Package ‘ASRgenomics’

March 22, 2021

Title ASReml-R Genomics Tools

Version 1.0.0

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='amanda.aoliveira@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 to assist in analytical pipelines before and after use of ASReml-R or another library to perform tasks such as Genomic Selection (GS) or Genome-Wide Associaton Analyses (GWAS).

License GPL-3

Encoding UTF-8

LazyData true

SystemRequirements GNU make

RoxygenNote 7.1.1

Suggests knitr,
rmarkdown,

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports Matrix,
stats,
ggplot2,
AGHmatrix,
factoextra,
ggpubr,
reshape2,
methods,
superheat,
scattermore

URL www.vsni.co.uk/software/asreml

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 to assist in analytical pipelines before and after use of ASReml-R or another library to perform tasks such as Genomic Selection (GS) or Genome-Wide Associaton Analyses (GWAS).

Details

The sections considered are:

- Preparing/exploring pedigree/phenotypic data
- Preparing/exploring genomic data
- Complementing genomic breeding values

The implemented functions consider aspects such as: filtering SNP data for quality control; obtaining a genomic matrix; assessing a kinship matrix by reporting diagnostics (statistics and plots); generating PCA based on kinship or SNP matrices; calculating the inverse of a genomic matrix and assessing its quality; matching pedigree- against genomic-based matrices; tuning-up a genomic matrix (bending, blending or aligning); obtaining the hybrid matrix as required with ssGBLUP, to name a few.

To learn more about ASRgenomics, start with the vignettes: ‘browseVignettes(package = "ASRgenomics")‘

`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. Individual names should be assigned to `rownames` and `colnames`.

Usage

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

Arguments

`K`	A kinship matrix in full form (n x n).
`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` and the attributes `rowNames` and `colNames`.

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

```
## Not run:
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(attr(G.sparse, 'rowNames'))
head(attr(G.sparse, 'colNames'))
head(G.sparse)

## End(Not run)
```

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

Description

Generates the inverse of a genomic relationship matrix **G** that is provided. This matrix should be of the full form (n x 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.

Usage

```
G.inverse(
  G = NULL,
  A = NULL,
  rcn.thr = 1e-12,
  blend = FALSE,
  pblend = 0.02,
  bend = FALSE,
  eig.tol = 1e-06,
  align = FALSE,
  digits = 8,
  sparseform = FALSE,
  message = TRUE
)
```

Arguments

`G`	Input of the genomic relationship matrix **G**, in full form (ng x ng), to obtain its inverse.
`A`	Input of the pedigree relationship matrix **A** to perform blending or aligning, in full form (na x na). It should be of the same dimension as the **G** matrix.
`rcn.thr`	A threshold for identifying the **G** matrix as an ill-conditioned matrix. Based on the reciprocal conditional number (default = 1e-12).
`blend`	If `TRUE` a 'blending' with identity matrix **I** or pedigree relationship matrix **A** (if provided) is performed (default = `FALSE`).
`pblend`	If blending is requested this is the proportion of the identity matrix **I** or pedigree relationship matrix **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 relationship matrix **G** is aligned to the pedigree relationship matrix **A** (default = `FALSE`).
`digits`	Set up the number of digits used to round the matrix (default = 8).
`sparseform`	If `TRUE` it generates an inverse matrix in sparse form to be used directly in ASReml-R with required attributes (default = `FALSE`).
`message`	If `TRUE` print report messages on screen (default = `TRUE`).

Details

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

Value

A list with three of the following elements:

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

```
## Not run:
# Example 1: 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 2: Performing some blending, A is not provided
GINV <- G.inverse(G=G, blend=TRUE)
GINV$Ginv[1:5, 1:5]

# Example 3: Performing some bending
GINV <- G.inverse(G=G, bend=TRUE)
GINV$Ginv[1:5, 1:5]

# Example 4: Performing some bending result as sparse form matrix
GINV <- G.inverse(G=G, bend=TRUE, sparseform=TRUE)
head(GINV$Ginv.sparse)

## End(Not run)
```

`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 is of the form n x 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
)
```

Arguments

`M`	A matrix with SNP data of form n x 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).
`method`	The method considered for calculation of genomic matrix. `'VanRaden'` and `'Yang'` for additive, 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-R with required attributes (default = `FALSE`).
`digits`	Set up the number of digits used to round the matrix (default = 8).

Details

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

It uses function `Gmatrix()` from package `AGHmatrix` v. 0.0.5 (Amadeu *et al.* 2019)

Value

A list with ONLY one of these two elements:

- `G`: the **G** matrix in full form.
- `G.sparse`: if requested, the **G** matrix in sparse form.

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

