Package ‘forensIT’

June 22, 2023

Title Information Theory Tools for Forensic Analysis
Version 1.0.0
Description The ‘forensIT’ package is a comprehensive statistical toolkit tailored for handling missing person cases. By leveraging information theory metrics, it enables accurate assessment of kinship, particularly when limited genetic evidence is available. With a focus on optimizing statistical power, ‘forensIT’ empowers investigators to effectively prioritize family members, enhancing the reliability and efficiency of missing person investigations.
License GPL (>= 3)
Encoding UTF-8
LazyData true
RoxygenNote 7.2.3
Imports ggplot2, mispitools, forrel, pedprobr, dplyr, tidyr, magrittr,
      fbnet, foreach, hrbrthemes, gtools, pedtools, reshape2,
      iterators, doParallel, paramlink
Depends R (>= 2.10)
NeedsCompilation no
Author Franco Marsico [aut, cre],
      Ariel Chernomoretz [aut]
Maintainer Franco Marsico <franco.lmarsico@gmail.com>
Repository CRAN
Date/Publication 2023-06-22 19:10:02 UTC

R topics documented:

  buildEnsembleCPTs ........................................ 2
  buildEnsembleITValues .................................. 3
  compareBnetPopGenoPDFs .................................. 4
  convertPed ................................................ 4
  crossH ..................................................... 5
  distKL ...................................................... 5
  elimLangeGoradia ......................................... 6
**buildEnsembleCPTs**

**Summary**

Build ensemble of CPTs from a list of simulations.

**Usage**

```r
buildEnsembleCPTs(lsimu, lminimalProbGenoMOI)
```

**Arguments**

- `lsimu`: list of simulations
- `lminimalProbGenoMOI`: list of minimal probabilities of genotypes given MOI # nolint

**Value**

list of CPTs
**buildEnsembleITValues**

### Description

Build ensemble of IT values from a list of simulations

### Usage

```r
buildEnsembleITValues(
  lsimu = lsimulation,
  ITtab = sim$ITtable,
  bFullIT = FALSE
)
```

### Arguments

- `lsimu`: list of simulations
- `ITtab`: IT table
- `bFullIT`: boolean to return full IT table

### Value

list of IT values

### Examples

```r
library(forrel)
library(mispitools)
freqs <- lapply(getfreqs(Argentina)[1:15], function(x) {x[x!=0]})
fam <- linearPed(2)
fam <- addChildren(fam, father = 1, mother = 2)
fam <- pedtools::setMarkers(fam, locusAttributes = freqs)
ped <- profileSim(fam, N = 1, ids = c(6) , numCores = 1, seed=123)
lsimEnsemble <- simTestIDMarkers(ped,2,numSim=5,seed=123)
lensembleIT <- buildEnsembleITValues(lsimu=lsimEnsemble,ITtab=simME$ITtable,bFullIT = TRUE)
lensembleCPTs <- buildEnsembleCPTs(lsimu=lsimEnsemble,lminimalProbGenoMOI=simME$lprobGenoMOI)
```
compareBnetPopGenoPDFs

Compare population and Bayesian network genotype probability density functions # nolint

Description

Compare population and Bayesian network genotype probability density functions # nolint

Usage

compareBnetPopGenoPDFs(lprobTable)

Arguments

lprobTable list of probability tables

Value

list of KL divergences

convertPed

Convert a pedigree to a paramlink object

Description

Convert a pedigree to a paramlink object

Usage

convertPed(x, verbose = FALSE)

Arguments

x pedigree
   verbose print progress

Value

paramlink object
Examples

library(forrel)
  x = linearPed(2)
  plot(x)
  x = setMarkers(x, locusAttributes = NorwegianFrequencies[1:2])
  x = profileSim(x, N = 1, ids = 2)
  convertPed(x)

crossH

Description

Cross entropy

Usage

crossH(px, py, epsilon = 1e-20)

Arguments

(px, py) probability distribution
epsilon small number to avoid log(0)

Value

cross entropy

distKL

distKL: KL distribution obtained for specific relative contributor

Description

distKL: KL distribution obtained for specific relative contributor

Usage

distKL(ped, missing, relative, frequency, numsims = 100, cores = 1)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ped</td>
<td>Reference pedigree. It could be an input from read_fam() function or a pedigree built with pedtools. # nolint</td>
</tr>
<tr>
<td>missing</td>
<td>Missing person</td>
</tr>
<tr>
<td>relative</td>
<td>Selected relative.</td>
</tr>
<tr>
<td>frequency</td>
<td>Allele frequency database.</td>
</tr>
<tr>
<td>numsims</td>
<td>Number of simulated genotypes.</td>
</tr>
<tr>
<td>cores</td>
<td>Enables parallelization.</td>
</tr>
</tbody>
</table>

Value

An object of class data.frame with KLs.

