Package 'ASRgenomics'

May 26, 2022

Title ASReml-R Genomics Tools

Version 1.1.0

Author c(person('Salvador', 'Gezan', email='salvador.gezan@vsni.co.uk', role=c('aut', 'cre')),
person('Darren', 'Murray', email='darren.murray@vsni.co.uk', role=c('aut', 'cre')),
person('Amanda', 'Avelar de Oliveira', email='support@vsni.co.uk', role='aut')
person('Giovanni', 'Galli', email='giovanni.galli@vsni.co.uk', role='aut')

Maintainer Salvador A. Gezan <salvador.gezan@vsni.co.uk>

Description This package presents a series of molecular and genetic routines in the R environment with the aim of assisting in analytical pipelines before and after use of ASReml-R or another library to perform analyses such as Genomic Selection (GS) or Genome-Wide Associaton Analyses (GWAS).

License MIT + file license

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

Suggests knitr,
rmarkdown,

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports data.table,
Matrix,
stats,
methods,
utils,
ggplot2,
AGHmatrix,
factoextra,
cowplot,
crayon,
superheat,
scattermore,
ellipse

R topics documented:

`ASRgenomics` *ASRgenomics: A package with complementary genomic functions*

Description

This package presents a series of molecular and genetic routines in the R environment with the aim of assisting in analytical pipelines before and after use of ASReml-R or another library to perform analyses such as Genomic Selection (GS) or Genome-Wide Association Analyses (GWAS).

The main tasks considered are:

- Preparing and exploring pedigree, phenotypic and genomic data.
- Calculating and evaluating genomic matrices and their inverse.
- Complementing and expanding results from genomic analyses.

The functions implemented consider aspects such as: filtering SNP data for quality control; assessing a kinship matrix **K** by reporting diagnostics (statistics and plots); performing Principal Component Analyses (PCA) based on kinship or SNP matrices for understanding population structure; calculating the genomic matrix **G** and its inverse and assessing their quality; matching pedigree-against genomic-based matrices; tuning up a genomic matrix (bend, blend or align); and obtaining the hybrid matrix **H** as required with single-step GBLUP (ssGBLUP).

full2sparse *Generates a sparse form matrix from a full form matrix*

Description

Modifies the input square matrix into its sparse form with three columns per line, corresponding to the set: `Row, Col, Value`. This matrix defines the lower triangle row-wise of the original matrix and it is sorted as columns within row. Values of zero on the matrix are dropped by default. Individual names should be assigned to `rownames` and `colnames`.

Usage

```
full2sparse(K = NULL, drop.zero = TRUE)
```

Arguments

`K`	A square matrix in full form ($n \times n$) (default = `NULL`).
`drop.zero`	If `TRUE` observations equal to zero are dropped (default = `TRUE`).

Details

Based on the function published by Borgognone *et al.* (2016).

Value

A matrix in sparse form with columns: `Row, Col, Value` together with the attributes `rowNames` and `colNames`. If attribute `INVERSE` is found this is also passed to the sparse matrix.

References

Borgognone, M.G., Butler, D.G., Ogbonnaya, F.C and Dreccer, M.F. 2016. Molecular marker information in the analysis of multi-environment trial helps differentiate superior genotypes from promising parents. Crop Science 56:2612-2628.

Examples

```
data(geno.apple)
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA",
              sparseform = FALSE)$G
G[1:5, 1:5]
G.sparse <- full2sparse(K = G)
head(G.sparse)
head(attr(G.sparse, "rowNames"))
head(attr(G.sparse, "colNames"))
```

`G.inverse` *Obtains the inverse of the genomic relationship matrix G*

Description

Generates the inverse of a genomic relationship matrix $\boldsymbol{G}$ that is provided. This input matrix should be of the full form ($n \times n$) with individual names assigned to `rownames` and `colnames`. Several checks for the stability of the matrix are presented based on the reciprocal conditional number.

In case of an ill-conditioned matrix, options of blending, bending or aligning before inverting are available. These options will be deprecated (discontinued) in future versions of `ASRgenomics` as they can be better implemented in the function `G.tuneup()`.

Based on procedures published by Nazarian and Gezan *et al.* (2016).

Usage

```
G.inverse(
  G = NULL,
  A = NULL,
  rcn.thr = 1e-12,
  blend = FALSE,
  pblend = 0.02,
  bend = FALSE,
  eig.tol = NULL,
  align = FALSE,
  digits = 8,
  sparseform = FALSE,
  message = TRUE
)
```

Arguments

`G`	Input of the symmetric genomic relationship matrix $\boldsymbol{G}$ in full form ($n \times n$), to obtain its inverse (default = `NULL`).
`A`	Input of the pedigree relationship matrix $\boldsymbol{A}$ to perform blending or aligning in full form. It should be of the same dimension as the $\boldsymbol{G}$ matrix ($n \times n$) (default = `NULL`) (to be deprecated).
`rcn.thr`	A threshold for identifying the $\boldsymbol{G}$ matrix as an ill-conditioned matrix. Based on the reciprocal conditional number (default = `1e-12`).
`blend`	If `TRUE` a "blending" with identity matrix $\boldsymbol{I}$ or pedigree relationship matrix $\boldsymbol{A}$ (if provided) is performed (default = `FALSE`) (to be deprecated).
`pblend`	If blending is requested this is the proportion of the identity matrix $\boldsymbol{I}$ or pedigree relationship matrix $\boldsymbol{A}$ to blend for (default = `0.02`) (to be deprecated).
`bend`	If `TRUE` a "bending" is performed by making the matrix near positive definite (default = `FALSE`) (to be deprecated).
`eig.tol`	Defines relative positiveness (*i.e.*, non-zero) of eigenvalues compared to the largest one. It determines which threshold of eigenvalues will be treated as zero (default = `NULL`) (to be deprecated).
`align`	If `TRUE` the genomic relationship matrix $\boldsymbol{G}$ is aligned to the pedigree relationship matrix $\boldsymbol{A}$ (default = `FALSE`) (to be deprecated).

digits	Set up the number of digits in used to round the output matrix (default = 8).
sparseform	If TRUE it generates an inverse matrix in sparse form to be used directly in **asreml** with required attributes (default = FALSE).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Value

A list with three of the following elements:

- Ginv: the inverse of $\boldsymbol{G}$ matrix in full form (only if sparseform = FALSE).
- Ginv.sparse: the inverse of $\boldsymbol{G}$ matrix in sparse form (only if sparseform = TRUE).
- status: the status (ill-conditioned or well-conditioned) of the inverse of $\boldsymbol{G}$ matrix.
- rcn: the reciprocal conditional number of the inverse of $\boldsymbol{G}$ matrix.

References

Nazarian A., Gezan S.A. 2016. GenoMatrix: A software package for pedigree-based and genomic prediction analyses on complex traits. Journal of Heredity 107:372-379.

Examples

```
# Example: An ill-conditioned matrix.
data(geno.apple)
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA")$G
G[1:5, 1:5]
GINV <- G.inverse(G = G, bend = FALSE, blend = FALSE, align = FALSE)
GINV$Ginv[1:5, 1:5]

# Example: An well-conditioned matrix.
data(geno.pine655)
G <- G.matrix(M = geno.pine655, method = "VanRaden", na.string = "-9")$G
G[1:5, 1:5]
GINV <- G.inverse(G = G, bend = FALSE, blend = FALSE, align = FALSE)
GINV$Ginv[1:5, 1:5]
```

G.matrix *Obtains the genomic matrix from SNP data for additive or dominant relationships*

Description

Generates the genomic numerator relationship matrix for additive (VanRaden or Yang) or dominant (Su or Vitezica) relationships. Matrix provided **M** is of the form $n \times p$, with n individuals and p markers. Individual and marker names are assigned to rownames and colnames, respectively. SNP data is coded as 0, 1, 2 (integers or decimal numbers). Missing values, if present, need to be specified.

Usage

```
G.matrix(
  M = NULL,
  method = "VanRaden",
  na.string = "NA",
  sparseform = FALSE,
  digits = 8,
  message = TRUE
)
```

Arguments

M	A matrix with SNP data of form $n \times p$, with n individuals and p markers. Individual and marker names are assigned to rownames and colnames, respectively. SNP data is coded as 0, 1, 2 (integers or decimal numbers) (default = NULL).
method	The method considered for calculation of genomic matrix. Options are: "VanRaden" and "Yang" for additive matrix, and "Su" and "Vitezica" for dominant matrix (default = "VanRaden").
na.string	A character that is interpreted as missing values (default = "NA").
sparseform	If TRUE it generates a matrix in sparse form to be used directly in **asreml** with required attributes (default = FALSE).
digits	Set up the number of digits used to round the output matrix (default = 8).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Details

Note: If data is provided with missing values, it will process calculations of relationships on pairwise non-missing data.

It uses function Gmatrix for calculations from R package **AGHmatrix** (Amadeu *et al.* 2019).

Value

A list with one of these two elements:

- G: the $\boldsymbol{G}$ matrix in full form (only if sparseform = FALSE).
- G.sparse: the $\boldsymbol{G}$ matrix in sparse form (only if sparseform = TRUE).

