Package ‘synbreed’

January 13, 2012

Type Package

Title Framework for the analysis of genomic prediction data using R

Version 0.8-1

Date 2011-12-20

Author Valentin Wimmer, Theresa Albrecht, Hans-Juergen Auinger, Chris-Carolin Schoen with contributions by Larry Schaeffer, Malena Erbe, Ulrike Ober and Christian Reimer

Depends R (>= 2.10), lattice, igraph, MASS, LDheatmap, qtl, doBy, BLR, regress (>= 1.3-5), abind, kinship

Maintainer Valentin Wimmer <Valentin.Wimmer@wzw.tum.de>

Description The package was developed within the Synbreed project for synergistic plant and animal breeding (www.synbreed.tum.de). It contains a collection of function required for genomic prediction in both plant and animal breeding. This covers data processing, data visualization and data analysis. All functions are embedded within the framework of a single, unified data object. The implementation is flexible with respect to a wide range of data formats. This research was funded by the German Federal Ministry of Education and Research (BMBF) within the AgroClustEr Synbreed - Synergistic plant and animal breeding (FKZ 0315528A).

URL http://synbreed.r-forge.r-project.org/

License GPL-2

LazyLoad yes

LazyData no

ZipData no

R topics documented:

`add.individuals` *Adding new individuals to objects of class gpData*

Description

This function adds new data for individuals to an object of class `gpData` by adding new phenotypes, genotypes and pedigree.

Usage

```
add.individuals(gpData, pheno = NULL, geno = NULL,
                pedigree = NULL, covar = NULL, repl=NULL)
```

Arguments

gpData	object of class gpData
pheno	data.frame with new rows for phenotypes. For unrepeated values the ID of the Genotypes can be the rownames. Otherwise a column with name "ID" is expected.
geno	matrix with new rows for genotypic data
pedigree	data.frame with new rows for pedigree data
covar	data.frame with new rows for covar information
repl	The column of the pheno data.frame for the replicated measures. If the values are unrepeated or this column is named "repl" this argument is not needed.

Details

colnames in geno, pheno and pedigree must match with existing names in gpData object.

Value

object of class gpData with new individuals

Author(s)

Valentin Wimmer

See Also

add.markers, discard.individuals

Examples

```
# adding one new DH line to maize data
data(maize)
newDHpheno <- data.frame(Trait=1000,row.names="newDH")
# simulating genotypic data
newDHgeno <- matrix(sample(c(0,1),ncol(maize$geno),replace=TRUE),nrow=1)
rownames(newDHgeno) <- "newDH"
# new pedigree
newDHpedigree <- data.frame(ID="newDH",Par1=0,Par2=0,gener=0)
# new covar information
newDHcovar <- data.frame(family=NA,DH=1,tbv=1000,row.names="newDH")

# add individual
maize2 <- add.individuals(maize,newDHpheno,newDHgeno,newDHpedigree,newDHcovar)
summary(maize2)
```

add.markers *Adding new markers to an object of class gpData*

Description

This function adds new markers to the element geno of an object of class `gpData` and updates the marker map.

Usage

```
add.markers(gpData, geno, map)
```

Arguments

`gpData`	object of class gpData
`geno`	`matrix` with new columns
`map`	`data.frame` with columns 'chr' and 'pos' for new markers

Details

`rownames` in argument geno must match with `rownames` in the element geno object of class `gpData`.

Value

object of class `gpData` with new markers

Author(s)

Valentin Wimmer

See Also

`add.individuals`, `discard.markers`

Examples

```
# creating gpData object
# phenotypic data
pheno <- data.frame(Yield = rnorm(10,100,5), Height = rnorm(10,10,1))
rownames(pheno) <- 1:10
# genotypic data
geno <- matrix(sample(c(1,0,2,NA),size=120,replace=TRUE,
prob=c(0.6,0.2,0.1,0.1)),nrow=10)
rownames(geno) <-  1:10
# genetic map
map <- data.frame(chr=rep(1:3,each=4),pos=rep(1:12))
colnames(geno) <- rownames(map) <-  paste("M",1:12,sep="")
# as gpData object
gp <- create.gpData(pheno,geno,map)


# new data
geno2 <- matrix(c(0,0,1,1,1,2,2,1,1,2,1,2,0,2,1,1,1,2,2,2),ncol=2)
```

```
rownames(geno2) <- 1:10
map2 <- data.frame(pos=c(0.3,5),chr=c(1,2))
rownames(map2) <- colnames(geno2) <- c("M13","M14")

# adding new markers
gp2 <- add.markers(gp,geno2,map2)
summary(gp2)
```

---

cattle *Dairy cattle data*

Description

Data set contains genotypic, phenotypic, map and pedigree data of 500 bulls. All individuals are labeled with a unique ID, starting with ID1430 and ending with ID1929. Genotypic and pedigree data is based on a real cattle data set while phenotypes were built artificially. Pedigree information is available at least on parents and grandparents of the phenotyped individuals.

There are two quantitative phenotypes available. The heritabilities of these traits are 0.41 and 0.66 , estimated with a pedigree-based animal model using the data set on hand.

Genotypic data consists of 7250 biallelic SNP markers for every phenotyped individual with missing data included. SNPs are mapped across all 29 autosomes. Distances in the SNP map are given in mega bases (Mb).

Usage

```
data(cattle)
```

Format

Object of class `gpData`

Examples

```
data(cattle)
summary(cattle)
```

---

codeGeno *Recoding of genotypic data, imputing of missing values and preselection of markers*

Description

This function combines all algrorithms for processing of marker data within `synbreed` package. Raw marker data could be in any format (e.g. alleles coded as pair of observed alleles "A/T","G/C", ... , or by genotypes "AA", "BB", "AB"). Function is limited to biallelic markers with a maximum of 3 genotypes per locus. Raw data is recoded into the number of copies of the minor allele, i.e. 0, 1 and 2. Imputing of missing values can be done by random sampling from allele distribution, the `Beagle` software or family information (see details). Additional preselection of markers can be done according to the minor allele frequency and/or fraction of missing values.

Usage

```
codeGeno(gpData,impute=FALSE,impute.type=c("random","family",
         "beagle", "beagleAfterFamily","fix"), replace.value=NULL, maf=NULL,
         nmiss=NULL, label.heter="AB", keep.identical=TRUE, verbose=FALSE,
         minFam=5, showBeagleOutput=FALSE, tester=NULL)
```

Arguments

gpData	object of class gpData with arbitrary coding in element geno. Missing values have to be coded as NA.
impute	logical. Should missing value be replaced by imputing?
impute.type	character with one out of "fix", "random" , "family", "beagle", "beagleAfterFamily" (default = "random"). See details.
replace.value	numeric scalar to replace missing values in case impute.type="fix".
maf	numeric scalar. Threshold to discard markers due to the minor allle frequency (MAF). Markers with a MAF < maf are discarded, thus maf in [0,0.5]. If map in gpData is available, markers are also removed from map.
nmiss	numeric scalar. Markers with more than nmiss fraction of missing values are discarded, thus nmiss in [0,1]. If map in gpData is available, markers are also removed from map.
label.heter	This is either a scalar or vector of characters to identify heterozygous genotypes or a function which evaluates if an element of the marker matrix is the heterozygous genotype. Defining a function is useful, if number of unique heterozygous genotypes is high, i.e. if genotypes are coded by alleles. In case the heterozygous genotypes are coded like "A/T","G/C", ... label.heter="alleleCoding" can be used. Note that heterozygous values must be identified unambiguously by label.heter. Use label.heter=NULL if there are only homozygous genotypes, i.e. in DH lines, to speed up computation and to impute only 0 and 2.
keep.identical	logical. Should duplicated markers be kept?
verbose	logical. If TRUE verbose output is generated during the steps of the algorithm. This is useful to obtain numbers of discarded markers due to different criteria.
minFam	For impute.type family and beagleAfterFamily, each family should have at least minFam members with available information for a marker to impute missing values according to the family. The default is 5.
showBeagleOutput	logical. Do you like to see the output of the Beagle software package. The default is FASLE.
tester	This option is in testing mode at the moment.

Details

Coding of genotypic data is done in the following order (depending on choice of arguments not all steps are performed):

1. Discarding markers with fraction > nmiss of missing values

2. Recoding alleles from character/factor/numeric into the number of copies of the minor alleles, i.e. 0, 1 and 2. In codeGeno, in the first step heterozygous genotypes are coded as 1. From the other genotypes, the less frequent genotype is coded as 2 and the remaining genotype as 0. Note that function codeGeno will terminate with an error whenever more than three genotypes are found.

3. Replace of missing values by `replace.value` or impute missing values according to one of the following methods:

Imputing is done according to `impute.type`

"family" Suppose an observation i is missing (NA) for a marker j in family k. If marker j is fixed in family k, the imputed value will be the fixed allele. If marker j is segregating for the population k, the value is 0 with probability 0.5 and 1 with probability 0.5. To use this algorithm, family information has to be stored as variable `family` in list element `covar` of an object of class `gpData`. This column should contain a `character` or `numeric` to identify family of all genotyped individuals.

"beagle" Use Beagle Genetic Analysis Software Package (Browning and Browning 2007; 2009) to infer missing genotypes. If you use this option, please cite in publications the original papers. Beagle uses a HMM to reconstruct missing genotypes by the flanking markers. The beagle executive file `beagle.jar` (version 3.3.1) is in the directory `exec` of the package. Function `codeGeno` will create (if it does not exist) a directory `beagle` for Beagle input and output files and run Beagle with default settings. The information on marker postion is taken from element `map`. Indeed, the postion in `map$pos` must be available for all markers. By default, three genotypes 0, 1, 2 are imputed. To restrict the imputation only to homozygous genotypes, use `label.heter=NULL`.

