Package 'forrel'

February 25, 2022

Type Package

Title Forensic Pedigree Analysis and Relatedness Inference

Version 1.4.1

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

Depends pedtools (>= 1.1.0), R (>= 4.1.0)

Imports glue, pedmut, pedprobr (>= 0.4), ribd (>= 1.3.0)

Suggests ggplot2, poibin, scales, testthat

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2022-02-25 20:50:02 UTC

R topics documented:

---

checkPairwise   *Check pedigree data for relationship errors*

Description

This function provides a convenient way to check for pedigree errors, given the available marker data. The function calls [ibdEstimate()](#) to estimate IBD coefficients for all pairs of typed pedigree members, and computes the likelihood ratio (LR) comparing each estimate to the coefficients implied by the pedigree. By default, the estimates are shown in a colour-coded plot where unlikely relationships are easy to spot.

Usage

```
checkPairwise(
  x,
  excludeInbred = TRUE,
  plot = TRUE,
  labels = FALSE,
  LRthreshold = 1000,
  ...
)
```

Arguments

x	A ped object or a list of such.
excludeInbred	A logical, by default TRUE, indicating if inbred individuals should be excluded from the analysis.
plot	A logical (default: TRUE). If TRUE, a plot is produced, showing the IBD estimates in the IBD triangle.
labels	A logical (default: FALSE). If TRUE, labels are included in the IBD triangle plot.
LRthreshold	A positive number (default: 1000). IBD estimates whose LR exceed this, when compared to the coefficients implied by the pedigree, are encircled in the plot.
...	Further parameters passed on to [ribd::ibdTriangle()](#).

Details

By default, inbred individuals are excluded from the analysis, since pairwise relationships involving inbred individuals have undefined kappa coefficients (and therefore no position in the triangle). In some cases it may still be informative to include their estimates; set excludeInbred = FALSE to achieve this.

Value

A data frame containing both the estimated and pedigree-based IBD coefficients for each pair of typed individuals. The last column contains the likelihood ratio comparing the estimated coefficients to the pedigree-based ones.

Author(s)

Magnus Dehli Vigeland

See Also

ibdEstimate()

Examples

```
### Example with realistic data

x = addSon(nuclearPed(nch = 2), parent = 4)
x = setMarkers(x, locus = NorwegianFrequencies)
x = profileSim(x, N = 1, seed = 1729)[[1]]

checkPairwise(x)

### Create sample swap between 1 and 3
als = getAlleles(x)
als[c(1,3), ] = als[c(3,1), ]
y = setAlleles(x, alleles = als)

checkPairwise(y)


# Combined plot of pedigree and IBD estimates
dev.new(height = 5, width = 8, noRStudioGD = TRUE)
layout(rbind(1:2), widths = 2:3)
plot(x, margins = c(4,2,4,2))
checkPairwise(x, labels = TRUE)
```

---

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 by simulation.
`nsim`	A positive integer; the number of simulations used for markers whose number of alleles exceeds `exactMaxL`.
`seed`	An integer 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).

It should be noted that `claimPed` and `truePed` may be any (lists of) pedigrees, as long as they both contain the individuals specified by `ids`. In particular, either alternative may have inbred founders (with the same or different coefficients), but this must be set individually for each.

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 two markers
m1 = marker(claim, alleles = 1:2)
m2 = marker(claim, alleles = 1:3)
claim = setMarkers(claim, list(m1, m2))

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

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

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 https://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)
```

---

findExclusions *Find markers excluding an identification*

Description

Find markers for which the genotypes of a candidate individual is incompatible with a pedigree

Usage

```
findExclusions(x, id, candidate, removeMut = TRUE)
```

Arguments

x	A ped object or a list of such.
id	A character of length 1; the name of an untyped member of x.
candidate	A singleton pedigree, with genotypes for the same markers as x.
removeMut	A logical. If TRUE (default), all mutations models are stripped.

Value

A character vector containing the names of incompatible markers.

Examples

```
# Pedigree with 3 siblings; simulate data for first two
x = nuclearPed(3) |>
  setMarkers(locusAttributes = NorwegianFrequencies[1:5]) |>
  profileSim(ids = 3:4, seed = 1)

# Simulate random person
poi = singleton(1) |>
  setMarkers(locusAttributes = NorwegianFrequencies[1:5]) |>
  profileSim(seed = 1)

