Package 'fbnet'

October 27, 2022

Title Forensic Bayesian Networks

Version 1.0.1

Description

Open-source package for computing likelihood ratios in kinship testing and human identification cases (Chernomoretz et al. (2021) <doi:10.1016/j.fsir.2020.100132>). It has the core function of the software GENis, developed by Fundación Sadosky. It relies on a Bayesian Networks framework and is particularly well suited to efficiently perform large-size queries against databases of missing individuals (Darwiche (2009) <doi:10.1017/CBO9780511811357>).

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Imports graphics, grDevices, igraph, stats, Rsolnp, assertthat, utils

NeedsCompilation no

Author Ariel Chernomoretz [aut],
Franco Marsico [aut, cre]

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

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2022-10-26 22:15:06 UTC

R topics documented:

`allGenotypes` *Genotype combinations*

Description

Auxiliary functions computing possible genotype combinations in a pedigree. These are not normally intended for end users.

Usage

```
allGenotypes(n)

fast.grid(argslist, as.list = FALSE)

geno.grid.subset(x, partialmarker, ids, chrom, make.grid = T)
```

Arguments

`n`	a positive integer.
`argslist`	a list of vectors.
`as.list`	if TRUE, the output is a list, otherwise a matrix.
`x`	a linkdat object.
`partialmarker`	a marker object compatible with `x`.
`ids`	a numeric with ID labels of one or more pedigree members.
`chrom`	a character, either 'X' or 'AUTOSOMAL'. If missing, the 'chrom' attribute of `partialmarker` is used. If this is also missing, then 'AUTOSOMAL' is taken as the default value.
`make.grid`	a logical. If FALSE, a list is returned, otherwise `fast.grid` is applied to the list before returning it.

Value

`allGenotypes` returns a matrix with 2 columns and `n + n*n(n-1)/2` rows containing all possible (unordered) genotypes at a biallelic locus with alleles `1,2,...{},n`. `fast.grid` is basically a stripped down version of expand.grid.

Examples

```
m = allGenotypes(2)
stopifnot(m == rbind(c(1,1), c(2,2), 1:2))
```

Argentina_STRs *STRs allelic frequencies from Argentina.*

Description

STRs allelic frequencies from Argentina.

Usage

```
Argentina_STRs
```

Format

A data frame with allele frequencies

as.matrix.linkdat *linkdat to matrix conversion*

Description

Converts a `linkdat` object to a matrix (basically following a pre-makeped LINKAGE format), with marker annotations and other info attached as attributes.

Usage

```
## S3 method for class 'linkdat'
as.matrix(x, include.attrs = TRUE, ...)

restore_linkdat(x, attrs = NULL, checkped = TRUE)
```

Arguments

`x`	a `linkdat` object. In `restore_linkdat`: A numerical matrix in LINKAGE format.
`include.attrs`	a logical indicating if marker annotations and other info should be attached as attributes. See value.
`...`	not used.
`attrs`	a list containing marker annotations and other `linkdat` info compatible with `x`, in the format produced by `as.matrix`. If NULL, the attributes of `x` itself are used.
`checkped`	a logical, forwarded to `linkdat`. If FALSE, no checks for pedigree errors are performed.

Details

restore_linkdat is the reverse of as.matrix.

The way linkdat objects are created in paramlink, marker data are stored as a list of marker objects. Each of these is essentially a matrix with various attributes like allele frequencies, map info a.s.o.. This format works well for marker-by-marker operations (e.g. likelihoods and LOD scores), but makes it somewhat awkward to operate 'horizontally', i.e. individual-by-individual, for instance if one wants to delete all genotypes of a certain individual, or rearrange the pedigree in some way.

It is therefore recommended to convert the linkdat object to a matrix first, do the necessary manipulations on the matrix, and finally use restore_linkdat. Attributes are often deleted during matrix manipulation, so it may be necessary to store them in a variable and feed them manually to restore_linkdat using the attrs argument.

With default parameters, restore_linkdat(as.matrix(x)) should reproduce x exactly.

Value

For as.matrix: A matrix with x$nInd rows and 6 + 2*x$nMark columns. The 6 first columns describe the pedigree in LINKAGE format, and the remaining columns contain marker alleles, using the internal (numerical) allele coding and 0 for missing alleles. If include.attrs = TRUE the matrix has the following attributes:

- markerattr (a list of marker annotations)
- available (the availability vector)
- model (the disease model, if present)
- plot.labels (plot labels, if present)
- orig.ids (original individual IDs)

For restore_linkdat: A linkdat object.

See Also

linkdat, as.data.frame.linkdat

---

buildBN *buildBN: a function for building the bayesian network.*

Description

buildBN: a function for building the bayesian network.

Usage

```
buildBN(pbn, QP)
```

Arguments

pbn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
QP	Query Persons Ids

Value

A bayesian network based on pedigree evidence and QP definition.

Examples

```
pbn  <- initBN(toyped)
bnet <- buildBN(pbn,QP=3)
```

---

buildCPTs — *buildCPTs: a function for building conditional probability tables based on pedigree bayesian network.*

Description

buildCPTs: a function for building conditional probability tables based on pedigree bayesian network.

