Package 'synbreed'

March 16, 2018

Type Package

Title Framework for the Analysis of Genomic Prediction Data using R

Version 0.12-9

Date 2018-03-08

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

Depends R (>= 2.14)

Imports abind, BGLR, doBy, doParallel, foreach, igraph, lattice, qtl, regress (>= 1.3-8), MASS, methods, LDheatmap

Suggests synbreedData (>= 1.5)

Maintainer Hans-Juergen Auinger <auinger@tum.de>

Description A collection of functions required for genomic prediction which were developed within the Synbreed project for synergistic plant and animal breeding (<http://www.synbreed.tum.de>). This covers data processing, data visualization, and 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 in plant and animal breeding. 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-3

LazyLoad yes

LazyData no

ZipData no

Repository CRAN

Repository/R-Forge/Project synbreed

Repository/R-Forge/Revision 722

Repository/R-Forge/DateTimeStamp 2018-03-15 07:17:43

Date/Publication 2018-03-16 05:01:36 UTC

NeedsCompilation no

R topics documented:

add.gpData . . . 3
add.individuals . . . 4
add.markers . . . 5
add.pedigree . . . 7
codeGeno . . . 8
create.gpData . . . 13
create.pedigree . . . 16
crossVal . . . 17
discard.individuals . . . 20
discard.markers . . . 21
gpData2cross . . . 23
gpData2data.frame . . . 24
gpMod . . . 26
kin . . . 29
LDDist . . . 32
LDMap . . . 34
manhattanPlot . . . 35
MME . . . 36
pairwiseLD . . . 38
plot.LDdf . . . 40
plot.LDmat . . . 42
plot.pedigree . . . 43
plot.relationshipMatrix . . . 44
plotGenMap . . . 45
plotNeighbourLD . . . 47
predict.gpMod . . . 48
read.vcf2list . . . 50
read.vcf2matrix . . . 51
simul.pedigree . . . 52
simul.phenotype . . . 53
summary.cvData . . . 54
summary.gpData . . . 55
summary.gpMod . . . 56
summary.LDdf . . . 56
summary.pedigree . . . 57
summary.relationshipMatrix . . . 58
summaryGenMap . . . 59
write.beagle . . . 60
write.plink . . . 61
write.relationshipMatrix . . . 62
write.vcf . . . 63
[.GenMap . . . 64
[.relationshipMatrix . . . 65

Index **66**

add.gpData *Join two* gpData *objects*

Description

Function for joining two gpData objects

Usage

```
add.gpData(gpData1, gpData2)
```

Arguments

gpData1	A gpData object with at least elements geno and map
gpData2	Second gpData object with at least elements geno and map

Details

The function writes a vcf-file. The format of the output is "GT". Other formats are not supported.

Value

It is returned a gpData object, which contains gpData1 and gpData2

Author(s)

Hans-Juergen Auinger

See Also

create.gpData codeGeno

Examples

```
## Not run: add.gpData(maize, maize)
```

add.individuals *Add new individuals to objects of class gpData*

Description

This function extends 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 to be updated
pheno	data.frame with new rows for phenotypes with rownames indicating individuals. For repeated values the ID should be stored in a column with name "ID".
geno	matrix with new rows for genotypic data with rownames indicating individuals
pedigree	data.frame with new rows for pedigree data
covar	data.frame with new rows for covar information with rownames indicating individuals
repl	The column of the pheno data.frame for the replicated measures. If the values are not repeated or this column is named "repl" this argument is not needed.

Details

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

Value

object of class gpData with new individuals

Author(s)

Valentin Wimmer

See Also

add.markers, discard.individuals

Examples

```
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)

#new phenotypic data
newPheno <- data.frame(Yield=200,Height=100,row.names="newLine")
# simulating genotypic data
newGeno <- matrix(sample(c("A","A/B","B"),ncol(gp$geno),replace=TRUE),nrow=1)
rownames(newGeno) <- "newLine"
# new pedigree
newPedigree <- create.pedigree(ID="newLine",Par1=0,Par2=0,gener=0)

gp2 <- add.individuals(gp,pheno=newPheno,geno=newGeno,pedigree=newPedigree)


## Not run:
# add one new DH line to maize data
library(synbreedData)
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 <-  create.pedigree(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)

## End(Not run)
```

`add.markers` *Add 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 to be updated
geno	matrix with new columns
map	data.frame with columns 'chr' and 'pos' for new markers

Details

rownames in argument geno must match 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)
summary(gp)
```

`add.pedigree` *Merge* `pedigree` *object*

Description

This function can be used to add some pedigree information to a existing `pedigree` object.

Usage

```
add.pedigree(ped, IDadd, add.ancestors = FALSE)
```

Arguments

`ped`	`pedigree` object
`IDadd`	`pedigree` object
`add.ancestors`	`logical`. Add ancestors which do not occur in `ID` to the pedigree.

Details

Missing values for parents in the 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)

Hans-Juergen Auinger

See Also

`plot.pedigree`, `create.pedigree`

Examples

```
# example with 9 individuals
id <- paste("ID", 1:9, sep="0")
par1 <- paste("ID", c("","","","",1,1,1,4,7), sep="0")
par2 <- paste("ID", c("","","","",2,3,2,5,8), sep="0")
ped1 <- create.pedigree(id,par1,par2,unknown="ID0")

# create 2nd pedigree object
Id <- paste("ID", 10:16, sep="")
Par1 <- paste("ID", c("","",1,1,6,7,7), sep="0")
Par2 <- paste("ID", c("","",10,"08","09",11,14), sep="")
ped2 <- create.pedigree(Id,Par1,Par2,unknown=c("ID0", "ID"))
ped2

ped <- add.pedigree(ped1, ped2)
ped
```

---

codeGeno *Recode genotypic data, imputation of missing values and preselection of markers*

Description

This function combines all algorithms for processing of marker data within synbreed package. Raw marker data is a matrix with elements of arbitrary format (e.g. alleles coded as pair of observed alleles "A/T","G/C", ... , or by genotypes "AA", "BB", "AB"). The function is limited to biallelic markers with a maximum of 3 genotypes per locus. Raw data is recoded into the number of copies of a reference allele, i.e. 0, 1 and 2. Imputation 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 carried out according to the minor allele frequency and/or fraction of missing values.

Usage

