

Package ‘ggroups’

March 19, 2020

Title Pedigree and Genetic Groups

Version 2.0.3

Description Calculates additive and dominance genetic relationship matrices and their inverses, in matrix and tabular-sparse formats. It includes functions for checking and processing pedigree, as well as functions to calculate the matrix of genetic group contributions (Q), and adding those contributions to the genetic merit of animals (Quaas (1988) <doi:10.3168/jds.S0022-0302(88)79691-5>). Calculation of Q is computationally extensive. There are computationally optimized functions to calculate Q.

License GPL-3

LazyData true

URL <https://github.com/nilforooshan/ggroups>

BugReports <https://github.com/nilforooshan/ggroups/issues>

Suggests doParallel (>= 1.0.14), foreach (>= 1.4.4)

RoxygenNote 7.0.2

Encoding UTF-8

Repository CRAN

NeedsCompilation no

Author Mohammad Ali Nilforooshan [aut, cre]
(<<https://orcid.org/0000-0003-0339-5442>>)

Maintainer Mohammad Ali Nilforooshan <m.a.nilforooshan@gmail.com>

Date/Publication 2020-03-19 06:50:02 UTC

R topics documented:

gggroups-package	2
buildA	3
buildD	3
gghead	4
inb	5
mat2tab	6

offspring	6
pedcheck	7
peddown	8
pedup	8
pruneped	9
Qgpu	10
qmat	11
qmatL	11
qmatXL	12
renum	13
rg	13
smgsped	14
tab2mat	15
tabA	15
tabAinv	16
tabD	17
tabDinv	17
Index	19

 gggroups-package

Pedigree and genetic groups

Description

This package contains pedigree processing and analyzing functions, including functions for checking and renumbering the pedigree, making the additive and dominance pedigree relationship matrices and their inverses, in matrix and tabular formats, as well as functions related to genetic groups.

Details

First, it is recommended to check the pedigree data.frame with the pedcheck function. Pedigree relationship matrix and its inverse are fundamentals in the conventional and modern animal breeding. The concept of genetic groups stems from the fact that not all the unknown parents are of the same genetic level. The genetic group contribution matrix (\mathbf{Q}) is required to weight and add genetic group effects ($\hat{\mathbf{g}}$) to the genetic merit of animals ($\hat{\mathbf{u}}$), which is equal to $\mathbf{Q}\hat{\mathbf{g}} + \hat{\mathbf{u}}$ (Quaas, 1988). Calculating \mathbf{Q} is computationally challenging, and for large pedigree, large RAM and long computational time is required. Therefore, the functions qmatL and its parallel version, qmatXL are introduced. Overlap between sire and dam genetic groups is supported.

Author(s)

Mohammad Ali Nilforooshan <m.a.nilforooshan@gmail.com>

References

Mrode, R. A. 2005. Linear Models for the Prediction of Animal Breeding Values, 2nd ed. Cambridge, MA: CABI Publishing.

Quaas, R. L. 1988. Additive Genetic Model with Groups and Relationships. J. Dairy Sci., 71:1338-1345. <doi:10.3168.jds.S0022-0302(88)79691-5>

buildA	<i>Relationship matrix A</i>
--------	------------------------------

Description

Builds the pedigree-based additive genetic relationship matrix.

Usage

```
buildA(ped)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

Value

Relationship matrix A

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
buildA(ped)
```

buildD	<i>Relationship matrix D</i>
--------	------------------------------

Description

Builds the pedigree-based dominance relationship matrix.

Usage

```
buildD(ped, A)
```

Arguments

`ped` : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

`A` : Relationship matrix **A** created by function `buildA`.

Value

Relationship matrix **D**

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
buildD(ped, buildA(ped))
```

gghead

Append genetic groups to the pedigree

Description

This function appends parents that are not available in the first column of the pedigree, to the head of the pedigree, and sorts it. Given a pedigree with all missing parents replaced with the corresponding genetic groups, this functions appends genetic groups to the head of the pedigree.

Usage

```
gghead(ped)
```

Arguments

`ped` : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

Details

Consider this simple pedigree:

```
3 0 0
```

```
4 3 0
```

```
6 4 5
```

```
5 0 0
```

First, unknown parents are replaced with the corresponding genetic groups.

Please note that unknown parent IDs should be smaller than progeny IDs.

