

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

References

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

See Also

The core functions of the package mmer

Examples

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

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

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

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

DT_gryphon *Gryphon data from the Journal of Animal Ecology*

Description

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

The dataset contains 3 elements:

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

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

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

Usage

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

Format

The format is: chr "DT_gryphon"

Source

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

References

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

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####============================================####
# data(DT_gryphon)
# DT <- DT_gryphon
# A <- A_gryphon
# P <- P_gryphon

# #### look at the data
# head(DT)
# #### fit the model with no fixed effects (intercept only)
# mix1 <- mmer(BWT~1,
#              random=~vs(ANIMAL,Gu=A),
#              rcov=~units,
#              data=DT)
# summary(mix1)

# #### fit the multivariate model with no fixed effects (intercept only)
# mix2 <- mmer(cbind(BWT,TARSUS)~1,
#              random=~vs(ANIMAL,Gu=A),
#              rcov=~vs(units),
#              na.method.Y = "include2",
#              data=DT)
# summary(mix2)
# cov2cor(mix2$sigma$`u:ANIMAL`)
# cov2cor(mix2$sigma$`u:units`)
```

DT_h2 *Broad sense heritability calculation.*

Description

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

Usage

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

Format

The format is: chr "DT_h2"

Source

This data was generated by a potato study.

References

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

See Also

The core functions of the package mmer

Examples

```
####============================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
####============================================####
data(DT_h2)
DT <- DT_h2
head(DT)
# ####============================================####
# #### fit the mixed model (very heavy model)
# ####============================================####
# ans1 <- mmer(y~Env,
#              random=~vs(ds(Env),Name) + vs(ds(Env),Block),
#              rcov=~vs(ds(Env),units),
#              data=DT)
# summary(ans1)
```

DT_halfdiallel *half diallel data for corn hybrids*

Description

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

Usage

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

Format

The format is: chr "DT_halfdiallel"

Source

This data was generated by a corn study.

References

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

See Also

The core functions of the package mmer

Examples

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

data(DT_halfdiallel)
DT <- DT_halfdiallel
head(DT)

####============================================####
####============================================####
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)
#### model using overlay
modh <- mmer(sugar~1,
             random=~vs(overlay(femalef,malef)) + genof,
             rcov=~units,
             data=DT)
summary(modh)
#### model using overlay and covariance structures

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 <- mmer(sugar~1,
             random=~ vs(overlay(female,male),Gu=A) + geno,
             data=DT)
summary(modh2)
```

DT_legendre *Simulated data for random regression*

Description

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

Usage

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

Format

The format is: chr "DT_legendre"

Source

This data was simulated for fruit breeding applications.

References

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

See Also

The core functions of the package mmer

Examples

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

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

Description

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

Usage

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

Format

The format is: chr "DT_polyploid"

Source

This data was extracted from Rosyara (2016).

References

If using this data for your own research please cite:

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

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

See Also

The core functions of the package mmer

Examples

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

data(DT_polyploid)
# DT <- DT_polyploid
# GT <- GT_polyploid
# MP <- MP_polyploid
```

```
# ####=========================================####
# ####### convert markers to numeric format
# ####=========================================####
# numo <- atcg1234(data=GT, ploidy=4);
# numo$M[1:5,1:5];
# numo$ref.allele[,1:5]
#
# ###=========================================####
# ###### plants with both genotypes and phenotypes
# ###=========================================####
# common <- intersect(DT$Name,rownames(numo$M))
#
# ###=========================================####
# ### get the markers and phenotypes for such inds
# ###=========================================####
# marks <- numo$M[common,]; marks[1:5,1:5]
# DT2 <- DT[match(common,DT$Name),];
# DT2 <- as.data.frame(DT2)
# DT2[1:5,]
#
# ###=========================================####
# ###### Additive relationship matrix, specify ploidy
# ###=========================================####
# A <- A.mat(marks, ploidy=4)
# D <- D.mat(marks, ploidy=4)
# ###=========================================####
# ### run as mixed model
# ###=========================================####
# ans <- mmer(tuber_shape~1,
#             random=~vs(Name, Gu=A),
#             data=DT2)
# summary(ans)
#
# ###=========================================####
# ### run it as GWAS model
# ###=========================================####
# ans2 <- GWAS(tuber_shape~1,
#              random=~vs(Name,Gu=A),
#              rcov=~units,
#              gTerm = "Name",
#              M=marks, data=DT2)
# summary(ans2)
# plot(ans2$scores[1,])
# plot(ans2$r2m[1,])
```

DT_rice *Rice lines dataset*

Description

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

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

Usage

```
data(DT_rice)
```

Format

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

Source

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

References

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

See Also

The core functions of the package mmer

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####=========================================####
data(DT_rice)
# DT <- DT_rice
# GT <- GT_rice
# GTn <- GTn_rice
# head(DT)
# M <- atcg1234(GT)
# A <- A.mat(M$M)
# mix <- mmer(Protein.content~1,
#             random = ~vs(geno, Gu=A) + geno,
#             rcov=~units,
#             data=DT)
```

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

Description

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

Usage

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

Format

The format is: chr "DT_technow"

Source

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

References

If using this data for your own research please cite:

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

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

See Also

The core functions of the package mmer

Examples

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

```
Md <- Md_technow
Mf <- Mf_technow
Ad <- Ad_technow
Af <- Af_technow
####=========================================####
####=========================================####
# ans2 <- mmer(GY~1,
#              random=~vs(dent,Gu=Ad) + vs(flint,Gu=Af),
#              rcov=~units,
#              data=DT)
# summary(ans2)

####=========================================####
#### multivariate overlayed model
####=========================================####
# M <- rbind(Md,Mf)
# A <- A.mat(M)
# ans3 <- mmer(cbind(GY,GM)~1,
#              random=~vs(overlay(dent,flint),Gu=A),
#              rcov=~vs(units,Gtc=diag(2)),
#              data=DT)
# summary(ans2)
# cov2cor(ans3$sigma[[1]])
# ####=========================================####
# #### Hybrid GWAS
# ####=========================================####
# M <- (rbind(Md,Mf) *2 )-1
# inds <- colnames(overlay(DT$dent,DT$flint)[[1]])
# Minds <- M[inds,]
#
# A <- A.mat(Minds)
# A[1:4,1:4]
# ans3 <- GWAS(GM~1, iters = 20,
#              random=~vs(overlay(dent,flint),Gu=A),
#              rcov=~vs(units),na.method.Y = "include",
#              M=Minds, gTerm="dent",
#              data=DT)
# plot(ans3$scores[1,])
```

`DT_wheat` *wheat lines dataset*

Description

Information from a collection of 599 historical CIMMYT wheat lines. The wheat data set is from CIMMYT's Global Wheat Program. Historically, this program has conducted numerous international trials across a wide variety of wheat-producing environments. The environments represented in these trials were grouped into four basic target sets of environments comprising four main agroclimatic regions previously defined and widely used by CIMMYT's Global Wheat Breeding Program. The phenotypic trait considered here was the average grain yield (GY) of the 599 wheat lines evaluated in each of these four mega-environments.

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

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

Usage

```
data(DT_wheat)
```

Format

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

Source

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

References

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

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

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

See Also

The core functions of the package mmer

Examples

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

# data(DT_wheat)
```

```
# DT <- DT_wheat
# GT <- GT_wheat
# DT <- as.data.frame(DT);colnames(DT) <- paste0("x",1:4);DT$line <- rownames(DT);
# rownames(GT) <- DT$line
# K <- A.mat(GT) # additive relationship matrix
# K[1:4,1:4]
####=========================================####
####=========================================####
#### using formula based 'mmer2'
####=========================================####
####=========================================####
# head(DT)
# #### univariate
# mix0 <- mmer(x1~1,
#              random = ~vs(line,Gu=K),
#              rcov=~vs(units),
#              data=DT)
```

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

Description

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

Format

`block` block factor with 6 levels

`nitro` nitrogen treatment in hundredweight per acre

`Variety` genotype factor, 3 levels

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

`row` row location

`column` column location

Source

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

References

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

Examples

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

`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`

Examples

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

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

EM *Expectation Maximization Algorithm*

Description

Univariate version of the expectation maximization (EM) algorithm.

Usage

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

Arguments

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

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

Details

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

y = Xb + Zu + e

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

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

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

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

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

.

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

.

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

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

.

where G is:

.

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

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

.

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

.

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

.

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

.

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

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

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

.

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

Value

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

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

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

$u.hat a vector with BLUPs for random effects

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

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

$beta.hat a vector for BLUEs of fixed effects

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

$X incidence matrix for fixed effects

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

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

References

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744 Bernardo Rex. 2010. Breeding for quantitative traits in plants. Second edition. Stemma Press. 390 pp. Searle. 1993. Applying the EM algorithm to calculating ML and REML estimates of variance components. Paper invited for the 1993 American Statistical Association Meeting, San Francisco.

See Also

The core functions of the package mmer

Examples

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

# ################################################
# ################################################
# # IMPORT 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=20)
```