References

Amadeu, R.R., Cellon, C., Olmstead, J.W., Garcia, A.A.F, Resende, M.F.R. and P.R. Munoz. 2016. AGHmatrix: R package to construct relationship matrices for autotetraploid and diploid species: A blueberry example. The Plant Genome 9(3). doi: 10.3835/plantgenome2016.01.0009

Examples

```
# Example 1: Requesting a full matrix by VanRanden.
data(geno.apple)
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA")$G
G[1:5, 1:5]

# Example 2: Requesting a sparse form by VanRanden.
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA",
```

```
              sparseform = TRUE)$G.sparse
head(G)
head(attr(G, "rowNames"))
```

`G.predict` *Generates the conditional predictions of random effects (BLUPs)*

Description

Predicts random effects values for individuals with unobserved responses (here called `x`, a vector of length nx) based on known random effect values for individuals with observed responses (here called `y`, a vector of length ny). This is done using the common genomic relationship matrix $\boldsymbol{G}$ for all individuals (full matrix of dimension $(nx + ny) \times (nx + ny)$).

The prediction of unobserved responses will be performed through the multivariante Normal conditional distribution. These predictions are identical to what would be obtained if the entire set of individuals ($nx + ny$) were included into a GBLUP animal model fit with individuals in the set `x` coded as missing.

The user needs to provide the matrix $\boldsymbol{G}$ in full form. Individual names ($nx+ny$) should be assigned to `rownames` and `colnames`, and these can be in any order. If the variance-covariance matrix of the set `y` is provided, standard errors of random effects in set `x` are calculated.

Usage

```
G.predict(G = NULL, gy = NULL, vcov.gy = NULL)
```

Arguments

`G`	Input of the genomic relationship matrix $\boldsymbol{G}$ in full form (of dimension $(nx + ny) \times (nx + ny)$) (default = `NULL`).
`gy`	Input of random effects (*e.g.* breeding values) for individuals with known values. Individual names should be assigned to `rownames` of this vector and be found on the matrix $\boldsymbol{G}$ (default = `NULL`).
`vcov.gy`	The variance-covariance matrix associated with the random effects from the individuals with known values (set `y`, of dimension $ny \times ny$) (default = `NULL`).

Value

A data frame with the predicted random effect values for individuals with unobserved responses in the set `x`. If the variance-covariance matrix is provided, standard errors are included in an additional column.

Examples

```
library(asreml)

# Example 1: Apple data creating 100 missing observations
data(geno.apple)
data(pheno.apple)

# 1) Preparing G (nx+ny)
```

```
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA",
              sparseform = FALSE)$G
dim(G)

# 2) Preparing Response gy

# Select only 147 from 247 individuals from pheno.apple and geno.apple
Gy <- G[1:147, 1:147]
phenoy <- pheno.apple[1:147, ]

# Obtaining the BLUPs for the 147 individuals using ASReml-R
# Blending Gy
Gyb <- G.tuneup(G = Gy, blend = TRUE, pblend = 0.02)$Gb

# Getting the Gy inverse
Gyinv  <- G.inverse(G = Gyb, sparseform = TRUE)$Ginv.sparse

# Fitting a GBLUP model
phenoy$INDIV <- as.factor(phenoy$INDIV)
modelGBLUP <- asreml(fixed = JUI_MOT ~ 1,
                     random = ~vm(INDIV, Gyinv),
                     workspace = 128e06,
                     data = phenoy)

# Obtaining Predictions - BLUP (set y)
BLUP <- summary(modelGBLUP,coef=TRUE)$coef.random
head(BLUP)
gy <- as.matrix(BLUP[,1])
rownames(gy) <- phenoy$INDIV

# 3) Ready to make conditional predictions
g.cond <- G.predict(G = G, gy = gy)
head(g.cond)
```

`G.tuneup` *Tune-up the the genomic relationship matrix G*

Description

Generates a new matrix that can be *blended*, *bended* or *aligned* in order to make it stable for future use or matrix inversion. The input matrix should be of the full form ($n \times n$) with individual names assigned to `rownames` and `colnames`.

This routine provides three options of tune-up:

- *Blend*. The $\boldsymbol{G}$ matrix is blended (or averaged) with another matrix. $\boldsymbol{G}^* = (1-p)\boldsymbol{G} + p\boldsymbol{A}$, where $\boldsymbol{G}^*$ is the blended matrix, $\boldsymbol{G}$ is the original matrix, p is the proportion of the identity matrix or pedigree-based relationship matrix to consider, and $\boldsymbol{A}$ is the matrix to blend. Ideally, the pedigree-based relationship matrix should be used, but if this is not available (or it is of poor quality), then it is replaced by an identity matrix $\boldsymbol{I}$.
- *Bend*. It consists on adjusting the original $\boldsymbol{G}$ matrix to obtain a near positive definite matrix, which is done by making all negative or very small eigenvalues slightly positive.

- *Align*. The original $\boldsymbol{G}$ matrix is aligned to the matching pedigree relationship matrix $\boldsymbol{A}$ where an α and β parameters are obtained. More information can be found in the manual or in Christensen *et al.* (2012).

The user should provide the matrices $\boldsymbol{G}$ and $\boldsymbol{A}$ in full form ($n \times n$) and matching individual names should be assigned to the `rownames` and `colnames` of the matrices.

Based on procedures published by Nazarian and Gezan *et al.* (2016).

Usage

```
G.tuneup(
  G = NULL,
  A = NULL,
  blend = FALSE,
  pblend = 0.02,
  bend = FALSE,
  eig.tol = 1e-06,
  align = FALSE,
  rcn = TRUE,
  digits = 8,
  sparseform = FALSE,
  determinant = TRUE,
  message = TRUE
)
```

Arguments

`G`	Input of the genomic matrix $\boldsymbol{G}$ to tune-up in full form ($n \times n$) (default = `NULL`).
`A`	Input of the matching pedigree relationship matrix $\boldsymbol{A}$ in full form ($n \times n$) (default = `NULL`).
`blend`	If `TRUE` a *blending* with identity matrix $\boldsymbol{I}$ or pedigree relationship matrix $\boldsymbol{A}$ (if provided) is performed (default = `FALSE`).
`pblend`	If blending is requested this is the proportion of the identity matrix $\boldsymbol{I}$ or pedigree-based relationship matrix $\boldsymbol{A}$ to blend for (default = `0.02`).
`bend`	If `TRUE` a *bending* is performed by making the matrix near positive definite (default = `FALSE`).
`eig.tol`	Defines relative positiveness (*i.e.*, non-zero) of eigenvalues compared to the largest one. It determines which threshold of eigenvalues will be treated as zero (default = `1e-06`).
`align`	If `TRUE` the genomic matrix $\boldsymbol{G}$ is *aligned* to the matching pedigree relationship matrix $\boldsymbol{A}$ (default = `FALSE`).
`rcn`	If `TRUE` the reciprocal conditional number of the original and the bended, blended or aligned matrix will be calculated (default = `TRUE`).
`digits`	Set up the number of digits used to round the output matrix (default = `8`).
`sparseform`	If `TRUE` it generates an inverse matrix in sparse form to be used directly in **asreml** with required attributes (default = `FALSE`).
`determinant`	If `TRUE` the determinant will be calculated, otherwise, this is obtained for matrices of a dimension of less than 1,500 $\times$ 1,500 (default = `TRUE`).
`message`	If `TRUE` diagnostic messages are printed on screen (default = `TRUE`).

Value

A list with six of the following elements:

- `Gb`: the inverse of $\boldsymbol{G}$ matrix in full form (only if sparseform = `FALSE`).
- `Gb.sparse`: if requested, the inverse of $\boldsymbol{G}$ matrix in sparse form (only if sparseform = `TRUE`).
- `rcn0`: the reciprocal conditional number of the original matrix. Values near zero are associated with an ill-conditioned matrix.
- `rcnb`: the reciprocal conditional number of the blended, bended or aligned matrix. Values near zero are associated with an ill-conditioned matrix.
- `det0`: if requested, the determinant of the original matrix.
- `blend`: if the matrix was *blended*.
- `bend`: if the matrix was *bended*.
- `align`: if the matrix was *aligned*.

References

Christensen, O.F., Madsen, P., Nielsen, B., Ostersen, T. and Su, G. 2012. Single-step methods for genomic evaluation in pigs. Animal 6:1565-1571. doi:10.1017/S1751731112000742.

Nazarian A., Gezan S.A. 2016. GenoMatrix: A software package for pedigree-based and genomic prediction analyses on complex traits. Journal of Heredity 107:372-379.

Examples

```
# Example 1: Apple dataset.
# Original G matrix.
data(geno.apple)
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA")$G
G[1:5, 1:5]

# 1.2 Blended G matrix.
G_blended <- G.tuneup(G = G, blend = TRUE, pblend = 0.05)
G_blended$Gb[1:5, 1:5]

# 1.3 Bended G matrix.
G_bended <- G.tuneup(G = G, bend = TRUE, eig.tol = 1e-03)
G_bended$Gb[1:5, 1:5]

# Example 2 - Loblolly Pine dataset with pedigree - Aligned G matrix.
data(geno.pine655)
data(ped.pine)
head(ped.pine)
A <- AGHmatrix::Amatrix(ped.pine)
dim(A)
M_filter <- qc.filtering(
   M = geno.pine655, base = FALSE, ref = NULL, maf = 0.05, marker.callrate = 0.2,
   ind.callrate = 0.20, impute = FALSE, na.string = "-9", plots = TRUE)
G <- G.matrix(M = M_filter$M.clean, method = "VanRaden", na.string = "-9")$G
G[1:5, 1:5]
dim(G)
Aclean <- match.G2A(A = A, G = G, clean = TRUE, ord = TRUE, mism = TRUE)$Aclean
G_align <- G.tuneup(G = G, A = Aclean, align = TRUE)
G_align$Gb[1:5, 1:5]
```

geno.apple *Genotypic data for apple dataset*

Description

Genotypic data on 247 apple clones (*i.e.*, genotypes) with a total of 2,828 SNP markers (coded as 0, 1, 2 and there are no missing records). Dataset obtained from supplementary material in Kumar *et al.* (2015).

