Package ‘relSim’

October 14, 2022

Type Package

Title Relative Simulator

Version 0.3-4

Date 2022-09-23

Author James M. Curran

Maintainer James M. Curran <j.curran@auckland.ac.nz>

Description A set of tools to explore the behaviour statistics used for forensic DNA interpretation when close relatives are involved. The package also offers some useful tools for exploring other forensic DNA situations.

Encoding UTF-8

License GPL (>= 2)

Imports stats, graphics, xtable, multicool, utils, rvest, stringr, xml2, methods

Depends R (>= 2.10)

LinkingTo Rcpp

LazyData true

RoxygenNote 7.2.1

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-09-26 23:10:02 UTC

R topics documented:

`allPairsLR` *Compute the likehood ratio for all pairs of profiles in a database*

Description

This function takes every pair of profiles in a database of profiles and computes the likelihood ratio (LR) for a specific relationship given by `nCode`. That means there will be $N(N-1)/2$ LRs computed for N profiles.

Usage

```
allPairsLR(Profiles, listFreqs, nCode)
```

Arguments

`Profiles`	an integer vector of stacked profiles representing the database. This vector has $2NL$ entries, where N is the number of profiles and L is the number of loci.
`listFreqs`	is a set of allele frequencies representing a particular multiplex. The function assumes that that loci in the profiles are in the same order as the loci in this list. The data structure is a `List` of `NumericVector`'s.
`nCode`	if `1` then compute the LR for siblings, otherwise computer the LR for parent/child.

Value

a `NumericVector` containing the LRs. They are stored in sequential order so if for example there were three profiles, then there are 3 possible LRs, and the result vector would contain the LRs for the profile pairs (1, 2), (1, 3), and (2, 3).

Author(s)

James Curran

Examples

```
data("USCaucs")
N = 600
profs = relSim:::.randomProfiles(USCaucs$freqs, N)
system.time({lr = relSim:::allPairsLR(profs, USCaucs$freqs, 1)})
plot(density(log10(lr)))
mean(lr > 1) ## estimate the probability that the LR is incorrectly above 1
```

`blockSim` *Perform relatives simulations using large memory blocks in C*

Description

Generate N pairs with a given relationship, calculate the LR for sibs, parent-child and the number of matching alleles and count the number of pairs that meet the threshold criteria.

Usage

```
blockSim(
  N,
  Freqs,
  rel = "UN",
  ibsthresh = NULL,
  kithresh = NULL,
  code = 1,
  falseNeg = TRUE,
  BlockSize = N/10,
  showProgress = FALSE
)
```

Arguments

`N`	The number of iterations to carry out
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.
`rel`	generate unrelated (`rel = 'UN'`), full-sibs (`rel = 'FS'`), or parent child (`rel = 'PC'`) pairs
`ibsthresh`	A vector of one or more IBS thresholds
`kithresh`	A vector of one or more KI/LR thresholds
`code`	A code from 1 to 6 which dictates the events that will be counted. 1. the LR for siblings will be compared to the values in `kithresh` and incremented if the LR is greater than the threshold 2. the LR for parent/child will be compared to the values in `kithresh` and incremented if the LR is greater than the threshold 3. the number of matching alleles (IBS) will be compared to the values in `ibsthresh` and incremented if the IBS is greater than the threshold 4. the LR for siblings and the number of matching alleles will be compared to the values in `kithresh` and `ibsthresh` and incremented if both the LR and IBS is greater than the thresholds. `ibsthresh` and `kithresh` must be of equal length for this option to work 5. the LR for parent/child and the number of matching alleles will be compared to the values in `kithresh` and `ibsthresh` and incremented if both the LR and IBS is greater than the thresholds. `ibsthresh` and `kithresh` must be of equal length for this option to work 6. this option is equivalent to performing code 4 and 5 simulataneously. It is not currently implemented
`falseNeg`	if TRUE then the number of results that DO NOT satisfy the conditions are counted, otherwise the number of results DO satisfy the conditions are counted
`BlockSize`	Sets the number of random profiles to be generated in each iteration. By default the block size is set to 10 percent of the total sample size. It is unclear whether the procedure is more efficient if a bigger percentage of the total is used. Users must take care to make sure that the block size evenly divides `N` otherwise the procedure will exit. Users must also make sure that they have enough memory.
`showProgress`	If `TRUE` then a progress bar will be displayed in the console showing the progress of the simulation.

Details

This function is used for fast accurate estimation of false positive and false negative rates. It achieves part of its speed by block exectution in C, and part by not saving the LR or IBS results. It can do 1 billion iterations in about an hour.

Value

A vector containing the number of profile pairs that satisfied the threshold conditions

Author(s)

James M. Curran

See Also

sim

Examples

```
## not run
## this counts the number of unrelated pairs that are falsely identified
## as siblings using the policy that there are 16 or more matching
## alleles, and the LR/KI is greater than 100,000
## this is a very rare event for the FBI Caucasians with a frequency of
## about 4-5 times in 10 million pairs
## Not run:
data(fbiCaucs)
N = 1e8
ki = 1e5
ibs = 16
code = 5
BlockSize = 1e6
blockSim(N, fbiCaucs, rel = "UN", ibsthresh = ibs, kithresh = ki,
         code = code, falseNeg = FALSE, BlockSize = BlockSize)

## End(Not run)
```

`breedFst` *Breed a population with an approximate level of $\theta(F\_ST)$*

Description

This function simulates a population with an approximate level of population substructure. This is achieved by subdividing a population into equal sized subpopulations and allowing them to breed within themselves for

$$t =$$

$$\lceil \frac{\log_e(1-\theta)}{\log\left(1-\frac{1}{2N_s}\right)} \rceil$$

generations, where N_s is the number of individuals in each subpopulation. This will produce a population with an estimated coancestry coefficient approximately equal to θ

Usage

```
breedFst(Freqs, theta = 0.01, N = 10000, ns = 10, DNAtools = FALSE)
```

Arguments

`Freqs`	A list with an element, `freqs` which contains a list of vectors, where each vector is a set of allele frequencies for a locus
`theta`	A desired level of inbreeding, where $0 < \theta < 0.5$
`N`	Total population size
`ns`	The number of subpopulations. N/n_s needs to be greater than 100
`DNAtools`	If `TRUE` then the profiles in the return population will be formatted as a data frame with an id column and two columns per locus.

