Package 'pedmut'

November 16, 2022

Title Mutation Models for Pedigree Likelihood Computations

Version 0.4.0

Description A collection of functions for modelling mutations in pedigrees with marker data, as used e.g. in likelihood computations with microsatellite data. Implemented models include proportional and stepwise models, as well as random models for experimental work, and custom models allowing the user to apply any valid mutation matrix. Allele lumping is done following the lumpability criteria of Kemeny and Snell (1976), ISBN:0387901922.

Depends R (>= 3.5.0)

License GPL-3

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

Encoding UTF-8

Language en-GB

RoxygenNote 7.2.2

Suggests testthat

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 2022-11-16 22:50:02 UTC

R topics documented:

lumpedMatrix *Combine alleles in a mutation matrix*

Description

Reduce a mutation matrix by combining a set of alleles into one "lump", if this can be done without distorting the mutation process of the remaining alleles. Such "allele lumping" can give dramatic efficiency improvements in likelihood computations with multi-allelic markers, in cases where only some of the alleles are observed in the pedigree.

Usage

```
lumpedMatrix(mutmat, lump, afreq = attr(mutmat, "afreq"))
```

Arguments

mutmat	A mutationModel object
lump	A nonempty subset of the colnames of mutmat (i.e. the allele labels)
afreq	A vector with frequency vector, of the same length as the size of mutmat

Value

A reduced mutation model. If the original matrix has dimensions $n \times n$, the result will be $k \times k$, where $k = n - length(lump) + 1$.

Examples

```
m = mutationMatrix("eq", alleles = 1:10, rate = 0.1)
afreq = rep(1/100, 100)

# Suppose only alleles 1 and 2 are observed.
# The lumped model is then equivalent to `m`:
mLump = lumpedMatrix(m, afreq = afreq, lump = 3:10)
mLump
```

model_properties *Mutation model properties*

Description

Functions for checking various properties of a mutation model, including stationarity, reversibility and lumpability.

Usage

```
isStationary(mutmat, afreq)

isReversible(mutmat, afreq)

isLumpable(mutmat, lump)

alwaysLumpable(mutmat)
```

Arguments

mutmat	A mutation matrix
afreq	A vector with frequency vector, of the same length as the size of mutmat
lump	A nonempty subset of the colnames of mutmat (i.e. the allele labels)

Value

Each of these functions returns TRUE of FALSE

Examples

```
# "proportional" models are stationary and reversible
afr = c(0.2, 0.3, 0.5)
m_prop = mutationMatrix(model = "prop", alleles = 1:3, afreq = afr, rate = 0.1)
stopifnot(isStationary(m_prop, afr), isReversible(m_prop, afr))

# "equal" model is stationary and reversible only when freqs are equal
m_eq = mutationMatrix(model = "eq", alleles = 1:3, rate = 0.1)
stopifnot(isStationary(m_eq, rep(1/3, 3)), isReversible(m_eq, rep(1/3, 3)))
stopifnot(!isStationary(m_eq, afr), !isReversible(m_eq, afr))

# "equal" and "proportional" models allow allele lumping
stopifnot(isLumpable(m_eq, lump = 1:2))
stopifnot(isLumpable(m_prop, lump = 1:2))

# In fact lumpable for any allele subset
stopifnot(alwaysLumpable(m_eq), alwaysLumpable(m_prop))
```

---

mutationMatrix *Mutation matrix*

Description

Construct mutation matrices for pedigree likelihood computations.

Usage

```
mutationMatrix(
  model = c("custom", "equal", "proportional", "random", "onestep", "stepwise",
    "trivial"),
  matrix = NULL,
  alleles = NULL,
  afreq = NULL,
  rate = NULL,
  seed = NULL,
  rate2 = NULL,
  range = NULL
)

validateMutationMatrix(mutmat, alleles = NULL)
```