Usage

```
buildCPTs(
  bn,
  bNodePrunning = TRUE,
  bStateRemoval = TRUE,
  bStateRemoval2 = TRUE,
  lumpingParameter = NULL,
  renorm = "row-wise",
  verbose = FALSE
)
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
bNodePrunning	Standard pruning.
bStateRemoval	State based pruning.
bStateRemoval2	State based pruning (model 2).
lumpingParameter	Used for stepwise mutational model.
renorm	If "row-wise" is selected, zero probability is assigned for transitions out of range.
verbose	Computations output.

Value

A bayesian network based on pedigree evidence and QP definition.

Examples

```
pbn  <- initBN(toyped)
bnet <- buildBN(pbn,QP=3)
bn1  <- buildCPTs(bnet)
```

---

convertPedformat — *convertPedformat: a function for converting a pedtools ped onject to a famlink ped object.*

Description

convertPedformat: a function for converting a pedtools ped onject to a famlink ped object.

Usage

```
convertPedformat(x, verbose = FALSE)
```

Arguments

x	A pedtools ped object.
verbose	Function output.

Value

A dataframe with LRs.

---

evidencePrunning — *evidencePrunning: a fuction for pruning instantiated variables.*

Description

evidencePrunning: a fuction for pruning instantiated variables.

Usage

```
evidencePrunning(bn)
```

Arguments

bn	A bayesian network (output of buildBN function).

Value

A preprocessed bayesian network.

`factorHeteroFounders` *factorHeteroFounders: a function for multiplying probabilities in case of heterocigote founders.*

Description

factorHeteroFounders: a function for multiplying probabilities in case of heterocigote founders.

Usage

```
factorHeteroFounders(rresQ, bn)
```

Arguments

`rresQ`	List of CPTs.
`bn`	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.

Value

A dataframe with genotype probabilities.

`Familias2linkdat` *Convert 'Familias' output to linkdat objects*

Description

This function is extracted from paramlik package, not currently maintained. Familias is a widely used program for computations in forensic genetics. The function documented here facilitates the use of `paramlink` for specialized computations which are not implemented in Familias, e.g. conditional simulations.

Usage

```
Familias2linkdat(familiasped, datamatrix, loci)

readFamiliasLoci(loci)

connectedComponentsfb(ID, FID, MID)
```

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.
ID	An integer vector: Individual ID.
FID	An integer vector: ID of father.
MID	An integer vector: ID of mother.

Details

The Familias program represents pedigrees and marker data in a way that differs from paramlink in several ways, mostly because of paramlink's stricter definition of a 'pedigree'. In paramlink, a pedigree must be connected, have numerical IDs, and each member must have either 0 or 2 parentsfb present in the pedigree. None of this is required by FamiliasPedigree objects. The conversion function Familias2linkdat takes care of all of these potential differences: It converts each FamiliasPedigree into a list of connected linkdat objects, additional parentsfb are added where needed, and non-numerical ID labels are stored in the plot.labels slot of the linkdat object(s).

Value

A linkdat object, or a list of such.

Author(s)

Magnus Dehli Vigeland, Thore Egeland

References

Windows Familias is freely available from https://familias.name.

FamiliasLocus *FamiliasLocus: a function for reading Familias locus data.*

Description

FamiliasLocus: a function for reading Familias locus data.

Usage

```
FamiliasLocus(
  frequencies,
  allelenames,
  name,
  MutationModel = "Stepwise",
  MutationRate = 0,
```

```
  MutationRange = 0.5,
  MutationRate2 = 0,
  MutationMatrix,
  Stabilization = "None",
  MaxStabilizedMutrate = 1,
  femaleMutationModel,
  femaleMutationRate,
  femaleMutationRange,
  femaleMutationRate2,
  femaleMutationMatrix,
  maleMutationModel,
  maleMutationRate,
  maleMutationRange,
  maleMutationRate2,
  maleMutationMatrix
)
```

Arguments

`frequencies`	allele frequencies
`allelenames`	names
`name`	system name
`MutationModel`	model
`MutationRate`	rates
`MutationRange`	range
`MutationRate2`	rate two, applied for extended stepwise
`MutationMatrix`	matrix
`Stabilization`	stabilization factor
`MaxStabilizedMutrate`	mix factor
`femaleMutationModel`	for females
`femaleMutationRate`	rate
`femaleMutationRange`	range
`femaleMutationRate2`	rate 2
`femaleMutationMatrix`	females matrix
`maleMutationModel`	male matrix
`maleMutationRate`	male rate
`maleMutationRange`	male range

maleMutationRate2
: rate 2

maleMutationMatrix
: matrix

Value

Locus analysis.

Examples

```
frequencies <- c(0.1, 0.2, 0.3, 0.4)
allelenames <- c("A", "B", "C", "D")
marker <- FamiliasLocus(frequencies, allelenames)
```

FamiliasPedigree *FamiliasPedigree: a function for constructing Familias pedigree format.*

Description

FamiliasPedigree: a function for constructing Familias pedigree format.

Usage

```
FamiliasPedigree(id, dadid, momid, sex)
```

Arguments

id	individual id
dadid	father id
momid	mother id
sex	biological sex

Value

A dataframe with probabilities.

Examples

```
persons <- c("mother", "child", "AF")
sex <- c("female", "female", "male")
ped1 <- FamiliasPedigree(id = persons, dadid = c(NA, "AF", NA),momid = c(NA, "mother", NA), sex=sex)
```

`fbnet` *fbnet: Forensic Bayesian Networks*

Description

'fbnet' is an open source software package written in R statistical languaje. It relies on a Bayesian Networks framework doi: 10.1017/CBO9780511811357. It is particularly well suited to efficiently perform large-size queries against databases of missing individuals. It could interact with the main functionallities of other packages for pedigree analysis. In particular, 'fbnet' imports the 'Familias' software doi: 10.1016/S03790738(00)00147X. In addition 'pedtools', a software for creating and manipulating pedigrees and markers, is supported. 'fbnet' allows computing LRs and obtaining genotype probability distributions for query individual, based on the pedigree data. 'fbnet' implements the complete GENis functionality, a recently published open-source multi-tier information system developed to run forensic DNA databases to perform kinship analysis based on DNA profiles doi: 10.1016/j.fsir.2020.100132.

`getConditional` *getConditional: a function for obtaining the coditional probability tables based on a given evidence.*

Description

getConditional: a function for obtaining the coditional probability tables based on a given evidence.

Usage

