Package 'forrel'

March 22, 2020

Type Package

Title Forensic Pedigree Analysis and Relatedness Inference

Version 1.0.1

Description Forensic applications of pedigree analysis, including likelihood ratios for relationship testing, general relatedness inference, marker simulation, and power analysis. General computation of exclusion powers is based on Egeland et al. (2014) <doi:10.1016/j.fsigen.2013.05.001>. Several functions deal specifically with family reunion cases, implementing and developing ideas from Kling et al. (2017) <doi:10.1016/j.fsigen.2017.08.006>. A novelty of 'forrel' is the ability to model background inbreeding in forensic pedigree computations. This can have significant impact in applications, as exemplified in Vigeland and Egeland (2019) <doi:10.1016/j.fsigss.2019.10.175>. 'forrel' is part of the ped suite, a collection of packages for pedigree analysis. In particular, 'forrel' imports 'pedtools' for creating and manipulating pedigrees and markers, 'pedprobr' for likelihood computations, and 'pedmut' for mutation modelling. Pedigree data may be created from scratch, or loaded from text files. Data import from the 'Familias' software (Egeland et al. (2000) <doi:10.1016/S0379-0738(00)00147-X>) is supported.

License GPL-3

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

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

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 3.1.0), pedtools (>= 0.9.3)

Imports pedprobr, maxLik, glue, pedmut,

Suggests testthat, ggplot2, poibin

RoxygenNote 7.1.0

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2020-03-22 08:50:05 UTC

R topics documented:

`exclusionPower` *Power of exclusion*

Description

Computes the power (of a single marker, or for a collection of markers) of excluding a claimed relationship, given the true relationship.

Usage

```
exclusionPower(
  claimPed,
  truePed,
  ids,
  markers = NULL,
  source = "claim",
  disableMutations = NA,
  exactMaxL = Inf,
  nsim = 1000,
```

```
  seed = NULL,
  alleles = NULL,
  afreq = NULL,
  knownGenotypes = NULL,
  Xchrom = FALSE,
  plot = FALSE,
  plotMarkers = NULL,
  verbose = TRUE
)
```

Arguments

`claimPed`	A ped object (or a list of such), describing the claimed relationship. If a list, the sets of ID labels must be disjoint, that is, all ID labels must be unique.
`truePed`	A ped object (or a list of such), describing the true relationship. ID labels must be consistent with `claimPed`.
`ids`	Individuals available for genotyping.
`markers`	A vector indicating the names or indices of markers attached to the source pedigree. If NULL (default), then all markers attached to the source pedigree are used. If `alleles` or `afreq` is non-NULL, then this parameter is ignored.
`source`	Either "claim" (default) or "true", deciding which pedigree is used as source for marker data.
`disableMutations`	This parameter determines how mutation models are treated. Possible values are as follows: • `NA` (the default): Mutations are disabled only for those markers whose known genotypes are compatible with both `claimPed` and `truePed`. This is determined by temporarily removing all mutation models and checking which markers have nonzero likelihood in both alternatives. • `TRUE`: Mutations are disabled for all markers. • `FALSE`: No action is done to disable mutations. • A vector containing the names or indices of those markers for which mutations should be disabled.
`exactMaxL`	A positive integer, or `Inf` (default). Exact EPs are calculated for markers whose number of alleles is less or equal to `exactMaxL`; remaining markers are handled using a simulation.
`nsim`	A positive integer; the number of simulations used for markers whose number of alleles exceeds `exactMaxL`.
`seed`	A numeric seed for the random number generator (optional).
`alleles, afreq, Xchrom`	If these are given, they are used (together with `knownGenotypes`) to create a marker object on the fly.
`knownGenotypes`	A list of triplets (a, b, c), indicating that individual a has genotype b/c. Ignored unless `alleles` or `afreq` is non-NULL.
`plot`	Either a logical or the character "plotOnly". If the latter, a plot is drawn, but no further computations are done.

plotMarkers	A vector of marker names or indices whose genotypes are to be included in the plot.
verbose	A logical.

Details

This function implements the formula for exclusion power as defined and discussed in (Egeland et al., 2014).

Value

If `plot = "plotOnly"`, the function returns NULL after producing the plot.

Otherwise, the function returns an `EPresult` object, which is essentially a list with the following entries:

- `EPperMarker`: A numeric vector containing the exclusion power of each marker. If the known genotypes of a marker are incompatible with the true pedigree, the corresponding entry is `NA`.
- `EPtotal`: The total exclusion power, computed as `1 -prod(1 -EPperMarker,na.rm = TRUE)`.
- `expectedMismatch`: The expected number of markers giving exclusion, computed as `sum(EPperMarker,na.rm = TRUE)`.
- `distribMismatch`: The probability distribution of the number of markers giving exclusion. This is given as a numeric vector of length `n+1`, where `n` is the number of nonzero elements of `EPperMarker`. The vector has names `0:n`.
- `time`: The total computation time.
- `params`: A list containing the (processed) parameters `ids`, `markers` and `disableMutations`.

Author(s)

Magnus Dehli Vigeland

References

T. Egeland, N. Pinto and M.D. Vigeland, *A general approach to power calculation for relationship testing.* Forensic Science International: Genetics 9 (2014): 186-190. doi: 10.1016/j.fsigen.2013.05.001

