Package: gaston (via r-universe)

September 3, 2024

Type Package

Title Genetic Data Handling (QC, GRM, LD, PCA) & Linear Mixed Models

Version 1.6

Date 2023-12-06

Encoding UTF-8

VignetteBuilder knitr

Description Manipulation of genetic data (SNPs). Computation of GRM and dominance matrix, LD, heritability with efficient algorithms for linear mixed model (AIREML). Dandine et al <doi:10.1159/000488519>.

License GPL-3

LinkingTo Rcpp, RcppParallel, RcppEigen

Suggests knitr, RcppEigen

Depends Rcpp, RcppParallel, methods

NeedsCompilation yes

LazyLoad yes

LazyData yes

Author Hervé Perdry [cre, aut, cph], Claire Dandine-Roulland [aut, cph], Deepak Bandyopadhyay [cph] (C++ gzstream class), Lutz Kettner [cph] (C++ gzstream class)

Maintainer Hervé Perdry <herve.perdry@universite-paris-saclay.fr>

Repository CRAN

Date/Publication 2023-12-07 09:00:02 UTC

Contents

gaston-package *gaston*

Description

Manipulation of genetic data (SNPs), computation of Genetic Relationship Matrix, Linkage Disequilibrium, etc. Efficient algorithms for Linear Mixed Model (AIREML, diagonalisation trick).

Introducing gaston

Gaston offers functions for efficient manipulation of large genotype (SNP) matrices, and state-of-the-art implementation of algorithms to fit Linear Mixed Models, that can be used to compute heritability estimates or to perform association tests.

Thanks to the packages `Rcpp`, `RcppParallel`, `RcppEigen`, gaston functions are mainly written in C++.

Many functions are multithreaded; the number of threads can be setted through `RcppParallel` function `setThreadOptions`. It is advised to try several values for the number of threads, as using too many threads might be conterproductive due to an important overhead.

Some functions have a `verbose` argument, which controls the function verbosity. To mute all functions at once you can use `options(gaston.verbose = FALSE)`.

Genotype matrices

An S4 class for genotype matrices is defined, named `bed.matrix`. Each row corresponds to an individual, and each column to a SNP. They can be read from files using `read.bed.matrix` and saved using `write.bed.matrix`. The function `read.vcf` reads VCF files.

In first approach, a bed.matrix behaves as a "read-only" matrix containing only 0, 1, 2 and NAs, unless the genotypes are standardized (use `standardize<-`). They are stored in a compact form, each genotype being coded on 2 bits (hence 4 genotypes per byte).

Bed.matrices can be converted to numerical matrices with `as.matrix`, and multiplied with numeric vectors or matrices with `%*%` (this feature can be used e.g. to simulate quantitative phenotypes, see a basic example in the example section of `association.test`).

It is possible to subset bed.matrices just as base matrices, writing e.g. `x[1:100,]` to extract the first 100 individuals, or `x[1:100,1000:1999]` for extract the SNPs 1000 to 1999 for these 100 individuals. The use of logical vectors for subsetting is allowed too. The functions `select.inds` and `select.snps` can also be used for subsetting with a nice syntax.

Some basic descriptive statistics can be added to a bed.matrix with `set.stats` (since `gaston 1.4`, this function is called by default by all functions that create a bed.matrix, unless `options(gaston.auto.set.stats = FALSE)` was set. Hardy-Weinberg Equilibrium can be tested for all SNPs with `set.hwe`.

Crossproducts of standardized matrices

If X is a standardized $n \times q$ genotype matrix, a Genetic Relationship Matrix (GRM) of the individuals can be computed as

$$GRM = \frac{1}{q-1} XX'$$

where q is the number of SNPs. This computation is done by the function GRM. The eigen decomposition of the GRM produces the Principal Components (PC) of the data set. If needed, the loadings corresponding to the PCs can be retrieved using bed.loadings.

Doing the above crossproduct in the reverse order produces a moment estimate of the Linkage Disequilibrium:

$$LD = \frac{1}{n-1} X'X$$

where n is the number of individuals. This computation is done by the function LD (usually, only parts of the whole LD matrix is computed). This method is also used by LD.thin to extract a set of SNPs in low linkage disequilibrium (it is often recommended to perform this operation before computing the GRM).

Linear Mixed Models

lmm.aireml is a function for linear mixed models parameter estimation and BLUP computations.

The model considered is of the form

$$Y = X\beta + \omega_1 + \ldots + \omega_k + \varepsilon$$

with $\omega_i \sim N(0, \tau_i K_i)$ for $i \in 1, \ldots, k$ and $\varepsilon \sim N(0, \sigma^2 I_n)$.

Note that very often in genetics a mixed model is written as

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

with Z a standardized genotype matrix, and $u \sim N(0, \tau I_q)$. In that case, denoting $\omega = Zu$, $\omega \sim N(0, \tau ZZ')$ and letting $K = ZZ'$ we get a mixed model of the previous form.

When $k = 1$ in the above general model (only one random term ω), the likelihood can be computed very efficiently using the eigen decomposition of $K = \mathrm{var}(\omega)$. This "diagonalization trick" is used in lmm.diago.likelihood and lmm.diago, to compute the likelihood and for parameter estimation, respectively.

Two small functions complete this set of functions: lmm.simu, to simulate data under a linear mixed model, and random.pm, to generate random positive matrices. Both are used in examples and can be useful for data simulation.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

Maintainer: Hervé Perdry

AGT *AGT data set*

Description

These data have been extracted from the 1000 Genomes data. The data set contains the genotype matrix `AGT.gen`, the pedigree matrix `AGT.fam` and a matrix `AGT.bim`, corresponding to 503 individuals of European populations and 361 SNPs on chromosome 1, on a ~100kb segment containing the Angiotensinogen gene. There is also a factor `AGT.pop`, which gives the population from which each individual is drawn (CEU = Utah residents of Northern Western European ancestry, FIN = Finnish, GBR = England and Scottland, IBS = Iberian, TSI = Toscani).

Usage

```
data(AGT)
```

Format

There are three data objects in the dataset:

`AGT.gen` Genotype matrix

`AGT.fam` Data frame containing all variables corresponding to a `.fam` file

`AGT.bim` Data frame containing all variables corresponding to a `.bim` file

`AGT.pop` Factor giving the population from which each individual is drawn

Source

The data were obtained from the 1000 Genomes project (see https://www.internationalgenome.org/).

References

McVean et al, 2012, *An integrated map of genetic variation from 1,092 human genomes*, Nature **491, 56-65** doi:10.1038/nature11632

Examples

```
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)
x
```

`as.bed.matrix` *Creation of a bed.matrix*

Description

Creates a bed.matrix using a numeric matrix and two data frame for ped / snps slots

Usage

```
as.bed.matrix(x, fam, bim)
```

Arguments

`x`	A numeric matrix
`fam`	(Optionnal) A data frame (the contents of a .fam file)
`bim`	(Optionnal) A data frame (the contents of a .bim file)

Details

The data frame `fam` should have columns named "famid", "id", "father", "mother", "sex" and "pheno". The data frame `bim` should have columns named "chr", "id", "dist", "pos", "A1" and "A2".

Value

A bed.matrix condensing all three arguments.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`bed.matrix-class`

Examples

```
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)
x
```

`association.test` *Association Test*

Description

Association tests between phenotype and SNPs.

Usage

```
association.test(x, Y = x@ped$pheno, X = matrix(1, nrow(x)),
                 method = c("lm", "lmm"), response = c("quantitative", "binary"),
                  test = c("score", "wald", "lrt"), K, eigenK, beg = 1,
                  end = ncol(x), p = 0, tol = .Machine$double.eps^0.25, ...)
```

Arguments

`x`	A `bed.matrix`
`Y`	The phenotype vector. Default is the column (`pheno`) of `x@ped`
`X`	A covariable matrix. The default is a column vector of ones, to include an intercept in the model
`method`	Method to use: `"lm"` for (generalized) linear model, and `"lmm"` for (generalized) linear mixed model
`response`	Is `"Y"` a quantitative or a binary phenotype?
`test`	Which test to use. For binary phenotypes, `test = "score"` is mandatory
`K`	A Genetic Relationship Matrix (as produced by `GRM`), or a list of such matrices. For `test = "score"`.
`eigenK`	Eigen decomposition of the Genetic Relationship Matrix (as produced by the function `eigen`). For `test = "wald"` or `"lrt"`.
`beg`	Index of the first SNP tested for association
`end`	Index of the last SNP tested for association
`p`	Number of Principal Components to include in the model with fixed effect (for `test = "wald"` or `"lrt"`)
`tol`	Parameter for the likelihood maximization (as in `optimize`)
`...`	Additional parameters for `lmm.aireml` or `logistic.mm.aireml` (if `test = "score"`).

Details

Tests the association between the phenotype and requested SNPs in `x`.

If `method = "lm"` and `response = "quantitative"` are used, a simple linear regression is performed to test each SNP in the considered interval. Precisely, the following model is considered for each SNP,

$$Y = (X|PC)\alpha + G\beta + \varepsilon$$

with $\varepsilon \sim N(0, \sigma^2 I_n)$, G the genotype vector of the SNP, X the covariates matrix, and PC the matrix of the first p principal components. A Wald test is performed, independently of the value of `test`.

If`method` = `"lm"` and `response` = `"binary"`, a similar model is used for a logistic regression (Wald test).

If `method` = `"lmm"` and `response` = `"quantitative"`, the following model in considered for each SNP

$$Y = (X|PC)\alpha + G\beta + \omega + \varepsilon$$

with $\omega \sim N(0, \tau K)$ and $\varepsilon \sim N(0, \sigma^2 I_n)$. G is the genotype vector of the SNP, K is a Genetic Relationship Matrix (GRM) X the covariates matrix, and PC the matrix of the first p principal components.

If `test` = `"score"`, all parameters are estimated with the same procedure as in `lmm.aireml` and the argument `K` is used to specify the GRM matrix (or a list of GRM matrices for inclusion of several random effects in the model). If `p` is positive, the paramater `eigenK` needs to be given as well. For Wald and LRT tests the procedure used is the same as in `lmm.diago` and `eigenK` is used to specify the GRM matrix.