Value

An object of class 'population' which is a list with the following elements

- `profiles` - a vector of profiles where the level of inbreeding is approximately equal to θ
- `nProfiles` - the total number of individuals in the population
- `nSubpops` - the number of sub-populations in the population
- `nLoci` - the number of loci each individual is typed at
- `theta` - the desired level of substructure in the population. The actual value will be near to this.
- `Freqs` - a Freq object representing the ancestral frequencies of the population

Author(s)

James M. Curran

Examples

```
data(USCaucs)
pop = breedFst(USCaucs)
```

`calcFst` *Caculate locus-wise and population F_ST values*

Description

This procedure uses the method of Weir and Cockerham to estimate θ (F_{ST}) for a population with substructure

Usage

```
calcFst(Pop, subPopIdx = NULL)
```

Arguments

`Pop`	An object type 'population'
`subPopIdx`	If this vector is not null, then it must consist of N elements with values from 1 to n_s representing which subpopulation each member of `Pop$profiles` belongs to. If it is null then it is assumed that the population consists of n_s subpopulations of equal size N_s so that $n_s \times N_s = N$

Value

A vector of length $n_{loci} + 1$ with locus-wise θ values and an overall θ value for the population

Author(s)

James M. Curran

References

Weir, B.S., Genetic Data Analysis II, (1996) p.173–179, Sinauer, Sunderland, MA.

See Also

breedFst

Examples

```
data(USCaucs)
p = breedFst(USCaucs)
fst = calcFst(p)
fst
```

calcFStats *Calculate locus-wise and population $\theta = F\_ST$, $F = F\_IT$, and $f = F\_IS$ values*

Description

This procedure uses the method of Weir and Cockerham to estimate $\theta = F_{ST}$, $F = F_{IT}$, and $f = F_{IS}$ for a population with known substructure

Usage

```
calcFStats(Pop, subPopIdx = NULL)
```

Arguments

`Pop`	An object type 'population'
`subPopIdx`	If this vector is not null, then it must consist of N elements with values from 1 to n_s representing which subpopulation each member of `Pop$profiles` belongs to. If it is null then it is assumed that the population consists of n_s subpopulations of equal size N_s so that $n_s \times N_s = N$

Value

A vector of length $n_{loci} + 1$ with locus-wise θ values and an overall θ value for the population

Author(s)

James M. Curran

References

Weir, B.S., Genetic Data Analysis II, (1996) p.173–179, Sinauer, Sunderland, MA.

See Also

breedFst

Examples

```
data(USCaucs)
set.seed(123)
p = breedFst(USCaucs)
fstats = calcFStats(p)
fstats
```

checkFreqs — *Make sure that the frequencies are such*

Description

Checks whether a list of frequencies at a series of genetic loci both sum to one and lie between 0 and 1.

Usage

```
checkFreqs(Freqs)
```

Arguments

`Freqs`	A list containg elements `loci` and `freqs`. `freqs` is a list of vectors containing the frequencies at the given loci.

Details

If a locus fails to sum to one, or there are alleles which fall below zero or above one, then a warning message will be returned for each item in error.

Author(s)

James M. Curran

See Also

normalizeFreqs

Examples

```
data(fbiCaucs)
checkFreqs(fbiCaucs)

## induce an error
fbiCaucs$freqs[[1]] = runif(10)
checkFreqs(fbiCaucs)
```

errorRate	*Returns the false positive or false negative rates for a set of IBS and/or KI thresholds*

Description

This function is used to calcalate the various tables in the work of Ge et al. and Balding et al. Specifically it can be used to calculate the false positive rate for unrelated pairs being identified as full-sibs or parent-child pairs under differing levels of IBS or KI (or both) thresholds. It can also be used to calculate the false negative rates for full-sib, or parent-child, pairs being identified as unrelated, again with differing levels of IBS, KI or both.

Usage

```
errorRate(
  simResults,
  bIBS = TRUE,
  bKI = FALSE,
  rel = "UN",
  IBSthresh = 14:17,
  KIthresh = c(1000, 10000, 1e+05, 1e+06),
  nLoci = 13
)
```

Arguments

`simResults`	A data.frame with three columns labelled sib, pc and ibs. This will usually be obtained from a call to `sim` or `readResults`.
`bIBS`	If `TRUE` then IBS thresholds are used to generate the error rates. If both `bIBS` and `bKI` are `TRUE` then both criteria are used.
`bKI`	If `TRUE` then KI thresholds are used to generate the error rates. If both `bIBS` and `bKI` are `TRUE` then both criteria are used.
`rel`	The relationship used in the simulation. Must be one of 'UN', 'FS' or 'PC'.
`IBSthresh`	A vector of IBS values that can be used to classify the results as being related (or not).
`KIthresh`	A vector of KI threshold values that can be used to classify the results as being related (or not).
`nLoci`	The number of loci being used in the multiplex. This dictates the upper bound on the IBS values.