`fcm` *fixed effect constraint indication matrix*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

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

Examples

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

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

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

`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`

Examples

```
####=========================================####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples using
#### command + shift + C |OR| control + shift + C
####=========================================####
# data(DT_cpdata)
# DT <- DT_cpdata
# #### look at the data
# head(DT)
# #### fill the design
# gg <- fill.design(x=DT, 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`

`fixm` *fixed indication matrix*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

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

Examples

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

GWAS *Genome wide association study analysis*

Description

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

The package also provides functions to estimate additive (`A.mat`), dominance (`D.mat`), and epistatic (`E.mat`) relationship matrices to model known covariances among genotypes typical in plant and animal breeding problems. Other functions to build known covariance structures among levels of random effects are autoregresive (`AR1`), compound symmetry (`CS`) and autoregressive moving average (`ARMA`) where the user needs to fix the correlation value for such models (this is different to estimating unknown covariance structures). Additionally, overlayed models can be implemented as well (`overlay` function). Spatial modeling can be done through the two dimensional splines (`spl2D`). Random regression models can also be fitted through the (`leg`) function (orthopolynom package installation is needed for using the leg function).

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

library(devtools)

install_github("covaruber/sommer")

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

vignette("sommer.start")

vignette("sommer.changes")

vignette("sommer.FAQ")

vignette("sommer")

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

Usage

```
GWAS(fixed, random, rcov, data, weights,
    iters=20, tolpar = 1e-03, tolparinv = 1e-06,
    init=NULL, constraints=NULL,method="NR",
    getPEV=TRUE,na.method.X="exclude",
    na.method.Y="exclude",return.param=FALSE,
    date.warning=TRUE,verbose=TRUE,
    M=NULL, gTerm=NULL, n.PC = 0, min.MAF = 0.05,
    n.core = 1, P3D = TRUE)
```

Arguments

`fixed`	a formula specifying the **response variable(s) and fixed effects**, i.e: *Yield ~ Location* for univariate models *cbind(Yield,color) ~ Location* for multivariate models.
`random`	a formula specifying the name of the **random effects**, i.e. *random= ~ genotype + year*. Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*): `vs(...,Gu,Gt,Gtc)` is the main function to specify variance models and special structures for random effects. On the ... argument you provide the unknown variance-covariance structures (i.e. us,ds,at,cs) and the random effect where such covariance structure will be used (the random effect of interest). Gu is used to provide known covariance matrices among the levels of the random effect, Gt initial values and Gtc for constraints. Auxiliar functions for building the variance models are: ** `ds(x)`, `us(x)`, `cs(x)` and `at(x,levs)` can be used to specify unknown diagonal, unstructured and customized unstructured and diagonal covariance structures to be estimated by REML. ** `unsm(x)`, `uncm(x)`, `fixm(x)` and `diag(x)` can be used to build easily matrices to specify constraints in the Gtc argument of the `vs()` function. ** `overlay()`, `spl2D()`, and `leg()` functions can be used to specify overlayed of design matrices of random effects, two dimensional spline and random regression models within the `vs()` function.
`rcov`	a formula specifying the name of the **error term**, i.e. *rcov= ~ units*. The functions that can be used to fit heterogeneous residual variances are the same used on the random term but the random effect is always "units", i.e. *rcov=~vs(ds(Location),units)*
`data`	a data frame containing the variables specified in the formulas for response, fixed, and random effects.
`weights`	name of the covariate for weights. To be used for the product R = Wsi*R*Wsi, where * is the matrix product, Wsi is the square root of the inverse of W and R is the residual matrix.
`iters`	Maximum number of iterations allowed. Default value is 15.
`tolpar`	Convergence criteria.
`tolparinv`	tolerance parameter for matrix inverse used when singularities are encountered.
`init`	initial values for the variance components. By default this is NULL and variance components are estimated by the method selected, but in case the user want to provide initial values for ALL var-cov components this argument is functional. It has to be provided as a list or an array, where each list element is one variance component and if multitrait model is pursued each element of the list is a matrix of variance covariance components among traits. Initial values can also be provided in the Gt argument of the vs function.Is highly encouraged to use the Gt and Gtc arguments of the vs function instead of this argument

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

`method` this refers to the method or algorithm to be used for estimating variance components. Direct-inversion Newton-Raphson **NR** and Average Information **AI** (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2015), and **EMMA** efficient mixed model association (Kang et al. 2008).

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

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

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

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

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

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

`M` The marker matrix containing the marker scores for each level of the random effect selected in the gTerm argument, coded as -1,0,1 = aa,Aa,AA, levels (i.e. individuals) in rows and markers in columns. No additional columns should be provided, is a purely numerical matrix.

`gTerm` a character vector indicating the random effect linked to the marker matrix M (i.e. the genetic term) in the model. The random effect selected should have the same number of levels than the number of rows of M.

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

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

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (use only at UNIX command line).

`P3D` When P3D=TRUE, 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.

Details

Models Enabled

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

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

with X = ZMi

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

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

Special Functions

`vs(at(x,levels),y)`

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

`vs(ds(x),y)`

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

`vs(us(x),y)`

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

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

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

`vs(spl2D(x.coord,y.coord,at,at.levels,type,nseg,pord,degree,nest.div))`

can be used to fit a 2-dimensional spline (i.e. spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class). The 2D spline can be fitted at specific levels using the `at` and `at.levels` arguments. For example *random=~spl2D(x.coord=Row,y.coord=Range,at=FIELD)*.

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

vignette('sommer.start')

Bug report and contact

If you have any technical questions or suggestions please post it in https://stackoverflow.com or https://stats.stackexchange.com and send me an email with the link at cova_ruber@live.com.mx

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

Example Datasets

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

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

* DT_h2 to calculate heritability

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

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

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

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

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

* DT_btdata dataset contains an animal (birds) model.

Additional Functions

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

Additional functions for genetic analysis have been included such as heritability (h2.fun), build a genotypic hybrid marker matrix (build.HMM), plot of genetic maps (map.plot), creation of manhattan plots (manhattan). If you need to use pedigree you need to convert your pedigree into a relationship matrix (i.e. use the getA function from the pedigreemm package).

Useful functions for analyzing field trials are included such as the spl2D, spatPlots, and fill.design.

Value

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

Vi	the inverse of the phenotypic variance matrix V^- = (ZGZ+R)^-1
sigma	a list with the values of the variance-covariance components with one list element for each random effect.
sigma_scaled	a list with the values of the scaled variance-covariance components with one list element for each random effect.
sigmaSE	Hessian matrix containing the variance-covariance for the variance components. SE's can be obtained taking the square root of the diagonal values of the Hessian.
Beta	a data frame for trait BLUEs (fixed effects).
VarBeta	a variance-covariance matrix for trait BLUEs
U	a list (one element for each random effect) with a data frame for trait BLUPs.
VarU	a list (one element for each random effect) with the variance-covariance matrix for trait BLUPs.

PevU	a list (one element for each random effect) with the predicted error variance matrix for trait BLUPs.
fitted	Fitted values y.hat=XB
residuals	Residual values e = Y - XB
AIC	Akaike information criterion
BIC	Bayesian information criterion
convergence	a TRUE/FALSE statement indicating if the model converged.
monitor	The values of log-likelihood and variance-covariance components across iterations during the REML estimation.
method	The method for extimation of variance components specified by the user.
call	Formula for fixed, random and rcov used.
scores	marker scores (-log_10p) for the traits.
betasm	marker effects.
Fstatm	F statistic associate to the test.
r2m	R2 value for each marker.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

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

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

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

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

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

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

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

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

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

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

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

Examples

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

# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT) # additive relationship matrix
# #### look at the data and fit the model
# head(DT)
# mix1 <- GWAS(color~1,
#              random=~vs(id,Gu=A)
#              + Rowf + Colf,
#              rcov=~units,
#              data=DT,
#              M=GT, gTerm = "u:id")
# summary(mix1)
# ms <- as.data.frame(t(mix1$scores))
# ms$Locus <- rownames(ms)
# MP2 <- merge(MP,ms,by="Locus",all.x = TRUE);
# manhattan(MP2, pch=20,cex=.5, PVCN = "color score")
#####==========================================####
##### potato example
#####==========================================####
#
# data(DT_polyploid)
# DT <- DT_polyploid
# GT <- GT_polyploid
# MP <- MP_polyploid
# ####==========================================####
# ####### convert markers to numeric format
# ####==========================================####
# numo <- atcg1234(data=GT, ploidy=4);
# numo$M[1:5,1:5];
# numo$ref.allele[,1:5]
#
# ###==========================================####
```

```
# ###### plants with both genotypes and phenotypes
# ###=========================================####
# common <- intersect(DT$Name,rownames(numo$M))
#
# ###=========================================####
# ### get the markers and phenotypes for such inds
# ###=========================================####
# marks <- numo$M[common,]; marks[1:5,1:5]
# DT2 <- DT[match(common,DT$Name),];
# DT2 <- as.data.frame(DT2)
# DT2[1:5,]
#
# ###=========================================####
# ###### Additive relationship matrix, specify ploidy
# ###=========================================####
# A <- A.mat(marks, ploidy=4)
# ###=========================================####
# ### run it as GWAS model
# ###=========================================####
# ans2 <- GWAS(tuber_shape~1,
#              random=~vs(Name,Gu=A),
#              rcov=~units,
#              gTerm = "u:Name",
#              M=marks, data=DT2)
# summary(ans2)
# plot(ans2$scores[1,])
# plot(ans2$r2m[1,])
```