```
getConditional(lf)
```

Arguments

`lf`	A list of joint probabilities.

Value

A list of conditioned probabilities.

`getGenotypeTables` *getGenotypeTables: a function for obtaining genotypetables after variable elimination and using available genetic evidence.*

Description

getGenotypeTables: a function for obtaining genotypetables after variable elimination and using available genetic evidence.

Usage

```
getGenotypeTables(bn, resQ, geno = NULL, lqp = NULL)
```

Arguments

`bn`	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
`resQ`	List of CPTs.
`geno`	data.frame with genotypes.
`lqp`	list of individuals genotypes.

Value

A dataframe with genotype probabilities.

`getLocusCPT` *getLocusCPT: a function for obtaining the coditional probability table from a specific locus.*

Description

getLocusCPT: a function for obtaining the coditional probability table from a specific locus.

Usage

```
getLocusCPT(bn, locus, lumpingParameter = NULL, renorm = "row-wise")
```

Arguments

`bn`	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
`locus`	Specified locus.
`lumpingParameter`	Used for stepwise mutational model.
`renorm`	If "row-wise" is selected, zero probability is assigned for transitions out of range.

Value

A bayesian network based on pedigree evidence and QP definition.

Examples

```
pbn  <- initBN(toyped)
bnet <- buildBN(pbn,QP=3)
bn1  <- buildCPTs(bnet)
locCPT <- getLocusCPT(bn1,"M1")
```

`getMAP` *factorHeteroFounders: a function for multiplying probabilities in case of heterocigote founders.*

Description

factorHeteroFounders: a function for multiplying probabilities in case of heterocigote founders.

Usage

```
getMAP(resQ, topn = 3)
```

Arguments

`resQ`	List of CPTs.
`topn`	Format parameter.

Value

A MAP from the probability table.

`getQSetRMP` *getGenotypeTables: a function for obtaining genotypetables after variable elimination and using available genetic evidence.*

Description

getGenotypeTables: a function for obtaining genotypetables after variable elimination and using available genetic evidence.

Usage

```
getQSetRMP(bn, lqp)
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
lqp	list of individuals genotypes.

Value

A dataframe with genotype probabilities.

---

getValuesOut *getValuesOut: a function for getting out variables with zero probability in the bayesian network*

Description

getValuesOut: a function for getting out variables with zero probability in the bayesian network

Usage

```
getValuesOut(cpt, condVar = c())
```

Arguments

cpt	conditional probability table from the bayesian network
condVar	variables from the conditioning table

Value

A processed conditional probability table

---

imposeEvidence *imposeEvidence: a fuction for imposing evidence in the bayesian network.*

Description

imposeEvidence: a fuction for imposing evidence in the bayesian network.

Usage

```
imposeEvidence(bn)
```

Arguments

bn	A bayesian network (output of buildBN function).

Value

A preprocessed bayesian network.

`initBN` *initBN: a function to initialize the bayesian network.*

Description

initBN: a function to initialize the bayesian network.

Usage

```
initBN(ped = NULL, bplotped = FALSE)
```

Arguments

`ped`	A ped object with information of the genotyped members. The ped object must be in Familias format.
`bplotped`	An alternative ped object to be compared.

Value

A bayesian network.

Examples

```
pbn  <- initBN(toyped)
```

`initBN.fromPed` *initBN.fromPed: a function to initialize the bayesian network.*

Description

initBN.fromPed: a function to initialize the bayesian network.

Usage

```
initBN.fromPed(ped, bplotped)
```

Arguments

`ped`	A ped object in Familias format.
`bplotped`	An alternative ped object to be compared.

Value

A bayesian network.

initBN.fromVars *initBN.fromVars: a function to initialize the bayesian network.*

Description

initBN.fromVars: a function to initialize the bayesian network.

Usage

```
initBN.fromVars(bplotped)
```

Arguments

bplotped	An alternative ped object to be compared.

Value

A bayesian network.

is.linkdat *Is an object a linkdat object?*

Description

Functions for checking whether an object is a linkdat object, a singletonfb or a list of such.

Usage

```
is.linkdat(x)

is.singletonfbfb(x)

is.linkdat.list(x)
```

Arguments

x	Any R object.

Details

Note that the singletonfb class inherits from linkdat, so if x is a singletonfb, is.linkdat(x) returns TRUE.

Value

For `is.linkdat`: TRUE if `x` is a linkdat (or singletonfb) object, and FALSE otherwise.
For `is.singletonfbfb`: TRUE if `x` is a singletonfb object, and FALSE otherwise.
For `is.linkdat.list`: TRUE if `x` is a list of linkdat/singletonfb objects.

See Also

`linkdat`

`likelihood.linkdat` *Pedigree likelihood*

Description

Calculates various forms of pedigree likelihoods based on paramlink functions.