Value

A vector (or a two-column matrix) of false negative or false positive rates. If the relationship is 'UN' then false positive rates are returned for parent-child and full-sibs, with parent-child being in column 1 and full-sibs in column 2. If the relationship is 'PC' then the false negative rate is returned for parent-child pairs, and if it is 'FS' then the false negative rate for full-sibs.

Author(s)

James M. Curran

See Also

sim, readResults

Examples

```
## not run
## Not run: data(fbiCaucs)
unrel = sim(10000)
errorRate(unrel)

## End(Not run)
```

`exclusionPower` *Calculate the exclusion power of a multiplex by locus*

Description

Calculates the exclusion power

$$1 - 2\left(\sum_{i=1}^{n_l} p_i^2\right)^2 - 4\sum_{i=1}^{n_l} p_i^4$$

at each locus for a set of allele frequencies.

Usage

```
exclusionPower(Freqs)
```

Arguments

`Freqs`	A list containing two vectors and a list, called loci, counts, and freqs. The elements of loci are the loci present in the multiplex. The elements are freqs a vectors of allele frequencies for the locus. The elements of counts are irrelevant here.

Value

The exclusion power for each locus.

Author(s)

James M. Curran

References

NRC II, Evaluation of Forensic Evidence, (1996), p.96, National Academy Press.

Examples

```
data(USCaucs)
ep(USCaucs)

## get the multiplex wide exclusion power
1 - prod(1-ep(USCaucs))
```

`famSearch` *Search a database for siblings or children*

Description

This function searches a database of profiles for either a sibling or a child

Usage

```
famSearch(profiles, siblings, children, listFreqs, step)
```

Arguments

`profiles`	an integer vector of stacked profiles representing the database. This vector has $2NL$ entries, where N is the number of profiles and L is the number of loci.
`siblings`	an integer vector of stacked profiles representing the siblings of the profiles in database. The first entry is a sibling of the first entry in `profiles` and so on. This vector has $2NL$ entries, where N is the number of profiles and L is the number of loci.
`children`	an integer vector of stacked profiles representing the children of the profiles in database. The first entry is a child of the first entry in `profiles` and so on. This vector has $2NL$ entries, where N is the number of profiles and L is the number of loci.
`listFreqs`	is a set of allele frequencies representing a particular multiplex. The function assumes that that loci in the profiles are in the same order as the loci in this list. The data structure is a `List` of `NumericVector`'s.
`step`	A step size for progress reporting, i.e. print out progress every `step` iterations. If `step` = `-1`, then there is no printing.

Value

a `List` containing two dataframes, one called `sibs` and one called `children`. Each dataframe has results from searching for either the sibling or the child in the database. For each entry there is a record of which profile gave the highest LR (and its value), and the position of the actual sibling or parent/child in the database (and its respective LR).

Author(s)

James Curran

`fbiCaucs` *CODIS STR Loci allele frequency data*

Description

This data structure

Format

This data set is a list which has two sub-lists. The lists are named loci and freqs. loci is a vector of the 13 CODIS STR locus names. freqs is a list of 13 vectors, each vector contains the allele frequencies published for US Caucasians in Budowle et al. (2001).

Author(s)

James M. Curran

References

Budowle B, Shea B, Niezgoda S, Chakraborty R. (2001), *CODIS STR loci data from 41 sample populations*, J. Forensic Sci. 46:453-89.

See Also

USCaucs

Examples

```
data(fbiCaucs)
names(fbiCaucs)
fbiCaucs$loci
names(fbiCaucs$freqs)
fbiCaucs$freqs[[1]]
names(fbiCaucs$freqs[[1]])
fbiCaucs$freqs[[1]][1]
```

fetchBMdata *Retrieve data from Budowle and Moretti (1999) from the web*

Description

Retreives the Budowle and Moretti (1999) and compiles the allele frequency tables needed for the other parts of this package such as `sim`.

Usage

```
fetchBMdata(url = NULL, id = NULL)
```

Arguments

`url`	- the location of the webpage this data is stored on. If `NULL` then hardcoded values in the function are used. The argument allows the user to get the function to work given this values may change. However, it is unlikely that it will help.
`id`	- the id of the HTML element where the data is stored on the webpage. If `NULL` then a hardcoded value in the function is used. The argument allows the user to get the function to work given this values may change. However, it is unlikely that it will help. NOTE: this is super flakey because of how the FBI web authors have created it. This may change in which case, the function will more likely change than the id.

Details

The first three populations have data on 20 loci, the second three on 13 loci. The missing values (0's in the raw data) have been dropped and are not used in calculating the frequencies. This function will not work if you are not connected to the internet, or access to the internet is blocked.

Value

A list consisting of six elements corresponding to the six populations detailed in the data set. Each of the list elements is a list in itself with three further elements named `loci`, `profiles` and `freqs`. `loci` is a vector of the 13-20 STR locus names. `freqs` is a list of 13-20 vectors, each vector contains the allele frequencies. `profiles` contains the raw profiles that the allele frequency tables were constructed from.

Author(s)

James M. Curran

References

Budowle, B. and Moretti, T.R. (1999), *Genotype Profiles for Six Population Groups at the 13 CODIS Short Tandem Repeat Core Loci and Other PCR Based Loci*, Forensic Science Communications 1(2).

See Also

fbiCaucs, USCaucs

Examples

```
## not run
## Not run:
db = fetchBMdata()
names(db)
f = db[["TRINIDADIAN"]]$freqs
dbExpect(f, k = "UN", collapse = TRUE)

## End(Not run)
```

IBS *Identity by state*

Description

Calculates the total number of alleles that are shared by two profiles. If the two profiles in question are indeed relatives then the matching alleles may be identical by descent, or by random chance alone, hence identity by state.