If `method` = `"lmm"` and `response` = `"binary"`, the following model in considered for each SNP

$$\text{logit}(P[Y = 1|X, G, \omega]) = X\alpha + G\beta + \omega$$

with $\omega \sim N(0, \tau K)$. G is the genotype vector of the SNP, K is a Genetic Relationship Matrix (GRM), X the covariable matrix. A score test is performed, independently of the value of `test`. All parameters under null model are estimated with the same procedure as in `logistic.mm.aireml`. In case of convergence problems of the null problem, the user can try several starting values (in particular with parameter `tau`, trying e.g. `tau = 0.1` or another value). It is possible to give a list of matrices in parameter `K` for inclusion of several random effects in the model. If `p` is positive, the paramater `eigenK` needs to be given as well.

Note: this function is not multithreaded. Wald test with Linear Mixed Models are computationally intensive, to run a GWAS with such tests consider using `association.test.parallel` in package `gaston.utils` (on github). Association tests with dosages can be done with `association.test.dosage` and `association.test.dosage.parallel` in the same package.

Value

A data frame, giving for each considered SNP, its position, id, alleles, and some of the following columns depending on the values of `method` and `test`:

`score`	Score statistic for each SNP
`h2`	Estimated value of $\frac{\tau}{\tau+\sigma^2}$
`beta`	Estimated value of β
`sd`	Estimated standard deviation of the β estimation
`LRT`	Value of the Likelihood Ratio Test
`p`	The corresponding p-value

See Also

`qqplot.pvalues`, `manhattan`, `lmm.diago`, `lmm.aireml`, `logistic.mm.aireml`, `optimize`

Examples

```
# Load data
data(TTN)
x <- as.bed.matrix(TTN.gen, TTN.fam, TTN.bim)
standardize(x) <- "p"

# simulate quantitative phenotype with effect of SNP #631
set.seed(1)
y <- x %*% c(rep(0,630),0.5,rep(0,ncol(x)-631)) + rnorm(nrow(x))

# association test with linear model
test <- association.test(x, y, method="lm", response = "quanti")

# a p-values qq plot
qqplot.pvalues(test)

# a small Manhattan plot
# hihlighting the link between p-values and LD with SNP #631
lds <- LD(x, 631, c(1,ncol(x)))
manhattan(test, col = rgb(lds,0,0), pch = 20)

# use y to simulate a binary phenotype
y1 <- as.numeric(y > mean(y))

# logistic regression
t_binary <- association.test(x, y1, method = "lm", response = "binary")
# another small Manhattan plot
manhattan(t_binary, col = rgb(lds,0,0), pch = 20)
```

---

bed.loadings *SNP loadings*

Description

Compute the loadings corresponding to given PCs.

Usage

```
bed.loadings(x, pc)
```

Arguments

x	A bed.matrix
pc	A matrix with Principal Components in columns

Value

A matrix with the corresponding loadings in columns.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

Examples

```
# load chr2 data set (~10k SNPs in low LD)
x <- read.bed.matrix( system.file("extdata", "chr2.bed", package="gaston") )

# Compute Genetic Relationship Matrix
standardize(x) <- "p"
K <- GRM(x)

# Eigen decomposition
eiK <- eigen(K)
# deal with small negative eigen values
eiK$values[ eiK$values < 0 ] <- 0

# Note: the eigenvectors are normalized, to compute 'true' PCs
# multiply them by the square root of the associated eigenvalues
PC <- sweep(eiK$vectors, 2, sqrt(eiK$values), "*")

# Compute loadings for the 2 first PCs
# one can use PC[,1:2] instead of eiK$vectors[,1:2] as well
L <- bed.loadings(x, eiK$vectors[,1:2])
dim(L)
head(L)

# the loadings are normalized
colSums(L**2)

# Verify that these are loadings
head( (x %*% L) / sqrt(ncol(x)-1) )
head( PC[,1:2] )
```

---

bed.matrix-class *Class* "bed.matrix"

Description

S4 class for SNP genotype matrices

Objects from the Class

Objects can be created by calls of the form `new("bed.matrix", ...)`.

Slots

`ped`: `data.frame` containing information for each individual: `famid` = Family ID, `id` = Individual ID, `father` = Father ID, `mother` = Mother ID, `sex` = Sex and `pheno` = Phenotype. Can also contain individuals statistic, for example: `N0`, `N1` and `N2` = Number of genotypes equal to 0, 1 and 2 respectively, `NAs` = Number of missing genotypes, `callrate` = Individual callrate.

`snps`: `data.frame` containing information for each SNP: `chr` = Chromosome, `id` = SNP ID, `dist` = Genetic Distance, `pos` = Physical position, `A1` = Reference Allele, `A2` = Alternative Allele. Can also contain SNPs statistic, for example: `N0`, `N1` and `N2` = Number of genotypes equal to 0, 1 and 2 repectively, `NAs` = Number of missing genotypes, `callrate` = SNP callrate, `maf` = Minor allele frequency), `hz` = heterozygosity

`bed`: `externalptr` (pointing to the genotype matrix).

`p`: `vector` or `NULL` for allelic frequencies (allèle `A2`).

`mu`: `vector` or `NULL` for genotype means (usually `mu = 2*p`).

`sigma`: `vector` or `NULL` for genotypic standard deviation

`standardize_p`: `logical`. If `TRUE`, the genotype matrix is standardized using means `2*p` and genotypic standard deviation `sqrt(2*p*(1-p))`

`standardize_mu_sigma`: `logical`. If `TRUE`, the genotype matrix is standardize using means `mu` and genotypic standard deviation `sigma`.

For more details please check the vignette.

Methods

[`signature(x = "bed.matrix", i = "numeric" or "logical" or "missing", j = "numeric" or "logical" or "missing", drop = "missing")`:
Extract a sub-matrix (a new `bed.matrix`).

%*% `signature(x = "bed.matrix", y = "matrix" or "vector")`:
Right matrix multiplication of the genotype matrix (possibly centered and reduced) with a `matrix` or a `vector`.

%*% `signature(x = "matrix" or "vector", y = "bed.matrix")`:
Left matrix multiplication of the genotype matrix with a `matrix` or a `vector`.

as.matrix `signature(x = "bed.matrix")`:
Convert a `bed.matrix` into a `matrix`. The resulting matrix can be huge, use this method only for a small bed.matrix!

standardize `signature(x = "bed.matrix")`:
Get the standardize parameter of `bed.matrix`. Can be "none", "p" or "mu_sigma".

standardize<- `signature(x = "bed.matrix")`:
Set the standardize parameter of a `bed.matrix`.

dim `signature(x = "bed.matrix")`:
Get the number of individuals (rows) and the number of SNPs (columns).

head `signature(x = "bed.matrix")`:
Print the head of the genotype matrix of a `bed.matrix` object.

mu `signature(x = "bed.matrix")`:
Get the `mu` slot of a `bed.matrix`.

mu<- `signature(x = "bed.matrix")`:
Set the `mu` slot of a `bed.matrix`.

p `signature(x = "bed.matrix")`:
Get the `p` slot of a `bed.matrix`.

p<- `signature(x = "bed.matrix")`:
Set the `p` slot of a `bed.matrix`.

show `signature(object = "bed.matrix")`:
Displays basic information about a `bed.matrix`.

sigma `signature(x = "bed.matrix")`:
Get the `sigma` slot of a `bed.matrix`.

sigma<- `signature(x = "bed.matrix")`:
Set the `sigma` slot of a `bed.matrix`.

cbind `signature(... = "bed.matrix")`:
Combine a sequence of `bed.matrix` by columns.

rbind `signature(... = "bed.matrix")`:
Combine a sequence of `bed.matrix` by rows.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`read.bed.matrix`, `write.bed.matrix`, `set.stats`, `select.snps`, `select.inds`, `GRM`

Examples

```
showClass("bed.matrix")

# Conversion example
data(LCT)
x1 <- as(LCT.gen, "bed.matrix")
x1
head(x1@ped)
head(x1@snps)

# the function as.bed.matrix is an alternative
x2 <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)
x2
head(x2@ped)
head(x2@snps)
```

DM *Dominance Matrix*

Description

Compute the Dominance Matrix

Usage

```
DM(x, which.snps, autosome.only = TRUE, chunk = 1L)
```

Arguments

x	A bed.matrix
which.snps	Logical vector, giving which snps to use in the computation. The default is to use all autosomal SNPs
autosome.only	If TRUE, only autosomal SNPs will be considered.
chunk	Parameter for the parallelization: how many SNPs are treated by each task

Details

The Dominance Matrix (DM) gives for each pair of individuals an estimation of their probability of sharing two alleles Identical By Descent.

It is computed by a moment estimator, $\frac{1}{q}ZZ'$ with Z the matrix with entries $\frac{p}{1-p}$, -1, $\frac{1-p}{p}$ according to the values 0, 1, 2 in the genotype matrix x (here p is the frequency of the alternate allele, and q is the number of SNPs (ncol(x)).

Value

A symmetric square matrix of dimension equal to the number of individuals. Each entry can be interpreted as an estimated probability of sharing two alleles IBD — as it is a moment estimator, the value can (and will) fall outside of the range (0,1).

See Also

GRM, reshape.GRM

Examples

```
# load chr2 data set (~10k SNPs in low LD)
x <- read.bed.matrix( system.file("extdata", "chr2.bed", package="gaston") )

# Compute Dominance Matrix
D <- DM(x)
dim(D)
```

dupli *Small data set to illustrate* SNP.rm.duplicates

Description

The SNPs in this data frame are as follows:

SNP 1. Unduplicated SNP

SNPs 2a and 2b. Two duplicated SNPs with identical alleles

SNPs 3a and 3b. Two duplicated SNPs with swapped alleles

SNPs 4a and 4b. Two duplicated SNPs with flipped reference strand

SNPs 5a and 5b. Two duplicated SNPs with swapped alleles and flipped reference strand

SNPs 6a and 6b. Two duplicated SNPs with incompatible alleles

SNPs 7a and 7b. Two duplicated SNPs including one monomorphic SNP (one allele set to `"0"`)

SNPs 8a, 8b and 8c. Three duplicated SNPs

SNPs 9a, 9b and 9c. Three duplicated SNPs with incompatible alleles

Usage

```
data(dupli)
```

Format

There are three data objects in the dataset:

`dupli.gen` Genotype matrix

`dupli.ped` Data frame containing all variables corresponding to a `.fam` file

`dupli.bim` Data frame containing all variables corresponding to a `.bim` file

See Also

SNP.rm.duplicates

Examples

```
data(dupli)
x <- as.bed.matrix(dupli.gen, fam = dupli.ped, bim = dupli.bim)
```

GRM *Genetic Relationship Matrix*

Description

Compute the Genetic Relationship Matrix

Usage

```
GRM(x, which.snps, autosome.only = TRUE, chunk = 1L)
```

Arguments

x	A bed.matrix
which.snps	Logical vector, giving which snps to use in the computation. The default is to use all autosomal SNPs
autosome.only	If TRUE, only autosomal SNPs will be considered.
chunk	Parameter for the parallelization: how many SNPs are treated by each task

Details

The Genetic Relationship Matrix (GRM) is computed by the formula $\frac{1}{q}XX'$, with X the standardized genotype matrix and q the number of SNPs (ncol(x)).