```
codeGeno(gpData,impute=FALSE,
       impute.type=c("random","family","beagle","beagleAfterFamily","beagleNoRand",
                     "beagleAfterFamilyNoRand","fix"),
         replace.value=NULL, maf=NULL, nmiss=NULL, label.heter="alleleCoding",
       reference.allele="minor", keep.list=NULL, keep.identical=TRUE, verbose=FALSE,
       minFam=5, showBeagleOutput=FALSE, tester=NULL, print.report=FALSE, check=FALSE,
         ploidy=2, cores=1)
```

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" , "beagleAfterFamilyNoRand", "beagleAfterFamilyNoRand" (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 allele 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 returning TRUE if an element of the marker matrix is the heterozygous genotype. Defining a function is useful, if number of unique heterozygous genotypes is large, i.e. if genotypes are coded by alleles. If the heterozygous genotype is coded like "A/T","G/C", ..., "AG", "CG", ..., "T:C", "G:A", ... or "G|T", "A|C", ... then label.heter="alleleCoding" can be used (This is the default). 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 restrict imputation to values 0 and 2.

reference.allele — Define the reference allele which is used for the coding. Default is "minor", i.e. data is coded by the number of copies of the minor allele. Alternatively, reference.allele can specify a single character defining the reference allele for all markers, or a vector defining marker-specific reference alleles (using the same order as of the markers in gpData). In case you have already a gpObject with info$codeGeno == TRUE, and like only to use higher maf or remove duplicated markers, you can use the option "keep", than the coding of the original object is kept.

keep.list — A vector with the names of markers, which should be kept during the process of coding and filtering.

keep.identical — logical. Should duplicated markers be kept? NOTE: From a set of identical markers (with respect to the non-missing alleles) the one with the smallest number of missing values is kept. For those with an identical number of missing values, the first one is kept and all others are removed.

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. Would you like to see the output of the Beagle software package? The default is FALSE.

tester — This option is in testing mode at the moment.

print.report	logical. Should a file SNPreport.txt be generated containing further information on SNPs. This includes SNP name, original coding of major and minor allele, MAF and number of imputed values.
check	This option has as default FALSE. If something seems to be wrong with the coding, with the option check=TRUE the function tries to catch the error.
ploidy	numeric. Here you can specify the number homologous alleles. For this option you need a coding with A and B, e.g. for tetraploids "AAAA", "AAAB", "AABB", "ABBB" and "BBBB". In the geno element of your gpData-object will be than the dosage of the minor allele. The argument label.heter is ignored for ploidy >2
cores	numeric. Here you can specify the number of cores you like to use.

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.

2.1 Discarding duplicated markers if keep.identical=FALSE before starting of the imputing step. From identical marker based on pairwise complete oberservations one is discarded randomly. For getting identical results use the function set.seed() before code.geno().

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

Imputing is done according to impute.type

"family" This option is only suitable for homozygous individuals (such as doubled-haploid lines) structured in families. 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 of 0.5 and 2 with probability of 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 version 4.1 (21Jan17.6cc) (Browning and Browning 2007; 2013; 2016) to infer missing genotypes is used. This software is a java program, so that you have to install java (>=1.7) and make it available at your computer. If you use the beagle option, please cite the original papers in publications. Beagle uses a HMM to reconstruct missing genotypes by the flanking markers. Function codeGeno will create a directory beagle for Beagle input and output files (if it does not exist) and run Beagle with default settings. The information on marker position is taken from element map. Indeed, the postion in map$pos must be available for all markers. The program can only handle the position units "bp", "kb" and "Mb". Make sure that there are than only integer numbers for the unit "bp", because beagle can only work with integer numbers. 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. For details of this see the description of the `beagle` option.

"beagleNoRand" and "beagleAfterFamilyNoRand" The same as the option `beagle`, respectively `beagleAfterFamily`, except that markers without map information will be not imputed.

"random" The missing values for a marker j are sampled from the marginal allele distribution of marker j. With 2 possible genotypes (to force this option, use `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. In the standardd 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 $P(x = 2) = p^2$ assuming Hardy-Weinberg equilibrium for all loci.

"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 the imputed alleles

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

6. Discarding duplicated markers if `keep.identical=FALSE`. From identical marker based on pairwise complete oberservations one is discarded randomly. For getting identical results use the function `set.seed()` before `code.geno()`.

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

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

Note: Beagle is included in the synbreed package. Once required, Beagle is called using `path.package()`.

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 genotype 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 (2013) Improving the accuracy and efficiency of identity by descent detection in population data. Genetics 194(2):459-471

S R Browning and B L Browning (2016) Genotype imputation with millions of reference samples. Am J Hum Genet 98:116-126

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")

# comparison
gp.coded$geno
gp$geno

# example with heterogeneous stock mice
## Not run:
library(synbreedData)
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=1)

# 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)

## End(Not run)
```

`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, marker map, pedigree and further covariate data. All elements are 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, na.string="NA", cores=1)
```

Arguments

`pheno`	`data.frame` with individuals organized in rows and traits organized in columns. For unrepeated measures unique `rownames` should identify individuals. For repeated measures, the first column identifies individuals and a second column indicates repetitions (see also argument `repeated`).
`geno`	`matrix` with individuals organized in rows and markers organized in columns. Genotypes could be coded arbitrarily. Missing values should be coded as `NA`. Colums or rows with only missing values not allowed. Unique `rownames` identify individuals and unique `colnames` markers. If no `rownames` are available, they are taken from element `pheno` (if available and if dimension matches). 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 (`numeric` or `character` but not `factor`) and second column the position on the chromosome in centimorgan or the physical distance relative to the reference sequence in basepairs. Unique

	rownames indicate the marker names which should match with marker names in geno. Note that order and number of markers must not be identical with the order in geno. If this is the case, gaps in the map are filled with NA to ensure the same number and order as in element geno of the resulting gpData object.
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 (default = 'cM').
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. This argument is only required for repeated measurements.
modCovar	vector with colnames which identify columns with covariables in pheno. This argument is only required for repeated measurements.
na.string	character or vector of characters. You can specify values with which NA is coded in your geno object. In case you read missing values from a file not as missing, but as character strings. It can be specified more than one value for missings in a vector. Default is "NA".
cores	numeric. Here you can specify the number of cores you like to use.

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.

An object of class gpData can contain different subsets of individuals or markers in the elements pheno, geno and pedigree. 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 are automatically generated to identify individuals that appear in the corresponding gpData object.

Value

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

covar	data.frame with information on individuals
pheno	array (individuals x traits x replications) with phenotypic data
geno	matrix marker matrix containing genotypic data. Columns (marker) are in the same order as in map (if reorderMap=TRUE.)

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) From synbreed version 0.11-11 on the function codeGeno adds here the package version which was used to do the coding. There are differences in codings between version 0.10-11 and 0.11-0!

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 specifying the assumptions is returned. Please check them carefully.

Author(s)

Valentin Wimmer and Hans-Juergen Auinger with contributions be Peter VandeHaar

See Also

codeGeno, summary.gpData, gpData2data.frame

Examples

