Package 'pedprobr'

July 5, 2020

Type Package

Title Probability Computations on Pedigrees

Version 0.3

Description An implementation of the Elston-Stewart algorithm for calculating pedigree likelihoods given genetic marker data (Elston and Stewart (1971) <doi:10.1159/000152448>). The standard algorithm is extended to allow inbred founders. Mutation modelling is supported by the 'pedmut' package. 'pedprobr' is part of the ped suite, a collection of packages for pedigree analysis in R, based on 'pedtools' for handling pedigrees and markers.

License GPL-3

URL https://github.com/magnusdv/pedprobr

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 3.1.0), pedtools

Imports pedmut

Suggests testthat

RoxygenNote 7.1.0

NeedsCompilation no

Author Magnus Dehli Vigeland [aut, cre]
(<https://orcid.org/0000-0002-9134-4962>)

Maintainer Magnus Dehli Vigeland <m.d.vigeland@medisin.uio.no>

Repository CRAN

Date/Publication 2020-07-05 04:50:19 UTC

R topics documented:

`allGenotypes` *Genotype matrix*

Description

An autosomal marker with `n` alleles has `choose(n+1,2)` possible unordered genotypes. This function returns these as rows in a matrix.

Usage

```
allGenotypes(n)
```

Arguments

`n`	A positive integer.

Value

An integer matrix with two columns and `choose(n+1,2)` rows.

Examples

```
allGenotypes(3)
```

`genoCombinations` *Genotype combinations*

Description

Returns the possible genotype combinations in a pedigree, given partial marker data. This function is mainly for internal use.

Usage

```
genoCombinations(x, partialmarker, ids, make.grid = TRUE)
```

Arguments

x	a ped() object.
partialmarker	a marker() object compatible with x.
ids	a vector with ID labels of one or more pedigree members.
make.grid	a logical indicating if the result should be simplified to a matrix.

Value

If make.grid = FALSE (the default) the function returns a list of integer vectors, one vector for each element of ids. Each integer represents a genotype, in the form of a row number of the matrix allGenotypes(n), where n is the number of alleles of the marker.

If make.grid = TRUE, the cartesian product of the vectors is taken, resulting in a matrix with one column for each element of ids.

HWprob *Hardy-Weinberg probabilities*

Description

Hardy-Weinberg probabilities

Usage

```
HWprob(allele1, allele2, afreq, f = 0)
```

Arguments

allele1, allele2	Vectors of equal length, containing alleles in the form of indices of afreq
afreq	A numeric vector with allele frequencies
f	A single number in [0, 1]; the inbreeding coefficient

Value

A numeric vector of the same length as allele1 and allele2

Examples

```
p = 0.1; q = 1-p
hw = HWprob(c(1,1,2), c(1,2,2), c(p, q))
stopifnot(all.equal(hw, c(p^2, 2*p*q, q^2)))
```

`likelihood` *Pedigree likelihood*

Description

This function is the heart of pedprobr. It computes the likelihood of a pedigree (or a list of pedigrees) given genotypes for a marker or a pair of linked markers.

Usage

```
likelihood(x, ...)

## S3 method for class 'ped'
likelihood(
  x,
  marker1 = NULL,
  marker2 = NULL,
  rho = NULL,
  setup = list(),
  eliminate = 0,
  logbase = NULL,
  loop_breakers = NULL,
  verbose = FALSE,
  theta = NULL,
  ...
)

## S3 method for class 'list'
likelihood(x, marker1, marker2 = NULL, logbase = NULL, total = TRUE, ...)
```

Arguments

`x`	A ped object, a `singleton` object, or a list of such objects.
`...`	Further arguments.
`marker1`	One or several markers compatible with `x`. Several input forms are possible: • A `marker()` object compatible with `x`. • A list of marker objects • A vector of names or indices of markers attached to `x`. If `x` is a list, this is the only valid input.
`marker2`	Either NULL, or a single marker compatible with `x`. See Details.
`rho`	The recombination rate between `marker1` and `marker2`. To make biological sense `rho` should be between 0 and 0.5.
`setup`	For internal use.

eliminate	Mostly for internal use: a non-negative integer indicating the number of iterations in the internal genotype-compatibility algorithm. Positive values can save time if the number of alleles is large.
logbase	A numeric, or NULL. If numeric the log-likelihood is returned, with logbase as basis for the logarithm.
loop_breakers	A vector of ID labels indicating loop breakers. If NULL (default), automatic selection of loop breakers will be performed. See breakLoops().
verbose	A logical.
theta	Deprecated; renamed to rho.
total	A logical; if TRUE, the product of the likelihoods is returned, otherwise a vector with the individual likelihoods.

Details

The implementation is based on the peeling algorithm of Elston and Stewart (1971). A variety of situations are covered; see the Examples section for some demonstrations.

- complex inbred pedigrees
- pedigrees with inbred founders
- autosomal and X-linked markers
- a single marker or two linked markers
- markers with mutation models

Value

A numeric with the same length as the number of markers indicated by marker1. If logbase is a positive number, the output is log(likelihood,logbase).