If x is not standardized before this computation, the function will use standardize(x) <- "p" by default.

Value

The GRM is a symmetric square matrix of dimension equal to the number of individuals. Each entry can be interpreted as an estimated kinship coefficient between individuals, although some authors might disagree. Note in particular that some entries will be negative.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

DM, reshape.GRM, lmm.aireml, lmm.diago, standardize, bed.loadings

Examples

```
# load chr2 data set (~10k SNPs in low LD)
x <- read.bed.matrix( system.file("extdata", "chr2.bed", package="gaston") )

# Compute Genetic Relationship Matrix
K <- GRM(x)
dim(K)
```

is.autosome *Autosomes and X, Y, MT chromosomes*

Description

Test if a chromosome id corresponds to an autosome or to X, Y, MT chromosomes

Usage

```
is.autosome(chr)
is.chr.x(chr)
is.chr.y(chr)
is.chr.mt(chr)
```

Arguments

chr	A vector of chromosome ids

Details

These functions work by comparing the ids given in parameters with the options `gaston.autosomes`, `gaston.chr.x`, `gaston.chr.y`, `gaston.chr.mt`.

For example, `is.autosome(chr)` is a short cut for `chr %in% getOption("gaston.autosomes")`.

Value

A logical vector.

Author(s)

Hervé Perdry

LCT *LCT data set*

Description

These data have been extracted from the 1000 Genomes data. The data set contains the genotype matrix `LCT.gen`, the pedigree matrix `LCT.fam` and a matrix `LCT.bim`, corresponding to 503 individuals of European populations and 607 SNPs on chromosome 2, on a ~300kb segment containing the Lactase gene. There is also a factor `LCT.pop`, which gives the population from which each individual is drawn (CEU = Utah residents of Northern Western European ancestry, FIN = Finnish, GBR = England and Scottland, IBS = Iberian, TSI = Toscani).

Note that the SNP rs4988235 is associated with lactase persistence / lactose intolerence.

Usage

```
data(LCT)
```

Format

There are three data objects in the dataset:

`LCT.gen` Genotype matrix

`LCT.fam` Data frame containing all variables corresponding to a `.fam` file

`LCT.bim` Data frame containing all variables corresponding to a `.bim` file

`LCT.pop` Factor giving the population from which each individual is drawn

Source

The data were obtained from the 1000 Genomes project (see https://www.internationalgenome.org/).

References

McVean et al, 2012, *An integrated map of genetic variation from 1,092 human genomes*, Nature **491, 56-65** doi:10.1038/nature11632

Examples

```
data(LCT)
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)
x
which(x@snps$id == "rs4988235")
```

LD *Linkage Disequilibrium*

Description

Compute Linkage Disequilibrium (LD) between given SNPs.

Usage

```
LD(x, lim, lim2, measure = c("r2", "r", "D"), trim = TRUE)
```

Arguments

`x`	A `bed.matrix`
`lim`	Range of SNPs for which the LD is computed
`lim2`	(Optional) Second range of SNPs (see Details)
`measure`	The LD measure
`trim`	`Logical`. If `TRUE`, the values above 1 or below -1 are replaced by 1 and -1 respectively.

Details

If `lim2` is missing, the LD is computed between all SNPs with indices between `lim[1]` and `lim[2]`; else, the LD is computed between the SNPs in the range given by `lim` and those in the range given by `lim2`.

Note that the LD estimates are moment estimates (which are less precise than Maximum Likelihood Estimates). If `standardize(x) = "none"`, `x` will be standardized using `x@mu` and `x@sigma`. If `standardize(x) = "p"`, the moment estimates can produce r values outside of the range $[-1; 1]$, hence the parameter `trim`. We recommend to set `standardize(x) <- "mu"` (trimming can still be necessary due to rounding errors).

Value

A matrix of LD values.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`LD.thin`, `LD.plot`

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute LD
ld.x <- LD(x, c(1,ncol(x)))

# Plot a tiny part of the LD matrix
LD.plot( ld.x[1:20,1:20], snp.positions = x@snps$pos[1:20] )
```

`LD.clump` *LD clumping*

Description

Construct group of SNPs in LD with 'top associated SNPs'

Usage

```
LD.clump(x, p, r2.threshold, p.threshold, max.dist = 500e3)
```

Arguments

x	A bed.matrix
p	A vector of p-values, or a data frame including p-values, such as sent back by association.test
r2.threshold	The maximum LD (measured by r^2) between SNPs in a group
p.threshold	The threshold used to define associated SNPs
max.dist	The maximum distance for which the LD is computed

Details

The p-values provided through argument p are assumed to correspond to the result of an association test with the SNPs of x.

The aim of the function is to construct cluster of SNPs in strong LD with associated SNPs. The algorithm first seeks the SNP with the lowest p-value (below p.threshold) ; this SNP will be the 'index' of a cluster. The corresponding cluster is constructed by aggregating SNPs that are in LD (above r2.threshold) with the index. The cluster's name is the position of the index SNP. The processus is repeated on the SNPs which are not yet attributed to a cluster, until there is no associated SNP (ie SNP with a p-value below threshold) left. The remaining SNPs are attributed to cluster 0.

The LD is computed only for SNP pairs for which distance is inferior to max.dist, expressed in number of bases: above this distance it is assumed to be null.

Value

If p was a data frame, then the function returns the same data frame with to extra columns, cluster and is.index. If p was a vector of p-values, it returns a data frame with columns chr, id, pos, p, cluster and is.index.

See Also

LD, LD.thin

Examples

```
# Construct a bed matrix
x <- as.bed.matrix(TTN.gen, TTN.fam, TTN.bim)
standardize(x) <- "p"

# simulate quantitative phenotype with effect of SNPs #108 and #631
beta <- numeric(ncol(x))
beta[c(108,631)] <- 0.5
set.seed(1)
y <- x %*% beta + rnorm(nrow(x))

# association test with linear model
test <- association.test(x, y, method="lm", response = "quanti")

# LD clumping
```

```
test <- LD.clump(x, test, r2.threshold = 0.25, p.threshold = 1e-8)

# use as.factor for a quick-and-dirty cluster colouring on the manhattan plot
manhattan(test, col = as.factor(test$cluster), pch = 20)
```

LD.plot *Plot Linkage Disequilibrium*

Description

Pretty plot of a Linkage Disequilibrium (LD) matrix

Usage

```
LD.plot(LD, snp.positions, max.dist = Inf, depth = nrow(LD),
               graphical.par = list(mar = c(0,0,0,0)), cex.ld, cex.snp,
               polygon.par = list(border = "white"),
               color.scheme = function(ld) rgb(1,1-abs(ld),1-abs(ld)),
              write.snp.id = TRUE, write.ld = function(ld) sprintf("%.2f", ld),
               draw.chr = TRUE, above.space = 1 + 2*write.snp.id + draw.chr,
               below.space = 1, pdf.file, finalize.pdf = TRUE)
```

Arguments

LD	A symmetric LD matrix (such as produced by LD
snp.positions	A vector of SNP positions
max.dist	Maximal distance above which the LD is not plotted
depth	Maximal number of neighbouring SNPs for which the LD is plotted
graphical.par	A list of graphical parameters for function par
cex.ld	The magnification to be used for LD values (if missing, an ad-hoc value is computed)
cex.snp	The magnification to be used for SNPs ids (if missing, an ad-hoc value is computed)
polygon.par	A list of parameters for function polygon
color.scheme	A function to set the background color of a cell
write.snp.id	Logical. If TRUE, SNP ids will be displayed above the plot
write.ld	NULL, or a function which outputs the string used for displaying a LD value in a cell
draw.chr	Logical. If TRUE, a chromosome with SNP positions is sketched above the plot
above.space	Space above the plot (in user units = height of a cell)
below.space	Space below the plot (in user units = height of a cell)
pdf.file	The name of a pdf file in which to plot the LD matrix. If missing, current plot device will be used
finalize.pdf	Logical. If TRUE, dev.off() will be called to finalize the pdf file

Details

This function displays a LD plot similar to Haploview plots.

To add anotations to the plot, it is useful to know that each cell has width and height equal to one user unit, the first cell in the upper row being centered at coordinates `(1.5, -0.5)`.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

LD

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute LD
ld.x <- LD(x, c(1,ncol(x)))

# Plot a tiny part of the LD matrix
LD.plot( ld.x[1:20,1:20], snp.positions = x@snps$pos[1:20] )

# Customize the plot
LD.plot( ld.x[1:20,1:20], snp.positions = x@snps$pos[1:20],
         graphical.par = list(cex = 1.3, bg = "gray"),
         polygon.par = list(border = NA), write.ld = NULL )
## Not run:
# Plotting the whole matrix in X11 display is very long (lots of polygons)
# but it is ok with a pdf file
# (please uncomment to run)
#LD.plot(ld.x, snp.positions = x@snps$pos, max.dist = 50e3, write.ld = NULL, pdf.file = "LDAGT.pdf")

## End(Not run)
```

LD.thin *LD thinning*

Description

Select SNPs in LD below a given threshold.