```
3 1 2
```

```
4 3 2
```

```
6 4 5
```

```
5 1 2
```

Then, gghead is applied to this pedigree (see the example).

Value

Processed pedigree data . frame

Examples

```
ped = data.frame(ID=c(3,4,6,5), SIRE=c(1,3,4,1), DAM=c(2,2,5,2))
gghead(ped)
```

inb	<i>Inbreeding coefficient</i>
-----	-------------------------------

Description

Calculates inbreeding coefficient for an individual.

Usage

```
inb(ped, id)
```

Arguments

ped : data . frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

id : Numeric ID of an individual

Value

Inb : Inbreeding coefficient of the individual

Examples

```
ped = data.frame(ID=1:7, SIRE=c(0,0,1,1,3,1,5), DAM=c(0,0,0,2,4,4,6))
inb(ped, 7)
```

mat2tab	<i>Matrix to tabular</i>
---------	--------------------------

Description

Converts matrix data to tabular data.

Usage

```
mat2tab(mat)
```

Arguments

mat : matrix

Value

tab : data.frame with 2 integer (IDs) and 1 numeric (values) columns.

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
mat2tab(buildA(ped))
```

offspring	<i>Descendants of an individual per generation</i>
-----------	--

Description

Counts and collects progeny and phenotyped progeny of an individual in successive generations.

Usage

```
offspring(ped, id, pheno)
```

Arguments

ped : data.frame with columns corresponding to ID, SIRE, DAM. Missing value is 0.
id : The ID of the individual, for which the descendants to be extracted.
pheno : Vector of phenotyped individuals.

Value

prgn : list of progeny per generation.
prgn.ph : list of phenotyped progeny per generation.

Examples

```

ped = data.frame(V1 = 1:19,
  V2 = c(0,0,1,1,0,0,0,0,0,4,5,5,7,0,0,9,0,0,12),
  V3 = c(0,0,0,2,0,2,0,3,3,3,0,6,8,8,8,10,11,11,0))
pheno = 10:18
# Find progeny and phenotyped progeny of individual 1.
offspring(ped, 1, pheno)
# Find phenotyped progeny of individual 1, in the 2nd generation.
offspring(ped, 1, 10:18)$prgn.ph[[2]]
# If only interested in finding the progeny of individual 1:
offspring(ped, 1, c())$prgn

```

pedcheck

Basic pedigree checks

Description

Performs basic pedigree checks.

Usage

```
pedcheck(ped)
```

Arguments

`ped` : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

Examples

```

set.seed(127)
ped = data.frame(ID=c(1:50,NA,0,1:3),
  SIRE=c(0, sample(c(0,10:25), 53, replace=TRUE), 51),
  DAM=c(0, NA, 52, sample(c(0,20:35), 52, replace=TRUE)))
pedcheck(ped)

```

peddown	<i>Downward pedigree extraction</i>
---------	-------------------------------------

Description

Extracts pedigree downward for one or a group of individuals to find their descendants

Usage

```
peddown(ped, parents, maxgen = c())
```

Arguments

`ped` : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

`parents` : Vector of individual ID(s), from which the new pedigree is being extracted.

`maxgen` : (optional) a positive integer for the maximum number of generations to proceed. If no value is provided, there is no limitation on the maximum number of generations to proceed.

Value

`newped` : Extracted pedigree data.frame

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
peddown(ped, c(1,4))
peddown(ped, 1, maxgen=1)
```

pedup	<i>Upward pedigree extraction</i>
-------	-----------------------------------

Description

Extracts pedigree upward for one or a group of individuals to find their ascendants

Usage

```
pedup(ped, progeny, maxgen = c())
```


Arguments

- ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.
- progeny : Vector of individual ID(s), from which the new pedigree is being extracted.
- maxgen : (optional) a positive integer for the maximum number of generations (continuing from parents of progeny) to proceed. If no value is provided, there is no limitation on the maximum number of generations to proceed.

Value

newped : Extracted pedigree data.frame

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
pedup(ped, c(1,4))
pedup(ped, 6, maxgen=1)
```

pruneped

Pedigree pruning

Description

Pruning pedigree in two different modes (strict, loose)

