Package ‘dvir’

May 18, 2021

Type  Package
Title  Disaster Victim Identification
Version  2.1.0
Description  Joint DNA-based disaster victim identification (DVI), as described in Vigeland and Egeland (2021) <doi:10.21203/rs.3.rs-296414/v1>. Identification is performed by optimising the joint likelihood of all victim samples and reference individuals. Individual identification probabilities, conditional on all available information, are derived from the joint solution in the form of posterior pairing probabilities. ‘dvir’ is part of the ‘ped suite’ collection of packages for pedigree analysis. In particular it uses ‘forrel’ for calculation of likelihood ratios.
License  GPL-3
URL  https://github.com/thoree/dvir
BugReports  https://github.com/thoree/dvir/issues
Encoding  UTF-8
Language  en-GB
Depends  R (>= 3.5.0), pedtools (>= 0.9.7)
Imports  forrel (>= 1.3.0), pedprobr (>= 0.5.0)
LazyData  true
RoxygenNote  7.1.1
NeedsCompilation  no
Author  Thore Egeland [aut, cre] (<https://orcid.org/0000-0002-3465-8885>), Magnus Dehli Vigeland [aut] (<https://orcid.org/0000-0002-9134-4962>)
Maintainer  Thore Egeland <thore.egeland@nmbu.no>
Repository  CRAN
Date/Publication  2021-05-18 11:50:05 UTC
R topics documented:

Bmarginal ......................................................... 2
dataCh4 .......................................................... 3
dataExample481 .............................................. 4
dataExercise497 .............................................. 5
dataExercise498 .............................................. 5
dviCompare ..................................................... 6
dvir ................................................................. 8
eexample1 ....................................................... 8
eexample2 ....................................................... 8
exercise497 ..................................................... 9
expand.grid.nodup ............................................ 10
findUndisputed ............................................... 11
generatePairings ............................................. 12
grave ............................................................. 13
icmp .............................................................. 14
jointDVI .......................................................... 15
ncomb ............................................................ 16
pairwiseLR ..................................................... 17
planecrash ..................................................... 18
sequentialDVI ................................................ 19

Index 21

Bmarginal  Posterior pairing probabilities

Description
Compute posterior pairing and non-pairing probabilities, based on a prior and the output from jointDVI().

Usage
Bmarginal(jointRes, missing, prior = NULL)

Arguments
  jointRes  Output from jointDVI().
  missing   Character vector with names of missing persons.
  prior     A numeric vector of length equal the number of rows in jointRes. Default is a flat prior.

Details
The prior assigns a probability to each assignment, each row of jointRes. If the prior is not specified, a flat prior is used. The prior needs not sum to 1 since the user may rather choose a flat prior on the a priori possible assignments.
Value

A matrix. Row i gives the posterior probability that victim i is one of the missing persons or someone else, denoted '*'.

See Also

jointDVI()

Examples

```r
pm = example1$pm
am = example1$am
missing = example1$missing
jointRes = jointDVI(pm, am, missing)
Bmarginal(jointRes, missing)

# Artificial example: all but optimal solution excluded by prior
Bmarginal(jointRes, missing, prior = c(1, rep(0,26)))
```

Data used in the book Kling et al. (2021)

Description

Data used in last example of Chapter 4 in Kling et al. (2021) "Mass Identifications: Statistical Methods in Forensic Genetics". There are 2 female victims, 2 male victims. There are four reference families with 2 missing females and 2 missing males. There are 21 markers. An 'equal mutation mode with rate 0.005 is specified.

Usage

dataCh4

Format

A list of 3 elements:

- pm: A list of 4 singletons (victims).
- am: A list of 3 pedigrees.
- missing: A vector containing the names of the missing persons.
Examples

```r
pm = dataCh4$pm
am = dataCh4$am
missing = dataCh4$missing

# res = jointDVI(pm, am, missing, disableMutations = FALSE)
# head(res[c(1, 2, 30, 49),])
```

---

**Description**

Data used in Example 4.8.1 in Kling et al. (2021) "Mass Identifications: Statistical Methods in Forensic Genetics". There are victims V1 and V2, both females. There is one reference family with 2 missing persons, both females. There are 21 markers, no mutation model.

**Usage**

```r
dataExample481
```

**Format**

A list of 3 elements:

- **pm**: A list of 2 singletons (victims).
- **am**: A list of 1 pedigree.
- **missing**: A vector containing the names of the missing persons.