```
set.seed(123)
# 9 plants with 2 traits
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) and will therefore be removed
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 traits , 3 replcations
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,unknown=0)
```

Arguments

`ID`	vector of unique IDs identifying individuals
`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 for 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.
`unknown`	value for unknown or missing ancestors.

Details

Missing values for parents in the pedigree should be coded `NA`. 0 is treaded as unknown, too.

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`, `add.pedigree`

Examples

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

# create pedigree object (using argument gener)
ped <- create.pedigree(id,par1,par2,gener,unknown=c("ID0", "ID"))
ped
plot(ped)

# create pedigree object (without using argument gener)
ped2 <- create.pedigree(id,par1,par2,unknown=c("ID0", "ID"))
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 in the gpData object to be used as trait.
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 marker genotypes are used for a marker regression. In general, a covariance matrix should be non-singular and positive definite to be invertible, if this is not the case, a constant of 1e-5 is added to the diagonal elements of the covariance matrix.
k	numeric. Number of folds for k-fold cross validation, thus k should be in [2,nrow(y)] (default=2).
Rep	numeric. Number of replications (default = 1).

Seed	numeric. 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 (see Details).
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. Only required for options sampling="within popStruc" or sampling="across popStruc"
VC.est	Should variance components be reestimated with "ASReml" or with Bayesian Ridge Regression "BRR" or Bayesian Lasso "BL" of the BLR package 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 has 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 and used 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 (Albrecht et al. 2011), 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

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

gives the mixed model equations

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

Value

An 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 correlation coefficient $r(y_{TS}, \hat{y}_{TS})$.
`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 inflation of predicted values, and a coefficient of > 1 deflation of predicted values.
`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

Albrecht T, Wimmer V, Auinger HJ, Erbe M, Knaak C, Ouzunova M, Simianer H, Schoen CC (2011) Genome-based prediction of testcross values in maize. Theor Appl Genet 123:339-350

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

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

summary.cvData

Examples

```
# loading the maize data set
## Not run:
library(synbreedData)
data(maize)
maize2 <- codeGeno(maize)
U <- kin(maize2,ret="realized")
# cross validation
cv.maize  <- crossVal(maize2,cov.matrix=list(U),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)

## End(Not run)
```

---

discard.individuals *Subsets for objects of class gpData*

Description

The function produce subsets from an object of class gpData with reduced individuals. Individual information will be discarded from elements geno, pheno, covar and pedigree.

Usage

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

Arguments

gpData	object of class gpData
which	character vector identifying names of individuals get discarded from a gpData-object.
keepPedigree	logical. Should the individual only be removed from elements geno and pheno but kept in the pedigree?
whichNot	character vector identifying names of individuals get kept in a gpData-object. Overwrites argument which!

Value

Object of class gpData

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

See Also

create.gpData, add.individuals, add.markers, discard.markers

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)

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

## Not run:
# add one new DH line to maize data
library(synbreedData)
data(maize)

# delete individual
maize2 <- discard.individuals(maize,rownames(maize$geno)[1:10])
summary(maize2)


## End(Not run)
```

---

discard.markers *Subsets for objects of class gpData*

Description

The function produces subsets from an object of class gpData with reduced markers. Marker informartion will be discarded from elements geno and map

Usage

```
discard.markers(gpData, which=NULL, whichNot=NULL)
```

Arguments

gpData	object of class gpData
which	character vector identifying names of markers which get discarded in geno from a gpData-object.
whichNot	character vector identifying names of markers which get kept in geno from a gpData-object. Overwrites argument which!

Value

Object of class gpData

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

See Also

create.gpData, add.markers, add.individuals, discard.individuals

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)
```

```
## Not run:
# add one new DH line to maize data
library(synbreedData)
data(maize)

# delete markers
maize2 <- discard.individuals(maize,colnames(maize$geno)[1:50])
summary(maize2)

## End(Not run)
```

---

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

Description

Function to convert an object of class gpData to an object of class cross (F2 intercross class in the package qtl) and vice versa. If not done before, function codeGeno is used for recoding in gpData2cross.

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 because separation into chromosomes in not required for genomic prediction. 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 of gpData for function gpData2cross and 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

```
## Not run:
library(synbreedData)
# from gpData to cross
data(maize)
maizeC <- codeGeno(maize)
maize.cross <- gpData2cross(maizeC)
# descriptive statistics
summary(maize.cross)
plot(maize.cross)

# use function scanone
maize.cross <- calc.genoprob(maize.cross, step=2.5)
result <- scanone(maize.cross, pheno.col=1, method="em")
# display of LOD curve along the chromosome
plot(result)


