Package 'pedicure'

May 7, 2021

Title pedigree tools

Version 2.0.1

Date 2019-07-24

Author David Butler [aut, cre]

Copyright The University of Wollongong

Maintainer David Butler <dbutler@uow.edu.au>

Description pedigree tools.

Depends R (>= 3.5.0), pedigree

Suggests asreml, Matrix

Imports methods

License CC BY-ND 4.0

RoxygenNote 7.1.0

R topics documented:

A.dot *Dot file reporesenting a pedigree tree.*

Description

Exports the relationship structure in a relationship matrix as a directed graph in a dot file suitable for plotting.

Usage

```
A.dot(A, width = NA, dotfile = "A.dot")
```

Arguments

A	The numerator relationship matrix.
width	Width of the graph node character identifiers. If `NA` (the default) the complete identity names are written to the dot file.
dotfile	The output dot file name, default is `"dot.dot"`.

Details

The matrix `A` can be decomposed as `A=LDU`, where `U=t(L)` and `inv(L)` represents the directed graph of additive genetic relationships in the pedigree. This matrix can be converted to a graph object in dot file format suitable for plotting by `Graphviz`, say.

The function calls `zapsmall` to round nuisance values to zero; this can be controlled through the `"digits"` component of `options()`

Value

A two column matrix of the graph node relationships.

Side effects

The `dotfile` is written to the working directory.

---

ainv *Calculate an inverse relationship matrix.*

Description

Generates an inverse relationship matrix in sparse triplet form from a pedigree data frame.

Usage

```
ainv(
  pedigree,
  fgen = list(character(0), 0.01),
  gender = character(0),
  groups = 0,
  groupOffset = 0,
  selfing = NA,
  inBreed = NA,
  mgs = FALSE,
  mv = c("NA", "0", "*"),
  psort = FALSE
)
```

Arguments

`pedigree` A data frame where the first three columns correspond to the identifiers for the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or NA in the parental columns.

`fgen` An optional list of length 2 where `fgen[[1]]` is a character string naming the column in `pedigree` that contains the level of selfing or the level of inbreeding of an individual. In `pedigree[,fgen[[1]]]`, 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not as individuals), they will be assumed base non-inbred individuals unless their inbreeding level is set with `fgen[[2]]`, where `0 < fgen[[2]] < 1` is the inbreeding level of such individuals.

`gender` An optional character string naming the column of `pedigree` that codes for the gender of an individual. `pedigree[,gender]` is coerced to a factor and must only have two (arbitrary) levels, the first of which is taken to mean "male". An inverse relationship matrix is formed for the X chromosome as described by *Fernando and Grossman, 1990* for species where the male is XY and the female is XX.

`groups` An integer scalar (g) indicating genetic groups in the pedigree. The first g lines of the pedigree identify the genetic groups (with zero in both the male and female parent columns). All other rows must specify one of the genetic groups as the male or female parent if the actual parent is unknown. The default is $g = 0$.

`groupOffset` A numeric scalar *e > 0* added to the diagonal elements of A^{-1} pertaining to `groups`, shrinking the group effects by e. When a constant is added, no adjustment of the degrees of freedom is made for genetic groups. Set to -1 to add no offset but to suppress insertion of constraints where empty groups appear; the empty groups are then not counted in the degress of freedom adjustment. The default is $e = 0$.

`selfing` A numeric scalar (s) allowing for partial selfing when the third field of `pedigree` is unknown. It indicates that progeny from a cross where the male parent is unknown is assumed to be from selfing with probability s and from outcrossing with probability *(1-s)*. This is appropriate in some forestry tree breeding studies where seed collected from a tree may have been pollinated by the mother tree or pollinated by some other tree (*Dutkowski and Gilmour, 2001*). Do not use the `selfing` argument in conjunction with `inBreed` or `mgs`.