Usage

```
likelihood.linkdat(
  x,
  locus1,
  locus2 = NULL,
  theta = NULL,
  startdata = NULL,
  eliminate = 0,
  logbase = NULL,
  loop_breakers = NULL,
  ...
)

likelihood.singleton(x, locus1, logbase = NULL, ...)
```

Arguments

`x`	a `linkdat` object, a singleton object, or a list of such objects. In `likelihood_LINKAGE`, `x` must be a `linkdat` object, with `x$model` different from NULL.
`locus1`	a `marker` object compatible with `x`. If `x` is a list, then `locus1` must be a list of corresponding `marker` objects.
`locus2`	either NULL, the character 'disease', or a markerfb object compatible with `x`. See Details.
`theta`	the recombination rate between locus1 and locus2 (in `likelihood_LINKAGE`: between the marker and the disease locus). To make biological sense theta should be between 0 and 0.5.
`startdata`	for internal use. linkage computations with few-allelic markers.

eliminate	mostly for internal use: a non-negative integer indicating the number of iterations in the internal genotype-compatibility algorithm. Positive values can save time if partialmarker is non-empty and the number of alleles is large.
logbase	a numeric, or NULL. If numeric the log-likelihood is returned, with logbase as basis for the logarithm.
loop_breakers	a numeric containing IDs of individuals to be used as loop breakers. If NULL, automatic selection of loop breakers will be performed.
...	further arguments.

Details

All likelihoods are calculated using the Elston-Stewart algorithm.

Value

The likelihood of the data. If the parameter logbase is a positive number, the output is log(likelihood, logbase).

---

linkdat *Linkdat objects*

Description

This function has been obtained from paramlink package, no longer maintained. Functions to create and display 'linkdat' objects.

Usage

```
linkdat(
  ped,
  model = NULL,
  map = NULL,
  dat = NULL,
  freq = NULL,
  annotations = NULL,
  missing = 0,
  header = FALSE,
  checkped = TRUE,
  verbose = TRUE,
  ...
)

singletonfb(id, sex = 1, famid = 1, verbose = FALSE, ...)

## S3 method for class 'linkdat'
print(x, ..., markers)
```

```
## S3 method for class 'linkdat'
summary(object, ...)

## S3 method for class 'linkdat'
subset(x, subset = x$orig.ids, ..., markers = seq_len(x$nMark))
```

Arguments

ped	a matrix, data.frame or a character with the path to a pedigree file in standard LINKAGE format. (See details)
model	either a linkdat.model object (typically y$model for some linkdat object y), or a single integer with the following meaning: 1 = autosomal dominant; 2 = autosomal recessive; 3 = X-linked dominant; 4 = X-linked recessive. In each of these cases, the disease is assumed fully penetrant and the disease allele frequency is set to 0.00001. If model=NULL, no model is set.
map	a character with the path to a map file in MERLIN format, or NULL. If non-NULL, a dat file must also be given (next item).
dat	a character with the path to a dat file in MERLIN format, or NULL. (Only needed if map is non-NULL.)
freq	a character with the path to a allele frequency file in MERLIN (short) format, or NULL. If NULL, all markers are interpreted as equifrequent.
annotations	a list (of the same length and in the same order as the marker columns in x) of marker annotations. If this is non-NULL, then all of map, dat, freq should be NULL.
missing	the character (of length 1) used for missing alleles. Defaults to '0'.
header	a logical, relevant only if ped points to a ped file: If TRUE, the first line of the ped file is skipped.
checkped	a logical. If FALSE, no checks for pedigree errors are performed.
verbose	a logical: verbose output or not.
...	further arguments.
id, sex	single numerics describing the individual ID and gender of the singletonfb.
famid	a numeric: the family ID of the singletonfb.
x, object	a linkdat object.
markers	a numeric indicating which markers should be included/printed.
subset	a numeric containing the individuals in the sub-pedigree to be extracted. NB: No pedigree checking is done here, so make sure the subset form a meaningful, closed pedigree.

Details

The file (or matrix or data.frame) ped must describe one or several pedigrees in standard LINKAGE format, i.e. with the following columns in correct order:

1 Family id (optional) (FAMID)

2 Individual id (ID),

3 Father id (FID),

4 Mother id (MID),

5 Gender (SEX): 1 = male, 2 = female,

6 Affection status (AFF): 1 = unaffected, 2 = affected, 0 = unknown,

7 First allele of first marker,

8 Second allele of first marker,

9 First allele of second marker,

a.s.o.

Only columns 2-6 are mandatory. The first column is automatically interpreted as family id if it has repeated elements.

Internally the individuals are relabelfbed as 1,2,..., but this should rarely be of concern to the end user. Some pedigree checking is done, but it is recommended to plot the pedigree before doing any analysis.

Details on the formats of map, dat and frequency files can be found in the online MERLIN tutorial: http://csg.sph.umich.edu/abecasis/Merlin/

A singletonfb is a special `linkdat` object whose pedigree contains 1 individual. The class attribute of a singletonfb is `c('singletonfb','linkdat')`

Value

A `linkdat` object, or a list of `linkdat` objects. A linkdat object is essentially a list with the following entries, some of which can be NULL.