# from cross to gpData
data(fake.f2)
fake.f2.gpData <- cross2gpData(fake.f2)
summary(fake.f2.gpData)

## End(Not run)
```

---

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's.

Usage

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

Arguments

gpData	object of class gpData
trait	numeric or character. A vector with the names or numbers of the trait that should be extracted from pheno. Default is 1.
onlyPheno	scalar logical. Only return phenotypic data.
all.pheno	scalar logical. Include all individuals with phenotypes in the data.frame and fill the genotypic data with NA.
all.geno	scalar logical. Include all individuals with genotypes 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 from the phenotypic values and covariates. Default is NULL, i.e. all values are used.
phenoCovars	logical. If TRUE, columns with the phenotypic covariables are attached from element phenoCovars to the data.frame. Only required for repeated measurements.
...	further arguments to be used in function reshape. The argument times could be useful to rename the levels of the grouping variable (such as locations or environments).

Details

Argument all.geno can be used to predict the genetic value of individuals without phenotypic records using the BGLR 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 (such as replications, years of locations in multi-environment trials) 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 and 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)
gp$covar

# as data.frame with individuals with genotypes and phenotypes
gpData2data.frame(gp,trait=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 fits genomic prediction models based on phenotypic and genotypic data in an object of class gpData. The possible models are Best Linear Unbiased Prediction (BLUP) using a

pedigree-based or a marker-based genetic relationship matrix and Bayesian Lasso (BL) or Bayesian Ridge regression (BRR). BLUP models are fitted using the REML implementation of the `regress` package (Clifford and McCullagh, 2012). The Bayesian regression models are fitted using the Gibbs-Sampler of the `BGLR` 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 used to fit the model.

Usage

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

Arguments

`gpData`	object of class `gpData`
`model`	`character`. Type of genomic prediction model. `"BLUP"` indicates best linear unbiased prediction (BLUP) using REML for both pedigree-based (P-BLUP) and marker-based (G-BLUP) model. `"BL"` and `"BRR"` indicate Bayesian Lasso and Bayesian Ridge Regression, respectively.
`kin`	object of class `relationshipMatrix` (only required for `model = "BLUP"`). Use a pedigree-based kinship to evaluate P-BLUP or a marker-based kinship to evaluate G-BLUP. For `"BL"` and `"BRR"`, also a kinship structure may be used as additional polygenic effect u in the Bayesian regression models (see `BGLR` package).
`predict`	`logical`. If `TRUE`, genetic values will be predicted for genotyped but not phenotyped individuals. Default is `FALSE`. Note that this option is only meaningful for marker-based models. For pedigree-based model, please use function `predict.gpMod`.
`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`	`logical`. Should marker effects be estimated for a G-BLUP model, i.e. RR-BLUP? In this case, argument `kin` is ignored (see Details). Plose note, that in this case also the variance components pertaining to model G-BLUP are reported instead of those from the G-BLUP model (see vignette). If the variance components are committed to `crossVal`, it must be guaranteed that there also the RR-BLUP model is used, e.g. no `cov.matrix` object should be specified.
`fixed`	A formula for fixed effects. The details of model specification are the same as for `lm` (only right hand side required). Only for `model="BLUP"`.
`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`. Only for `model="BLUP"`

`...` further arguments to be used by the genomic prediction models, i.e. prior values and MCMC options for the `BLR` function (see `BLR`) or parameters for the REML algorithm in regress.

Details

By default, an overall mean is added to the model. If no `kin` is specified and `model = "BLUP"`, a G-BLUP model will be fitted. For BLUP, further fixed and random effects can be added through the arguments `fixed` and `random`.

The marker effects $\hat{m}$ in the RR-BLUP model (available with `markerEffects`) are calculated as

$$\hat{m} = X'G^{-1}\hat{g}$$

with X being the marker matrix, $G = XX'$ and $hatg$ the vector of predicted genetic values.

Only a subset of the individuals - the training set - is used to fit the model. 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.

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` in the working directory to prevent the screen output from overload.

Author(s)

Valentin Wimmer, Hans-Juergen Auinger and Theresa Albrecht

References

Clifford D, McCullagh P (2012). regress: Gaussian Linear Models with Linear Covariance Structure. R package version 1.3-8, 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=BGLR

See Also

`kin`, `crossVal`

Examples

```
## Not run:
library(synbreedData)
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'))
mod3 <- gpMod(maizeC,model="BL",prior=prior,nIter=6000,burnIn=1000,thin=5)

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

---

kin *Relatedness based on pedigree or marker data*

Description

This function implements different measures of relatedness between individuals in an 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","gaussian"),
            DH=NULL, maf=NULL, selfing=NULL, lambda=1, P=NULL, cores=1)
```

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 doubled-haploid (DH) line and FALSE or 0 otherwise. This option is only used, if ret argument is "add" or "kin".
maf	numeric vector of length equal the number of markers. Supply values for the p_i of each marker, which were used to correct the allele counts in ret="realized" and ret="realizedAB". If not specified, p_i equals the minor allele frequency of each locus.
selfing	numeric vector of length n. It is used as the number of selfings of an recombinant inbred line individual. Be awere, that this should only be used for single seed descendants This option is only used, if ret argument is "add" or "kin".
lambda	numeric bandwidth parameter for the gaussian kernel. Only used for calculating the gaussian kernel.
P	numeric matrix of the same dimension as geno of the gpData object. This option can be used for own allelefrequencies of different groups in the genotypes.
cores	numeric. Here you can specify the number of cores you like to use.

Details

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

Function kin provides different types of measures for pedigree based relatedness. An 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. individual A has alleles $A1$ and $A2$). The gametic relationship is defined as the matrix of probabilities that two alleles are identical by descent (IBD). Note that the diagonal elements of the gametic relationship matrix are 1. The off-diagonals of individuals with unknown or unrelated parents in the 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 construct 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

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

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

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

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. An element geno must be available in the object of class gpData. Furthermore, genotypes must be coded 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 $Z = W - P$, W is the marker matrix, P contains the allele frequencies multiplied by 2, p_i is the allele frequency of marker i, and the sum is over all loci.

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, and the sum is over all 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 can 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"`.

If `ret="gaussian"`, the euklidian distances `distEuk` for all individuals are calculated. The values of `distEuk` are than used to calculate similarity coefficients between the individuals with `exp(distEuk^2/numMarker)`. Be aware that this relationship matrix scales theoretically between 0 and 1!

Value

An object of class "relationshipMatrix".

Author(s)

Valentin Wimmer and Theresa Albrecht, with contributions by Yvonne Badke

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.

Reif, J.C.; Melchinger, A. E. and Frisch, M. Genetical and mathematical properties of similarity and dissimilarity coefficients applied in plant breeding and seed bank management. Crop Science, January-February 2005, vol. 45, no. 1, p. 1-7.

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
#=========================
## Not run:
library(synbreedData)
data(maize)
K <- kin(maize,ret="kin")
plot(K)

## End(Not run)

#=========================
# (2) marker based relatedness
#=========================
## Not run:
data(maize)
U <- kin(codeGeno(maize),ret="realized")
plot(U)

## End(Not run)


### 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 Linkage Disequilibrium (LD) estimates generated by function `pairwiseLD` versus marker distance. A single plot is generated for every chromosome.

Usage

```
LDDist(LDdf,chr=NULL,type="p",breaks=NULL,n=NULL,file=NULL,fileFormat="pdf",
       onefile=TRUE,colL=2,colD=1,...)
```

Arguments

`LDdf`	object of class `LDdf` which is the output of function `pairwiseLD` and argument `type="data.frame"`
`chr`	`numeric` scalar or vector. Return value is a plot for each chromosome in `chr`. To plot the complete LD within Chromosomes in a single plot use `"all"`. Note: This includes no values for between chromosom LD! The default is `NULL`. This givs a single plot for each chromosome. Note: Remember to add one empty line for each chromosome in batch-scripts , if you use more than one chromosome!
`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.
`n`	`numeric`. Number of observations used to estimate LD. Only required for `type="nls"`.
`file`	`character`. path to a file where plot is saved to (optional).
`fileFormat`	`character`. At the moment two file formats are supported: pdf and png. Default is `"pdf"`.
`onefile`	`logical`. If `fileFormat = "pdf"` you can decide, if you like to have all graphics in one file or in multiple files.
`colL`	The color for the line if `type="nls"` is used. In other cases without a meaning.
`colD`	The color for the dots in the plot of `type="nls"` and `type="p"`
`...`	Further arguments for `plot`

Author(s)

Valentin Wimmer, Hans-Juergen Auinger and Theresa Albrecht

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

```
## Not run:
library(synbreedData)
# 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,colD=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)))

## End(Not run)
```

LDMap *LD Heatmap*

Description

Visualization of pairwise Linkage Disequilibrium (LD) estimates generated by function `pairwiseLD` in a LD heatmap for each chromosome using the `LDheatmap` package (Shin et al, 2006) .

Usage