Usage

```
LD.thin(x, threshold, max.dist = 500e3, beg = 1, end = ncol(x),
        which.snps, dist.unit = c("bases", "indices", "cM"),
        extract = TRUE, keep = c("left", "right", "random"))
```

Arguments

x	A bed.matrix
threshold	The maximum LD (measured by r^2) between SNPs
max.dist	The maximum distance for which the LD is computed
beg	The index of the first SNP to consider
end	The index of the last SNP to consider
which.snps	Logical vector, giving which SNPs are considerd. The default is to use all SNPs
dist.unit	Distance unit in max.dist
extract	A logical indicating whether the function return a bed.matrix (TRUE) or a logical vector indicating which SNPs are selected (FALSE)
keep	Which SNP is selected in a pair with LD above threshold

Details

The SNPs to keep are selected by a greedy algorithm. The LD is computed only for SNP pairs for which distance is inferior to max.dist, expressed in number of bases if dist.unit = "bases", in number of SNPs if dist.unit = "indices", or in centiMorgan if dist.unit = "cM".

The argument which.snps allows to consider only a subset of SNPs.

The algorithm tries to keep the largest possible number of SNPs: it is not appropriate to select tag-SNPs.

Value

If extract = TRUE, a bed.matrix extracted from x with SNPs in pairwise LD below the given threshold. If extract = FALSE, a logical vector of length end - beg + 1, where TRUE indicates that the corresponding SNPs is selected.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

LD, set.dist

Examples

```
# Load data
data(TTN)
x <- as.bed.matrix(TTN.gen, TTN.fam, TTN.bim)

# Select SNPs in LD r^2 < 0.4, max.dist = 500 kb
y <- LD.thin(x, threshold = 0.4, max.dist = 500e3)
y

# Verifies that there is no SNP pair with LD r^2 > 0.4
# (note that the matrix ld.y has ones on the diagonal)
```

```
ld.y <- LD( y, lim = c(1, ncol(y)) )
sum( ld.y > 0.4 )
```

`lik.contour` *Contour plot for two parameters likelihood*

Description

Create a contour plot (superimposed with a heat map)

Usage

```
lik.contour(x, y, z, levels = NULL, nlevels = 11, heat = TRUE, col.heat = NULL, ...)
```

Arguments

`x, y, z`	As in `contour`
`levels`	As in `contour`. If `NULL`, the function computes appropriate levels.
`nlevels`	As in `contour`
`heat`	If `TRUE`, a heat map is superimposed to the contour plot
`col.heat`	Vector of heat colors
`...`	Additional arguments to `image` and `contour`

Details

This function is a wrapper for `contour`, with a different method to compute a default value for levels. If `heat = TRUE`, a heatmap produced by `image` is added to the plot. See `contour` for details on parameters.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`lmm.diago.likelihood`, `contour`, `image`

Examples

```
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute Genetic Relationship Matrix
K <- GRM(x)

# eigen decomposition of K
eiK <- eigen(K)
```

```
# simulate a phenotype
set.seed(1)
y <- 1 + lmm.simu(tau = 1, sigma2 = 2, eigenK = eiK)$y

# Likelihood
TAU <- seq(0.5,2.5,length=30)
S2 <- seq(1,3,length=30)
lik1 <- lmm.diago.likelihood(tau = TAU, s2 = S2, Y = y, eigenK = eiK)
lik.contour(TAU, S2, lik1, heat = TRUE, xlab = "tau", ylab = "sigma^2")
```

`lmm.aireml` *Linear mixed model fitting with AIREML*

Description

Estimate the parameters of a linear mixed model, using Average Information Restricted Maximum Likelihood (AIREML) algorithm.

Usage

```
lmm.aireml(Y, X = matrix(1, nrow = length(Y)), K,
           EMsteps = 0L, EMsteps_fail = 1L, EM_alpha = 1,
           min_tau, min_s2 = 1e-06, theta, constraint = TRUE, max_iter = 50L,
           eps = 1e-05, verbose = getOption("gaston.verbose", TRUE),
           contrast = FALSE, get.P = FALSE)
```

Arguments

`Y`	Phenotype vector
`X`	Covariable matrix. By default, a column of ones to include an intercept in the model
`K`	A positive definite matrix or a `list` of such matrices
`EMsteps`	Number of EM steps ran prior the AIREML
`EMsteps_fail`	Number of EM steps performed when the AIREML algorithm fail to improve the likelihood value
`EM_alpha`	Tweaking parameter for the EM (see Details)
`min_tau`	Minimal value for model parameter τ (if missing, will be set to 10^{-6})
`min_s2`	Minimal value for model parameter σ^2
`theta`	(Optional) Optimization starting point `theta = c(sigma^2, tau)`
`constraint`	If `TRUE`, the model parameters respect the contraints given by `min_tau` and `min_s2`
`max_iter`	Maximum number of iterations
`eps`	The algorithm stops when the gradient norm is lower than this parameter

verbose	If TRUE, display information on the algorithm progress
contrast	If TRUE, use a contrast matrix to compute the Restricted Likelihood (usually slower)
get.P	If TRUE, the function sends back the last matrix P computed in the optimization process

Details

Estimate the parameters of the following linear mixed model, using AIREML algorithm:

$$Y = X\beta + \omega_1 + \ldots + \omega_k + \varepsilon$$

with $\omega_i \sim N(0, \tau_i K_i)$ for $i \in 1, \ldots, k$ and $\varepsilon \sim N(0, \sigma^2 I_n)$.

The variance matrices K_1, ..., K_k, are specified through the parameter K.

If EMsteps is positive, the function will use this number of EM steps to compute a better starting point for the AIREML algorithm. Setting EMsteps to a value higher than max_iter leads to an EM optimization. It can happen that after an AIREML step, the likelihood did not increase: if this happens, the functions falls back to EMsteps_fail EM steps. The parameter EM_alpha can be set to a value higher than 1 to attempt to accelerate EM convergence; this could also result in uncontrolled behaviour and should be used with care.

After convergence, the function also compute Best Linear Unbiased Predictors (BLUPs) for β and ω, and an estimation of the participation of the fixed effects to the variance of Y.

Value

A named list with members:

sigma2	Estimate of the model parameter σ^2
tau	Estimate(s) of the model parameter(s) $\tau_1, \ldots, \tau_k$
logL	Value of log-likelihood
logL0	Value of log-likelihood under the null model (without random effect)
niter	Number of iterations done
norm_grad	Last computed gradient's norm
P	Last computed value of matrix P (see reference)
Py	Last computed value of vector Py (see reference)
BLUP_omega	BLUPs of random effects
BLUP_beta	BLUPs of fixed effects β
varbeta	Variance matrix for β estimates
varXbeta	Participation of fixed effects to variance of Y

If get.P = TRUE, there is an additional member:

P	The last matrix P computed in the AIREML step

Author(s)

Hervé Perdry and Claire Dandine-Roulland

References

Gilmour, A. R., Thompson, R., & Cullis, B. R. (1995), *Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models*, Biometrics, **1440-1450**

See Also

lmm.diago, logistic.mm.aireml, lmm.simu

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute Genetic Relationship Matrix
standardize(x) <- "p"
K <- GRM(x)

# Simulate a quantitative genotype under the LMM
set.seed(1)
y <- 1 + x %*% rnorm(ncol(x), sd = 1)/sqrt(ncol(x)) + rnorm(nrow(x), sd = sqrt(2))

# Estimates
estimates <- lmm.aireml(y, K = K, verbose = FALSE)
str(estimates)
```

---

lmm.diago *Linear mixed model fitting with the diagonalization trick*

Description

Estimate the parameters of a linear mixed model, using the "diagonalization trick".

Usage

```
lmm.diago(Y, X = matrix(1, nrow=length(Y)), eigenK, p = 0,
                method = c("newton", "brent"), min_h2 = 0, max_h2 = 1,
                verbose = getOption("gaston.verbose", TRUE),
                tol = .Machine$double.eps^0.25)
```

Arguments

Y	Phenotype vector
X	Covariable matrix
eigenK	Eigen decomposition of K (a positive symmetric matrix)
p	Number of Principal Components included in the mixed model with fixed effect
method	Optimization method to use

min_h2	Minimum admissible value
max_h2	Maximum admissible value
verbose	If TRUE, display information on the function actions
tol	Accuracy of estimation

Details

Estimate the parameters of the following linear mixed model, computing the restricted likelihood as in lmm.diago.likelihood, and using either a Newton algorithm, or Brent algorithm as in optimize:

$$Y = (X|PC)\beta + \omega + \varepsilon$$

with $\omega \sim N(0, \tau K)$ and $\varepsilon \sim N(0, \sigma^2 I_n)$.

The matrix K is given through its eigen decomposition, as produced by eigenK = eigen(K, symmetric = TRUE). The matrix $(X|PC)$ is the concatenation of the covariable matrix X and of the first p eigenvectors of K, included in the model with fixed effects.

Value

If the parameter p is a scalar, a list with following elements :

sigma2	Estimate of the model parameter σ^2
tau	Estimate(s) of the model parameter(s) $\tau_1, \ldots, \tau_k$
Py	Last computed value of vector Py (see reference)
BLUP_omega	BLUPs of random effects
BLUP_beta	BLUPs of fixed effects β (only the components corresponding to X)
Xbeta	Estimate of $(X|PC)\beta$
varbeta	Variance matrix for β estimates (only the components corresponding to X)
varXbeta	Participation of fixed effects to variance of Y
p	Number of Principal Components included in the linear mixed model with fixed effect

If the paramer p is a vector of length > 1, a list of lists as described above, one for each value in p.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

lmm.diago.likelihood, lmm.aireml, optimize

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute Genetic Relationship Matrix
K <- GRM(x)

# eigen decomposition of K
eiK <- eigen(K)

# simulate a phenotype
set.seed(1)
y <- 1 + lmm.simu(tau = 1, sigma2 = 2, eigenK = eiK)$y

# Estimations
R <- lmm.diago(Y = y, eigenK = eiK, p = c(0,10))
str(R)
```

`lmm.diago.likelihood` *Likelihood of a linear mixed model*

Description

Compute the Restricted or the Full Likelihood of a linear mixed model, using the "diagonalization trick".