Usage

```
data(geno.apple)
```

Format

matrix

References

Kumar S., Molloy C., Muñoz P., Daetwyler H., Chagné D., and Volz R. 2015. Genome-enabled estimates of additive and nonadditive genetic variances and prediction of apple phenotypes across environments. G3 Genes, Genomes, Genetics 5:2711-2718.

Examples

```
data(geno.apple)
```

geno.pine655 *Genotypic data of 655 genotypes for loblolly pine dataset*

Description

Genotypic data for a total of 4,853 SNPs (coded as 0, 1, 2 and -9 for missing) on 655 genotypes of Loblolly Pine (*Pinus taeda* L.). Dataset modified from supplementary material from Resende *et al.* (2012). This dataset differs from the original as some genotypes were made artificially missing by full-sib family.

Usage

```
data(geno.pine655)
```

Format

matrix

References

Resende, M.F.R., Munoz, P. Resende, M.D.V., Garrick, D.J., Fernando, R.L., Davis, J.M., Jokela, E.J., Martin, T.A., Peter, G.F., and Kirst, M. 2012. Accuracy of genomic selection methods in a standard data set of loblolly pine (*Pinus taeda* L.). Genetics 190:1503-1510.

Examples

```
data(geno.pine655)
```

`geno.pine926` *Genotypic data of 926 genotypes for loblolly pine dataset*

Description

Genotypic data for a total of 4,853 SNPs (coded as 0, 1, 2 and -9 for missing) on 926 genotypes of Loblolly Pine (*Pinus taeda* L.). Dataset obtained from supplementary material in Resende *et al.* (2012).

Usage

```
data(geno.pine926)
```

Format

matrix

References

Resende, M.F.R., Munoz, P. Resende, M.D.V., Garrick, D.J., Fernando, R.L., Davis, J.M., Jokela, E.J., Martin, T.A., Peter, G.F., and Kirst, M. 2012. Accuracy of genomic selection methods in a standard data set of loblolly pine (*Pinus taeda* L.). Genetics 190:1503-1510.

Examples

```
data(geno.pine926)
```

`geno.salmon` *Genotypic data for Atlantic salmon dataset*

Description

Genotypic data on 1,481 Atlantic salmon samples. A total of 17,156 SNP markers (coded as 0, 1, 2 and `NA` for missing) are included in this dataset. Dataset obtained from supplementary material in Robledo *et al.* (2018).

Usage

```
data(geno.salmon)
```

Format

matrix

References

Robledo D., Matika O., Hamilton A., and Houston R.D. 2018. Genome-wide association and genomic selection for resistance to amoebic gill disease in Atlantic salmon. G3 Genes, Genomes, Genetics 8:1195-1203.

Examples

```
data(geno.salmon)
```

`H.inverse` *Generates the inverse of the hybrid H matrix*

Description

The single-step GBLUP approach combines the information from the pedigree relationship matrix $\boldsymbol{A}$ and the genomic relationship matrix $\boldsymbol{G}$ in one hybrid relationship matrix called $\boldsymbol{H}$. This function will calculate directly the inverse of this matrix $\boldsymbol{H}$. The user should provide the matrices $\boldsymbol{A}$ or its inverse (only one of these is required) and the inverse of the matrix $\boldsymbol{G}$ ($\boldsymbol{G_{inv}}$) in its full form. Individual names should be assigned to `rownames` and `colnames`, and individuals from $\boldsymbol{G_{inv}}$ are verified to be all a subset within individuals from $\boldsymbol{A}$ (or $\boldsymbol{A_{inv}}$).

Usage

```
H.inverse(
  A = NULL,
  Ainv = NULL,
  Ginv = NULL,
  lambda = NULL,
  tau = 1,
  omega = 1,
  sparseform = FALSE,
  keep.order = TRUE,
  digits = 8,
  inverse = TRUE,
  message = TRUE
)
```

Arguments

`A`	Input of the pedigree relationship matrix $\boldsymbol{A}$ in full form ($na \times na$) (default = `NULL`).
`Ainv`	Input of the inverse of the pedigree relationship matrix $\boldsymbol{A}^{-1}$ in full form ($na \times na$) (default = `NULL`).
`Ginv`	Input of the inverse of the genomic relationship matrix $\boldsymbol{G}^{-1}$ in full form ($ng \times ng$) (default = `NULL`).
`lambda`	The scaling factor for $(\boldsymbol{G}^{-1} - \boldsymbol{A}_{22}^{-1})$ (default = `NULL`).
`tau`	The scaling factor for $\boldsymbol{G}^{-1}$ (default = 1).
`omega`	The scaling factor for $\boldsymbol{A}_{22}^{-1}$ (default = 1).

sparseform	If TRUE it generates the requested matrix in sparse form to be used directly in **asreml** with required attributes (default = FALSE).
keep.order	If TRUE the original order of the individuals from the $\boldsymbol{A}$ or $\boldsymbol{A_{inv}}$ matrix is kept. Otherwise the non-genotyped individuals are placed first and then genotyped individuals (default = TRUE).
digits	Set up the number of digits used to round the output matrix (default = 8).
inverse	If TRUE it generates the inverse of $\boldsymbol{H}$ matrix (default = TRUE) (to be deprecated).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Details

The generation of the $\boldsymbol{H^{-1}}$ matrix contains a few scaling factors to help with the calculation of this inverse and to allow further exploration of the combination of the information from the $\boldsymbol{A^{-1}}$ and $\boldsymbol{G^{-1}}$. We follow the specifications described by Martini *et. al* (2018), which is done by specifying the parameters λ, or the pair τ and ω.

If inverse = FALSE the $\boldsymbol{H}$ matrix is provided instead of its inverse. This option will be deprecated and it is better to use the function H.matrix.

Value

The inverse of the hybrid matrix $\boldsymbol{H}$ matrix, in full or sparse form with required attributes to be used in **asreml**.

References

Christensen, O.F., Lund, M.S. 2010. Genomic prediction matrix when some animals are not genotyped. Gen. Sel. Evol. 42(2):1–8.

Christensen, O., Madsen, P., Nielsen, B., Ostersen, T., and Su, G. 2012. Single-step methods for genomic evaluation in pigs. Animal 6(10):1565–1571.

Legarra, A., Aguilar, I., and Misztal, I. 2009. A relationship matrix including full pedigree and genomic information. J. Dairy Sci. 92:4656-4663.

Martini, J.W.R., Schrauf, M.F., Garcia-Baccino, C.A., Pimentel, E.C.G., Munilla, S., Rogberg-Muñoz, A., Cantet, R.J.C., Reimer, C., Gao, N., Wimmer, V., and Simianer, H. 2018. The effect of the H^{-1} scaling factors τ and ω on the structure of H in the single-step procedure. Genet. Sel. Evol. 50:1-9.

Examples

```
## Not run:
# Example: Pine data.
data(ped.pine)
data(geno.pine655)

# Getting A matrix.
A <- AGHmatrix::Amatrix(data = ped.pine)
A[1:5,1:5]
dim(A)

# Reading genotypic data and making some filters.
M_filter <- qc.filtering(
   M = geno.pine655, base = FALSE, ref = NULL, maf = 0.05, marker.callrate = 0.2,
   ind.callrate = 0.20, impute = FALSE, na.string = "-9", plots = TRUE)
```

```
# Getting G matrix.
G <- G.matrix(M = M_filter$M.clean, method = "VanRaden", na.string = "-9")$G
G[1:5, 1:5]
dim(G)

# Match G2A.
check <- match.G2A(A = A, G = G, clean = TRUE, ord = TRUE, mism = TRUE,
                   RMdiff = TRUE)

# Align G matrix.
G_align <- G.tuneup(G = check$Gclean, A = check$Aclean, align = TRUE,
                    sparseform = FALSE)$Gb

# Getting Ginverse using the G aligned.
Ginv <- G.inverse(G = G_align, sparseform = FALSE)$Ginv
Ginv[1:5, 1:5]
dim(Ginv)

# Obtaining Hinv.
Hinv <- H.inverse(A = A, G = Ginv, lambda = 0.90, sparseform = TRUE)
head(Hinv)
attr(Hinv, "INVERSE")

## End(Not run)
```

`H.matrix` *Generates the hybrid H matrix*

Description

The single-step GBLUP approach combines the information from the pedigree relationship matrix $\boldsymbol{A}$ and the genomic relationship matrix $\boldsymbol{G}$ in one hybrid relationship matrix called $\boldsymbol{H}$. This function will calculate directly this matrix $\boldsymbol{H}$. The user should provide the matrices $\boldsymbol{A}$ or its inverse (only one of these is required) and the inverse of the matrix $\boldsymbol{G}$ ($\boldsymbol{G_{inv}}$) in its full form. Individual names should be assigned to `rownames` and `colnames`, and individuals from $\boldsymbol{G_{inv}}$ are verified to be all a subset within individuals from $\boldsymbol{A}$ (or $\boldsymbol{A_{inv}}$). This function is a wrapper of the H.inverse function.

