Package 'pedprobr'

June 22, 2026

Type Package

Title Probability Computations on Pedigrees

Version 1.1.0

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. 'pedprobr' is part of the 'pedsuite', a collection of packages for pedigree analysis in R. In particular, 'pedprobr' depends on 'pedtools' for pedigree manipulations and 'pedmut' for mutation modelling. For more information, see 'Pedigree Analysis in R' (Vigeland, 2021, ISBN:9780128244302).

License GPL (>= 2)

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

BugReports https://github.com/magnusdv/pedprobr/issues

Depends pedtools (>= 2.11.0), R (>= 4.2.0)

Imports pedmut (>= 0.9.0)

Suggests testthat

Encoding UTF-8

Language en-GB

SystemRequirements MERLIN (https://csg.sph.umich.edu/abecasis/merlin/) for calculations involving multiple linked markers.

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-06-22 15:50:02 UTC

Contents

allGenotypes . . . 2
genoCombinations . . . 3
haldane . . . 3
HWprob . . . 4
likelihood . . . 5
lumpAlleles . . . 8
merlin . . . 10
oneMarkerDistribution . . . 13
setMutationModel . . . 14
twoMarkerDistribution . . . 16

Index **18**

`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 = x$MARKERS[[1]], ids, make.grid = TRUE)
```

Arguments

x	a pedtools::ped() object.
partialmarker	a pedtools::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.

haldane *Genetic map functions*

Description

Simple implementations of the classical map functions of Haldane and Kosambi, relating the genetic distance and the recombination rate between two linked loci.

Usage

```
haldane(cM = NULL, rho = NULL)

kosambi(cM = NULL, rho = NULL)
```

Arguments

cM	A numeric vector with genetic distances in centiMorgan, or NULL.
rho	A numeric vector with recombination rates, or NULL.

Value

A numeric of the same length as the input.

Examples

```
cM = 0:200
dat = cbind(Haldane = haldane(cM = cM),
            Kosambi = kosambi(cM = cM))
matplot(cM, dat, ylab = "Recombination rate", type = "l")
legend("topleft", legend = colnames(dat), col = 1:2, lty = 1:2)


rho = seq(0, 0.49, length = 50)
dat2 = cbind(Haldane = haldane(rho = rho),
             Kosambi = kosambi(rho = rho))
matplot(rho, dat2, xlab = "Recombination rate", ylab = "cM", type = "l")
legend("topleft", legend = colnames(dat), col = 1:2, lty = 1:2)
```

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

The `likelihood()` and `likelihood2()` functions constitute the heart of **pedprobr**. The former computes the pedigree likelihood for each indicated marker. The latter computes the likelihood for a pair of linked markers separated by a given recombination rate.

Usage

```
likelihood(x, ...)

## S3 method for class 'ped'
likelihood(
  x,
  markers = NULL,
  peelOrder = NULL,
  lump = TRUE,
  special = FALSE,
  alleleLimit = Inf,
  theta = 0,
  logbase = NULL,
  loopBreakers = NULL,
  allX = NULL,
  verbose = FALSE,
  .diagnostics = FALSE,
  ...
)

## S3 method for class 'list'
likelihood(x, markers = NULL, logbase = NULL, loopBreakers = NULL, ...)

likelihood2(x, ...)

## S3 method for class 'ped'
likelihood2(
  x,
  marker1,
  marker2,
  rho = NULL,
  peelOrder = NULL,
  lump = TRUE,
  special = TRUE,
  alleleLimit = Inf,
  logbase = NULL,
  loopBreakers = NULL,
```

```
  verbose = FALSE,
  .diagnostics = FALSE,
  ...
)

