Package 'BGmisc'

September 27, 2023

Title An R Package for Extended Behavior Genetics Analysis

Version 1.0.1

Date 2023-09-26

Description The BGmisc R package offers a comprehensive suite of functions tailored for extended behavior genetics analysis, including model identification, calculating relatedness, pedigree conversion, pedigree simulation, and more.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports Matrix, stats, kinship2, igraph

Suggests knitr, rmarkdown, EasyMx, OpenMx, dplyr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5.0)

NeedsCompilation no

Author S. Mason Garrison [aut, cre] (<https://orcid.org/0000-0002-4804-6003>),
Michael D. Hunter [aut] (<https://orcid.org/0000-0002-3651-6709>),
Xuanyu Lyu [aut] (<https://orcid.org/0000-0002-2841-5529>),
Jonathan D. Trattner [aut] (<https://orcid.org/0000-0002-1097-7603>),
S. Alexandra Burt [aut] (<https://orcid.org/0000-0001-5538-7431>)

Maintainer S. Mason Garrison <garrissm@wfu.edu>

Repository CRAN

Date/Publication 2023-09-26 22:40:04 UTC

R topics documented:

`allGens` *allGens A function to calculate the number of individuals in each generation. This is a supporting function for* `simulatePedigree`.

Description

allGens A function to calculate the number of individuals in each generation. This is a supporting function for `simulatePedigree`.

Usage

```
allGens(kpc, Ngen, marR)
```

Arguments

`kpc`	Number of kids per couple (integer >= 2).
`Ngen`	Number of generations (integer >= 1).
`marR`	Mating rate (numeric value ranging from 0 to 1).

Value

Returns a vector containing the number of individuals in every generation.

BGmisc_package *Behavior Genetic Miscellaneous functions in R*

Description

This collection contains functions for behavior genetic modeling. These functions include model identification, calculating relatedness, and various others (e.g. Hunter, Garrison, et al, 2021 <doi:10.1007/s10519-019-09973-8>). See the package vignette for details:

Author(s)

S. Mason Garrison, Michael D. Hunter, Xuanyu Lyu, Jonathan D. Trattner, and S. Alexandra Burt

calculateRelatedness *Calculate Relatedness Coefficient*

Description

The relatedness coefficient between two people (b & c) is defined in relation to their common ancestors: $r_{bc} = \sum \left(\frac{1}{2}\right)^{n+n'+1} (1 + f_a)$

Usage

```
calculateRelatedness(
  generations = 2,
  path = NULL,
  full = TRUE,
  maternal = FALSE,
  empirical = FALSE,
  segregating = TRUE,
  total_a = 6800 * 1e+06,
  total_m = 16500,
  weight_a = 1,
  weight_m = 1,
  denom_m = FALSE,
  ...
)
```

Arguments

generations	Number of generations back of common ancestors the pair share.
path	Traditional method to count common ancestry, which is twice the number of generations removed from common ancestors. If not provided, it is calculated as 2*generations.
full	Logical. Indicates if the kin share both parents at the common ancestor's generation. Default is TRUE.
maternal	Logical. Indicates if the maternal lineage should be considered in the calculation.
empirical	Logical. Adjusts the coefficient based on empirical data, using the total number of nucleotides and other parameters.
segregating	Logical. Adjusts for segregating genes.
total_a	Numeric. Represents the total size of the autosomal genome in terms of nucleotides, used in empirical adjustment. Default is 6800*1000000.
total_m	Numeric. Represents the total size of the mitochondrial genome in terms of nucleotides, used in empirical adjustment. Default is 16500.
weight_a	Numeric. Represents the weight of phenotypic influence from additive genetic variance, used in empirical adjustment.
weight_m	Numeric. Represents the weight of phenotypic influence from mitochondrial effects, used in empirical adjustment.
denom_m	Logical. Indicates if 'total_m' and 'weight_m' should be included in the denominator of the empirical adjustment calculation.
...	Further named arguments that may be passed to another function.

Details

This function calculates the relatedness coefficient between two individuals based on their shared ancestry, as described by Wright (1922).