Usage

```
H.matrix(
  A = NULL,
  Ainv = NULL,
  Ginv = NULL,
  lambda = NULL,
  tau = 1,
  omega = 1,
  sparseform = FALSE,
  keep.order = TRUE,
  digits = 8,
  message = TRUE
)
```

Arguments

A	Input of the pedigree relationship matrix $\boldsymbol{A}$ in full form ($na \times na$) (default = NULL).
Ainv	Input of the inverse of the pedigree relationship matrix $\boldsymbol{A}^{-1}$ in full form ($na \times na$) (default = NULL).
Ginv	Input of the inverse of the genomic relationship matrix $\boldsymbol{G}^{-1}$ in full form ($ng \times ng$) (default = NULL).
lambda	The scaling factor for $(\boldsymbol{G}^{-1} - \boldsymbol{A}_{22}^{-1})$ (default = NULL).
tau	The scaling factor for $\boldsymbol{G}^{-1}$ (default = 1).
omega	The scaling factor for $\boldsymbol{A}_{22}^{-1}$ (default = 1).
sparseform	If TRUE it generates the requested matrix in sparse form to be used directly in **asreml** with required attributes (default = FALSE).
keep.order	If TRUE the original order of the individuals from the $\boldsymbol{A}$ or $\boldsymbol{A_{inv}}$ matrix is kept. Otherwise the non-genotyped individuals are placed first and then genotyped individuals (default = TRUE).
digits	Set up the number of digits used to round the output matrix (default = 8).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Details

This function is currently equivalent to using H.inverse with (inverse = FALSE).

Value

The hybrid matrix $\boldsymbol{H}$ matrix, in full or sparse form.

References

Christensen, O.F., Lund, M.S. 2010. Genomic prediction matrix when some animals are not genotyped. Gen. Sel. Evol. 42(2):1–8.

Christensen, O., Madsen, P., Nielsen, B., Ostersen, T., and Su, G. 2012. Single-step methods for genomic evaluation in pigs. Animal 6(10):1565–1571.

Legarra, A., Aguilar, I., and Misztal, I. 2009. A relationship matrix including full pedigree and genomic information. J. Dairy Sci. 92:4656-4663.

Martini, J.W.R., Schrauf, M.F., Garcia-Baccino, C.A., Pimentel, E.C.G., Munilla, S., Rogberg-Muñoz, A., Cantet, R.J.C., Reimer, C., Gao, N., Wimmer, V., and Simianer, H. 2018. The effect of the H^{-1} scaling factors τ and ω on the structure of H in the single-step procedure. Genet. Sel. Evol. 50:1-9.

Examples

```
## Not run:
# Example: Pine data.
data(ped.pine)
data(geno.pine655)

# Getting A matrix.
A <- AGHmatrix::Amatrix(data = ped.pine)
A[1:5,1:5]
dim(A)
```

```
# Reading genotypic data and making some filters.
M_filter <- qc.filtering(
   M = geno.pine655, base = FALSE, ref = NULL, maf = 0.05, marker.callrate = 0.2,
   ind.callrate = 0.20, impute = FALSE, na.string = "-9", plots = TRUE)

# Getting G matrix.
G <- G.matrix(M = M_filter$M.clean, method = "VanRaden", na.string = "-9")$G
G[1:5, 1:5]
dim(G)

# Match G2A.
check <- match.G2A(A = A, G = G, clean = TRUE, ord = TRUE, mism = TRUE,
                   RMdiff = TRUE)

# Align G matrix.
G_align <- G.tuneup(G = check$Gclean, A = check$Aclean, align = TRUE,
                    sparseform = FALSE)$Gb

# Getting Ginverse using the G aligned.
Ginv <- G.inverse(G = G_align, sparseform = FALSE)$Ginv
Ginv[1:5, 1:5]
dim(Ginv)

# Obtaining Hinv.
H <- H.matrix(A = A, G = Ginv, lambda = 0.90, sparseform = FALSE)
H[1:5, 1:5]

## End(Not run)
```

`kinship.diagnostics` *Reports summary statistics, plots and filter options for a given kinship matrix K*

Description

It reports summary statistics, plots and allows for some filter options for diagonal and off-diagonal elements for a given kinship matrix. The input matrix can be a pedigree-based relationship matrix $\boldsymbol{A}$, a genomic relationship matrix $\boldsymbol{G}$ or a hybrid relationship matrix $\boldsymbol{H}$. Individual names should be assigned to `rownames` and `colnames`.

Usage

```
kinship.diagnostics(
  K = NULL,
  diagonal.thr.large = 1.2,
  diagonal.thr.small = 0.8,
  duplicate.thr = 0.95,
  clean.diagonal = FALSE,
  clean.duplicate = FALSE,
  plots = TRUE,
  sample.plot = 1,
  message = TRUE
)
```

Arguments

K	Input of a kinship matrix in full format ($n \times n$) (default = NULL).
diagonal.thr.large	A threshold value to flag large diagonal values (default = 1.2).
diagonal.thr.small	A threshold value to flag small diagonal values (default = 0.8).
duplicate.thr	A threshold value to flag possible duplicates. Any calculation larger than the threshold based on $\boldsymbol{k}_{i,i} \ / \ \sqrt{\boldsymbol{k}_{i,i} \times \boldsymbol{k}_{j,j}}$ is identified as a duplicate (default = 0.95).
clean.diagonal	If TRUE returns a kinship matrix filtered by values smaller than diagonal.thr.large and larger than diagonal.thr.small (default = FALSE).
clean.duplicate	If TRUE return a kinship matrix without the flagged duplicate individuals (default = FALSE).
plots	If TRUE generates graphical output of the diagonal and off-diagonal values of the kinship matrix (default = TRUE).
sample.plot	A numeric value between 0 and 1 indicating the proportion of the data points to be sampled for fast plotting of off-diagonal values. Note that for proportions other than 1, the method is not exact and low proportions are only recommended for large kinship matrices (default = 1).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Value

A list with the following elements:

- list.diagonal: a data frame with the list of flagged large or small diagonal values.
- list.duplicate: a data frame with the list of possible duplicates.
- clean.kinship: output of kinship matrix filtered without the flagged diagonal and/or duplicate individuals.
- plot.diagonal: histogram with the distribution of diagonal values from the kinship matrix.
- plot.offdiag: histogram with the distribution of off-diagonal values from kinship matrix.

Examples

```
# Example 1: Apple dataset.
data(geno.apple)
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA")$G

G_summary <- kinship.diagnostics(
    K = G, diagonal.thr.large = 1.3, diagonal.thr.small = 0.7,
    duplicate.thr = 0.8, clean.diagonal = TRUE, clean.duplicate = TRUE)
ls(G_summary)
dim(G_summary$clean.kinship)
G_summary$clean.kinship[1:5, 1:5]
G_summary$list.duplicate
G_summary$list.diagonal
G_summary$plot.diag
G_summary$plot.offdiag

# Example 2: Apple dataset - sampling 50% of points
```

```
G_summ.sample <- kinship.diagnostics(
    K = G, diagonal.thr.large = 1.3, diagonal.thr.small = 0.7,
    duplicate.thr = 0.8, clean.diagonal = TRUE, clean.duplicate = TRUE,
    sample.plot = 0.50)
G_summ.sample$plot.offdiag
```

`kinship.heatmap` *Enhanced heatmap plot for a kinship matrix K*

Description

Generates a heatmap with dendrogram based on a provided kinship matrix. This matrix can be a pedigree relationship matrix $\boldsymbol{A}$, a genomic relationship matrix $\boldsymbol{G}$ or a hybrid relationship matrix $\boldsymbol{H}$. Individual names should be assigned to `rownames` and `colnames`. It sorts individuals according to dendrogram in both columns and rows.

Usage

```
kinship.heatmap(
  K = NULL,
  dendrogram = TRUE,
  clustering.method = c("hierarchical", "kmeans"),
  dist.method = c("euclidean", "maximum", "manhattan", "canberra", "binary",
    "minkowski"),
  row.label = TRUE,
  col.label = FALSE
)
```

Arguments

`K`	Input of a kinship matrix in full format ($n \times n$) (default = `NULL`).
`dendrogram`	If `TRUE` a dendrogram is added to the columns based on the kinship matrix (default = `TRUE`).
`clustering.method`	The clustering method considered for the dendrogram. Options are: `"hierarchical"` and `"kmeans"` (default = `"hierarchical"`).
`dist.method`	The method considered to calculate the distance matrix between individuals used for hierarchical clustering. Options are: `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"` and `"minkowski"` (default = `"euclidean"`).
`row.label`	If `TRUE` the individual names (`rownames`) are added as labels to the left of the heatmap (default = `TRUE`).
`col.label`	If `TRUE` the individual names (`colnames`) are added as labels to the bottom of the heatmap (default = `FALSE`).

Details

Uses the library `superheat` from Barter and Yu (2018) to generate plots.

Value

A plot with the properties specified by the above arguments.

References

Barter, R.L. and Yu, B. 2018. Superheat: An R package for creating beautiful and extendable heatmaps for visualizing complex data. J. Comput. Graph. Stat. 27(4):910-922.

Examples

```
data(geno.apple)
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string="NA")$G
G[1:5, 1:5]

# Plotting a subset of the individuals
kinship.heatmap(K = G[1:30, 1:30], dendrogram = TRUE, row.label = TRUE, col.label = TRUE)

# Plotting the full set of individuals
kinship.heatmap(K = G, dendrogram = TRUE, row.label = FALSE, col.label = FALSE)
```

---

kinship.pca *Performs a Principal Component Analysis (PCA) based on a kinship matrix K*

Description

Generates a PCA and summary statistics from a given kinship matrix for population structure. This matrix can be a pedigree-based relationship matrix $\boldsymbol{A}$, a genomic relationship matrix $\boldsymbol{G}$ or a hybrid relationship matrix $\boldsymbol{H}$. Individual names should be assigned to `rownames` and `colnames`. There is additional output such as plots and other data frames to be used on other downstream analyses (such as GWAS).