```
LDMap(LDmat,gpData,chr=NULL,file=NULL,fileFormat="pdf",onefile=TRUE,...)
```

Arguments

`LDmat`	Object of class `LDmat` generated by function `pairwiseLD` and argument `type="matrix"`
`gpData`	Object of class `gpData` that was used in `pairwiseLD`
`chr`	`numeric`. Return value is a plot for each chromosome in `chr`.
`file`	Optionally a path to a file where the plot is saved to
`fileFormat`	`character`. At the moment two file formats are supported: pdf and png. Default is `"pdf"`.
`onefile`	`logical`. If `fileFormat = "pdf"` you can decide, if you like to have all graphics in one file or in multiple files.
`...`	Further arguments that could be passed to function `LDheatmap`

Details

Note: If you have an LDmat-object with more than one chromosome and you like to plot all chromosomes, you need to put an empty line for each chromosome in your script after the LDMap function!

Author(s)

Hans-Juergen Auinger, Theresa Albrecht and Valentin Wimmer

References

Shin JH, Blay S, McNeney B, Graham J (2006). LDheatmap: An R Function for Graphical Display of Pairwise Linkage Disequilibria Between Single Nucleotide Polymorphisms. Journal of Statistical Software, 16, Code Snippet 3. URL http://stat-db.stat.sfu.ca: 8080/statgen/research/LDheatmap/.

See Also

pairwiseLD, LDheatmap, LDDist

Examples

```
## Not run:
library(synbreedData)
data(maize)
maizeC <- codeGeno(maize)

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

## End(Not run)
```

---

manhattanPlot *Manhattan plot for SNP effects*

Description

Plot of SNP effects along the chromosome, e.g. for the visualization of marker effects generated by function gpMod.

Usage

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

Arguments

b	object of class gpMod with marker effects or numeric vector of marker effects to plot
gpData	object of class gpData with map position
colored	logical. Color the chromosomes?. The default is FALSE with chromosomes distinguished by grey tones.
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.
colP	color for the different chromosomes and points.
...	further arguments for function plot

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

Examples

```
## Not run:
library(synbreedData)
data(mice)
# plot only random noise
b <- rexp(ncol(mice$geno),3)
manhattanPlot(b,mice)

## End(Not run)
```

MME *Mixed Model Equations*

Description

Set up Mixed Model Equations for given design matrices, 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 linear mixed model is given by

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

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 corresponding mixed model equations (Henderson, 1984)

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

Matrix on left hand side of mixed model equation is denoted by LHS and matrix on the right hand side 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 prediction 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

```
## Not run:
library(synbreedData)
data(maize)

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

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

# solution with MME
diag(U) <- diag(U) + 0.000001  # to avoid singularities
# determine shrinkage parameter
lambda <- m$fit$sigma[2]/ m$fit$sigma[1]
# multiply G with shrinkage parameter
GI <- solve(U)*lambda
y <- maizeC$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)

## End(Not run)
```

---

pairwiseLD *Pairwise LD between markers*

Description

Estimate pairwise Linkage Disequilibrium (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 only 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. Additionally, the euclidian distance between position of markers is computed and returned.

Usage

```
pairwiseLD(gpData, chr = NULL, type = c("data.frame", "matrix"),use.plink=FALSE,
           ld.threshold=0, ld.window=99999, rm.unmapped = TRUE, cores=1)
```

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 (see 'Value').
use.plink	logical. Should the software PLINK be used for the computation?
ld.threshold	numeric. Threshold for the LD to thin the output. Only pairwise LD>ld.threshold is reported when PLINK is used. This argument can only be used for type="data.frame".
ld.window	numeric. Window size for pairwise differences which will be reported by PLINK (only for use.plink=TRUE; argument --ld-window-kb in PLINK) to thin the output dimensions. Only SNP pairs with a distance < ld.window are reported (default = 99999).
rm.unmapped	logical. Remove markers with unknown postion in map before using PLINK?
cores	numeric. Here you can specify the number of cores you like to use.

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 or through path variables). The function pairwiseLD calls PLINK and reads the results. The evaluation is performed separately 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 frequency of haplotype AB, $p_A = 1 - p_a$ and $p_B = 1 - p_b$ the observed frequencies of alleles A and B. If the number of markers is high, a threshold for the LD can be used to thin the output. In this case, only pairwise LD above the threshold is reported (argument --ld-window-r2 in PLINK).

Default PLINK options used –no-parents –no-sex –no-pheno –allow-no-sex –ld-window p –ld-window-kb 99999

Value

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

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

```
## Not run:
library(synbreedData)
data(maize)
maizeC <- codeGeno(maize)
maizeLD <- pairwiseLD(maizeC,chr=1,type="data.frame")

## End(Not run)
```

---

plot.LDdf *Plot function for class* LDdf

Description

The function visualises wheter the LD between adjacent values or visualization of pairwise Linkage Disequilibrium (LD) estimates generated by function pairwiseLD versus marker distance. A single plot is generated for every chromosome.

Usage

```
## S3 method for class 'LDdf'
plot(x, gpData, plotType = "dist", dense = FALSE, nMarker = TRUE,
                    centr = NULL, chr = NULL, type = "p",breaks = NULL, n = NULL,
                     file = NULL, fileFormat = "pdf", onefile = TRUE, colL = 2,
                     colD = 1, ...)
```

Arguments

x	Object of class LDdf, i.e the output of function pairwiseLD with argument type="data.frame".
gpData	Object of class gpData with object map
plotType	You can decide, if you like to have a plot with the LD of the neighbouring markers (option "neighbour"), or you like to have a scatter plot of distance and LD (default option "dist").
dense	For plotType="neighbour", logical. Should density visualization for high-density genetic maps be used?
nMarker	For plotType="neighbour", logical. Print number of markers for each chromosome?
centr	For plotType="neighbour", numeric vector. Positions for the centromeres in the same order as chromosomes in map. If "maize", centromere positions of maize in Mbp are used.
chr	For plotType="dist", numeric scalar or vector. Return value is a plot for each chromosome in chr. Note: Remember to add in a batch-script one empty line for each chromosome, if you use more than one chromosome!
type	For plotType="dist", 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	For plotType="dist", 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.
n	For plotType="dist", numeric. Number of observations used to estimate LD. Only required for type="nls".
file	Optionally a path to a file where the plot is saved to
fileFormat	character. At the moment two file formats are supported: pdf and png. Default is "pdf".
onefile	logical. If fileFormat = "pdf" you can decide, if you like to have all graphics in one file or in multiple files.
colL	The color for the line if type="nls" is used. In other cases without a meaning.
colD	The color for the dots in the plot of type="nls" and type="p"
...	further graphical arguments for function plot

Details

For more Details see at plotNeighbourLD or LDDist

Author(s)

Hans-Juergen Auinger

See Also

plotNeighbourLD, LDDist, plotGenMap, pairwiseLD

plot.LDmat *Plot function for class* LDmat

Description

A function to visualize Linkage Disequilibrium estimates between adjacent markers or isualization of pairwise Linkage Disequilibrium (LD) estimates generated by function pairwiseLD in a LD heatmap for each chromosome using the LDheatmap package (Shin et al, 2006) .

Usage