GWAS2 *Genome wide association study*

Description

This function is deprecated. Use GWAS instead. Now the GWAS function can run both types of models; formula-based and matrix-based models. Type ?GWAS.

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

vignette("sommer.start")

vignette("sommer")

Usage

```
GWAS2(fixed, random, rcov, data, weights,
    iters=20, tolpar = 1e-03, tolparinv = 1e-06,
    init=NULL, constraints=NULL,method="NR",
    getPEV=TRUE,na.method.X="exclude",
    na.method.Y="exclude",return.param=FALSE,
```

```
date.warning=TRUE,verbose=TRUE,
M=NULL, gTerm=NULL, n.PC = 0, min.MAF = 0.05,
n.core = 1, P3D = TRUE)
```

Arguments

fixed	a formula specifying the **response variable(s) and fixed effects**, i.e: *Yield ~ Location* for univariate models *cbind(Yield,color) ~ Location* for multivariate models.
random	a formula specifying the name of the **random effects**, i.e. *random= ~ genotype + year*. Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*): `vs(...,Gu,Gt,Gtc)` is the main function to specify special variance-covariance structures for random effects. On the ... argument you provide the unknown variance-covariance structures (i.e. us,ds,at,cs) and the random effect where such covariance structure will be used (the random effect of interest). `at(x,levs)` can be used to specify heterogeneous variance for specific levels of a random effect `ds(x)`, `us(x)`, `cs(x)` can be used to specify unknown diagonal, unstructured and customized covariance structures respectively among levels of a random effect to be estimated by REML. `overlay(...,rlist,prefix)` can be used to specify overlay of design matrices of random effects `spl2D(...)` can be used to fit a 2-dimensional spline (i.e. spatial modeling; see Special functions section below).
rcov	a formula specifying the name of the **error term**, i.e. *rcov= ~ units*. The functions that can be used to fit heterogeneous residual variances are the same used on the random term but the random effect is always "units", i.e. *rcov=~vs(ds(Location),units)*
data	a data frame containing the variables specified in the formulas for response, fixed, and random effects.
weights	name of the covariate for weights. To be used for the product R = Wsi*R*Wsi, where * is the matrix product, Wsi is the square root of the inverse of W and R is the residual matrix.
iters	Maximum number of iterations allowed. Default value is 15.
tolpar	Convergence criteria.
tolparinv	tolerance parameter for matrix inverse used when singularities are encountered.
init	initial values for the variance components. By default this is NULL and variance components are estimated by the method selected, but in case the user want to provide initial values for ALL var-cov components this argument is functional. It has to be provided as a list or an array, where each list element is one variance component and if multitrait model is pursued each element of the list is a matrix of variance covariance components among traits. Initial values can also be provided in the Gt argument of the vs function.Is highly encouraged to use the Gt and Gtc arguments of the vs function instead of this argument

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

`method` this refers to the method or algorithm to be used for estimating variance components. Direct-inversion Newton-Raphson **NR** and Average Information **AI** (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2015), and **EMMA** efficient mixed model association (Kang et al. 2008).

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

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

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

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

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

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

`M` The marker matrix containing the marker scores for each line, coded as -1,0,1 = aa,Aa,AA, individuals in rows and markers in columns. No additional columns should be provided, is a purely numerical matrix.

`gTerm` a character vector indicating the genetic term in the model.

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

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

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (use only at UNIX command line).

`P3D` When P3D=TRUE, 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.

Details

Special Functions

`vs(at(x,levels),y)`

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

`vs(ds(x),y)`

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

`vs(us(x),y)`

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

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

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

`vs(spl2D(x.coord,y.coord,at,at.levels,type,nseg,pord,degree,nest.div))`

can be used to fit a 2-dimensional spline (i.e. spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class). The 2D spline can be fitted at specific levels using the `at` and `at.levels` arguments. For example *random=~spl2D(x.coord=Row,y.coord=Range,at=FIELD)*.

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

vignette('sommer.start')

Bug report and contact

If you have any questions or suggestions please post it in https://stackoverflow.com and send me an email with the link at cova_ruber@live.com.mx

Example Datasets

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

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

* `DT_h2` to calculate heritability

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

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

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

* `DT_polyploid` to fit genomic prediction and GWAS2 analysis in polyploids.

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

* DT_btdata dataset contains an animal (birds) model.

Additional Functions

Other functions such as summary, fitted, randef (notice here is randef not ranef), anova, variogram, residuals, coef and plot applicable to typical linear models can also be applied to models fitted using the GWAS2-type of functions.

Additional functions for genetic analysis have been included such as heritability (h2.fun), build a genotypic hybrid marker matrix (build.HMM), plot of genetic maps (map.plot), creation of manhattan plots (manhattan). If you need to use pedigree you need to convert your pedigree into a relationship matrix (i.e. use the getA function from the pedigreemm package).

Useful functions for analyzing field trials are included such as the spl2D, spatPlots, and fill.design.

Models Enabled

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

Value

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

Vi	the inverse of the phenotypic variance matrix V^- = (ZGZ+R)^-1
sigma	a list with the values of the variance-covariance components with one list element for each random effect.
sigma_scaled	a list with the values of the scaled variance-covariance components with one list element for each random effect.
sigmaSE	standard errors for the variance covariance components.
Beta	a data frame for trait BLUEs (fixed effects).
VarBeta	a variance-covariance matrix for trait BLUEs
U	a list (one element for each random effect) with a data frame for trait BLUPs.
VarU	a list (one element for each random effect) with the variance-covariance matrix for trait BLUPs.
PevU	a list (one element for each random effect) with the predicted error variance matrix for trait BLUPs.
fitted	Fitted values y.hat=XB
residuals	Residual values e = Y - XB
AIC	Akaike information criterion
BIC	Bayesian information criterion
convergence	a TRUE/FALSE statement indicating if the model converged.
monitor	The values of log-likelihood and variance-covariance components across iterations during the REML estimation.
method	The method for extimation of variance components specified by the user.
call	Formula for fixed, random and rcov used.
scores	marker scores (-log_10p) for the traits.
betasm	marker effects.
Fstatm	F statistic associate to the test.
r2m	R2 value for each marker.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

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

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

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

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

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

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

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

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

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

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

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

`h2.fun` *Obtain heritabilities with three different methods*

Description

Obtain heritabilities based on three different methods; Cullis et al. (2006), Oakey at al. (2006), and line-mean h2 (Falconer, 1995).

Cullis at al. (2006): h2= 1 - (PEM.mu/2*Vg)

Oakey at al. (2006): h2= 1 - (tr((0.5*G^-1) * Czz)/m)

Falconer (1995): h2 = Vg / (Vg+(Ve/r*e))

where "PEV.mu" is the average prediction error variance for the genetic term, "Vg" and "Ve" are the genetic and residual variance respectively estimated by REML, "G^-1" is the inverse of A*Vg where "A" is the additive relationship matrix, "Czz" is the prediction error variance for the genetic term, "m" is the number of test lines, "r" is the replicates per environment and "e" the number of environments.

Usage

```
h2.fun(object, data, gTerm=NULL, eTerm=NULL, md=NULL)
```

Arguments

object	a model fitted with the mmer or mmer2 functions.
data	the dataset used to fit the model provided in the object argument.
gTerm	a character vector specifying the genetic terms fitted in the model.
eTerm	a character vector specifying the environment term fitted in the model.
md	a numeric value to multiply the genetic variance in the heritability formulas (see details below). If NULL it will use the mean value of the diagonal from the genomic or relationship matrix.