Usage

```
kinship.pca(
  K = NULL,
  scale = TRUE,
  label = FALSE,
  ncp = 10,
  groups = NULL,
  ellipses = FALSE
)
```

Arguments

`K`	Input of a kinship matrix in full form ($n \times n$) (default = `NULL`).
`scale`	If `TRUE` the PCA analysis will scale the kinship matrix, otherwise it is used in its original scale (default = `TRUE`).
`label`	If `TRUE` then includes in output individuals names (default = `FALSE`).

ncp	The number of PC dimensions to be shown in the screeplot, and to provide in the output data frame (default = 10).
groups	Specifies a vector of class factor that will be used to define different colors for individuals in the PCA plot. It must be presented in the same order as the individuals in the kinship matrix (default = NULL).
ellipses	If TRUE, ellipses will will be drawn around each of the define levels in groups (default = FALSE).

Details

It calls function `eigen()` to obtain eigenvalues and later generate the PCA and the `factoextra` R package to extract and visualize results.

Value

A list with the following four elements:

- `eigenvalues`: a data frame with the eigenvalues and its variances associated with each dimension including only the first `ncp` dimensions.
- `pca.scores`: a data frame with scores (rotated observations on the new components) including only the first `ncp` dimensions.
- `plot.pca`: a scatterplot with the first two-dimensions (PC1 and PC2) and their scores.
- `plot.scree`: a barchart with the percentage of variances explained by the `ncp` dimensions.

Examples

```
# Apple dataset
data(geno.apple)
data(pheno.apple)

# Get the G matrix.
G <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA")$G
G[1:5, 1:5]

# Perform the PCA.
G_pca <- kinship.pca(K = G, ncp = 10)
ls(G_pca)
G_pca$eigenvalues
head(G_pca$pca.scores)
G_pca$plot.pca
G_pca$plot.scree

# PCA plot by family (17 groups).
grp <- as.factor(pheno.apple$Family)
G_pca_grp <- kinship.pca(K = G, groups = grp, label = FALSE, ellipses = TRUE)
G_pca_grp$plot.pca
```

`match.G2A` *Check the genomic relationship matrix G against the pedigree relationship matrix A or vice versa*

Description

Assesses a given genomic relationship matrix $\boldsymbol{G}$ against the pedigree relationship matrix $\boldsymbol{A}$, or vice versa, to determine the matched and mismatched individuals. If requested, it provides the cleaned versions containing only the matched individuals between both matrices. The user should provide the matrices $\boldsymbol{G}$and $\boldsymbol{A}$ in full form ($ng \times ng$ and $na \times na$, respectively). Individual names should be assigned to `rownames` and `colnames` for both matrices.

Usage

```
match.G2A(
  A = NULL,
  G = NULL,
  clean = TRUE,
  ord = TRUE,
  mism = FALSE,
  RMdiff = FALSE,
  message = TRUE
)
```

Arguments

`A`	Input of the pedigree relationship matrix $\boldsymbol{A}$ in full form ($na \times na$) (default = `NULL`).
`G`	Input of the genomic relationship matrix $\boldsymbol{G}$ in full form ($ng \times ng$) (default = `NULL`).
`clean`	If `TRUE` generates new clean $\boldsymbol{G}$ and $\boldsymbol{A}$ matrices in full form containing only matched individuals (default = `TRUE`).
`ord`	If `TRUE` it will order by ascending order of individual names both of the clean $\boldsymbol{A}$ and $\boldsymbol{G}$ matrices (default = `TRUE`).
`mism`	If `TRUE` generates two data frames with mismatched individual names from the $\boldsymbol{G}$ and $\boldsymbol{A}$ matrices (default = `FALSE`).
`RMdiff`	If `TRUE` it generates the matrix (in lower diagonal row-wise sparse form) of matched observations from both the $\boldsymbol{G}$ and $\boldsymbol{A}$ matrices. This matrix can be used to identify inconsistent values between matched matrices, but it can be very large (default = `FALSE`).
`message`	If `TRUE` diagnostic messages are printed on screen (default = `TRUE`).

Value

A list with the following elements:

- `Gclean`: a matrix with the portion of $\boldsymbol{G}$ containing only matched individuals.
- `Aclean`: a matrix with the portion of $\boldsymbol{A}$ containing only matched individuals.

- `mismG`: a vector containing the names of the individuals from matrix $\boldsymbol{G}$ that are missing in matrix $\boldsymbol{A}$.
- `mismA`: a vector containing the names of the individuals from matrix $\boldsymbol{A}$ that are missing in matrix $\boldsymbol{G}$.
- `RM`: a data frame with the observations from both the $\boldsymbol{G}$ and $\boldsymbol{A}$ matched matrices, together with their absolute relationship difference.
- `plotG2A`: scatterplot with the pairing of matched pedigree- against genomic-based relationship values. This graph might take a long to plot with large datasets.

Examples

```
# Example: Pine data.
data(ped.pine)
data(geno.pine655)

# Getting A matrix.
A <- AGHmatrix::Amatrix(data = ped.pine)
A[1:5,1:5]
dim(A)

# Reading genotypic data and making some filters.
M_filter <- qc.filtering(
   M = geno.pine655, base = FALSE, ref = NULL, maf = 0.05, marker.callrate = 0.2,
   ind.callrate = 0.20, impute = FALSE, na.string = "-9", plots = TRUE)

# Getting G matrix.
G <- G.matrix(M = M_filter$M.clean, method = "VanRaden", na.string = "-9")$G
G[1:5, 1:5]
dim(G)

# Match G2A.
check <- match.G2A(A = A, G = G, clean = TRUE, ord = TRUE, mism = TRUE,
                   RMdiff = TRUE)
ls(check)
dim(check$Aclean)
dim(check$Gclean)
check$Aclean[1:5, 1:5]
check$Gclean[1:5, 1:5]
head(check$mismG)
head(check$mismA)
check$plotG2A
head(check$RM)
```

`match.kinship2pheno` *Check any kinship matrix K against phenotypic data*

Description

Assesses a given kinship matrix against the provided phenotypic data to determime if all genotypes are in the kinship matrix or not. It also reports which individuals match or are missing from one set or another. If requested, a reduced kinship matrix is generated that has only the matched individuals. The input kinship matrix can be a pedigree-based relationship matrix $\boldsymbol{A}$, a genomic-based

relationship matrix $\boldsymbol{G}$, or a hybrid relationship matrix $\boldsymbol{H}$. Individual names should be assigned to `rownames` and `colnames` of input matrix.

Usage

```
match.kinship2pheno(
  K = NULL,
  pheno.data = NULL,
  indiv = NULL,
  clean = FALSE,
  ord = TRUE,
  mism = FALSE,
  message = TRUE
)
```

Arguments

`K`	Input of a kinship matrix in full form ($n \times n$) (default = `NULL`);
`pheno.data`	A data fame with the phenotypic data to assess (for `n` individuals) (default = `NULL`).
`indiv`	The string for the column name for genotypes/individuals in the phenotypic data (default = `NULL`).
`clean`	If `TRUE`, generates a new clean kinship matrix containing only the matched phenotyped individuals (default = `FALSE`).
`ord`	If `TRUE`, it will order the kinship matrix as in the phenotypic data, which is recommended for some downstream genomic analyses (default = `TRUE`).
`mism`	If `TRUE`, it generates data frames with matched and mismatched individual's names from the kinship matrix and the phenotypic data (default = `FALSE`).
`message`	If `TRUE` diagnostic messages are printed on screen (default = `TRUE`).

Value

A list with the following elements:

- `mismatchesK`: a vector containing the names of the individuals from the provided kinship matrix that *mismatch* with the phenotypic data.
- `matchesK`: a vector containing the names of the individuals from the provided kinship matrix that *match* with the phenotypic data.
- `mismatchesP`: a vector containing the names of phenotyped individuals that *mismatch* with those from the kinship matrix.
- `matchesP`: a vector containing the names of phenotyped individuals that *match* with those from the kinship matrix.
- `Kclean`: a clean kinship matrix containing only the matched phenotyped individuals.

Examples

```
# Pine data
data(geno.pine655)
data(pheno.pine)
dim(pheno.pine)
head(pheno.pine)
```

```
# Get the G matrix.
G <- G.matrix(M = geno.pine655, method = "VanRaden", na.string = "-9",
              sparseform = FALSE)$G
dim(G)

# Match G and the phenotype.
check <- match.kinship2pheno(K = G, pheno.data = pheno.pine, indiv = "Genotype",
                             clean = TRUE, mism = TRUE)
ls(check)
length(check$matchesK)
length(check$mismatchesK)
length(check$matchesP)
length(check$mismatchesP)
dim(check$Kclean)
```

`ped.pine` *Pedigree data for loblolly pine dataset*

Description

Individual pedigree data for a total of 2,034 records of loblolly pine (*Pinus taeda* L.). Missing parental information coded as 0. Dataset obtained from supplementary material in Resende *et al.* (2012).

Usage

```
data(ped.pine)
```

Format

data.frame

References

Resende, M.F.R., Munoz, P. Resende, M.D.V., Garrick, D.J., Fernando, R.L., Davis, J.M., Jokela, E.J., Martin, T.A., Peter, G.F., and Kirst, M. 2012. Accuracy of genomic selection methods in a standard data set of loblolly pine (*Pinus taeda* L.). Genetics 190:1503-1510.

Examples

```
data(ped.pine)
```

ped.salmon *Pedigree data for Atlantic salmon dataset*

Description

Pedigree data of 1,481 Atlantic salmon samples. Missing parental information coded as 0. Dataset obtained from supplementary material in Robledo *et al.* (2018).