## S3 method for class 'list'
likelihood2(x, marker1, marker2, logbase = NULL, loopBreakers = NULL, ...)
```

Arguments

x	A ped object, a singleton object, or a list of such objects.
...	Further arguments.
markers	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.
peelOrder	For internal use.
lump	Activate allele lumping when this does not change the likelihood. If any markers use a non-lumpable mutation model, the special argument may be used to apply more advanced methods.
special	A logical indicating if special lumping procedures should be attempted if the mutation model is not generally lumpable. By default FALSE in likelihood() and TRUE in likelihood2().
alleleLimit	A positive number or Inf (default). If the mutation model is not generally lumpable, and the allele count exceeds this limit, the calculation switches to an equal mutation model to enable lumping. This is an approximation intended to avoid very large computations.
theta	Theta correction.
logbase	Either NULL (default) or a positive number indicating the basis for logarithmic output. Typical values are exp(1) and 10.
loopBreakers	A vector of ID labels indicating loop breakers. Repeated values are allowed, as are founders. If NULL, loop breakers are chosen automatically. This is usually recommended. See pedtools::breakLoops().
allX	For internal use; set to TRUE if all markers are X-chromosomal.
verbose, .diagnostics	Logicals. If .diagnostics = TRUE, the peeling process prints diagnostic messages.
marker1, marker2	Single markers compatible with x.
rho	The recombination rate between marker1 and marker2. To make biological sense rho should be between 0 and 0.5.

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.

- pedigrees with loops
- pedigrees with inbred founders
- autosomal and X-linked markers
- markers with mutation models
- single markers or two linked markers

For more than two linked markers, see `likelihoodMerlin()`.

Allele lumping merges unobserved alleles when this does not change the likelihood. This can greatly reduce computation time for highly polymorphic markers, particularly in `likelihood2()` which is prone to run out of memory without lumping. If a non-lumpable mutation model is used, specialised lumping may still be possible in some situations. This is attempted if `special = TRUE`, which is the default in `likelihood2()` but not in `likelihood()`.

Value

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

`likelihood2()` returns a single value for the two linked markers.

If `x` is a list, component likelihoods are multiplied (or log-likelihoods are added).

Author(s)

Magnus Dehli Vigeland

References

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

See Also

`likelihoodMerlin()`, for likelihoods involving more than 2 linked markers.

Examples

```
### Simple likelihoods ###
p = 0.1
q = 1 - p
afr = c("1" = p, "2" = q)

# Singleton
s = singleton() |> addMarker(geno = "1/2", afreq = afr)
stopifnot(all.equal(likelihood(s), 2*p*q))
```

```
# Trio
x = nuclearPed() |> addMarker(geno = c("1/1", "1/2", "1/1"), afreq = afr)
lik = likelihood(x, verbose = TRUE)
stopifnot(all.equal(lik, p^2 * 2*p*q * 0.5))


### Example with inbred founder ###

# Set 100% inbreeding for the father in the previous example
y = setFounderInbreeding(x, ids = 1, value = 1)

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

stopifnot(all.equal(likelihood(y), p * 2*p*q * 0.5))


### Example with two linked markers

# Add a second marker, highly polymorphic
x = addMarker(x, geno = c(NA, NA, "1/1"), alleles = 1:10)

# Likelihood assuming complete linkage
likelihood2(x, 1, 2, rho = 0, verbose = TRUE)
```

`lumpAlleles` *Allele lumping*

Description

Perform allele lumping (i.e., merging unobserved alleles) for all markers attached to the input pedigree.

Usage

```
lumpAlleles(
  x,
  markers = NULL,
  always = FALSE,
  special = TRUE,
  alleleLimit = Inf,
  verbose = FALSE
)
```

Arguments

x	A ped object or a list of such.

markers	A vector of names or indices referring to markers attached to x. (Default: All markers.)
always	A logical. If TRUE, lumping is always attempted. By default (FALSE) lumping is skipped for markers where no individuals, or all individuals, are genotyped.
special	A logical. If TRUE, special lumping procedures (depending on the pedigree) will be attempted if the marker has a mutation model that is not generally lumpable (in the Kemeny-Snell sense).
alleleLimit	A positive number or Inf (default). If the mutation model is not generally lumpable, and the allele count exceeds this limit, switch to an equal model with the same rate and reapply lumping.
verbose	A logical.

Value

An object similar to x, but whose attached markers have reduced allele set.

Examples