Examples

```
##########################################
### A standard case paternity case:
### Compute the power of exclusion when the claimed father is in fact
### unrelated to the child.
##########################################

# Claim: Individual 1 is the father of individual 3
claim = nuclearPed(nch = 1, sex = 2)

# Truth: 1 and 3 are unrelated
true = list(singleton(id = 1), singleton(id = 3, sex = 2))
```

```
# Attach 3 markers
m1 = marker(claim, alleles = 1:2)
m2 = marker(claim, alleles = 1:3)
m3 = marker(claim, alleles = 1:2, chrom = "X")
claim = setMarkers(claim, list(m1, m2, m3))

# Compute EP when father and child is available for genotyping
exclusionPower(claim, true, ids = c(1,3))

# Suppose child is genotyped
genotype(claim, marker = 1, id = 3) = c(1, 1)
genotype(claim, marker = 2, id = 3) = c(1, 1)
genotype(claim, marker = 3, id = 3) = c(1, 2)

# Compute EP when father is available
exclusionPower(claim, true, ids = 1)


##########################################
### Two females claim to be mother and daughter, but are in reality sisters.
### We compute the power of various markers to reject the claim.
##########################################

mother_daughter = nuclearPed(1, sex = 2)
sisters = relabel(nuclearPed(2, sex = c(2, 2)), c(101, 102, 2, 3))
ids = 2:3

# SNP with MAF = 0.1:
PE1 = exclusionPower(claimPed = mother_daughter, truePed = sisters,
                     ids = ids, alleles = 2, afreq = c(0.9, 0.1))

# Tetra-allelic marker with one major allele:
PE2 = exclusionPower(claimPed = mother_daughter, truePed = sisters,
                     ids = ids, alleles = 4, afreq = c(0.7, 0.1, 0.1, 0.1))

stopifnot(all.equal(c(PE1$EPtotal, PE2$EPtotal), c(0.00405, 0.03090)))

### How does the power change if the true pedigree is inbred?
sisters_LOOP = addParents(sisters, 101, father = 201, mother = 202)
sisters_LOOP = addParents(sisters_LOOP, 102, father = 201, mother = 203)


# SNP with MAF = 0.1:
PE3 = exclusionPower(claimPed = mother_daughter, truePed = sisters_LOOP,
                     ids = ids, alleles = 2, afreq = c(0.9, 0.1))

stopifnot(all.equal(PE3$EPtotal, 0.00765))
```

---

expectedLR *Expected likelihood ratio*

Description

This function computes the expected LR for a single marker, in a kinship test comparing two hypothesised relationships between a set of individuals. The true relationship may differ from both hypotheses. Some individuals may already be genotyped, while others are available for typing. The implementation uses `oneMarkerDistribution()` to find the joint genotype distribution for the available individuals, conditional on the known data, in each pedigree.

Usage

```
expectedLR(numeratorPed, denominatorPed, truePed = numeratorPed, ids, marker)
```

Arguments

`numeratorPed`	A ped object.
`denominatorPed`	A ped object.
`truePed`	A ped object.
`ids`	A vector of ID labels corresponding to untyped pedigree members. (These must be members of all three input pedigrees).
`marker`	either a marker object compatible with `numeratorPed`, or the name or index of a marker attached to `numeratorPed`.

Value

A positive number.

Examples

```
#---------
# Curious example showing that ELR may decrease
# by typing additional reference individuals
#---------

# Numerator ped
numPed = nuclearPed(father = "fa", mother = "mo", child = "ch")

# Denominator ped: fa, mo, ch are unrelated. Ugly hack!
denomPed = nuclearPed(father = "fa", mother = "mo", nch = 1)
denomPed = addChildren(denomPed, father = "ch", mother = "mo", nch = 1)

# Scenario 1: Only mother is typed; genotype 1/2
p = 0.9
m1 = marker(numPed, mo = 1:2, afreq = c("1" = p, "2" = 1-p))
expectedLR(numPed, denomPed, ids = "ch", marker = m1)

1/(8*p*(1-p)) + 1/2 # exact formula

# Scenario 2: Include father, with genotype 1/1
m2 = m1
```

```
genotype(m2, id = "fa") = c(1, 1)
expectedLR(numPed, denomPed, ids = "ch", marker = m2)

1/(8*p*(1-p)) + 1/(4*p^2) # exact formula
```

Familias2ped *Convert* Familias *objects to* ped *format*

Description

Familias is a widely used software for forensic kinship computations, which also features an accompanying R package (also called Familias). The function documented here converts pedigrees and marker data from the R version of Familias to pedtools::ped() format, used by forrel. This may be of interest for specialized computations not implemented in Familias, e.g. conditional simulations. Note: For importing ".fam" files created by (the stand-alone) Familias, see readFam().

Usage

```
Familias2ped(familiasped, datamatrix, loci, matchLoci = FALSE)

readFamiliasLoci(loci)
```

Arguments

familiasped	A FamiliasPedigree object or a list of such.
datamatrix	A data frame with two columns per marker (one for each allele) and one row per individual.
loci	A FamiliasLocus object or a list of such.
matchLoci	A logical. If TRUE, the column names of datamatrix must be found either within names(loci) or within the name entries of loci. The column names of datamatrix are assumed to come in pairs with suffixes ".1" and ".2", e.g. "TH01.1", "TH01.2", etc. If FALSE (the default) it is assumed that the loci correspond to the (pairs of) columns in datamatrix sequentially.

Details

The Familias program represents pedigrees and marker data in a way that differs from the ped format in several ways, mostly because of the latter's stricter definition of a *pedigree*. A ped object always represent a connected pedigree, and each member must have either 0 or 2 parents. None of this is required by FamiliasPedigree objects. The conversion function Familias2ped takes care of all potential differences: It converts each Familias pedigree into a list of connected ped objects, adding missing parents where needed.

Value

A ped object, or a list of such.

Author(s)

Magnus Dehli Vigeland, Thore Egeland

References

Familias is freely available from http://familias.name.