Examples

```r
library(forrel)
x = linearPed(2)
x = setMarkers(x, locusAttributes = NorwegianFrequencies[1:2])
x = profileSim(x, N = 1, ids = 2)
distKL(ped = x, missing = 5, relative = 1, cores = 1, frequency = NorwegianFrequencies[1:2], numsims = 3)
```

__elimLangeGoradia__

_Eliminate Mendelian errors using Lange-Goradia algorithm_

Description

Eliminate Mendelian errors using Lange-Goradia algorithm

Usage

```r
elimLangeGoradia(ped, iMarker = 1, bitera = TRUE, bverbose = TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ped</td>
<td>pedigree</td>
</tr>
<tr>
<td>iMarker</td>
<td>index of marker to be used</td>
</tr>
<tr>
<td>bitera</td>
<td>iterate until no more errors are found</td>
</tr>
<tr>
<td>bverbose</td>
<td>print progress</td>
</tr>
</tbody>
</table>

Value

pedigree with Mendelian errors eliminated
**exportPed**

*Export a pedigree to a file*

**Description**

Export a pedigree to a file

**Usage**

```r
exportPed(ped, fname, iMarker = 1)
```

**Arguments**

- `ped`: pedigree
- `fname`: file name
- `iMarker`: index of marker to be used

**Value**

pedigree with Mendelian errors eliminated

---

**forensIT**

*forensIT: Information Theory Tools for Forensic Analysis*

**Description**

The 'forensIT' package, available on CRAN, is a comprehensive statistical toolkit tailored for handling missing person cases. By leveraging information theory metrics, it enables accurate assessment of kinship, particularly when limited genetic evidence is available. With a focus on optimizing statistical power, 'forensIT' empowers investigators to effectively prioritize family members, enhancing the reliability and efficiency of missing person investigations. Experience the power of information theory in kinship testing with the user-friendly 'forensIT' package, freely accessible on CRAN. # nolint
**genotypeProbs**

*Genotype probabilities*

**Description**

Calculate genotype probabilities from parental probabilities

**Usage**

```r
genotypeProbs(probP, probM)
```

**Arguments**

- `probP` vector of parental probabilities
- `probM` vector of parental probabilities

**Value**

matrix of genotype probabilities

---

**genotypeProbTable**

*Genotype Probability Table*

**Description**

Genotype Probability Table

**Usage**

```r
genotypeProbTable(bbn1, resQQ, bplot = FALSE, numMarkers = 4, lLoci)
```

**Arguments**

- `bbn1` Bayesian network
- `resQQ` results from bn
- `bplot` boolean to plot
- `numMarkers` number of markers
- `lLoci` list of loci

**Value**

Genotype Probability Table
**genotypeProbTable_bis**

**Description**
function to calculate the probability of genotypes given the MOI

**Usage**
```
genotypeProbTable_bis(bbn1, resQQ, bplot = FALSE, numMarkers = 4, freq)
```

**Arguments**
- `bbn1`: bayesian network
- `resQQ`: list of results from the inference
- `bplot`: plot results
- `numMarkers`: number of markers
- `freq`: allele frequencies

**Value**
matrix of genotype probabilities

---

**getAllelesFromGenotypes**

**Description**
Get alleles from genotypes

**Usage**
```
getAllelesFromGenotypes(g)
```

**Arguments**
- `g`: genotypes

**Value**
alleles
Entropy of a discrete probability distribution

Description
Entropy of a discrete probability distribution

Usage
H(px, epsilon = 1e-20, normalized = FALSE)

Arguments
- px: probability distribution
- epsilon: small number to avoid log(0)
- normalized: boolean to normalize entropy

Value
- entropy

index2Genotypes2

Description
index2Genotypes2

Usage
index2Genotypes2(ped, id, iMarker, alleleSet)

Arguments
- ped: pedigree
- id: individual id
- iMarker: marker index
- alleleSet: allele set

Value
- genotypes
**index2Genotypes2.pedtools**

**index2Genotypes**

**Description**

*index2Genotypes*

**Usage**

`index2Genotypes2.pedtools(ped, id, iMarker, alleleSet)`

**Arguments**

- `ped` : pedigree
- `id` : individual id
- `iMarker` : marker index
- `alleleSet` : allele set

**Value**

`genotypes`

---

**KLd**

**KL divergence**

**Description**

*KL divergence*

**Usage**

`KLd(ppx, ppy, epsilon = 1e-20, bsigma = FALSE)`

**Arguments**

- `ppx` : probability distribution
- `ppy` : probability distribution
- `epsilon` : small number to avoid log(0)
- `bsigma` : boolean to compute sigma

**Value**

KL divergence
### KLde
**KL divergence**

**Description**
KL divergence

**Usage**
KLde(px, py, epsilon = 1e-20)

**Arguments**
- **px**: probability distribution
- **py**: probability distribution
- **epsilon**: small number to avoid log(0)

**Value**
KL divergence

### perMarkerKLs

**Description**
perMarkerKLs

**Usage**
perMarkerKLs(ped, MP, frequency)

**Arguments**
- **ped**: Reference pedigree.
- **MP**: missing person
- **frequency**: Allele frequency database.

**Value**
An object of class data.frame with KLs.
Examples