**Examples**

```r
pm = dataExample481$pm
am = dataExample481$am
missing = dataExample481$missing

# Find number of assignments
ncomb(2, 2, 0, 0)

# Plot and find joint solution
plotPedList(list(pm, am), marker = 1:2, hatched = typedMembers, 
col = list(red = missing))
jointDVI(pm, am, missing, verbose = FALSE)
```
dataExercise497

Data used in the book Kling et al. (2021)

Description

Data used in Exercise 4.9.7 in Kling et al. (2021) "Mass Identifications: Statistical Methods in Forensic Genetics". There are 3 female victims and 3 reference families with 3 missing females. There are 23 markers, equal mutation model, rate 0.001.

Usage

dataExercise497

Format

A list of 3 elements:

- pm: A list of 3 singletons (victims).
- am: A list of 3 pedigrees.
- missing: A vector containing the names of the missing persons.

dataExercise498

Data used in the book Kling et al. (2021)

Description

Data used in Exercise 4.9.8 in Kling et al. (2021) "Mass Identifications: Statistical Methods in Forensic Genetics". There are 2 female victims and 1 male. There is one reference family with 2 missing females and one missing male. There are 16 markers, equal mutation model, rate 0.001.

Usage

dataExercise498

Format

A list of 3 elements:

- pm: A list of 3 singletons (victims).
- am: A list of 1 pedigree.
- missing: A vector containing the names of the missing persons.
**dviCompare**  
*Compare DVI approaches*

**Description**

Compare the efficiency of different computational approaches to DVI.

**Usage**

```r
dviCompare(
  pm,
  am,
  missing,
  true,
  refs = typedMembers(am),
  methods = 1:6,
  markers = NULL,
  threshold = 1,
  simulate = TRUE,
  db = getFreqDatabase(am),
  Nsim = 1,
  returnSims = FALSE,
  seed = NULL,
  numCores = 1,
  verbose = TRUE
)
```

**Arguments**

- `pm`: PM data: List of singletons
- `am`: AM data: A ped object or list of such.
- `missing`: Character vector with names of the missing persons.
- `true`: A character of the same length as `pm`, with the true solution, e.g., `true = c("M2", ",", "M3) if the truth is V1 = M2 and V3 = M3.
- `refs`: Character vector with names of the reference individuals. By default the typed members of `am`.
- `methods`: A subset of the numbers 1,2,3,4,5,6.
- `markers`: If `simulate = FALSE`: A vector indicating which markers should be used.
- `threshold`: An LR threshold passed on to the sequential methods.
- `simulate`: A logical, indicating if simulations should be performed.
- `db`: A frequency database used for simulation, e.g., `forrel::NorwegianFrequencies`. By default the frequencies attached to `am` are used.
- `Nsim`: A positive integer; the number of simulations.
returnSims: A logical: If TRUE, the simulated data are returned without any DVI comparison.

seed: A seed for the random number generator, or NULL.

numCores: The number of cores used in parallelisation. Default: 1.

verbose: A logical.

Details

The following methods are available for comparison, through the methods parameter:

1. Sequential, without LR updates
2. Sequential, with LR updates
3. Sequential (undisputed) + joint (remaining). Always return the most likely solution(s).
4. Joint - brute force. Always return the most likely solution(s).
5. Like 3, but return winner(s) only if LR > threshold; otherwise the empty assignment.
6. Like 4, but return winner(s) only if LR > threshold; otherwise the empty assignment.

Value

A list of solution frequencies for each method, and a vector of true positive rates for each method.

Examples

```r
pm = example1$pm
am = example1$am
missing = example1$missing
refs = "R1"

db = forrel::NorwegianFrequencies[1:3]

# True solution
ture = c("M1", "M2", "M3")

# Run comparison
dviCompare(pm, am, missing, refs, true = true, db = db, Nsim = 2, seed = 123)

# Alternatively, simulations can be done first...
sims = dviCompare(pm, am, missing, refs, true = true, simulate = TRUE,
          db = db, Nsim = 2, seed = 123, returnSims = TRUE)

# ... and computations after:
dviCompare(sims$pm, sims$am, missing, refs, true = true, simulate = FALSE)
```
dvir

Description

Disaster Victim Identification.

dvi: Disaster Victim Identification

example1

DVI dataset: Generational trio

Description

A proof-of-concept dataset involving three missing members (child, father, grandfather) of a single family. With the given data, stepwise victim identification fails to find the correct solution, while joint identification succeeds.