```
## S3 method for class 'LDmat'
plot(x, gpData, plotType = "map", dense = FALSE,
                    nMarker = TRUE, centr = NULL, chr = NULL,
                    file = NULL, fileFormat = "pdf", onefile = TRUE, ...)
```

Arguments

x	Object of class LDmat, i.e the output of function pairwiseLD with argument type="matrix".
gpData	Object of class gpData with object map
plotType	You can decide, if you like to have a plot with the LD of the neighbouring markers (option "neighbour"), or you like to have a heatmap of the LD (default option "map").
dense	For plotType="neighbour", logical. Should density visualization for high-density genetic maps be used?
nMarker	For plotType="neighbour", logical. Print number of markers for each chromosome?
centr	For plotType="neighbour", numeric vector. Positions for the centromeres in the same order as chromosomes in map. If "maize", centromere positions of maize in Mbp are used.
chr	For plotType="map", numeric scalar or vector. Return value is a plot for each chromosome in chr. Note: Remember to add in a batch-script one empty line for each chromosome, if you use more than one chromosome!
file	Optionally a path to a file where the plot is saved to
fileFormat	character. At the moment two file formats are supported: pdf and png. Default is "pdf".
onefile	logical. If fileFormat = "pdf" you can decide, if you like to have all graphics in one file or in multiple files.
...	Further arguments that could be passed to function LDheatmap

Details

For more details see at plotNeighbourLD or LDMap

Author(s)

Hans-Juergen Auinger

See Also

plotNeighbourLD, LDDist, plotGenMap, pairwiseLD

---

plot.pedigree *Visualization of pedigree*

Description

A function to visualize pedigree structure by a graph using the igraph package. 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 effects to plot on the x axis
...	Other arguments for function igraph.plotting

Details

The pedigree is structured top to bottom. The first generation is printed in the first line. 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 in the horizontal direction and a labeled axis is plotted at the bottom.

Value

A named graph visualizing the pedigree structure. Color is used to distinguish sex.

Note

This function uses the plotting method for graphs in the library igraph

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

See Also

create.pedigree, simul.pedigree

Examples

```
id <- paste("ID", 1:9, sep="0")
par1 <- paste("ID", c("","","","",1,1,1,4,7), sep="0")
par2 <- paste("ID", c("","","","",2,3,2,5,8), sep="0")
ped1 <- create.pedigree(id,par1,par2,unknown="ID0")
ped1
plot(ped1)

# create 2nd pedigree object
Id <- paste("ID", 10:16, sep="")
Par1 <- paste("ID", c("","",1,1,6,7,7), sep="0")
Par2 <- paste("ID", c("","",10,"08","09",11,14), sep="")
ped2 <- create.pedigree(Id,Par1,Par2,unknown=c("ID0", "ID"))
ped2

ped <- add.pedigree(ped1, ped2)
plot(ped)
```

---

plot.relationshipMatrix

Heatmap for relationship Matrix

Description

Visualization for objects of class `relationshipMatrix` using a heatmap of pairwise relatedness coefficients.

Usage

```
## S3 method for class 'relationshipMatrix'
plot(x, y=NULL, levelbreaks=NULL, axes=TRUE, cols=NULL, groupLines = NULL, ...)
```

Arguments

`x`	Object of class `relationshipMatrix`
`y`	Optional for comparisons of objects of class `relationshipMatrix`
`levelbreaks`	`list` with one element for `x` andy. Define breaks in the color scheme of the plot. If you make to many breaks, the color scheme repeats! If `y=NULL` this can be an `vector`. If you like to have the same breaks or both relationship matrices, you can also use just one vector.
`axes`	a `logical` value indicating whether axes should be drawn on the plot. Default is `TRUE`.
`cols`	a `list` with one element for each relationship matrix. Colors and the number of levelbreaks should fit. But also if not, a plot is drawn. In case option `y=NULL`, `cols` can bei a `vector`.
`groupLines`	add positions to make groups more visible
`...`	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 and Hans-Juergen Auinger

Examples

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

# big pedigree
## Not run:
library(synbreedData)
data(maize)
K <- kin(maize,ret="kin")
U <- kin(codeGeno(maize),ret="realized")/2
# equal colorkeys
plot(K,levelbreaks=seq(0,2,length=9))
plot(U,levelbreaks=seq(0,2,length=9))

## End(Not run)
```

---

plotGenMap *Plot marker map*

Description

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

Usage

```
## S3 method for class 'GenMap'
plot(x, dense = FALSE, nMarker = TRUE, bw=1,
                      centr=NULL, file=NULL, fileFormat="pdf",...)

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

Arguments

x	object of class GenMap, i. e. the map object in a gpData-object
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. Bandwidth to use for dense=TRUE to control the resolution (default = 1 [map unit]).
centr	numeric vector. Positions for the centromeres in the same order as chromosomes in map. If "maize", centromere positions of maize in Mbp are used (according to maizeGDB, version 2).
file	Optionally a path to a file where the plot is saved to
fileFormat	character. At the moment two file formats are supported: pdf and png. Default is "pdf".
...	further graphical arguments for function plot

Details

In the low density plot, the unique positions 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 an interval of equal bandwidth bw is counted. The high density plot is typically useful if the number of markers exceeds 200 per chromosome on average.

Value

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

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

See Also

create.gpData

Examples

```
## Not run:
library(synbreedData)
# low density plot
data(maize)
plotGenMap(maize)

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

## End(Not run)
```

plotNeighbourLD *Plot neighbour linkage disequilibrium*

Description

A function to visualize Linkage Disequilibrium estimates between adjacent markers.

Usage

```
plotNeighbourLD(LD, gpData, dense=FALSE, nMarker = TRUE,
                centr=NULL, file=NULL, fileFormat="pdf", ...)
```

Arguments

LD	object of class LDmat, i.e the output of function pairwiseLD using argument type="matrix".
gpData	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. Positions for the centromeres in the same order as chromosomes in map. If "maize", centromere positions of maize in Mbp are used.
file	Optionally a path to a file where the plot is saved to
fileFormat	character. At the moment two file formats are supported: pdf and png. Default is "pdf".
...	further graphical arguments for function plot

Details

The plot 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 and Hans-Juergen Auinger

See Also

plotGenMap, pairwiseLD

Examples