Value

Relatedness Coefficient ('coef'): A measure of the genetic relationship between two individuals.

Examples

```
## Not run:
# For full siblings, the relatedness coefficient is expected to be 0.5:
calculateRelatedness(generations = 1, full = TRUE)
# For half siblings, the relatedness coefficient is expected to be 0.25:
calculateRelatedness(generations = 1, full = FALSE)

## End(Not run)
```

comp2vech *comp2vech Turn a variance component relatedness matrix into its half-vectorization*

Description

comp2vech Turn a variance component relatedness matrix into its half-vectorization

Usage

```
comp2vech(x, include.zeros = FALSE)
```

Arguments

x	Relatedness component matrix (can be a matrix, list, or object that inherits from 'Matrix').
include.zeros	logical. Whether to include all-zero rows. Default is FALSE.

Details

This function is a wrapper around the vech function, extending it to allow for blockwise matrices and specific classes. It facilitates the conversion of a variance component relatedness matrix into a half-vectorized form.

Value

The half-vectorization of the relatedness component matrix.

Examples

```
comp2vech(list(matrix(c(1, .5, .5, 1), 2, 2), matrix(1, 2, 2)))
```

evenInsert *evenInsert A function to insert m elements evenly into a length n vector.*

Description

evenInsert A function to insert m elements evenly into a length n vector.

Usage

```
evenInsert(m, n)
```

Arguments

m	A numeric vector of length less than or equal to n. The elements to be inserted.
n	A numeric vector. The vector into which the elements of m will be inserted.

Details

The function takes two vectors, m and n, and inserts the elements of m evenly into n. If the length of m is greater than the length of n, the vectors are swapped, and the insertion proceeds. The resulting vector is a combination of m and n, with the elements of m evenly distributed within n.

Value

Returns a numeric vector with the elements of m evenly inserted into n.

See Also

SimPed for the main function that uses this supporting function.

---

famSizeCal *famSizeCal A function to calculate the total number of individuals in a pedigree given parameters. This is a supporting function for function* simulatePedigree

---

Description

famSizeCal A function to calculate the total number of individuals in a pedigree given parameters. This is a supporting function for function simulatePedigree

Usage

```
famSizeCal(kpc, Ngen, marR)
```

Arguments

kpc	Number of kids per couple (integer >= 2).
Ngen	Number of generations (integer >= 1).
marR	Mating rate (numeric value ranging from 0 to 1).

Value

Returns a numeric value indicating the total pedigree size.

`fitComponentModel` *fitComponentModel Fit the estimated variance components of a model to covariance data*

Description

fitComponentModel Fit the estimated variance components of a model to covariance data

Usage

```
fitComponentModel(covmat, ...)
```

Arguments

`covmat`	The covariance matrix of the raw data, which may be blockwise.
`...`	Comma-separated relatedness component matrices representing the variance components of the model.

Details

This function fits the estimated variance components of a model to given covariance data. The rank of the component matrices is checked to ensure that the variance components are all identified. Warnings are issued if there are inconsistencies.

Value

A regression (linear model fitted with `lm`). The coefficients of the regression represent the estimated variance components.

Examples

```
## Not run:
# install.packages("OpenMX")
data(twinData, package = "OpenMx")
selVars <- c("ht1", "ht2")
mzData <- subset(twinData, zyg %in% c(1), c(selVars, "zyg"))
dzData <- subset(twinData, zyg %in% c(3), c(selVars, "zyg"))

fitComponentModel(
 covmat = list(cov(mzData[, selVars], use = "pair"), cov(dzData[, selVars], use = "pair")),
  A = list(matrix(1, nrow = 2, ncol = 2), matrix(c(1, 0.5, 0.5, 1), nrow = 2, ncol = 2)),
  C = list(matrix(1, nrow = 2, ncol = 2), matrix(1, nrow = 2, ncol = 2)),
  E = list(diag(1, nrow = 2), diag(1, nrow = 2))
)

## End(Not run)
```

`hazard` *Simulated pedigree with two extended families and an age-related hazard*

Description

A dataset simulated to have an age-related hazard. There are two extended families that are sampled from the same population.