Usage

```
pruneped(ped, pheno, mode)
```

Arguments

- ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.
- pheno : Vector of phenotyped individuals
- mode : strict or loose

Details

In strict pruning, individuals without progeny and phenotype are recursively deleted from the pedigree, and then individuals without known parent and without progeny (if any) are deleted. Therefore, all uninfluential individuals are deleted. The downside is that individuals without phenotype or phenotyped progeny cannot receive any genetic merit based on the information from their phenotyped relatives. In loose pruning, the pedigree is upward extracted for phenotyped individuals to their founders, and then the pedigree is downward extracted from the founders.

Value

newped : Pruned pedigree data.frame

Examples

```
ped = data.frame(ID=1:7, SIRE=c(0,0,1,3,1,4,0), DAM=c(0,0,2,2,2,5,0))
pheno = c(1,4)
pruneped(ped, pheno, mode="strict")
pruneped(ped, pheno, mode="loose")
```

Qgpu

Vector $\mathbf{Qg} + \mathbf{u}$

Description

Adds genetic group contributions to the genetic merit of individuals.

Usage

Qgpu(Q, sol)

Arguments

Q : The output matrix from qmatL (for more details: ?qmatL)

sol : data.frame with 2 numeric columns corresponding to ID, EBV ($[\hat{\mathbf{g}}, \hat{\mathbf{u}}]$), where $\hat{\mathbf{g}}$ and $\hat{\mathbf{u}}$ are the genetic group and genetic merit solutions, respectively. The order of solutions must be the order of columns and the order of rows in matrix **Q**.

Value

Vector of $\mathbf{Qg} + \hat{\mathbf{u}}$

Examples

```
ped = data.frame(ID=c(3,4,6,5), SIRE=c(1,3,4,1), DAM=c(2,2,5,2))
Q = qmatL(gghead(ped))
ghat = c(0.1, -0.2)
uhat = seq(-1.5, 1.5, 1)
sol = data.frame(ID=1:6, EBV=c(ghat, uhat))
Qgpu(Q, sol)
```

qmat	<i>Matrix Q</i>
------	-----------------

Description

Creates the genetic group contribution matrix.

Usage

```
qmat(ped2)
```

Arguments

ped2 : The output data.frame from gghead (for more details: ?gghead)

Value

Q matrix

Examples

```
ped = data.frame(ID=c(3,4,6,5), SIRE=c(1,3,4,1), DAM=c(2,2,5,2))
ped2 = gghead(ped)
qmat(ped2)
```

qmatL	<i>Matrix Q for large pedigrees</i>
-------	-------------------------------------

Description

Creates the genetic group contribution matrix for large pedigrees.

Usage

```
qmatL(ped2)
```

Arguments

ped2 : The output data.frame from gghead (for more details: ?gghead)

Value

Q matrix

Examples

```
ped = data.frame(ID=c(3,4,6,5), SIRE=c(1,3,4,1), DAM=c(2,2,5,2))
ped2 = gghead(ped)
qmatL(ped2)
```

qmatXL

*Matrix **Q** for large pedigrees (parallel processing)*

Description

Creates the genetic group contribution matrix for large pedigrees, with parallel processing.

Usage

```
qmatXL(ped2, ncl)
```

Arguments

ped2 : The output data.frame from gghead (for more details: ?gghead)

ncl : User defined number of nodes; if the number of user defined nodes is greater than the number of genetic groups, the number genetic groups is considered as the number of nodes.

Details

This function is the parallel version of qmatL. It requires foreach and doParallel packages.

Value

Q matrix

Examples

```
ped = data.frame(ID=c(3,4,6,5), SIRE=c(1,3,4,1), DAM=c(2,2,5,2))
ped2 = gghead(ped)
qmatXL(ped2, 2)
```

renum	<i>Pedigree renumbering</i>
-------	-----------------------------

Description

Renumbering pedigree to numerical IDs, so that progeny's ID is smaller than parents' IDs.