See Also

readFam()

Examples

```
famPed = structure(
  list(id = c('mother', 'daughter', 'AF'),
       findex = c(0, 3, 0),
       mindex = c(0, 1, 0),
       sex = c('female', 'female', 'male')),
  class = "FamiliasPedigree")

datamatrix = data.frame(
  M1.1 = c(NA, 8, NA),
  M1.2 = c(NA, 9.3, NA),
  row.names = famPed$id)

famLoc = structure(
  list(locusname = "M1",
       alleles = c("8" = 0.2, "9" = 0.5, "9.3" = 0.3)),
  class = "FamiliasLocus")

Familias2ped(famPed, datamatrix, loci = famLoc, matchLoci = TRUE)
```

---

forrel *forrel: Forensic Pedigree Analysis and Relatedness Inference*

Description

Forensic applications of pedigree analysis, including likelihood ratios for relationship testing, general relatedness inference, marker simulation, and power analysis. General computation of exclusion powers is based on Egeland et al. (2014) doi: 10.1016/j.fsigen.2013.05.001. Several functions deal specifically with family reunion cases, implementing and developing ideas from Kling et al. (2017) doi: 10.1016/j.fsigen.2017.08.006. A novelty of 'forrel' is the ability to model background inbreeding in forensic pedigree computations. This can have significant impact in applications, as exemplified in Vigeland and Egeland (2019) doi: 10.1016/j.fsigss.2019.10.175. 'forrel' is part of the ped suite, a collection of packages for pedigree analysis. In particular, 'forrel' imports 'pedtools' for creating and manipulating pedigrees and markers, 'pedprobr' for likelihood computations,

and 'pedmut' for mutation modelling. Pedigree data may be created from scratch, or loaded from text files. Data import from the 'Familias' software (Egeland et al. (2000) doi: 10.1016/S0379-0738(00)00147X) is supported.

`IBDestimate` *Relatedness estimation*

Description

Estimate the pairwise IBD coefficients $(\kappa_0, \kappa_1, \kappa_2)$ for specified pairs of pedigree members, using maximum likelihood methods. The optimisation machinery is imported from the `maxLik` package.

Usage

```
IBDestimate(x, ids = NULL, markers = NULL, start = c(0.99, 0.001), tol = 1e-07)
```

Arguments

`x`	A ped object or a list of such.
`ids`	Either a vector with ID labels, or a data frame/matrix with two columns, where each row contains the ID labels of two individuals. The entries are coerced to characters, and must match uniquely against the ID labels of `x`. If `ids` is a vector, it is converted to a matrix containing all pairs. By default, all individuals of `x` are included.
`markers`	A numeric indicating which marker(s) to include. If NULL (default), all markers are used.
`start`	Numeric of length 2, indicating the initial value of (κ_0, κ_2) in the optimisation (passed on to `maxLik`).
`tol`	A single numeric: the optimising tolerance value; passed on to `maxLik()`).

Details

This function optimises the log-likelihood function first described in (Thompson, 1975). Optimisation is done in the (κ_0, κ_2)-plane and restricted to the probability triangle defined by

$$\kappa_0 \geq 0, \kappa_2 \geq 0, \kappa_0 + \kappa_2 \leq 1.$$

Value

A data frame with 6 columns: `ID1`, `ID2`, `N` (the number of markers with no missing alleles), `k0`, `k1` and `k2`.

Author(s)

Magnus Dehli Vigeland

References

- E. A. Thompson (1975). *The estimation of pairwise relationships.* Annals of Human Genetics 39.
- E. A. Thompson (2000). *Statistical Inference from Genetic Data on Pedigrees.* NSF-CBMS Regional Conference Series in Probability and Statistics. Volume 6.

See Also

`maxLik::maxLik()`, `showInTriangle()`

Examples

```
### Example 1: Siblings
x = nuclearPed(children = c("sib1", "sib2"))

# Simluate 200 equifrequent SNPs
x = markerSim(x, N = 200, alleles = 1:2, verbose = FALSE)

# Estimate IBD coefficients (exact = (0.25, 0.5, 0.25))
est = IBDestimate(x, ids = c("sib1", "sib2"))
showInTriangle(est, labels = TRUE)

### Example 2: Unrelated singletons
y = list(singleton(1), singleton(2))
y = markerSim(y, N = 200, alleles = 1:2, verbose = FALSE)

IBDestimate(y, ids = 1:2)
```

IBDtriangle *IBD triangle plot*

Description

The IBD triangle is typically used to visualize the pairwise relatedness of non-inbred individuals. Various annotations are available, including points marking the most common relationships, contour lines for the kinship coefficients, and shading of the unattainable region.

Usage

```
IBDtriangle(
  relationships = c("UN", "PO", "MZ", "S", "H,U,G", "FC"),
  kinshipLines = numeric(),
  shading = "lightgray",
  pch = 16,
  cex_points = 1.2,
```

```
  cex_text = 1.2,
  axes = FALSE,
  xlim = c(0, 1),
  ylim = c(0, 1),
  xlab = expression(kappa[0]),
  ylab = expression(kappa[2]),
  cex_lab = cex_text,
  mar = c(3.1, 3.1, 1, 1),
  keep.par = TRUE
)
```

Arguments

`relationships`	A character vector indicating relationships points to be included in the plot. See Details for a list of valid entries.
`kinshipLines`	A numeric vector (see Details).
`shading`	The shading colour for the unattainable region.
`pch`	Symbol used for the relationship points (see `par()`).
`cex_points`	A number controlling the symbol size for the relationship points.
`cex_text`	A number controlling the font size for the relationship labels.
`axes`	A logical: Draw surrounding axis box?
`xlim, ylim, mar`	Graphical parameters; see `par()`.
`xlab, ylab`	Axis labels
`cex_lab`	A number controlling the font size for the axis labels.
`keep.par`	A logical. If TRUE, the graphical parameters are not reset after plotting, which may be useful for adding additional annotation.

Details

For any pair of non-inbred individuals A and B, their genetic relationship can be summarized by the IBD coefficients $(\kappa_0, \kappa_1, \kappa_2)$, where κ_i = P(A and B share i alleles IBD at random autosomal locus). Since $\kappa_0 + \kappa_1 + \kappa_2 = 1$, any relationship corresponds to a point in the triangle in the (κ_0, κ_2)-plane defined by $\kappa_0 \geq 0, \kappa_2 \geq 0, \kappa_0 + \kappa_2 \leq 1$. The choice of κ_0 and κ_2 as the axis variables is done for reasons of symmetry and is not significant (other authors have used different views of the triangle).

As shown in (Thompson, 1976) points in the subset of the triangle defined by $4\kappa_0\kappa_2 > \kappa_1^2$ are unattainable for pairwise relationships. By default this region in shaded in a 'lightgray' colour.