```
x = nuclearPed() |> addMarker(geno = c(NA, NA, "1/1"), alleles = 1:5)

# Before lumping
afreq(x, 1)

# Lump
y = lumpAlleles(x, verbose = TRUE)
afreq(y, 1)

# With lumpable mutation model
x2 = setMutmod(x, model = "equal", rate = 0.1)
mutmod(x2, 1)

y2 = lumpAlleles(x2, verbose = TRUE)
mutmod(y2, 1)

# Mutation model requiring special lumping
x3 = setMutmod(x, model = "random", rate = 0.1, seed = 1)
mutmod(x3, 1)

# Lump
y3 = lumpAlleles(x3, verbose = TRUE)

mutmod(y3, 1)

stopifnot(all.equal(likelihood(x), likelihood(y)),
          all.equal(likelihood(x2), likelihood(y2)),
          all.equal(likelihood(x3), likelihood(y3)))
```

merlin *Pedigree likelihoods computed by MERLIN*

Description

These functions enable users to call MERLIN (Abecasis et al., 2002) from within R.

Usage

```
merlin(
  x,
  options,
  markers = NULL,
  linkageMap = NULL,
  verbose = TRUE,
  generateFiles = TRUE,
  cleanup = TRUE,
  dir = tempdir(),
  logfile = NULL,
  merlinpath = NULL,
  checkpath = TRUE
)

likelihoodMerlin(
  x,
  markers = NULL,
  linkageMap = NULL,
  rho = NULL,
  logbase = NULL,
  perChrom = FALSE,
  options = "--likelihood --bits:100 --megabytes:4000 --quiet",
  ...
)

checkMerlin(program = NULL, version = TRUE, error = FALSE)
```

Arguments

x	A pedtools::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).
linkageMap	A data frame with three columns (chromosome; marker name; centiMorgan position) to be used as the marker map by MERLIN.
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.
merlinpath	The path to the folder containing the merlin executables. If the executables are on the system's search path, this can be left as NULL (default).
checkpath	A logical indicating whether to check that the merlin executable is found.
rho	A vector of length one less than the number of markers, specifying the recombination rate between each consecutive pair.
logbase	Either NULL (default) or a positive number indicating the basis for logarithmic output. Typical values are exp(1) and 10.
perChrom	A logical; if TRUE, likelihoods are reported per chromosome.
...	Further arguments passed on to merlin().
program	A character containing "merlin", "minx" or both (default), optionally including full paths.
version	A logical. If TRUE (default), it is checked that running program produces a printout starting with "MERLIN 1.1.2".
error	A logical, indicating if an error should be raised if program is not found. Default: FALSE.

Details

For these functions to work, the program MERLIN must be installed (see link in the Reference section below) 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.

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.

For likelihood computations with linked markers, the argument rho should indicate the recombination fractions between each consecutive pair of markers (i.e., rho[i] is the recombination rate between markers i-1 and i). These will be converted to centiMorgan distances using Haldane's map function, and used to create genetic marker map in a MERLIN-friendly format.

Value

`merlin()` returns the screen output of MERLIN invisibly.

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

`checkMerlin()` returns TRUE if the MERLIN executable indicated by `program` is found on the system. Otherwise FALSE, or (if `error = TRUE`) an error is raised.

Author(s)

Magnus Dehli Vigeland

References

Abecasis et al. (2002) Nat Gen 30:97-101. https://csg.sph.umich.edu/abecasis/merlin/.

Examples

```
if(checkMerlin()) {

### Trivial example for validation
x = nuclearPed(1) |>
  addMarker("1" = "1/2") |>            # likelihood = 1/2
  addMarker("1" = "1/1", "3" = "1/2")  # likelihood = 1/8