```
## Not run:
# Example 1: Requesting a full matrix by VanRanden
data(geno.apple)
G <- G.matrix(M=geno.apple, method='VanRaden', na.string='NA', digits=8)$G
G[1:5,1:5]

# Example 2: Requesting a sparse form by VanRanden for ASReml-R
G <- G.matrix(M=geno.apple, method='VanRaden', na.string='NA', sparseform=TRUE)
head(G$G.sparse)
head(attr(G$G.sparse, 'rowNames'))

## End(Not run)
```

`G.predict` *Generates the conditional predictions of random variables (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 **G** for all individuals (full matrix of dimension (nx + ny) x (nx + ny)).

Usage

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

Arguments

`G`	Input of the genomic relationship matrix **G** in full form ((nx + ny) x (nx + ny)).
`gy`	Input of random effects (*e.g.* breeding values) for individuals with known values. Individual names should be assigned to `rownames` of the matrix **G**.
`vcov.gy`	The variance-covariance matrix associated with the random effects from the individuals with known values (set `y`, of dimension ny x ny).

Details

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

The user needs to provide the matrix **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 are presented.

Value

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

Examples

```
## Not run:
# 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) # 247 x 247

# 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,]
table(rownames(Gy) == phenoy$INDIV)

# Obtaining the BLUPs for the 147 individuals using ASReml-R
## Getting the G inverse
Gyinv  <- G.inverse(G=Gy, bend=TRUE, blend=FALSE, align=FALSE, sparseform=TRUE)$Ginv.sparse
# Fitting a GBLUP model
phenoy$INDIV<-as.factor(phenoy$INDIV)
modelGBLUP<-asreml(fixed=JUI_MOT~1,
                  random=~vm(INDIV,Gyinv),
                  workspace=128e06,
                  data=phenoy)
## Obtaining Predictions - BLUP -
BLUP<-summary(modelGBLUP,coef=TRUE)$coef.random
head(BLUP)
gy <- as.matrix(BLUP[,1])
rownames(gy) <- phenoy$INDIV

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

# 4) Adding a dummy var-cov
vcov <- diag(as.numeric(var(gy)),ncol=147,nrow=147)
dim(vcov)
rownames(vcov) <- rownames(gy)
colnames(vcov) <- rownames(gy)
g.cond2 <- G.predict(G=G, gy=gy, vcov.gy=vcov)
head(g.cond2)

## End(Not run)
```

`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 inversion. The input matrix should be of the full form (n x n) with individual names assigned to `rownames` and `colnames`.

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 **G** to tune-up in full form (ng x ng).
`A`	Input of the pedigree relationship matrix **A** in full form (na x na).
`blend`	If `TRUE` a *blending* with identity matrix **I** or pedigree relationship matrix **A** (if provided) is perfomed (default = `FALSE`).
`pblend`	If blending is requested this is the proportion of the identity matrix **I** or pedigree relationship matrix **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 **G** is *aligned* to the pedigree relationship matrix **A** (default = `FALSE`).
`rcn`	If `TRUE` the reciprocal conditional number of the original and the bended/blended/aligned matrix will be calculated (default = `TRUE`).
`digits`	Set up the number of digits used to round the matrix (default = 8).
`sparseform`	If `TRUE` it generates an inverse matrix in sparse form to be used directly in ASReml-R with required attributes (default = `FALSE`).
`determinant`	If `TRUE` the determinant will be always calculated, otherwise, this is obtained for matrices of a dimension of less than 1500 (default = `TRUE`).
`message`	If `TRUE` print report messages on screen (default = `TRUE`).

Details

This routine provides three options of tune-up:

- *Blend*. The **G** matrix is blended (or averaged) with another matrix.
  $\boldsymbol{G}^* = p\boldsymbol{G} + (1-p)\boldsymbol{A}$, where **G** is the original matrix, $\boldsymbol{G}^*$ is the blended matrix, p is the proportion of the original matrix, and **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 **I**.

- *Bend.* It consists on adjusting the original **G** matrix to obtain a near positive definite matrix, which is done by making negative or very small eigenvalues slightly positive.

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

The user should provide the matrices **G** and **A** in full form (n x n). Individual names should be assigned to the `rownames` and `colnames` of the matrices.

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

Value

A list with six of the following elements:

- `Gb`: the inverse of **G** matrix in full form (only if sparseform = `FALSE`).

- `Gb.sparse`: if requested, the inverse of **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/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