Details

Please see the description or go to the canonical papers where methods are explained with more detail.

For each level from the eTerm (environment) the heritability is calculated as:

h2.stdr= Vg/(Vg+Ve/(ne*nr)) h2.cullis= 1 - (PEV/(md*Vg)) h2.oakey= 1 - tr[(C22 (Gi*1/md) / m)]

where "Vg" refers to the genotype variance "Ve" the error variance, "ne" number of environments, "nr" number of replicates, "PEV" is the predicted error variance for the genotype (gTerm), "md" is the mean value from the diagonal of the relationship (pedigree or genomic) matrix "G"" and "m" is the number of lines, "Gi" is the inverse of the relationship matrix.

References

Oakey, Helena, et al. "Joint modeling of additive and non-additive genetic line effects in single field trials." Theoretical and Applied Genetics 113.5 (2006): 809-819.

Cullis, Brian R., Alison B. Smith, and Neil E. Coombes. "On the design of early generation variety trials with correlated data." Journal of Agricultural, Biological, and Environmental Statistics 11.4 (2006): 381-393.

Falconer, Douglas S., Trudy FC Mackay, and Richard Frankham. "Introduction to quantitative genetics (4th edn)." Trends in Genetics 12.7 (1996): 280.

See Also

sommer

Examples

```
data(DT_example)
DT <- DT_example
A <- A_example
head(DT)
# ####==========================================####
# #### fit the mixed model (very heavy model)
# ####==========================================####
# ans1 <- mmer(Yield~Env,
#               random=~vs(ds(Env),Name) + vs(ds(Env),Block),
#               rcov=~vs(ds(Env),units),
#               data=DT)
# summary(ans1)
```

---

imputev *Imputing a numeric or character vector*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The core function of the package mmer

Examples

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

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

Description

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

Usage

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

Arguments

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

Value

A vector of colors along the jet colorramp.

See Also

The core function of the package mmer

Examples

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

leg *Legendre polynomial matrix*

Description

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

Usage

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

Arguments

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

Value

$S3 an Legendre polynomial matrix of order n.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The core functions of the package mmer

Examples

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

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

`list2usmat` *list or vector to unstructured matrix*

Description

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

Usage

```
list2usmat(sigmaL)
```

Arguments

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

Value

$res a matrix with the values provided.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

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

Examples

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

manhattan *Creating a manhattan plot*

Description

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

Usage

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

Arguments

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

Value

If all parameters are correctly indicated the program will return:

$plot.data a manhattan plot

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The core functions of the package mmer

Examples

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

---

map.plot *Creating a genetic map plot*

Description

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

Usage

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

Arguments

data	the data frame with 3 columns with names; Locus, LG and Position
trait	if something wants to be plotted next the linkage groups the user must indicate the name of the column containing the values to be ploted, i.e. p-values, LOD scores, X2 segregation distortion values, etc.
trait.scale	is trait is not NULL, this is a character value indicating if the y axis limits for the trait plotted next to the chromosomes should be the same or different for each linkage group. The default value is "same", which means that the same y axis limit is conserved across linkage groups. For giving an individual y axis limit for each linkage group write "diff".
col.chr	a vector with color names for the chromosomes. If NULL they will be drawn in gray-black scale.

col.trait	a vector with color names for the dots, lines or histogram for the trait plotted next to the LG's
type	a character value indicating if the trait should be plotted as scatterplot 'dot', histogram 'hist', line 'line' next to the chromosomes.
cex	the cex value determining the size of the cM position labels in the LGs
lwd	the width of the lines in the plot
cex.axis	the cex value for sizing the labels of LGs and traits plotted (top labels)
cex.trait	the cex value for sizing the dots or lines of the trait plotted
jump	a scalar value indicating how often should be drawn a number next to the LG indicating the position. The default is 5 which means every 5 cM a label will be drawn, i.e. 0,5,10,15,... cM.

Value

If all parameters are correctly indicated the program will return:

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The core functions of the package mmer

Examples

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

```
map.plot(geno, trait="X2", type="line")
map.plot(geno, trait="X2", type="hist")
map.plot(geno, trait="X2", type="dot")
```

MEMMA *Multivariate Efficient Mixed Model Association Algorithm*

Description

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

Usage

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

Arguments

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

Details

.

The likelihood function optimized in this algorithm is:

.

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

.

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

.

The algorithm can be summarized in the next steps:

.

1) provide initial value for the ridge parameter

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

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

4) perform an eigen decomposition of SVS

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

6) estimate eta=U'y

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

Value

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

$Vu a scalar value for the variance component estimated

$Ve a scalar value for the error variance estimated

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

$u.hat a vector with BLUPs for random effects

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

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

$beta.hat a vector for BLUEs of fixed effects

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

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

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

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

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

References

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

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

See Also

The core functions of the package `mmer`

Examples

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

`mmer` **m***ixed* **m***odel* **e***quations in* **R**

Description

Fits a multivariate/univariate linear mixed model by likelihood methods (REML). The optimization methods are; Direct-Inversion Newton-Raphson (NR), Average Information (AI) (Tunnicliffe 1989; Lee et al. 2015; Maier et al. 2015), and Efficient Mixed Model Association (EMMA) (Kang et al. 2008) coded in C++ using the Armadillo library to opmitime dense matrix operations common in the derect-inversion algorithms. These algorithms are **intended to be used for problems of the type p > n or dense matrices**. For problems with sparse covariance structures, or problems of the

type n > p, the MME-based algorithms are faster and we recommend to shift to the use of such software (i.e. lme4, breedR, or asreml-R).

The package also provides functions to estimate additive (`A.mat`), dominance (`D.mat`), and epistatic (`E.mat`) relationship matrices to model known covariances among genotypes typical in plant and animal breeding problems. Other functions to build known covariance structures among levels of random effects are autoregresive (`AR1`), compound symmetry (`CS`) and autoregressive moving average (`ARMA`) where the user needs to fix the correlation value for such models (this is different to estimating unknown covariance structures). Additionally, overlayed models can be implemented as well (`overlay` function). Spatial modeling can be done through the two dimensional splines (`spl2D`). Random regression models can also be fitted through the (`leg`) function (orthopolynom package installation is needed for using the leg function).

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

library(devtools)

install_github("covaruber/sommer")

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

vignette("sommer.start")

vignette("sommer.changes")

vignette("sommer.FAQ")

vignette("sommer")

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

Usage

```
mmer(fixed, random, rcov, data, weights, iters=20, tolpar = 1e-03,
     tolparinv = 1e-06, init=NULL, constraints=NULL,method="NR", getPEV=TRUE,
     na.method.X="exclude", na.method.Y="exclude",return.param=FALSE,
     date.warning=TRUE,verbose=TRUE, reshape.output=TRUE)
```

Arguments

`fixed`	a formula specifying the **response variable(s) and fixed effects**, i.e: *response ~ covariate* for univariate models *cbind(response.i,response.j) ~ covariate* for multivariate models The `fcm()` function can be used to constrain fixed effects in multi-response models.
`random`	a formula specifying the name of the **random effects**, i.e. *random= ~ genotype + year*. Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*):

vs(...,Gu,Gt,Gtc) is the main function to specify variance models and special structures for random effects. On the ... argument you provide the unknown variance-covariance structures (i.e. us,ds,at,cs) and the random effect where such covariance structure will be used (the random effect of interest). Gu is used to provide known covariance matrices among the levels of the random effect, Gt initial values and Gtc for constraints. Auxiliar functions for building the variance models are:

** ds(x), us(x), cs(x) and at(x,levs) can be used to specify unknown diagonal, unstructured and customized unstructured and diagonal covariance structures to be estimated by REML.

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

** overlay(), spl2D(), and leg() functions can be used to specify overlayed of design matrices of random effects, two dimensional spline and random regression models within the vs() function.

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

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

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

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

`iters` Maximum number of iterations allowed.

`tolpar` Convergence criteria for the change in log-likelihood.

`tolparinv` tolerance parameter for matrix inverse used when singularities are encountered in the estimation procedure.