"beagleAfterFamily" In the first step, missing genotypes are imputed according to the algorithm with `impute.type="family"`, but only for markers that are fixed within the family. Moreover, markers with a missing position (`map$pos=NA`) are imputed using the algorithm of `impute.type="family"`. In the second step, the remaining genotypes are imputed by Beagle.

"random" The missing values for a marker j are sampled from the marginal allele distribution of marker j. With 2 possible genotypes (this is only when `label.heter=NULL`), i.e. 0 and 2, values are sampled from distribution with probabilities $P(x=0)=1-p$ and $P(x=2)=p$, where p is the minor allele frequency of marker j. To use this distribution for the sampling of missing values, specify `label.heter=NULL`. In case of 3 genotypes, i.e. with heterozygous genotypes, values are sampled from distribution $P(x=0)=(1-p)^2$, $P(x=1)=p(1-p)$ and $P(x=2)=p^2$.

"fix" All missing values are imputed by `replace.value`. Note that only 0, 1 or 2 should be chosen.

4. Recoding of alleles after imputation, if necessary due to changes in allele frequencies caused by imputed alleles

5. Discarding markers with a minor allele frequency of <= `maf`

6. Discarding duplicated markers if `keep.identical=FALSE`. The first marker of a block of duplicated markers is retained.

7. Restoring original data format (`gpData`, `matrix` or `data.frame`)

Information about imputing is reported after a call of `codeGeno`.

Value

An object of class `gpData` containing the recoded marker matrix. If `maf` or `nmiss` were specified or `keep.identical=FALSE`, dimension of `geno` and `map` may be reduced due to selection of markers. The genotype which is homozygous for the minor allele is coded as 2, the other homozygous is coded as 0 and heterozygous genotype is coded as 1.

Author(s)

Hans-Juergen Auinger and Valentin Wimmer

References

S R Browning and B L Browning (2007) Rapid and accurate haplotype phasing and missing data inference for whole genome association studies using localized haplotype clustering. Am J Hum Genet 81:1084-1097

B L Browning and S R Browning (2009) A unified approach to genotype imputation and haplotype phase inference for large data sets of trios and unrelated individuals. Am J Hum Genet 84:210-22

Examples

```
# create marker data for 9 SNPs and 10 homozygous individuals
snp9 <- matrix(c(
  "AA",   "AA",   "AA",   "BB",   "AA",   "AA",   "AA",   "AA",  NA,
  "AA",   "AA",   "BB",   "BB",   "AA",   "AA",   "BB",   "AA",  NA,
  "AA",   "AA",   "AB",   "BB",   "AB",   "AA",   "AA",   "BB",  NA,
  "AA",   "AA",   "BB",   "BB",   "AA",   "AA",   "AA",   "AA",  NA,
  "AA",   "AA",   "BB",   "AB",   "AA",   "BB",   "BB",   "BB",  "AB",
  "AA",   "AA",   "BB",   "BB",   "AA",   NA,     "BB",   "AA",  NA,
  "AB",   "AA",   "BB",   "BB",   "BB",   "AA",   "BB",   "BB",  NA,
  "AA",   "AA",    NA,    "BB",    NA,    "AA",   "AA",   "AA",  "AA",
  "AA",    NA,     NA,    "BB",   "BB",   "BB",   "BB",   "BB",  "AA",
  "AA",    NA,    "AA",   "BB",   "BB",   "BB",   "AA",   "AA",  NA),
  ncol=9,byrow=TRUE)

# set names for markers and individuals
colnames(snp9) <- paste("SNP",1:9,sep="")
rownames(snp9) <- paste("ID",1:10+100,sep="")

# create object of class 'gpData'
gp <- create.gpData(geno=snp9)

# code genotypic data
gp.coded <- codeGeno(gp,impute=TRUE,impute.type="random")


# example with heterogeneous stock mice
data(mice)
summary(mice)
# heterozygous values must be labeled  (may run some seconds)
mice.coded <- codeGeno(mice,label.heter=function(x) substr(x,1,1)!=substr(x,3,3))


# example with maize data and imputing by family
data(maize)
# first only recode alleles
maize.coded <- codeGeno(maize,label.heter=NULL)

# set 200 random chosen values to NA
set.seed(123)
ind1 <- sample(1:nrow(maize.coded $geno),200)
ind2 <- sample(1:ncol(maize.coded $geno),200)
original <- maize.coded$geno[cbind(ind1,ind2)]

maize.coded$geno[cbind(ind1,ind2)] <- NA
# imputing of missing values by family structure
maize.imputed <- codeGeno( maize.coded,impute=TRUE,impute.type="family",label.heter=NULL)
```

```
# compare in a cross table
imputed <- maize.imputed$geno[cbind(ind1,ind2)]
(t1 <- table(original,imputed) )
# sum of correct replacements
sum(diag(t1))/sum(t1)

# compare with random imputation
maize.random <- codeGeno(maize.coded,impute=TRUE,impute.type="random",label.heter=NULL)
imputed2 <- maize.random$geno[cbind(ind1,ind2)]
(t2 <- table(original,imputed2) )
# sum of correct replacements
sum(diag(t2))/sum(t2)
```

---

create.gpData *Create genomic prediction data object*

Description

This function combines all raw data sources in a single, unified data object of class gpData. This is a list with elements for phenotypic, genotypic, pedigree and further covariate data. Moreover, the marker map is an element. Any element is optional.

Usage

```
create.gpData(pheno = NULL, geno = NULL, map = NULL, pedigree = NULL,
              family = NULL, covar = NULL, reorderMap = TRUE,
              map.unit = "cM", repeated  = NULL, modCovar = NULL)
```

Arguments

pheno	data.frame with individuals organized in rows and traits organized in columns. For unrepeated measures unique rownames should identify individuals. In other cases the first column identify individuals. In this case a value for argument repeated is necessary. This can be also used for unrepeated trait, but than a column with a unique value in the whole column is needed.
geno	matrix with individuals organized in rows and markers organized in columns. Genotypes could be coded arbitrarily. Missing values should be coded as NA. Unique rownames identify individuals and unique colnames markers. If no rownames are available, they are taken from element pheno (if available and if dimensions match). If no colnames are used, the rownames of map are used if dimension matches.
map	data.frame with one row for each marker and two columns (named chr and pos). First columns gives the chromosome and second column the position on the chromosome in centimorgan or the physical distance relative to the reference sequence in basepairs. Unique rownames give the marker names which should match with marker names in geno. Note that order and number of markers must not be identical to the order in geno. If this is the case, gaps in map are filled with NA to ensure that the order is identical to geno.
pedigree	Object of class pedigree.

family — data.frame assigning individuals to families with names of individuals in rownames This information could be used for replacing of missing values with function codeGeno.

covar — data.frame with further covariates for all individuals that either appear in pheno, geno or pedigree$ID, e.g. sex or age. rownames must be specified to identify individuals. Typically this element is not specified by the user.

reorderMap — logical. Should markers in geno and map be reordered by chromosome number and position within chromosome according to map (default = TRUE)?

map.unit — Character. Unit of position in map, i.e. 'cM' for genetic distance or 'bp' for physical distance.

repeated — This column is used to identify the replications of the phenotypic values. The unique values become the names of the third dimension of the pheno object in the gpData

modCovar — vector with colnames which identify columns with covariables in pheno

Details

The class gpData is designed to provide a unified framework for data related to genomic prediction analysis. Every data source can be omitted. In this case, the corresponding argument must be NULL. By default (argument reorderMap), markers in geno are ordered by their position in map. Individuals are ordered in alphabetical order.

In an object of class gpData different individuals may occur in pheno, geno and pedigree are possible. In this case the id in covar comprises all individuals that either appear in pheno, geno and pedigree. Two additional columns in covar named phenotyped and genotyped identify individuals that appear in the corresponding object.

Value

Object of class gpData which is a list with the following items

covar — list with information on individuals

pheno — array (individuals x traits x replications) with phenotypic data ordered by rownames(pheno)

geno — matrix marker matrix containing genotypic data. Columns (marker) are in the same order as in map. Rows ordered by rownames(geno)

pedigree — object of class pedigree

map — data.frame with columns 'chr' and 'pos' and markers sorted by 'pos' within 'chr'

phenoCovars — array with phenotypic covariates

info — list with additional information on data (coding of data, unit in map)

Note

In case of missing row names or column names in one item, information is substituted from other elements (assuming the same order of individuals/markers) and a warning is given.

Author(s)

Valentin Wimmer

Hans-Juergen Auinger

See Also

codeGeno, summary.gpData, gpData2data.frame

Examples

```
set.seed(123)
# 9 plants with 2 phenotypes
n <- 9  # only for n > 6
pheno <- data.frame(Yield = rnorm(n,200,5), Height=rnorm(n,100,1))
rownames(pheno) <- letters[1:n]

# marker matrix
geno <- matrix(sample(c("AA","AB","BB",NA),size=n*12,replace=TRUE,
prob=c(0.6,0.2,0.1,0.1)),nrow=n)
rownames(geno) <-  letters[n:1]
colnames(geno) <- paste("M",1:12,sep="")

# genetic map
# one SNP is not mapped (M5)
map <- data.frame(chr=rep(1:3,each=4),pos=rep(1:12))
map  <- map[-5,]
rownames(map) <- paste("M",c(1:4,6:12),sep="")

# simulate pedigree
ped <- simul.pedigree(3,c(3,3,n-6))

# combine in one object
gp <- create.gpData(pheno,geno,map,ped)
summary(gp)


# 9 plants with 2 phenotypes , 3 reps
n <- 9  #
pheno <- data.frame(ID = rep(letters[1:n],3), rep = rep(1:3,each=n),
                    Yield = rnorm(3*n,200,5), Height=rnorm(3*n,100,1))

# combine in one object
gp2 <- create.gpData(pheno,geno,map,repeated="rep")
summary(gp2)
```

---

create.pedigree *Create pedigree object*

Description

This function can be used to create a pedigree object.