```
## Not run:
# Example 1: Apple dataset
# 1.1 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, sparseform=FALSE)
ls(G_blended)
G_blended$Gb[1:5,1:5]

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

# Example 2 - Loblolly Pine dataset with pedigree - Aligned G matrix
data(geno.pine926)
data(ped.pine)
head(ped.pine)
A <- AGHmatrix::Amatrix(ped.pine)
dim(A)
G <- G.matrix(M=geno.pine926, 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, sparseform=FALSE)
G_align$Gb[1:5,1:5]

## End(Not run)
```

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., 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., 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 approach combines the information from the pedigree relationship matrix **A** and the genomic relationship matrix **G** in one hybrid relationship matrix called **H**. The user should provide the matrices **A** or its inverse (only one version is required) and the inverse of the matrix **G** in the full form. Individual names should be assigned to `rownames` and `colnames`, and individuals from **Ginv** are verified to be all a subset within individuals from **A** (or **Ainv**).

Usage

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

Arguments

`A`	Input of the pedigree relationship matrix **A** in full form (na x na).
`Ainv`	Input of the inverse of the pedigree relationship matrix $\boldsymbol{A}^{-1}$ in full form (na x na).
`Ginv`	Input of the inverse of the genomic relationship matrix $\boldsymbol{G}^{-1}$ in full form (ng x ng).
`lambda`	The scaling factor for $(\boldsymbol{G^{-1}} - \boldsymbol{A_{22}^{-1}})$
`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-R with required attributes (default = `FALSE`).
`keep.order`	If `TRUE` the original order of the individuals from the `A` or `Ainv` matrix is kept. Otherwise the non-genotyped individuals are placed first and then genotyped individuals.
`digits`	Set up the number of digits used to round the matrix (default = 8).
`message`	If `TRUE` print report messages on screen (default = `TRUE`).
`inverse`	If `TRUE` it generates the inverse of **H** matrix (default = `TRUE`).

Value

The inverse of the hybrid matrix **H** matrix, in full or sparse form with required attributes to be used in ASReml-R. If (inverse = `FALSE`) the **H** matrix is provided instead of its inverse.

References

Cappa, E.P., Y.A. El-Kassaby, J. Klapste, F. Munoz, M.N. Garcia, P. V Villalba, and S.N.M. Poltri (2017). Improving accuracy of breeding values by incorporating genomic information in spatial-competition mixed models. Mol. Breed. 37. doi:10.1007/s11032-017-0725-6.

Christensen, O.F., Lund, M.S. (2009). Genomic relationship matrix when some animals are not genotyped. In: Proc. 60th Annual Meeting EAAP, Barcelona, Spain. Wageningen Press, Wageningen, the Netherlands. P. 299

Legarra, A., Aguilar, I., 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., 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 1: Using pine data
data(ped.pine)
data(geno.pine655)

# Getting A matrix
A <- AGHmatrix::Amatrix(data=ped.pine)
A[1:5,1:5]
dim(A) # 2034 x 2034

# Getting Ainv matrix
Ainv <- G.inverse(G=A)$Ginv
Ainv[1:5,1:5]
dim(Ainv) # 2034 x 2034

# 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)

dim(geno.pine655) # 655 x 4853
dim(M_filter$M.clean) # 644 x 3068

# Getting G matrix
G <- G.matrix(M=M_filter$M.clean, method='VanRaden', na.string='-9')$G
G[1:5,1:5]
dim(G) # 644 x 644
check_G <- kinship.diagnostics(K=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)

# Example 1a: Obtaining Hinv
Hinv <- H.inverse(A=A, G=Ginv, lambda=0.90, sparseform=TRUE)
head(Hinv)

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

## End(Not run)
```

kinship.diagnostics *Reports summary statistics, plots and filter options for diagonal and off-diagonal elements of a given kinship matrix*