Usage

```
data(hazard)
```

Format

A data frame with 43 rows and 14 variables

Details

The variables are as follows:

- FamID. ID of the extended family
- ID. Person identification variable
- sex. Sex of the ID: 1 is female; 0 is male
- dadID. ID of the father
- momID. ID of the mother
- affected. logical. Whether the person is affected or not
- DA1. Binary variable signifying the meaninglessness of life
- DA2. Binary variable signifying the fundamental unknowability of existence
- birthYr. Birth year for person
- onsetYr. Year of onset for person
- deathYr. Death year for person
- available. logical. Whether
- Gen. Generation of the person
- proband. logical. Whether the person is a proband or not

`identifyComponentModel` *identifyComponentModel Determine if a variance components model is identified*

Description

identifyComponentModel Determine if a variance components model is identified

Usage

```
identifyComponentModel(..., silent = FALSE)
```

Arguments

`...`	Comma-separated relatedness component matrices representing the variance components of the model.
`silent`	logical. If TRUE, suppresses messages about identification; FALSE by default.

Details

This function checks the identification status of a given variance components model by examining the rank of the concatenated matrices of the components. If any components are not identified, their names are returned in the output.

Value

A list of length 2 containing:

- `identified`: TRUE if the model is identified, FALSE otherwise.
- `nidp`: A vector of non-identified parameters, specifying the names of components that are not simultaneously identified.

Examples

```
identifyComponentModel(A = list(matrix(1, 2, 2)), C = list(matrix(1, 2, 2)), E = diag(1, 2))
```

inbreeding *Artificial pedigree data on seven families with inbreeding*

Description

A dataset created purely from imagination that includes several types of inbreeding. Different kinds of inbreeding occur in each extended family.

Usage

```
data(inbreeding)
```

Format

A data frame (and ped object) with 113 rows and 7 variables

Details

The variables are as follows:

- ID. Person identification variable
- sex. Sex of the ID: 1 is female; 0 is male
- dadID. ID of the father
- momID. ID of the mother
- FamID. ID of the extended family
- Gen. Generation of the person
- proband. Always FALSE

inferRelatedness *Infer Relatedness Coefficient*

Description

The function uses the ACE framework to infer the relatedness between two individuals.

Usage

```
inferRelatedness(cor_obs, ace_A = 0.9, ace_C = 0, shared_C = 0)
```

Arguments

cor_obs	Numeric. Observed correlation between the two groups. Must be between -1 and 1.
ace_A	Numeric. Proportion of variance attributable to additive genetic variance. Must be between 0 and 1. Default is 0.9.
ace_C	Numeric. Proportion of variance attributable to shared environmental variance. Must be between 0 and 1. Default is 0.
shared_C	Numeric. Proportion of shared environment shared between the two individuals. Must be between 0 and 1. Default is 0.

Details

Infers the relatedness coefficient between two groups based on the observed correlation between their additive genetic variance and shared environmental variance.

Value

Calculated relatedness coefficient ('est_r').

Examples

```
## Not run:
# Infer the relatedness coefficient:
inferRelatedness(cor_obs = 0.5, ace_A = 0.9, ace_C = 0, shared_C = 0)

## End(Not run)
```

---

ped2add *Take a pedigree and turn it into an additive genetics relatedness matrix*

Description

Take a pedigree and turn it into an additive genetics relatedness matrix

Usage

```
ped2add(
  ped,
  max.gen = Inf,
  sparse = FALSE,
  verbose = FALSE,
  gc = FALSE,
  flatten.diag = FALSE
)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns
max.gen	the maximum number of generations to compute (e.g., only up to 4th degree relatives). The default of Inf uses as many generations as there are in the data.
sparse	logical. If TRUE, use and return sparse matrices from Matrix package
verbose	logical If TRUE, print progress through stages of algorithm
gc	logical. If TRUE, do frequent garbage collection via gc to save memory
flatten.diag	Logical. The default is FALSE. If TRUE, overwrites the diagonal of the final relatedness matrix with ones.

Details

source examplePedigreeFunctions

---

ped2ce *Take a pedigree and turn it into an extended environmental relatedness matrix*

Description

Take a pedigree and turn it into an extended environmental relatedness matrix

Usage