Usage

example1

Format

A list of 3 elements:

- pm: A list of 3 singletons (victims).
- am: A pedigree with three missing persons and one typed reference individual.
- missing: A vector containing the names of the missing persons.

example2

DVI dataset: Two reference families

Description

A small DVI example with three victims, and three missing persons from two reference families

Usage

example2

Format

A list of 3 elements:

- pm: A list of 3 singletons (victims).
- am: A list of 2 pedigrees with three missing persons and one typed reference individual.
- missing: A vector containing the names of the missing persons.
Examples

    pm = example2$pm
    am = example2$am
    missing = example2$missing
    jointDVI(pm, am, missing)

Solution Exercise 4.9.7 in the book Kling et al. (2021)

Description

This is a DVI case with 3 female victims and 3 missing females in three reference families. There are 23 markers with equal mutation rate 0.001. Data are simulated from the solution V1 = MP1, V2 = MP2, V3 = MP3 and the purpose is to check fraction of times the ‘correct’ solutions is obtained.

Usage

    exercise497(
        pm,
        am,
        missing,
        nsim = 2,
        seed = NULL,
        simRef = TRUE,
        disableMutations = FALSE,
        undisputed = FALSE,
        verbose = FALSE
    )

Arguments

    pm          A list of singletons.
    am          A list of pedigrees.
    missing     Character vector with names of missing persons.
    nsim        Number of simulations.
    seed        Integer.
    simRef      Logical. If TRUE, references are also simulated.
    disableMutations Logical, see jointDVI().
    undisputed  Logical, see jointDVI().
    verbose     A logical.
**Description**

This is a simple extension of `expand.grid()` which removes all combinations with repeated elements.

**Usage**

`expand.grid.nodup(lst)`

**Arguments**

- `lst` A list of vectors.

**Value**

A data frame.

**Author(s)**

Magnus Dehli Vigeland

**See Also**

`expand.grid()`

**Examples**

```r
lst = list(1:2, 1:2)
# Compare
expand.grid.nodup(lst)
expand.grid(lst)
```
Description

This function uses the pairwise LR matrix to find "undisputed" matches between victims and missing individuals. An identification $V_i = M_j$ is called undisputed if the corresponding likelihood ratio $LR_{i,j}$ exceeds the given threshold, while all other pairwise LRs involving $V_i$ or $M_j$ are at most 1.

Usage

```r
findUndisputed(
  pm,
  am,
  missing,
  pairings = NULL,
  threshold = 10000,
  limit = 0,
  check = TRUE,
  verbose = FALSE
)
```

Arguments

- **pm**: PM data: List of singletons.
- **am**: AM data: A ped object or list of such.
- **missing**: Character vector with names of the missing persons.
- **pairings**: A list of possible pairings for each victim. If NULL, all sex-consistent pairings are used.
- **threshold**: A non-negative number. If no pairwise LR exceed this, the iteration stops.
- **limit**: A positive number. Only pairwise LR values above this are considered.
- **check**: A logical, indicating if the input data should be checked for consistency.
- **verbose**: A logical.

Value

A list with the following entries:

- **undisputed**: A list of undisputed matches and the corresponding LR values.
- **pmReduced**: Same as pm, but with the undisputed victims removed.
- **amReduced**: Same as am, but with the data from undisputed victims inserted for the corresponding missing persons.
- **missingReduced**: Same as missing, but without the undisputed identified missing persons.
- **LRmatrix, LList, pairings**: Output from pairwiseLR() applied to the reduced problem.
See Also

`pairwiseLR()`

Examples

```r
pm = planecrash$pm
am = planecrash$am
missing = planecrash$missing

findUndisputed(pm, am, missing, threshold = 1e4)
```

Description

Generate a list of sex-consistent pairings for each victim in a DVI problem. By default, the empty pairing (denoted *) is included for each victim.

Usage

```r
generatePairings(pm, am, missing, includeEmpty = TRUE, ignoreSex = FALSE)
```

Arguments

- `pm`: A list of singletons.
- `am`: A list of pedigrees.
- `missing`: Character vector with names of missing persons.
- `includeEmpty`: A logical. If TRUE (default), the do-nothing symbol (*) is included for each victim.
- `ignoreSex`: A logical.

Value

A list of character vectors. Each vector is a subset of `missing`, plus the character * denoting no pairing.