# Identify incompatible markers
findExclusions(x, id = 5, candidate = poi)   # D21S11

# Inspect
plotPedList(c(x, poi), marker = "D21S11", frames = FALSE)
```

---

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.

---

ibdBootstrap   *Bootstrap estimation of IBD coefficients*

Description

This function produces (parametric or nonparametric) bootstrap estimates of the IBD coefficients between two individuals. Both kappa and delta coefficients are supported (see `ibdEstimate()`).

Usage

```
ibdBootstrap(
  x = NULL,
  ids = NULL,
  param = NULL,
  kappa = NULL,
  delta = NULL,
  N,
  method = "parametric",
  freqList = NULL,
  plot = TRUE,
  seed = NULL
)
```

Arguments

x	A ped object. If method = "parametric", this is only used to extract the allele frequencies, and can be skipped if freqList is provided.
ids	A pair of ID labels.
param	Either NULL (default), "kappa" or "delta". (See below.)
kappa, delta	Probability vectors of length 3 (kappa) or 9 (delta). Exactly one of param, kappa and delta must be non-NULL. If kappa and delta are both NULL, the appropriate set of coefficients is computed as ibdEstimate(x,ids,param).
N	The number of simulations.
method	Either "parametric" (default) or "nonparametric". Abbreviations are allowed. see Details for more information about each method.
freqList	A list of probability vectors: The allele frequencies for each marker.
plot	A logical, only relevant for bootstraps of kappa. If TRUE, the bootstrap estimates are plotted in the IBD triangle.
seed	An integer seed for the random number generator (optional).

Details

The parameter method controls how bootstrap estimates are obtained in each replication.

If method = "parametric", new profiles for two individuals are simulated from the input coefficients, followed by a re-estimation of the coefficients.

If method = "nonparametric", the original markers are sampled with replacement, before the coefficients are re-estimated.

Value

A data frame with N rows containing the bootstrap estimates. The last column (dist) gives the euclidean distance to the original coefficients, viewed as a point in R^3 (kappa) or R^9 (delta).

See Also

[ibdEstimate()](#)

Examples

```
# Frequency list of 15 standard STR markers
freqList = NorwegianFrequencies[1:15]

# Number of bootstrap simulations (increase!)
N = 5

# Bootstrap estimates for kappa of full siblings
boot1 = ibdBootstrap(kappa = c(0.25, .5, .25), N = N, freqList = freqList)
boot1

# Mean deviation
```

```
mean(boot1$dist)

# Same, but with the 9 identity coefficients.
delta = c(0, 0, 0, 0, 0, 0, .25, .5, .25)
boot2 = ibdBootstrap(delta = delta, N = N, freqList = freqList)

# Mean deviation
mean(boot2$dist)

#### Non-parametric bootstrap.
# Requires `x` and `ids` to be provided

x = nuclearPed(2)
x = markerSim(x, ids = 3:4, N = 50, alleles = 1:10, seed = 123)

bootNP = ibdBootstrap(x, ids = 3:4, param = "kappa", method = "non", N = N)

# Parametric bootstrap can also be done with this syntax
bootP = ibdBootstrap(x, ids = 3:4, param = "kappa", method = "par", N = N)
```

---

ibdEstimate   *Pairwise relatedness estimation*

Description

Estimate the IBD coefficients $\kappa = (\kappa_0, \kappa_1, \kappa_2)$ or the condensed identity coefficients $\Delta = (\Delta_1, ..., \Delta_9)$ between a pair (or several pairs) of pedigree members, using maximum likelihood methods.

Usage