Usage

```
IBS(prof1, prof2, nLoci = length(prof1)/2, bPrint = FALSE)
```

Arguments

`prof1`	A matrix consisting of 2 columns and nLoci rows. Each entry in the matrix is the (coded) allele held by the individual.
`prof2`	See `prof1`
`nLoci`	The number of loci in the profiles. Specifying this value speeds up computation enormously.
`bPrint`	If true then the result is printed locus by locus. This feature exists primarily for debugging purposes.

Value

An integer between 0 and 2*nLoci representing the total number of alleles that match in the two profiles.

Author(s)

James M. Curran

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
C1 = randomChild(P1, fbiCaucs)
IBS(P1, C1)
IBS(P1, C1, bPrint = TRUE)
```

`IS` *Use importance sampling to determine the probability of m peaks from n contributors to a mixture*

Description

WARNING: This function is experimental.

Usage

```
IS(
  Freqs,
  numContributors = 4,
  maxPeaks = NULL,
  numIterations = 100,
  bTail = FALSE
)
```

Arguments

`Freqs`	a set of allele frequencies. The format can be found in `readFreqs`
`numContributors`	the number of contributors to each mixture. Must be >= 1.
`maxPeaks`	either the number of peaks observed in the mixture or such that 1 <=1 maxPeaks <= min(2 * numContribuors, numAlleles). That is, if used in this way maxPeaks must be between 1 and the smaller of twice the number of contributors or the number of possible alleles because you cannot see more peaks than there are possible alleles.
`numIterations`	the number of iterations to use in the importance sampling scheme.
`bTail`	if `TRUE` then the tail probability is calculated.

Value

a list with as many elements as loci. If tail probabilities are selected then each locus element will be a vector of probabilities

Examples

```
## Not run:
data(USCaucs)
IS(USCaucs, numContributors = 4, maxPeaks = 3, numIterations = 1e4)

## End(Not run)
```

`locusIBS` *Identity by state at a locus*

Description

Calculates the number of alleles that are shared by two profiles at a single locus. If the two profiles in question are indeed relatives then the matching alleles may be identical by descent, or by random chance alone, hence identity by state.

Usage

```
locusIBS(profMat)
```

Arguments

`profMat`	A matrix consisting of 4 columns and N rows. Each row in the matrix consists of the genotypes of two individuals.

Value

A vector of length N containing values 0, 1, or 2 depending on how many alleles each pair of profiles share at a locus.

Author(s)

James M. Curran

Examples

```
data(fbiCaucs)
G = randomSample(1, fbiCaucs, rel = 'FS', N = 1000)
ibs = locusIBS(G)
barplot(tabulate(ibs+1, nbins = 3))
```

lrMix *Calculate locuswise likelihood ratios for two person victim/suspect mixtures*

Description

Calculates the likelihood ratio for pairs of profiles under the propositions $H_p : \quad V + S$ and $H_d : \quad V + U$, where V, S and U are the victim, the suspect and someone unrelated to the suspect respectively. The calculation does not employ θ so there are no assumptions about the subpopulations of the contributors.

Usage

```
lrMix(profiles, Freqs)
```

Arguments

`profiles`	A vector of profile lists, from `randomProfilePairs`. `randomPCPairs` and `randomSibPairs` also work but should not really be used as the calculations do not take account of the relationship between the two individuals.
`Freqs`	A list containing elements `freqs`, `loci` and `counts`. The element `freqs` is a list of vectors of allele frequencies at the loci listed in `loci`. These frequencies are used to evaluate the LR

Value

A matrix of LRs calculated at each locus for every pair of profiles. Note this is the set of N profile pairs supplied in `profiles`, not a pairwise comparison.

Author(s)

James M. Curran

Examples

```
data(USCaucs)
p = randomProfilePairs(USCaucs, 10000)
log.lrs = log10(lrMix(p, USCaucs))
boxplot(log.lrs, las = 2)
```

`lrPC` *Likelihood Ratio for Parent-Child / Paternity Index*

Description

Calculates Likelihood Ratio comparing the probability of two profiles if they are indeed parent-child compared to unrelated. This is the paternity index or PI.

Usage

```
lrPC(parent, child, Freqs = NULL, nLoci = length(parent)/2, f = NULL, n = NULL)
```

Arguments

`parent`	A matrix consisting of 2 columns and nLoci rows. Each entry in the matrix is the (coded) allele held by the individual. This represents the alleged parent. The relationship is reflexive so it does not matter which profile is labelled parent and child.
`child`	See `parent`
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex. This argument or both f and n must be specified
`nLoci`	The number of loci in the profiles
`f`	A concatenated vector of allele frequencies. Specifying this speeds up computation enormously
`n`	A vector of length `nLoci` giving the number of alleles at each locus. Specifying this in advance enormously speeds up computation

Value

A value between 0 and infinity representing support (or lack of support if the value is less than 1) for the hypothesis that the two profiles are parent and child. There is no mutation built into this calculation. This means that the LR will be zero if the profiles do not share at least one allele in common at each locus in the multiplex.

Author(s)

James M. Curran

References

Buckleton, J, Triggs, C.M., and Walsh, S.J. (2005)*Forensic DNA Evidence Interpretation*, CRC Press., Boca Raton, FL. p.410

See Also

lrSib, IBS

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
C1 = randomChild(P1, fbiCaucs)
lrPC(P1, C1, fbiCaucs)
```

`lrSib` *Likelihood Ratio / Kinship Index for full-siblings*

Description

Calculates Likelihood Ratio comparing the probability of two profiles if they are indeed full-sibs compared to unrelated. This is sometimes called the kinship index (KI) for full-sibs.