Description

It reports summary statistics, plots and allows for some filter options for diagonal and off-diagonal elements of a given kinship matrix. This matrix can be a pedigree-based relationship matrix **A**, a genomic relationship matrix **G** or a hybrid relationship matrix **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
)
```

Arguments

`K`	Input of a kinship matrix in full format (n x n).
`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 in the calculation $\boldsymbol{K}_{i,i}/\sqrt{\boldsymbol{K}_{i,i} * \boldsymbol{K}_{j,j}}$ is identified (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 with the non-flagged duplicate individuals (default = `FALSE`).

Value

A list with the following elements:

- `list.diagonal`: a dataframe with the list of flagged large diagonal values.
- `list.duplicate`: a dataframe with the list of possible duplicates.
- `clean.kinship`: output of kinship matrix filtered according to non-flagged diagonal and/or duplicate individuals.

- `dirty.kinship`: output of kinship matrix with flagged 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

```
## Not run:
# 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
head(G_summary$list.diagonal,20)
G_summary$plot.diag
G_summary$plot.offdiag

## End(Not run)
```

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

Description

Generates a heatmap with dendrogram based on a provided kinship matrix. This matrix can be a pedigree relationship matrix **A**, a genomic relationship matrix **G** or a hybrid relationship matrix **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 x n).
`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

```
## Not run:
data(geno.apple)
G <- G.matrix(M=geno.apple, method='VanRaden', na.string='NA')$G
G[1:5,1:5]
kinship.heatmap(K=G[1:30,1:30], dendrogram=TRUE, row.label=TRUE, col.label=TRUE)
kinship.heatmap(K=G, dendrogram=TRUE, row.label=FALSE, col.label=FALSE)

## End(Not run)
```

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

Description

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

Usage

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

Arguments

`K`	Input of a kinship matrix in full format (n x n).
`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 dataframe (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`).

Details

Uses the `factoextra` R package to extract and visualize results.

Value

A list with the following four elements:

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

Examples

```
## Not run:
data(geno.apple)
data(pheno.apple)
G <- G.matrix(M=geno.apple, method='VanRaden', na.string='NA', sparseform=FALSE)$G
G[1:5,1:5]
G_pca <- kinship.pca(K=G, ncp=10)
ls(G_pca)
G_pca$eigenvalues  # eigenvalue and their var.explained
head(G_pca$pca.scores)
G_pca$plot.pca
G_pca$plot.scree

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

## End(Not run)
```

---

match.G2A — *Check the genomic relationship matrix* **G** *against the pedigree relationship matrix* **A** *or viceversa*

Description

Assesses a given genomic relationship matrix **G** against the pedigree relationship matrix **A** to determime the matched and mismatched individuals. If requested, it provides the cleaned version containing only the matched individuals of both matrices. The user should provide the matrices **G** and **A** in full form (ng x ng and na x na, respectively). Individual names should be assigned to `rownames` and `colnames` of both matrices.

Usage

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

Arguments

`A`	Input of the pedigree relationship matrix **A** in full form (na x na).
`G`	Input of the genomic relationship matrix **G** in full form (ng x ng).
`clean`	If `TRUE` generates new clean **G** and **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 new clean **A** and **G** matrices (default = `TRUE`).
`mism`	If `TRUE` generates two dataframes with mismatched individual names from the **G** and **A**matrices (default = `FALSE`).
`RMdiff`	If `TRUE` it generates the matrix (in sparse form) of matched observations from both the **G** and **A** matrices. This matrix can be used for identifying errors between matrices, but it can be very large (default = `FALSE`).

Value

A list with the following elements:

- `Gclean`: the portion of matrix **G** containing only matched individuals.

- `Aclean`: the portion of matrix **A** containing only matched individuals.
- `mismG`: a vector containing individuals from matrix **G** that are missing in matrix **A**.
- `mismA`: a vector containing individuals from matrix **A** that are missing in matrix **G**.
- `RM`: a data frame with the observations from both the **G** and **A** matched matrices, together with their absolute difference.
- `plotG2A`: scatterplot with the pairing of pedigree- against genomic-based values. This plot might take a long time with larger datasets.

Examples

```
## Not run:
# Example 1: Pine data
data(geno.pine655)
data(ped.pine)
head(ped.pine)
A <- AGHmatrix::Amatrix(ped.pine)
A[1:5,1:5]
dim(A)
G <- G.matrix(M=geno.pine655, method='VanRaden', na.string=-9, sparseform=FALSE)$G
G[1:5,1:5]
dim(G)
check <- match.G2A(A=A, G=G, clean=TRUE, ord=TRUE, mism=TRUE)
ls(check)
dim(check$Aclean) # Same dimension and order as G
dim(check$Gclean) # Same dimension and order
check$Aclean[1:5,1:5]
check$Gclean[1:5,1:5]
head(check$mismG)
head(check$mismA)