Usage

```
create.pedigree(ID, Par1, Par2, gener=NULL,sex=NULL,add.ancestors=FALSE)
```

Arguments

`ID`	vector of unique IDs identifying e.g. each genotype.
`Par1`	vector of IDs identifying parent 1 (with animals: sire)
`Par2`	vector of IDs identifying parent 2 (with animals: dam)
`gener`	vector identifying the generation. If `NULL` gener will be 0 for unknown parents and max(gener(Par1),gener(Par2))+1 ofor generations 1,... .
`sex`	vector identifying the sex (female=0 and male=1).
`add.ancestors`	`logical`. Add ancestors which do not occur in `ID` to the pedigree.

Details

Missing values for pedigree should be coded with 0 for numeric ID or `NA` for character ID.

Value

An object of class `pedigree`. Column `gener` starts from 0 and pedigree is sorted by generation.

Author(s)

Valentin Wimmer

See Also

`plot.pedigree`

Examples

```
# example with 9 individuals
id <- 1:9
par1 <- c(0,0,0,0,1,1,1,4,7)
par2 <- c(0,0,0,0,2,3,2,5,8)
gener <- c(0,0,0,0,1,1,1,2,3)

# create pedigree object (using argument gener)
ped <- create.pedigree(id,par1,par2,gener)
ped
plot(ped)

# create pedigree object (without using argument gener)
ped2 <- create.pedigree(id,par1,par2)
ped2
```

`crossVal` *Cross validation of different prediction models*

Description

Function for the application of the cross validation procedure on prediction models with fixed and random effects. Covariance matrices must be committed to the function and variance components can be committed or reestimated with ASReml or the BLR function.

Usage

```
crossVal(gpData, trait=1, cov.matrix = NULL,  k = 2, Rep = 1, Seed = NULL,
         sampling = c("random", "within popStruc", "across popStruc","commit"),
         TS=NULL,ES=NULL, varComp = NULL, popStruc = NULL, VC.est = c("commit",
         "ASReml","BRR","BL"),verbose=FALSE,...)
```

Arguments

`gpData`	Object of class `gpData`
`trait`	`numeric` or `character`. The name or number of the trait to be selected out of `gpData`
`cov.matrix`	`list` including covariance matrices for the random effects. Size and order of rows and columns should be equal to rownames of `y`. If no covariance is given, an identity matrix and genotypes are used for a marker regression.
`k`	Number of folds for k-fold cross validation, thus `k` should be in [2,`nrow(y)`].
`Rep`	Number of replications.
`Seed`	Number for `set.seed()` to make results reproducable.
`sampling`	Different sampling strategies can be `"random"`, `"within popStruc"` or `"across popStruc"`. If sampling is `"commit"` test sets have to specified in TS.
`TS`	A (optional) list of vectors with IDs for the test set in each fold within a list of replications, same layout as output for `id.TS` .
`ES`	A (optional) list of IDs for the estimation set in each fold within each replication.
`varComp`	A `vector` of variance components for the random effects, which has to be specified if `VC.est="commit"`. The first variance components should be the same order as the given covariance matrices, the last given variance component is for the residuals.
`popStruc`	Vector of length `nrow(y)` assigning individuals to a population structure. If no popStruc is defined, family information of `gpData` is used.
`VC.est`	Should variance components be reestimated with `"ASReml"` or with a Bayesian approach `"BRR"` and `"BL"` within the estimation set of each fold in the cross validation? If `VC.est="commit"`, the variance components have to be defined in `varComp`. For `ASReml`, ASReml software have to be installed on the system.
`verbose`	Logical. Whether output shows replications and folds.
`...`	further arguments to be used by the genomic prediction models, i.e. prior values and MCMC options for the `BLR` function (see `BLR`).

Details

In cross validation the data set is splitted into an estimation (ES) and a test set (TS). The effects are estimated with the ES to predict observations in the TS. For sampling into ES and TS, k-fold cross validation is applied, where the data set is splitted into k subsets and k-1 comprising the ES and 1 is the TS, repeated for each subset.

To account for the family structure, `sampling` can be defined as:

random Does not account for family structure, random sampling within the complete data set

within popStruc Accounts for within population structure information, e.g. each family is splitted into k subsets

across popStruc Accounts for across population structure information, e.g. ES and TS contains a set of complete families

The following mixed model equation is used for `VC.est="commit"`:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Zu} + \mathbf{e}$$

with

$$\mathbf{u} \sim \mathbf{N}(\mathbf{0}, \mathbf{G}\sigma^2_\mathbf{u})$$

$$\begin{pmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} + \mathbf{G}^{-1}\frac{\sigma^2_\mathbf{e}}{\sigma^2_\mathbf{u}} \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{Z}'\mathbf{y} \end{pmatrix}$$

Value

A object of class `list` with following items:

`bu`	Estimated fixed and random effects of each fold within each replication.
`n.DS`	Size of the data set (ES+TS) in each fold.
`y.TS`	Predicted values of all test sets within each replication.
`n.TS`	Size of the test set in each fold.
`id.TS`	List of IDs of each test sets within a list of each replication.
`PredAbi`	Predictive ability of each fold within each replication calculated as $r(y_{TS}, \hat{y}_{TS})$.
`rankCor`	Spearman's rank correlation of each fold within each replication calculated between y_{TS} and $\hat{y}_{TS}$.
`mse`	Mean squared error of each fold within each replication calculated between y_{TS} and $\hat{y}_{TS}$.
`bias`	Regression coefficients of a regression of the observed values on the predicted values in the TS. A regression coefficient < 1 implies extreme high (low) values of the predicted values over- (under-) estimated the observed values, and vice versa for a regression coefficient > 1.
`m10`	Mean of observed values for the 10% best predicted of each replication. The k test sets are pooled within each replication.
`k`	Number of folds
`Rep`	Replications
`sampling`	Sampling method
`Seed`	Seed for `set.seed()`
`rep.seed`	Calculated seeds for each replication
`nr.ranEff`	Number of random effects
`VC.est.method`	Method for the variance components (`committed` or `reestimated with ASReml/BRR/BL`)

Author(s)

Theresa Albrecht

References

Legarra A, Robert-Granie C, Manfredi E, Elsen J (2008) Performance of genomic selection in mice. Genetics 180:611-618

Luan T, Wooliams JA, Lien, S, Kent M, Svendsen M, Meuwissen THE (2009) The accuracy of genomic selection in Norwegian red cattle assessed by cross-validation. Genetics 183:1119-1126

Mosier CI (1951) I. Problems and design of cross-validation 1. Educ Psychol Measurement 11:5-11

Crossa J, de los Campos G, Perez P, Gianola D, Burgueno J, et al. (2010) Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers, Genetics 186:713-724

See Also

summary.cvData

Examples

```
# loading the maize data set
data(maize)
maize2 <- codeGeno(maize)
rad <- kin(maize2,ret="realized")
diag(rad) <- diag(rad)+0.00001 # to avoid singularities
# cross validation
cv.maize  <- crossVal(maize2,cov.matrix=list(rad),k=5,Rep=1,
            Seed=123,sampling="random",varComp=c(26.5282,48.5785),VC.est="commit")
cv.maize2 <- crossVal(maize2,k=5,Rep=1,
             Seed=123,sampling="random",varComp=c(0.0704447,48.5785),VC.est="commit")
# comparing results, both are equal!
cv.maize$PredAbi
cv.maize2$PredAbi
summary(cv.maize)
summary(cv.maize2)
```

---

discard.markers *Subsets for objects of class gpData*

Description

Both functions could be used to get subsets of objects of class `gpData`. Use function `discard.markers` to discard markers from an object of class `gpData`. Note that markers will also be removed from `map`, if available. Use function `discard.individuals` to discard individuals from the object of class `gpData`. Note that individuals will also be removed from `covar`, `pheno` and `pedigree`.

Usage

```
discard.markers(gpData, which)
discard.individuals(gpData, which, keepPedigree = FALSE)
```

Arguments

gpData	object of class gpData
which	character vector either identifying the colnames of markers in geno to discard (function discard.markers) or the rownames of individuals to discard (function discard.individuals).
keepPedigree	logical. Should the individual kept in the pedigree?

Value

Object of class gpData

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

See Also

create.gpData

Examples

```
# example data
set.seed(311)
pheno <- data.frame(Yield = rnorm(10,200,5),Height=rnorm(10,100,1))
rownames(pheno) <- letters[1:10]
geno <- matrix(sample(c("A","A/B","B",NA),size=120,replace=TRUE,
prob=c(0.6,0.2,0.1,0.1)),nrow=10)
rownames(geno) <-  letters[1:10]
colnames(geno) <- paste("M",1:12,sep="")
# one SNP is not mapped (M5)
map <- data.frame(chr=rep(1:3,each=4),pos=rep(1:12))
map  <- map[-5,]
rownames(map) <- paste("M",c(1:4,6:12),sep="")
gp <- create.gpData(pheno=pheno,geno=geno,map=map)
summary(gp)

# remove unmapped SNP M5 (which has no postion in the map)
gp2 <- discard.markers(gp,"M5")
summary(gp2)

# discard genotypes with missing values in the marker matrix
gp3 <- discard.individuals(gp,names(which(rowSums(is.na(gp$geno))>0)))
summary(gp3)
```

---

gpData2cross *Conversion between objects of class 'cross' and 'gpData'*

Description

Conversion of object class gpData to object class cross (F2 intercross) of package qtl and vice versa. Function codeGeno is applied in cross2gpData, if not done before.