inBreed A numeric scalar (default NA) giving the inbreeding coefficient for *base* individuals. This argument generates the numerator relationship matrix for inbred lines. Each cross is assumed to be selfed several times to stabilize as an inbred line as is usual for cereal crops, for example, before being evaluated or crossed with another line. Since inbreeding is usually associated with strong selection, it is not obvious that a pedigree assumption of covariance of 0.5 between parent and offspring actually holds. The inBreed argument cannot be used in conjunction with selfing or mgs.

mgs If TRUE (default FALSE), the third identity in the pedigree is the male parent of the female parent (maternal grand-sire) rather than the female parent.

mv A character vector of missing value indicators; elements of pedigree that exactly match any of the members of mv are treated as missing.

psort If TRUE (default FALSE), the pedigree data frame is returned in founder order after any insertions and permutations.

Details

Uses the method of *Meuwissen and Luo, 1992* to compute the inverse relationship matrix directly from the pedigree.

Value

A three-column matrix with class ginv holding the lower triangle of the inverse relationship matrix in sparse form. The first 2 columns are the *row* and *column* indices, respectively, and the third column holds the inverse matrix element itself. Sort order is columns within rows, that is, the lower triangle row-wise. This matrix has attributes:

rowNames A character vector of identifiers for the rows of the matrix.

inbreeding A numeric vector containing the inbreeding coefficient for each individual, calculated as diag(A-I).

geneticGroups A numeric vector of length 2 containing the groups and groupOffset arguments.

logdet The log determinant.

Examples

```
## Not run:

# Simple pedigree

ped <- data.frame(me = c(1,2,3,4,5,6,7,8,9,10),
                  dad = c(0,0,0,1,1,2,4,5,7,9),
                  mum = c(0,0,0,1,1,2,6,6,8,9))
p.ai <- ainv(ped)

# Known filial generation

pdfg <- data.frame(me = c(1,2,3,4,5,6,7),
                   dad = c(0,0,1,1,1,1,1),
                   mum = c(0,0,0,2,2,2,2),
                   fgen = c(0.8,0.0,2.0,0.0,2.0,3.0,0.0))
pdfg.ai <- ainv(pdfg,fgen=list('fgen',0.4))
pdfg.mat <- sparse2mat(pdfg.ai)
```

```
zapsmall(solve(pdfg.mat))
zapsmall(cbind(pdfg.a$inbreeding,diag(pdfg.mat)))

## End(Not run)
```

`AIsweep` *Remove individuals from an inverse relationship matrix.*

Description

Deletes individuals from an inverse numerator relationship matrix by sweeping out the respective rows.

Usage

```
AIsweep(ginv, keep)
```

Arguments

`ginv` A matrix containing an inverse numerator relationship matrix in three column co-ordinate form with a `rowNames` attribute.

`keep` A character vector naming the individuals to retain.

Value

A three column sparse coordinate form matrix in row major order with attribute `"INVERSE"` set to `TRUE`.

`amat` *Relationship matrix*

Description

Calculate the numerator relationship matrix from a pedigree.

Usage

```
amat(ped, keep = rep(TRUE, nrow(ped)))
```

Arguments

`ped` A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. Founders use 0 (zero) or `NA` in the parental columns.

`keep` A logical vector identifying the rows of ped to retain.

Value

A `matrix` object containing the numerator relationship matrix.

Side-effects

A file "A.txt" is created

checkPedigree *Check a pedigree file*

Description

Check a pedigree file for missing or out of order founders.

Usage

```
checkPedigree(
  pedigree,
  fgen = list(character(0), 0.01),
  gender = character(0),
  mv = c("0", "*", " "),
  verbose = FALSE
)
```

Arguments

pedigree	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or NA in the parental columns.
fgen	An optional list of length 2 where fgen[[1]] is a character string naming the column in pedigree that contains the level of selfing or the level of inbreeding of an individual. In pedigree[,fgen[[1]]], 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not as individuals), they will be assumed base non-inbred individuals unless their inbreeding level is set with fgen[[2]] where 0<fgen[[2]]<1 is the inbreeding level of such individuals.
gender	An optional character string naming the column of pedigree that codes for the gender of an individual. pedigree[,gender] is coerced to a factor and must only have two (arbitrary) levels, the first of which is taken to mean "male". An inverse relationship matrix is formed for the X chromosome as described by *Fernando and Grossman (1990)* for species where the male is XY and the female is XX.
mv	Missing value indicator; elements of pedigree that exactly match any element of mv are treated as missing.
verbose	If TRUE details of pedigree insertions and relocations are printed; the default is FALSE to supress printing.