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

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

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

getPEV	a TRUE/FALSE value indicating if the program should return the predicted error variance and variance for random effects. This option is provided since this can take a long time for certain models where p is > n by a big extent.
na.method.X	one of the two possible values; "include" or "exclude". If "include" is selected then the function will impute the X matrices for fixed effects with the median value. If "exclude" is selected it will get rid of all rows with missing values for the X (fixed) covariates. The default is "exclude". The "include" option should be used carefully.
na.method.Y	one of the three possible values; "include", "include2" or "exclude" (default) to treat the observations in response variable to be used in the estimation of variance components. The first option "include" will impute the response variables for all rows with the median value, whereas "include2" imputes the responses only for rows where there is observation(s) for at least one of the responses (only available in the multi-response models). If "exclude" is selected (default) it will get rid of rows in response(s) where missing values are present for at least one of the responses.
return.param	a TRUE/FALSE value to indicate if the program should return the parameters to be used for fitting the model instead of fitting the model.
date.warning	a TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package.
verbose	a TRUE/FALSE value to indicate if the program should return the progress of the iterative algorithm.
reshape.output	a TRUE/FALSE value to indicate if the output should be reshaped to be easier to interpret for the user, some information is missing from the multivariate models for an easy interpretation.

Details

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

Special Functions

```
vs(at(x,levels),y)
```

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

```
vs(ds(x),y)
```

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

```
vs(us(x),y)
```

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

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

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

`vs(spl2D(x.coord,y.coord,at,at.levels,type,nseg,pord,degree,nest.div))`

can be used to fit a 2-dimensional spline (i.e. spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class). The 2D spline can be fitted at specific levels using the `at` and `at.levels` arguments. For example *random=~vs(spl2D(x.coord=Row,y.coord=Range,at=FIELD))*.

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

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

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

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

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

vignette('sommer.start')

Bug report and contact

If you have any technical questions or suggestions please post it in https://stackoverflow.com or https://stats.stackexchange.com and send me an email with the link at cova_ruber@live.com.mx

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

Example Datasets

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

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

* `DT_h2` to calculate heritability

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

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

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

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

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

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

* `DT_legendre` simulated dataset for random regression model.

Additional Functions

Other standard functions such as summary, fitted, randef (notice here is randef not ranef), anova, predict, residuals, coef and plot applicable to typical linear models can also be applied to models fitted using the mmer function.

Additional functions for genetic analysis have been included such as heritability (h2.fun), build a genotypic hybrid marker matrix (build.HMM), plot of genetic maps (map.plot), creation of manhattan plots (manhattan). If you need to use pedigree you need to convert your pedigree into a relationship matrix (i.e. use the getA function from the pedigreemm package).

Useful functions for analyzing field trials are included such as the spl2D, spatPlots, and fill.design.

Models Enabled

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

Value

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Examples

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

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

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

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

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

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

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

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

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

ans3 <- mmer(Yield~Env,
             random=~ vs(us(Env),Name),
             rcov=~vs(us(Env),units),
             data=DT)
summary(ans3)

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

```
#
# ## Multivariate Compound simmetry (CS) + Diagonal (DIAG) model
# ans5 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(Name, Gtc = unsm(2)) + vs(ds(Env),Name, Gtc = unsm(2)),
#               rcov= ~ vs(ds(Env),units, Gtc = unsm(2)),
#               data=DT)
# summary(ans5)
#
# ####============================================####
# #### Multivariate unstructured variance models ####
# ####============================================####
#
# ans6 <- mmer(cbind(Yield, Weight) ~ Env,
#               random= ~ vs(us(Env),Name, Gtc = unsm(2)),
#               rcov= ~ vs(ds(Env),units, Gtc = unsm(2)),
#               data=DT)
# summary(ans6)
#
# ####============================================####
# ####============================================####
# #### EXAMPLE SET 2
# #### 2 variance components
# #### one random effect with variance covariance structure
# ####============================================####
# ####============================================####
#
# data("DT_cpdata")
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# head(DT)
# GT[1:4,1:4]
# #### create the variance-covariance matrix
# A <- A.mat(GT)
# #### look at the data and fit the model
# mix1 <- mmer(Yield~1,
#              random=~vs(id, Gu=A) + Rowf,
#              rcov=~units,
#              data=DT)
# summary(mix1)$varcomp
# #### calculate heritability
# pin(mix1, h1 ~ V1/(V1+V3) )
# #### multi trait example
# mix2 <- mmer(cbind(Yield,color)~1,
#               random = ~ vs(id, Gu=A, Gtc = unsm(2)) + # unstructured at trait level
#                           vs(Rowf, Gtc=diag(2)) + # diagonal structure at trait level
#                             vs(Colf, Gtc=diag(2)), # diagonal structure at trait level
#               rcov = ~ vs(units, Gtc = unsm(2)), # unstructured at trait level
#               data=DT)
# summary(mix2)
#
# ####============================================####
# #### EXAMPLE SET 3
```

```
# #### comparison with lmer, install 'lme4'
# #### and run the code below
# ####==========================================####
#
# #### lmer cannot use var-cov matrices so we will not
# #### use them in this comparison example
#
# library(lme4)
# library(sommer)
# data("DT_cornhybrids")
# DT <- DT_cornhybrids
# DTi <- DTi_cornhybrids
# GT <- GT_cornhybrids
#
# fm1 <- lmer(Yield ~ Location + (1|GCA1) + (1|GCA2) + (1|SCA),
#             data=DT )
# out <- mmer(Yield ~ Location,
#              random = ~ GCA1 + GCA2 + SCA,
#              rcov = ~ units,
#              data=DT)
# summary(fm1)
# summary(out)
# ### same BLUPs for GCA1, GCA2, SCA than lme4
# plot(out$U$GCA1$Yield, ranef(fm1)$GCA1[,1])
# plot(out$U$GCA2$Yield, ranef(fm1)$GCA2[,1])
# vv=which(abs(out$U$SCA$Yield) > 0)
# plot(out$U$SCA$Yield[vv], ranef(fm1)$SCA[,1])
#
# ### a more complex model specifying which locations
# head(DT)
# out2 <- mmer(Yield ~ Location,
#               random = ~ vs(at(Location,c("3","4")),GCA2) +
#                          vs(at(Location,c("3","4")),SCA),
#               rcov = ~ vs(ds(Location),units),
#               data=DT)
# summary(out2)
```

---

mmer2 **m***ixed* **m***odel* **e***quations in* **R**

Description

This function is deprecated. Use mmer instead. Now the mmer function can run both types of models; formula-based and matrix-based models. Type ?mmer.

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

vignette("sommer.start")

vignette("sommer")

Usage

```
mmer2(fixed, random, rcov, data, weights,
    iters=20, tolpar = 1e-03, tolparinv = 1e-06,
    init=NULL, constraints=NULL,method="NR",
    getPEV=TRUE,na.method.X="exclude",
    na.method.Y="exclude",return.param=FALSE,
    date.warning=TRUE,verbose=TRUE,
    reshape.output=TRUE)
```

Arguments

`fixed`	a formula specifying the **response variable(s) and fixed effects**, i.e: *Yield ~ Location* for univariate models *cbind(Yield,color) ~ Location* for multivariate models.
`random`	a formula specifying the name of the **random effects**, i.e. *random= ~ genotype + year*. Useful functions can be used to fit heterogeneous variances and other special models (*see 'Special Functions' in the Details section for more information*): `vs(...,Gu,Gt,Gtc)` is the main function to specify special variance-covariance structures for random effects. On the ... argument you provide the unknown variance-covariance structures (i.e. us,ds,at,cs) and the random effect where such covariance structure will be used (the random effect of interest). Auxiliar functions for building the variance models are: * `ds(x)`, `us(x)`, `cs(x)` can be used to specify unknown diagonal, unstructured and customized covariance structures respectively among levels of a random effect to be estimated by REML. * `at(x,levs)` can be used to specify heterogeneous variance for specific levels of a random effect * `overlay(...,rlist,prefix)` can be used to specify overlay of design matrices of random effects * `spl2D(...)` can be used to fit a 2-dimensional spline (i.e. spatial modeling; see Special functions section below). * `leg(...)` can be used to fit a random regression model.
`rcov`	a formula specifying the name of the **error term**, i.e. *rcov= ~ units*. The functions that can be used to fit heterogeneous residual variances are the same used on the random term but the random effect is always "units", i.e. *rcov=~vs(ds(Location),units)*
`data`	a data frame containing the variables specified in the formulas for response, fixed, and random effects.
`weights`	name of the covariate for weights. To be used for the product R = Wsi*R*Wsi, where * is the matrix product, Wsi is the square root of the inverse of W and R is the residual matrix.
`iters`	Maximum number of iterations allowed. Default value is 15.
`tolpar`	Convergence criteria.