`pedigree`	`data.frame` with 5 columns (ID, FID, MID, SEX, AFF) describing the pedigree in linkage format. (NB: Internal labeling used.)
`orig.ids`	the original individual id labels.
`nInd`	the number of individuals in the pedigree.
`founders`	vector of the founder individuals. (NB: Internal labeling used.)
`nonfounders`	vector of the nonfounder individuals (NB: Internal labeling used.)
`hasLoops`	a logical: TRUE if the pedigree is inbred.
`subnucs`	list containing all (maximal) nuclear families in the pedigree. Each nuclear family is given as a vector of the form c(pivot, father, mother, child1, ...), where the pivot is either the id of the individual linking the nuclear family to the rest of the pedigree, or 0 if there are none. (NB: Internal labeling used.)
`markerdata`	a list of `marker` objects.
`nMark`	the number of markers.
`available`	a numeric vector containing IDs of available individuals. Used for simulations and plots.
`model`	a `linkdat.model` object, essentially a list containing the model parameters. See `setModel` for details.
`loop_breakers`	a matrix with original loop breaker ID's in the first column and their duplicates in the second column. This is set by `breakLoopsfb`.

See Also

pedCreate, pedModify, pedParts, setModel

---

markers *Marker functions*

Description

Functions for setting and manipulating marker genotypes for 'linkdat' objects compatible with fbnet. It was extracted from paramlink package, no longer maintained.

Usage

```
markerfb(
  x,
  ...,
  allelematrix,
  alleles = NULL,
  afreq = NULL,
  missing = 0,
  chrom = NA,
  pos = NA,
  name = NA,
  mutmat = NULL
)

addMarkerfb(x, m, ...)

SetMarkersfb(x, m, annotations = NULL, missing = 0)

modifyMarkerfb(x, marker, ids, genotype, alleles, afreq, chrom, name, pos)

getMarkersfb(
  x,
  markernames = NULL,
  chroms = NULL,
  fromPos = NULL,
  toPos = NULL
)

removeMarkersfb(
  x,
  markers = NULL,
  markernames = NULL,
  chroms = NULL,
  fromPos = NULL,
```

```
  toPos = NULL
)

swapGenotypesfb(x, ids)

modifyMarkerfbMatrix(x, ids, new.alleles)
```

Arguments

x	a linkdat object
...	an even number of vectors, indicating individuals and their genotypes. See examples.
allelematrix	a matrix with one row per pedigree member and two columns per marker, containing the alleles for a single marker.
alleles	a numeric or character vector containing allele names.
afreq	a numerical vector with allele frequencies. An error is given if they don't sum to 1 (rounded to 3 decimals).
missing	a numeric - or character - of length 1, indicating the code for missing alleles.
chrom	NA or an integer (the chromosome number of the marker).
pos	NA or a non-negative real number indicating the genetic position (in cM) of the marker.
name	NA or a character (the name of the marker).
mutmat	a mutation matrix, or a list of two such matrices named 'female' and 'male'. The matrix/matrices must be square, with the allele labels as dimnames, and each row must sum to 1 (after rounding to 3 decimals).
m	a marker object or a matrix with alleles. (In SetMarkersfb this matrix can contain data of several markers.)
annotations	a list of marker annotations.
marker, markers	a numeric indicating which marker(s) to use/modify.
ids	a numeric indicating individual(s) to be modified. In swapGenotypesfb this must have length 2.
genotype	a vector of length 1 or 2, containing the genotype to be given the ids individuals. See examples.
markernames	NULL or a character vector.
chroms	NULL or a numeric vector of chromosome numbers.
fromPos, toPos	NULL or a single numeric.
new.alleles	a numerical matrix of dimensions length(ids),2*x$nMark. Entries refer to the internal allele numbering.

Value

The marker function returns an object of class marker: This is a numerical 2-column matrix with one row per individual, and attributes 'alleles' (a character vector with allele names), 'nalleles' (the number of alleles) and 'missing' (the input symbol for missing marker alleles), 'chrom' (chromosome number), 'name' (marker identifier), 'pos' (chromosome position in cM).

For addMarker, SetMarkersfb, removeMarkersfb, modifyMarkerfb, modifyMarkerfbMatrix and swapGenotypesfb, a linkdat object is returned, whose markerdata element has been set/modified.

For getMarkersfb a numeric vector containing marker numbers (i.e. their indices in x$markerdata) for the markers whose 'name' attribute is contained in markernames, 'chrom' attribute is contained in chroms, and 'pos' attribute is between from and to. NULL arguments are skipped, so getMarkersfb(x) will return seq_len(x$nMark) (i.e. all markers).

See Also

linkdat

---

markerSimfb *Marker simulation*

Description

Simulates marker genotypes conditional on the pedigree structure, affection statuses and disease model.

Usage

```
markerSimfb(
  x,
  N = 1,
  available = NULL,
  alleles = NULL,
  afreq = NULL,
  partialmarker = NULL,
  loop_breakers = NULL,
  eliminate = 0,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	a linkdat object
N	a positive integer: the number of markers to be simulated
available	a vector containing IDs of the available individuals, i.e. those whose genotypes should be simulated. By default, all individuals are included.

alleles	a vector containing the alleles for the marker to be simulation. If a single integer is given, this is interpreted as the number of alleles, and the actual alleles as 1:alleles. Must be NULL if partialmarker is non-NULL.
afreq	a vector of length 2 containing the population frequencies for the marker alleles. Must be NULL if partialmarker is non-NULL.
partialmarker	Either NULL (resulting in unconditional simulation), a marker object (on which the simulation should be conditioned) or the index of an existing marker of x.
loop_breakers	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 breakLoopsfb.
eliminate	A non-negative integer, indicating the number of iterations in the internal genotype-compatibility algorithm. Positive values can save time if partialmarker is non-NULL and the number of alleles is large.
seed	NULL, or a numeric seed for the random number generator.
verbose	a logical.

Details

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

Value

a linkdat object equal to x except its markerdata entry, which consists of the N simulated markers.

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

simpleSim, linkage.power

---

mendelianCheckfb *Check for Mendelian errors*

Description

Check marker data for Mendelian inconsistencies

Usage