Author(s)

Magnus Dehli Vigeland

References

Elston and Stewart (1971). *A General Model for the Genetic Analysis of Pedigree Data*. doi: 10.1159/000152448

Examples

```
### Example 1: Likelihood of trio with inbred father

x = cousinPed(0, child = TRUE)
x = addSon(x, 5)
x = relabel(x, old = 5:7, new = c("father", "mother", "child"))

# Equifrequent SNP marker: father homozygous, child heterozygous
m = marker(x, father = 1, child = 1:2)
```

```
x = addMarkers(x, m)

# Plot with genotypes
plot(x, marker = 1)

# Compute the likelihood
lik1 = likelihood(x, marker1 = 1)


### Example 2: Same as above, but using founder inbreeding

# Extract the trio
y = subset(x, c("father", "mother", "child"))

# Indicate that the father has inbreeding coefficient 1/4
founderInbreeding(y, "father") = 1/4

# Plot (notice the inbreeding coefficient)
plot(y, marker = 1)

# Likelihood should be the same as above
lik2 = likelihood(y, marker1 = 1)

stopifnot(all.equal(lik1, lik2))


### Example 3: Modelling mutations
# TODO after next pedtools release
```

`merlin` *Pedigree likelihood computed by MERLIN*

Description

For this functions to work, the program MERLIN (see References below) must be installed and correctly pointed to in the PATH variable. The `merlin()` function is a general wrapper which runs MERLIN with the indicated options, after creating the appropriate input files. For convenience, MERLIN's "–likelihood" functionality is wrapped in a separate function.

Usage

```
merlin(
  x,
  options,
  markers = NULL,
  verbose = TRUE,
  generateFiles = TRUE,
  cleanup = TRUE,
```

```
  dir = tempdir(),
  logfile = NULL
)

likelihoodMerlin(x, ...)
```

Arguments

x	a ped object.
options	a single string containing all arguments to merlin except for the input file indications.
markers	a vector of names or indices of markers attached to x. (Default: all markers).
verbose	a logical.
generateFiles	a logical. If TRUE (default), input files to MERLIN named '_merlin.ped', '_merlin.dat', '_merlin.map', and '_merlin.freq' are created in the directory indicated by dir. If FALSE, no files are created.
cleanup	a logical. If TRUE (default), the MERLIN input files are deleted after the call to MERLIN.
dir	the name of the directory where input files should be written.
logfile	a character. If this is given, the MERLIN screen output will be dumped to a file with this name.
...	Further arguments passed on to merlin

Details

The merlin() function creates input files "_merlin.ped", "_merlin.dat", "_merlin.map" and "_merlin.freq" in the dir directory, and then runs the following command through a call to system():

```
merlin -p _merlin.ped -d _merlin.dat -m _merlin.map -f
_merlin.freq  <options>
```

likelihoodMerlin() first runs merlin() with options = "--likelihood --bits:100 --megabytes:4000 --quiet", and then extracts the likelihood values from the MERLIN output. Note that the output is the *total* likelihood including all markers.

Value

merlin() returns the screen output of MERLIN invisibly.

likelihoodMerlin() returns a single number; the total likelihood using all indicated markers.

Author(s)

Magnus Dehli Vigeland

References

http://csg.sph.umich.edu/abecasis/Merlin/

Examples

```
### Requires MERLIN to be installed ###

x = nuclearPed(1)
m1 = marker(x, "1" = 1:2)           # likelihood = 1/2
m2 = marker(x, "1" = 1, "3" = 1:2)    # likelihood = 1/8
x = setMarkers(x, list(m1,m2))

# Likelihood computation by MERLIN:
lik1 = likelihoodMerlin(x, markers = 1, verbose = FALSE)
lik2 = likelihoodMerlin(x, markers = 2, verbose = FALSE)
likTot = likelihoodMerlin(x, verbose = FALSE)
stopifnot(all.equal(
  round(c(lik1, lik2, likTot), c(3,3,4)), c(1/2, 1/8, 1/16)))

y = list(singleton(1), singleton(2))
y = setMarkers(y, locus = list(alleles=1:2))
genotype(y[[1]], marker = 1, id = '1') = 1:2
genotype(y[[2]], marker = 1, id = '2') = 1
lik = likelihoodMerlin(y, verbose = FALSE)
stopifnot(all.equal(round(lik, 3), 1/8))
```

---

oneMarkerDistribution *Genotype distribution for a single marker*

Description

Computes the genotype probability distribution of one or several pedigree members, possibly conditional on known genotypes for the marker.