Usage

```
gpData2cross(gpData,...)
cross2gpData(cross)
```

Arguments

gpData	object of class gpData with non-empty elements for pheno, geno and map.
cross	object of class cross.
...	further arguments for function codeGeno. Only used in gpData2cross.

Details

In cross, genotypic data is splitted into chromosomes while in gpData genotypic data comprises all chromosomes. Note that coding of genotypic data differs between classes. In gpData, genotypic data is coded as the number of copies of the minor allele, i.e. 0, 1 and 2. Thus, function codeGeno should be applied to gpData before using gpData2cross to ensure correct coding. In cross, coding for F2 intercross is: AA = 1, AB = 2, BB = 3. When using gpData2cross or cross2gpData, resulting genotypic data has correct format.

Value

Object of class cross or gpData for function gpData2cross or cross2gpData, respectively.

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

References

Broman, K. W. and Churchill, S. S. (2003). R/qtl: Qtl mapping in experimental crosses. Bioinformatics, (19):889-890.

See Also

create.gpData, read.cross , codeGeno

Examples

```
# from gpData to cross
data(maize)
maizeC <- codeGeno(maize)
maize.cross <- gpData2cross(maizeC)
# descriptive statistics
summary(maize.cross)
plot(maize.cross)

# search for QTL on chr 1
maize.cross <- calc.genoprob(maize.cross, step=2.5)
result <- scanone(maize.cross, pheno.col=1,method="em")
# display of LOD curve
plot(result)


# from cross to gpData
```

```
data(fake.f2)
fake.f2.gpData <- cross2gpData(fake.f2)
summary(fake.f2.gpData)
```

`gpData2data.frame` *Merge of phenotypic and genotypic data*

Description

Create a `data.frame` out of phenotypic and genotypic data in object of class `gpData` by merging datasets using the common id. The shared data set could either include individuals with phenotypes and genotypes (default) or additional unphenotyped or ungenotyped individuals. In the latter cases, the missing observations are filled by `NA`.

Usage

```
gpData2data.frame(gpData,trait=1,onlyPheno=FALSE,all.pheno=FALSE,
                  all.geno=FALSE,repl=NULL,...)
```

Arguments

`gpData`	object of class `gpData`
`trait`	`numeric` or `character`. A vector with the names or numbers of the trait that should be extracted of pheno. Default is `1`.
`onlyPheno`	scalar `logical`. Only return phenotypic data.
`all.pheno`	scalar `logical`. Include all individuals with genotypes in the `data.frame` and fill the genotypic data with `NA`.
`all.geno`	scalar `logical`. Include all individuals with phenotypes in the `data.frame` and fill the phenotypic data with `NA`.
`repl`	`character` or `numeric`. A vector which contains names or numbers of replication that should be drawn out of the phenoypic values and covariates. Default is `NULL`, that means all values are taken.
`...`	further arguments to be used in function `reshape`. The argument `times` could be useful to rename levels of grouping variable.

Details

Argument `all.geno` can be used to predict the genetic value of individuals without phenotypic records using the `BLR` package. Here, the genetic value of individuals with `NA` as phenotype is predicted by the marker profile.

For multiple measures, phenotypic data in object `gpData` is arranged with replicates in an `array`. With `gpData2data.frame` this could be reshaped to "long" format with multiple observations in one column. In this case, one column for the phenotype and 2 additional columns for the `id` and the levels of the grouping variable are added.

Value

A `data.frame` with the individuals names in the first column, the phenotypes in the next column(s) and the marker genotypes in subsequent columns.

Author(s)

Valentin Wimmer Hans-Juergen Auinger

See Also

`create.gpData`, `reshape`

Examples

```
# example data with unrepeated observations
set.seed(311)

# simulating genotypic and phenotypic data
pheno <- data.frame(Yield = rnorm(12,100,5),Height=rnorm(12,100,1))
rownames(pheno) <- letters[4:15]
geno <- matrix(sample(c("A","A/B","B",NA),size=120,replace=TRUE,
prob=c(0.6,0.2,0.1,0.1)),nrow=10)
rownames(geno) <-  letters[1:10]
colnames(geno) <- paste("M",1:12,sep="")
# different subset of individuals in pheno and geno

# create 'gpData' object
gp <- create.gpData(pheno=pheno,geno=geno)
summary(gp)

# as data.frame with individuals with genotypes and phenotypes
gpData2data.frame(gp,1:2)
# as data.frame with all individuals with phenotypes
gpData2data.frame(gp,1:2,all.pheno=TRUE)
# as data.frame with all individuals with genotypes
gpData2data.frame(gp,1:2,all.geno=TRUE)

# example with repeated observations
set.seed(311)

# simulating genotypic and phenotypic data
pheno <- data.frame(ID = letters[1:10], Trait = c(rnorm(10,1,2),rnorm(10,2,0.2),
                    rbeta(10,2,4)), repl = rep(1:3, each=10))
geno <- matrix(rep(c(1,0,2),10),nrow=10)
colnames(geno) <- c("M1","M2","M3")
rownames(geno) <-  letters[1:10]

# create 'gpData' object
gp <- create.gpData(pheno=pheno,geno=geno, repeated="repl")

# reshape of phenotypic data and merge of genotypic data,
# levels of grouping variable loc are named "a", "b" and "c"
gpData2data.frame(gp,onlyPheno=FALSE,times=letters[1:3])
```

gpMod *Genomic predictions models for objects of class gpData*

Description

This function can be used to fit genomic prediction models to an object of class gpData. The possible models are Best Linear Unbiased Prediction (BLUP) using a pedigree-based or marker-based genetic relationship matrix and Bayesian Lasso (BL) or Bayesian Ridge regression (BRR). BLUP models are fitted using the implementation of the regress package (Clifford and McCullagh, 2011). The Bayesian regression models are fitted using the implementaion of the BLR package (de los Campos and Perez, 2010). The covariance structure in the BLUP model is defined by an object of class relationshipMatrix. The training set for the model fit consists of all individuals with phenotypes and genotypes. All data is restricted to individuals from the training set.

Usage

```
gpMod(gpData, model = c("BLUP","BL","BRR"), kin = NULL, trait = 1,
      repl = NULL, markerEffects = FALSE, fixed = NULL, random = NULL, ...)
```

Arguments

gpData	object of class gpData
model	character. The model to use. "BL" and "BRR" are at the moment only for two-stage analysis. So you have to fit a model for phenotypic covariats first.
kin	object of class relationshipMatrix. Kinship structure to be used in the BLUP model or as additional polygenic effect u in the Bayesian regression models.
trait	numeric or character. A vector with names or numbers of the traits to fit the model
repl	numeric or character. A vector with names or numbers of the repeated values of gpData$pheno to fit the model
markerEffects	If fitted a BLUP model the effects of markers can be calculated, i.e. Random Regression BLUP. In this case, the matrix kin is calculated as the transposed crossproduct of the genotype matrix.
fixed	A formula for fixed effects. The details of model specification are the same as for lm exept that you do not have to specify the left side of the model.
random	A formula for random effects of the model. Specifies the matrices to include in the covariance structure. Each term is either a symmetric matrix, or a factor. Independent Gaussian random effects are included by passing the corresponding block factor. For mor details see regress.
...	further arguments to be used by the genomic prediction models, i.e. prior values and MCMC options for the BLR function (see BLR).

Details

Only a subset of the individuals - the training set - is used for the model fit. This contains all individuals with phenotypes and genotypes. If kin does not match the dimension of the training set (if, e.g. ancestors are included), the respective rows and columns from the trainings set are choosen. Marker effects for model=BLUP are extracted from the corresponding G-BLUP model using their functional relationship. In this case, fit reports the G-BLUP model

Value

Object of class gpMod which is a list of

fit	The model fit returned by the genomic prediction method
model	The model type, see 'Arguments'
y	The phenotypic records for the individuals in the training set
g	The predicted genetic values for the individuals in the training set
m	Predicted SNP effects (if available)
kin	Matrix kin

Note

The verbose output of the BLR function is written to a file BLRout.txt to prevent the screen output from overload.

Author(s)

Valentin Wimmer, Hans-Juergen Auinger, and Theresa Albrecht

References

Clifford D, McCullagh P (2011). regress: Gaussian Linear Models with Linear Covariance Structure. R package version 1.3-4, URL http://www.csiro.au.

Gustavo de los Campos and Paulino Perez Rodriguez, (2010). BLR: Bayesian Linear Regression. R package version 1.2. http://CRAN.R-project.org/package=BLR

See Also

kin, crossVal

Examples

```
data(maize)
maizeC <- codeGeno(maize)

# pedigree-based (expected) kinship matrix
K <- kin(maizeC,ret="kin",DH=maize$covar$DH)

# marker-based (realized) relationship matrix
# divide by an additional factor 2
# because for testcross prediction the kinship of DH lines is used
U <- kin(maizeC,ret="realized")/2

# BLUP models
# P-BLUP
mod1 <- gpMod(maizeC,model="BLUP",kin=K)
# G-BLUP
mod2 <- gpMod(maizeC,model="BLUP",kin=U)

# Bayesian Lasso
prior <- list(varE=list(df=3,S=35),lambda = list(shape=0.52,rate=1e-4,value=20,type='random'))
## Not run: mod3 <- gpMod(maizeC,model="BL",prior=prior,nIter=6000,burnIn=1000,thin=5)
```

```
summary(mod1)
summary(mod2)
## Not run: summary(mod3)
```

`kin` *Relatedness based on pedigree or marker data*

Description

This function implements different measures of relatedness between individuals in a object of class `gpData`: (1) Expected relatedness based on pedigree and (2) realized relatedness based on marker data. See 'Details'. The function uses as first argument an object of class `gpData`. An argument `ret` controls the type of relatedness coefficient.