```
mendelianCheckfb(x, remove = FALSE, verbose = !remove)
```

Arguments

x	a linkdat object
remove	a logical. If FALSE, the function returns the indices of markers found to incorrect. If TRUE, a new linkdat object is returned, where the incorrect markers have been deleted.
verbose	a logical. If TRUE, details of the markers failing the tests are shown.

Value

A numeric containing the indices of the markers that did not pass the tests, or (if `remove=TRUE`) a new `linkdat` object where the failing markers are removed.

---

minOrdering — *minOrdering: a function for getting an ordering of bayesian network variables not in Q using min fill criteria on interaction graphs.*

Description

minOrdering: a function for getting an ordering of bayesian network variables not in Q using min fill criteria on interaction graphs.

Usage

```
minOrdering(bn, vars = NULL, method = c("min_degree", "min_fill")[1])
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
vars	Subset of tables where the order is calculated
method	Elimination method, min_degree or min_fill

Value

A bayesian network after ordering process.

pedigreeLoops *Pedigree loops*

Description

Functions for identifying, breaking and restoring loops in pedigrees.

Usage

```
pedigreeLoops(x)

breakLoopsfb(x, loop_breakers = NULL, verbose = TRUE)

tieLoopsfb(x)

findLoopBreakersfb(x)

findLoopBreakersfb2(x)
```

Arguments

x	a linkdat object.
loop_breakers	either NULL (resulting in automatic selection of loop breakers) or a numeric containing IDs of individuals to be used as loop breakers.
verbose	a logical: Verbose output or not?

Details

Most of paramlink's handling of pedigree loops is done under the hood - using the functions described here - without need for explicit action from end users. When a linkdat object x is created, an internal routine detects if the pedigree contains loops, in which case x$hasLoops is set to TRUE. In analyses of x where loops must be broken (e.g. lod score computation or marker simulation), this is done automatically by calling breakLoopsfb.

In some cases with complex inbreeding, it can be instructive to plot the pedigree after breaking the loops. Duplicated individuals are plotted with appropriate labels (see examples).

The function findLoopBreakersfb identifies a set of individuals breaking all inbreeding loops, but not marriage loops. These require more machinery for efficient detection, and paramlink does this is a separate function, findLoopBreakersfb2, utilizing methods from the igraph package. Since this is rarely needed for most users, igraph is not imported when loading paramlink, only when findLoopBreakersfb2 is called.

In practice, breakLoopsfb first calls findLoopBreakersfb and breaks at the returned individuals. If the resulting linkdat object still has loops, findLoopBreakersfb2 is called to break any marriage loops.

Value

For `breakLoopsfb`, a `linkdat` object in which the indicated loop breakers are duplicated. The returned object will also have a non-null `loop_breakers` entry, namely a matrix with the IDs of the original loop breakers in the first column and the duplicates in the second.

For `tieLoopsfb`, a `linkdat` object in which any duplicated individuals (as given in the `x$loop_breakers` entry) are merged. For any linkdat object `x`, the call `tieLoopsfb(breakLoopsfb(x))` should return `x`.

For `pedigreeLoops`, a list containing all inbreeding loops (not marriage loops) found in the pedigree. Each loop is represented as a list with elements 'top', a 'bottom' individual, 'pathA' (individuals forming a path from top to bottom) and 'pathB' (creating a different path from top to bottom, with no individuals in common with pathA). Note that the number of loops reported here counts all closed paths in the pedigree and will in general be larger than the genus of the underlying graph.

For `findLoopBreakersfb` and `findLoopBreakersfb2`, a numeric vector of individual ID's.

---

pedModify *Modify the pedigree of 'linkdat' objects*

Description

Functions to modify the pedigree of a 'linkdat' object.

Usage

```
addOffspring(
  x,
  father,
  mother,
  noffs,
  ids = NULL,
  sex = 1,
  aff = 1,
  verbose = TRUE
)

removeIndividualsfb(x, ids, verbose = TRUE)

trim(x, keep = c("available", "affected"), return.ids = FALSE, verbose = TRUE)

relabelfb(x, new, old)
```

Arguments

`x`	A `linkdat` object
`father, mother`	Integers indicating the IDs of parentsfb. If missing, a new founder individual is created (whose ID will be 1+the largest ID already in the pedigree).

noffs	A single integer indicating the number of offspringfb to be created.
ids	individuals
sex, aff	Integer vectors indicating the gender and affection statuses of the offspringfb to be created (recycled if less than noffs elements).
verbose	A logical: Verbose output or not.
keep	A character, either 'available' (trimming the pedigree for unavailable members) or 'affected' (trimming for unaffected members).
return.ids	A logical. If FALSE, the trimmed pedigree is returned as a new linkdat object. If TRUE, a vector containing the IDs of 'removable' individuals is returned
new	a numeric containing new labels to replace those in old.
old	a numeric containing ID labels to be replaced by those in new. If missing, old is set to x$orig.ids, i.e. all members in their original order.

Details

When removing an individual, all descendantsfb are also removed as well as founders remaining without offspringfb.

Value

The modified linkdat object.

See Also

linkdat, nuclearPed

---

pedParts *Pedigree subsets*

Description

Utility functions for 'linkdat' objects, mainly for extracting various pedigree information.

