Package ‘pedigreeTools’

October 12, 2024

Version 0.3

Date 2024-10-10

Title Versatile Functions for Working with Pedigrees

Author Ana Ines Vazquez [aut],
Douglas Bates [aut],
Siddharth Avadhanam [aut],
Paulino Perez Rodriguez [aut, cre],
Gregor Gorjanc [ctb]

Maintainer Paulino Perez Rodriguez <`perpdgo@colpos.mx`>

Description Tools to sort, edit and prune pedigrees and to extract the inbreeding coefficients and the relationship matrix (includes code for pedigrees from self-pollinated species).
The use of pedigree data is central to genetics research within the animal and plant breeding communities to predict
breeding values. The relationship matrix between the individuals can be derived from pedigree structure
('Vazquez et al., 2010') <doi:10.2527/jas.2009-1952>.

Depends R(>= 3.0.0), methods

Imports Matrix (>= 1.0)

LazyLoad yes

License GPL-3

URL `https://github.com/Rpedigree/pedigreeTools/`

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-10-12 03:50:02 UTC

Contents

`Dmat` *Mendelian sampling variance*

Description

Determine the diagonal factor in the decomposition of the relationship matrix A as TDT' where T is unit lower triangular.

Usage

```
Dmat(ped, vector = TRUE)

getD(ped, vector = TRUE)

getDInv(ped, vector = TRUE)
```

Arguments

`ped`	`pedigree`
`vector`	logical, return a vector or sparse matrix

Value

a numeric vector

Functions

- `getD()`: Mendelian sampling variance
- `getDInv()`: Mendelian sampling precision (= 1 / variance)

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(D <- getD(ped))
(DInv <- getDInv(ped))

# Test for correctness
DExp <- c(1.00, 1.00, 0.50, 0.75, 0.50, 0.46875)
stopifnot(!any(abs(D - DExp) > .Machine$double.eps))

DInvExp <- 1 / DExp
stopifnot(!any(abs(DInv - DInvExp) > .Machine$double.eps))
```

`editPed` *Edits a disordered or incomplete pedigree*

Description

Edits a disordered or incomplete pedigree by: 1) adding labels for the sires and dams not listed as labels before and 2) ordering pedigree based on recursive calls to `getGenAncestors`.

Usage

```
editPed(sire, dam, label, verbose = FALSE)
```

Arguments

`sire`	integer vector or factor representation of the sires
`dam`	integer vector or factor representation of the dams
`label`	character vector of labels
`verbose`	logical to print the row of the pedigree that the function is ordering. Default is FALSE.

Value

a data frame with the pedigree ordered.

Examples

```
ped <- data.frame(sire=as.character(c(NA,NA,NA,NA,NA,1,3,5,6,4,8,1,10,8)),
                  dam=as.character(c(NA,NA,NA,NA,NA,2,2,NA,7,7,NA,9,9,13)),
                  label=as.character(1:14))
ped <- ped[sample(replace=FALSE, 1:14),]
ped <- editPed(sire = ped$sire, dam = ped$dam, label = ped$label)
ped <- with(ped, pedigree(label = label, sire = sire, dam = dam))
```

`getA` *Additive relationship matrix*

Description

Returns the additive relationship matrix for the pedigree.

Usage

```
getA(ped, labs = NULL)
```

Arguments

`ped`	`pedigree`
`labs`	a character vector or a factor giving individual labels to which to restrict the relationship matrix and corresponding factor. If `labs` is a factor then the levels of the factor are used as the labels. Default is the complete set of individuals in the pedigree.

Value

matrix (`dsCMatrix` - symmetric sparse)

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(A <- getA(ped))

# Test for correctness
AExp <- matrix(data = c(1.0000, 0.0000, 0.5000, 0.5000, 0.5000, 0.2500,
                        0.0000, 1.0000, 0.5000, 0.0000, 0.2500, 0.6250,
                        0.5000, 0.5000, 1.0000, 0.2500, 0.6250, 0.5625,
                        0.5000, 0.0000, 0.2500, 1.0000, 0.6250, 0.3125,
                        0.5000, 0.2500, 0.6250, 0.6250, 1.1250, 0.6875,
                        0.2500, 0.6250, 0.5625, 0.3125, 0.6875, 1.1250),
               byrow = TRUE, nrow = 6)
stopifnot(!any(abs(A - AExp) > .Machine$double.eps))
stopifnot(Matrix::isSymmetric(A))
```

getAInv *Inverse of the additive relationship matrix*

Description

Returns the inverse of additive relationship matrix for the pedigree.