Usage

```
kin(gpData,ret=c("add","kin","dom","gam","realized","realizedAB","sm","sm-smin"),
    DH=NULL)
```

Arguments

`gpData`	object of class `gpData`
`ret`	`character`. The type of relationship matrix to be returned. See 'Details'.
`DH`	`logical` vector of length n. `TRUE` or 1 if individual is a DH line and `FALSE` or 0 otherwise. Only used for pedigree based relatedness coefficients

Details

Pedigree based relatedness (return arguments `"add"`, `"kin"`, `"dom"`, and `"gam"`)

Function `kin` provides different types of measures for pedigree based relatedness. A element `pedigree` must be available in the object of class `gpData`. In all cases, the first step is to build the gametic relationship. The gametic relationship is of order $2n$ as each individual has two alleles (e.g. for individual A $A1$ and $A2$). The gametic relationship is defined as the matrix of probabilities that two genes are identical by descent (IBD). Note that the diagonal elements of the gametic relationship matrix are 1. The off-diagonals of individuals with unknown pedigree are 0. If `ret="gam"` is specified, the gametic relationship matrix constructed by pedigree is returned.

The gametic relationship matrix can be used to set up other types of relationship matrices. If `ret="add"`, the additive numerator relationship matrix is returned. The additive relationship of individuals A (alleles $A1, A2$) and B (alleles $B1, B2$) is given by the entries of the gametic relationship matrix

$$0.5 \cdot \left[(A1, B1) + (A1, B2) + (A2, B1) + (A2, B2)\right],$$

where $(A1, B1)$ denotes the element [A1,B1] in the gametic relationship matrix. If `ret="kin"`, the kinship matrix is returned which is half of the additive relationship matrix.

If `ret="dom"`, the dominance relationship matrix is returned. The dominance relationship matrix between individuals A ($A1, A2$) and B ($B1, B2$) in case of no inbreeding is given by

$$\left[(A1, B1) \cdot (A2, B2) + (A1, B2) \cdot (A2, B1)\right],$$

where $(A1, C1)$ denotes the element [A1,C1] in the gametic relationship matrix.

Marker based relatedness (return arguments `"realized"`,`"realizedAB"`, `"sm"`, and `"sm-smin"`)

Function `kin` provides different types of measures for marker based relatedness. A element `geno` must be available in the object of class `gpData`. Furthermore, genotypes must be code by the number of copies of the minor allele, i.e. function `codeGeno` must be applied in advance.

If `ret="realized"`, the realized relatedness between individuals is computed according to the formulas in Habier et al. (2007) or vanRaden (2008)

$$U = \frac{ZZ'}{2\sum p_i(1-p_i)}$$

where $Z = W - P$ and W is the marker matrix and P contains the allele frequencies multiplied by 2 and p_i is the allele frequency of marker i.

If `ret="realizedAB"`, the realized relatedness between individuals is computed according to the formula in Astle and Balding (2009)

$$U = \frac{1}{M}\sum \frac{(w_i - 2p_i)(w_i - 2p_i)'}{2p_i(1-p_i)}$$

where w_i is the marker genotype, p_i is the allele frequency at marker locus i, and M is the number of marker loci.

If `ret="sm"`, the realized relatedness between individuals is computed according to the simple matching coefficient (Reif et al. 2005). The simple matching coefficient counts the number of shared alleles across loci. It could only be applied to homozygous inbred lines, i.e. only genotypes 0 and 2. To account for loci that are alike in state but not identical by descent (IBD), Hayes and Goddard (2008) correct the simple matching coefficient by the minimum of observed simple matching coefficients

$$\frac{s - s_{min}}{1 - s_{min}}$$

where s is the matrix of simple matching coefficients. This formula is used with argument `ret="sm-smin"`.

Value

An object of class "relationshipMatrix".

Author(s)

Valentin Wimmer and Theresa Albrecht

References

Habier D, Fernando R, Dekkers J (2007). The Impact of Genetic Relationship information on Genome-Assisted Breeding Values. Genetics, 177, 2389 – 2397.

vanRaden, P. (2008). Efficient methods to compute genomic predictions. Journal of Dairy Science, 91:4414 – 4423.

Astle, W., and D.J. Balding (2009). Population Structure and Cryptic Relatedness in Genetic Association Studies. Statistical Science, 24(4), 451 – 471.

Rogers, J., 1972 Measures of genetic similarity and genetic distance. In Studies in genetics VII, volume 7213. Univ. of Texas, Austin

Hayes, B. J., and M. E. Goddard. 2008. Technical note: Prediction of breeding values using marker derived relationship matrices. J. Anim. Sci. 86

See Also

plot.relationshipMatrix

Examples

```
#========================
# (1) pedigree based relatedness
#========================

data(maize)
K <- kin(maize,ret="kin")
plot(K)

#========================
# (2) marker based relatedness
#========================

data(maize)
U <- kin(codeGeno(maize),ret="realized")
plot(U)


### Example for Legarra et al. (2009), J. Dairy Sci. 92: p. 4660
id <- 1:17
par1 <- c(0,0,0,0,0,0,0,0,1,3,5,7,9,11,4,13,13)
par2 <- c(0,0,0,0,0,0,0,0,2,4,6,8,10,12,11,15,14)
ped <- create.pedigree(id,par1,par2)
gp <- create.gpData(pedigree=ped)

# additive relationship
A <- kin(gp,ret="add")
# dominance relationship
D <- kin(gp,ret="dom")
```

LDDist *LD versus distance Plot*

Description

Visualization of pairwise LD versus distance. A single plot is generated for every chromosome.

Usage

```
LDDist(LDdf,chr=NULL,type="p",breaks=NULL,file=NULL,n=NULL,...)
```

Arguments

LDdf	object of class `LDdf` which is the output of function `pairwiseLD`
chr	`numeric`. Return value is a plot for each chromosome in `chr`.
type	Character string to specify the type of plot. Use `"p"` for a scatterplot, `"bars"` for stacked bars or `"nls"` for scatterplot together with nonlinear regression curve according to Hill and Weir (1988).

breaks	`list` containing breaks for stacked bars (optional, only for `type="bars"`). Components are `dist` with breaks for distance on x-axis and `r2` for breaks on for r2 on y-axis. By default, 5 equal spaced categories for dist and r2 are used.
file	`character`. path to a file were plot is saved as `pdf` (optional).
n	`numeric`. Number of observations. Only required for `type="nls"`.
...	Further arguments for `plot`

References

For nonlinear regression curve: Hill WG, Weir BS (1988) Variances and covariances of squared linkage disequilibria in finite populations. Theor Popul Biol 33:54-78.

See Also

`pairwiseLD`, `LDMap`

Examples

```
# maize data example
data(maize)
maizeC <- codeGeno(maize)

# LD for chr 1
maizeLD <- pairwiseLD(maizeC,chr=1,type="data.frame")
# scatterplot
LDDist(maizeLD,type="p",pch=19,col=hsv(alpha=0.1,v=0))

# stacked bars  with default categories
LDDist(maizeLD,type="bars")

# stacked bars  with user-defined categories
LDDist(maizeLD,type="bars",breaks=list(dist=c(0,10,20,40,60,180),
r2=c(1,0.6,0.4,0.3,0.1,0)))
```

`LDMap` *LD Heatmap*

Description

Visualization of pairwise LD in LD heatmap for each linkage group (chromosome) using the `LDheatmap` package.

Usage

```
LDMap(LDmat,gpData,chr=NULL,file=NULL,...)
```

Arguments

LDmat	Object of class LDmat generated by function pairwiseLD.
gpData	Object of class gpData that was used to infer LDmat.
chr	numeric. Return value is a plot for each chromosome in chr.
file	Optionally a path to a file were plot is saved as pdf.
...	Further arguments that could be passed to function LDheatmap.

See Also

pairwiseLD, LDheatmap, LDDist

Examples

```
data(maize)
maizeC <- codeGeno(maize)

# LD for chr 1
maizeLD <- pairwiseLD(maizeC,chr=1,type="matrix")
LDMap(maizeLD,maizeC)
```

---

maize *Simulated maize data*

Description

This is a simulated dataset of a maize breeding program. Data comprises 1250 doubled haploid (DH) lines that were genotyped with 1117 polymorphic SNP markers and phenotyped in a testcross with a single tester for one quantitative trait. Markers are distributed along all 10 chromosomes of maize. Pedigree information starts with basis population and is available up to 15 generations. The 1250 lines belong to 25 full sib families with 50 individuals in each family. In the simulation of true breeding values (TBV), 1000 biallelic quantitative trait loci (QTL) with equal and additive (no dominance or epistasis) effects were generated. True breeding values for individuals were calculated according to

$$tbv = \sum_{k=1}^{1000} QTL_k$$

where QTL_k is the effect of the k-th QTL. Phenotypic values were simulated according to

$$y_i = tbv_i + \epsilon_i$$

where $\epsilon \sim N(0, I\sigma^2)$. The value for σ^2 was chosen in a way that a given plot heritability of $h^2 = 0.197$ is realized. Note that true breeding values for 1250 phenotyped lines are stored as tbv in covar of gpData object. Reported phenotypic values of lines are adjusted values testcross means for yield [dt/ha] evaluated in 3 locations.