# Example 2: Identifying issues on pedigree (by checking Kinship matrix)
G_align <- G.tuneup(G=check$Gclean, A=check$Aclean, align=TRUE, sparseform=FALSE)$Gb
G2A.check <- match.G2A(A=A, G=G_align, clean=TRUE, ord=TRUE, mism=FALSE, RMdiff=TRUE)
ls(G2A.check)
G2A.check$plotG2A
head(G2A.check$RM)

## End(Not run)
```

`match.kinship2pheno` *Check any kinship matrix 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 is generated that has only the matched individuals. This matrix can be a pedigree-based relationship matrix **A**, a genomic-based relationship matrix **G**, or a hybrid relationship matrix **H**. Individual names should be assigned to `rownames` and `colnames`.

Usage

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

Arguments

`K`	Input of a kinship matrix in full form (n x n).
`pheno.data`	A datafame with the phenotypic data to assess (for n individuals).
`indiv`	The string for the column name for genotypes/individuals in the phenotypic data.
`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 recomended for downstream genomic analyses (default = `TRUE`).
`mism`	If `TRUE` generates dataframes with matched and mismatched individual names from the kinship matrix and the phenotypic data (default = `FALSE`).

Value

A list with the following elements:

- `mismatchesG`: a vector containing the names of the individuals from the provided kinship matrix that *mismatch* with the phenotypic data.
- `matchesG`: 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 kinship matrix containing only the matched phenotyped individuals.

Examples

```
## Not run:
data(geno.pine655)
data(pheno.pine)
head(pheno.pine)
GHAT <- G.matrix(M=geno.pine655, method='VanRaden', na.string='-9', sparseform=FALSE)
check <- match.kinship2pheno(K=GHAT$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)

## End(Not run)
```

`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., 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., 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 diameter at breast height (DBH) for site Nassau at age 6, for 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 Atalntic 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., 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` *Reading/filtering SNP molecular data*

Description

Reads molecular data in the format (0, 1, 2) or as a bi-allele SNP data format (AA, AG, GG, CC, etc.) and performs some basic quality control filters and simple imputation (if requested). Matrix provided is of the full form (n x p), with n individuals and p markers. Individual and marker names are assigned to `rownames` and `colnames`, respectively. 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,
  ref = NULL,
  marker.callrate = 1,
  ind.callrate = 1,
  maf = 0,
  na.string = NA,
  impute = FALSE,
  message = TRUE,
  Mrecode = FALSE,
  plots = TRUE,
  digits = 2
)
```

Arguments

`M`	A matrix with SNP data of full form (n x 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) or bi-allele SNP data format of AA, AG, GG, CC, etc. (character).
`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`).
`ref`	A character vector of same size of the number of markers (columns) in **M**, indicating the reference allele to be used when recoding markers from character to numeric; if absent, then conversion will be based on the major allele (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).
`na.string`	A character that is interpreted as NA values (default = NA).
`impute`	If `TRUE` imputation of missing values is done using the mean of each SNP (default = *TRUE*).
`message`	If `TRUE` print report messages on screen (default = `TRUE`).
`Mrecode`	If `TRUE` it provides the recoded `M` matrix from the bi-allelic to numeric SNP (default = `FALSE`).
`plots`	If `TRUE` generates graphical output of the quality control based on the original input **M** matrix (default = `TRUE`).
`digits`	set up the number of digits used to round the matrix (default = 2).

Details

The procedure first filters according to marker information (based on `marker.callrate` and `maf`) and after this individuals are filtered (based on `ind.callrate`).

The recodification of bi-allele SNP data format to a numeric matrix is based on the function `geno_to_allelecnt()` found at:

https://github.com/ekfchan/evachan.org-Rscripts/blob/master/rscripts/geno_to_allelecnt.R

Value

A list with the following elements:

- `M.recode`: the recoded **M** matrix from the bi-allelic to numeric SNP.
- `M.clean`: the cleaned **M** matrix after the quality control filters have been applied.
- `plot.missing.ind`: plot of missing data per individual.
- `plot.missing.SNP`: plot of missing data per SNP.
- `plot.heteroz`: plot of heterozygocity per SNP.
- `plot.maf`: plot of the minor allele frequency (MAF).