Usage

```
getAInv(ped)
```

Arguments

`ped`	`pedigree`

Value

matrix (`dsCMatrix` - symmetric sparse)

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(AInv <- getAInv(ped))

# Test for correctness
AInvExp <- matrix(data = c( 1.833,  0.500, -1.000, -0.667,  0.000,  0.000,
                            0.500,  2.033, -1.000,  0.000,  0.533, -1.067,
                           -1.000, -1.000,  2.500,  0.500, -1.000,  0.000,
                           -0.667,  0.000,  0.500,  1.833, -1.000,  0.000,
                            0.000,  0.533, -1.000, -1.000,  2.533, -1.067,
                            0.000, -1.067,  0.000,  0.000, -1.067,  2.133),
                  byrow = TRUE, nrow = 6)
stopifnot(!any(abs(round(AInv, digits = 3) - AInvExp) > .Machine$double.eps))
AInvExp <- solve(getA(ped))
stopifnot(!any(abs(round(AInv, digits = 14) - round(AInvExp, digits = 14)) > .Machine$double.eps))
stopifnot(is(AInv, "sparseMatrix"))
stopifnot(Matrix::isSymmetric(AInv))
```

---

getASelfing	*Extends the pedigree according to number of selfing cycles and also optionally computes the Additive Relationship Matrix for that pedigree.*

Description

Extends the pedigree according to number of selfing cycles and also optionally computes the Additive Relationship Matrix for that pedigree.

Usage

```
getASelfing(
  ID,
  Par1,
  Par2,
  nCycles,
  nCyclesDefault,
  sepChar = "-F",
  verbose = FALSE,
  fileNewPed = NULL,
  computeA = TRUE
)
```

Arguments

`ID`	is a vector of individual IDs
`Par1`	vector of IDs of one of the parents
`Par2`	vector of IDs of the other parent
`nCycles`	vector that indicates number of selfing cycles for each individual.
`nCyclesDefault`	default value of nCycles
`sepChar`	character, used for expanded pedigree IDs
`verbose`	logical, print progress
`fileNewPed`	Output csv file (comma separated value) with columns 'label', 'sire', 'dam', with the full pull pedigree expanded taking into account the selfing cycles
`computeA`	Indicates if the A matrix is to be computed

Value

Returns A matrix computed for the extended pedigree if computeA=TRUE

`getASubset` *Subset of additive relationship matrix*

Description

Returns subset of the additive relationship matrix for the pedigree.

Usage

```
getASubset(ped, labs)
```

Arguments

`ped`	`pedigree`
`labs`	a character vector or a factor giving individual labels to which to restrict the relationship matrix and corresponding factor using Colleau et al. (2002) algorithm. If `labs` is a factor then the levels of the factor are used as the labels. Default is the complete set of individuals in the pedigree.

Value

matrix (`dsCMatrix` - symmetric sparse)

References

Colleau, J.-J. An indirect approach to the extensive calculation of relationship coefficients. Genet Sel Evol 34, 409 (2002). https://doi.org/10.1186/1297-9686-34-4-409

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(A <- getA(ped))
(ASubset  <- A[4:6, 4:6])
(ASubset2 <- getASubset(ped, labs = 4:6))

(ASubset3  <- A[6:4, 6:4])
(ASubset4 <- getASubset(ped, labs = 6:4))

# Test for correctness
stopifnot(!any(abs(ASubset - ASubset2) > .Machine$double.eps))
stopifnot(!any(abs(ASubset3 - ASubset4) > .Machine$double.eps))
stopifnot(Matrix::isSymmetric(ASubset2))
stopifnot(Matrix::isSymmetric(ASubset4))
```

`getGenAncestors` *Counts number of generations of ancestors for one subject. Use recursion.*

Description

Counts number of generations of ancestors for one subject. Use recursion.

Usage

```
getGenAncestors(ped, id, ngen = NULL)
```

Arguments

`ped`	data.frame with a pedigree and a column for the number of generations of each subject.
`id`	subject for which we want the number of generations.
`ngen`	number of generation

Value

a data frame object with the pedigree and generation of ancestors for subject id.