See Also

`jointDVI()`
Examples

```
pm = list(singleton("V1", sex = 1),
          singleton("V2", sex = 2))

missing = paste0("M", 1:4)
am = list(nuclearPed(children = missing[1:3]),
          nuclearPed(children = missing[4], sex = 2))
generatePairings(pm, am, missing)
```

**grave**  

*DVI dataset: Family grave*

Description

Family grave data in Kling et al. (2021) "Mass Identifications: Statistical Methods in Forensic Genetics". There are 5 female victims and 3 male victims. There is one reference family with 5 missing females and 3 missing males. There are 23 markers, no mutation model.

Usage

```
grave
```

Format

A list of 3 elements:

- **pm**: A list of 8 singletons (victims).
- **am**: A pedigree with 8 missing persons.
- **missing**: A vector containing the names of the missing persons.

Examples

```
pm = grave$pm # The list of missing persons
am = grave$am # The reference family pedigree
missing = grave$missing # The names of the missing persons
plot(am, marker = 1)

# jointDVI(pm, am, missing)
```
DVI dataset: A large reference pedigree

Description

DVI dataset based loosely on the ICMP workshop material http://www.few.vu.nl/~ksn560/Block-III-PartI-KS-ISFG2017.pdf (page 18). There are 3 female victims, 2 male victims and 6 missing persons of both sexes. We have renamed the individuals and simulated data for 13 CODIS markers (see Details).

Usage

icmp

Format

A list of 3 elements:

- pm: A list of 5 singletons (victims).
- am: A reference pedigree with 6 genotyped members and 12 missing persons.
- missing: A vector containing the names of the missing persons.

Details

The 13 markers are, in order: CSF1PO, D3S1358, D5S818, D7S820, D8S1179, D13S317, D16S539, D18S51, D21S11, FGA, TH01, TPOX, and vWA.

Source code for the simulation, and a file containing the allele frequencies, can be found in the data-raw folder of the GitHub repository: https://github.com/thoree/dvir.

Examples

# PM data
icmp$pm

# AM data
icmp$am

# Missing individuals
icmp$missing

# Markers and allele frequencies
db = pedtools::getFreqDatabase(icmp$pm)

db
jointDVI

Description

Victims are given as a list of singletons, and references as a list of pedigrees. All possible assignments are evaluated and solutions ranked according to the likelihood.

Usage

jointDVI(
  pm,
  am,
  missing,
  pairings = NULL,
  assignments = NULL,
  limit = 0,
  undisputed = TRUE,
  markers = NULL,
  threshold = 10000,
  disableMutations = NA,
  numCores = 1,
  check = TRUE,
  verbose = TRUE
)

Arguments

pm: A list of singletons.
am: A list of pedigrees.
missing: Character vector with names of missing persons.
pairings: A list of possible pairings for each victim. If NULL, all sex-consistent pairings are used.
assignments: A data frame containing the assignments to be considered in the joint analysis. By default, this is automatically generated by taking all combinations from pairings.
limit: A positive number. Only pairwise LR values above this are considered.
undisputed: A logical.
markers: A vector indicating which markers should be included in the analysis. By default all markers are included.
threshold: A positive number, passed onto findUndisputed().
disableMutations: A logical, or NA (default). The default action is to disable mutations in all reference families without Mendelian errors.
ncomb

numCores: Integer. The number of cores used in parallelisation. Default: 1.
check: A logical, indicating if the input data should be checked for consistency.
verbose: A logical.

Value

A data frame. Each row describes an assignment of victims to missing persons, accompanied with its log likelihood, the LR compared to the null (i.e., no identifications), and the posterior corresponding to a flat prior.

See Also

pairwiseLR()

Examples

pm = example2$pm
am = example2$am
missing = example2$missing

jointDVI(pm, am, missing)

ncomb: The number of assignments for DVI problem

Description

The number of victims and missing persons of each sex is given. The number of possible assignments, i.e., the number of ways the victims can be identified with the missing persons, is calculated.

Usage

ncomb(nVfemales, nMPfemales, nVmales, nMPmales)

Arguments

nVfemales: Integer. The number of female victims.

nMPfemales: Integer. The number of female missing persons.

nVmales: Integer. The number of male victims.

nMPmales: Integer. The number of male missing persons.

Value

The total number of possible assignments.
Examples

# Example: 3 male victims; 2 male missing persons.
# The number of a priori possible assignments is
m1 = ncomb(0,0,3,2) # 13

# Compare with the complete list of assignments
m2 = expand.grid.nodup(list(V1 = c("*", "M1", "M2"),
                       V2 = c("*", "M1", "M2"),
                       V3 = c("*", "M1", "M2")))
stopifnot(m1 == nrow(m2))

pairwiseLR

Pairwise LR matrix

Description

For a given DVI problem, compute the matrix consisting of pairwise likelihood ratios \( LR_{i,j} \) comparing \( V_i = M_j \) to the null. The output may be reduced by specifying arguments \( \text{limit} \) or \( \text{nkeep} \).