Usage

```
data(maize)
```

Format

Object of class gpData

Examples

```
data(maize)
summary(maize)
```

---

manhattanPlot *Manhattan plot for SNP effects*

Description

Plot of SNP effect along the chromosome

Usage

```
manhattanPlot(b, gpData = NULL, colored = FALSE, add = FALSE,
              pch = 19, ylab = NULL, ...)
```

Arguments

b	numeric vector of effects to plot
gpData	object of class gpData with map position
colored	Is the plot colored or in grey. The default is "grey"
add	If TRUE the plot is added to an existing plot. The default is FALSE.
pch	a vector of plotting characters or symbols: see points. The default is an open circle.
ylab	a title for the y axis: see title.
...	further arguments for function plot

Author(s)

Valentin Wimmer

Examples

```
data(mice)
# plot only random noise
b <- rexp(ncol(mice$geno),3)
manhattanPlot(b,mice)
```

mice *Heterogenous stock mice population*

Description

Data set comprises public available data of 2527 (1293 males and 1234 females) heterogenous stock mice derived from eight inbred strains (A/J, AKR/J, BALBc/J, CBA/J, C3H/HeJ, C57BL/6J, DBA/2J and LP/J) followed by 50 generations of pseudorandom mating. All individuals are labeled with a unique ID, starting with `A048005080`. For all individuals, family, sex (females=0, males=1), month of birth (1-12), birthyear, coat color, cage density and litter is available and stored in `covar`.

The measured traits are described in Solberg et al. (2006). Here, the body weight at age of 6 weeks [g] and growth slope between 6 and 10 weeks age [g/day] are available. The heritabilities of these traits are reported as 0.74 and 0.30, respectively (Valdar et al, 2006b). Data was taken from http://mus.well.ox.ac.uk/GSCAN/HS_PHENOTYPES/Weight.txt.

Genotypic data consists of 12545 biallelic SNP markers and is available for 1940 individuals. Raw genotypic data from http://mus.well.ox.ac.uk/GSCAN/HS_GENOTYPES/All is given in the `Ped-File Format` with two columns for each marker. Both alleles were combined to a single genotype for each marker in `mice` data. The SNPs are mapped in a sex-averaged genetic map with distances given in centimorgan (Shifman et al. (2006)). SNPs are mapped across all 19 autosomes and X-chromosome where distances between adjacent markers vary form 0 to 3 cM.

Usage

```
data(mice)
```

Format

Object of class `gpData`

Source

Welcome Trust Centre for Human Genetics, Oxford University, data available from http://gscan.well.ox.ac.uk

References

Shifman S, Bell JT, Copley RR, Taylor MS, Williams RW, et al. (2006) A High-Resolution Single Nucleotide Polymorphism Genetic Map of the Mouse Genome. PLoS Biol 4(12)

Solberg L.C. et al, A protocol for high-throughput phenotyping, suitable for quantitative trait analysis in mice. Mamm. Genome 17, 129-146 (2006)

Valdar W, Solberg LC, Gauguier D, Burnett S, Klenerman P, Cookson WO, Taylor MS, Rawlins JN, Mott R, Flint J. (2006a) Genome-wide genetic association of complex traits in heterogeneous stock mice. Nat Genet. 2006 Aug;38(8):879-87.

Valdar W, Solberg LC, Gauguier D, Cookson WO, Rawlins NJ, Mott R, Flint J.(2006b) Genetic and environmental effects on complex traits in mice. Genetics. 2006 Aug 3;

Examples

```
data(mice)
summary(mice)
```

MME *Mixed Model Equations*

Description

Set up Mixed Model Equations for given design matrix, i.e. variance components for random effects must be known.

Usage

```
MME(X, Z, GI, RI, y)
```

Arguments

X	Design matrix for fixed effects
Z	Design matrix for random effects
GI	Inverse of (estimated) variance-covariance matrix of random (genetic) effects multplied by the ratio of residual to genetic variance
RI	Inverse of (estimated) variance-covariance matrix of residuals (without multiplying with a constant, i.e. σ_e^2)
y	Vector of phenotypic records

Details

The Mixed Model is given by

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{Z}\mathbf{u} + \mathbf{e}$$

with $\mathbf{u} \sim \mathbf{N}(\mathbf{0}, \mathbf{G})$ and $\mathbf{e} \sim \mathbf{N}(\mathbf{0}, \mathbf{R})$. Solutions for fixed effects b and random effects u are obtained by solving the mixed model equations

$$\begin{pmatrix} \mathbf{X'R^{-1}X} & \mathbf{X'R^{-1}Z} \\ \mathbf{Z'R^{-1}X} & \mathbf{Z'R^{-1}Z} + \mathbf{G^{-1}} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \mathbf{X'R^{-1}y} \\ \mathbf{Z'R^{-1}y} \end{pmatrix}$$

Matrix on left hand side of mixed model equation is denoted by LHS and RHS of MME is denoted as RHS. Generalized Inverse of LHS equals prediction error variance matrix. Square root of diagonal values multiplied with σ_e^2 equals standard error of prediction. Note that variance components for fixed and random effects are not estimated by this function but have to be specified by the user, i.e. G^{-1} must be multiplied with shrinkage factor $\frac{\sigma_e^2}{\sigma_g^2}$.

Value

A list with the following arguments

b	Estimations for fixed effects vector
u	Predictions for random effects vector
LHS	left hand side of MME
RHS	right hand side of MME
C	Generalized inverse of LHS. This is the prediction error variance matrix
SEP	Standard error of preciction for fixed and random effects
SST	Sum of Squares Total
SSR	Sum of Squares due to Regression
residuals	Vector of residuals

Author(s)

Valentin Wimmer

References

Henderson, C. R. 1984. Applications of Linear Models in Animal Breeding. Univ. of Guelph, Guelph, ON, Canada.

See Also

`regress`, `crossVal`

Examples

```
data(maize)

# realized kinship matrix
U <- kin(codeGeno(maize),ret="realized")/2

# solution with gpMod
m <- gpMod(maize,kin=U,model="BLUP")

# solution with MME
diag(U) <- diag(U) + 0.000001
GI <- solve(U)*(m$fit$sigma[2]/ m$fit$sigma[1])
y <- maize$pheno[,1,]
n <- length(y)
X <- matrix(1,ncol=1,nrow=n)
mme <- MME(y=y,Z=diag(n),GI=GI,X=X,RI=diag(n))

# comparison
head(m$fit$predicted[,1]-m$fit$beta)
head(mme$u)
```

`pairwiseLD` *Pairwise LD between markers*

Description

Compute pairwise LD between markers measured as r^2 using an object of class `gpData`. For the general case, a gateway to the software PLINK (Purcell et al. 2007) is established to estimate the LD. A within-R solution is available for marker data with only 2 genotypes, i.e. homozgous inbred lines. Return value is an object of class `LDdf` which is a `data.frame` with one row per marker pair or an object of class `LDMat` which is a `matrix` with all marker pairs. Additional, the euclidian distance between position of markers can be computed ant returned.

Usage

```
pairwiseLD(gpData, chr = NULL, type = c("data.frame", "matrix"),
           use.plink=FALSE, ld.threshold=0, rm.unmapped = TRUE)
```

Arguments

gpData	object of class gpData with elements geno and map
chr	numeric scalar or vector. Return value is a list with pairwise LD of all markers for each chromosome in chr.
type	character. Specifies the type of return value.
use.plink	logical. Should the software PLINK be used for the computation?
ld.threshold	numeric. Threshold for the LD. Only pairwise LD>ld.threshold is reported when PLINK is used. This argument is can only be used for type="data.frame".
rm.unmapped	logical. Remove markers with unknown postion in map before using PLINK?

Details

The function write.plink is called to prepare the input files and the script for PLINK. The executive PLINK file plink.exe must be available (e.g. in the working directory). The function pairwiseLD calls PLINK and reads the results. The evaluation is performed separate for every chromosome. The measure for LD is r^2. This is defined as

$$D = p_{AB} - p_A p_B$$

and

$$r^2 = \frac{D^2}{p_A p_B p_a p_b}$$

where p_{AB} is defined as the observed probability of haplotype AB, $p_A = 1 - p_a$ and $p_B = 1 - p_b$ the observed probabilities of alleles A and B. If the number of markers is higt, a threshold for the LD can be used. In this case, only pairwise LD above the threshold is reported (argument --ld-window-r2 in PLINK).

Value

For type="data.frame" an object of class LDdf with one element for each chromosome. Each element is a data.frame with columns marker1, marker2, r2 and distance for all $p(p-1)/2$ marker pairs.

For type="matrix" an object of class LDmat with one element for each chromosome. Each element is a list of 2: a $p \times p$ matrix with pairwise LD and a $p \times p$ matrix with pairwise distances.

Author(s)

Valentin Wimmer

References

Hill WG, Robertson A (1968). Linkage Disequilibrium in Finite Populations. Theoretical and Applied Genetics, 6(38), 226 - 231.

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

See Also

LDDist, LDMap

Examples

```
data(maize)
maizeC <- codeGeno(maize)
maizeLD <- pairwiseLD(maizeC,chr=1,type="data.frame")
```

---

plot.pedigree *Visualization of pedigree*

Description

A function to visualize pedigree structure by a graph. Each genotype is represented as vertex and direct offsprings are linked by an edge.