```
ibdEstimate(
  x,
  ids = typedMembers(x),
  param = c("kappa", "delta"),
  markers = NULL,
  start = NULL,
  tol = sqrt(.Machine$double.eps),
  beta = 0.5,
  sigma = 0.5,
  contourPlot = FALSE,
  levels = NULL,
  verbose = TRUE
)
```

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. By default, all pairs of members of x are included.
param	Either "kappa" (default) or "delta"; indicating which set of coefficients should be estimated.
markers	A vector with names or indices of markers attached to x, indicating which markers to include. If NULL (default), all markers are used.
start	A probability vector (i.e., with nonnegative entries and sum 1) of length 3 (if param = "kappa") or 9 (if param = "delta"), indicating the initial value of for the optimisation. By default, start is set to (1/3, 1/3, 1/3) if param = "kappa" and (1/9, ..., 1/9) if param = "delta".
tol, beta, sigma	
	Control parameters for the optimisation routine; can usually be left untouched.
contourPlot	A logical. If TRUE, contours of the log-likelihood function are plotted overlaying the IBD triangle.
levels	(Only relevant if contourPlot = TRUE.) A numeric vector of levels at which to draw contour lines. If NULL (default), the levels are chosen automatically.
verbose	A logical.

Details

It should be noted that this procedure estimates the *realised* identity coefficients of each pair, i.e., the actual fractions of the autosomes in each IBD state. These may deviate substantially from the theoretical pedigree coefficients.

Maximum likelihood estimation of relatedness coefficients originates with Thompson (1975). Optimisation of κ is done in the (κ_0, κ_2)-plane and restricted to the triangle defined by

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

. Optimisation of Δ is done in unit simplex of R^8, using the first 8 coefficients.

The implementation optimises the log-likelihood using a projected gradient descent algorithm, combined with a version of Armijo line search.

Value

An object of class ibdEst, which is basically a dataframe with either 6 columns (if param = "kappa") or 12 columns (if param = "delta"). The first three columns are id1 (label of first individual), id2 (label of second individual) and N (the number of markers with no missing alleles). The remaining columns contain the coefficient estimates.

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

[ibdBootstrap()](#)

Examples

```
### Example 1: Siblings
x = nuclearPed(2)

# Simulate 100 markers
x = markerSim(x, N = 100, alleles = 1:4, seed = 123, verbose = FALSE)

# Estimate kappa (expectation: (0.25, 0.5, 0.25)
ibdEstimate(x, ids = 3:4)

# Plot contours of the log-likelihood function
ibdEstimate(x, ids = 3:4, contourPlot = TRUE)

### Example 2: Full sib mating
y = fullSibMating(1)

# Simulate 200 SNP markers
y = markerSim(y, N = 1000, alleles = 1:10, seed = 123, verbose = FALSE)

# Estimate
ibdEstimate(y, ids = 5:6, param = "delta")
```

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),
  xpd = TRUE,
  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, xpd	
	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 by 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 is shaded in light grey colour, but this can be modified with the `shading` argument.

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 list of pedigrees. One of the pedigrees (the last one, by default) is designated as 'reference', to be used in the denominator in all LR calculations. To ensure that all pedigrees use the same data set, one of the pedigrees may be chosen as 'source', from which data is transferred to all the other pedigrees.

Usage