Usage

```
lmm.diago.likelihood(tau, s2, h2, Y, X, eigenK, p = 0)
lmm.diago.profile.likelihood(tau, s2, h2, Y, X, eigenK, p = 0)
```

Arguments

`tau`	Value(s) of model parameter (see Details)
`s2`	Value(s) of model parameter (see Details)
`h2`	Value(s) of heritability (see Details)
`Y`	Phenotype vector
`X`	Covariable matrix
`eigenK`	Eigen decomposition of K (a positive symmetric matrix)
`p`	Number of Principal Components included in the mixed model with fixed effect

Details

Theses function respectively compute the Restricted and the Profile Likelihood under the linear mixed model

$$Y = (X|PC)\beta + \omega + \varepsilon$$

with $\omega \sim N(0, \tau K)$ and $\varepsilon \sim N(0, \sigma^2 I_n)$.

The matrix K is given through its eigen decomposition, as produced by `eigenK = eigen(K, symmetric = TRUE)`. The matrix $(X|PC)$ is the concatenation of the covariable matrix X and of the first p eigenvectors of K, included in the model with fixed effects.

If both `tau` and `s2` (for σ^2) are provided, `lmm.diago.likelihood` computes the restricted likelihood for these values of the parameters; if these parameters are vectors of length > 1, then a matrix of likelihood values is computed.

The function `lmm.diago.profile.likelihood` computes the full likelihood, profiled for β. That is, the value β which maximizes the full likelihood for the given values of τ and σ^2 is computed, and then the full likelihood is computed.

If `h2` is provided, both functions compute τ and σ^2 which maximizes the likelihood under the constraint $\frac{\tau}{\tau+\sigma^2} = h^2$, and output these values as well as the likelihood value at this point.

Value

If `tau` and `s2` are provided, the corresponding likelihood values.

If `tau` or `s2` are missing, and `h2` is provided, a named list with members

`tau`	Corresponding values of τ
`sigma2`	Corresponding values of σ^2
`likelihood`	Corresponding likelihood values

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`lmm.restricted.likelihood`, `lmm.profile.restricted.likelihood`, `lmm.diago`, `lmm.aireml`

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute Genetic Relationship Matrix
K <- GRM(x)

# eigen decomposition of K
eiK <- eigen(K)

# simulate a phenotype
```

```
set.seed(1)
y <- 1 + lmm.simu(tau = 1, sigma2 = 2, eigenK = eiK)$y

# Likelihood
TAU <- seq(0.5,1.5,length=30)
S2 <- seq(1,3,length=30)
lik1 <- lmm.diago.likelihood(tau = TAU, s2 = S2, Y = y, eigenK = eiK)

H2 <- seq(0,1,length=51)
lik2 <- lmm.diago.likelihood(h2 = H2, Y = y, eigenK = eiK)

# Plotting
par(mfrow=c(1,2))
lik.contour(TAU, S2, lik1, heat = TRUE, xlab = "tau", ylab = "sigma^2")
lines(lik2$tau, lik2$sigma2)
plot(H2, exp(lik2$likelihood), type="l", xlab="h^2", ylab = "likelihood")
```

`lmm.restricted.likelihood`

Likelihood of a linear mixed model

Description

Compute the Restricted or the Full Likelihood of a linear mixed model.

Usage

```
lmm.restricted.likelihood(Y, X = matrix(1, nrow = length(Y)), K, tau, s2)
lmm.profile.restricted.likelihood(Y, X = matrix(1, nrow = length(Y)), K, h2)
```

Arguments

`Y`	Phenotype vector
`X`	Covariable matrix
`K`	A positive definite matrix or a `list` of such matrices
`tau`	Value(s) of parameter(s) τ
`s2`	Value of parameter σ^2
`h2`	Value(s) of heritability

Details

Theses function respectively compute the Restricted and the Profile Likelihood under the linear mixed model

$$Y = X\beta + \omega_1 + \ldots + \omega_k + \varepsilon$$

with $\omega_i \sim N(0, \tau_i K_i)$ for $i \in 1, \ldots, k$ and $\varepsilon \sim N(0, \sigma^2 I_n)$.

The variance matrices K_1, ..., K_k, are specified through the parameter `K`. The parameter `tau` should be a vector of length k.

The function `lmm.restricted.likelihood` computes the restricted likelihood for the given values of τ and σ^2. Whenever $k = 1$, it is similar to `lmm.diago.likelihood(tau, s2, Y = Y, X = X, eigenK = eigen(K))` which should be prefered (with a preliminary computation of `eigen(K)`).

The function `lmm.profile.restricted.likelihood` computes a profile restricted likelihood: the values of τ and σ^2 which maximizes the likelihood are computed under the constraint $\frac{\tau}{\tau+\sigma^2} = h^2$, and the profiled likelihood value for these parameters is computed. Whenever $k = 1$, it is similar to `lmm.diago.likelihood(h2 = h2, Y = Y, X = X, eigenK = eigen(K))`.

Value

The restricted likelihood value.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`lmm.diago.likelihood`, `lmm.diago`, `lmm.aireml`

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute Genetic Relationship Matrix and its eigen decomposition
K <- GRM(x)
eiK <- eigen(K)

# simulate a phenotype
set.seed(1)
y <- 1 + lmm.simu(tau = 1, sigma2 = 2, eigenK = eiK)$y

# compute restricted likelihood for tau = 0.2 and s2 = 0.8
lmm.restricted.likelihood(y, K=K, tau = 0.2, s2 = 0.8)

# compute profile restricted likelihood for h2 = 0.2
lmm.profile.restricted.likelihood(y, K=K, h2 = 0.2)

# identity with the values computed with the diagonalisation trick
lmm.diago.likelihood(tau = 0.2, s2 = 0.8, Y = y, eigenK = eiK)
lmm.diago.likelihood(h2 = 0.2, Y = y, eigenK = eiK)
```

`lmm.simu` *Linear mixed model data simulation*

Description

Simulate data under a linear mixed model, using the eigen decomposition of the variance matrix.

Usage

```
lmm.simu(tau, sigma2, K, eigenK = eigen(K), X, beta)
```

Arguments

`tau`	Model parameter
`sigma2`	Model parameter
`K`	(Optional) A positive symmetric matrix K
`eigenK`	Eigen decomposition of K
`X`	Covariable matrix
`beta`	Fixed effect vector of covariables

Details

The data are simulated under the following linear mixed model :

$$Y = X\beta + \omega + \varepsilon$$

with $\omega \sim N(0, \tau K)$ and $\varepsilon \sim N(0, \sigma^2 I_n)$.

The simulation uses K only through its eigen decomposition; the parameter `K` is therefore optional.

Value

A named list with two members:

`y`	Simulated value of Y
`omega`	Simulated value of ω

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`random.pm`

Examples

```
# generate a random positive matrix
set.seed(1)
R <- random.pm(503)

# simulate data with a "polygenic component"
y <-  lmm.simu(0.3, 1, eigenK = R$eigen)
str(y)
```

`logistic.mm.aireml` *Logistic mixed model fitting with Penalized Quasi-Likelihood / AIREML*

Description

Estimate the parameters of a logistic linear mixed model using the Penalized Quasi-Likelihood with an AIREML step for the linear model.

Usage

```
logistic.mm.aireml(Y, X = matrix(1, nrow = length(Y)), K,
                min_tau, tau, beta, constraint = TRUE, max.iter = 50L, eps = 1e-5,
             verbose = getOption("gaston.verbose",TRUE), get.P = FALSE, EM = FALSE)
```

Arguments

`Y`	Binary phenotype vector
`X`	Covariable matrix. By default, a column of ones to include an intercept in the model
`K`	A positive definite matrix or a `list` of such matrices
`min_tau`	Minimal value for model parameter τ (if missing, will be set to 10^{-6})
`tau`	(Optional) Optimization starting point for variance component(s) `tau`
`beta`	(Optional) Optimization starting point for fixed effect(s) `beta`
`constraint`	If `TRUE`, the model parameters respect the contraints given by `min_tau`
`max.iter`	Maximum number of iterations
`eps`	The algorithm stops when the gradient norm is lower than this parameter
`verbose`	If `TRUE`, display information on the algorithm progress
`get.P`	If `TRUE`, the function sends back the last matrix P computed in the optimization process
`EM`	If `TRUE`, the AIREML step is replaced by an EM step

Details

Estimate the parameters of the following logistic mixed model:

$$\text{logit}(P[Y = 1|X, \omega_1, \ldots, \omega_k]) = X\beta + \omega_1 + \ldots + \omega_k$$

with $\omega_i \sim N(0, \tau_i K_i)$ for $i \in 1, \ldots, k$.

The estimation is based on the Penalized Quasi-Likelihood with an AIREML step for the linear model (the algorithm is similar to the algorithm described in Chen et al 2016). If `EM = TRUE` the AIREML step is replaced by an EM step. In this case the convergence will be much slower, you're advised to use a large value of `max.iter`.

The variance matrices K_1, ..., K_k, are specified through the parameter `K`.

After convergence, the function also compute Best Linear Unbiased Predictors (BLUPs) for β and ω.

Value

A named list with members:

`tau`	Estimate(s) of the model parameter(s) $\tau_1, \ldots, \tau_k$
`niter`	Number of iterations done
`P`	Last computed value of matrix P (see reference)
`BLUP_omega`	BLUPs of random effects
`BLUP_beta`	BLUPs of fixed effects β
`varbeta`	Variance matrix for β estimates

If `get.P = TRUE`, there is an additional member:

`P`	The last matrix P computed in the AIREML step

References

Gilmour, A. R., Thompson, R., & Cullis, B. R. (1995), *Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models*, Biometrics, **1440-1450**

Chen, Han et al. (2016), *Control for Population Structure and Relatedness for Binary Traits in Genetic Association Studies via Logistic Mixed Models*, The American Journal of Human Genetics, **653–666**

See Also

`lmm.aireml`, `lmm.diago`, `lmm.simu`

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)

# Compute Genetic Relationship Matrix
```

```
standardize(x) <- "p"
K <- GRM(x)

# Simulate a quantitative genotype under the LMM
set.seed(1)
mu <- 1 + x %*% rnorm(ncol(x), sd = 2)/sqrt(ncol(x))
pi <- 1/(1+exp(-mu))
y <- 1*( runif(length(pi))<pi )

# Estimates
estimates <- logistic.mm.aireml(y, K = K, verbose = FALSE)
str(estimates)
```

`manhattan` *Manhattan plot*

Description

Draws a Manhattan plot

Usage