```
ped2ce(ped)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns

Details

source examplePedigreeFunctions

ped2cn *Take a pedigree and turn it into a common nuclear environmental relatedness matrix*

Description

Take a pedigree and turn it into a common nuclear environmental relatedness matrix

Usage

```
ped2cn(
  ped,
  max.gen = Inf,
  sparse = FALSE,
  verbose = FALSE,
  gc = FALSE,
  flatten.diag = FALSE
)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns
max.gen	the maximum number of generations to compute (e.g., only up to 4th degree relatives). The default of Inf uses as many generations as there are in the data.
sparse	logical. If TRUE, use and return sparse matrices from Matrix package
verbose	logical If TRUE, print progress through stages of algorithm
gc	logical. If TRUE, do frequent garbage collection via gc to save memory
flatten.diag	Logical. The default is FALSE. If TRUE, overwrites the diagonal of the final relatedness matrix with ones.

Details

source examplePedigreeFunctions

ped2com *Take a pedigree and turn it into a relatedness matrix*

Description

Take a pedigree and turn it into a relatedness matrix

Usage

```
ped2com(
  ped,
  component,
  max.gen = Inf,
  sparse = FALSE,
  verbose = FALSE,
  gc = FALSE,
  flatten.diag = FALSE
)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns
component	character. Which component of the pedigree to return. See Details.
max.gen	the maximum number of generations to compute (e.g., only up to 4th degree relatives). The default of Inf uses as many generations as there are in the data.
sparse	logical. If TRUE, use and return sparse matrices from Matrix package
verbose	logical If TRUE, print progress through stages of algorithm
gc	logical. If TRUE, do frequent garbage collection via gc to save memory
flatten.diag	Logical. The default is FALSE. If TRUE, overwrites the diagonal of the final relatedness matrix with ones.

Details

source examplePedigreeFunctions

---

ped2fam *Add an extended family ID variable to a pedigree*

Description

Add an extended family ID variable to a pedigree

Usage

```
ped2fam(
  ped,
  personID = "ID",
  momID = "momID",
  dadID = "dadID",
  famID = "famID"
)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns
personID	character. Name of the column in ped for the person ID variable
momID	character. Name of the column in ped for the mother ID variable
dadID	character. Name of the column in ped for the father ID variable
famID	character. Name of the column to be created in ped for the family ID variable

Details

The general idea of this function is to use person ID, mother ID, and father ID to create an extended family ID such that everyone with the same family ID is in the same (perhaps very extended) pedigree. That is, a pair of people with the same family ID have at least one traceable relation of any length to one another.

This function works by turning the pedigree into a mathematical graph using the igraph package. Once in graph form, the function uses weakly connected components to search for all possible relationship paths that could connect anyone in the data to anyone else in the data.

Value

A pedigree dataset with one additional column for the newly created extended family ID

---

ped2graph *Turn a pedigree into a graph*

Description

Turn a pedigree into a graph

Usage

```
ped2graph(
  ped,
  personID = "ID",
  momID = "momID",
  dadID = "dadID",
  famID = "famID",
  directed = TRUE
)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns
personID	character. Name of the column in ped for the person ID variable
momID	character. Name of the column in ped for the mother ID variable
dadID	character. Name of the column in ped for the father ID variable

famID	character. Name of the column to be created in ped for the family ID variable
directed	Logical scalar. Default is TRUE. Indicates whether or not to create a directed graph.

Details

The general idea of this function is to represent a pedigree as a graph using the igraph package.

Once in graph form, several common pedigree tasks become much simpler.