Usage

```
lrSib(sib1, sib2, Freqs = NULL, nLoci = length(sib1)/2, f = NULL, n = NULL)
```

Arguments

`sib1`	A matrix consisting of 2 columns and nLoci rows. Each entry in the matrix is the (coded) allele held by the individual. This represents the alleged sibling. The relationship is reflexive so it does not matter which profile is labelled sib1 and sib2.
`sib2`	See `sib1`
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex. This argument or both f and n must be specified
`nLoci`	The number of loci in the profiles
`f`	A concatenated vector of allele frequencies. Specifying this speeds up computation enormously
`n`	A vector of length `nLoci` giving the number of alleles at each locus. Specifying this in advance enormously speeds up computation

Value

A value between 0 and infinity representing support (or lack of support if the value is less than 1) for the hypothesis that the two profiles are full-siblings. There is no mutation built into this calculation.

Author(s)

James M. Curran

References

Buckleton, J, Triggs, C.M., and Walsh, S.J. (2005)*Forensic DNA Evidence Interpretation*, CRC Press., Boca Raton, FL. p.411

See Also

lrSibDebug, lrPC, IBS

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
S1 = randomSib(P1, fbiCaucs)
P2 = randomProfile(fbiCaucs)
lrSib(P1, S1, fbiCaucs)
lrSib(P1, P2, fbiCaucs)
```

`lrSibDebug` *Likelihood Ratio / Kinship Index for full-siblings*

Description

Calculates Likelihood Ratio comparing the probability of two profiles if they are indeed full-sibs compared to unrelated. This is sometimes called the kinship index (KI) for full-sibs. This function is identical to lrSib except that the calculation is performed in R, and provides full calculation detail at each locus. It exists primarily to check that the correct formula and logic is being applied in the LR calculation so that the result can be manually verified.

Usage

```
lrSibDebug(sib1, sib2, Freqs)
```

Arguments

`sib1`	A matrix consisting of 2 columns and nLoci rows. Each entry in the matrix is the (coded) allele held by the individual. This represents the alleged sibling. The relationship is reflexive so it does not matter which profile is labelled sib1 and sib2.
`sib2`	See `sib1`
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.

Value

A list containing three elements Lines, lr, and Cases. Lines is a list of strings containing the calculation at each locus so that the result can be written to file for example. Cases is a numeric code listing which logical case (1-11) the locus falls into for the profiles in question. lr is the KI for full-sibs for the two profiles.

Author(s)

James M. Curran

References

Buckleton, J, Triggs, C.M., and Walsh, S.J. (2005)*Forensic DNA Evidence Interpretation*, CRC Press., Boca Raton, FL. p.411

See Also

lrSib, lrPC, IBS

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
S1 = randomSib(P1, fbiCaucs)
P2 = randomProfile(fbiCaucs)
cat(paste(lrSibDebug(P1, S1, fbiCaucs)$Lines))
cat(paste(lrSibDebug(P1, P2, fbiCaucs)$Lines))
```

`normalizeFreqs` *Normalize frequencies to 1*

Description

Normalize a list of frequencies at a series of genetic loci both sum to one. Not that this does not deal with the problem of values larger than one or smaller than zero.

Usage

```
normalizeFreqs(Freqs)
```

Arguments

`Freqs`	A list containg elements `loci` and `freqs`. `freqs` is a list of vectors containing the frequencies at the given loci.

Details

Divides vector in Freqs$freqs by the vector sum.

Value

A list containg elements `loci` and `freqs`. `freqs` is a list of vectors containing the frequencies at the given loci.

Author(s)

James M. Curran

See Also

checkFreqs

Examples

```

data(fbiCaucs)

## induce an error
fbiCaucs$freqs[[1]] = rgamma(10,1,1)
checkFreqs(fbiCaucs)
fbiCaucs = normalizeFreqs(fbiCaucs)
checkFreqs(fbiCaucs)
```

`print.population` *Print summary details of a substructed population*

Description

Nicely prints summary information about a substructured population created using `breedFst`

Usage

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

Arguments

`x`	The population object to be printed
`...`	Ignored - really should be passed to print, but given cat is actually called they are ignored

Author(s)

James M. Curran

See Also

breedFst

Examples

```
data(fbiCaucs)
p = breedFst(fbiCaucs)
print(p)
```

---

print.profile *Print a DNA profile*

Description

Nicely prints a profile object out in genotype pairs

Usage

```
## S3 method for class 'profile'
print(x, horizontal = FALSE, ...)
```

Arguments

x	The profile object to be printed
horizontal	if TRUE then the profile will print on a single line instead of multiple lines. Useful for comparing two profiles
...	Ignored - really should be passed to print, but given cat is actually called they are ignored

Author(s)

James M. Curran

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
P2 = randomProfile(fbiCaucs)
P1
print(P1, horizontal = TRUE)
print(P2, horizontal = TRUE)
```

`randomChild` *Generate a random child from a given DNA profile and a given set of allele frequencies*

Description

Generates a random child (or parent) from a given DNA profile from a given set of allele frequencies. At each locus, the child inherits the first allele of the given profile with one half, or the second allele with probability one half. The second allele is chosen at random with probability proportional to the allele frequencies.

Usage

```
randomChild(profile, Freqs)
```

Arguments

`profile`	A vector of length 2*nLoci. Each entry in the vector is the (coded) allele held by the individual. This represents the parent. The relationship is reflexive so it does not matter if the profile is a parent or a child.
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.

Details

The alleles are simply integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

A vector with 2*nLoci elements. Each pair of elements represents the genotpe of the random individual at that locus. The genotype alleles are always ordered so that allele1 <= allele2.

Author(s)

James M. Curran

See Also

randomChild, randomSample, randomSib

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
C1 = randomChild(P1,fbiCaucs)
P1
C1
```

`randomPCPairs` *Generate one or more random parent/child pairs from a given set of allele frequencies*

Description

Generates one or more pairs random parent/child pairs from a given set of allele frequencies.