```
manhattan(x, bty = "n", chrom.col = c("black", "gray50"), thinning = TRUE, ... )
```

Arguments

`x`	A data.frame with columns named `chr`, `pos` and `p`.
`bty`	Type of box to draw about the plot. Default is to draw none.
`thinning`	`Logical`. If `TRUE`, not all points are displayed.
`chrom.col`	Alternating colors for chromosomes.
`...`	Graphical parameters to be passed to `plot`.

Details

If there is only one chromosome value in `x$chr`, the x-axis will be labeled with the SNP position. In the general case, the x-axis is labeled with the chromosome name and the color of the points alternates between the colors in `chrom.col`.

The default value `bty = "n"` should give the best result for GWAS Manhattan plots. See `par` for other possible values of `bty` and their meaning.

The thinning procedure suppress some points to avoid generating too heavy graphs. The user should check that setting `thinning = FALSE` does not change the final aspect of the plot.

Value

An invisible copy of `x` is returned, in which a column `coord` has been added if there is more than one chromosome value in `x$chr`. This column contains the x-coordinates of each SNP on the plot, and should prove helpful to annotate it.

See Also

association.test, qqplot.pvalues, par, plot.default, points.default

---

qqplot.pvalues *QQ plot of p-values*

Description

Draws a QQ plot of p-values

Usage

```
qqplot.pvalues(p, col.abline = "red", CB = TRUE, col.CB = "gray80",
                      CB.level = 0.95, thinning = TRUE, ...)
```

Arguments

p	A vector of p-values, or a data.frame with a column named p
col.abline	Color of the line of slope 1. Set to NA to suppress.
CB	Logical. If TRUE, a confidence band is included in the plot.
col.CB	The color of the confidence band.
CB.level	The level of the confidence band.
thinning	Logical. If TRUE, not all points are displayed.
...	Graphical parameters to be passed to plot and points

Details

The QQ plot is on the $-\log_{10}$ scale, as is usual when reporting GWAS results.

The confidence band is not a global confidence region: it is the mere juxtaposition of confidence intervals for each quantile. Moreover it assumes independance of the p-values, an hypothesis hich is false for the p-values resulting from an association test in presence of linkage disequilibrium. Therefore, the probability that some of the points lie outsite of this band is greater that CB.level.

The thinning procedure suppress some points to avoid generating too heavy graphs. The user should check that setting thinning = FALSE does not change the final aspect of the QQ plot.

See Also

association.test, manhattan, qqplot, plot.default, points.default

Examples

```
# a vector of uniform p-values
p <- runif(1e6)
qqplot.pvalues(p)
# if we don't thin the points, using pch = "." is advised
qqplot.pvalues(p, pch = ".", cex = 2, thinning = FALSE)
```

random.pm *Random square definite positive matrix*

Description

Generate a random definite positive matrix with specified dimension

Usage

```
random.pm(n, values)
```

Arguments

n	Dimension of matrix
values	(Optional) A numeric vector of dimension n : the eigenvalues of the matrix

Details

If values isn't given, it is chosen (deterministically) so that the eigenvalues of the resulting matrix are similar to eigenvalues observed on Genetic Relationship Matrices.

The random matrix is generated as $Udiag(values)U'$ with U a random orthogonal matrix.

Value

A named list with members:

K	A n x n symmetric positive matrix
eigen	The eigen decomposition of K as eigen(K) would output it

See Also

lmm.simu, eigen

Examples

```
# generate a random positive matrix
set.seed(1)
R <- random.pm(500)
str(R)
```

read.bed.matrix *Read a* bed.matrix

Description

Create a bed.matrix from a .bed file, and either a .rds file or a .bim and a .fam file.

Usage

```
read.bed.matrix(basename, bed = paste(basename, ".bed", sep=""),
                          fam = paste(basename, ".fam", sep=""),
                          bim = paste(basename, ".bim", sep=""),
                          rds = paste(basename, ".rds", sep=""),
                          verbose = getOption("gaston.verbose",TRUE))
```

Arguments

basename	Basename of all files
bed	Name of the .bed file
fam	Name of the .fam file
bim	Name of the .bim file
rds	Name of the .rds file (ignored if NULL)
verbose	If TRUE, display information on the function actions

Details

The .bed, .fam and .bim files follow the PLINK specifications (http://zzz.bwh.harvard.edu/plink/binary.shtml).

If a .rds file exists (created by write.bed.matrix), the .fam and .bim files will be ignored. To ignore an existing .rds file, set rds = NULL.

If the .bed file does not exist, and basename ends by ".bed", the function will try to generate a new basename by trimming the extension out. This allows to write read.bed.matrix("file.bed") instead of read.bed.matrix("file").

If the option gaston.auto.set.stats is set to TRUE (the default), the function set.stats will be called before returning the bed.matrix, unless a .rds file is present: in this case, the bed.matrix obtained is identical to the bed.matrix saved with write.bed.matrix.

Value

A bed.matrix

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

write.bed.matrix, set.stats

Examples

```
# Read RDS and bed files
x <- read.bed.matrix( system.file("extdata", "LCT.bed", package="gaston") )
x
```

---

read.vcf *Create a* bed.matrix *from VCF files*

Description

Create a bed.matrix from a .vcf file.

Usage

```
read.vcf(file, max.snps, get.info = FALSE, convert.chr = TRUE,
                 verbose = getOption("gaston.verbose",TRUE))
```

Arguments

file	The name of the VCF file to read
max.snps	The maximal number of SNPs to read
get.info	If TRUE, the INFO field from the VCF file will integrated in @ped$info
convert.chr	If TRUE, chromosomes ids "X", "Y" and "MT" will be converted in their numeric equivalents
verbose	If TRUE, display information on the function progress

Details

The vcf format is described in https://github.com/samtools/hts-specs

In addition to the usual data in the slot @snps, the bed.matrices produced by read.vcf have @snps$quality and @snps$filter columns corresponding to the QUAL and FILTER fields in the VCF file. If get.info = TRUE, an additionnal column @snps$info is added, corresponding to the INFO field.

The information about individuals in VCF files is incomplete: in the slot @ped, the columns @ped$famid and @ped$id will both contain the sample id; sex and phenotypes will be set to unknown.

The function currently assumes that the GT field is the first field in the genotypes format. If it is not the case, the variants are discarded.

Value

A bed.matrix

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

read.bed.matrix

Examples

```
## Read vcf file (from file name)
filepath <-system.file("extdata", "LCT.vcf.gz", package="gaston")
x1 <- read.vcf( filepath )
x1
```

---

reshape.GRM *Reshape a Genetic Relationship Matrix*

Description

Reshapes a GRM into a data frame listing relationship of (possibly all) pairs of individuals. Options are provided to specify ranges of relationship values to include or exclude. This is useful in the Quality Control process.

Usage

```
reshape.GRM(K, include = c(-Inf, +Inf), exclude)
```

Arguments

K	A symmetric matrix (such as produced by GRM)
include	Range of values to include (default is to include all values)
exclude	Range of values to exclude (default it to exclude nothing)

Details

The relationship between individuals i and j is the coefficient k_{ij} in the matrix K. The functions lists all pair i, j with $i < j$ and k_{ij} in the range defined by include and outside the range defined by exclude.

Value

A data frame with three columns named i, j, k.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

GRM

Examples

```
# load chr2 data set (~10k SNPs in low LD)
x <- read.bed.matrix( system.file("extdata", "chr2.bed", package="gaston") )

# Compute Genetic Relationship Matrix
K <- GRM(x)

# List all pairs if individuals with a relationship above 0.07
pairs <- reshape.GRM(K, exclude = c(-Inf, 0.07))

# Exclude first individual from each such pair
x1 <- x[ -pairs$i, ]
```

---

score.fixed.linear/score.fixed.logistic

Score Test for Covariates with Fixed Effects in Linear or Logistic Mixed Model

---

Description

Score Test for association between covariates and phenotype.

Usage

```
score.fixed.linear(x, Y, X = matrix(1, length(Y)), K, ...)
score.fixed.logistic(x, Y, X = matrix(1, length(Y)), K, ...)
```

Arguments

x	A matrix of covariates
Y	The phenotype vector
X	A covariable matrix. The default is a column vector of ones, to include an intercept in the model
K	A positive definite matrix or a list of such matrices
...	Optional arguments used to fit null model in lmm.aireml or logistic.mm.aireml function.

Details

The function `score.fixed.linear` considers the linear mixed model

$$Y = X\alpha + x\beta + \omega_1 + \ldots + \omega_k + \varepsilon$$

whereas the `score.fixed.logistic` function considers the following logistic model

$$\text{logit}(P[Y = 1|X, x, \omega_1, \ldots, \omega_k]) = X\alpha + x\beta + \omega_1 + \ldots + \omega_k$$

with $\omega_j \sim N(0, \tau_j K_j)$ where K_j are Genetic Relationship Matrix (GRM), $\varepsilon \sim N(0, \sigma^2 I_n)$ and fixed effects α and β.

The two functions give score test for $H_0 : \beta = 0$ vs $H_1 : \beta \neq 0$. In this aim, all parameters under null model are estimated with `lmm.aireml` or `logistic.mm.aireml`.

Value

A named list of values:

`score`	Estimated score
`p`	The corresponding p-value
`log.p`	The logarithm of corresponding p-value

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`lmm.aireml`, `logistic.mm.aireml`

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)
standardize(x) <- "p"

# Calculate GRM et its eigen decomposition
k <- GRM(x)
eig <- eigen(k)
eig$values <- round(eig$values, 5)

# generate covariate matrix
set.seed(1)
X <- cbind( rbinom(nrow(x), 1, prob=1/2), rnorm(nrow(x)) )


# simulate quantitative phenotype with polygenic component and covariate effects
y <- X %*% c(-1,0.5) + lmm.simu(0.3,1,eigenK=eig)$y

t <- score.fixed.linear(X, y, K=k, verbose=FALSE)
```

```
str(t)


# simulate binary phenotype with polygenic component and covariate effects
mu <- X %*% c(-1,0.5) + lmm.simu(1, 0, eigenK=eig)$y
pi <- 1/(1+exp(-mu))
y <- 1*( runif(length(pi))<pi )

tt <- score.fixed.logistic(X, y, K=k, verbose=FALSE)
str(tt)
```

`score.variance.linear/score.variance.logistic`
Variance Component Test in Linear or Logistic Mixed Model

Description

Test if a variance component is significaly different from 0 using score test in a Linear or Logistic Mixed Model.