tolparinv	tolerance parameter for matrix inverse used when singularities are encountered.
init	initial values for the variance components. By default this is NULL and variance components are estimated by the method selected, but in case the user want to provide initial values for ALL var-cov components this argument is functional. It has to be provided as a list or an array, where each list element is one variance component and if multitrait model is pursued each element of the list is a matrix of variance covariance components among traits. Initial values can also be provided in the Gt argument of the vs function.Is highly encouraged to use the Gt and Gtc arguments of the vs function instead of this argument
constraints	when initial values are provided these have to be accompanied by their constraints. See the vs function for more details on the constraints. Is highly encouraged to use the Gt and Gtc arguments of the vs function instead of this argument.
method	this refers to the method or algorithm to be used for estimating variance components. Direct-inversion Newton-Raphson **NR** and Average Information **AI** (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2015).
getPEV	a TRUE/FALSE value indicating if the program should return the predicted error variance and variance for random effects. This option is provided since this can take a long time for certain models where p is > n by a big extent.
na.method.X	one of the two possible values; "include" or "exclude". If "include" is selected then the function will impute the X matrices for fixed effects with the median value. If "exclude" is selected it will get rid of all rows with missing values for the X (fixed) covariates. The default is "exclude". The "include" option should be used carefully.
na.method.Y	one of the three possible values; "include", "include2" or "exclude". If "include" is selected then the function will impute the response variables with the median value. The difference between "include" and "include2" is only available in the multitrait models when the imputation can happen for the entire matrix of responses or only for complete cases ("include2"). If "exclude" is selected it will get rid of rows in responses where missing values are present for the estimation of variance components. The default is "exclude".
return.param	a TRUE/FALSE value to indicate if the program should return the parameters used for modeling without fitting the model.
date.warning	a TRUE/FALSE value to indicate if the program should warn you when is time to update the sommer package.
verbose	a TRUE/FALSE value to indicate if the program should return the progress of the iterative algorithm.
reshape.output	a TRUE/FALSE value to indicate if the output should be reshaped to be easier to use, some information is missing from the multivariate models for an easy interpretation.

Details

Special Functions

```
vs(at(x,levels),y)
```

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

`vs(ds(x),y)`

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

`vs(us(x),y)`

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

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

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

`vs(spl2D(x.coord,y.coord,at,at.levels,type,nseg,pord,degree,nest.div))`

can be used to fit a 2-dimensional spline (i.e. spatial modeling) using coordinates `x.coord` and `y.coord` (in numeric class). The 2D spline can be fitted at specific levels using the `at` and `at.levels` arguments. For example *random=~vs(spl2D(x.coord=Row,y.coord=Range,at=FIELD))*.

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

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

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

vignette('sommer.start')

Bug report and contact

If you have any questions or suggestions please post it in https://stackoverflow.com or https://stats.stackexchange.com and send me an email with the link at cova_ruber@live.com.mx

Example Datasets

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

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

* `DT_h2` to calculate heritability

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

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

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

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

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

* DT_btdata dataset contains an animal (birds) model.

* DT_legendre simulated dataset for random regression model.

Additional Functions

Other functions such as summary, fitted, randef (notice here is randef not ranef), anova, variogram, residuals, coef and plot applicable to typical linear models can also be applied to models fitted using the mmer2-type of functions.

Additional functions for genetic analysis have been included such as heritability (h2.fun), build a genotypic hybrid marker matrix (build.HMM), plot of genetic maps (map.plot), creation of manhattan plots (manhattan). If you need to use pedigree you need to convert your pedigree into a relationship matrix (i.e. use the getA function from the pedigreemm package).

Useful functions for analyzing field trials are included such as the spl2D, spatPlots, and fill.design.

Models Enabled

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

Value

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

Vi	the inverse of the phenotypic variance matrix V^- = (ZGZ+R)^-1
sigma	a list with the values of the variance-covariance components with one list element for each random effect.
sigma_scaled	a list with the values of the scaled variance-covariance components with one list element for each random effect.
sigmaSE	standard errors for the variance covariance components.
Beta	a data frame for trait BLUEs (fixed effects).
VarBeta	a variance-covariance matrix for trait BLUEs
U	a list (one element for each random effect) with a data frame for trait BLUPs.
VarU	a list (one element for each random effect) with the variance-covariance matrix for trait BLUPs.
PevU	a list (one element for each random effect) with the predicted error variance matrix for trait BLUPs.
fitted	Fitted values y.hat=XB
residuals	Residual values e = Y - XB
AIC	Akaike information criterion
BIC	Bayesian information criterion
convergence	a TRUE/FALSE statement indicating if the model converged.
monitor	The values of log-likelihood and variance-covariance components across iterations during the REML estimation.
method	The method for extimation of variance components specified by the user.
call	Formula for fixed, random and rcov used.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

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

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

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

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

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

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

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

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

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

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

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

`overlay` *Overlay Matrix*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

See Also

The core functions of the package `mmer`

Examples

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

A <- diag(7); colnames(A) <- rownames(A) <- 1:7;A # if you want to provide a covariance matrix
#### model using overlay
```

```
modh <- mmer(sugar~1,
             random=~vs(overlay(femalef,malef), Gu=A)
                     + genof,
              data=DT)
```

---

pedtoK *Pedigree to matrix*

Description

This function creates takes the inverse of the asreml Ainverse function (ginv element) and creates the additive relationship matrix to use it as a covariance matrix for a random effect. Other packages that allows you to obtain an additive relationship matrix from a pedigree is the 'pedigreemm' package.

Usage

```
pedtoK(x, row="Row",column="Column",value="Ainverse", returnInverse=TRUE)
```

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The core functions of the package mmer

Examples

```
# Ks <- pedtoK(asreml.model$ginv)
# A <- Ks$K
```

`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(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT)
# #### look at the data and fit the model
# head(DT)
# mix1 <- mmer(Yield~1,
#               random=~vs(id,Gu=A),
#               data=DT)
# summary(mix1)
# #### run the pin function
# pin(mix1, h2 ~ V1 / ( V1 + V2 ) )

# ####============================================####
# ####============================================####
# #### EXAMPLE 2
# #### simple example with multivariate models
# ####============================================####
# ####============================================####
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT)
# #### look at the data and fit the model
# head(DT)
# mix2 <- mmer(cbind(Yield,color)~1,
#               random=~vs(id,Gu=A, Gt=unsm(2)),
#               rcov=~vs(units, Gt=unsm(2)),
#              data=DT)
# summary(mix2)
# ## genetic correlation
# pin(mix2, gen.cor ~ V2 / sqrt(V1*V3))
#
# ####============================================####
# ####============================================####
# #### EXAMPLE 3
# #### more complex multivariate model
# ####============================================####
# ####============================================####
# data(DT_btdata)
```

```
# DT <- DT_btdata
# mix3 <- mmer(cbind(tarsus, back) ~ sex,
#                random = ~ vs(dam, Gtc=unsm(2)) + vs(fosternest,Gtc=diag(2)),
#                rcov=~vs(units,Gtc=unsm(2)),
#                data = DT)
# summary(mix3)
# #### calculate the genetic correlation
# pin(mix3, gen.cor ~ V2 / sqrt(V1*V3))
#
# ####=========================================####
# ####=========================================####
# #### EXAMPLE 4
# #### going back to simple examples
# ####=========================================####
# ####=========================================####
# data(DT_btdata)
# DT <- DT_btdata
# mix4 <- mmer2(tarsus ~ sex, random = ~ dam + fosternest,
#                data = DT)
# summary(mix4)
# #### calculate the ratio and its SE
# pin(mix4, dam.prop ~ V1 / ( V1 + V2 + V3 ) )
```

---

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

Description

plot method for class "mmer".

Usage

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

Arguments

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

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

plot, mmer

---

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

Description

predict method for class "mmer".

Usage

```
## S3 method for class 'mmer'
predict(object, classify=NULL,
        RtermsToForce=NULL,
        FtermsToForce=NULL,
        ...)
```

Arguments

object	an object of class "mmer"
classify	an optional character string with the variables that define the margins of the multiway table to be predicted (see Details).
RtermsToForce	numeric vector to force the random effects to be used instead of the ones defined in the classify argument.
FtermsToForce	numeric vector to force the fixed effects to be used instead of the ones defined in the classify argument.
...	Further arguments to be passed

Details

This function allows to produce predictions, either specifying: (1) the data frame on which to obtain the predictions (argument newdata), or (2) those variables that define the margins of the multiway table to be predicted (argument classify). In the first case, all fixed and random components need to be present in the data frame. In the second case, predictions are obtained for each combination of values of the specified variables that is present in the data set used to fit the model.

Value

The data frame used for obtaining the predictions, jointly with the predicted values and the corresponding standard errors.