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
ped <- ped2DF(ped)
ped$id <- row.names(ped)
ped$generation <- NA
(tmp1 <- getGenAncestors(ped, id = 1))
(tmp2 <- getGenAncestors(ped, id = 4))
(tmp3 <- getGenAncestors(ped, id = 6))

# Test for correctness
stopifnot(tmp1$generation[1] == 0)
stopifnot(all(is.na(tmp1$generation[-1])))
stopifnot(all(tmp2$generation[c(1, 4)] == c(0, 1)))
stopifnot(all(is.na(tmp2$generation[-c(1, 4)])))
stopifnot(all(tmp3$generation == c(0, 0, 1, 1, 2, 3)))
```

getT *Gene flow from a pedigree*

Description

Get gene flow matrix from a pedigree.

Usage

```
getT(ped)
```

Arguments

`ped`	`pedigree`

Value

matrix (`dtCMatrix` - lower unitriangular sparse)

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(T <- getT(ped))

# Test for correctness
TExp <- matrix(data = c(1.00, 0.000, 0.00, 0.00, 0.0, 0,
                        0.00, 1.000, 0.00, 0.00, 0.0, 0,
                        0.50, 0.500, 1.00, 0.00, 0.0, 0,
                        0.50, 0.000, 0.00, 1.00, 0.0, 0,
                        0.50, 0.250, 0.50, 0.50, 1.0, 0,
                        0.25, 0.625, 0.25, 0.25, 0.5, 1),
               byrow = TRUE, nrow = 6)
stopifnot(!any(abs(T  - TExp) > .Machine$double.eps))
```

getTInv *Inverse gene flow from a pedigree*

Description

Get inverse gene flow matrix from a pedigree.

Usage

```
getTInv(ped)
```

Arguments

`ped`	`pedigree`

Value

matrix (`dtCMatrix` - lower unitriangular sparse)

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(TInv <- getTInv(ped))

# Test for correctness
TInvExp <- matrix(data = c( 1.0,  0.0,  0.0,  0.0,  0.0,  0.0,
                            0.0,  1.0,  0.0,  0.0,  0.0,  0.0,
                           -0.5, -0.5,  1.0,  0.0,  0.0,  0.0,
                           -0.5,  0.0,  0.0,  1.0,  0.0,  0.0,
                            0.0,  0.0, -0.5, -0.5,  1.0,  0.0,
                            0.0, -0.5,  0.0,  0.0, -0.5,  1.0),
                  byrow = TRUE, nrow = 6)
stopifnot(!any(abs(TInv  - TInvExp) > .Machine$double.eps))
stopifnot(is(TInv, "sparseMatrix"))
```

`inbreeding` *Inbreeding coefficients from a pedigree*

Description

Create the inbreeding coefficients according to the algorithm given in "Comparison of four direct algorithms for computing inbreeding coefficients" by Mehdi Sargolzaei and Hiroaki Iwaisaki, Animal Science Journal (2005) 76, 401–406.

Usage

```
inbreeding(ped)
```

Arguments

`ped`	`pedigree`

Value

the inbreeding coefficients as a numeric vector

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(F <- inbreeding(ped))

# Test for correctness
FExp <- c(0.000, 0.000, 0.000, 0.000, 0.125, 0.125)
stopifnot(!any(abs(F - FExp) > .Machine$double.eps))
```

`ped2DF` *Convert a pedigree to a data frame*

Description

Express a pedigree as a data frame with `sire` and `dam` stored as factors. If the pedigree is an object of class pedinbred then the inbreeding coefficients are appended as the variable `F`

Usage

```
ped2DF(x)
```

Arguments

`x`	`pedigree`

Value

a data frame

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
ped2DF(ped)
```

`pedigree` *Constructor for pedigree objects*

Description

A simple constructor for a pedigree object. The main point for the constructor is to use coercions to make the calls easier.

Usage

```
pedigree(sire, dam, label)
```

Arguments

`sire`	integer vector or factor representation of the sires
`dam`	integer vector or factor representation of the dams
`label`	character vector of individual labels

Value

an pedigree object of class pedigree

Note

`sire`, `dam` and `label` must all have the same length and all labels in `sire` and `dam` must occur in `label`

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
ped
```

`pedigree-class` *Pedigree class*

Description

Pedigree class

`prunePed` *Subsets a pedigree for a specified vector of individuals up to a specified number of previous generations using recursion.*

Description

Subsets a pedigree for a specified vector of individuals up to a specified number of previous generations using recursion.

