

# Package ‘pedigree’

June 24, 2022

**Type** Package  
**Title** Pedigree Functions  
**Version** 1.4.1  
**Date** 2012-11-27  
**Depends** Matrix, HaploSim (>= 1.8.4), reshape  
**Imports** methods  
**Description** Pedigree related functions.  
**License** GPL (>= 2)  
**Author** Albart Coster [aut, cre]  
**Maintainer** Albart Coster <albart@dairyconsult.nl>  
**Repository** CRAN  
**Date/Publication** 2022-06-24 11:41:48 UTC  
**NeedsCompilation** yes

## R topics documented:

pedigree-package . . . . .	2
add.Inds . . . . .	2
blup . . . . .	3
calcG . . . . .	4
calcInbreeding . . . . .	4
countGen . . . . .	5
countOff . . . . .	6
gblup . . . . .	6
makeA . . . . .	7
makeAinv . . . . .	8
orderPed . . . . .	9
trimPed . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

pedigree-package      *Package to deal with pedigree data*

---

### Description

Package with functions to analyse and transform pedigree data. A pedigree is a `data.frame` where the first column contains an ID, and the second and third columns contain ID of first and second parent.

### Author(s)

Albart Coster: <albart.coster@wur.nl>

### See Also

[trimPed](#) [orderPed](#) [countGen](#) [makeA](#) [makeAinv](#) [calcInbreeding](#) [add.Inds](#)

---

add.Inds      *Function to add missing individuals to a pedigree*

---

### Description

Function `add.Inds()` adds missing individuals to a pedigree and returns the complete pedigree as a `data.frame` with the same headers as the original pedigree. Remember to check for errors beforehand with function `errors.ped`. Unknown parents should be coded as NA.

### Usage

```
add.Inds(ped)
```

### Arguments

ped      `data.frame` with three columns: id,id parent1,id parent2

### Value

`data.frame` of three columns with identical header as input.

### Author(s)

Albart Coster, [Albart.Coster@wur.nl](mailto:Albart.Coster@wur.nl)

### See Also

[orderPed](#)

**Examples**

```
ID <- 3:5
DAM <- c(1,1,3)
SIRE <- c(2,2,4)
pedigree <- data.frame(ID,DAM,SIRE)
pedigree <- add.Inds(pedigree)
```

---

**blup***Function to calculate breeding values using an animal model*

---

**Description**

Fit an animal model to data, use a given variance ratio ( $\alpha = \frac{\sigma_e^2}{\sigma_a^2}$ ). Calculate inverse of the additive genetic relationship matrix using function `makeInv()` of this package.

**Usage**

```
blup(formula, ped, alpha, trim = FALSE)
```

**Arguments**

formula	formula of the model, do not include the random effect due to animal (generally ID).
ped	data.frame with columns corresponding to ID, SIRE, DAM and the columns in the formula.
alpha	Variance ratio ( $\frac{\sigma_e^2}{\sigma_a^2}$ ).
trim	If TRUE, trims the pedigree using the available phenotype data using function <a href="#">trimPed</a> .

**Value**

Vector of solutions to the model, including random animal effects.

**See Also**

[SamplePedigree](#), [gblup](#), [makeAinv](#), [blup](#)

**Examples**

```
example(gblup)
sol <- blup(P~1, ped = ped, alpha = 1/h2 - 1)
```

---

calcG	<i>Function to calculate a relationship matrix from marker data (usually allele count data), G matrix.</i>
-------	------------------------------------------------------------------------------------------------------------

---

**Description**

Function to calculate a relationship matrix from marker data. Option to return the inverse of matrix. Inverse calculated using Matrix package.

**Usage**

```
calcG(M, data = NULL, solve = FALSE)
```

**Arguments**

M	Matrix of marker genotypes, usually the count of one of the two SNP alleles at each markers (0, 1, or 2).
data	Optional logical vector which can tell of which individuals we have phenotypes.
solve	Logic, if TRUE then function returns the inverse of the relationship matrix.

**Value**

Matrix of class dgeMatrix.