Value

A graph

`ped2mit` *Take a pedigree and turn it into a mitochondrial relatedness matrix*

Description

Take a pedigree and turn it into a mitochondrial relatedness matrix

Usage

```
ped2mit(
  ped,
  max.gen = Inf,
  sparse = FALSE,
  verbose = FALSE,
  gc = FALSE,
  flatten.diag = FALSE
)
```

Arguments

ped	a pedigree dataset. Needs ID, momID, and dadID columns
max.gen	the maximum number of generations to compute (e.g., only up to 4th degree relatives). The default of Inf uses as many generations as there are in the data.
sparse	logical. If TRUE, use and return sparse matrices from Matrix package
verbose	logical If TRUE, print progress through stages of algorithm
gc	logical. If TRUE, do frequent garbage collection via gc to save memory
flatten.diag	Logical. The default is FALSE. If TRUE, overwrites the diagonal of the final relatedness matrix with ones.

Details

source examplePedigreeFunctions

plotPedigree	*plotPedigree A wrapped function to plot simulated pedigree from function* simulatePedigree. *This function require the installation of package* kinship2.

Description

plotPedigree A wrapped function to plot simulated pedigree from function simulatePedigree. This function require the installation of package kinship2.

Usage

```
plotPedigree(
  ped,
  code_male = NULL,
  quiet = TRUE,
  cex = 0.5,
  col = 1,
  symbolsize = 1,
  branch = 0.6,
  packed = TRUE,
  align = c(1.5, 2),
  width = 8,
  density = c(-1, 35, 65, 20),
  mar = c(2.1, 1, 2.1, 1),
  angle = c(90, 65, 40, 0),
  keep.par = FALSE,
  pconnect = 0.5,
  ...
)
```

Arguments

ped	The simulated pedigree data.frame from function simulatePedigree. Or a pedigree dataframe with the same colnames as the dataframe simulated from function simulatePedigree.
code_male	This optional input allows you to indicate what value in the sex variable codes for male.
quiet	Logical. If TRUE, suppresses the printing of the pedigree object. Default is TRUE.
cex	The font size of the IDs for each individual in the plot.
col	color for each id. Default assigns the same color to everyone.
symbolsize	controls symbolsize. Default=1.
branch	defines how much angle is used to connect various levels of nuclear families.
packed	default=T. If T, uniform distance between all individuals at a given level.

`align`	these parameters control the extra effort spent trying to align children underneath parents, but without making the pedigree too wide. Set to F to speed up plotting.
`width`	default=8. For a packed pedigree, the minimum width allowed in the realignment of pedigrees.
`density`	defines density used in the symbols. Takes up to 4 different values.
`mar`	margin parmeters, as in the `par` function
`angle`	defines angle used in the symbols. Takes up to 4 different values.
`keep.par`	Default = F, allows user to keep the parameter settings the same as they were for plotting (useful for adding extras to the plot)
`pconnect`	when connecting parent to children the program will try to make the connecting line as close to vertical as possible, subject to it lying inside the endpoints of the line that connects the children by at least `pconnect` people. Setting this option to a large number will force the line to connect at the midpoint of the children.
`...`	Extra options that feed into the plot function.

Value

A plot of the simulated pedigree

`relatedness` *relatedness (Deprecated)*

Description

When calling this function, a warning will be issued about its deprecation.

Usage

```
relatedness(...)
```

Arguments

`...`	Arguments to be passed to ‘inferRelatedness‘.

Details

This function is a wrapper around the new ‘inferRelatedness‘ function. ‘relatedness‘ has been deprecated, and it’s advised to use ‘inferRelatedness‘ directly.

Value

The same result as calling ‘inferRelatedness‘.

See Also

`inferRelatedness` for the updated function.

Examples

```
## Not run:
# This is an example of the deprecated function:
relatedness(...)
# It is recommended to use:
inferRelatedness(...)

## End(Not run)
```

`related_coef` *related_coef (Deprecated)*

Description

When calling this function, a warning will be issued about its deprecation.

Usage

```
related_coef(...)
```

Arguments

`...`	Arguments to be passed to ‘calculateRelatedness‘.

Details

This function is a wrapper around the new ‘calculateRelatedness‘ function. ‘related_coef‘ has been deprecated, and it’s advised to use ‘calculateRelatedness‘ directly.

Value

The same result as calling ‘calculateRelatedness‘.