```
kinshipLR(
  ...,
  ref = NULL,
  source = NULL,
  markers = NULL,
  linkageMap = NULL,
  verbose = FALSE
)
```

Arguments

`...`	Pedigree alternatives. Each argument should be either a single ped object or a list of such. The pedigrees may be named; otherwise they are assigned names "H1", "H2", ... automatically.
	It is also possible to pass a single list containing all the pedigrees.
`ref`	An index or name indicating which of the input pedigrees should be used as "reference pedigree", i.e., used in the denominator of each LR. If NULL (the default), the last pedigree is used as reference.
`source`	An index or name designating one of the input pedigrees as source for marker data. If given, marker data is transferred from this to all the other pedigrees (replacing any existing markers). The default action (`source = NULL`) is as follows: If all pedigree have attached markers, no transfers are done. If exactly one of the pedigrees have attached markers, these are transferred to the others. all other cases give an error.
`markers`	A vector of marker names or indices indicating which markers should be included. If NULL (the default) all markers are used.
`linkageMap`	Either NULL (default), or a data frame with three columns: chromosome; marker name; centiMorgan position. If given, it signifies to the program that the markers are linked and invokes MERLIN for computing the likelihoods.
`verbose`	A logical.

Details

By default, all markers are assumed to be unlinked. To accommodate linkage, a genetic map should be supplied with the argument `linkageMap`. This requires the software MERLIN to be installed.

Value

A `LRresult` object, which is essentially a list with entries

- `LRtotal` : A vector of length L, where L is the number of input pedigrees. The i'th entry is the total LR (i.e., the product over all markers) comparing pedigree i to the reference pedigree. The entry corresponding to the reference will always be 1.

- `LRperMarker` : A numerical matrix, where the i'th column contains the marker-wise LR values comparing pedigree i to the reference. The product of all entries in a column should equal the corresponding entry in `LRtotal`.

- `likelihoodsPerMarker` : A numerical matrix of the same dimensions as `LRperMarker`, but where the entries are likelihood of each pedigree for each marker.

- `time` : Elapsed time

Author(s)

Magnus Dehli Vigeland and Thore Egeland

See Also

[LRpower()](), [pedprobr::likelihoodMerlin()]()

Examples

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

# Create two alternative hypotheses
halfsibs = relabel(halfSibPed(), old = 4:5, new = ids)
unrel = list(singleton("A"), singleton("B"))

# Compute LRs. By default, the last ped is used as reference
kinshipLR(sibs, halfsibs, unrel)

# Input pedigrees can be named, reflected in the output
kinshipLR(S = sibs, H = halfsibs, U = unrel)

# Select non-default reference (by index or name)
kinshipLR(S = sibs, H = halfsibs, U = unrel, ref = "H")

# Alternative syntax: List input
peds = list(S = sibs, H = halfsibs, U = unrel)
kinshipLR(peds, ref = "H", source = "S", verbose = TRUE)

# Detailed results
res = kinshipLR(peds)
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 pedi-
        gree. 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 pedi-
                        gree 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   An integer 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
  truth = addMarker(truth, alleles = als, afreq = afr)

  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	(Only if `partialmarker` is NULL.) A vector with allele labels. If NULL, the following are tried in order:

- `names(afreq)`
- 'seq_along(afreq)'
- `1:2` (Fallback if both `alleles` and `afreq` are NULL.)

afreq	(Only if `partialmarker` is NULL.) A numeric vector with allele frequencies, possibly named with allele labels.
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	An integer seed for the random number generator (optional).
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 = addMarker(x, "3" = "1/1", alleles = 1:3)
markerSim(x, N = 2, partialmarker = 1)
markerSim(x, N = 1, ids = 4, partialmarker = 1, verbose = FALSE)
```

---

markerSimParametric   *Simulate marker data given IBD coefficients*

Description

This function simulates genotypes for two individuals given their IBD distribution, for N identical markers.

Usage

```
markerSimParametric(
  kappa = NULL,
  delta = NULL,
  states = NULL,
  N = 1,
  alleles = NULL,
  afreq = NULL,
  seed = NULL,
  returnValue = c("singletons", "alleles", "genotypes", "internal")
)
```

Arguments

kappa   A probability vector of length 3, giving a set of realised kappa coefficients (between two noninbred individuals).

delta   A probability vector of length 9, giving a set of condensed identity coefficients (Jacquard coefficients).

states	An integer vector of length N, with entries in 1-9. Each entry gives the identity state of the corresponding marker. (See details.)
N	A positive integer: the number of independent markers to be simulated.
alleles	A vector with allele labels. If NULL, the following are tried in order:

- names(afreq)
- 'seq_along(afreq)'
- 1:2 (fallback if both alleles and afreq are NULL)

afreq	A numeric vector with allele frequencies, possibly named with allele labels.
seed	An integer seed for the random number generator (optional).
returnValue	Either "singleton" (default) or "alleles". (see Value).

Details

Exactly one of kappa, delta and states must be given; the other two should remain NULL.

If states is given, it explicitly determines the condensed identity state at each marker. The states are described by integers 1-9, using the tradition order introduced by Jacquard.

If kappa is given, the states are generated by the command states = sample(9:7, size = N, replace = TRUE, prob = kappa). (Note that identity states 9, 8, 7 correspond to IBD status 0, 1, 2, respectively.)

If delta is given, the states are generated by the command states = sample(1:9, size = N, replace = TRUE, prob = delta).

Value

The output depends on the value of the returnValue parameter:

- "singletons": a list of two singletons with the simulated marker data attached.
- "alleles": a list of four vectors of length N, named a, b, c and d. These contain the simulated alleles, where a/b and c/d are the genotypes of the to individuals.
- "genotypes": a list of two vectors of length N, containing the simulated genotypes. Identical to paste(a,b,sep = "/") and paste(c,d,sep = "/"), where a, b, c, d are the vectors returned when returnValue == "alleles".
- "internal": similar to "alleles", but using the index integer of each allele. (This option is mostly for internal use.)