**See Also**

[SamplePedigree](#), [gblup](#), [makeAinv](#), [blup](#)

**Examples**

```
example(gblup)
G <- calcG(M)
Ginv <- calcG(M, solve = TRUE)
```

---

calcInbreeding	<i>Calculates inbreeding coefficients for individuals in a pedigree.</i>
----------------	--------------------------------------------------------------------------

---

**Description**

Calculates inbreeding coefficients of individuals in a pedigree.

**Usage**

```
calcInbreeding(ped)
```

**Arguments**

ped                    data.frame with three columns: id,id parent1,id parent2

**Value**

Logical.

**Examples**

```
id <- 1:6
dam <- c(0,0,1,1,4,4)
sire <- c(0,0,2,2,3,5)
ped <- data.frame(id,dam,sire)
(F <- calcInbreeding(ped))
```

---

countGen	<i>Count generation number for each individual in a pedigree.</i>
----------	-------------------------------------------------------------------

---

**Description**

Counts generation number for individuals in a pedigree.

**Usage**

```
countGen(ped)
```

**Arguments**

ped                    data.frame with three columns: id,id parent1,id parent2

**Value**

Numeric vector

**Examples**

```
id <- 1:5
dam <- c(0,0,1,1,4)
sire <- c(0,0,2,2,3)
ped <- data.frame(id,dam,sire)
(gens <- countGen(ped))
```

---

countOff	<i>Function that counts the number of offspring (and following generations for each individual in a pedigree.</i>
----------	-------------------------------------------------------------------------------------------------------------------

---

**Description**

Function to count the number of offspring for each individual in a pedigree. With loops, offspring of later generations will be counted several times.

**Usage**

```
countOff(ped)
```

**Arguments**

ped	data.frame with three columns: id,id parent1,id parent2
-----	---------------------------------------------------------

**Value**

Numeric vector with number of offspring for each individual in the pedigree.

**Author(s)**

Albart Coster

**Examples**

```
example(countGen)
countOff(ped)
```

---

gblup	<i>Function to calculate breeding values using an animal model and a relationship matrix calculated from the markers (G matrix)</i>
-------	-------------------------------------------------------------------------------------------------------------------------------------

---

**Description**

Fit an animal model to data, use a given variance ratio ( $\alpha = \frac{\sigma_a^2}{\sigma_a^2 + \sigma_e^2}$ ). Calculate genetic relationship matrix using the function calcG of this package.

**Usage**

```
gblup(formula, data, M, lambda)
```

**Arguments**

formula	formula of the model, do not include the random effect due to animal (generally ID).
data	data.frame with columns corresponding to ID and the columns mentioned in the formula.
M	Matrix of marker genotypes, usually the count of one of the two SNP alleles at each markers (0, 1, or 2).
lambda	Variance ratio ( $\frac{\sigma_g^2}{\sigma_a^2}$ )

**Value**

Vector of solutions to the model, including random animal effects.

**See Also**

[SamplePedigree](#), [gblup](#), [makeAinv](#), [blup](#)

**Examples**

```
h2 <- 0.5
example(SamplePedigree)
ped <- phList$ped
hList <- phList$hList
qtlList <- ListQTL(hList = hList, frqtl = 0.1, sigma2qtl = 1)
qtl <- tapply(unlist(qtlList), list(rep(names(qtlList), times = unlist(lapply(qtlList, length))),
                                unlist(lapply(qtlList, function(x) seq(1, length(x)))))), mean, na.rm = TRUE)
qtl <- melt(qtl)
names(qtl) <- c("POS", "TRAIT", "a")
HH <- getAll(hList, translatePos = FALSE)
rownames(HH) <- sapply(hList, function(x) x@hID)
QQ <- HH[, match(qtl$POS, colnames(HH))]
g <- QQ
ped$G <- with(ped, g[match(hID0, rownames(g))] + g[match(hID1, rownames(g))])
sigmae <- sqrt(var(ped$G)/h2 - var(ped$G))
ped$P <- ped$G + rnorm(nrow(ped), 0, sigmae)
M <- with(ped, HH[match(hID0, rownames(HH)), ] + HH[match(hID1, rownames(HH)), ])
rownames(M) <- ped$hID
sol <- gblup(P~1, data = ped[, c('ID', 'P')], M = M, lambda = 1/h2 - 1)
```

---

makeA

*Makes the A matrix for a part of a pedigree*

---

**Description**

Makes the A matrix for a part of a pedigree and stores it in a file called A.txt.