```r
library(forrel)
x = linearPed(2)
plot(x)
x = setMarkers(x, locusAttributes = NorwegianFrequencies[1:5])
x = profileSim(x, N = 1, ids = 2)
perMarkerKLs(x, MP = 5, NorwegianFrequencies[1:5])
```

**plotKL**

*Plot KL distances.*

Description

Plot KL distances.

Usage

`plotKL(res)`

Arguments

- `res` output from `distKL` function.

Value

A scatterplot.

Examples

```r
library(forrel)
x = linearPed(2)
plot(x)
x = setMarkers(x, locusAttributes = NorwegianFrequencies[1:5])
x = profileSim(x, N = 1, ids = 2)
res <- distKL(ped = x, missing = 5, relative = 1, cores = 1, frequency = NorwegianFrequencies[1:5], numsims = 5)
plotKL(res)
```
runIT

**Description**

run information theory (IT) metrics

**Usage**

```r
runIT(
  lped = NULL,
  freqs,
  QP,
  dbg,
  numCores,
  bOnlyIT = FALSE,
  lprobg_ped = NULL,
  bsigma = FALSE,
  blog = FALSE,
  dep = TRUE
)
```
simLR

Arguments

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lped</td>
<td>list of pedigree objects</td>
</tr>
<tr>
<td>freqs</td>
<td>list of allele frequencies</td>
</tr>
<tr>
<td>QP</td>
<td>QP</td>
</tr>
<tr>
<td>dbg</td>
<td>debug</td>
</tr>
<tr>
<td>numCores</td>
<td>number of cores</td>
</tr>
<tr>
<td>bOnlyIT</td>
<td>boolean to only run IT</td>
</tr>
<tr>
<td>lprobg_ped</td>
<td>list of probG</td>
</tr>
<tr>
<td>bsigma</td>
<td>boolean to compute sigma</td>
</tr>
<tr>
<td>blog</td>
<td>boolean to write log</td>
</tr>
<tr>
<td>dep</td>
<td>check fbnet dependency</td>
</tr>
</tbody>
</table>

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runIT</td>
<td>Simulate LR</td>
</tr>
</tbody>
</table>

Description

Simulate LR

Usage

```r
simLR(
  lprobg_ped,
  numSim = 10000,
  epsilon = 1e-20,
  bplot = FALSE,
  bLRs = FALSE,
  seed = 123457
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lprobg_ped</td>
<td>list of probability distributions</td>
</tr>
<tr>
<td>numSim</td>
<td>number of simulations</td>
</tr>
<tr>
<td>epsilon</td>
<td>small number to avoid log(0)</td>
</tr>
<tr>
<td>bplot</td>
<td>boolean to plot</td>
</tr>
<tr>
<td>bLRs</td>
<td>boolean to return LRs</td>
</tr>
<tr>
<td>seed</td>
<td>seed</td>
</tr>
</tbody>
</table>

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRs</td>
<td></td>
</tr>
</tbody>
</table>
simMinimalEnsemble

Description

simME: output from simMinimalEnsemble considering an uncle

Usage

simME

Format

A list with minimalEnsemble of genotypes

simMinimalEnsemble

Description

It performs simulations of minimal ensembles of genotypes

Usage

simMinimalEnsemble(
  ped,
  QP,
  testID,
  freqs,
  numCores = 1,
  seed = 123457,
  bVerbose = TRUE,
  bJustGetNumber = FALSE,
  bdbg = FALSE,
  dep = TRUE
)