Usage

```
data(ped.salmon)
```

Format

data.frame

References

Robledo D., Matika O., Hamilton A., and Houston R.D. 2018. Genome-wide association and genomic selection for resistance to amoebic gill disease in Atlantic salmon. G3 Genes, Genomes, Genetics 8:1195-1203.

Examples

```
data(ped.salmon)
```

pheno.apple *Phenotypic data for apple dataset*

Description

Phenotypic data on 247 apple clones (*i.e.*, genotypes) evaluated for several fruit quality traits at two New Zealand sites, Motueka (MOT) and Hawkes Bay (HB). Dataset obtained from supplementary material in Kumar *et al.* (2015).

Usage

```
data(pheno.apple)
```

Format

data.frame

References

Kumar S., Molloy C., Muñoz P., Daetwyler H., Chagné D., and Volz R. 2015. Genome-enabled estimates of additive and nonadditive genetic variances and prediction of apple phenotypes across environments. G3 Genes, Genomes, Genetics 5:2711–2718.

Examples

```
data(pheno.apple)
```

`pheno.pine` *Phenotypic data for loblolly pine dataset*

Description

Deregressed estimated breeding values (DEBV) for the trait diameter at breast height (DBH) at 6 years of age from trees grown at site Nassau. The dataset contains a total of 861 genotypes of loblolly pine (*Pinus taeda* L.). Dataset obtained from supplementary material in Resende *et al.* (2012).

Usage

```
data(pheno.pine)
```

Format

data.frame

References

Resende, M.F.R., Munoz, P. Resende, M.D.V., Garrick, D.J., Fernando, R.L., Davis, J.M., Jokela, E.J., Martin, T.A., Peter, G.F., and Kirst, M. 2012. Accuracy of genomic selection methods in a standard data set of loblolly pine (*Pinus taeda* L.). Genetics 190:1503-1510.

Examples

```
data(pheno.pine)
```

`pheno.salmon` *Phenotypic data for Atlantic salmon dataset*

Description

Phenotypic data on 1,481 Atlantic salmon individuals. All fish were phenotyped for mean gill score (mean of the left gill and right gill scores) and amoebic load (qPCR values using *Neoparamoeba perurans* specific primers, amplified from one of the gills). Dataset obtained from supplementary material in Robledo *et al.* (2018).

Usage

```
data(pheno.salmon)
```

Format

data.frame

References

Robledo D., Matika O., Hamilton A., and Houston R.D. 2018. Genome-wide association and genomic selection for resistance to amoebic gill disease in Atlantic salmon. G3 Genes, Genomes, Genetics 8:1195-1203.

Examples

```
data(pheno.salmon)
```

`qc.filtering` *Quality control filtering of molecular matrix M for downstream analyses*

Description

Reads molecular data in the format 0, 1, 2 and performs some basic quality control filters and simple imputation. Matrix provided is of the full form ($n \times p$), with n individuals and p markers. Individual and marker names are assigned to `rownames` and `colnames`, respectively. Filtering of markers can be done with the some of the following options by specifying thresholds for: missing values on individuals, missing values on markers, minor allele frequency, inbreeding Fis value, and observed heterozygosity. String used for identifying missing values can be specified. If requested, missing values will be imputed based on the mean of each SNP.

Usage

```
qc.filtering(
  M = NULL,
  base = FALSE,
  na.string = NA,
  map = NULL,
  marker = NULL,
  chrom = NULL,
  pos = NULL,
  ref = NULL,
  marker.callrate = 1,
  ind.callrate = 1,
  maf = 0,
  heterozygosity = 1,
  Fis = 1,
  impute = FALSE,
  Mrecode = FALSE,
  plots = TRUE,
  digits = 2,
  message = TRUE
)
```

Arguments

M	A matrix with SNP data of full form ($n \times p$), with n individuals and p markers Individual and marker names are assigned to rownames and colnames, respectively. Data in matrix is coded as 0, 1, 2 (integer or numeric) (default = NULL).
base	If TRUE matrix **M** is considered as bi-allele SNP data format (character) and the SNPs are recoded to numerical values before performing the quality control filters (default = FALSE) (currently deprecated).
na.string	A character that will be interpreted as NA values (default = "NA").
map	(Optional) A data frame with the map information with p rows (default = NULL).
marker	A character indicating the name of the column in data frame map with the identification of markers. This is mandatory if map is provided (default = NULL).
chrom	A character indicating the name of the column in data frame map with the identification of chromosomes (default = NULL).
pos	A character indicating the name of the column in data frame map with the identification of marker positions (default = NULL).
ref	A character indicating the name of the column in the map containing the reference allele for recoding. If absent, then conversion will be based on the major allele (most frequent). The marker information of a given individuals with two of the specified major alleles in ref will be coded as 2 (default = NULL).
marker.callrate	A numerical value between 0 and 1 used to remove SNPs with a rate of missing values equal or larger than this value (default = 1, *i.e.* no removing).
ind.callrate	A numerical value between 0 and 1 used to remove individuals with a rate of missing values equal or larger than this value (default = 1, *i.e.* no removing).
maf	A numerical value between 0 and 1 used to remove SNPs with a Minor Allele Frequency (MAF) below this value (default = 0, *i.e.* no removing).
heterozygosity	A numeric value indicating the maximum value of accepted observed heterozygosity (Ho) (default = 1, *i.e.* no removing).
Fis	A numeric value indicating the maximum value of accepted inbreeding (Fis) following the equation $\|1 - (Ho/He)\|$ (default = 1, *i.e.* no removing).
impute	If TRUE imputation of missing values is done using the mean of each SNP (default = FALSE).
Mrecode	If TRUE it provides the recoded **M** matrix from the bi-allelic to numeric SNP (default = FALSE) (currently deprecated).
plots	If TRUE generates graphical output of the quality control based on the original input matrix (default = TRUE).
digits	Set up the number of digits used to round the output matrix (default = 2).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Details

Warning: The arguments base, ref, and Mrecode currently are deprecated and will be removed on the next version of ASRgenomics. Use function snp.recode to recode the matrix prior to using qc.filtering.

The filtering process is carried out as expressed in the following simplified pseudo-code that consists on a loop repeated twice:

for i in 1 to 2

Filter markers based on call rate.

Filter individuals based on call rate.

Filter markers based on minor allele frequency.

Filter markers based on observed heterozygosity.

Filter markers based on inbreeding.

end for

Value

A list with the following elements:

- `M.clean`: the cleaned **M** matrix after the quality control filters have been applied.
- `map`: if provided, a cleaned `map` data frame after the quality control filters have been applied.
- `plot.missing.ind`: a plot of missing data per individual (original marker matrix).
- `plot.missing.SNP`: a plot of missing data per SNP (original marker matrix).
- `plot.heteroz`: a plot of observed heterozygocity per SNP (original marker matrix).
- `plot.Fis`: a plot of Fis per SNP (original marker matrix).
- `plot.maf`: a plot of the minor allele frequency (original marker matrix).

Examples

```
# Example 1: Pine dataset (coded as 0,1,2 with missing as -9).
data(geno.pine926)
M_filter <- qc.filtering(
   M = geno.pine926, map = NULL, marker.callrate = 0.9, ind.callrate = 0.9,
   maf = 0.05, heterozygosity = 0.9, Fis = 0.6, impute = FALSE, na.string = "-9")
ls(M_filter)
M_filter$M.clean[1:5, 1:5]
dim(M_filter$M.clean)
head(M_filter$map)
M_filter$plot.maf
M_filter$plot.missing.ind
M_filter$plot.missing.SNP
M_filter$plot.heteroz
M_filter$plot.Fis

# Example 2: Salmon dataset (coded as 0,1,2 with missing as NA).
data(geno.salmon)
M_filter <- qc.filtering(
   M = geno.salmon, map = NULL, marker.callrate = 0.10, ind.callrate = 0.20,
   maf = 0.02, heterozygosity = 0.9, Fis = 0.4, impute = FALSE, na.string = NA)
M_filter$M.clean[1:5, 1:5]
dim(M_filter$M.clean)
head(M_filter$map)
M_filter$plot.maf
M_filter$plot.missing.ind
M_filter$plot.missing.SNP
M_filter$plot.heteroz
M_filter$plot.Fis
```

snp.pca *Performs a Principal Component Analysis (PCA) based on a molecular matrix M*

Description

Generates a PCA and summary statistics from a given molecular matrix for population structure. Matrix provided is of full form ($n \times p$), with n individuals and p markers. Individual and marker names are assigned to `rownames` and `colnames`, respectively. SNP data is coded as 0, 1, 2 (integers or decimal numbers). Missing values are not accepted and these need to be imputed (see function `qc.filtering()` for implementing mean imputation). There is additional output such as plots and other data frames to be used on other downstream analyses (such as GWAS).

Usage

```
snp.pca(M = NULL, label = FALSE, ncp = 10, groups = NULL, ellipses = FALSE)
```

Arguments

`M`	A matrix with SNP data of full form ($n \times p$), with n individuals and p markers (default = `NULL`).
`label`	If `TRUE` then includes in output individuals names (default = `FALSE`).
`ncp`	The number of PC dimensions to be shown in the screeplot, and to provide in the output data frame (default = `10`).
`groups`	Specifies a vector of class factor that will be used to define different colors for individuals in the PCA plot. It must be presented in the same order as the individuals in the molecular **M** matrix (default = `NULL`).
`ellipses`	If `TRUE`, ellipses will will be drawn around each of the define levels in `groups` (default = `FALSE`).

Details

It calls function `prcomp()` to generate the PCA and the `factoextra` R package to extract and visualize results. Methodology uses normalized allele frequencies as proposed by Patterson *et al.* (2006).