The IBD coefficients are linearly related to the kinship coefficient ϕ by the formula

$$\phi = 0.25\kappa_1 + 0.5\kappa_2.$$

By indicating values for ϕ in the `kinshipLines` argument, the corresponding contour lines are shown as dashed lines in the triangle plot.

The following abbreviations are valid entries in the `relationships` argument:

- UN = unrelated

- PO = parent/offspring
- MZ = monozygotic twins
- S = full siblings
- H,U,G = half sibling/avuncular (**u**ncle)/grandparent
- FC = first cousins
- SC = second cousins
- DFC = double first cousins
- Q = quadruple first half cousins

Value

None

Author(s)

Magnus Dehli Vigeland

References

- E. A. Thompson (1975). *The estimation of pairwise relationships.* Annals of Human Genetics 39.
- E. A. Thompson (1976). *A restriction on the space of genetic relationships.* Annals of Human Genetics 40.

See Also

IBDestimate()

Examples

```
opar = par(no.readonly = TRUE) # store graphical parameters

IBDtriangle()
IBDtriangle(kinshipLines = c(0.25, 0.125), shading = NULL, cex_text = 0.8)

par(opar) # reset graphical parameters
```

kinshipLR *Likelihood ratios for kinship testing*

Description

This function computes likelihood ratios (LRs) for a given a list of pedigrees with attached markers. The user must indicate which of the pedigrees is the 'reference', which will be used in the denominator in each LR.

Usage

```
kinshipLR(x, ref, markers)
```

Arguments

x	A list of pedigree alternatives. Each alternative should be either a single ped object or a list of such.
ref	A single integer, indicating the index (in the list x) of the reference alternative. This is used in the denominator of each LR.
markers	A vector of integers, indexing which markers should be included. If NULL (the default) all markers are used.

Value

A LRresultobject, which is essentially a list with entries

- LRtotal : Total likelihood ratios
- LRperMarker : Likelihood ratios for each marker
- likelihoodsPerMarker : Likelihoods for each marker
- time user system and elapsed time

Author(s)

Magnus Dehli Vigeland and Thore Egeland

See Also

LRpower(), pedtools::transferMarkers()

Examples

```
# Simulate 5 markers for a pair of full sibs
set.seed(123)
sibs = nuclearPed(children = c("A", "B"))
sibs = simpleSim(sibs, N = 5, alleles = 1:4, ids = c("A", "B"))
```

```
# Create two alternative hypotheses and transfer the simulated genotypes to them
halfsibs = relabel(halfSibPed(), old = 4:5, new = c("A", "B"))
halfsibs = transferMarkers(sibs, halfsibs)

unrel = list(singleton("A"), singleton("B"))
unrel = transferMarkers(sibs, unrel)

# Compute LR with 'unrelated' as reference
res = kinshipLR(list(sibs, halfsibs, unrel), ref = 3)
res

# Detailed results
res$LRperMarker
res$likelihoodsPerMarker
```

LRpower *Power simulation for kinship LR*

Description

This function uses simulations to estimate the likelihood ratio (LR) distribution in a given kinship testing scenario. In the most general setting, three pedigrees are involved: the two pedigrees being compared, and the true relationship (which may differ from the other two). A subset of individuals are available for genotyping. Some individuals may already be genotyped; all simulations are then conditional on these.

Usage

```
LRpower(
  numeratorPed,
  denominatorPed,
  truePed = numeratorPed,
  ids,
  markers = NULL,
  source = "true",
  nsim = 1,
  threshold = NULL,
  disableMutations = NA,
  alleles = NULL,
  afreq = NULL,
  Xchrom = FALSE,
  knownGenotypes = NULL,
  plot = FALSE,
  plotMarkers = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

numeratorPed, denominatorPed	ped objects (or lists of such), describing the two relationships under comparison.
truePed	A ped object (or a list of such), describing the true relationship. By default equal to numeratorPed.
ids	Individuals available for genotyping.
markers	A vector indicating the names or indices of markers attached to the source pedigree. If NULL (default), then all markers attached to the source pedigree are used. If alleles or afreq is non-NULL, then this parameter is ignored.
source	Either "true" (default), "numerator" or "denominator", indicating which pedigree is used as source for marker data.
nsim	A positive integer: the number of simulations.
threshold	A numeric vector with one or more positive numbers used as LR thresholds.
disableMutations	Not implemented yet.
alleles, afreq, Xchrom	If these are given, they are used (together with knownGenotypes) to create a marker object on the fly.
knownGenotypes	A list of triplets (a, b, c), indicating that individual a has genotype b/c. Ignored unless alleles or afreq is non-NULL.
plot	Either a logical or the character "plotOnly". If the latter, a plot is drawn, but no further computations are done.
plotMarkers	A vector of marker names or indices whose genotypes are to be included in the plot.
seed	A numeric seed for the random number generator (optional).
verbose	A logical.

Value

A LRpowerResult object, which is essentially a list with the following entries:

- LRperSim: A numeric vector of length nsim containing the total LR for each simulation.
- meanLRperMarker: The mean LR per marker, over all simulations.
- meanLR: The mean total LR over all simulations.
- meanLogLR: The mean total log10(LR) over all simulations.
- IP: A named numeric of the same length as threshold. For each element of threshold, the fraction of simulations resulting in a LR exceeding the given number.
- time: The total computation time.
- params: A list containing the input parameters missing, markers, nsim, threshold and disableMutations

Examples

```
# Paternity LR of siblings
claim = nuclearPed(fa = "A", mo = "NN", children = "B")
unrel = list(singleton("A"), singleton("B"))
truth = nuclearPed(children = c("A", "B"))

# Simulation parameters
nsim = 10   # increase!
thresh = 1
ids = c("A", "B")

# Simulation 1:
als = 1:5
afr = runif(5)
afr = afr/sum(afr)

pow1 = LRpower(claim, unrel, truth, ids = ids, nsim = nsim, threshold = thresh,
               alleles = als, afreq = afr, seed = 123)
pow1