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

References

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

See Also

predict, mmer

`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, ...)
```

Arguments

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

Value

vector of residuals

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

residuals, mmer

---

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

Description

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

Usage

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

Arguments

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

Details

Simple simulation of a correlation matrix for environments and megaenvironments.

Value

G the correlation matrix

$G the correlation matrix

References

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

See Also

mmer – the core function of the package

Examples

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

---

spatPlots *Spatial plots*

Description

Plot fitted values for all terms in a model to assess spatial fit.

Usage

```
spatPlots(object, by=NULL, colfunc=NULL,
         row="ROW",range="RANGE", wire=FALSE,
         terms=NULL)
```

Arguments

object	a mixed model fitted using mmer2.
by	A character argument specifying the column name in the data frame to fit a difference variance component for each level of such column. For example, if the model to be fitted should have a variance component for each field where the spatial effect row*range will be estimated, then the column that specifies the different fields should be provided to the 'by' argument as a character vector of length one.
colfunc	an optional function to creat colors for the spatial plots.
row	name of the column in the dataset that identifies the spatial position in the x coordinate.
range	name of the column in the dataset that identifies the spatial position in the y coordinate.
wire	a TRUE/FALSE statement indicating if the function should return a wire plot or the default levelplot.
terms	a character vector specifying the term(s) to be fitted in the spatial plot.

Details

The function only takes the asreml model and builds the fitted values for the terms required by building the incidence matrix and getting the BLUPs from the asreml model to build fitted = Zu. Then it makes a levelplot or wireplot to asses the spatial fit.

Value

plots: spatial plots for all fitted values from the mmer2 model terms in the random part.

fits A new dataset with the fitted values for all terms in the model.

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

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

Rodriguez-Alvarez, M.X, Boer, M.P., van Eeuwijk, F.A., and Eilers, P.H.C. (2017). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics (to appear). https://doi.org/10.1016/j.spasta.2017.10.003.

See Also

mmer – the core function of the package

Examples

```
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# ### mimic two fields
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
# A <- A.mat(GT)
# mix <- mmer(Yield~1,
#             random=~vs(ds(FIELD),id, Gu=A) +
#               vs(ds(FIELD),Rowf) +
#               vs(ds(FIELD),Colf) +
#                 vs(ds(FIELD),spl2D(Row,Col, at=FIELD)),
#             rcov=~vs(ds(FIELD),units),
#             data=DT2)
# summary(mix)$varcomp
#
#
# ss <- spatPlots(object=mix, by="FIELD", colfunc=NULL,
#           row="Row",range="Col", wire=FALSE, terms = c("Rowf","Colf","Row"))
```

spl2D *Two-dimensional penalised tensor-product of marginal B-Spline basis.*

Description

Auxiliary function used for modelling the spatial or environmental effect as a two-dimensional penalised tensor-product of marginal B-spline basis functions with anisotropic penalties on the basis of the spl2D approach by Lee et al. (2013) and Rodriguez-Alvarez et al. (2018). This is a modified wrapper of some functions from the SpATS package associated with the proposal described in Rodriguez-Alvarez et al. (2018). You may be interested in reading and citing the publication if using this methodology.

Usage

```
spl2D(x.coord,y.coord,at,at.levels, type="PSANOVA",
      nseg = c(10,10), pord = c(2,2), degree = c(3,3),
      nest.div = c(1,1))
```

Arguments

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

Details

The following documentation is taken from the SpATS package. Please refer to this package and associated publications if you are interested in going deeper on this technique. You may be interested in reading and citing this publication if using this methodology:

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

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

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

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

ternative method to the SAP function. In this case, the smooth bivariate surface (or spatial trend) is decomposed in five different components each of them depending on a single smoothing parameter (see Lee et al., 2013).

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

References

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

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

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

Gilmour, A.R., Cullis, B.R., and Verbyla, A.P. (1997). Accounting for Natural and Extraneous Variation in the Analysis of Field Experiments. Journal of Agricultural, Biological, and Environmental Statistics, 2, 269 - 293.

See Also

mmer, vs

Examples

```
## ============================ ##
## example to use spl2D() + vs() structure
## ============================ ##
data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
### mimic two fields
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
# A <- A.mat(GT)
# mix <- mmer(Yield~1,
#             random=~vs(ds(FIELD),id, Gu=A) +
#               vs(ds(FIELD),Rowf) +
```

```
#               vs(ds(FIELD),Colf) +
#                 vs(ds(FIELD),spl2D(Row,Col)),
#             rcov=~vs(ds(FIELD),units),
#             data=DT2)
```

---

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

---

transformConstraints *transformConstraints*

Description

transformConstraints takes a list of matrices with constraints and transforms all the non-zero values to the value desired. The purpose of this function is to make easy the transformation of initial constraints to a fixed-constraint list to be provided to a mixed model fitted with the mmer function.

Usage

```
transformConstraints(list0,value=1)
```

Arguments

list0	a list of matrices with constraints according to the rules specified in the vs function (0: not to be estimated, 1: positive, 2:unconstrained, 3:fixed).
value	value to be used to replace all the non-zero values in the constraint matrices.

Value

$res a list with the modified constraint matrices.

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function vs to know how to use transformConstraints in the mmer solver.

Examples

```
(a <- list(unsm(4), diag(4)))
transformConstraints(a, value=3)
```

---

transp *Creating color with transparency*

Description

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

Usage

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

Arguments

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

Details

No major details.

Value

If arguments are correctly specified the function returns:

$res A new color with certain grade of transparency

References

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

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

See Also

The core functions of the package mmer

Examples

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

---

uncm *unconstrained indication matrix*

Description

uncm creates a square matrix with 2's in the diagnals and off-diagonals to quickly specify an unconstrained constraint in the Gtc argument of the vs function.

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function `vs` to know how to use `uncm` in the `mmer` solver.

Examples

```
uncm(4)
```

`unsBLUP` *unsBLUP*

Description

`unsBLUP` takes a list of BLUPs from the U component of an mmer model fitted as unstructured to add the main variance BLUP and the covariance BLUPs to provide the right BLUPs for the main variances.

Usage

```
unsBLUP(blups)
```

Arguments

`blups`	a list of BLUPs from the U component of the model.

Value

$res a list of BLUPs from the U component of the model adjusting the main variance BLUPs by adding the covariance BLUPs .

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function `vs` to know how to use `unsBLUP` in the `mmer` solver.

Examples

```
data(DT_example)
DT <- DT_example

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

ans3 <- mmer(fixed=Yield~Env-1,
             random=~ vs(us(Env),Name),
             rcov=~vs(us(Env),units),
             data=DT)#, return.param = T)

u <- unsBLUP(ans3$U[1:6])
# adjusted BLUP vs non-adjusted BLUP
plot(u$`CA.2011:Name`$Yield, ans3$U$`CA.2011:Name`$Yield)
```

---

unsm *unstructured indication matrix*

Description

unsm creates a square matrix with ones in the diagonals and 2's in the off-diagonals to quickly specify an unstructured constraint in the Gtc argument of the vs function.

Usage

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

Arguments

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function `vs` to know how to use `unsm` in the `mmer` solver.

Examples

```
unsm(3)
unsm(3,2)
```

`us` *unstructured covariance structure*

Description

`us` creates an unstructured covariance structure for specific levels of the random effect.

Usage

```
us(x)
```

Arguments

`x`	vector of observations for the random effect.

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

See Also

The function `vs` to know how to use `us` in the `mmer` solver.

Examples

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

vs *variance structure specification*

Description

vs is the main function to build the variance-covariance structure for the random effects to be fitted in the mmer solver.

Usage