Value

A data frame containing the expanded or re-ordered pedigree.

`count_gen` *Count generation number.*

Description

Counts the generation number for each individual in a pedigreee.

Usage

```
count_gen(ped)
```

Arguments

`ped`	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or `NA` in the parental columns.

Value

A numeric vector of length `nrow(ped)` containing the generation number for each individual.

`equiv.mm` *Equivalence classes for (near) co-incident genetic markers.*

Description

Identify redundant genetic genetic markers based on the Hamming distance between marker pairs.

Usage

```
equiv.mm(M, na.match = FALSE, threshold = 0.05)
```

Arguments

`M`	An `n x m` matrix of marker scores for m markers on n individuals. The matrix must be numeric and capable of being coerced to integer values.
`na.match`	If `FALSE` (the default), `NA` in either marker counts as a mismatch, that is `NA`s match nothing. If `TRUE`, `NA` in either marker counts as a match, that is `NA`s match everything.
`threshold`	Marker pairs whose Hamming distance is less than `threshold` (the default is 0.05) are deemed equivalent and assigned to an equivalence class.

Details

For a given pair of markers, the Hamming distance is calculated as the total number of mismatches scaled by n. If `na.match = FALSE`, a missing value in either marker will count as a mismatch and contribute to the Hamming distance summation.

Beginning with column 1 of `M`, the $m(m-1)/2$ pairs of markers are scanned sequentially and markers allocated to equivalence classes if the pairwise Hamming distance is less than `threshold`. The algorithm uses the method of D. Eardley in Section 8.6 of *Press et al., 2002.* Pairwise Hamming distances are computed on the fly from `M` to avoid storing the distance matrix.

Value

A list of length the number of equivalence classes, where each component is a numeric vector of equivalent marker column numbers. The names of the list are the root marker numbers of the equivalence classes and can be coerced to `numeric`.

References

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2002). *Numerical Recipes in C++. The Art of Scientific Computing*, Second edition. Cambridge University Press.

`hdist` *Calculate Hamming distances for genetic markers.*

Description

Calculate the Hamming distance for each marker pair from a matrix of marker scores.

Usage

```
hdist(M, na.match = FALSE)
```

Arguments

`M`	An `n x m` matrix of marker scores for m markers on n individuals. The matrix must be numeric and capable of being coerced to integer values.
`na.match`	If `FALSE` (the default), `NA` in either marker counts as a mismatch, that is `NA`s match nothing. If `TRUE`, `NA` in either marker counts as a match, that is `NA`s match everything.

Details

For a given pair of markers, the Hamming distance is calculated as the total number of mismatches scaled by n. If `na.match = FALSE`, a missing value in either marker will count as a mismatch and contribute to the Hamming distance summation.

Value

A `dist` class numeric vector of distances being the strict lower triangle of the distance matrix in column major order.

`ld` *Linkage disequilibrium*

Description

Calculate the LD coefficient r from genotype marker scores.

Usage

```
ld(x, method = c("rh", "em"), verbose = FALSE)
```

Arguments

`x`	An $n \times m$ matrix of biallelic marker scores for n individuals and m markers. The scores are assumed coded as {-1, 0, 1} representing genotypes `aa`, `aA` and `AA`, respectively, with missing values coded as `NA`.
`method`	If method is `"rh"` (the default), r is calculated for each pair of markers in `x` using the approximate method of *Rogers and Huff*, otherwise a modified EM algorithm is used.
`verbose`	If `TRUE` and `method="em"`, any irregularities in calculating the haplotype frequencies are reported. Warning: this may generate considerable console output and would be most useful in debugging a limited set of markers. The default is `FALSE`.