Arguments

model	A string: either "custom", "equal", "proportional", "random", "stepwise" or "onestep"
matrix	When model is "custom", this must be a square matrix with nonnegative real entries and row sums equal to 1
alleles	A character vector (or coercible to character) with allele labels. Required in all models, except "custom" if matrix has dimnames
afreq	A numeric vector of allele frequencies. Required in model "proportional"
rate	A number between 0 and 1. Required in models "equal", "proportional", "stepwise" and "onestep"
seed	A single number. Optional parameter in the "random" model, passed on to set.seed()
rate2	A number between 0 and 1. The mutation rate between integer alleles and microvariants. Required in the "stepwise" model
range	A positive number. The relative probability of mutating n+1 steps versus mutating n steps. Required in the "stepwise" model
mutmat	An object of class mutationMatrix

Details

Descriptions of the models:

- custom : Allows any mutation matrix to be provided by the user, in the matrix parameter
- equal : All mutations equally likely; probability $1 - rate$ of no mutation
- proportional : Mutation probabilities are proportional to the target allele frequencies
- random : This produces a matrix of random numbers, where each row is normalised so that it sums to 1

- `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.
- `trivial` : The identity matrix; i.e. no mutations are possible.

Value

A square matrix with entries in `[0, 1]`, with the allele labels as both colnames and rownames.

Examples

```
mutationMatrix(alleles = 1:3, model = "equal", rate = 0.05)
```

`mutationModel` *Mutation models*

Description

Constructor for the class `mutationModel`. An object of this class is essentially a list of two mutation matrices, named "female" and "male".

Usage

```
mutationModel(
  model,
  alleles = NULL,
  afreq = NULL,
  matrix = NULL,
  rate = NULL,
  rate2 = NULL,
  range = NULL,
  seed = NULL,
  validate = TRUE
)

validateMutationModel(mutmod, alleles = NULL)
```

Arguments

`model` Either:

- a `mutationModel` object (returned unchanged after validation)
- a single `mutationMatrix` object (will be applied to both genders)

- a list of two mutationMatrix objects, named "female" and "male"
- a single model name (see mutationMatrix() for valid options)
- a list of two model names, named "female" and "male"

alleles	A character vector with allele labels; passed on to mutationMatrix().
afreq	A numeric vector of allele frequencies; passed on to mutationMatrix().
matrix	A matrix, or a list of two (named "female" and "male")
rate	A numeric mutation rate, or a list of two (named "female" and "male")
rate2	A numeric mutation rate, or a list of two (named "female" and "male"). Required in the "stepwise" model; see mutationMatrix() for details.
range	A positive number, or a list of two (named "female" and "male"). Required in the "stepwise" model; see mutationMatrix() for details.
seed	An integer, or a list of two (named "female" and "male").
validate	A logical, by default TRUE.
mutmod	A mutationModel object.

Value

An object of class mutationModel. This is a list of two mutationMatrix objects, named "female" and "male", and the following attributes:

- sexEqual : TRUE if both genders have identical models, otherwise FALSE
- alwaysLumpable : TRUE if both genders have models that are lumpable for any allele subset, otherwise FALSE

Examples

```
# "Equal" model, same parameters for both genders
M1 = mutationModel("eq", alleles = 1:2, rate = 0.1)
M1

# Different mutation rates
M2 = mutationModel("eq", alleles = 1:2, rate = list(male = 0.1, female = 0.01))
M2

stopifnot(identical(M1$male, M1$female), identical(M2$male, M1$male))

# A custom mutation matrix:
mat = matrix(c(0,0,1,1), ncol = 2, dimnames = list(1:2, 1:2))
M3 = mutationModel(model = "custom", matrix = mat)

# Under the hood arguments are passed to `mutationMatrix()`.
# Alternatively, this can be done explicitly in the `model` argument
M4 = mutationModel(model = mutationMatrix("custom", matrix = mat))

stopifnot(identical(M3, M4))

# The latter strategy is needed e.g. in pedtools::marker(), which gives the
# user access to `model`, but not `matrix`.
```

pedmut *pedmut: Mutation Models for Pedigree Likelihood Computations*

Description

A collection of functions for modelling mutations in pedigrees with marker data, as used e.g. in likelihood computations with microsatellite data. Implemented models include proportional and stepwise models, as well as random models for experimental work, and custom models allowing the user to apply any valid mutation matrix. Allele lumping is done following the lumpability criteria of Kemeny and Snell (1976), ISBN:0387901922.

Index