```
vs(..., Gu=NULL, Gt=NULL, Gtc=NULL)
```

Arguments

...	variance structure to be specified following the logic desired in the internal kronecker product. For example, if user wants to define a diagonal variance structure for the random effect 'genotypes'(g) with respect to a random effect 'environments'(e), this is:
	`var(g) = G.e @ I.g`
	being `G.e` a matrix containing the variance covariance components for g (genotypes) in each level of e (environments), `I.g` is the covariance among levels of g (genotypes; i.e. relationship matrix), and @ is the kronecker product. This would be specified in the mmer solver as:
	`random=~vs(ds(e),g)`
	One strength of sommer is the ability to specify very complex structures with as many kronecker products as desired. For example:
	`var(g) = G.e @ G.f @ G.h @ I.g`
	is equivalent to
	`random=~vs(e,f,h,g)`
	where different covariance structures can be applied to the levels of `e,f,h` (i.e. ds, us, cs, at or a combination of these). For more examples please see the vignettes 'sommer.start' available in the package.

Gu	matrix with the known variance-covariance structure for the random effect levels (i.e. relationship matrix among individuals or any other known covariance structure). If NULL, then an identity matrix is assumed.
Gt	matrix with dimensions t x t (t equal to number of traits) with initial values of the variance-covariance components for the random effect specified in the .... argument. If NULL the program will provide the initial values.
Gtc	matrix with dimensions t x t (t equal to number of traits) of constraints for the variance-covariance components for the random effect specified in the ... argument according to the following rules: `0`: not to be estimated `1`: estimated and constrained to be positive (i.e. variance component) `2`: estimated and unconstrained (can be negative or positive,i.e. covariance component)

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

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

Value

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

Author(s)

Giovanny Covarrubias-Pazaran

References

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

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

See Also

The core function of the package: mmer

Examples

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

## ============================ ##
## example to use ds() structure (DIAGONAL)
## ============================ ##
ds(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vs(ds(Year),Name),
            rcov=~ vs(ds(Year),units),
            data=DT)

## ============================ ##
## example to use at() structure (level-specific)
## ============================ ##
unique(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vs(at(Year,c("2011","2012")),Name),
            rcov=~ vs(ds(Year),units),
            data=DT)
```

```
## ============================ ##
## example to use us() structure (UNSTRUCTURED)
## ============================ ##
us(DT$Year)
mix <- mmer(Yield~Env,
            random= ~ vs(us(Year),Name),
            rcov=~ vs(ds(Year),units),
            data=DT)

## ============================ ##
## example to use cs() structure (CUSTOMIZED)
## ============================ ##
unique(DT$Year)
mm <- matrix(1,3,3); mm[1,3] <- mm[3,1] <- 0;mm #don't estimate cov 2011-2013
mix <- mmer(Yield~Env,
            random= ~ vs(cs(Year,mm),Name),
            rcov=~ vs(ds(Year),units),
            data=DT)

## ============================ ##
## example to use overlay() + vs() structure
## ============================ ##
data("DT_halfdiallel")
DT <- DT_halfdiallel
head(DT)
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)
A <- diag(7); colnames(A) <- rownames(A) <- 1:7;A # if you want to provide a covariance matrix
#### model using overlay
modh <- mmer(sugar~1,
             random=~vs(overlay(femalef,malef), Gu=A)
                    + genof,
              data=DT)

## ============================ ##
## example to use spl2D() + vs() structure
## ============================ ##
# ### mimic two fields
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# aa <- DT; bb <- DT
# aa$FIELD <- "A";bb$FIELD <- "B"
# set.seed(1234)
# aa$Yield <- aa$Yield + rnorm(length(aa$Yield),0,4)
# DT2 <- rbind(aa,bb)
# head(DT2)
#
# mix <- mmer(Yield~1,
#             random=~vs(ds(FIELD),id, Gu=A) +
#               vs(ds(FIELD),Rowf) +
```

```
#               vs(ds(FIELD),Colf) +
#                 vs(ds(FIELD),spl2D(Row,Col)),
#             rcov=~vs(ds(FIELD),units),
#             data=DT2)
```

Index

∗Topic **R package**
sommer-package, 3
∗Topic **array**
adiag1, 11
∗Topic **datasets**
DT_augment, 28
DT_btdata, 29
DT_cornhybrids, 30
DT_cpdata, 32
DT_example, 34
DT_fulldiallel, 38
DT_gryphon, 39
DT_h2, 40
DT_halfdiallel, 41
DT_legendre, 43
DT_polyploid, 44
DT_rice, 45
DT_technow, 47
DT_wheat, 48
∗Topic **models**
anova.mmer, 13
coef.mmer, 23
fitted.mmer, 59
plot.mmer, 103
predict.mmer, 104
residuals.mmer, 105
summary.mmer, 112

A.mat, *3*, 9, *61*, *85*
A_example *(*DT_example*)*, 34
A_gryphon *(*DT_gryphon*)*, 39
Ad_technow *(*DT_technow*)*, 47
adiag1, 11
Af_technow *(*DT_technow*)*, 47
anova, *4*, *13*, *65*, *72*, *89*, *97*
anova.mmer, 13
AR1, 14, *61*, *85*
ARMA, 15, *61*, *85*
at, 16, *62*, *64*, *69*, *71*, *86, 87*, *94, 95*, *118*
atcg1234, *3*, 17

bathy.colors, 19
bivariateRun, 19
build.HMM, *4*, 21, *65*, *72*, *89*, *97*

coef, *4*, *23*, *65*, *72*, *89*, *97*
coef.mmer, 23
CS, 23, *61*, *85*
cs, 24, *62*, *69*, *86*, *94*, *118*

D.mat, *3*, 25, *61*, *85*
diag, *62*, *86*
ds, 27, *62*, *64*, *69*, *71*, *86, 87*, *94*, *96*, *118*
DT_augment, 28
DT_btdata, *4*, 29, *65*, *72*, *88*, *97*
DT_cornhybrids, *4*, 30, *65*, *71*, *88*, *96*
DT_cpdata, *4*, 32, *65*, *71*, *88*, *96*
DT_example, 34
DT_expdesigns, 36
DT_fulldiallel, *4*, 38, *65*, *71*, *88*, *96*
DT_gryphon, *4*, 39, *65*, *71*, *88*, *97*
DT_h2, *4*, 40, *65*, *71*, *88*, *96*
DT_halfdiallel, *4*, 41, *65*, *71*, *88*, *96*
DT_legendre, *4*, 43, *88*, *97*
DT_polyploid, *4*, 44, *65*, *71*, *88*, *96*
DT_rice, 45
DT_technow, *4*, 47, *65*, *71*, *88*, *96*
DT_wheat, *4*, 48, *65*, *71*, *88*, *96*
DT_yatesoats, 50
DTi_cornhybrids *(*DT_cornhybrids*)*, 30

E.mat, *3*, 51, *61*, *85*
EM, 53

fcm, 56, *85*, *88*
fill.design, *4*, 58, *65*, *72*, *89*, *97*
fitted, *4*, *59*, *65*, *72*, *89*, *97*
fitted.mmer, 59
fixm, 60, *62*, *86*

GT_cornhybrids *(*DT_cornhybrids*)*, 30
GT_cpdata *(*DT_cpdata*)*, 32

GT_polyploid (DT_polyploid), 44
GT_rice (DT_rice), 45
GT_wheat (DT_wheat), 48
GTn_rice (DT_rice), 45
GWAS, *4–6*, 61, *68*
GWAS2, 68

h2.fun, *4*, *65*, *72*, 73, *89*, *97*

imputev, 75

jet.colors, 76

leg, *61*, *62*, 77, *85*, *86*, *88*, *94*, *96*
list2usmat, 78

manhattan, *4*, *65*, *72*, 79, *89*, *97*
map.plot, *4*, *65*, *72*, 80, *89*, *97*
mclapply, *20*
Md_technow (DT_technow), 47
MEMMA, 82
Mf_technow (DT_technow), 47
mmer, *3–5*, *10*, *12–16*, *18*, *20*, *22–25*, *27–29*, *31*, *33*, *34*, *38*, *40–44*, *46*, *47*, *49*, *52*, *55*, *57–61*, *76–79*, *81*, *82*, *84*, 84, *93*, *99–101*, *104–108*, *111–115*, *117–119*
mmer2, 93
MP_cpdata (DT_cpdata), 32
MP_polyploid (DT_polyploid), 44

overlay, *61*, *62*, *64*, *69*, *71*, *85*, *86*, *88*, *94*, *96*, 98

P_gryphon (DT_gryphon), 39
pedtoK, 100
pin, *3*, 101
plot, *4*, *65*, *72*, *89*, *97*, *104*
plot.mmer, 103
predict, *89*, *105*
predict.mmer, 104
print.mmer (summary.mmer), 112
print.summary.mmer (summary.mmer), 112

randef, *4*, *65*, *72*, *89*, *97*, 105
residuals, *4*, *65*, *72*, *89*, *97*, *106*
residuals.mmer, 105

simGECorMat, 106
sommer, *37*, *64*, *72*, *75*, *89*, *97*
sommer (sommer-package), 3
sommer-package, 3
spatPlots, *65*, *72*, *89*, *97*, 107
spl2D, *4*, *61*, *62*, *64*, *69*, *71*, *85*, *86*, *88*, *94*, *96*, 109
summary, *4*, *65*, *72*, *89*, *97*, *112*
summary.mmer, 112

transformConstraints, 112
transp, 113

uncm, *62*, *86*, 114
unsBLUP, 115
unsm, *62*, *86*, 116, *119*
us, *62*, *64*, *69*, *71*, *86*, *87*, *94*, *96*, 117, *118*

variogram, *4*, *65*, *72*, *97*
vs, *5*, *16*, *25*, *28*, *56*, *57*, *60*, *62–64*, *69–71*, *78*, *86–88*, *94–96*, *111*, *113–117*, 118