# MERLIN likelihoods
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)))

# Example with ped lists
y = singletons(1:2) |>
  addMarker(`1` = "1/2", `2` = "1/1", alleles = 1:2)
lik = likelihoodMerlin(y, verbose = FALSE)
stopifnot(all.equal(round(lik, 3), 1/8))

### Linked markers
z = nuclearPed(2)
m = marker(z, geno = c("1/1", "1/2", "1/2", "1/2"))
z = setMarkers(z, list(m, m))

# By MERLIN...
L1 = likelihoodMerlin(z, markers = 1:2, rho = 0.25, verbose = FALSE)

# ...and by pedprobr
L2 = likelihood2(z, marker1 = 1, marker2 = 2, rho = 0.25)

# stopifnot(all.equal(signif(L1, 3), signif(L2, 3)))
}
```

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,
  marker = 1,
  loopBreakers = NULL,
  grid.subset = NULL,
  output = c("array", "table", "sparse"),
  verbose = TRUE
)
```

Arguments

x	A ped object or a list of such.
ids	A vector of ID labels of one or more members of x.
marker	Either a marker object or the name (or index) of a marker attached to x. If x has multiple components, only the latter is allowed.
loopBreakers	(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 pedtools::breakLoops().
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.
output	A character string, either "array" (default), "table" or "sparse". See Value.
verbose	A logical.

Value

The output format depends on the output argument:

- "array": A named k-dimensional array, where k = length(ids), with the joint genotype distribution for the ids individuals, conditional on the known genotypes if present.
- "table": A data frame with k+1 columns, where each row corresponds to a genotype combination, and the last column prob gives the probability.

- "sparse": A data frame with the same structure as the "table" output, but only combinations with non-zero probability are included.

See Also

twoMarkerDistribution()

Examples

```
# Trivial example: Hardy-Weinberg probabilities for an equifrequent SNP
s = singleton(id = 1) |> addMarker(alleles = 1:2, afreq = c(0.5, 0.5))
oneMarkerDistribution(s, ids = 1)

# Conditioning on a partial genotype
s = setGenotype(s, ids = 1, geno = "1/-")
oneMarkerDistribution(s, ids = 1)

# Genotype distribution for a child of heterozygous parents
trio = nuclearPed(father = "fa", mother = "mo", child = "ch") |>
  addMarker(fa = "1/2", mo = "1/2")
oneMarkerDistribution(trio, ids = "ch")

# Joint distribution of the parents, given that the child is heterozygous
trio = addMarker(trio, ch = "1/2")
ids = c("fa", "mo")
oneMarkerDistribution(trio, ids = ids, marker = 2)

# Table output of the previous example
oneMarkerDistribution(trio, ids = ids, marker = 2, output = "table")
oneMarkerDistribution(trio, ids = ids, marker = 2, output = "sparse")

# A different example: The genotype distribution of an individual (id = 8)
# whose half cousin (id = 9) is homozygous for a rare allele.
y = halfCousinPed(degree = 1) |>
  addMarker("9" = "a/a", afreq = c(a = 0.01, b = 0.99))

oneMarkerDistribution(y, ids = 8)

# Multi-component (trivial) example
z = singletons(1:2) |> addMarker(`1` = "1/2", `2` = "1/2", alleles = 1:2)
oneMarkerDistribution(z, 1:2)
oneMarkerDistribution(z, 1:2, output = "sparse")
```

---

setMutationModel *Set a mutation model*

Description

NB: This function has been replaced by `pedtools::setMutmod()`. This function attaches mutation models to a pedigree with marker data, calling `pedmut::mutationModel()` for creating the models.

Usage

```
setMutationModel(x, model, markers = NULL, ...)
```

Arguments

`x`	A ped object or a list of such.
`model`	A model name implemented by `pedmut::mutationModel()` (see Details), or NULL.
`markers`	A vector of names or indices referring to markers attached to `x`. (Default: All markers.)
`...`	Arguments forwarded to `pedmut::mutationModel()`, e.g., `rate`.