Usage

```
score.variance.linear(K0, Y, X = matrix(1, length(Y)), K, acc_davies=1e-10, ...)
score.variance.logistic(K0, Y, X = matrix(1, length(Y)), K, acc_davies=1e-10, ...)
```

Arguments

`K0`	A positive definite matrix
`Y`	The phenotype vector
`X`	A covariate matrix. The default is a column vector of ones, to include an intercept in the model
`K`	A positive definite matrix or a `list` of such matrices
`acc_davies`	Accuracy in Davies method used to compute p-value
`...`	Optional arguments used to fit null model with `lmm.aireml` of `logistic.mm.aireml` function.

Details

In `score.variance.linear`, we consider the linear mixed model

$$Y = X\alpha + \gamma + \omega_1 + \ldots + \omega_k + \varepsilon$$

or, in `score.variance.logistic`, we consider the following logistic model

$$\text{logit}(P[Y = 1|X, x, \omega_1, \ldots, \omega_k]) = X\alpha + \gamma + \omega_1 + \ldots + \omega_k$$

with $\gamma \sim N(0, \kappa K_0)\gamma$, $\omega_j \sim N(0, \tau_j K_j)$, $\varepsilon \sim N(0, \sigma^2 I_n)$. K_0 and K_j are Genetic Relationship Matrix (GRM).

`score.variance.linear` and `score.variance.logistic` functions permit to test

$$H_0 : \kappa = 0 \text{ vs } H_1 : \kappa > 0$$

with, for linear mixed model, the score

$$Q = Y' P_O K_0 P_0 Y/2$$

or, for logistic mixed model, the score

$$Q = (Y - \pi_0)' K_0 (Y - \pi_0)/2$$

where P_0 is the last matrix P computed in the optimization process for null model and π_0 the vector of fitted values under null logistic model.

The associated p-value is computed with Davies method.

In this aim, all parameters under null model are estimated with `lmm.aireml` or `logistic.mm.aireml`. The p-value corresponding to the estimated score is computed using Davies method implemented in 'CompQuadForm' R package.

Value

A named list of values:

`score`	Estimated score
`p`	The corresponding p-value

Author(s)

Hervé Perdry and Claire Dandine-Roulland

References

Davies R.B. (1980) *Algorithm AS 155: The Distribution of a Linear Combination of chi-2 Random Variables*, Journal of the Royal Statistical Society. Series C (Applied Statistics), **323-333**

See Also

`lmm.aireml`, `logistic.mm.aireml`

Examples

```
# Load data
data(AGT)
x <- as.bed.matrix(AGT.gen, AGT.fam, AGT.bim)
standardize(x) <- "p"

# Calculate GRM et its eigen decomposition
K0 <- GRM(x)
eig <- eigen(K0)
eig$values <- round(eig$values, 5)
```

```
# generate an other positive matrix (to play the role of the second GRM)
set.seed(1)
R <- random.pm(nrow(x))


# simulate quantitative phenotype with two polygenic components
y <- lmm.simu(0.1,1,eigenK=eig)$y + lmm.simu(0.2,0,eigenK=R$eigen)$y

t <- score.variance.linear(K0, y, K=R$K, verbose=FALSE)
str(t)


# simulate binary phenotype with two polygenic components
mu <- lmm.simu(0.1,0.5,eigenK=eig)$y + lmm.simu(0.2,0,eigenK=R$eigen)$y
pi <- 1/(1+exp(-mu))
y <- 1*(runif(length(pi))<pi)

tt <- score.variance.logistic(K0, y, K=R$K, verbose=FALSE)
str(tt)
```

`select.inds` *Subsetting from a* `bed.matrix`

Description

Returns subset of individuals satisfying a condition.

Usage

```
select.inds(x, condition)
```

Arguments

`x`	A `bed.matrix`
`condition`	Condition used to select individuals

Details

The conditions can involve global variables and all variables defined in the data frame `x@ped`, in particular

- `famid`, `id`, `father`, `mother`, `sex`, `pheno`
- If basic stats have been computed (see `set.stats`), `N0`, `N1`, `N2`, `NAs`, `callrate`, etc.

If some condition evaluate to `NA` (e.g. `sex == 1` when `sex` is undefined for some individuals), a warning is issued and the corresponding individuals are removed.

Value

A `bed.matrix` similar to `x`, containing the selected individuals only

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

select.snps, set.stats

Examples

```
# Load data
data(LCT)
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

# Select individuals with a call rate > 95%
# and more than 5% of heterozygous genotypes
y <- select.inds(x, callrate > 0.95 & N1/(N0+N1+N2) > 0.05)
y
```

---

select.snps *Subsetting from a* bed.matrix

Description

Returns subset of SNPs satisfying a condition.

Usage

```
select.snps(x, condition)
```

Arguments

x	A bed.matrix
condition	Condition used to select SNPs

Details

The conditions can involve global variables and all variables defined in the data frame x@snps, in particular

- chr, id, dist, pos, A1, A2
- If basic stats have been computed (see set.stats), N0, N1, N2, NAs, callrate, maf, hz, etc.
- If Hardy-Weinberg Equilibrium test has been performed (see set.hwe), hwe.

If some condition evaluate to NA (e.g. maf > 0 when maf is undefined for some SNPs), a warning is issued and the corresponding SNPs are removed.

Value

A bed.matrix similar to x, containing the selected SNPs only

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

select.snps, set.stats, set.hwe

Examples

```
# Load data
data(LCT)
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

# Select SNPs with a maf > 5%
y <- select.snps(x, maf > 0.05)
y
```

---

set.dist *Set Genetic Distance*

Description

Returns an updated bed.matrix with genetic distances in centimorgan computed from the variant positions

Usage

```
set.dist(x, map, verbose = getOption("gaston.verbose", TRUE))
```

Arguments

x	A bed.matrix
map	The genetic map, given by a list of data frames (see Details)
verbose	If TRUE, display information on the function actions

Details

A map is a list of data frames, with names corresponding to chromosomes. Each of these data frames must have columns pos and dist corresponding to positions in bp and cM, respectively.

Such maps are too large to be included in a CRAN package. You can get two genetic maps for the Human Genome (build 36 and 37) in the package HumanGeneticMap on GitHub.

To install this package, run

```
install.packages("HumanGeneticMap", repos="https://genostats.github.io/R/")
```

You can then use this function with set.dist(x, HumanGeneticMap::genetic.map.b36) for example, for positions on the build 36. Use map = HumanGeneticMap::genetic.map.b37) for the build 37.

Value

A bed.matrix similar to x, with updated values in x@snps$dist.

---

set.genomic.sex *Genomic Sex*

Description

Returns an updated bed.matrix with a new variable for the genomic sex of each individual.

Usage

```
set.genomic.sex(x, plot = FALSE, verbose = getOption("gaston.verbose",TRUE))
```

Arguments

x	A bed.matrix
plot	If TRUE, plots the variables used for the clustering
verbose	If TRUE, displays information on the function actions

Details

For each individual, the function uses the hetorozygosity rate for SNPs on X chromosome, and the call rate for SNPs on the Y chromosomes (both statistics computed by set.stats), to cluster the individuals using kmeans.

If plot = TRUE, a plot is produced with the two variables used and the clusters determined by kmeans.

Value

A bed.matrix similar to x, with a new variable x@ped$genomic.sex containing the genomic sex for each individual.

Author(s)

Hervé Perdry

See Also

set.stats, set.hwe

set.hwe *Hardy-Weinberg Equilibrium*

Description

Returns an updated bed.matrix with a new variable for the p-values of an Hardy-Weinberg Equilibrium test.

Usage

```
set.hwe(x, method = c("chisquare", "exact"),
          verbose = getOption("gaston.verbose", TRUE))
```

Arguments

x	A bed.matrix
method	The method to use, either "chisquare" or "exact"
verbose	If TRUE, display information on the function actions

Details

Two tests of Hardy-Weinberg Equilibrium are proposed:

- if method = "chisquare", the good old Chi-square test
- if method = "exact", Haldane's exact test (see Wigginton et al)

The function set.stats will be called first if necessary.

The p-value is set to 1.0 for SNPs on chromosomes Y and MT. For SNPs on chromosomes X, currently, the test is performed using only the genotypic counts of women.

Value

A bed.matrix similar to x, with a new variable x@snps$hwe containing the p-values for each SNP.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

References

Wigginton, J. E., Cutler, D. J., & Abecasis, G. R. (2005). *A note on exact tests of Hardy-Weinberg equilibrium.* The American Journal of Human Genetics, **76(5)**, **887-893**

See Also

set.stats, set.genomic.sex

Examples

```
# Load data
data(LCT)
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

# Compute Hardy-Weinberg p-values
x <- set.hwe(x)
head( x@snps[,c("id","hwe")] )
```

---

set.stats *Basic statistics for a* bed.matrix

Description

Return an updated bed.matrix with new variables for several basic statistics.

Usage

```
set.stats(x, set.p = TRUE, set.mu_sigma = TRUE,
          verbose = getOption("gaston.verbose",TRUE))

set.stats.snps(x, set.p = TRUE, set.mu_sigma = TRUE,
               verbose = getOption("gaston.verbose",TRUE))

set.stats.ped(x, verbose = getOption("gaston.verbose",TRUE))
```

Arguments

x	A bed.matrix
set.p	If TRUE, x@p is updated
set.mu_sigma	If TRUE, x@mu and x@sigma are updated
verbose	If TRUE, display information on the function actions

Details

set.stats is called by default by all functions that create a bed.matrix, unless the global option gaston.auto.set.stats is FALSE (cf example below).

set.stats and set.stats.ped update x@ped, adding the following variables:

- N0, N1, N2 and NAs give for each individual the number of autosomal SNPs with a genotype equal to 0, 1, 2 and missing, respectively
- N0.x, N1.x, N2.x and NAs.x idem for chromosome X
- N0.y, N1.y, N2.y and NAs.y idem for chromosome Y
- N0.mt, N1.mt, N2.mt and NAs.mt idem for mitochondrial SNPs
- callrate, callrate.x, callrate.y, callrate.mt is the individual callrate for autosomal, X, Y, mitochondrial SNPs

- hz, hz.x, hz.y, hz.mt is the individual heterozygosity for autosomal, X, Y, mitochondrial SNPs

set.stats and set.stats.snps update x@snps, adding the following variables:

- N0, N1, N2 and NAs give for each SNP the number of individuals with a genotype equal to 0, 1, 2 and missing, respectively
- N0.f, N1.f, N2.f and NAs.f give, only for SNPs on chromosome X, the number of female individuals with a genotype equal to 0, 1, 2 and missing, respectively
- callrate is the SNP callrate (for Y linked SNPs, the callrate is computed usin males only).
- maf is the Minor Allele Frequency
- hz is the SNP heterozygosity (for X linked SNPs, the heterozygosity is computed using females only).