Details

This function accessess a modification to the EM method of *Excoffier and Slatkin, 1995*, and the approximation of *Rogers and Huff, 2009*. The approximation works with pairs of biallelic loci, and the EM method has been modified to also deal only with pairs of loci.

The advantages of the approximate method are its ease of coding and speed improvement over the EM optimisation method, see *Rogers and Huff, 2009* for details. This implementation uses a simple high level multithreading strategy for both methods, so timing is proportional to the number of cores.

The underlying compiled code is from `https://github.com/alanrogers/covld` with minor modifications to capture printed results in **R**.

Value

The $m(m+1)/2$ LD r_{ij} values stored as a dense symmetric `"dspMatrix"` class Matrix; the diagonal elements are set to 1.

References

Rogers AR, Huff C (2009). “Linkage Disequilibriun Between Loci with Unknown Phase.” *Genetics*, **182**, 839-844.

Excoffier L, Slatkin M (1995). “Maximum Likelihood Estimation of Molecular Haplotype Frequencies in a Diploid Population.” *Molecular Biology Evolution*, **12**, 921-927.

`ldu` *Generalised Cholesky decomposition.*

Description

Generalised Cholesky decomposition of a numerator relationship matrix.

Usage

```
ldu(A)
```

Arguments

`A`	The numerator relationship matrix.

Details

Determines the modified Cholesky decomposition of a positive-definite matrix A given by $A = LDL^T$, where L is lower triangular with ones on the diagonal, and D is a diagonal matrix. The function uses the `MCHOL` function in *McLeod and Holanda Sales, 1983*.

Value

A list with the following components:

L A lower triangular matrix.

D A diagonal matrix.

References

McLeod AI, Holanda Sales PR (1983). “Algorithm AS 191.2 An Algorithm for Approximate Likelihood Calculation of ARMA and Seasonal ARMA Models.” *Applied Statistics*, **32**, 211-223.

`mask` *Mask individual identifiers*

Description

Replace the identifiers of individuals in the pedigree with auto generated names using alphabetic combinations.

Usage

```
mask(pedigree, data = pedigree[, 1], mv = c("0", "*", " "))
```

Arguments

`pedigree`	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively.
`data`	The names column in data order (with dups); the default is the first column of the pedigree.
`mv`	missing value indicator.

Value

A list with components `pedigree` and `data` containing the pedigree with renamed individuals and their aliases, respectively.

`mat2sparse` *Convert a dense matrix.*

Description

Convert a dense matrix to a sparse matrix in three-column coordinate form.

Usage

```
mat2sparse(x, rowNames = dimnames(x)[[1]], tol = 1e-09)
```

Arguments

`x`	A dense matrix padded with zeros where needed.
`rowNames`	A character vector set as the "rowNames" attribute of the returned sparse matrix. The default is the `dimnames` attribute of the leading dimension of `x`.
`tol`	Matrix elements whose absolute value is less than tol are considered zero. The default is 1e-9.

Details

The `dimnames(X)[[1]]` attribute of `x` is preserved and returned as the `rowNames` attribute of the returned matrix.

Value

A matrix holding the lower triangle of a sparse symmetric matrix in coordinate form in row major order.

`mmat` *Realized relationship matrix.*

Description

Calculates an IBS (possibly) additive relationship matrix.

Usage

```
mmat(
  x,
  center = FALSE,
  scale = TRUE,
  na.method = c("mean", "svd", "knn", "none"),
  min.af = NULL,
  max.mv = NULL,
  ev = 10,
  knn = 10,
  maxit = 10,
  imputed = FALSE
)
```

Arguments

`x` An $n \times m$ matrix of biallelic marker scores for n individuals and m markers. The scores are assumed coded as {-1, 0, 1} with missing values coded as `NA`.

`center` If `TRUE`, the columns of `x` (or the imputed matrix W) are centred.