Usage

```
randomPCPairs(Freqs, BlockSize = 1)
```

Arguments

`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.
`BlockSize`	The number of pairs of profiles to generate

Details

The alleles are simply integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

A list of length `BlockSize`. Each element of the list has a sublist containing two profiles called `parent` and `child`

Author(s)

James M. Curran

See Also

randomSibPairs, randomProfilePairs

Examples

```
data(fbiCaucs)
P = randomPCPairs(fbiCaucs)
P$parent
P$child
```

`randomProfile` *Generate a random DNA profile from a given set of allele frequencies*

Description

Generates a random DNA profile from a given set of allele frequencies.

Usage

```
randomProfile(Freqs)
```

Arguments

`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.

Details

The alleles are simply integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

A vector with 2*nLoci elements. Each pair of elements represents the genotpe of the random individual at that locus. The genotype alleles are always ordered so that allele1 <= allele2.

Author(s)

James M. Curran

See Also

randomChild, randomSample, randomSib

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
```

randomProfilePairs — *Generate one or more random DNA profile pairs from a given set of allele frequencies*

Description

Generates one or more random DNA profile pairs from a given set of allele frequencies.

Usage

```
randomProfilePairs(Freqs, BlockSize = 1)
```

Arguments

`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.
`BlockSize`	The number of pairs of profiles to generate

Details

The alleles are simply integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

A list of length `BlockSize`. Each element of the list has a sublist containing two profiles called `prof1` and `prof2`

Author(s)

James M. Curran

See Also

randomPCPairs, randomSibPairs

Examples

```
data(fbiCaucs)
P = randomProfilePairs(fbiCaucs)
P$prof1
P$prof2
```

`randomSample` *Generate a random sample of related (or unrelated) pairs of people*

Description

Generate a random sample of unrelated, full-sib, or parent/child pairs of profiles at a single locus.

Usage

```
randomSample(nLoc, Freqs, rel = "UN", N = 10000)
```

Arguments

`nLoc`	The locus number to sample from
`Freqs`	A list containg elements `loci` and `freqs`. `freqs` is a list of vectors containing the frequencies at the given loci.
`rel`	One of ’UN’, ’FS’, or ’PC’ for unrelated, full-sib, or parent/child pairs respectively.
`N`	The sample size

Value

An N by 4 matrix of random profiles. The first two columns represent the genotype of person one and the second two columns represent the genotype of column two. Note that the random profiles do not use the orginal allele designations.

Author(s)

James M. Curran

See Also

randomProfile, randomSib, randomChild

Examples

```
data(fbiCaucs)
G = randomSample(1, fbiCaucs, "FS", 100)
```

`randomSib` *Generate a random sibling from a given DNA profile and a given set of allele frequencies*

Description

Generates a random sibling from a given DNA profile from a given set of allele frequencies. At each locus, the sibling inherits the first allele of the given profile with one quarter, or the second allele with probability one quarter, both alleles with probability one quarter, or neither with probability one quarter. If the sibling inherits zero or one identical alleles, the missing alleles are chosen at random with probability proportional to the allele frequencies.

Usage

```
randomSib(profile, Freqs)
```

Arguments

`profile`	A vector consisting of 2*nLoci elements. Each element in the vector is the (coded) allele held by the individual. This represents the sibling.
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.

Details

The alleles are simply integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

A vector with 2*nLoci elements. Each pair of elements represents the genotpe of the random individual at that locus. The genotype alleles are always ordered so that allele1 <= allele2.

Author(s)

James M. Curran

See Also

randomChild, randomSample

Examples

```
data(fbiCaucs)
P1 = randomProfile(fbiCaucs)
S1 = randomSib(P1,fbiCaucs)
P1
S1
```

---

randomSibPairs	*Generate one or more pairs of random siblings from a given set of allele frequencies*

Description

Generates one or more pairs of random siblings from a given set of allele frequencies.

Usage

```
randomSibPairs(Freqs, BlockSize = 1)
```

Arguments

`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.
`BlockSize`	The number of pairs of profiles to generate

Details

The alleles are simply integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

A list of length `BlockSize`. Each element of the list has a sublist containing two profiles called `sib1` and `sib2`

Author(s)

James M. Curran

See Also

randomPCPairs, randomProfilePairs

Examples

```
data(fbiCaucs)
P = randomSibPairs(fbiCaucs)
P$sib1
P$sib2
```

---

readFreqs *Read in a file of allele frequencies*

Description

Reads in a file of alleles in a particular format.

Usage