# Simulation 2: Same, but using an attached marker
m = marker(truth, alleles = als, afreq = afr)
truth = setMarkers(truth, m)

pow2 = LRpower(claim, unrel, truth, ids = ids, nsim = nsim, threshold = thresh,
               markers = 1, seed = 123)
stopifnot(identical(pow1$LRperSim, pow2$LRperSim))


# Founder inbreeding in true pedigree
founderInbreeding(truth, founders(truth)) = 0.5
truth
pow3 = LRpower(claim, unrel, truth, ids = ids, nsim = nsim, threshold = thresh,
               markers = 1, seed = 123, plot = TRUE)
pow3
```

`markerSim` *Marker simulation*

Description

Simulates marker genotypes conditional on the pedigree structure and known genotypes. Note: This function simulates independent realisations at a single locus. Equivalently, it can be thought of as independent simulations of identical, unlinked markers. For simulations of a *set* of markers, see `profileSim()`.

Usage

```
markerSim(
  x,
  N = 1,
  ids = NULL,
  alleles = NULL,
  afreq = NULL,
  mutmod = NULL,
  rate = NULL,
  partialmarker = NULL,
  loopBreakers = NULL,
  eliminate = 0,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	A ped object or a list of such.
N	A positive integer: the number of (independent) markers to be simulated.
ids	A vector containing ID labels of those pedigree members whose genotypes should be simulated. By default, all individuals are included.
alleles	A vector containing the alleles for the marker to be simulation. If a single integer is given, this is interpreted as the number of alleles, and the actual alleles as 1:alleles. Must be NULL if partialmarker is non-NULL.
afreq	A vector containing the population frequencies for the marker alleles. Must be NULL if partialmarker is non-NULL.
mutmod, rate	Arguments specifying a mutation model, passed on to pedtools::marker() (see there for explanations).
partialmarker	Either NULL (resulting in unconditional simulation), a marker object (on which the simulation should be conditioned) or the name (or index) of a marker attached to x.
loopBreakers	A numeric containing IDs of individuals to be used as loop breakers. Relevant only if the pedigree has loops, and only if partialmarker is non-NULL. See pedtools::breakLoops().
eliminate	A non-negative integer, indicating the number of iterations in the internal genotype-compatibility algorithm. Positive values can save time if partialmarker is non-NULL and the number of alleles is large.
seed	NULL, or a numeric seed for the random number generator.
verbose	A logical.

Details

This implements (with various time savers) the algorithm used in SLINK of the LINKAGE/FASTLINK suite. If partialmarker is NULL, genotypes are simulated by simple gene dropping, using simpleSim().

Value

A ped object equal to x except its MARKERS entry, which consists of the N simulated markers.

Author(s)

Magnus Dehli Vigeland

References

G. M. Lathrop, J.-M. Lalouel, C. Julier, and J. Ott, *Strategies for Multilocus Analysis in Humans*, PNAS 81(1984), pp. 3443-3446.

See Also

profileSim(), simpleSim()

Examples

```
x = nuclearPed(2)

# Unconditional simulation
markerSim(x, N = 2, alleles = 1:3)

# Conditional on one child being homozygous 1/1
x = setMarkers(x, marker(x, '3' = 1, alleles = 1:3))
markerSim(x, N = 2, partialmarker = 1)
markerSim(x, N = 1, ids = 4, partialmarker = 1, verbose = FALSE)
```

---

missingPersonEP *Exclusion power for missing person cases*

Description

This is a wrapper of exclusionPower() for the special case of a reference family with a single missing member. Some reference members should already be genotyped. The function computes the power to exclude an unrelated individual, i.e. the probability of observing (in a truly unrelated individual) a genotype incompatible with the reference.

Usage

```
missingPersonEP(
  reference,
  missing,
  markers = NULL,
  disableMutations = NA,
  verbose = TRUE
)
```

Arguments

reference	A ped object with attached markers.
missing	The ID label of the missing pedigree member.
markers	A vector indicating the names or indices of markers attached to the source pedigree. If NULL (default), then all markers attached to the source pedigree are used. If alleles or afreq is non-NULL, then this parameter is ignored.
disableMutations	This parameter determines how mutation models are treated. Possible values are as follows: • NA (the default): Mutations are disabled only for those markers whose known genotypes are consistent with reference. This is determined by temporarily removing all mutation models and checking which markers have nonzero likelihood. • TRUE: Mutations are disabled for all markers. This will result in an error if any markers are inconsistent with reference. • FALSE: No action is done to disable mutations. • A vector containing the names or indices of those markers for which mutations should be disabled.
verbose	A logical.

Value

The EPresult object returned by exclusionPower().

Examples

```
# Four siblings; the fourth is missing
x = nuclearPed(4)

# Remaining sibs typed with 4 triallelic markers
x = markerSim(x, N = 4, ids = 3:5, alleles = 1:3, seed = 577, verbose = FALSE)

# Add marker with inconsistency in reference genotypes
# (this should be ignored by `missingPersonEP()`)
badMarker = marker(x, `3` = 1, `4` = 2, `5` = 3)
x = addMarkers(x, badMarker)

# Compute exclusion power statistics
missingPersonEP(x, missing = 6)

# With marker names:
name(x, 1:5) = paste0("M", 1:5)
missingPersonEP(x, missing = 6)
```