Arguments

ped pedigree
QP QP
testID test ID
freqs frequencies
numCores number of cores
**simTestIDMarkers**

seed

bVerbose boolean to print information

bJustGetNumber boolean to just get the number of runs

bdbg boolean to debug

dep check dependency fbnet

**Value**

list of results

---

**simTestIDMarkers**  
*Simulate testID markers*

**Description**

Simulate testID markers

**Usage**

`simTestIDMarkers(ped, testID, numSim = 10, seed = 123457)`

**Arguments**

<table>
<thead>
<tr>
<th>ped</th>
<th>pedigree</th>
</tr>
</thead>
<tbody>
<tr>
<td>testID</td>
<td>test ID</td>
</tr>
<tr>
<td>numSim</td>
<td>number of simulations</td>
</tr>
<tr>
<td>seed</td>
<td>seed</td>
</tr>
</tbody>
</table>

**Value**

list of simulations

**Examples**

```r
library(forrel)
library(mispitools)
freqs <- lapply(getfreqs(Argentina)[1:15], function(x) {x[x!=0]})
fam <- linearPed(2)
fam <- addChildren(fam, father = 1, mother = 2)
fam <- pedtools::setMarkers(fam, locusAttributes = freqs)
ped <- profileSim(fam, N = 1, ids = c(6) , numCores = 1,seed=123)
lsimEnsemble <- simTestIDMarkers(ped,2,numSim=5,seed=123)
```
Description

strsplit2

Usage

strsplit2(x, split)

Arguments

x character vector
split character

Value

matrix

Description

Check for Mendelian errors in trios

Usage

trioCheckFast(ffa, mmo, oof)

Arguments

ffa father’s alleles
mmo mother’s alleles
oof offspring’s alleles

Value

TRUE if there is a Mendelian error
unidimKLplot

unidimKLplot: KL distributions presented in the same units (Log10(LR))

Description

unidimKLplot: KL distributions presented in the same units (Log10(LR))

Usage

unidimKLplot(res)

Arguments

res output from distKL function.

Value

A scatterplot.
## Index

<table>
<thead>
<tr>
<th><strong>datasets</strong></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>simME, 16</td>
<td></td>
</tr>
<tr>
<td>buildEnsembleCPTs, 2</td>
<td></td>
</tr>
<tr>
<td>buildEnsembleITValues, 3</td>
<td></td>
</tr>
<tr>
<td>compareBnetPopGenoPDFs, 4</td>
<td></td>
</tr>
<tr>
<td>convertPed, 4</td>
<td></td>
</tr>
<tr>
<td>crossH, 5</td>
<td></td>
</tr>
<tr>
<td>distKL, 5</td>
<td></td>
</tr>
<tr>
<td>elimLangeGoradia, 6</td>
<td></td>
</tr>
<tr>
<td>exportPed, 7</td>
<td></td>
</tr>
<tr>
<td>forensIT, 7</td>
<td></td>
</tr>
<tr>
<td>genotypeProbs, 8</td>
<td></td>
</tr>
<tr>
<td>genotypeProbTable, 8</td>
<td></td>
</tr>
<tr>
<td>genotypeProbTable_bis, 9</td>
<td></td>
</tr>
<tr>
<td>getAllelesFromGenotypes, 9</td>
<td></td>
</tr>
<tr>
<td>H, 10</td>
<td></td>
</tr>
<tr>
<td>index2Genotypes2, 10</td>
<td></td>
</tr>
<tr>
<td>index2Genotypes2.pedtools, 11</td>
<td></td>
</tr>
<tr>
<td>KLd, 11</td>
<td></td>
</tr>
<tr>
<td>KLde, 12</td>
<td></td>
</tr>
<tr>
<td>perMarkerKLs, 12</td>
<td></td>
</tr>
<tr>
<td>plotKL, 13</td>
<td></td>
</tr>
<tr>
<td>Px, 14</td>
<td></td>
</tr>
<tr>
<td>runIT, 14</td>
<td></td>
</tr>
<tr>
<td>simLR, 15</td>
<td></td>
</tr>
<tr>
<td>simME, 16</td>
<td></td>
</tr>
<tr>
<td>simMinimalEnsemble, 16</td>
<td></td>
</tr>
<tr>
<td>simTestIDMarkers, 17</td>
<td></td>
</tr>
<tr>
<td>strsplit2, 18</td>
<td></td>
</tr>
<tr>
<td>trioCheckFast, 18</td>
<td></td>
</tr>
<tr>
<td>unidimKLplot, 19</td>
<td></td>
</tr>
</tbody>
</table>