References

The recodification of bi-allele SNP data format to a numeric matrix is based on the function geno_to_allelecnt (license: GNU2) from Eva Chan found at: https://github.com/ekfchan/evachan.org-Rscripts/blob/master/rscripts/geno_to_allelecnt.R

Examples

```
## Not run:
# Example 1: Pine dataset (coded as 0,1,2 with missing)
data(geno.pine926)
M_filter <- qc.filtering(M=geno.pine926, base=FALSE, ref=NULL, marker.callrate=0.9,
                ind.callrate=0.9, maf=0.05, message=TRUE, impute=TRUE, na.string=-9,
                 plots=TRUE, digits=2)
ls(M_filter)
dim(geno.pine926) # 926 x 4853
M_filter$M.clean[1:5,1:5]
dim(M_filter$M.clean)
M_filter$plot.heteroz
M_filter$plot.maf
M_filter$plot.missing.ind
M_filter$plot.missing.SNP

# Example 2: Create nitrogenous base toy example
Mnb <- matrix(c("AA",   "AC",   "GG",   "CC",   "AT",   "CC",   "AA",   "AA",
               "AA",   "AC",   "GG",   "AC",   "AT",   "CG",   "AA",   "AT",
               "AA",   "AA",   "GG",   "CC",   "AA",   "CG",   "AA",   "AA",
               "AA",   "AA",   "GG",   "AA",   "AA",    NA,    "AA",   "AA",
               "AT",   "CC",   "GG",   "AA",   "TT",   "CC",   "AT",   "TT",
               "AA",   "AA",    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_fixed <- qc.filtering(M=Mnb, base=TRUE, ref=NULL, marker.callrate=0.8,
                 ind.callrate=0.8, maf=0.01, message=TRUE, impute=TRUE, na.string=NA,
                 Mrecode=TRUE, plots=FALSE, digits=0)
ls(Mnb_fixed)
Mnb
Mnb_fixed$M.clean
Mnb_fixed$M.recode

## End(Not run)
```

`snp.pca` *Performs a Principal Component Analysis (PCA) based on a provided molecular matrix for population structure*

Description

Generates a PCA and summary statistics from a given molecular matrix. Matrix provided is of full form (n x 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.

Usage

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

Arguments

`M`	A matrix with SNP data of full form (n x 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).

`label`	If `TRUE` then includes individuals names in output (default = `FALSE`).
`ncp`	The number of PC dimensions to be shown in the screeplot, and to provide in the output dataframe (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`).

Details

Uses 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:

- `eig_var`: a dataframe with the eigenvalues and its variances associated with each dimension including only the first `ncp` dimensions.
- `pca.scores`: a dataframe with scores (rotated observations on the new components) including only the first `ncp` dimensions.
- `plot.pca`: scatterplot with the first two-dimensions (PC1 and PC2) and their scores.
- `plot.scree`: 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

```
## Not run:
data(geno.apple)
M_filter <- qc.filtering(M=geno.apple, base=FALSE, ref=NULL, marker.callrate=0.9,
                  ind.callrate=0.9, maf=0.05, impute=TRUE, na.string='NA', plots=TRUE)
M_filter$M.clean[1:5,1:5]
dim(M_filter$M.clean)
SNP_pca <- snp.pca(M=M_filter$M.clean, label=FALSE, ncp=10)
ls(SNP_pca)
SNP_pca$eigenvalues  # eigenvalue and their var.explained
head(SNP_pca$pca.scores)
SNP_pca$plot.pca
SNP_pca$plot.scree

# PCA plot by family (17 groups)
grp <- as.factor(pheno.apple$Family)
SNP_pca_grp <- snp.pca(M=M_filter$M.clean, groups=grp)
SNP_pca_grp$plot.pca
```

```
## End(Not run)
```

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

Description

Generates a full matrix form from a sparse form matrix as provided by ASReml-R. The sparse form has three columns per line, corresponding to the set: `Row,Col,Value`. This matrix is defined by a lower triangle row-wise of the full matrix and is sorted as columns within row.

Usage

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

Arguments

`K`	A kinship matrix in sparse form.

Details

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.

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`.

References

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

Examples

```
## Not run: #'
data(geno.apple)
G <- G.matrix(M=geno.apple, method='VanRaden', na.string='NA', sparseform=TRUE)
head(G$G.sparse)
head(attr(G$G.sparse, 'rowNames'))
G <- sparse2full(K=G$G.sparse)
G[1:5,1:5]

## End(Not run)
```

Index