See Also

`calculateRelatedness` for the updated function.

Examples

```
## Not run:
# This is an example of the deprecated function:
related_coef(...)
# It is recommended to use:
calculateRelatedness(...)

## End(Not run)
```

SimPed *SimPed (Deprecated)*

Description

When calling this function, a warning will be issued about its deprecation.

Usage

```
SimPed(...)
```

Arguments

...	Arguments to be passed to ‘simulatePedigree‘.

Details

This function is a wrapper around the new ‘simulatePedigree‘ function. ‘SimPed‘ has been deprecated, and it’s advised to use ‘simulatePedigree‘ directly.

Value

The same result as calling ‘simulatePedigree‘.

See Also

simulatePedigree for the updated function.

Examples

```
## Not run:
# This is an example of the deprecated function:
SimPed(...)
# It is recommended to use:
simulatePedigree(...)

## End(Not run)
```

`simulatePedigree` *Simulate Pedigrees*

Description

This function simulates "balanced" pedigrees based on a group of parameters: 1) k - Kids per couple; 2) G - Number of generations; 3) p - Proportion of males in offspring; 4) r - Mating rate.

Usage

```
simulatePedigree(
  kpc = 3,
  Ngen = 4,
  sexR = 0.5,
  marR = 2/3,
  balancedSex = TRUE,
  balancedmar = TRUE
)
```

Arguments

`kpc`	Number of kids per couple. An integer >= 2 that determines how many kids each fertilized mated couple will have in the pedigree. Default value is 3. Returns an error when kpc equals 1.
`Ngen`	Number of generations. An integer >= 2 that determines how many generations the simulated pedigree will have. The first generation is always a fertilized couple. The last generation has no mated individuals.
`sexR`	Sex ratio of offspring. A numeric value ranging from 0 to 1 that determines the proportion of males in all offspring in this pedigree. For instance, 0.4 means 40 percent of the offspring will be male.
`marR`	Mating rate. A numeric value ranging from 0 to 1 which determines the proportion of mated (fertilized) couples in the pedigree within each generation. For instance, marR = 0.5 suggests 50 percent of the offspring in a specific generation will be mated and have their offspring.
`balancedSex`	Not fully developed yet. Always `TRUE` in the current version.
`balancedmar`	Not fully developed yet. Always `TRUE` in the current version.

Value

A `data.frame` with each row representing a simulated individual. The columns are as follows:

- fam: The family id of each simulated individual. It is 'fam1' in a single simulated pedigree.
- ID: The unique personal ID of each simulated individual. The first digit is the fam id; the fourth digit is the generation the individual is in; the following digits represent the order of the individual within his/her pedigree. For example, 100411 suggests this individual has a family id of 1, is in the 4th generation, and is the 11th individual in the 4th generation.

- gen: The generation the simulated individual is in.
- dadID: Personal ID of the individual's father.
- momID: Personal ID of the individual's mother.
- spt: Personal ID of the individual's mate.
- sex: Biological sex of the individual. F - female; M - male.

`sizeAllGens` *sizeAllGens An internal supporting function for* `simulatePedigree`.

Description

sizeAllGens An internal supporting function for `simulatePedigree`.

Usage

```
sizeAllGens(kpc, Ngen, marR)
```

Arguments

`kpc`	Number of kids per couple (integer >= 2).
`Ngen`	Number of generations (integer >= 1).
`marR`	Mating rate (numeric value ranging from 0 to 1).

Value

Returns a vector including the number of individuals in every generation.

`vech` *vech Create the half-vectorization of a matrix*

Description

vech Create the half-vectorization of a matrix

Usage

```
vech(x)
```

Arguments

`x`	a matrix, the half-vectorization of which is desired

Details

This function returns the vectorized form of the lower triangle of a matrix, including the diagonal. The upper triangle is ignored with no checking that the provided matrix is symmetric.

Value

A vector containing the lower triangle of the matrix, including the diagonal.

Examples

```
vech(matrix(c(1, 0.5, 0.5, 1), nrow = 2, ncol = 2))
```

Index