```
## Not run:
library(synbreedData)
data(maize)
maize2 <-codeGeno(maize)
LD <- pairwiseLD(maize2,chr=1:10,type="matrix")
plotNeighbourLD(LD,maize2,nMarker=FALSE)

## End(Not run)
```

`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. If the model includes a `relationshipMatrix`, this must include both the individuals in the training data used for fitting `gpMod` and those which sould be predicted in `newdata` (see example below).
`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),row.names=c("ID1","ID2","ID3","ID4"))
ped <- create.pedigree(ID=c("ID6","ID5","ID1","ID2","ID3","ID4"),
                       Par1=c(0,0,"ID5","ID5","ID1","ID6"),
                       Par2=c(0,0,0,0,"ID6","ID2"))
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("ID", 1:4, sep="")
mod1 <- list(fit=list(sigma=c(1,1),X = matrix(1,ncol=1,nrow=4)),kin=A,model="BLUP",y=y,m=NULL)
# matrix A included all individuals (including those which should be predicted)
class(mod1) <- "gpMod"
predict(mod1,c("ID5","ID6"))

# 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$u[1:2]
## Not run:
# prediction of genetic performance of the last 50 individuals in the maize data set
data(maize)
maizeC <- codeGeno(maize)
U <- kin(maizeC,ret="realized")
maizeC2 <- discard.individuals(maizeC,rownames(maizeC$pheno)[1201:1250])
modU <- gpMod(maizeC2,model="BLUP",kin=U)
```

```
predict(modU,rownames(maizeC$pheno)[1201:1250])

## End(Not run)
```

`read.vcf2list` *Read data of a vcf-file to a matrix*

Description

Function for easily read genomic data in vcf-Format to a list, which contains the map information and the marker information.

Usage

```
read.vcf2list(file, FORMAT = "GT", coding = c("allele", "ref"), IDinRow = TRUE, cores=1)
```

Arguments

`file`	`character`. The name of the file which the data are to be read from.
`FORMAT`	`character`. The default is `"GT"`. If there are more formats in your vcf-file you can decide which one you like to have in your output matrix.
`coding`	This option has only an effect with `FORMAT="GT"`. `allele` gives you back the alles as defined as REF and ALT in your vcf-file. `ref` gives you back `"0"` for the reference allele and `"1"` for the alternative allele.
`IDinRow`	`logical`. Default is `TRUE`, this means the genotypes are in the rows and the markers in the column. For `FALSE` it is the other way round.
`cores`	`numeric`. Specifies the number of cores for parallel computing.

Value

A list with a matrix (`matrix`) containing a representation of the genotypic data in the file and a map of classes GenMap and data.frame.

Author(s)

Hans-Juergen Auinger

See Also

`write.vcf`, `read.vcf2matrix`

Examples

```
## Not run:
library(synbreedData)
data(maize)
maize$info$map.unit <- "kb"
maize <- codeGeno(maize)
write.vcf(maize, "maize.vcf")
genInfo <- read.vcf2list("maize.vcf")

## End(Not run)
```

`read.vcf2matrix` *Read data of a vcf-file to a matrix*

Description

To easily read genomic data in vcf-Format to a matrix. Function `codeGeno` uses `read.vcf2matrix` with imputing by beagle.

Usage

```
read.vcf2matrix(file, FORMAT = "GT", coding = c("allele", "ref"), IDinRow = TRUE, cores=1)
```

Arguments

`file`	`character`. The name of the file which the data are to be read from.
`FORMAT`	`character`. The default is `"GT"`. If there are more formats in your vcf-file you can decide which one you like to have in your output matrix.
`coding`	This option has only an effect with `FORMAT="GT"`. `allele` gives you back the alles as defined as REF and ALT in your vcf-file. `ref` gives you back `"0"` for the reference allele and `"1"` for the alternative allele.
`IDinRow`	`logical`. Default is `TRUE`, this means the genotypes are in the rows and the markers in the column. For `FALSE` it is the other way round.
`cores`	`numeric`. Specifies the number of cores for parallel computing.

Value

A matrix (`matrix`) containing a representation of the data in the file.

Author(s)

Hans-Juergen Auinger

See Also

`write.vcf`, `read.vcf2list`

Examples

```
## Not run:
library(synbreedData)
data(maize)
maize$info$map.unit <- "kb"
maize <- codeGeno(maize)
write.vcf(maize, "maize.vcf")
geno <- read.vcf2matrix("maize.vcf")

## End(Not run)
```

`simul.pedigree` *Simulation of pedigree structure*

Description

This function can be used to simulate a pedigree for a given number of generations and individuals. Function assumes random mating within generations. Inbred individuals may be generated by chance.

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 (no inbreeding)? 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

Simulates 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

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

Observations are simulated according to

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

If no 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

Results are sorted for locations within individuals.

Author(s)

Valentin Wimmer

See Also

`simul.pedigree`

Examples

```
## Not run:
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))

## End(Not run)
```

---

summary.cvData *Summary of 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 gpData
...	not used

Author(s)

Valentin Wimmer

Examples

```
## Not run:
library(synbreedData)
data(maize)
summary(maize)

## End(Not run)
```

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

```
## Not run:
library(synbreedData)
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)

## End(Not run)
```

summary.LDdf *Summary for LD objects*

Description

Summary method for class "LDdf" and "LDmat"

Usage

```
## S3 method for class 'LDdf'
summary(object,cores=1,...)
## S3 method for class 'LDmat'
summary(object,cores=1,...)
```

Arguments

object	object of class LDdf or LDmat which is the output of function pairwiseLD and argument type="data.frame" or type="matrix"
cores	numeric. Specifies the number of cores for parallel computing.
...	not used

Details

Returns for each chromosome: Number of markers; mean, minimum and maximum LD measured as r^2; fraction of markers with $r^2 > 0.2$; average and maximum distance of number of markers

Author(s)

Valentin Wimmer and Hans-Juergen Auinger

See Also

pairwiseLD, ~~~

Examples

```
## Not run:
library(synbreed)
data(maize)
maizeC <- codeGeno(maize)
maizeLD <- pairwiseLD(maizeC,chr=1:10,type="data.frame")
maizeLDm <- pairwiseLD(maizeC,chr=1:10,type="matrix")
summary(maizeLD)
summary(maizeLDm)

## End(Not run)
```

---

summary.pedigree *Summary of 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

Summary of 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

```
## Not run:
library(synbreedData)
data(maize)
U <- kin(codeGeno(maize),ret="realized")
summary(U)

## End(Not run)
```

---

summaryGenMap *Summary of marker map information*

Description

This function can be used to summarize information from a marker map in an object of class gpData. Return value is a data.frame with one row for each chromosome and one row summarizing all chromosomes.