Usage

```
oneMarkerDistribution(
  x,
  ids,
  partialmarker,
  loop_breakers = NULL,
  eliminate = 0,
  grid.subset = NULL,
  verbose = TRUE
)
```

Arguments

x	A ped object.
ids	A numeric with ID labels of one or more pedigree members.
partialmarker	Either a marker object or the name (or index) of a marker attached to x.
loop_breakers	(Only relevant if the pedigree has loops). A vector with ID labels of individuals to be used as loop breakers. If NULL (default) loop breakers are selected automatically. See breakLoops().
eliminate	A non-negative integer, indicating the number of iterations in the internal genotype-compatibility algorithm. Positive values can save time if partialmarker has many alleles.
grid.subset	(Optional; not relevant for most users.) A numeric matrix describing a subset of all marker genotype combinations for the ids individuals. The matrix should have one column for each of the ids individuals, and one row for each combination: The genotypes are described in terms of the matrix M = allGenotypes(n), where n is the number of alleles for the marker. If the entry in column j is the integer k, this means that the genotype of individual ids[j] is row k of M.
verbose	A logical.

Value

A named k-dimensional array, where k = length(ids), with the joint genotype distribution for the ids individuals. The probabilities are conditional on the known genotypes and the allele frequencies of partialmarker.

Author(s)

Magnus Dehli Vigeland

See Also

twoMarkerDistribution()

Examples

```
# Trivial example giving Hardy-Weinberg probabilities
s = singleton(id = 1)
m = marker(s, alleles = 1:2) # equifrequent SNP
oneMarkerDistribution(s, ids = 1, partialmarker = m)

# Conditioning on a partial genotype
genotype(m, id = 1) = c(1, NA)
oneMarkerDistribution(s, ids = 1, partialmarker = m)

# Genotype distribution for a child of heterozygous parents
trio = nuclearPed(father = "fa", mother = "mo", child = "ch")
m1 = marker(trio, fa = 1:2, mo = 1:2)
oneMarkerDistribution(trio, ids = "ch", partialmarker = m1)
```

```
# Joint distribution of the parents, given that the child is heterozygous
m2 = marker(trio, ch = 1:2, alleles = 1:2, afreq = c(0.5, 0.5))
oneMarkerDistribution(trio, ids = c("fa", "mo"), partialmarker = m2)

# A different example: The genotype distribution of an individual (id = 5)
# whose half cousin (id = 9) is homozygous for a rare allele.
y = halfCousinPed(degree = 1)
snp = marker(y, `9` = "a", alleles = c("a", "b"), afreq = c(0.01, 0.99))
plot(y, snp)
oneMarkerDistribution(y, ids = 5, partialmarker = snp)
```

`pedprobr` *pedprobr: Probability Computations on Pedigrees*

Description

An implementation of the Elston-Stewart algorithm for calculating pedigree likelihoods given genetic marker data (Elston and Stewart (1971), doi: 10.1159/000152448). The standard algorithm is extended to allow inbred founders. Mutation modelling is included via the 'pedmut' package. 'pedprobr' is part of the ped suite, a collection of packages for pedigree analysis in R, based on 'pedtools' for handling pedigrees and markers.

`twoMarkerDistribution` *Genotype distribution for two linked markers*

Description

Computes the joint genotype distribution of two markers for a specified pedigree member, conditional on known genotypes and the recombination rate between the markers.

Usage

```
twoMarkerDistribution(
  x,
  id,
  partialmarker1,
  partialmarker2,
  rho,
  loop_breakers = NULL,
  eliminate = 99,
  theta = NULL,
  verbose = TRUE
)
```

Arguments

x	A ped object.
id	A single ID label.
partialmarker1, partialmarker2	Either a marker object, or the name (or index) of a marker attached to x.
rho	A single numeric in the interval [0, 0.5]: the recombination fraction between the two markers.
loop_breakers	(Only relevant if the pedigree has loops). A vector with ID labels of individuals to be used as loop breakers. If NULL (default) loop breakers are selected automatically. See breakLoops().
eliminate	A non-negative integer, indicating the number of iterations in the internal algorithm for reducing the genotype space. Positive values can save time if partialmarker1 and/or partialmarker2 have many alleles.
theta	deprecated; renamed to rho.
verbose	A logical.

Value

A named matrix giving the joint genotype distribution.

Author(s)

Magnus Dehli Vigeland

See Also

oneMarkerDistribution()

Examples

```
# A sib-pair pedigree
x = nuclearPed(children = c("bro1", "bro2"))

# Two SNP markers; first brother homozygous for the `1` allele
SNP1 = SNP2 = marker(x, bro1 = c(1,1), alleles = 1:2)

plot(x, marker = list(SNP1, SNP2))

# Genotype distribution for the brother: Depends on rho
twoMarkerDistribution(x, id = "bro2", SNP1, SNP2, rho = 0)
twoMarkerDistribution(x, id = "bro2", SNP1, SNP2, rho = 0.5)

# X-linked
chrom(SNP1) = chrom(SNP2) = "X"

plot(x, marker = list(SNP1, SNP2))
```

```
twoMarkerDistribution(x, id = "bro2", SNP1, SNP2, rho = 0)
twoMarkerDistribution(x, id = "bro2", SNP1, SNP2, rho = 0.5)
```

Index