**Usage**

```
makeA(ped,which)
```

**Arguments**

ped	data.frame with three columns: id,id parent1,id parent2
which	Logical vector specifying between which individuals additive genetic relationship is required. Goes back through the whole pedigree but only for subset of individuals.

**Value**

Logical.

**Examples**

```
id <- 1:6
dam <- c(0,0,1,1,4,4)
sire <- c(0,0,2,2,3,5)
ped <- data.frame(id,dam,sire)
makeA(ped,which = c(rep(FALSE,4),rep(TRUE,2)))
A <- read.table("A.txt")
```

---

makeAinv

*Makes inverted A matrix for a pedigree*

---

**Description**

Makes inverted A matrix for a pedigree and stores it in a file called Ainv.txt.

**Usage**

```
makeAinv(ped)
```

**Arguments**

ped	data.frame with three columns: id,id parent1,id parent2
-----	---------------------------------------------------------

**Value**

Logical.

**Examples**

```

id <- 1:6
dam <- c(0,0,1,1,4,4)
sire <- c(0,0,2,2,3,5)
ped <- data.frame(id,dam,sire)
makeAinv(ped)
Ai <- read.table("Ainv.txt")
nInd <- nrow(ped)
Ainv <- matrix(0,nrow = nInd,ncol = nInd)
Ainv[as.matrix(Ai[,1:2])] <- Ai[,3]
dd <- diag(Ainv)
Ainv <- Ainv + t(Ainv)
diag(Ainv) <- dd

```

---

orderPed

*Orders a pedigree*


---

**Description**

Orders a pedigree so that offspring follow parents.

**Usage**

```
orderPed(ped)
```

**Arguments**

ped                    data.frame with three columns: id,id parent1,id parent2

**Value**

numerical vector

**Examples**

```

id <- 1:6
dam <- c(0,0,1,1,4,4)
sire <- c(0,0,2,2,3,5)
pedigree <- data.frame(id,dam,sire)
(ord <- orderPed(pedigree))
pedigree <- pedigree[6:1,]
(ord <- orderPed(pedigree))
pedigree <- pedigree[order(ord),]
pwrong <- pedigree
pwrong[1,2] <- pwrong[6,1]

```

---

trimPed	<i>Function to trim a pedigree based on available data</i>
---------	------------------------------------------------------------

---

**Description**

Trims a pedigree given a vector of data. Branches without data are trimmed off the pedigree.

**Usage**

```
trimPed(ped, data, ngenback = NULL)
```

**Arguments**

ped	data.frame with three columns: id, id parent1, id parent2
data	TRUE-FALSE vector. Specifies if data for an individual is available.
ngenback	Number of generations back. Specifies the number of generations to keep before the individuals with data.

**Value**

Logical vector specifying if an individual should stay in the pedigree.

**Examples**

```
id <- 1:5
dam <- c(0,0,1,1,4)
sire <- c(0,0,2,2,3)
data <- c(FALSE, FALSE, TRUE, FALSE, FALSE)
ped <- data.frame(id, dam, sire)
yn <- trimPed(ped, data)
ped <- ped[yn,]
```

# Index

## \* utilities

- add.Inds, 2
- blup, 3
- calcG, 4
- calcInbreeding, 4
- countGen, 5
- countOff, 6
- gblup, 6
- makeA, 7
- makeAinv, 8
- orderPed, 9
- pedigree-package, 2
- trimPed, 10

add.Inds, 2, 2

blup, 3, 3, 4, 7

calcG, 4

calcInbreeding, 2, 4

countGen, 2, 5

countOff, 6

gblup, 3, 4, 6, 7

makeA, 2, 7

makeAinv, 2–4, 7, 8

orderPed, 2, 9

pedigree (pedigree-package), 2

pedigree-package, 2

SamplePedigree, 3, 4, 7

trimPed, 2, 3, 10