Usage

```
prunePed(ped, selectVector, ngen = 2)
```

Arguments

`ped`	Data Frame pedigree to be subset
`selectVector`	Vector of individuals to select from pedigree
`ngen`	Number of previous generations of parents to select starting from selectVector.

Value

Returns Subsetted pedigree as a DataFrame.

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
```

`relfactor` *Relationship factor from a pedigree*

Description

Determine the right Cholesky factor of the relationship matrix for the pedigree `ped`, possibly restricted to the specific labels that occur in `labs`.

Usage

```
relfactor(ped, labs = NULL)

getL(ped, labs = NULL)
```

Arguments

`ped`	`pedigree`
`labs`	a character vector or a factor giving individual labels to which to restrict the relationship matrix and corresponding factor using Colleau et al. (2002) algorithm. If `labs` is a factor then the levels of the factor are used as the labels. Default is the complete set of individuals in the pedigree.

Details

Note that the right Cholesky factor is returned, which is upper triangular, that is from A = LL’ = R’R (lower (upper triangular) and not L (lower triangular) as the function name might suggest.

Value

matrix (`dtCMatrix` - upper triangular sparse)

Functions

- `getL()`: Relationship factor from a pedigree

References

Colleau, J.-J. An indirect approach to the extensive calculation of relationship coefficients. Genet Sel Evol 34, 409 (2002). https://doi.org/10.1186/1297-9686-34-4-409

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(L <- getL(ped))
chol(getA(ped))

# Test for correctness
LExp <- matrix(data = c(1.0000, 0.0000, 0.5000, 0.5000, 0.5000, 0.2500,
                        0.0000, 1.0000, 0.5000, 0.0000, 0.2500, 0.6250,
                        0.0000, 0.0000, 0.7071, 0.0000, 0.3536, 0.1768,
                        0.0000, 0.0000, 0.0000, 0.8660, 0.4330, 0.2165,
                        0.0000, 0.0000, 0.0000, 0.0000, 0.7071, 0.3536,
                        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6847),
               byrow = TRUE, nrow = 6)
stopifnot(!any(abs(round(L, digits = 4) - LExp) > .Machine$double.eps))
LExp <- chol(getA(ped))
stopifnot(!any(abs(L - LExp) > .Machine$double.eps))

(L <- getL(ped, labs = 4:6))
(LExp <- chol(getA(ped)[4:6, 4:6]))
stopifnot(!any(abs(L - LExp) > .Machine$double.eps))
```

`relfactorInv` *Inverse relationship factor from a pedigree*

Description

Get inverse of the left Cholesky factor of the relationship matrix for the pedigree ped.

Usage

```
relfactorInv(ped)

getLInv(ped)
```

Arguments

`ped`	`pedigree`

Details

Note that the inverse of the left Cholesky factor is returned, which is lower triangular, that is from A = LL' (lower inv(A) = inv(LL') = inv(L)' inv(L) (upper triangular).

Value

matrix (`dtCMatrix` - triangular sparse)

Functions

- `getLInv()`: Inverse relationship factor from a pedigree

Examples

```
ped <- pedigree(sire = c(NA, NA, 1,  1, 4, 5),
                dam =  c(NA, NA, 2, NA, 3, 2),
                label = 1:6)
(LInv <- getLInv(ped))
solve(Matrix::t(getL(ped)))

# Test for correctness
LInvExp <- matrix(data = c( 1.0000,  0.0000,  0.0000,  0.0000,  0.0000, 0.0000,
                            0.0000,  1.0000,  0.0000,  0.0000,  0.0000, 0.0000,
                           -0.7071, -0.7071,  1.4142,  0.0000,  0.0000, 0.0000,
                           -0.5774,  0.0000,  0.0000,  1.1547,  0.0000, 0.0000,
                            0.0000,  0.0000, -0.7071, -0.7071,  1.4142, 0.0000,
                            0.0000, -0.7303,  0.0000,  0.0000, -0.7303, 1.4606),
                  byrow = TRUE, nrow = 6)
stopifnot(!any(abs(round(LInv, digits = 4) - LInvExp) > .Machine$double.eps))
L <- t(chol(getA(ped)))
LInvExp <- solve(L)
```

```
stopifnot(!any(abs(LInv - LInvExp) > .Machine$double.eps))
stopifnot(is(LInv, ”sparseMatrix”))
```

Index