```
readFreqs(strPath, FSIGenFormat = TRUE, delim = ",")
```

Arguments

`strPath`	The file from which to read the frequencies
`FSIGenFormat`	Tells the function whether the file is either in FSI Genetics format (see below) or 'Curran' format
`delim`	This argument is used when `FSIGenFormat` is `TRUE`, and is the regular expression used to delimit columns of the table. it is set to a single comma by default, and multiple delimiters are considered empty separate fields. There probably should be an additional argument which specifies the missing or empty cell symbol, but I won't programme this unless somebody asks for it

Details

This function reads frequencies in the rectangular allele freqency table format used by FSI Genetics and other journals. This file format assumes a comma separated value file (CSV) (although the column delimeter can be specified). The first column should be labelled 'Allele' and contain the STR allele designations that are used in the data set. The remaining columns will have the locus name as a header, and frequencies that are either blank, zero, or non-zero. Blanks or zeros are used to specify that the allele is not observed (and not used) at the locus. The final row of the file should start with 'N' or 'n' in the first column and give the number of individuals typed (or the number of alleles recorded) in assessing the frequency of the alleles.

The second format is a very particular 'Curran' text format. The first line contains the number of loci in the multiplex. The next line will contain the name of the first locus and the number of alleles, nA, the locus separated by a comma. The next nA lines contain the allele number (from 1 to nA), the STR designation of the allele, and the frequency separated by commas. This pattern is repeated for each locus. In the future this function will read the rectangular allele freqency table used by FSI Genetics and other journals.

Value

a list containing two vectors and a list, loci, counts, and freqs. The vector loci is a vector of the locus names in the frequency file. The vector counts is a vector of the number of individuals (or sometimes alleles) typed at each locus. This will null if the 'Curran' format is used. The list freqs, is a list of vectors with each vector containing the frequencies of the alleles at the locus. The names of the elements of the vectors are the STR allele designations.

Author(s)

James M. Curran

`readProfiles` *Read a set of profiles from a file*

Description

Reads a set of profiles from a file

Usage

```
readProfiles(
  fileName,
  freqs = NULL,
  sep = "\t",
  header = FALSE,
  id = 1,
  discardMissing = TRUE
)
```

Arguments

`fileName`	a path to the profile file.
`freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex. If this is left `NULL`, then it is calculated from the profile file.
`sep`	a character that delimits the fields in the profile file.
`header`	a boolean which is `TRUE` if the profile file has a column header line.
`id`	a column number indicating which column the profile id's are stored. If `id == -1`, then this means there is no id information.
`discardMissing`	if `TRUE`, then all profiles which have alleles which cannot be matched to the frequency file are ommitted and returned in the return list.

Details

The alleles are recorded integers rather than the STR repeat numbers. This speeds up computation immensely when calculating any of the LRs or IBS.

Value

a list containing a `data.frame` of profiles where the alleles have been recoded to the allele index number, rather than the allele itself, and a set of frequencies in the same format as you would get from `readFreqs`. If `freqs` have been supplied, then this will just be the same set of frequencies, if they have not, then this will be calculated from the profiles. Given that the profiles generally do not have any locus name information the loci will just be labelled Locus1, Locus2, ... If there are missing values then the raw missing profiles are returned

Author(s)

James M. Curran

`readResults` *Read a simulation result set from file*

Description

This function will read the output from `sim` that has been saved to disk

Usage

```
readResults(
  N = 0,
  rel = "UN",
  gzip = TRUE,
  strPath = "",
  strVer = "",
  fileName = NULL
)
```

Arguments

`N`	The number of iterations in the simulation
`rel`	'UN' = unrelated, 'FS' = full-sib, 'PC' = parent-child
`gzip`	If `TRUE` then it is assumed that the file is compressed
`strPath`	Optional location of files. Must terminate with / otherwise it will not work
`strVer`	A version string, useful if more than simulation has been run
`fileName`	This argument allows the user to override the default file naming conventions of the result file

Details

The arguments to this file are used to generate the input file name. The format is very rigid, being 'results-sim-rel-N(-strVer).csv(.gz)' That is, if strVer is something than an empty string then it is included after the number of interations. Similarly if `gzip` == `TRUE` then the filename is assumed to end with '.gz'

Value

a data frame with three columns labelled sib, pc, and ibs. These represent the LRs for sibs and parent-child calculated on each simulated profile pair, and the number of matching alleles (IBS).

Author(s)

James M. Curran

See Also

sim

Examples

```
data(fbiCaucs)
## not run
## write the results of 100 unrelated profile pairs to
## results-sim-UN-100.csv.gz
## and read it back in
## Not run:
sim(100, save = T)
unrel = readResults(100)
sim(100, rel = "FS", strVer = "01", save = T)
sibs = readResults(100, rel = "FS", strVer = "01")

## End(Not run)
```

`relSim` *Relative Simulator*

Description

relSim

Details

A set of tools to explore the behaviour statistics used for forensic DNA interpretation when close relatives are involved. The package also offers some useful tools for exploring other forensic DNA situations.

Author(s)

James Curran

`sim` *Perform the relatives simulation*

Description

Generate N pairs with a given relationship and calculate the LR for sibs, parent-child and the number of matching alleles

Usage

```
sim(
  N,
  Freqs,
  rel = "UN",
  save = FALSE,
  strPath = "",
  strVer = "",
  BlockSize = N/100,
  fileName = NULL
)
```

Arguments

`N`	The number of iterations to carry out
`Freqs`	A list containing two lists labelled loci and freqs. The second list is a list of vectors containing the allele frequencies of each allele at each locus in the multiplex.
`rel`	generate unrelated (`rel = 'UN'`), full-sibs (`rel = 'FS'`), or parent child (`rel = 'PC'`) pairs
`save`	Write the results to disk if `TRUE`
`strPath`	Optional prefix to add to the results file path so that the output location can be specified
`strVer`	Optional suffix for the results file. This is useful when running multiple instances of R
`BlockSize`	Sets the number of random profiles to be generated in each iteration. By default the block size is set to 1 percent of the total sample size. It is unclear whether the procedure is more efficient if a bigger percentage of the total is used. Users must take care to make sure that the block size evenly divides `N` otherwise the procedure will exit
`fileName`	This argument lets the user override the default result file naming scheme

Details

This is the function that generates all the data for the results in the paper. WARNING: this function is not especially fast. To achieve the 100 million iterations used in the paper, 30 instances of R were launched on a multicore server. Each instance represented one relationship with 10 million iterations. The compute time for this arrangement was approximately 1 hours, meaning a full serial run would have taken over 30 hours to achieve the same result.

Value

a data frame with three columns: sib, pc, ibs containing the LRs for full-siblings, parent-child, and the number of matching alleles for each generated pair of profiles.

Author(s)

James M. Curran

See Also

readResults, errorRate

Examples