Usage

```
## S3 method for class 'pedigree'
plot(x, effect = NULL, ...)
```

Arguments

`x`	object of class `pedigree` or object of class `gpData` with element `pedigree`
`effect`	vector of length `nrow(pedigree)` with effect to for the x axis
`...`	Other arguments for function `igraph.plotting`

Details

The pedigree is structured top-bottom. In the first line the first generation is printed. Links over more than one generation are possible as well as genotypes with only one (known) parent. Usually, no structure in one generation is plotted. If an `effect` is given, the genotypes are ordered by this effect and a labeled axis is plotted at the bottom.

Value

A named graph visualizing the pedigree structure

Note

This function uses the plotting method for graphs in the libraray `igraph`

Author(s)

Valentin Wimmer

See Also

`create.pedigree`, `simul.pedigree`

Examples

```
# example with 9 individuals
id <- 1:9
par1 <- c(0,0,0,0,1,1,1,4,7)
par2 <- c(0,0,0,0,2,3,2,5,8)
gener <- c(0,0,0,0,1,1,1,2,3)

# create pedigree object
ped <- create.pedigree(id,par1,par2,gener)
plot(ped)
```

---

plot.relationshipMatrix

Heatmap for relationship Matrix

Description

Visualization for objects of class `relationshipMatrix`.

Usage

```
## S3 method for class 'relationshipMatrix'
plot(x, ...)
```

Arguments

`x`	Object of class `relationshipMatrix`
`...`	further graphical arguments passed to function `levelplot` in package `lattice`. To create equal colorkeys for two heatmaps, use `at=seq(from,to,length=9)`.

Author(s)

Valentin Wimmer

Examples

```
# small pedigree
ped <- simul.pedigree(gener=4,7)
gp <- create.gpData(pedigree=ped)
A <- kin(gp,ret="add")
plot(A)

# big pedigree
data(maize)
K <- kin(maize,ret="kin")
U <- kin(codeGeno(maize),ret="realized")/2
# equal colorkeys
plot(K,at=seq(0,2,length=9))
plot(U,at=seq(0,2,length=9))
```

plotGenMap *Plot marker map*

Description

A function to visualize low and high-density marker maps.

Usage

```
plotGenMap(map, dense = FALSE, nMarker = TRUE, bw=1, centr=NULL,...)
```

Arguments

map	object of class gpData with object map or a data.frame with columns 'chr' (specifying the chromosome of the marker) and 'pos' (position of the marker within chromosome measured with genetic or physical distances)
dense	logical. Should density visualization for high-density genetic maps be used?
nMarker	logical. Print number of markers for each chromosome?
bw	numeric. Bandwith to use for dense=TRUE to control the resolution.
centr	numeric vector. Position for the centromeres in the same order as chromosomes in map. If "maize", centromere positions of maize in Mbp are used.
...	further graphical arguments for function plot

Details

In the low density plot, the unique position of markers are plotted as horizontal lines. In the high-density plot, the distribution of the markers is visualized as a heatmap of density estimation together with a color key. In this case, the number of markers within a interval of equal bandwith bw is counted. The high density plot is typically useful if number of makers exceeds 200 on average per chromosome.

Value

Plot of the marker positions within each chromosome. One chromosome is displayed from the first to the last marker.

Author(s)

Valentin Wimmer

See Also

create.gpData

Examples

```
# low density plot
data(maize)
plotGenMap(maize)

# high density plot
data(mice)
plotGenMap(mice,dense=TRUE,nMarker=FALSE)
```

`plotNeighbourLD` *Plot neighbour linkage disequilibrium*

Description

A function to visualize LD between adjacent markers.

Usage

```
plotNeighbourLD(LD, map, dense=FALSE, nMarker = TRUE, centr=NULL, ...)
```

Arguments

`LD`	object of class `LDmat`, i.e the output of function `pairwiseLD` using argument `type="matrix"`.
`map`	object of class `gpData` with object `map` or a `data.frame` with columns 'chr' (specifying the chromosome of the marker) and 'pos' (position of the marker within chromosome measured with genetic or physical distances)
`dense`	`logical`. Should density visualization for high-density genetic maps be used?
`nMarker`	`logical`. Print number of markers for each chromosome?
`centr`	`numeric` vector. Position for the centromeres in the same order as chromosomes in `map`. If `"maize"`, centromere positions of maize in Mbp are used.
`...`	further graphical arguments for function `plot`

Details

The graph is similar to `plotGenMap` with the option `dense=TRUE`, but here the LD between adjacent markers is plotted along the chromosomes.

Value

Plot of neighbour LD along each chromosome. One chromosome is displayed from the first to the last marker.

Author(s)

Theresa Albrecht

See Also

`plotGenMap`, `pairwiseLD`

Examples

```
data(maize)
maize2 <-codeGeno(maize)
LD <- pairwiseLD(maize2,chr=1:10,type="matrix")
plotNeighbourLD(LD,maize2,nMarker=FALSE)
```

`predict.gpMod` *Prediction for genomic prediction models.*

Description

S3 `predict` method for objects of class `gpMod`. A genomic prediction model is used to predict the genetic performance for e.g. unphenotyped individuals using an object of class `gpMod` estimated by a training set.

Usage

```
## S3 method for class 'gpMod'
predict(object,newdata,...)
```

Arguments

`object`	object of class `gpMod` which is the model used for the prediction.
`newdata`	for `model="BL"` and `"BRR"` an object of class `gpData` with the marker data of the unphenotyped individuals. For `model="BLUP"` a `character` vector with the names of the individuals to predict. If `newdata=NULL`, the genetic performances of the individuals for the training set are returned.
`...`	not used

Details

For models, `model="RR"` and `"BL"` the prediction for the unphenotyped individuals is given by

$$\hat{g}' = \hat{\mu} + W'\hat{m}$$

with the estimates taken from the `gpMod` object. For the prediction using `model="BLUP"`, the full relationship matrix including individuals of the training set and the prediction set must be specified in the `gpMod`. This model is used to predict the unphenotyped individuals of the prediction set by solving the corresponding mixed model equations using the variance components of the fit in `gpMod`.

Value

a named vector with the predicted genetic values for all individuals in `newdata`.

Author(s)

Valentin Wimmer

References

Henderson C (1977) Best linear unbiased prediction of breeding values not in the model for records. Journal of Dairy Science 60:783-787

Henderson CR (1984). Applications of linear models in animal breeding. University of Guelph.

See Also

gpMod

Examples

```
# Example from Henderson (1977)
dat <- data.frame(y=c(132,147,156,172),time=c(1,2,1,2),animal=c(1,2,3,4))
ped <- create.pedigree(ID=c(6,5,1,2,3,4),Par1=c(0,0,5,5,1,6),Par2=c(0,0,0,0,6,2))
gp <- create.gpData(pheno=dat,pedigree=ped)
A <- kin(gp,ret="add")

# assuming h2=sigma2u/(sigma2u+sigma2)=0.5
# no REML fit possible due to the limited number of observations
y <- c(132,147,156,172)
names(y) <- paste(1:4)
mod1 <- list(fit=list(sigma=c(1,1)),kin=A,model="BLUP",y=y,m=NULL)
class(mod1) <- "gpMod"
predict(mod1,c("5","6"))

# prediction 'by hand'
X <- matrix(1,ncol=1,nrow=4)
Z <- diag(6)[-c(1,2),]
AI <- solve(A)
RI <- diag(4)

res <- MME(X,Z,AI,RI,y)
res$b + res$u[1:2]

# prediction of genetic performance of the last 50 individuals in the maize data set
data(maize)
maizeC <- codeGeno(maize)
U <- kin(maizeC,ret="realized")
maizeC2 <- discard.individuals(maizeC,rownames(maizeC$pheno)[1201:1250])
modU <- gpMod(maizeC2,model="BLUP",kin=U)
predict(modU,rownames(maizeC$pheno)[1201:1250])
```

---

simul.pedigree *Simulation of pedigree structure*

Description

This function could be used to simulate a pedigree for a given number of generations and individuals. Function uses random mating within generations. Fully inbred may be generated optionally using selfing.

Usage

```
simul.pedigree(generations = 2, ids = 4, animals=FALSE,familySize=1)
```

Arguments

generations	integer. Number of generations to simulate

`ids`	`integer` or vector of integers. Number of genotypes in each generation. If length equal one, the same number will be replicated and used for each generation.
`animals`	`logical`. Should a pedigree for animals be simulated? See details.
`familySize`	`numeric`. Number of individuals in each full-sib family in the last generation.

Details

If `animals=FALSE` the parents for the current generation will be randomly chosen out of the genotypes in the last generation. If Par1 = Par2, an inbreed is generated. If `animal=TRUE` each ID is either sire or dam. Each ID is progeny of one sire and one dam.

Value

An object of class `pedigree` with N=sum(ids) genotypes.

Author(s)

Valentin Wimmer

See Also

`simul.phenotype`, `create.pedigree`, `plot.pedigree`

Examples

```
# example for plants
ped <- simul.pedigree(gener=4,ids=c(3,5,8,8))
plot(ped)
#example for animals
peda <- simul.pedigree(gener=4,ids=c(3,5,8,8),animals=TRUE)
plot(peda)
```

`simul.phenotype` *Simulation of a field trial with single trait*

Description

Simulate observations from a field trial using an animal model. The field trial consists of multiple locations and randomized complete block design within locations. A single quantitative trait is simulated according to the model `Trait ~ id(A) + block + loc + e`.