Usage

```
offspringfb(x, id, original.id = TRUE)

spousesfb(x, id, original.id = TRUE)

related.pairs(
  x,
 relation = c("parentsfb", "siblingsfb", "grandparentsfbfb", "nephews_niecesfb",
    "cousins", "spousesfb", "unrelatedfb"),
  available = F,
  interfam = c("none", "founders", "all"),
```

```
  ...
)

unrelatedfb(x, id, original.id = TRUE)

leavesfb(x)

parentsfb(x, id, original.id = TRUE)

grandparentsfbfb(x, id, degree = 2, original.id = TRUE)

siblingsfb(x, id, half = NA, original.id = TRUE)

cousins(x, id, degree = 1, removal = 0, half = NA, original.id = TRUE)

nephews_niecesfb(x, id, removal = 1, half = NA, original.id = TRUE)

ancestorsfb(x, id)

descendantsfb(x, id, original.id = TRUE)
```

Arguments

x	a linkdat object. In related.pairs possibly a list of linkdat objects.
id	a numerical ID label.
original.id	a logical indicating whether 'id' refers to the original ID label or the internal labeling.
relation	one of the words (possibly truncated) parentsfb, siblingsfb, grandparentsfbfb, nephews_niecesfb, cousins, spousesfb, unrelatedfb.
available	a logical, if TRUE only pairs of available individuals are returned.
interfam	one of the words (possibly truncated) none, founders or all, specifying which interfamiliar pairs should be included as unrelatedfb in the case where x is a list of several pedigrees. If none, only intrafamiliar pairs are considered; if founders all interfamiliar pairs of (available) founders are included; if all, all interfamiliar (available) pairs are included.
...	further parameters
degree	a non-negative integer.
half	a logical or NA. If TRUE (resp FALSE), only half (resp. full) siblingsfb/cousins/nephews/nieces are returned. If NA, both categories are included.
removal	a non-negative integer

Value

For ancestorsfb(x,id), a vector containing the ID's of all ancestorsfb of the individual id. For descendantsfb(x,id), a vector containing the ID's of all descendantsfb (i.e. children, grandchildren, a.s.o.) of individual id.

The functions `cousins`, `grandparentsfbfb`, `nephews_niecesfb`, `offspringfb`, `parentsfb`, `siblingsfb`, `spousesfb`, `unrelatedfb`, each returns an integer vector containing the ID's of all pedigree members having the specified relationship with `id`.

For `related.pairs` a matrix with two columns. Each row gives of a pair of pedigree members with the specified relation. If the input is a list of multiple pedigrees, the matrix entries are characters of the form 'X-Y' where X is the family ID and Y the individual ID of the person.

For `leavesfb`, a vector of IDs containing all pedigree members without children.

`preparePed` *preparePed: a function for simulating genetic data from untyped individuals conditioned on known genotypes.*

Description

preparePed: a function for simulating genetic data from untyped individuals conditioned on known genotypes.

Usage

```
preparePed(ped, available, lLociFreq, rseed = NULL)
```

Arguments

`ped`	A ped object with information of the genotyped members. The ped object must be in Familias format.
`available`	Genotyped individuals IDs.
`lLociFreq`	Allele frequencies.
`rseed`	Seed used for simulations.

Value

A ped object.

`prodFactor` *prodFactor: a function for performing product between probability tables.*

Description

prodFactor: a function for performing product between probability tables.

Usage

```
prodFactor(laux)
```

Arguments

laux	probability distribution aux

Value

A dataframe with probabilities.

---

pruneNodes *pruneNodes: a fuction for clasical pruning in bayesian networks.*

Description

pruneNodes: a fuction for clasical pruning in bayesian networks.

Usage

```
pruneNodes(bn)
```

Arguments

bn	A bayesian network (output of buildBN function).

Value

A preprocessed bayesian network.

---

removeEvidenceFromPed *removeEvidenceFromPed: a function for removing evidence from specific individuals in a ped object.*

Description

removeEvidenceFromPed: a function for removing evidence from specific individuals in a ped object.

Usage

```
removeEvidenceFromPed(pped, idNotEv)
```

Arguments

pped	A ped object with information of the genotyped members. The ped object must be in Familias format.
idNotEv	A set of individuals whom evidence should be removed.

Value

A ped object.

reportLR *reportLR: a function for calculating the LRs of specified genotypes in a pedigree.*

Description

reportLR: a function for calculating the LRs of specified genotypes in a pedigree.

Usage

```
reportLR(bn, resQ, geno = NULL)
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
resQ	List of CPTs.
geno	data.frame with genotypes.

Value

A dataframe with LRs.

reportPQ *reportPQ: a function for calculating the probability of specified genotypes in a pedigree.*

Description

reportPQ: a function for calculating the probability of specified genotypes in a pedigree.

Usage

```
reportPQ(bn, resQ, geno = NULL)
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
resQ	List of CPTs.
geno	data.frame with genotypes.

Value

A dataframe with genotype probabilities.

reverseSplit *reverseSpit: a function for formatting.*

Description

reverseSpit: a function for formatting.

Usage

```
reverseSplit(inList)
```

Arguments

inList	input for formatting.

Value

A bayesian network.

setAvailable *Functions for modifying availability vectors*

Description

Functions to set and modify the availability vector of a 'linkdat' object. This vector is used in 'linkage.power' and 'linkageSim', indicating for whom genotypes should be simulated.

Usage

```
setAvailable(x, available)

swapAvailable(x, ids)
```

Arguments

x	a linkdat object
available	a numeric containing the IDs of available individuals.
ids	individuals

Value

The modified linkdat object.