```
## not run
## this replicates Ge et al.'s experiment and takes about 45 minutes
## to run (I think)
## Not run:
data(fbiCaucs)
N = 1000000
sim(N, fbiCaucs, save = T)
sim(N, fbiCaucs, 'FS', save = T)
sim(N, fbiCaucs, 'PC', save = T)

## End(Not run)
```

`simNpersonMixture` *Simulate and count unique alleles in N person mixtures*

Description

This function simulates N persons mixtures using the supplied frequencies and records the number of times they share 1, 2, . . . , 2N alleles locus by locus.

Usage

```
simNpersonMixture(freqs, numContributors, numIterations = 10000)
```

Arguments

`freqs`	a set of allele frequencies. The format can be found in `readFreqs`
`numContributors`	the number of contributors to each mixture. Must be >= 2.
`numIterations`	the number of N person mixtures to simulate in total.

Value

an object of class npmresult

`toNexus` *Export a population with substructure to a Nexus file*

Description

Exports a population with population substructure to a Nexus formatted file so that GDA can be used to check the Fst calculations

Usage

```
toNexus(Pop, fileName = "output.nex")
```

Arguments

`Pop`	An object of type 'population' - see `breedFst` for a description of the object
`fileName`	The name of the file output file

Author(s)

James M. Curran

References

Maddison DR, Swofford DL, Maddison WP (1997), NEXUS: An extensible file format for systematic information, Systematic Biology 46 (4): 590–621.

Zaykin, D. and Lewis, P., GDA - software to accompany Genetic Data Analysis II, <http://phylogeny.uconn.edu/software/>.

See Also

breedFst

Examples

```
## Don't run
## Not run:
data(USCaucs)
p = breedFst(USCaucs)
toNexus(p)

## End(Not run)
```

`USCaucs` *CODIS STR Loci allele frequency data*

Description

This data structure

Format

This data set is a list which has two sub-lists. The lists are named loci and freqs. loci is a vector of the 13 CODIS STR locus names. freqs is a list of 13 vectors, each vector contains the allele frequencies published for US Caucasians in Budowle and Moretti (1999). The raw data is available from `https://www.fbi.gov/about-us/lab/forensic-science-communications/fsc/july1999/dnaloci.txt`

Author(s)

James M. Curran

References

Budowle, B. and Moretti, T.R. (1999), *Genotype Profiles for Six Population Groups at the 13 CODIS Short Tandem Repeat Core Loci and Other PCR Based Loci*, Forensic Science Communications 1(2).

See Also

fbiCaucs

Examples

```
data(USCaucs)
names(USCaucs)
USCaucs$loci
names(USCaucs$freqs)
USCaucs$freqs[[1]]
names(USCaucs$freqs[[1]])
USCaucs$freqs[[1]][1]
```

`writeCSV` *Saves/writes population frequencies to disk*

Description

Writes a population of profiles to disk using the original allele designations rather than the internal integer representations that are used for the other functions.

Usage

```
writeCSV(fileName, pop, n = 100, delim = ",")
```

Arguments

`fileName`	the name and path where the population profiles are to be saved to.
`pop`	a `list` containing elements `loci` and `freqs`. `loci` is a vector with the loci in the data set. `freqs` is a list of vectors with elements named after the elements in `loci`. Each locus in `freqs` is a vector of allele frequencies with the allele names given by the named elements. `TRUE` then an Amelogenin marker is added to the population, and all the profiles are set to male XY, although this is coded to 1,2 to keep the allele designations numeric.
`n`	the number of people in the database. This is arbitrarily set to 100 by default.
`delim`	The allele delimiter.

Value

a matrix which contains the table written to file.

Note

Rare alleles are recoded to 108.1. This is unlikely to do the right thing when you have things like <5 or >20 in your allele names. Given it is impossible to predict what a user would like to do, I suggest you recode them yourself before using this function.

Author(s)

James M. Curran

See Also

breedFst USCaucs

Examples

```
data(USCaucs)
## Not run:
  writeCSV("USCaucs.csv", USCaucs)

## End(Not run)
```

---

writePop *Saves/writes population profiles to disk*

Description

Writes a population of profiles to disk using the original allele designations rather than the internal integer representations that are used for the other functions.

Usage

```
writePop(fileName, pop, addAmelo = FALSE, delim = ",", dupLoci = TRUE)
```

Arguments

`fileName`	the name and path where the population profiles are to be saved to.
`pop`	an object of class `population`, most likely produced by `breedFst`
`addAmelo`	The simulated populations do not have Amelogenin. If `TRUE` then an Amelogenin marker is added to the population, and all the profiles are set to male XY, although this is coded to 1,2 to keep the allele designations numeric.
`delim`	The allele delimiter.
`dupLoci`	If `TRUE` the locus names are written twice in the header, otherwise just once.

Note

Rare alleles are recoded to 108.1.

Author(s)

James M. Curran

See Also

breedFst

Examples

```
data(USCaucs)
pop = breedFst(USCaucs)
## Not run:
  writePop("USCaucs.csv", pop)

## End(Not run)
```

Index