`missingPersonIP` *Inclusion power for missing person cases*

Description

Inclusion power for missing person cases

Usage

```
missingPersonIP(
  reference,
  missing,
  markers,
  nsim = 1,
  threshold = NULL,
  disableMutations = NA,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`reference`	A ped object with attached markers.
`missing`	The ID label of the missing pedigree member.
`markers`	A vector indicating the names or indices of markers attached to the source pedigree. If NULL (default), then all markers attached to the source pedigree are used. If `alleles` or `afreq` is non-NULL, then this parameter is ignored.
`nsim`	A positive integer: the number of simulations
`threshold`	A numeric vector with one or more positive numbers used as the likelihood ratio thresholds for inclusion
`disableMutations`	This parameter determines how mutation models are treated. Possible values are as follows: • `NA` (the default): Mutations are disabled only for those markers whose known genotypes are consistent with `reference`. This is determined by temporarily removing all mutation models and checking which markers have nonzero likelihood. • `TRUE`: Mutations are disabled for all markers. This will result in an error if any markers are inconsistent with `reference`. • `FALSE`: No action is done to disable mutations. • A vector containing the names or indices of those markers for which mutations should be disabled.
`seed`	A numeric seed for the random number generator (optional)
`verbose`	A logical.

Value

A mpIP object, which is essentially a list with the following entries:

- LRperSim: A numeric vector of length nsim containing the total LR for each simulation.
- meanLRperMarker: The mean LR per marker, over all simulations.
- meanLR: The mean total LR over all simulations.
- meanLogLR: The mean total log10(LR) over all simulations.
- IP: A named numeric of the same length as threshold. For each element of threshold, the fraction of simulations resulting in a LR exceeding the given number.
- time: The total computation time.
- params: A list containing the input parameters missing, markers, nsim, threshold and disableMutations

Examples

```
# Four siblings; the fourth is missing
x = nuclearPed(4)

# Remaining sibs typed with 5 triallelic markers
x = markerSim(x, N = 5, ids = 3:5, alleles = 1:3, seed = 123, verbose = FALSE)

# Compute exclusion power statistics
missingPersonIP(x, missing = 6, nsim = 5, threshold = c(10, 100))

# Compare with genotypes
x
```

---

missingPersonPlot *Missing person plot*

Description

Visualises the competing hypotheses of a family reunion case. A plot with two panels is generated. The left panel shows a pedigree in which the *person of interest* (POI) is identical to the *missing person* (MP). The right panel shows the situation where these two are unrelated. See Details for further explanations.

Usage

```
missingPersonPlot(
  reference,
  missing,
  id.labels = labels(reference),
  MP.label = "MP",
```

```
  POI.label = "POI",
  marker = NULL,
  shaded = "typed",
  POI.col = "red",
  POI.shaded = FALSE,
  POI.height = 8,
  width = 4,
  newdev = TRUE,
  frametitles = c(expression(H[1] * ": POI is MP"), expression(H[2] *
    ": POI unrelated")),
  ...
)
```

Arguments

reference	A pedtools::ped() object.
missing	The ID label of the missing pedigree member.
id.labels	A character vector with labels for the pedigree members. See pedtools::plot.ped().
MP.label	The label of the missing member. Default: "MP".
POI.label	The label of the person of interest. Default: "POI".
marker	Optional vector of marker indices to be included in the plot.
shaded	A vector of ID labels indicating who should appear with shaded symbols. By default, all typed members.
POI.col	The plot colour of POI. Default: red.
POI.shaded	A logical: If TRUE (default), the POI is plotted with a shaded symbol.
POI.height	A numeric controlling the vertical placement of the POI singleton (in the right panel).
width	A positive number controlling the width of the plot. More specifically this number is the relative width of the reference pedigree, compared to a singleton. Default: 4.
newdev	A logical: If TRUE the plot is created in a new plot window.
frametitles	A character of length 2, with subtitles for the two frames.
...	Extra parameters passed on to pedtools::plotPedList().

Details

A standard family reunification case involves the following ingredients:

- A reference family in which a single member ("MP") is missing.
- Some of the family members have been genotyped
- A person of interest ("POI") is to be matched against the reference family

After genotyping of POI, the genetic evidence is typically assessed by computing the likelihood ratio of the following hypotheses:

- H1: POI is MP

- H2: POI is unrelated to the family

The goal of this function is to illustrate the above hypotheses, using labels, colours and shading to visualise the different aspects of the situation.

This function cannot handle cases with more complicated hypotheses (e.g. multiple missing persons, or where H2 specifies a different relationship). However, as it is basically a wrapper of `pedtools::plotPedList()`, an interested user should be able to extend the source code to such cases without too much trouble.

Value

None

Examples

```
x = nuclearPed(father = "fa", mother = "mo", children = c("b1", "b2"))

# Default plot
missingPersonPlot(x, missing = "b2")

# A bit nicer using various options
missingPersonPlot(x, missing = "b2", MP.label = "Missing", id.label = NULL,
                  shaded = "b1", POI.shaded = TRUE,
                  width = 2,      # adjust internal spacing (see above)
                  dev.width = 7,  # device width (see ?plotPedList())
                  dev.height = 3, # device height (see ?plotPedList())
                  fmar = 0.02,    # adjust frame margin (see ?plotPedList())
                  cex = 1.5,      # larger symbols and label font (see ?par())
                  cex.main = 1.3  # larger frame titles (see ?par())
                  )
```

`MPPsims` *Missing person power simulations*

Description

Estimate the exclusion/inclusion power for various selections of available individuals.

Usage

```
MPPsims(
  reference,
  missing = "MP",
  selections,
  ep = TRUE,
  ip = TRUE,
  addBaseline = TRUE,
  nProfiles = 1,
```

```
  lrSims = 1,
  thresholdIP = NULL,
  disableMutations = NA,
  numCores = NA,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

reference	A connected ped object, or a list of pedigrees. In the latter case, the list must have the same length as selections.
missing	The ID label of the missing pedigree member.
selections	A list of pedigree member subsets. In the special case that all subsets consist of a single individual, selections can be given as a simple vector.
ep	A logical: Estimate the exclusion power? (Default: TRUE)
ip	A logical: Estimate the inclusion power? (Default: TRUE)
addBaseline	A logical. If TRUE (default) an *empty* selection, named "Baseline", is added as the first element of selection.
nProfiles	The number of profile simulations for each selection.
lrSims, thresholdIP	Parameters passed onto missingPersonIP().
disableMutations	This parameter determines how mutation models are treated. Possible values are as follows: • NA (the default): Mutations are disabled only for those markers whose known genotypes are consistent with reference. This is determined by temporarily removing all mutation models and checking which markers have nonzero likelihood. • TRUE: Mutations are disabled for all markers. This will result in an error if any markers are inconsistent with reference. • FALSE: No action is done to disable mutations. • A vector containing the names or indices of those markers for which mutations should be disabled.
numCores	The number of cores used in the parallelisation setup.
seed	A numeric seed for the random number generator (optional)
verbose	A logical.