See Also

plot.linkdat, linkage.power, linkageSim

setModel *Set, change or display the model parameters for 'linkdat' objects*

Description

Functions to set, change and display model parameters involved in parametric linkage analysis.

Usage

```
setModel(x, model = NULL, chrom = NULL, penetrances = NULL, dfreq = NULL)

## S3 method for class 'linkdat.model'
print(x, ...)
```

Arguments

x	in setModel: a linkdat object. In print.linkdat.model: a linkdat.model object.
model	NULL, or an object of class linkdat.model, namely a list with elements chrom, penetrances and dfreq. In the setModel function, the model argument can be one of the integers 1-4, with the following meanings: 1 = autosomal dominant; fully penetrant, dfreq=1e-5 2 = autosomal recessive; fully penetrant, dfreq=1e-5 3 = X-linked dominant; fully penetrant, dfreq=1e-5 4 = X-linked recessive; fully penetrant, dfreq=1e-5
chrom	a character, either 'AUTOSOMAL' or 'X'. Lower case versions are allowed and will be converted automatically.
penetrances	if chrom=='AUTOSOMAL': a numeric of length 3 - (f0, f1, f2) - where fi is the probability of being affected given i disease alleles. If chrom=='X': a list of two vectors, containing the penetrances for each sex: penetrances = list(male=c(f0, f1), female=c(f0, f1, f2)).
dfreq	the population frequency of the disease allele.
...	further parameters

Value

setModel returns a new linkdat object, whose model entry is a linkdat.model object: A list containing the given chrom, penetrances and dfreq.

See Also

linkdat

setOrdering *setOrdering: a function for selecting the ordering method in the elimination process.*

Description

setOrdering: a function for selecting the ordering method in the elimination process.

Usage

```
setOrdering(bn, ordMethod, vars = NULL, orderElim = NULL)
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
ordMethod	Ordering method.
vars	Vars
orderElim	Order elimination criteria.

Value

A bayesian network after ordering process.

simpleSimfb *Unconditional marker simulation*

Description

Unconditional simulation of unlinked markers

Usage

```
simpleSimfb(
  x,
  N,
  alleles,
  afreq,
  available,
  Xchrom = FALSE,
  mutmat = NULL,
  seed = NULL,
  verbose = T
)
```

Arguments

x	a linkdat object
N	a positive integer: the number of markers to be simulated
alleles	a vector containing the allele names. If missing, the alleles are taken to be seq_along(afreq).
afreq	a vector of length 2 containing the population frequencies for the alleles. If missing, the alleles are assumed equifrequent.
available	a vector containing IDs of the available individuals, i.e. those whose genotypes should be simulated.
Xchrom	a logical: X linked markers or not?
mutmat	a mutation matrix, or a list of two such matrices named 'female' and 'male'. The matrix/matrices must be square, with the allele labels as dimnames, and each row must sum to 1 (after rounding to 3 decimals).
seed	NULL, or a numeric seed for the random number generator.
verbose	a logical.

Details

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

Value

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

See Also

markerSimfb, linkageSim

---

stateRemoval *stateRemoval: a function for processing the bayesian network.*

Description

stateRemoval: a function for processing the bayesian network.

Usage

```
stateRemoval(bn)
```

Arguments

bn	A bayesian network (output of buildBN function).

Value

A preprocessed bayesian network.

`stateRemoval2` *stateRemoval2: a function for processing the bayesian network. It implements another approach from the described in stateRemoval function.*

Description

stateRemoval2: a function for processing the bayesian network. It implements another approach from the described in stateRemoval function.

Usage

```
stateRemoval2(bn, verbose = FALSE)
```

Arguments

bn	A bayesian network (output of buildBN function).
verbose	Computation output.

Value

A preprocessed bayesian network.

`stateRemovalSubnucs` *stateRemovalSubnucs: a fuctiong for variable state pruning.*

Description

stateRemovalSubnucs: a fuctiong for variable state pruning.

Usage

```
stateRemovalSubnucs(bn, verbose = FALSE)
```

Arguments

bn	A bayesian network (output of buildBN function).
verbose	Computation output.

Value

A preprocessed bayesian network.

sumFactor *prodFactor: a function for performing sum between probability tables.*

Description

prodFactor: a function for performing sum between probability tables.

Usage

```
sumFactor(cpt, Z)
```

Arguments

cpt	Conditional probability table
Z	factor

Value

A dataframe with probabilities.

toybase *Toy allele frequency database.*

Description

Toy allele frequency database.

Usage

```
toybase
```

Format

A data frame two markers allele frequencies

toyped *STRs allelic frequencies from specified country.*

Description

STRs allelic frequencies from specified country.

Usage

```
toyped
```

Format

A toy pedigree. Nuclear family.

velim.bn *velim.bn: a function for variable elimination in a bayesian network.*

Description

velim.bn: a function for variable elimination in a bayesian network.

Usage

```
velim.bn(
  bn,
  ordMethod = c("id", "min_degree", "min_fill", "fixed")[2],
  orderElim = NULL,
  verbose = FALSE
)
```

Arguments

bn	A bayesian network for pedigree object with information of the genotyped members. The ped object must be in Familias format.
ordMethod	Selected ordering method between id, min_degree, min_fill and fixed.
orderElim	Elimination order.
verbose	Computation output.

Value

Variable elimination result.

Examples

```
pbn  <- initBN(toyped)
bnet <- buildBN(pbn,QP=3)
bn1  <- buildCPTs(bnet)
resQ <- velim.bn(bn1,ordMethod="min_fill",verbose=FALSE)
```

Index