If set.p = TRUE, x@p is updated with the alternate allele frequency.

If set.mu_sigma = TRUE, x@mu is updated with the genotype mean (equal to 2*x@p) and x@sigma with the genotype standard deviation (should be approximately sqrt(2*x@p*(1-x@p)) under Hardy-Weinberg Equilibrium).

Value

A bed.matrix similar to x, with slots updated as described above.

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

set.hwe, set.genomic.sex

Examples

```
# Disable auto set stats :
options(gaston.auto.set.stats = FALSE)

# Load data
data(TTN)
x <- as.bed.matrix(TTN.gen, TTN.fam, TTN.bim)
str(x@ped)
str(x@snps)

# Compute statistics
x <- set.stats(x)
str(x@ped)
str(x@snps)

# restore default behavior
options(gaston.auto.set.stats = TRUE)
```

SNP.duplicated *Duplicated SNPs*

Description

Determines which SNPs are duplicates of previous SNPs and returns their indices.

Usage

```
SNP.duplicated(x, by = "chr:pos")
```

Arguments

x	A bed.matrix or a data.frame
by	The criterium used to determined if SNP is duplicated.

Details

When x is a bed.matrix, the data.frame x@bed will be used. The columns that will be taken in consideration Are id, chr, pos, A1, and A2. Not all columns are mandatory, depending on the value of by.

The possible values for by are "chr:pos", "chr:pos:alleles", "id", "id:chr:pos" and "id:chr:pos:alleles". The default is by = "chr:pos", which means that two SNPs are considered as duplicated if they have same chr and pos values.

Currently, when using a criterium involving alleles, this function does not consider the possibility of alleles swaps or reference strand flips.

Value

An integer vector of indices of SNPs which are duplicates of previously seen SNPs.

See Also

SNP.match

SNP.match *SNP matching*

Description

Returns a vector of the positions of (first) SNP matching of its first argument in its second.

Usage

```
SNP.match(x, table, by = "chr:pos:alleles")
```

Arguments

x	A bed.matrix or a data.frame
table	A bed.matrix or a data.frame
by	The criterium used to matchSNPs

Details

When x is a bed.matrix, the data.frame x@bed will be used; the same holds for table. The columns that will be taken in consideration are id, chr, pos, A1, and A2. Not all columns are mandatory (see below).

The matching criterium is specified by parameter by. There are 5 possible criteria : (i) matching by chromosome and position with by = "chr:pos", (ii) matching by chromosome, position, and alleles with by = "chr:pos:alleles", (iii) matching by id with by = "id", (iv) matching by id, chromosome and position with by = "id:chr:pos", and (v) matching by id, chromosome, position and alleles with by = "id:chr:pos:alleles".

For each SNP in x, the function looks for the position of the first matching SNP in table. If alleles are included in the matching criterium (ie if allele columns A1 and A2 are present in x), the function also checks for SNP matching with swapped alleles (a SNP A/C would match a SNP C/A), or with reference strand flipped (i.e. a SNP A/C would match a SNP T/G) or both (a SNP A/C would match a SNP G/T).

This function should prove useful for data set merging.

Value

A named list with one or three members, depending on whether alleles are included in the matching criterium.

index	An integer vector giving the position of first match in table, or NA if there is no match
swap	A logical vector indicating whether the match is with swapped alleles
flip	A logical vector indicating whether the match is with flipped strand

See Also

SNP.duplicated

SNP.rm.duplicates *Remove duplicated SNPs*

Description

Remove duplicated SNPs, taking into account possible genotype mismatches

Usage

```
SNP.rm.duplicates(x, by = "chr:pos", na.keep = TRUE, incomp.rm = TRUE)
```

Arguments

x	A bed.matrix
by	The criterium used to determine duplicates
na.keep	If TRUE, duplicated genotypes which are missing for at least one SNP are set to NA.
incomp.rm	If TRUE, duplicated SNPs with allele incompatibility are removed.

Details

Positions of duplicated SNPs are determined using SNP.duplicated using parameter by (we recommend to use "chr:pos", the default).

Then the function considers the possibility of alleles swaps or reference strand flips. In case of allele incompatibility, the SNPs can be removed or not (according to incomp.rm parameter).

When alleles can be matched, only one of the two SNPs is conserved. If there are genotype incompatibilities between the duplicates for some individuals, these genotypes are set to NA. The parameter na.keep settles the case of genotypes missing in one of the SNPs.

Moreover the function takes special care of SNP with possible alleles "0". This case occurs for monomorphic SNPs, when data are read from a .ped file; for example, a whole column of A A's will result in a SNP with alleles "A" and "0". If there's a duplicate of the SNP with a few, says, A C's in it, it will have alleles "A" and "C". In that case, SNP.duplicated with by = "chr:pos:alleles" will not consider these SNPs as duplicates.

Value

A bed.matrix without duplicated SNPs.

See Also

SNP.match, SNP.duplicated, dupli

Examples

```
# Use example data of 10 individuals with 7 duplicated SNPs
data(dupli)
x <- as.bed.matrix(dupli.gen, fam = dupli.ped, bim = dupli.bim)

# There are any duplicated positions:
dupli.bim

x1 <- SNP.rm.duplicates(x)
# By default (na.keep = TRUE), as soon as the genotype is missing
# in one of the SNPs it is set to missing
# (here looking at duplicated SNPs 2a and 2b)
as.matrix(x[,2:3])
as.matrix(x1[,2])

# With na.keep = FALSE
x2 <- SNP.rm.duplicates(x, na.keep = FALSE)
```

```
as.matrix(x2[,2])

# Let's examinate SNP 3.a and 3.b (swapped alleles)
as.matrix(x[,4:5])
as.matrix(x1[,3])
as.matrix(x2[,3])

# and so on... (see also ?dupli)
```

Tests *Evaluation of a condition on SNPS or individuals in a* bed.matrix

---

Description

Evaluate a condition and return logical vector or indices

Usage

```
test.snps(x, condition, na.to.false = TRUE)
       test.inds(x, condition, na.to.false = TRUE)
       which.snps(x, condition)
       which.inds(x, condition)
```

Arguments

x	A bed.matrix
condition	Condition used to select SNPs
na.to.false	If TRUE, NAs are replaced by FALSE

Details

The conditions can involve global variables and all variables defined in the data frame x@snps, in particular for test.snps and which.snps

- chr, id, dist, pos, A1, A2
- If basic stats have been computed (see set.stats), N0, N1, N2, NAs, callrate, maf, hz, etc.
- If Hardy-Weinberg Equilibrium test has been performed (see set.hwe), hwe.

and for test.inds and which.inds

- famid, id, father, mother, sex, pheno
- If basic stats have been computed (see set.stats), N0, N1, N2, NAs, callrate, etc.

Value

test.snps and test.inds return a logical vector of length ncol(x) and nrow(x) respectively. which.snps(x, condition) is equivalent to which(test.snps(x, condition)) and which.inds(x, condition) to which(test.inds(x, condition)).

See Also

select.snps, select.inds, set.stats, set.hwe

Examples

```
# Load data
data(LCT)
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

# SNPs and individuals with a callrate < 100%
w <- test.snps(x, callrate < 1)
table(w)
which.snps(x, callrate < 1)
which.inds(x, callrate < 1)
```

TTN *TTN data set*

Description

These data have been extracted from the 1000 Genomes data. The data set contains the genotype matrix TTN.gen, the pedigree matrix TTN.fam and a matrix TTN.bim, corresponding to 503 individuals of European populations and 733 SNPs on chromosome 2, on a ~600kb segment containing the Titin gene. There is also a factor TTN.pop, which gives the population from which each individual is drawn (CEU = Utah residents of Northern Western European ancestry, FIN = Finnish, GBR = England and Scottland, IBS = Iberian, TSI = Toscani).

Usage

```
data(TTN)
```

Format

There are three data objects in the dataset:

TTN.gen Genotype matrix

TTN.fam Data frame containing all variables corresponding to a .fam file

TTN.bim Data frame containing all variables corresponding to a .bim file

TTN.pop Factor giving the population from which each individual is drawn

Source

The data were obtained from the 1000 Genomes project (see https://www.internationalgenome.org/).

References

McVean et al, 2012, *An integrated map of genetic variation from 1,092 human genomes*, Nature **491, 56-65** doi:10.1038/nature11632

Examples

```
data(TTN)
x <- as.bed.matrix(TTN.gen, TTN.fam, TTN.bim)
x
```

`write.bed.matrix` *Save a* `bed.matrix`

Description

Save a `bed.matrix` in several files

Usage

```
write.bed.matrix(x, basename, bed = paste(basename, ".bed", sep=""),
                              fam = paste(basename, ".fam", sep=""),
                              bim = paste(basename, ".bim", sep=""),
                              rds = paste(basename, ".rds", sep=""))
```

Arguments

`x`	A `bed.matrix`
`basename`	Basename of all files
`bed`	Name of the `.bed` file
`fam`	Name of the `.fam` file
`bim`	Name of the `.bim` file
`rds`	Name of the `.rds` file

Details

If any of `bed`, `fam`, `bim` and `rds` is `NULL`, the corresponding file will not be written.

The `.fam` and `.bim` files are useful for reading files with other softwares. The `.rds` file can be read by `read.bed.matrix`.

The `.bed`, `.fam` and `.bim` files follow the PLINK specifications (`http://zzz.bwh.harvard.edu/plink/binary.shtml`).

Author(s)

Hervé Perdry and Claire Dandine-Roulland

See Also

`read.bed.matrix`, `saveRDS`

Examples

```
# Load data
data(LCT)
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

# Write object in LCT.bed and LCT.RData
## Not run:
write.bed.matrix(x, "LCT")

## End(Not run)
```

Index