`scale` If `TRUE` (the default), the realized association matrix xx' (or WW') is scaled by the reciprocal of the average marker variance (v); that is, xx'/v. If `scale` is numeric then the realized association matrix is $xx' * scale$.

`na.method`

- **mean** missing values within each marker column are replaced by the column mean;
- **knn** a missing value for a marker is replaced by the mean of its `knn` nearest neighbours with non-missing data for that individual;
- **svd** iteratively impute missing values in a predicted (current) W_1 using the `ev` most significant eigenvalues from a singular value decomposition of the previous W_0;
- **none** no imputation.

`min.af` Minimum allele frequency below which markers are removed; monomorphic markers are automatically removed. If not set, the default is `1/(2*nrow(x))`.

`max.mv` Maximum proportion of missing values allowed for a marker; completely missing markers are removed. If not set, the default is `1-1/(2*nrow(x))`.

`ev` If `na.method` = `svd`, the number of significant eigenvectors to use when imputing missing values (the default is 10).

`knn` If `na.method` = `knn`, the cluster size of similar markers to use when imputing missing values (the default is 10).

`maxit` If `na.method` = `svd`, the maximum number of iterations used to impute missing values (the default is 10). The iteration sequence may terminate before `maxit` if the convergence criterion is met (see details).

`imputed` If `TRUE` (the default is `FALSE`), a list of length two is returned with components `rrm` and `inputed` containing the realized relationship and the imputed marker matrices, respectively. If `FALSE`, only the relationship matrix is returned.

Details

The realized relationship matrix is computed following two pre-processing steps: 1) the matrix `x` is scanned for consistency against the arguments `min.af` and `max.mv`, and 2) any missing values are imputed using na.method. Monomorphic and completely missing markers are removed in step (1). If center=TRUE, the imputed matrix W is optionally centred, and if scale=TRUE W is scaled by the reciprocal of the average marker variance (*VanRaden 2008*) before computing WW'. The imputation methods svd and knn replace the missing values in x with their column means before proceeding.

The svd method (*Troyanskaya etal 2001*) uses the first ev terms of the SVD of W to replace the previously missing values with their regression predictions from the SVD. The procedure iterates until convergence is achieved or maxit is exceeded. Convergence is declared when $|RSS_0 - RSS_1|/RSS_1 < 0.02$ (*Rutkoski 2013*), where RSS is the residual sum of squares between the non-missing values and their predictions from the SVD, and RSS_0 and RSS_1 are the RSS values from successive iterations.

The knn procedure (*Troyanskaya etal 2001*) orders the (marker) neighbours of the j^{th} marker according to euclidean distance, replacing the i^{th} missing value with the average of the non-missing values for genotype i from the knn nearest neighbours of marker j.

If imputed is TRUE, the imputed marker matrix is returned with monomorphic and completely missing markers removed, as well as those screened out by min.af and max.mv.

If scale is TRUE, the reciprocal of the average marker variance is returned in an attribute scale of the returned matrix.

Value

A list with components rrm and imputed, or just the relationship matrix object depending on the value of the imputed argument. For method=svd, the convergence sequence (RSS) is included as attribute rss of the returned object.

References

Rutkoski JE, Poland J, Jannink J, Sorrells ME (2013). “Imputation of Unordered Markers and the Impact on Genomic Selection Accuracy.” *G3 Genes Genomes Genetics*, **3**, 427-439.

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T. Tibshirani R, Botstein D, Altman RB (2001). “Missing value estimation methods for DNA microarrays.” *Bioinformatics*, **17**, 520-525.

VanRaden PM (2008). “Efficient Methods to Compute Genomic Predictions.” *Journal of Dairy Science*, **91**, 4414-4423.

Examples

```
## Not run:
## Wheat data from Rutkoski (2013) with 70\
data(WW_VersionNA70_rep1)
## the imputed marker matrix
W <-  mmat(WW_VersionNA70_rep1,na.method="knn",knn=4, imputed=TRUE)
## the realized relationship matrix
WW <-  mmat(WW_VersionNA70_rep1,na.method="knn",knn=4)

## End(Not run)
```

`mmat.ev` *Eigenvalues.*

Description

Calculate the eigenvalues of a symmetric matrix.