Usage

```
summaryGenMap(map, cores=1)
```

Arguments

map	data.frame with columns chr and pos or a gpData object with element map
cores	numeric. Specifies the number of cores for parallel computing.

Details

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

Value

A data.frame with one row for each chromosome and the intersection of all chromosomes 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

```
## Not run:
library(synbreedData)
data(maize)
summaryGenMap(maize)

## End(Not run)
```

---

write.beagle *Prepare genotypic data for Beagle*

Description

Create input file for Beagle software (Browning and Browning 2009) from an object of class gpData. This function is created for usage within function codeGeno to impute 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

No value is returned. Function creates 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 estimate pairwise LD through function `pairwiseLD`.

Usage

```
write.plink(gp, wdir = getwd(), prefix = paste(substitute(gp)),
            ld.threshold = 0, type = c("data.frame", "matrix"),
            ld.window=99999)
```

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.
`ld.window`	`numeric`. Window size for pairwise differences which will be reported by PLINK (only for `use.plink=TRUE`; argument `--ld-window-kb` in PLINK) to thin the output dimensions. Only SNP pairs with a distance < `ld.window` are reported (default = 99999).

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

---

write.relationshipMatrix

Writing relationshipMatrix in table format

Description

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

Usage

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

Arguments

x	Object of class "relationshipMatrix"
file	Path where 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

```
## Not run:
# 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")

## End(Not run)
```

---

write.vcf *Prepare genotypic data in vcf-Format*

Description

Create vcf-file for miscellaneous applications. Within the package it is used to write files for beagle usage.

Usage

```
write.vcf(gp, file, unphased=TRUE)
```

Arguments

gp	gpData object with elements geno and map
file	character. Filename for writing the file.
unphased	logical. The default is TRUE. Than the seperator between the alleles is "/", and the possible codings are "0/0" for 0 in the genotype matrix, "0/1" for 1 and "1/1" for 2. For getting a phased output, use unphased=FALSE. Than the seperator is "|". For hetercygous genotypes you have to change the 1 to -1 if you like to get the coding "1|0", So posible codings in this case are "0|0" for 0 in the genotype matrix, "0|1" for 1, "1|0" for -1 and "1|1" for 2.

Details

The function writes a vcf-file. The format of the output is "GT". Other formats are not supported.

Value

No value is returned. Function creates files [prefix]ingput.bgl with genotypic data in Beagle input format and [prefix]marker.txt with marker information used by Beagle.

Author(s)

Hans-Juergen Auinger

See Also

read.vcf2matrix, read.vcf2list, 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.vcf(gp1, file="test.vcf")
```

[.GenMap *Extract or replace part of map data.frame*

Description

Extract or replace part of an object of class GenMap.

Usage

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

Arguments

x	object of class "GenMap"
...	indices

Examples

```
## Not run:
data(maize)
head(maize$map)

## End(Not run)
```

[.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

```
## Not run:
data(maize)
U <- kin(codeGeno(maize),ret="realized")
U[1:3,1:3]

## End(Not run)
```

Index

∗Topic **IO**
write.relationshipMatrix, 62
∗Topic **\textasciitildekwd1**
plot.LDdf, 40
plot.LDmat, 42
∗Topic **\textasciitildekwd2**
plot.LDdf, 40
plot.LDmat, 42
∗Topic **hplot**
LDDist, 32
LDMap, 34
manhattanPlot, 35
plot.pedigree, 43
plot.relationshipMatrix, 44
plotGenMap, 45
plotNeighbourLD, 47
∗Topic **manip**
add.individuals, 4
add.markers, 5
codeGeno, 8
create.gpData, 13
create.pedigree, 16
discard.individuals, 20
discard.markers, 21
gpData2data.frame, 24
write.beagle, 60
write.plink, 61
write.vcf, 63
∗Topic **methods**
summary.cvData, 54
summary.gpData, 55
summary.gpMod, 56
summary.LDdf, 56
summary.pedigree, 57
summary.relationshipMatrix, 58
[.GenMap, 64
[.relationshipMatrix, 65

add.gpData, 3
add.individuals, 4, *6*, *21, 22*
add.markers, *4*, 5, *21, 22*
add.pedigree, 7, *16*

BLR, *18*, *28*

codeGeno, *3*, 8, *15*, *24*, *61*, *64*
create.gpData, *3*, 13, *21, 22*, *24, 25*, *46*, *60*
create.pedigree, *7*, 16, *43*, *53*
cross2gpData *(*gpData2cross*)*, 23
crossVal, 17, *28*, *38*, *55*

discard.individuals, *4*, 20, *22*
discard.markers, *6*, *21*, 21

gpData2cross, 23
gpData2data.frame, *15*, 24
gpMod, 26, *49*, *56*

kin, *28*, 29

LDDist, 32, *35*, *40, 41*, *43*
LDheatmap, *35*
LDMap, *33*, 34, *40*, *42*

manhattanPlot, 35
matrix, *50, 51*
MME, 36

pairwiseLD, *33*, *35*, 38, *41*, *43*, *47*, *57*, *62*
plot.GenMap *(*plotGenMap*)*, 45
plot.LDdf, 40
plot.LDmat, 42
plot.pedigree, *7*, *16*, 43, *53*
plot.relationshipMatrix, *32*, 44
plotGenMap, *41*, *43*, 45, *47*
plotNeighbourLD, *41–43*, 47
points, *36*
predict.gpMod, 48
print.summary.cvData *(*summary.cvData*)*, 54

print.summary.gpData *(*summary.gpData*)*, 55
print.summary.gpMod *(*summary.gpMod*)*, 56
print.summary.gpModList *(*summary.gpMod*)*, 56
print.summary.pedigree *(*summary.pedigree*)*, 57
print.summary.relationshipMatrix *(*summary.relationshipMatrix*)*, 58

read.cross, *24*
read.vcf2list, 50, *51*, *64*
read.vcf2matrix, *50*, 51, *64*
regress, *27*, *38*
reshape, *25*

simul.pedigree, *43*, 52, *54*
simul.phenotype, *53*, 53
summary.cvData, *20*, 54
summary.gpData, *15*, 55
summary.gpMod, 56
summary.gpModList *(*summary.gpMod*)*, 56
summary.LDdf, 56
summary.LDmat *(*summary.LDdf*)*, 56
summary.pedigree, 57
summary.relationshipMatrix, 58
summaryGenMap, 59

title, *36*

write.beagle, 60
write.plink, 61
write.relationshipMatrix, 62
write.vcf, *50, 51*, 63