Package 'ggroups'

February 25, 2019

Title Pedigree and Genetic Groups

Version 1.0.2

Date 2019-02-25

Description Pedigree-related functions including genetic groups.

License GPL-3

LazyData true

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

RoxygenNote 6.1.1

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

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

Encoding UTF-8

Repository GitHub

NeedsCompilation no

Author Mohammad Ali Nilforooshan [aut, cre] (0000-0003-0339-5442)

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

R topics documented:

ggroups-package *Pedigree and genetic groups*

Description

This package contains pedigree processing and analyzing functions, including functions for checking and renumbering the pedigree, making the pedigree relationship matrix and its inverse, in matrix and tubular 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 (**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 **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

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 *Relationship matrix* **A**

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)
```

---

gghead *Append genetic groups to the pedigree*

Description

Appends genetic groups to the head of the pedigree and sorts it.

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 animal 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)
```

---

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)
```

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

Description

Adds genetic group contributions to the genetic merit of animals.

Usage

```
Qgpu(Q, sol)
```

Arguments

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

Value

Vector of $\mathbf{Q}\hat{\mathbf{g}} + \hat{\mathbf{u}}$

Examples

```
ped = data.frame(ID=c(3,4,6,5), SIRE=c(1,3,4,1), DAM=c(2,2,5,2))
Q = qmat(gghead(ped))
ghat = c(0.1, -0.1)
uhat = seq(-0.15, 0.15, 0.1)
sol = data.frame(ID=1:6, EBV=c(ghat, uhat))
Qgpu(Q, sol)
```

---

qmat *Matrix* **Q**

Description

Creates the genetic group contribution matrix.

Usage

```
qmat(ped2)
```

Arguments

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

Value

Matrix **Q**

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 *Matrix* **Q** *for large pedigrees*

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)

Details

Calculation of the genetic group contribution matrix for large pedigrees requires a lot of memory and time. This might not be possible on ordinary computers, using the function qmat. The function qmatL takes less RAM and time, making the calculation possible for ordinary computers.

Value

Matrix **Q**

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 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

Matrix **Q**

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 *Pedigree renumbering*

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 renumberred IDs.

Cross-reference data.frame with 2 columns for original and renumberred 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)
```

tub2mat *Tubular to matrix*

Description

Converts tubular data to matrix data.

Usage

```
tub2mat(tub)
```

Arguments

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

Value

matrix

Examples

```
ped = data.frame(ID=1:6, SIRE=c(0,0,1,3,1,4), DAM=c(0,0,2,2,2,5))
tub2mat(tubA(ped))
```

tubA *Relationship matrix* **A** *in a tubular format*

Description

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

Usage

```
tubA(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))
tubA(ped)
```

---

tubAinv *Inverse of the relationship matrix* **A** *in a tubular format*

Description

Creates the inverse of the pedigree-based additive genetic relationship matrix in a `data.frame`.

Usage

```
tubAinv(ped, inb)
```

Arguments

`ped`	: `data.frame` with integer columns corresponding to ID, SIRE, DAM. Missing value is 0.
`inb`	: Inbreeding coefficients in the order of animals in the relationship matrix. It can be derived from `buildA` or `tubularA`.

Details

`inb = diag(buildA) - 1`, or

`inb = tubA(ped); inb = inb[inb[,1]==inb[,2],]$a - 1`

Value

Inverse of the 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))
inb = c(0, 0, 0, 0.25, 0, 0.25)
tubAinv(ped, inb)
```

Index