Package: forensIT (via r-universe)

March 6, 2025

Title Information Theory Tools for Forensic Analysis

Version 1.1.1

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 7.3.2

Imports ggplot2, mispitools, forrel, pedprobr, dplyr, tidyr, magrittr, fbnet, foreach, hrbrthemes, gtools, reshape2, pedtools, iterators, doParallel

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 2025-02-04 15:10:02 UTC

Config/pak/sysreqs libcairo2-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Contents

`buildEnsembleCPTs` *buildEnsembleCPTs*

Description

Build ensemble of CPTs from a list of simulations

Usage

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

Arguments

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

Value

list of CPTs

Examples

```
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)
```

`buildEnsembleITValues` *buildEnsembleITValues*

Description

Build ensemble of IT values from a list of simulations

Usage

```
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

```
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)
```

`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

`crossH` *Cross entropy*

Description

Cross entropy

Usage

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

Arguments

px	probability distribution
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

ped	Reference pedigree. It could be an input from read_fam() function or a pedigree built with pedtools. # nolint
missing	Missing person
relative	Selected relative.
frequency	Allele frequency database.
numsims	Number of simulated genotypes.
cores	Enables parallelization.

Value

An object of class data.frame with KLs.

Examples

```
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

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

Arguments

`ped`	pedigree
`iMarker`	index of marker to be used
`bitera`	iterate until no more errors are found
`bverbose`	print progress

Value

pedigree with Mendelian errors eliminated

`exportPed` *Export a pedigree to a file*

Description

Export a pedigree to a file

Usage

```
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

Author(s)

Maintainer: Franco Marsico `<franco.lmarsico@gmail.com>`

Authors:

- Ariel Chernomoretz `<ariel@df.uba.ar>`

`genotypeProbs` *Genotype probabilities*

Description

Calculate genotype probabilities from parental probabilities

Usage

```
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

```
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` *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` *getAllelesFromGenotypes*

Description

Get alleles from genotypes

Usage

```
getAllelesFromGenotypes(g)
```

Arguments

`g`	genotypes

Value

alleles

`H` *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` *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 *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

```
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

```
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)
```

`Px` *Px*

Description

Px

Usage

```
Px(p1, p0, dbg = FALSE)
```

Arguments

p1	probability distribution
p0	probability distribution
dbg	boolean to compute sigma

Value

Px

runIT *runIT*

Description

run information theory (IT) metrics

Usage

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

Arguments

`lped`	list of pedigree objects
`freqs`	list of allele frequencies
`QP`	QP
`dbg`	debug
`numCores`	number of cores
`bOnlyIT`	boolean to only run IT
`lprobg_ped`	list of probG
`bsigma`	boolean to compute sigma
`blog`	boolean to write log
`dep`	check fbnet dependency

Value

runIT

simLR *Simulate LR*

Description

Simulate LR

Usage

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

Arguments

`lprobg_ped`	list of probability distributions
`numSim`	number of simulations
`epsilon`	small number to avoid log(0)
`bplot`	boolean to plot
`bLRs`	boolean to return LRs
`seed`	seed

Value

LRs

simME *simME: output from simMinimalEnsemble considering an uncle*

Description

simME: output from simMinimalEnsemble considering an uncle

Usage

```
simME
```

Format

A list with minimalEnsemble of genotypes

simMinimalEnsemble *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
`seed`	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

`ped`	pedigree
`testID`	test ID
`numSim`	number of simulations
`seed`	seed

Value

list of simulations

Examples

```
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)
```

`strsplit2` *strsplit2*

Description

strsplit2

Usage

```
strsplit2(x, split)
```

Arguments

`x`	character vector
`split`	character

Value

matrix

`trioCheckFast` *trioCheckFast*

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