Value

A list with the following four elements:

- `eigenvalues`: a data frame with the eigenvalues and its variances associated with each dimension including only the first `ncp` dimensions.
- `pca.scores`: a data frame with scores (rotated observations on the new components) including only the first `ncp` dimensions.
- `plot.pca`: a scatterplot with the first two-dimensions (PC1 and PC2) and their scores.
- `plot.scree`: a barchart with the percentage of variances explained by the `ncp` dimensions.

References

Patterson N., Price A.L., and Reich, D. 2006. Population structure and eigenanalysis. PLoS Genet 2(12):e190. doi:10.1371/journal.pgen.0020190

Examples

```
# Apple dataset
data(geno.apple)
data(pheno.apple)

# Perform the PCA.
SNP_pca <- snp.pca(M = geno.apple, ncp = 10)
ls(SNP_pca)
SNP_pca$eigenvalues
head(SNP_pca$pca.scores)
SNP_pca$plot.pca
SNP_pca$plot.scree

# PCA plot by family (17 groups).
grp <- as.factor(pheno.apple$Family)
SNP_pca_grp <- snp.pca(M = geno.apple, groups = grp, label = FALSE, ellipses = TRUE)
SNP_pca_grp$plot.pca
```

`snp.pruning` *Reduces the number of redundant markers on a molecular matrix M by pruning*

Description

For a given molecular dataset **M** (in the format 0, 1 and 2) it produces a reduced molecular matrix by eliminating "redundant" markers using pruning techniques. This function finds and drops some of the SNPs in high linkage disequilibrium (LD).

Usage

```
snp.pruning(
  M = NULL,
  map = NULL,
  marker = NULL,
  chrom = NULL,
  pos = NULL,
  method = c("correlation"),
  criteria = c("callrate", "maf"),
  pruning.thr = 0.95,
  by.chrom = FALSE,
  window.n = 50,
  overlap.n = 5,
  iterations = 10,
  seed = NULL,
  message = TRUE
)
```

Arguments

`M`	A matrix with marker data of full form ($n \times p$), with n individuals and p markers. Individual and marker names are assigned to `rownames` and `colnames`, respectively. Data in matrix is coded as 0, 1, 2 (integer or numeric) (default = `NULL`).

map	(Optional) A data frame with the map information with p rows. If NULL a dummy map is generated considering a single chromosome and sequential positions for markers. A map is mandatory if by.chrom = TRUE, where also option chrom must also be non-null.
marker	A character indicating the name of the column in data frame map with the identification of markers. This is mandatory if map is provided (default = NULL).
chrom	A character indicating the name of the column in data frame map with the identification of chromosomes. This is mandatory if map is provided (default = NULL).
pos	A character indicating the name of the column in data frame map with the identification of marker positions (default = NULL).
method	A character indicating the method (or algorithm) to be used as reference for identifying redundant markers. The only method currently available is based on correlations (default = "correlation").
criteria	A character indicating the criteria to choose which marker to drop from a detected redundant pair. Options are: "callrate" (the marker with fewer missing values will be kept) and "maf" (the marker with higher minor allele frequency will be kept) (default = "callrate").
pruning.thr	A threshold value to identify redundant markers with Pearson's correlation larger than the value provided (default = 0.95).
by.chrom	If TRUE the pruning is performed independently by chromosome (default = FALSE).
window.n	A numeric value with number of markers to consider in each window to perform pruning (default = 50).
overlap.n	A numeric value with number of markers to overlap between consecutive windows (default = 5).
iterations	An integer indicating the number of sequential times the pruning procedure should be executed on remaining markers. If no markers are dropped in a given iteration/run, the algorithm will stop (default = 10).
seed	An integer to be used as seed for reproducibility. In case the criteria has the same values for a given pair of markers, one will be dropped at random (default = NULL).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Details

Pruning is recommended as redundancies can affect the quality of matrices used for downstream analyses. The algorithm used is based on the Pearson's correlation between markers as a *proxy* for LD. In the event of a pairwise correlation higher than the selected threshold markers will be eliminated as specified by: call rate, minor allele frequency. In case of tie, one marker will be dropped at random.

Filtering markers (qc.filtering) is of high relevance before pruning. Poor quality markers (*e.g.*, monomorphic markers) may prevent correlations from being calculated and may affect eliminations.

Value

- Mpruned: a matrix containing the pruned marker M matrix.
- map: an data frame containing the pruned map.

Examples

```
# Salmon data
data(geno.pine655)
dim(geno.pine655)

M_filter <- qc.filtering(M = geno.pine655, base = FALSE, ref = NULL,
                         marker.callrate = 0.20, ind.callrate = 0.20,
                         maf = 0.05, Fis = 1, heterozygosity = 0.98,
                         impute = FALSE, na.string = "-9", plots = TRUE)

# Prune correlations > 0.9.
Mpr <- snp.pruning(M = M_filter$M.clean, pruning.thr = 0.90,
                   by.chrom = FALSE, window.n = 40, overlap.n = 10)
head(Mpr$map)
Mpr$Mpruned[1:5, 1:5]

# Initial marker matrix M contains 655 individuals and 4853 markers.
# Final pruned marker matrix M contains 644 individuals and 3071 markers.

# Prune correlations > 0.9 (with criteria maf).
Mpr <- snp.pruning(M = M_filter$M.clean, criteria = "maf",
                   pruning.thr = 0.90, by.chrom = FALSE,
                   window.n = 80, overlap.n = 10, seed = 1208)

# Create dummy map to test with chromosomes.
map <- ASRgenomics:::dummy.map_(colnames(M_filter$M.clean))
map$chrom <- c(rep(x = 1, 1000), rep(x = 2, 1000), rep(x = 3, 1050))
head(map)

# Perform pruning by chromosome.
Mpr <- snp.pruning(M = M_filter$M.clean, pruning.thr = 0.90,
                   map = map, marker = "marker", chrom = "chrom", pos = "pos",
                   by.chrom = TRUE, window.n = 40, overlap.n = 10, seed = 1208)
head(Mpr$map)
Mpr$Mpruned[1:5, 1:5]
```

`snp.recode` *Recodes the molecular matrix M for downstream analyses*

Description

Reads molecular data in format of bi-allelic nucleotide bases (AA, AG, GG, CC, etc.) and recodes them as 0, 1, 2 and `NA` to be used in other downstream analyses.

Usage

```
snp.recode(
  M = NULL,
  map = NULL,
  marker = NULL,
  ref = NULL,
  alt = NULL,
```

```
  recoding = c("ATGCto012"),
  na.string = NA,
  rename.markers = TRUE,
  message = TRUE
)
```

Arguments

M	A character matrix with SNP data of full form ($n \times p$), with n individuals and p markers Individual and marker names are assigned to rownames and colnames, respectively. Data in matrix is coded as AA, AG, GG, CC, etc (default = NULL).
map	(Optional) A data frame with the map information with p rows. If NULL a dummy map is generated considering a single chromosome and sequential positions for markers and includes reference allele and alternative allele (default = NULL).
marker	A character indicating the name of the column in data frame map with the identification of markers. This is mandatory if map is provided (default = NULL).
ref	A character indicating the name of the column in the map containing the reference allele for recoding. If absent, then conversion will be based on the major allele (most frequent). The marker information of a given individual with two of the specified major alleles in ref will be coded as 2. This is mandatory if map is provided (default = NULL).
alt	A character indicating the name of the column in the map containing the alternative allele for recoding. If absent, then it will be inferred from the data. The marker information of a given individual with two of the specified alleles in alt will be coded as 0 (default = NULL).
recoding	A character indicating the recoding option to be performed. Currently, only the nucleotide bases (AA, AG, ...) to allele count is available ("ATGCto012") (default = "ATGCto012").
na.string	A character that is interpreted as missing values (default = "NA").
rename.markers	If TRUE marker names (as provided in **M**) will be expanded to store the reference and alternative alleles. For example, from AX-88234566 to AX-88234566_C_A. In the event of unidentified alleles, 0 will be used (default = TRUE).
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Value

A list with the following two elements:

- Mrecode: the molecular matrix $\boldsymbol{M}$ recoded to 0, 1, 2 and NA.
- mapr: the data frame with the map information including reference allele and alternative allele.

Examples

```
# Create bi-allelic nucleotide base data set.
Mnb <- matrix(c(
  "A-",  NA, "GG",   "CC",   "AT",   "CC",   "AA",   "AA",
  "AAA", NA, "GG",   "AC",   "AT",   "CG",   "AA",   "AT",
  "AA",  NA, "GG",   "CC",   "AA",   "CG",   "AA",   "AA",
  "AA",  NA, "GG",   "AA",   "AA",    NA,    "AA",   "AA",
  "AT",  NA, "GG",   "AA",   "TT",   "CC",   "AT",   "TT",
  "AA",  NA,   NA,   "CC",    NA,    "GG",   "AA",   "AA",
```

```
  "AA",  NA,   NA,   "CC",   "TT",   "CC",   "AA",   "AT",
  "TT",  NA, "GG",   "AA",   "AA",   "CC",   "AA",   "AA"),
  ncol = 8, byrow = TRUE, dimnames = list(paste0("ind", 1:8),
                                          paste0("m", 1:8)))
Mnb

# Recode without map (but map is created).
Mr <- snp.recode(M = Mnb, na.string = NA)
Mr$Mrecode
Mr$map

# Create map.
mapnb <- data.frame(
 marker = paste0("m", 1:8),
 reference = c("A", "T", "G", "C", "T", "C", "A", "T"),
 alternative = c("T", "G", "T", "A", "A", "G", "T", "A")
 )
 mapnb

# Recode with map without alternative allele.
Mr <- snp.recode(M = Mnb, map = mapnb, marker = "marker", ref = "reference",
           na.string = NA, rename.markers = TRUE)
Mr$Mrecode
Mr$map
# Notice that the alternative allele is in the map as a regular variable,
# but in the names it is inferred from data (which might be 0 (missing)).

# Recode with map with alternative allele.
Mr <- snp.recode(M = Mnb, map = mapnb, marker = "marker",
 ref = "reference", alt = "alternative",
 na.string = NA, rename.markers = TRUE)
Mr$Mrecode
Mr$map # Now the alternative is also on the names.

# We can also recode without renaming the markers.
Mr <- snp.recode(M = Mnb, map = mapnb, marker = "marker", ref = "reference",
           na.string = NA, rename.markers = FALSE)
Mr$Mrecode
Mr$map # Now the alternative is also on the names.
```

`sparse2full` *Generates a full matrix form from a sparse form matrix*

Description

Modifies the input sparse form matrix into its full form. The sparse form has three columns per line, corresponding to the set: `Row`, `Col`, `Value`, and is defined by a lower triangle row-wise of the full matrix and is sorted as columns within row. Individual names should be assigned as attributes: `attr(K, "rowNames")` and `attr(K, "colNames")`. If these are not provided they are considered as 1 to n.