Examples

```
# MZ twins
markerSimParametric(kappa = c(0,0,1), N = 5, alleles = 1:10)

# Equal distribution of states 1 and 2
markerSimParametric(delta = c(.5,.5,0,0,0,0,0,0,0), N = 5, alleles = 1:10)

# Force a specific sequence of states
markerSimParametric(states = c(1,2,7,8,9), N = 5, alleles = 1:10)
```

missingPersonEP | *Exclusion power for missing person cases*

Description

This is a special case of [exclusionPower()](#) for use in missing person cases. The function computes the probability that a random person is genetically incompatible with the typed relatives of the missing person.

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 the pedigree. 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.
- 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.

Details

This function is identical to [randomPersonEP()](#), but with different argument names. This makes it consistent with [missingPersonIP()](#) and the other 'missing person' functions.

Value

The EPresult object returned by exclusionPower().

See Also

randomPersonEP(), 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
# (by default this is ignored by `missingPersonEP()`)
x = addMarker(x, "3" = "1/1", "4" = "2/2", "5" = "3/3")

# Compute exclusion power statistics
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 the pedigree. 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.
- 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	An integer 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
```

---

missingPersonLR   *Likelihood ratio calculation for missing person identification*

Description

This is a wrapper function for [kinshipLR()](#) for the special case of missing person identification. A person of interest (POI) is matched against a reference dataset containing genotypes of relatives of the missing person.

Usage

```
missingPersonLR(reference, missing, poi)
```

Arguments

reference	A ped object with attached markers.
missing	The ID label of the missing member of reference.
poi	A singleton object with genotypes for the person of interest. The marker names of poi must coincide with those of reference. The ID label of poi is irrelevant.

Value

The LRresult object returned by [kinshipLR()](#), but without the trivial H2:H2 comparison.

Examples

```
#------------------------------------------------
# Example: Identification of a missing grandchild
#------------------------------------------------

set.seed(2509)

### Reference pedigree with missing grandchild (MP)
x = relabel(linearPed(2), old = 5, new = "MP")

# Simulate reference data for grandmother (5 STR markers)
x = setMarkers(x, locusAttributes = NorwegianFrequencies[1:5])
x = profileSim(x, N = 1, ids = 2)[[1]]
```

```
### Person of interest 1: Unrelated
poi1 = singleton("poi1")

# Transfer (empty) markers and simulate genotypes
poi1 = transferMarkers(from = x, to = poi1)
poi1 = profileSim(poi1, N = 1)[[1]]

# Compute LR
lr1 = missingPersonLR(x, missing = "MP", poi = poi1)
lr1
lr1$LRperMarker


### Person of interest 2: The true MP

# Simulate MP conditional on reference, and extract as singleton
poi2 = profileSim(x, N = 1, ids = "MP")[[1]]

# Extract MP as singleton
poi2 = subset(poi2, "MP")

# Compute LR
lr2 = missingPersonLR(x, missing = "MP", poi = poi2)
lr2
lr2$LRperMarker
```

---

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,
  labs = labels(reference),
  marker = NULL,
  hatched = typedMembers(reference),
  MP.label = "MP",
  POI.label = "POI",
  POI.sex = getSex(reference, missing),
  POI.col = "red",
```

```
    POI.hatched = FALSE,
    POI.height = 8,
   titles = c(expression(H[1] * ": POI = MP"), expression(H[2] * ": POI unrelated")),
    width = 4,
    newdev = interactive(),
    ...
)
```