Usage

```
simul.phenotype(pedigree = NULL, A = NULL, mu = 100, vc = NULL,
                Nloc = 1, Nrepl = 1)
```

Arguments

`pedigree`	object of class "pedigree"
`A`	object of class "relationshipMatrix"
`mu`	`numeric`; Overall mean of the trait.
`vc`	list containing the variance components. `vc` consists of elements `sigma2e`, `sigma2a`, `sigma2l`, `sigma2b` with the variance components of the residual, the additive genetic effect, the location effect and the block effect.
`Nloc`	`numeric`. Number of locations in the field trial.
`Nrepl`	`Numeric`. Number of complete blocks within location.

Details

Either `pedigree` or `A` must be specified. If `pedigree` is given, pedigree information is used to set up numerator relationship matrix with function `kinship`. If unrelated individuals should be used for simulation, use identity matrix for `A`. True breeding values for N individuals is simulated according to following distribution

$$tbv \sim N(0, \mathbf{A}\sigma^2_{\mathbf{a}})$$

Observations are simulated according to

$$y \sim N(mu + tbv + block + loc, \sigma^2_e)$$

If now location or block effects should appear, use `sigma2l=0` and/or `sigma2b=0`.

Value

A `data.frame` with containing the simulated values for trait and the following variables

`ID`	Factor identifying the individuals. Names are extracted from `pedigree` or rownames of `A`
`Loc`	Factor for Location
`Block`	Factor for Block within Location
`Trait`	trait observations
`TBV`	Simulated values for true breeding values of individuals

Result is sorted for locations within individuals.

Author(s)

Valentin Wimmer

See Also

`simul.pedigree`

Examples

```
ped <- simul.pedigree(gener=5)
varcom <- list(sigma2e=25,sigma2a=36,sigma2l=9,sigma2b=4)
# field trial with 3 locations and 2 blocks within locations
data.simul <- simul.phenotype(ped,mu=10,vc=varcom,Nloc=3,Nrepl=2)
head(data.simul)
# analysis of variance
anova(lm(Trait~ID+Loc+Loc:Block,data=data.simul))
```

summary.cvData — *Summarizing options and results of the cross validation procedure*

Description

summary method for class "cvData"

Usage

```
## S3 method for class 'cvData'
summary(object,...)
```

Arguments

object	object of class "cvData"
...	not used

Author(s)

Theresa Albrecht

See Also

crossVal

summary.gpData — *Summary for class gpData*

Description

S3 summary method for objects of class gpData

Usage

```
## S3 method for class 'gpData'
summary(object,...)
```

Arguments

object	object of class gpMod
...	not used

Author(s)

Valentin Wimmer

Examples

```
data(maize)
summary(maize)
```

summary.gpMod *Summary for class gpMod*

Description

S3 summary method for objects of class gpMod

Usage

```
## S3 method for class 'gpMod'
summary(object,...)
```

Arguments

`object`	object of class gpMod
`...`	not used

See Also

gpMod

Examples

```
data(maize)
maizeC <- codeGeno(maize)
# marker-based (realized) relationship matrix
U <- kin(maizeC,ret="realized")/2

# BLUP model
mod <- gpMod(maizeC,model="BLUP",kin=U)
summary(mod)
```

summary.pedigree *Summarizing pedigree information*

Description

summary method for class "pedigree"

Usage

```
## S3 method for class 'pedigree'
summary(object,...)
```

Arguments

`object`	object of class "pedigree"
`...`	not used

Author(s)

Valentin Wimmer

Examples

```
# plant pedigree
ped <- simul.pedigree(gener=4,7)
summary(ped)

# animal pedigree
ped <- simul.pedigree(gener=4,7,animals=TRUE)
summary(ped)
```

`summary.relationshipMatrix`

Summarizing relationship matrices

Description

`summary` method for class "relationshipMatrix"

Usage

```
## S3 method for class 'relationshipMatrix'
summary(object,...)
```

Arguments

`object`	object of class "relationshipMatrix"
`...`	not used

Author(s)

Valentin Wimmer

Examples

```
data(maize)
U <- kin(codeGeno(maize),ret="realized")
summary(U)
```

summaryGenMap — *Summarizing marker map information*

Description

This function could be used to summarize information from a marker map. Return value is a `data.frame` with one row for each chromosome and one row summarizing all chromosomes.

Usage

```
summaryGenMap(map)
```

Arguments

`map`	`data.frame` with columns `chr` and `pos` or a `gpData` object with element `map`

Details

Summary statistics of differences are based on euclidian distances between markers with position in `map`, i.e. `pos!=NA`.

Value

A `data.frame` with one row for each chromosome and columns

`noM`	number of markers
`range`	range of positions, i.e. difference between first and last marker
`avDist`	avarage distance of markers
`maxDist`	maximum distance of markers
`minDist`	minimum distance of markers

Author(s)

Valentin Wimmer

See Also

`create.gpData`

Examples

```
data(maize)
summaryGenMap(maize)
```

write.beagle — *Prepare genotypic data for Beagle*

Description

Create input file for Beagle software (Browning and Browning 2009) from a gpData object. This function is created for usage within function codeGeno for imputing missing values.

Usage

```
write.beagle(gp, wdir = getwd(), prefix)
```

Arguments

gp	gpData object with elements geno and map
wdir	character. Directory for Beagle input files
prefix	character. Prefix for Beagle input files.

Details

The Beagle software must be used chromosomewise. Consequently, gp should contain only data from one chromosome (use discard.markers, see Examples).

Value

Create files [prefix]ingput.bgl with genotypic data in Beagle input format and [prefix]marker.txt with marker information used by Beagle.

Author(s)

Valentin Wimmer

References

B L Browning and S R Browning (2009) A unified approach to genotype imputation and haplotype phase inference for large data sets of trios and unrelated individuals. Am J Hum Genet 84:210-22

See Also

codeGeno

Examples

```
map <- data.frame(chr=c(1,1,1,1,1,2,2,2,2),pos=1:9)
geno <-  matrix(sample(c(0,1,2,NA),size=10*9,replace=TRUE),nrow=10,ncol=9)
colnames(geno) <- rownames(map) <- paste("SNP",1:9,sep="")
rownames(geno) <- paste("ID",1:10+100,sep="")

gp <- create.gpData(geno=geno,map=map)
gp1 <- discard.markers(gp,rownames(map[map$chr!=1,]))
## Not run: write.beagle(gp1,prefix="test")
```

write.plink — *Prepare data for PLINK*

Description

Create input files and corresponding script for PLINK (Purcell et al. 2007) to compute the pairwise LD.

Usage

```
write.plink(gp, wdir = getwd(), prefix = paste(substitute(gp)),
            ld.threshold = 0, type = c("data.frame", "matrix"))
```

Arguments

gp	gpData object with elements geno and map
wdir	character. Directory for PLINK input files
prefix	character. Prefix for PLINK input files.
ld.threshold	numeric. Threshold for the LD used in PLINK.
type	character. Specifies the type of return value for PLINK.

Value

no value returned. Files prefix.map, prefix.ped and prefixPlinkScript.txt are created in the working directory

Author(s)

Valentin Wimmer

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

See Also

pairwiseLD

Examples

```
## Not run: write.plink(maize,type="data.frame")
```

`write.relationshipMatrix`
Writing relationshipMatrix in table format

Description

This function could be used to write an object of class "relationshipMatrix" in the table format used by other software, i.e. WOMBAT or ASReml. The table has three columns, the row, the column and the entry of the (inverse) relationshipMatrix.

Usage

```
write.relationshipMatrix(relationshipMatrix, file = NULL,
                  sorting=c("WOMBAT","ASReml"),
                  type=c("ginv", "inv", "none"), digits = 10)
```

Arguments

`relationshipMatrix`	Object of class "relationshipMatrix"
`file`	Path were the output should be written . If `NULL` the result is returned in R.
`sorting`	Type of sorting. Use "WOMBAT" for 'row-wise' sorting of the table and "ASReml" for 'column-wise' sorting.
`type`	A character string indicating which form of `relationshipMatrix` should be returned. One of "ginv" (Moore-Penrose generalized inverse), "inv" (inverse), or "none" (no inverse).
`digits`	Numeric. The result is rounded to `digits`.

Details

Note that "WOMBAT" only uses the generalized inverse relationship matrix and expects a file with the name "ranef.gin", where 'ranef' is the name of the random effect with option 'GIN' in the 'MODEL' part of the parameter file. For ASREML, either the relationship could be saved as "*.grm" or its generalized inverse as "*.giv".

Author(s)

Valentin Wimmer

References

Meyer, K. (2006) WOMBAT - A tool for mixed model analyses in quantitative genetics by REML, J. Zhejinag Uni SCIENCE B 8: 815-821.

Gilmour, A., Cullis B., Welham S., and Thompson R. (2000) ASREML. program user manual. NSW Agriculture, Orange Agricultural Institute, Forest Road, Orange, Australia .

Examples

```
# example with 9 individuals
id <- 1:9
par1 <- c(0,0,0,0,1,1,1,4,7)
par2 <- c(0,0,0,0,2,3,2,5,8)
gener <- c(0,0,0,0,1,1,1,2,3)
ped <- create.pedigree(id,par1,par2,gener)
gp <- create.gpData(pedigree=ped)

A <- kin(ped,ret="add")
write.relationshipMatrix(A,type="ginv")
```

[.relationshipMatrix *Extract or replace part of relationship matrix*

Description

Extract or replace part of an object of class relationshipMatrix.

Usage

```
## S3 method for class 'relationshipMatrix'
x[...]
```

Arguments

x	object of class "relationshipMatrix"
...	indices

Examples

```
data(maize)
U <- kin(codeGeno(maize),ret="realized")
U[1:3,1:3]
```

Index