Value

An object of class "MPPsim", which is basically a list with one entry for each element of selections. Each entry has elements ep and ip, each of which is a list of length nProfiles.

The output object has various attributes reflecting the input. Note that reference and selection may differ slightly from the original input, since they may be modified during the function run. (For instance, a "Baseline" entry is added to selection if addBaseline is TRUE.) The crucial point is that the output attributes correspond exactly to the output data.

- reference (always a list, of the same length as the selections attribute
- selections
- nProfiles,lrSims,thresholdIP,seed (as in the input)
- totalTime (the total time used)

Examples

```
x = nuclearPed(fa = "Gf", mo = "Gm", children = c("Uncle", "Mother"), sex = 1:2)
x = addChildren(x, fa = "Father", mo = "Mother", nch = 3, sex = c(1,2,1),
                id = c("S1", "S2", "MP"))
x = addSon(x, "Father", id = "HS")

# Brother S1 is already genotyped with a marker with 5 alleles
m = marker(x, S1 = 1:2, alleles = 1:4)
x = setMarkers(x, m)

# Alternatives for additional genotyping
sel = list("Father", "S2", "HS", c("Gm", "Uncle"))

plot(x, marker = 1, shaded = sel)

# Simulate
simData = MPPsims(x, selections = sel, nProfiles = 2, lrSims = 2, numCores = 1)

# Power plot
powerPlot(simData, type = 3)


### With  mutations
# Create inconsistent marker
m2 = m
genotype(m2, "Father") = 3
x = setMarkers(x, list(m, m2))

# Set mutation models for both
mutmod(x, 1:2) = list("equal", rate = 0.1)

# By default mutations are disabled for consistent markers
MPPsims(x, selections = "Father", addBaseline = FALSE, seed = 123)

# Don't disable anything
MPPsims(x, selections = "Father", addBaseline = FALSE, seed = 123,
        disableMutations = FALSE)


# Disable all mutation models. SHOULD GIVE ERROR FOR SECOND MARKER
MPPsims(x, selections = "Father", addBaseline = FALSE, seed = 123,
        disableMutations = TRUE)


# Effect of variable number of alleles
y = nuclearPed(father = "fa", child = "MP")
```

```
mlist = lapply(2:5, function(k) marker(y, alleles = 1:k))
y = setMarkers(y, mlist)
peds = lapply(1:nMarkers(y), function(i) selectMarkers(y, i))
sel = rep("fa", 4)
names(sel) = paste(2:5, "alleles")
pows = MPPsims(peds, selections = sel, addBaseline = FALSE, lrSims = 10)
powerPlot(pows, type = 3)
```

`powerPlot` *Exclusion/inclusion power plots*

Description

This function offers four different visualisations of exclusion/inclusion powers, particularly for missing person cases. Output from `MPPsims()` may be fed directly as input to this function. The actual plotting is done with `ggplot2`.

Usage

```
powerPlot(
  ep,
  ip,
  type = 1,
  majorpoints = TRUE,
  minorpoints = TRUE,
  ellipse = FALSE,
  col = NULL,
  labs = NULL,
  alpha = 1,
  shape = "circle",
  size = 2,
  hline = NULL,
  vline = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`ep, ip`	Lists of equal length, with outputs from one or more runs of `missingPersonEP()` and `missingPersonIP()` respectively. Alternatively, ep can be a single output from `MPPsims()`, in which case `ip` should be NULL. See Examples.
`type`	Plot type; either 1, 2, 3 or 4.

majorpoints	A logical indicating whether "major" points should be drawn (see Details).
minorpoints	A logical indicating whether "minor" points should be drawn (see Details).
ellipse	A logical. If TRUE, data ellipsis are drawn for each group containing more than 1 element. NB: This fails with a warning if all points in a group fall on a line.
col	A colour vector, recycle to match the top level length of ep.
labs	A character of the same length as ep. If NULL, the names of ep are used, if present.
alpha	Transparency for minor points (see Details).
shape	Either "circle", "square", "diamond", "triangleUp" or "triangleDown", determining the shapes of both minor and major points.
size	Point size.
hline, vline	Single numbers indicating positions for horizontal/vertical "threshold" lines. If NULL (default), no lines are drawn.
xlim, ylim	Axis limits; automatically chosen if NULL.
xlab, ylab	Axis labels; automatically chosen if NULL.

Details

The plot types are as follows:

type = 1: x = Exclusion power; y = Inclusion power

type = 2: x = Exclusion odds ratio; y = Inclusion odds ratio

type = 3: x = Expected number of exclusions; y = average log(LR)

type = 4: x = Exclusion power; y = average LR

In the most general case ep (and similarly for ip) can be a list of lists of EPresult objects. We refer to the inner lists as "groups". A group may consist of a single output, or several (typically many simulations of the same situation). Points within the same group are always drawn with the same colour and shape.

When plotting several groups, two sets of points are drawn by default:

- Major points: Group means.
- Minor points: Individual points in groups with more than one element.

The parameters majorpoints and minorpoints control which of the above points are included.

Value

A ggplot2 plot object.

See Also

MPPsims(), missingPersonEP(), missingPersonEP()

Examples

```
ref = nuclearPed(father = "fa", mother = "mo", child = "MP")
ref = setMarkers(ref, marker(ref, alleles = 1:5))

# Alternatives for genotyping
sel = list("fa", c("fa", "mo"))

# Simulate power for each selection
simData = MPPsims(ref, selections = sel, nProfiles = 3, lrSims = 5,
                  thresholdIP = 2, seed = 123, numCores = 1)