Arguments

`reference`	A [`pedtools::ped()`](pedtools::ped()) object.
`missing`	The ID label of the missing pedigree member.
`labs`	A character vector with labels for the pedigree members. See [`pedtools::plot.ped()`](pedtools::plot.ped()).
`marker`	Optional vector of marker indices to be included in the plot.
`hatched`	A vector of ID labels indicating who should appear with hatched symbols in the plot. By default, all typed members.
`MP.label`	The label of the missing member. Default: "MP".
`POI.label`	The label of the person of interest. Default: "POI".
`POI.sex`	The sex of POI. This defaults to that of the missing person, but may be set explicitly. This is particularly useful when the missing person has unknown sex.
`POI.col`	The plot colour of POI. Default: red.
`POI.hatched`	A logical: If TRUE (default), the POI is plotted with a hatched symbol.
`POI.height`	A numeric controlling the vertical placement of the POI singleton (in the right panel).
`titles`	A character of length 2, with subtitles for the two frames.
`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.
`...`	Extra parameters passed on to [`pedtools::plotPedList()`](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", labs = NULL,
                  hatched = "b1", POI.hatched = 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 = 1,
  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 the pedigree. 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.
- 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 for parallelisation, by default 1.
seed	An integer 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 4 alleles
x = addMarker(x, S1 = "1/2", alleles = 1:4)

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

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

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

# Power plot
powerPlot(simData, type = 3)


### With  mutations
# Add inconsistent marker
x = addMarker(x, S1 = "1/2", Father = "3/3", alleles = 1:4)

# 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)

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


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

NorwegianFrequencies   *Norwegian STR frequencies*

Description

A database of Norwegian allele frequencies for 35 STR markers. This database is also available in the R package `Familias`.

Usage

```
NorwegianFrequencies
```

Format

A list of length 35. Each entry is a numerical vector summing to 1, named with allele labels.

The following markers are included:

- `D3S1358`: 12 alleles
- `TH01`: 10 alleles
- `D21S11`: 26 alleles
- `D18S51` : 23 alleles
- `PENTA_E`: 21 alleles
- `D5S818`: 9 alleles
- `D13S317`: 9 alleles
- `D7S820`: 19 alleles
- `D16S539`: 9 alleles
- `CSF1PO`: 11 alleles
- `PENTA_D`: 24 alleles
- `VWA`: 12 alleles
- `D8S1179`: 12 alleles
- `TPOX`: 9 alleles
- `FGA`: 25 alleles
- `D19S433`: 17 alleles
- `D2S1338`: 13 alleles
- `D10S1248`: 9 alleles
- `D1S1656`: 17 alleles
- `D22S1045`: 9 alleles
- `D2S441`: 13 alleles
- `D12S391`: 23 alleles
- `SE33`: 55 alleles

- D7S1517: 11 alleles
- D3S1744: 8 alleles
- D2S1360: 10 alleles
- D6S474: 6 alleles
- D4S2366: 7 alleles
- D8S1132: 12 alleles
- D5S2500: 8 alleles
- D21S2055: 18 alleles
- D10S2325: 10 alleles
- D17S906: 78 alleles
- APOAI1: 41 alleles
- D11S554: 51 alleles

Source

Dupuy et al. (2013): *Frequency data for 35 autosomal STR markers in a Norwegian, an East African, an East Asian and Middle Asian population and simulation of adequate database size*. Forensic Science International: Genetics Supplement Series, Volume 4 (1).

---

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 = NULL,
  type = 1,
  majorpoints = TRUE,
  minorpoints = TRUE,
  ellipse = FALSE,
  col = NULL,
  labs = NULL,
  jitter = FALSE,
  alpha = 1,
  stroke = 1.5,
  shape = "circle",
  size = 1,
```

```
    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.
jitter	A logical (default: FALSE). If TRUE, a small jitter is added to the major points.
alpha	Transparency for minor points (see Details).
stroke	Border width for major 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

```
### Example 1: Comparing the power of 3 reference families ###

# Helper function for simulating a single profile
sim = function(x, id) {
  y = setMarkers(x, locusAttributes = NorwegianFrequencies[1:3])
  profileSim(y, ids = id)[[1]]
}

# Define pedigrees and simulate data
PAR = nuclearPed(1, child = "MP") |> sim(id = 1)
SIB = nuclearPed(2) |> relabel(old = 4, new = "MP") |> sim(id = 3)
GRA = linearPed(2) |> relabel(old = 5, new = "MP") |> sim(id = 1)

# Collect in list and plot
peds = list(PAR = PAR, SIB = SIB, GRA = GRA)
plotPedList(peds, marker = 1, hatched = typedMembers, frames = FALSE,
            col = list(red = "MP"))

# Compute exclusion/inclusion powers:
ep = lapply(peds, function(y)
  missingPersonEP(y, missing = "MP", verbose = FALSE))

ip = lapply(peds, function(y)    # increase nsim!
  missingPersonIP(y, missing = "MP", nsim = 5, threshold = 10, verbose = FALSE))

# Plot
powerPlot(ep, ip, size = 2)
powerPlot(ep, ip, size = 2, jitter = TRUE)

# Different plot type, not dependent of `threshold`
powerPlot(ep, ip, size = 2, type = 3)

### Example 2: Exploring powers for different sets of available relatives

# Create trio pedigree
ref = nuclearPed(father = "fa", mother = "mo", child = "MP")
```

```
# Add empty marker with 5 alleles
ref = addMarker(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(0.2, 1), ylim = c(0.5, 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,
    numCores = 1,
    verbose = TRUE,
    ...
)
```