Usage

pairwiseLR(
  pm, am, missing,
  pairings = NULL, limit = 0,
  nkeep = NULL, check = TRUE,
  verbose = FALSE
)

Arguments

pm A list of singletons, the victims.
am A list of pedigrees. The reference families.
missing A character vector with names of missing persons.
pairings A list of possible pairings for each victim. If NULL, all sex-consistent pairings are used.
limit A nonnegative number controlling the \( \text{pairing} \) slot of the output: Only pairings with LR greater or equal to \( \text{limit} \) are kept. If zero (default), pairings with LR > 0 are kept.
nkeep An integer. No of pairings to keep, all if NULL.
check A logical, indicating if the input data should be checked for consistency.
verbose A logical.
Value

A list with 3 elements:

- LRmatrix: A matrix containing the pairwise LR values.
- LRlist: A list of numerical vectors, containing the pairwise LRs in list format.
- pairings: A reduced version of the input pairings, keeping only entries with corresponding LR >= limit. For the default case limit = 0 a strict inequality is used, i.e., LR > 0.

Examples

```r
pm = example1$pm
am = example1$am
missing = example1$missing
pairwiseLR(pm, am, missing)
```

Description

A simulated dataset based on Exercise 3.3 in Egeland et al. "Relationship Inference with Familias and R" (2015).

Usage

planecrash

Format

A list of 3 elements:

- pm: A list of 8 female singletons (victims).
- am: A list of 5 pedigrees, each with one missing member and one genotyped member.
- missing: A vector containing the names of the missing persons.

Details

The 15 markers are CSF1PO, D13S317, D16S539, D18S51, D21S11, D3S1358, D5S818, D7S820, D8S1179, FGA, PENTA_D, PENTA_E, TH01, TPOX, and VWA.

Source code for the simulation, and a file containing the allele frequencies, can be found in the data-raw folder of the GitHub repository: https://github.com/thoree/dvir.
Examples

# PM data
planecrash$pm

# AM data
planecrash$am

# Missing individuals
planecrash$missing

# Markers and allele frequencies
db = pedtools::getFreqDatabase(planecrash$pm)
db

sequentialDVI Sequential DVI search

Description
Sequential DVI search

Usage

sequentialDVI(
  pm,
  am,
  missing,
  updateLR = TRUE,
  threshold = 1,
  check = TRUE,
  verbose = TRUE,
  debug = FALSE
)

Arguments

pm PM data: List of singletons.
am AM data: A ped object or list of such.
missing Character vector with names of the missing persons.
updateLR A logical. If TRUE, the LR matrix is updated in each iteration.
threshold A non-negative number. If no pairwise LR values exceed this, the iteration stops.
check A logical, indicating if the input data should be checked for consistency.
verbose A logical.
debug A logical. If TRUE, the LR matrix is printed
Value

A solution to the DVI problem in the form of an assignment vector.

Examples

```r
pm = example1$pm
am = example1$am
missing = example1$missing

sequentialDVI(pm, am, missing, updateLR = FALSE)
sequentialDVI(pm, am, missing, updateLR = TRUE)

# The output of can be fed into 'jointDVI()':
res = sequentialDVI(pm, am, missing, updateLR = TRUE)
jointDVI(pm, am, missing, assignments = res)
```
Index

* datasets
  dataCh4, 3
dataExample481, 4
dataExercise497, 5
dataExercise498, 5
  example1, 8
  example2, 8
grave, 13
icmp, 14
planecrash, 18

Bmarginal, 2
dataCh4, 3
dataExample481, 4
dataExercise497, 5
dataExercise498, 5
dviCompare, 6
dviar, 8
  example1, 8
  example2, 8
  exercise497, 9
  expand.grid(), 10
  expand.grid.nodup, 10
  findUndisputed, 11
  findUndisputed(), 15
  generatePairings, 12
  grave, 13
  icmp, 14
  jointDVI, 15
  jointDVI(), 2, 3, 9, 10, 12
ncomb, 16
  pairwiseLR, 17
  pairwiseLR(), 12, 16
  planecrash, 18
  sequentialDVI, 19