Usage

```
mmat.ev(x, k = nrow(x), tol = -1)
```

Arguments

`x`	A real symmetric matrix.
`k`	Calculate the k largest eigenvalues and eigenvectors; the default is `nrow(x)`.
`tol`	Error tolerance to which each eigenvalue is required. If `tol=-1` (the default), it is set to a machine dependent *safe* minimum.

Value

V An $n \times k$ numeric matrix of eigenvectors.

lambda The vector of eigenvalues of length k..

`mmat.svd` *Singular value decomposition.*

Description

Calculate the singular value decomposition of a rectangular matrix.

Usage

```
mmat.svd(x)
```

Arguments

`x`	A real $m \times n$ matrix.

Value

A list with three components:

u An $m \times m$ matrix of left singular vectors.

v An $n \times n$ transposed matrix of right singular vectors.

d The `min(m,n)` vector of singular values.

nrm *Simulated numerator relationship matrices.*

Description

Calculates components of the genetic covariance matrix for a given population defined in an ancestral tree.

Usage

```
nrm(
  pedigree,
  fgen = list(character(0), 0),
  nsim = 2000,
  rm = c("A", "D", "C", "Ct", "Dh", "Di", "E"),
  form = c("full", "upper"),
  mv = c("NA", "0", "*")
)
```

Arguments

`pedigree` A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or `NA` in the parental columns.

`fgen` An optional list of length 2 where `fgen[[1]]` is a character string naming the column in `pedigree` that contains the level of selfing (f) or the level of inbreeding b) of an individual. In `pedigree[,fgen[[1]]]`, 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not as individuals), they will be assumed base non-inbred individuals unless their inbreeding level is set with `fgen[[2]]` where `0<fgen[[2]]<1` is the inbreeding level of such individuals. For use in the simulation, f is determined from b using $b = 1 - (0.5)^f$ if `0<fgen[[2]]<1`.

`nsim` The number of traversals of the ancestral tree; default 2000.

`rm` A character vector naming the relationship matrices to be returned; the default is to return all simulated matrices. Setting `rm=c("A","D")` will return both the additive and dominance relationship matrices, for example; see *Lynch and Walsh (1998)* for details on the components of the genetic covariance matrix.

`form` if `"full"`, return the complete relationship matrices, otherwise if `form="upper"` the upper triangle is returned.

`mv` Missing value indicator; elements of `pedigree` that exactly match any element of `mv` are treated as missing.

Details

Initially, each founder individual starts with two distinct (and unique) alleles. For each simulation, the pedigree is traversed and two genes are sampled, one from each parent, and assigned to each individual. If $f > 0$, the assigned genes are sampled with replacement f times, resulting in homozygosity at the locus if f is sufficiently large. The genes of all individuals are examined pairwise

(i, j), and counts of events d_{ijs} contributing to identity state S (see *Lynch and Walsh, 1998*, Chapter 7) are accumulated. If the genes for individual i are identical, then the count of IBD events within the i^{th} individual, F_i, is incremented.

The procedure is repeated `nsim` (N) times, and the pairwise coefficients D_{ijs} for state S estimated as $D_{ijs} = d_{ijs}/N$, for $s = 1 : 9$; the inbreeding coefficients are estimated as $F_i = F_i/N$. The elements of the various relationship matrices are estimated using the (pairwise) coefficients of the contributing identity states as defined in *Lynch and Walsh (1998)*.

References

Lynch M, Walsh B (1998). *Genetics and Analysis of Quantitative Trails*. Sinaeur Associates, Sunderland, Massachusetts.

Verbyla AP, Butler DG (2011). "Estimating the Dominance Relationship Matrix Using a Simulation Approach." In *The International Biometric Society, Australasian Region Conference*.