# Power plot 1: EP vs IP
powerPlot(simData, type = 1)
powerPlot(simData, type = 1, minorpoints = FALSE, hline = 0.8)



# Change shape, and modify legend order
powerPlot(simData[3:1], type = 1, shape = c("ci", "sq", "di"))

# Zoom in, and add threshold lines
powerPlot(simData, type = 1, xlim = c(0.4, 1), ylim = c(0.4, 1),
          hline = 0.8, vline = 0.8)

# Power plot 3: Expected number of exclusions vs E[log LR]
powerPlot(simData, type = 3)

# With horizontal/vertical lines
powerPlot(simData, type = 3, hline = log10(2), vline = 1)

# Plot 4: Illustrating the general inequality ELR > 1/(1-EP)
powerPlot(simData, type = 4)
```

`profileSim` *Simulation of complete DNA profiles*

Description

Simulation of DNA profiles for specified pedigree members. Some pedigree members may already be genotyped; in that case the simulation is conditional on these. The main work of this function is done by `markerSim()`.

Usage

```
profileSim(x, N = 1, ids = NULL, markers = NULL, seed = NULL, ...)
```

Arguments

x	A ped object or a list of such.
N	The number of complete simulations to be performed.
ids	A character (or coercible to character) with ID labels indicating whose genotypes should be simulated.
markers	A list of marker objects, or a vector containing names or indices referring to markers attached to x. By default (markers = NULL) all attached markers are used. The simulations will be conditional on the locus attributes (allele frequencies, mutation models a.s.o.) and any existing genotypes in the indicated markers.
seed	NULL, or a numeric seed for the random number generator.
...	Further arguments passed on to markerSim().

Value

A list of N objects similar to x, but with simulated genotypes. Any previously attached markers are replaced by the simulated profiles. If the indicated markers contained genotypes for some pedigree members, these are still present in the simulated profiles.

Examples

```
# Example with two brothers
x = nuclearPed(children = c("B1", "B2"))

# Attach two markers; one brother is already genotyped
m1 = marker(x, B1 = 1:2, alleles = 1:3)
m2 = marker(x, B1 = 1, alleles = 1:4, afreq = (1:4)/10, chrom = "X")
x = setMarkers(x, list(m1, m2))

# Simulate 3 profiles of B2 conditional on the above
profileSim(x, N = 3, ids = "B2")
```

---

readFam *Read* Familias *.fam files*

Description

This function parses the content of a Familias-formatted ".fam" file, and converts it into suitable ped objects. This function does not depend on the Familias R package.

Usage

```
readFam(famfile, useDVI = NA, Xchrom = FALSE, verbose = TRUE)
```

Arguments

famfile	Path to a ".fam" file.
useDVI	A logical, indicating if the DVI section of the fam file should be identified and parsed. If NA (the default), the DVI section is included if it is present in the input file.
Xchrom	A logical. If TRUE, the chrom attribute of all markers will be set to "X". (Default = FALSE.)
verbose	A logical; if TRUE, various information is written to the screen during the parsing process.

Value

If the .fam file only contains a database, the output is a list of information (name, alleles, frequencies) about each locus. This list can be used as locusAttributes in e.g. setMarkers().

If the .fam file describes pedigree data, the output is a ped object or a list of such.

If useDVI = TRUE, then the families described under Reference Families are parsed and converted to ped objects. Each family generally describes multiple pedigrees, so the output gets another layer in this case.

---

showInTriangle *Add points to the IBD triangle*

Description

Utility function for plotting points in the IBD triangle.

Usage

```
showInTriangle(
  k0,
  k2 = NULL,
  new = TRUE,
  col = "blue",
  cex = 1,
  pch = 4,
  lwd = 2,
  labels = FALSE,
  col_labels = col,
  cex_labels = 0.8,
  pos = 1,
  adj = NULL,
  ...
)
```

Arguments

k0, k2	Numerical vectors giving coordinates for points to be plotted in the IBD triangle. Alternatively, k0 may be a data.frame containing columns named k0 and k2.
new	Logical indicating if a new IBDtriangle should be drawn.
col, cex, pch, lwd	Parameters passed onto points().
labels	A character of same length as k0 and k2, or a single logical TRUE or FALSE. If TRUE, and k0 is a data.frame, labels will be created by pasting columns "ID1" and "ID2", if these are present. By default, no labels are plotted.
col_labels, cex_labels, pos, adj	Parameters passed onto text() (if labels is non-NULL).
...	Plot arguments passed on to IBDtriangle().

Value

None

Author(s)

Magnus Dehli Vigeland

See Also

IBDestimate()

Examples

```
showInTriangle(k0 = 3/8, k2 = 1/8, label = "3/4 siblings", pos = 1)
```

---

simpleSim *Unconditional marker simulation*

Description

Unconditional simulation of unlinked markers

Usage

```
simpleSim(
  x,
  N,
  alleles,
  afreq,
  ids,
  Xchrom = FALSE,
```

```
  mutmod = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	a ped object
N	a positive integer: the number of markers to be simulated
alleles	a vector containing the allele names. If missing, the alleles are taken to be seq_along(afreq).
afreq	a vector of length 2 containing the population frequencies for the alleles. If missing, the alleles are assumed equifrequent.
ids	a vector containing ID labels of those pedigree members whose genotypes should be simulated.
Xchrom	a logical: X linked markers or not?
mutmod	a pedmut::mutationModel() object, i.e., list of mutation matrices named 'female' and 'male'.
seed	NULL, or a numeric seed for the random number generator.
verbose	a logical.

Details

This simulation is done by distributing alleles randomly to all founders, followed by unconditional gene dropping down throughout the pedigree (i.e. for each non-founder a random allele is selected from each of the parents). Finally the genotypes of any individuals not included in ids are removed.

Value

a ped object equal to x in all respects except its MARKERS entry, which consists of the N simulated markers.

Author(s)

Magnus Dehli Vigeland

See Also

markerSim()

Examples

```
library(pedtools)
x = nuclearPed(1)
simpleSim(x, N = 3, afreq = c(0.5, 0.5))

y = cousinPed(1, child = TRUE)
simpleSim(y, N = 3, alleles = LETTERS[1:10])
```

Index