Details

Currently, the following models are handled:

- `equal` : All mutations equally likely; probability $1 - rate$ of no mutation
- `proportional` : Mutation probabilities are proportional to the target allele frequencies
- `onestep`: A mutation model for microsatellite markers, allowing mutations only to the nearest neighbours in the allelic ladder. For example, '10' may mutate to either '9' or '11', unless '10' is the lowest allele, in which case '11' is the only option. This model is not applicable to loci with non-integral microvariants.
- `stepwise`: A common model in forensic genetics, allowing different mutation rates between integer alleles (like '16') and non-integer "microvariants" like '9.3'). Mutations also depend on the size of the mutation if the parameter 'range' differs from 1.
- `custom` : Allows any mutation matrix to be provided by the user, in the `matrix` parameter
- `random` : This produces a matrix of random numbers, where each row is normalised so that it sums to 1
- `trivial` : The identity matrix; i.e. no mutations are possible.

Value

An object similar to `x`.

Examples

```
### Example requires the pedmut package ###
if (requireNamespace("pedmut", quietly = TRUE)){

# A pedigree with data from a single marker
x = nuclearPed(1) |>
```

```
  addMarker(geno = c("a/a", NA, "b/b")) # mutation!

# Set `equal` model
y = setMutationModel(x, marker = 1, model = "equal", rate = 0.01)

# Inspect model
mutmod(y, 1)

# Likelihood
likelihood(y, 1)

# Remove mutation model
z = setMutationModel(y, model = NULL)
stopifnot(identical(z, x))
}
```

`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,
  marker1 = 1,
  marker2 = 2,
  rho = NULL,
  loopBreakers = NULL,
  lumpSpecial = TRUE,
  verbose = TRUE
)
```

Arguments

`x`	A ped object or a list of such.
`id`	A single ID label.
`marker1, marker2`	Either `marker` objects, or the names (or indices) of markers attached to `x`.
`rho`	A single numeric in the interval `[0, 0.5]`: the recombination fraction between the two markers.

loopBreakers	(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 pedtools::breakLoops().
lumpSpecial	A logical, passed on to likelihood2().
verbose	A logical.

Value

A named matrix giving the joint genotype distribution.

See Also

oneMarkerDistribution()

Examples

```
# A sib-pair with two SNPs. The first child is homozygous 1/1.
x = nuclearPed(children = c("bro1", "bro2")) |>
  addMarker(bro1 = "1/1", alleles = 1:2, afreq = c(0.5, 0.5)) |>
  addMarker(bro1 = "1/1", alleles = 1:2, afreq = c(0.5, 0.5))

plot(x, marker = 1:2)

# Genotype distribution for the brother depends on linkage
twoMarkerDistribution(x, id = "bro2", rho = 0)
twoMarkerDistribution(x, id = "bro2", rho = 0.5)

### Same example on X
y = setChrom(x, marker = 1:2, chrom = "X")

plot(y, marker = 1:2)

twoMarkerDistribution(y, id = "bro2", rho = 0)
twoMarkerDistribution(y, id = "bro2", rho = 0.5)
```

Index

allGenotypes, 2

checkMerlin *(*merlin*)*, 10

genoCombinations, 3

haldane, 3
HWprob, 4

kosambi *(*haldane*)*, 3

likelihood, 5
likelihood2 *(*likelihood*)*, 5
likelihood2(), *17*
likelihoodMerlin *(*merlin*)*, 10
likelihoodMerlin(), *7*
lumpAlleles, 8

merlin, 10

oneMarkerDistribution, 13
oneMarkerDistribution(), *17*

pedmut::mutationModel(), *15*
pedtools::breakLoops(), *6*, *13*, *17*
pedtools::marker(), *3*
pedtools::ped(), *3*, *10*
pedtools::setMutmod(), *15*

setMutationModel, 14
system(), *11*

twoMarkerDistribution, 16
twoMarkerDistribution(), *14*