Usage

```
renum(ped)
```

Arguments

ped : data.frame with columns corresponding to ID, SIRE, DAM. Missing value is 0.

Value

newped : Pedigree data.frame with renumbered IDs.

xrf : Cross-reference data.frame with 2 columns for original and renumbered IDs.

Examples

```
ped = data.frame(ID=letters[1:6], SIRE=c(0,0,letters[c(1,3,1,4)]), DAM=c(0,0,letters[c(2,2,2,5)]))
renum(ped)$newped
renum(ped)$xrf
```

rg	<i>Genetic relationship coefficient</i>
----	---

Description

Calculates genetic relationship coefficient between two individuals.

Usage

```
rg(ped, id1, id2)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

id1 : Numeric ID of an individual

id2 : Numeric ID of an individual

Value

rG : Genetic relationship coefficient between the two individuals

Examples

```
ped = data.frame(ID=1:7, SIRE=c(0,0,1,1,3,1,5), DAM=c(0,0,0,2,4,4,6))
rg(ped, 5, 6)
```

smgsped

Sire-maternal grandsire (S-MGS) pedigree

Description

Extract sire-maternal grandsire (S-MGS) pedigree from a sire-dam pedigree. Sire and MGS information is extracted for sires of phenotyped individuals.

Usage

```
smgsped(ped, pheno)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

pheno : Vector of phenotyped individuals

Value

newped : S-MGS pedigree data.frame

Examples

```
ped = data.frame(ID=1:10, SIRE=c(0,0,1,2,0,5,4,4,0,8), DAM=c(0,0,0,3,3,0,6,6,6,0))
smgsped(ped, 7:10)
```

tab2mat	<i>Tabular to matrix</i>
---------	--------------------------

Description

Converts tabular data to matrix data.

Usage

```
tab2mat(tab)
```

Arguments

tab : data.frame with 2 integer (IDs) and 1 numeric (values) columns.

Value

mat: matrix

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
tab2mat(tabA(ped))
```

tabA	<i>Relationship matrix \mathbf{A} in a tabular format</i>
------	--

Description

Creates the pedigree-based additive genetic relationship data.frame.

Usage

```
tabA(ped)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

Value

Genetic relationship data.frame

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
tabA(ped)
```

tabAinv

*Inverse of the relationship matrix **A** in a tabular format*

Description

Creates the data.frame of the inverse of the pedigree-based genetic relationship matrix.

Usage

```
tabAinv(ped, inbr)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.

inbr : Vector of inbreeding coefficients in the order of individuals in the relationship matrix.

Value

data.frame of the inverse of the genetic relationship matrix

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
inbr = c(0, 0, 0, 0.25, 0, 0.25)
# or
(inbr = diag(buildA(ped)) - 1)
# or
inbr = tabA(ped); (inbr[inbr[,1]==inbr[,2],]$a - 1)
# or
# For individual inbreeding values, use function inb.
tabAinv(ped, inbr)
```

tabD	<i>Dominance relationship matrix D in a tabular-sparse format</i>
------	--

Description

Creates the pedigree-based dominance relationship data.frame.

Usage

```
tabD(ped, A)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.
A : Relationship matrix **A** in a tabular format created by function tabA.

Value

Dominance relationship data.frame

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))  
tabD(ped, tabA(ped))
```

tabDinv	<i>Inverse of the dominance relationship matrix D in a tabular format</i>
---------	--

Description

Creates the data.frame of the inverse of the pedigree-based dominance relationship matrix.

Usage

```
tabDinv(ped, A)
```

Arguments

ped : data.frame with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.
A : Relationship matrix **A** in a tabular format created by function tabA.

Value

data.frame of the inverse of the dominance relationship matrix

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
tabDinv(ped, tabA(ped))
```

Index

buildA, 3

buildD, 3

gghead, 4

gggroups-package, 2

inb, 5

mat2tab, 6

offspring, 6

pedcheck, 7

peddown, 8

pedup, 8

prunedped, 9

Qgpu, 10

qmat, 11

qmatL, 11

qmatXL, 12

renum, 13

rg, 13

smgsped, 14

tab2mat, 15

tabA, 15

tabAinv, 16

tabD, 17

tabDinv, 17