`offspring` *Count descendents.*

Description

Counts the number of offspring for each individual in a pedigree.

Usage

```
offspring(ped)
```

Arguments

`ped`	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or `NA` in the parental columns.

Value

A numeric vector of length `nrow(ped)` containing the number of offspring for each individual.

`ped.dot` *Dot file representing a pedigree tree.*

Description

Exports the relationship structure in a pedigree as a directed graph in a dot file suitable for plotting.

Usage

```
ped.dot(
  ped,
  keep = rep(TRUE, nrow(ped)),
  dotfile = "ped",
  numeric = TRUE,
  url = "",
  height = 0.5,
  width = 0.75,
  rotate = 0
)
```

Arguments

`ped`	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. Founders use 0 (zero) or `NA` in the parental columns.
`keep`	A logical vector identifying the rows of `ped` to retain. The default is `rep(TRUE,nrow(ped))`.
`dotfile`	The output dot file primary name, default is `"ped"`; the suffix `".dot"` is appended to the file name. Also used to label the pedigree tree.
`numeric`	Convert `ped` to integer values for graphing. The default is `TRUE`.
`url`	If not `""` (the default) then `url` is linked to the resulting graph.
`height`	Node height.
`width`	Node width
`rotate`	If `rotate=90` landscape mode is selected; the default is `0`.

Details

The resulting dot file can be edited prior to rendering with *Graphviz*, say, or conversion to a graphics file format with the *dot* application (part of the *Graphviz* package). The default draws founder individuals in rectangles, offspring in ellipses, and offspring with a single parent in circles.

Side effects

The `dotfile` is written to the working directory using `sink()`.

recode.mm *Recode a marker matrix.*

Description

Recode the matrix of molecular marker scores using a simple mapping.

Usage

```
recode.mm(
  x,
  map = matrix(c(0, 1, 2, -1, 0, 1), ncol = 2),
  char.convert = TRUE,
  transpose = FALSE
)
```

Arguments

`x` An $n \times m$ matrix of scores for m molecular markers on n individuals; missing values are allowed and retained.

`map` A two column matrix where `map[,1]` are the existing scores in `x`, and `map[,2]` are the new scores. The default assumes scores `(0,1,2)` are to be recoded as `(-1,0,1)`.

`char.convert` If `TRUE` (the default) and `x` (and therefore `map`) is of type `character`, `x` will attempted to be coerced to `numeric` after recoding.

`transpose` If `TRUE` (the default is `FALSE`), transpose `x` if rows are markers and columns are individuals.

Value

The recoded (and possibly transposed) matrix with missing values preserved.

`sparse2mat` *Convert a sparse matrix.*

Description

Convert a sparse matrix held in three-column coordinate form to a dense matrix.

Usage

```
sparse2mat(x)
```

Arguments

`x` A matrix (or data frame) holding the lower triangle of a sparse symmetric matrix in coordinate form in row major order; typically the result of a call to `ainv`.

Details

The attribute `rowNames` of `x` is preserved and returned as the `dimnames` attribute of the returned matrix.

Value

A dense matrix containing `x` padded with zeros.

trim *Trim a pedigree.*

Description

Trim a pedigree to a given data frame; non-informative individuals without data are removed.

Usage

```
trim(ped, data, gen = NULL)
```

Arguments

ped	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or NA in the parental columns.
data	A data frame with a component named names(ped)[1] containing the identities of the individuals with data.
gen	The number of generations to keep before those with data. The default is max(count_gen(mmd)).

Details

Call chkPed (library asreml) to add any missing founders, resolve fgen, and reorder the pedigree (if necessary) prior to pruning.

Value

A data frame with the same structure as ped, with the pruned individuals removed.

trim.par *Extract parental records from a (pruned) pedigree.*

Description

Trim a pedigree to the set of unique identifiers in the parental columns.

Usage

```
trim.par(ped)
```

Arguments

ped	A data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively; typically the result of a previous call to trim.

Value

A data frame with the same structure as ped, with the pruned individuals removed.

Index