Usage

```
sparse2full(K = NULL)
```

Arguments

K	A square matrix in sparse form (default = NULL).

Details

Based on a function from ASReml-R 3 library by Butler *et al.* (2009).

Value

A full square matrix where individual names are assigned to rownames and colnames. If attribute INVERSE is found this is also passed to the full matrix.

References

Butler, D.G., Cullis, B.R., Gilmour, A.R., and Gogel, B.J. 2009. ASReml-R reference manual. Version 3. The Department of Primary Industries and Fisheries (DPI&F).

Examples

```
data(geno.apple)
Gsp <- G.matrix(M = geno.apple, method = "VanRaden", na.string = "NA",
              sparseform = TRUE)$G.sparse
head(Gsp)
head(attr(Gsp, "rowNames"))
G <- sparse2full(K = Gsp)
G[1:5, 1:5]
```

---

synthetic.cross *Generates a molecular matrix M for hypothetical crosses based on the genomic information of the parents*

Description

This function generates (or imputes) a molecular matrix for offsprings from hypothetical crosses based on the genomic information from the parents. This is a common procedure in species such as maize, where only the parents (inbred lines) are genotyped, and this information is used to generate/impute the genotypic data of each of the hybrid offspring. This function can be also used for bulked DNA analyses, in order to obtain an bulked molecular matrix for full-sib individuals were only parents are genotyped.

Usage

```
synthetic.cross(
  M = NULL,
  ped = NULL,
  indiv = NULL,
  mother = NULL,
  father = NULL,
  heterozygote.action = c("useNA", "exact", "fail", "expected"),
  na.action = c("useNA", "expected"),
  message = TRUE
)
```

Arguments

M	A matrix with marker data of full form ($n \times p$), with n individuals (mothers and fathers) and p markers. Individual and marker names are assigned to rownames and colnames, respectively. Data in matrix is coded as 0, 1, 2 (integer or numeric) (default = NULL).
ped	A data frame with three columns containing only the pedigree of the hypothetical offspring. (not pedigree of parents) It should include the three columns for individual, mother and father (default = NULL).
indiv	A character indicating the column in ped data frame containing the identification of the offspring (default = NULL).
mother	A character indicating the column in ped data frame containing the identification of the mother (default = NULL).
father	A character indicating the column in ped data frame containing the identification of the father (default = NULL).
heterozygote.action	Indicates the action to take when heterozygotes are found in a marker. Options are: "useNA", "exact", "fail", and "expected". See details for more information (default = "useNA")
na.action	Indicates the action to take when missing values are found in a marker. Options are: "useNA" and "expected". See details for more information (default = "useNA").
message	If TRUE diagnostic messages are printed on screen (default = TRUE).

Details

For double-haploids, almost the totality of the markers (except for genotyping errors) will be homozygotic reads. But in many other cases (including recombinant inbred lines) there will be a proportion of heterozygotic reads. In these case, it is very difficult to infer (impute) the exact genotype of a given offspring individual. For example, if parents are 0 (AA) and 1 (AC) then offsprings will differ given this Mendelian sampling. However, different strategies exist to determine the expected value for that specific cross (if required), which are detailed below using the option heterozygote.action.

- If heterozygote.action = "useNA", the generated offspring will have, for the heterozygote read, an NA, and no markers are removed. Hence, no attempt will be done to impute/estimate this value.
- If heterozygote.action = "exact", any marker containing one or more heterozygote reads will be removed. Hence, inconsistent markers are fully removed from the **M** matrix.
- If heterozygote.action = "fail", function stops and informs of the presence of heterozygote reads.
- If heterozygote.action = "expected", then an algorithm is implemented, on the heterozygote read to determine its expected value. For example, if parents are 0 and 1, then the expected value (with equal probability) is 0.5. For a cross between two heterozygotes, the expected value is: $0(1/4) + 1(1/2) + 2(1/4) = 1$. And for a cross between 1 and 2, the expected value is: $1(1/2) + 2(1/2) = 1.5$

Missing value require special treatment, and an imputation strategy is detailed below as indicated using the option na.action.

- If `na.action = "useNA"`, if at least one of the parental reads is missing values for a given marker then it will be assigned as missing for the hypothetical cross. Hence, no attempt will be done to impute/estimate this value.

  If `na.action = "expected"`, then an algorithm is implemented that will impute the expected read of the cross if the genotype of **one of the parents is missing** (*e.g.*, cross between 0 and NA). Calculations are based on parental allelic frequencies p and q for the given marker. The expressions for expected values are detailed below.

  - If the genotype of the non-missing parent read is 0.
    q^2 (probability that the missing parent is 0) x 0 (expected value of the offspring from a 0 x 0 cross: $0(1/1)$) +
    $2pq$ (probability that the missing parent is 1) x 0.5 (expected offspring from a 0 x 1 cross: $0(1/2) + 1(1/2)$) +
    q^2 (probability that the missing parent is 2) x 1 (expected offspring from a 0 x 2 cross: $1(1/1)$)
  - If the genotype of the non-missing parent read is 1.
    q^2 (probability that the missing parent is 0) x 0.5 (offspring: $0(1/2) + 1(1/2)$) +
    $2pq$ (probability that the missing parent is 1) x 1 (offspring: $0(1/4) + 1(1/2) + 2(1/4)$) +
    q^2 (probability that the missing parent is 2) x 1.5 (offspring: $1(1/2) + 2(1/2)$)
  - If the genotype of the non-missing parent read is 2.
    q^2 (probability that the missing parent is 0) x 1 (offspring: $1(1/1)$) +
    $2pq$ (probability that the missing parent is 1) x 1.5 (offspring: $1(1/2) + 2(1/2)$) +
    q^2 (probability that the missing parent is 2) x 2 (offspring: $2(1/1)$)

Similarly, the calculation of the expected read of a cross when **both parents are missing** is also based on population allelic frequencies for the given marker. The expressions for expected values are detailed below.

$q^2 \times q^2$ (probability that both parents are 0) x 0 (expected value of the offspring from a 0 x 0 cross: 0(1/1)) +

$2 \times (q^2 \times 2pq)$ (probability that the first parent is 0 and the second is 1; this requires the multiplication by 2 because it is also possible that the first parent is 1 and the second is 0) x 0.5 (offspring: $0(1/2) + 1(1/2)$) +

$2 \times (q^2 \times p^2)$ (this could be 0 x 2 or 2 x 0) x 1 (offspring: $1(1/1)$) +

$2pq \times 2pq$ (both parents are 1) x 1 (offspring: $0(1/4) + 1(1/2) + 2(1/4)$) +

$2 \times (2pq \times q2)$ (this could be 1 x 2 or 2 x 1) x 1.5 (offspring: $1(1/2) + 2(1/2)$) +

$p^2 \times p^2$ (both parents are 2) x 2 (offspring: $2(1/1)$)

Note that the use of `na.action = "expected"` is recommended when a large number of offspring will conform the hybrid cross (such as with bulked DNA analyses) for family groups with reasonable number of individuals.

Warning. If `"expected"` is used for `heterozygote.action` or `na.action`, direct transformation of the molecular data to other codings (*e.g.*, dominance matrix coded as `c(0,1,0)`) is not recommended.

Value

A molecular matrix **M** containing the genotypes generated/imputed for the hypothetical cross.

Examples

```
# Apple dataset
data(geno.apple)

# Create dummy pedigree (using first 10 as parents).
ped <- data.frame(male = rownames(geno.apple)[1:5],
                  female = rownames(geno.apple)[6:10])
ped$offs <- paste(ped$male, ped$female, sep = "_")
ped

# Select portion of M for parents.
Mp <- geno.apple[c(ped$male, ped$female), 1:15]

# Get genotype of crosses removing markers with heterozygotes.
synthetic.cross(M = Mp, ped = ped, indiv = "offs", mother = "female", father = "male",
                heterozygote.action = "exact", na.action = "useNA")

# Requesting the synthetic cross to be NA in the respective samples.
synthetic.cross(M = Mp, ped = ped, indiv = "offs", mother = "female", father = "male",
                heterozygote.action = "useNA", na.action = "useNA")

# Get genotype of crosses and use expected values.
synthetic.cross(M = Mp, ped = ped, indiv = "offs", mother = "female", father = "male",
                heterozygote.action = "expected", na.action = "expected")
```

Index