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 geno-types should be simulated.
markers	A list of marker objects, or a vector containing names or indices referring to markers attached to x. By default (NULL), all attached markers are used. The simulations will be conditional on the locus attributes (allele frequencies, muta-tion models a.s.o.) and any existing genotypes in the indicated markers.
seed	An integer seed for the random number generator (optional).
numCores	The number of cores used for parallelisation, by default 1.
verbose	A logical, by default TRUE.
...	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
x = addMarker(x, B1 = "1/2", alleles = 1:3)
x = addMarker(x, B1 = "1", alleles = 1:4, afreq = (1:4)/10, chrom = "X")

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

---

profileSimParametric   *Simulate complete DNA profiles given IBD coefficients*

Description

This function generalises `markerSimParametric()` in the same way that `profileSim()` generalises `markerSim()`.

Usage

```
profileSimParametric(
  kappa = NULL,
  delta = NULL,
  states = NULL,
  N = 1,
  freqList = NULL,
  seed = NULL,
  returnValue = c("singletons", "alleles", "genotypes", "internal")
)
```

Arguments

kappa	A probability vector of length 3, giving a set of realised kappa coefficients (between two noninbred individuals).
delta	A probability vector of length 9, giving a set of condensed identity coefficients (Jacquard coefficients).
states	An integer vector of length N, with entries in 1-9. Each entry gives the identity state of the corresponding marker. (See details.)
N	A positive integer: the number of complete profiles to be simulated
freqList	A list of numeric vectors. Each vector is the allele frequencies of a marker.
seed	An integer seed for the random number generator (optional).
returnValue	Either "singleton" (default) or "alleles". (see Value).

Value

A list of length N, whose entries are determined by `returnValue`, as explained in `markerSimParametric()`.

Examples

```
# A single profile with 9 markers, each with forced identity state
profileSimParametric(states = 1:9, freqList = NorwegianFrequencies[1:9])
```

---

randomPersonEP   *Random person exclusion power*

Description

This is a special case of `exclusionPower()`, computing the power to exclude a random person as a given pedigree member. More specifically, the function computes the probability of observing, in an individual unrelated to the family individual, a genotype incompatible with the typed family members.

Usage

```
randomPersonEP(x, id, markers = NULL, disableMutations = NA, verbose = TRUE)
```

Arguments

x   A ped object with attached markers.

id   The ID label of a single pedigree member.

markers   A vector indicating the names or indices of markers attached to the source pedi-
                  gree. 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 the pedigree. This is determined
                    by temporarily removing all mutation models and checking which mark-
                    ers have nonzero likelihood.
                  • TRUE: Mutations are disabled for all markers. This will result in an error if
                    any markers are inconsistent.
                  • FALSE: No action is done to disable mutations.
                  • A vector containing the names or indices of those markers for which muta-
                    tions should be disabled.

verbose   A logical.

Value

The EPresult object returned by `exclusionPower()`.

Examples

```
# Four siblings:
x = nuclearPed(4)

# First 3 sibs typed with 4 triallelic markers
```

```
x = markerSim(x, N = 4, ids = 3:5, alleles = 1:3, seed = 577, verbose = FALSE)

# Probability that a random man is excluded as the fourth sibling
randomPersonEP(x, id = 6)
```

---

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". (De-
              fault = 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, frequen-
cies) 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

This function is re-exported from the ribd package. For documentation see [ribd::showInTriangle()](#).

---

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 with allele labels.
afreq	a numeric vector of allele frequencies. If missing, the alleles are assumed to be equi-frequent.
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	An integer seed for the random number generator (optional).
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

```
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