Package ‘DNAtools’

October 12, 2022

Type Package
Title Tools for Analysing Forensic Genetic DNA Data
Version 0.2-4
Author Torben Tvedebrink [aut],
        James Curran [aut],
        Mikkel Meyer Andersen [aut, cre]
Maintainer Mikkel Meyer Andersen <mikl@math.aau.dk>
Description Computationally efficient tools for comparing all pairs of profiles
    in a DNA database. The expectation and covariance of the summary statistic
    is implemented for fast computing. Routines for estimating proportions of
    close related individuals are available. The use of wildcards (also called F-
    designation) is implemented. Dedicated functions ease plotting the results.
    Compute the distribution of the numbers of alleles in DNA mixtures.
License GPL (>= 2) | file LICENSE
Depends R (>= 3.3.0)
Imports Rsolnp (>= 1.16), multicool (>= 0.1-10), Rcpp (>= 0.12.12),
        RcppParallel (>= 4.3.20)
LinkingTo Rcpp, RcppParallel, RcppProgress
SystemRequirements C++17, GNU make
BugReports https://github.com/mikldk/DNAtools/issues
NeedsCompilation yes
Encoding UTF-8
RoxygenNote 7.1.2
Suggests testthat, testthis, knitr, rmarkdown
VignetteBuilder utils, knitr
Repository CRAN
Date/Publication 2022-03-17 13:10:06 UTC
DNAtools-package

Tools for analysing forensic genetic DNA databases

Description

Computational efficient tools for comparing all pairs of profiles in a DNA database. The expectation and covariance of the summary statistic is implemented for fast computing. Routines for estimating proportions of close related individuals are available. The use of wildcards (also called F-designation) is implemented. Dedicated functions ease plotting the results.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>DNAtools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>0.1</td>
</tr>
<tr>
<td>Date</td>
<td>2014-08-25</td>
</tr>
<tr>
<td>License</td>
<td>GPL (&gt;= 2)</td>
</tr>
</tbody>
</table>

dbCompare: Compares make all n(n-1)/2 pairwise comparisons between profiles of a database with n DNA profiles. dbExpect: Computes the expected number of matching and partial matching loci for a given number of profiles in a database. dbVariance: Calculates the associated covariance
**dbCollapse**

Collapse m/p output to vector

**Description**

Collapse a m/p-matrix from dbCompare/dbExpect to a vector.

**Usage**

```r
dbCollapse(x)
```

**Arguments**

- `x` Either a object of class ‘dbcompare’ (result from dbCompare) or ‘matrix’.

**Details**

Collapse a m/p-matrix from dbCompare/dbExpect to a vector with entry i being the sum of all entries from m/p-matrix satisfying 2*m+p=i.

**Value**

A vector of length 2*max(m)+1 with entries begin the sum of entries i in m/p-matrix satisfying i=2*m+p.
**dbCompare**

**Author(s)**
Torben Tvedebrink

**Examples**

```r
## Not run:
data(dbExample)
res <- dbCompare(dbExample, hit=5, trace=TRUE)
dbCollapse(res) ## same as dbCompare(dbExample, hit=5, trace=TRUE, collapse=TRUE)
## End(Not run)
```

---

**dbCompare**

**Compare DNA profiles**

**Description**

Compare DNA profiles

**Usage**

```r
dbCompare(
  x,
  profiles = NULL,
  hit = 7,
  trace = TRUE,
  vector = FALSE,
  collapse = FALSE,
  wildcard = FALSE,
  wildcard.effect = FALSE,
  wildcard.impose = FALSE,
  Rallele = FALSE,
  threads = 2
)
```

**Arguments**

- `x` Database with DNA profiles. The database format is expected to be a data frame with each column containing an allelic number such that for each DNA marker there are two columns in the data frame. See `data(dbExample)` for an example of the format.
- `profiles` One or more profiles to be compared with all profiles in the database. Input is a vector, matrix or data frame of same length/width as a row in the database `x`. If profiles is non-null only one CPU will be used. In case `threads>1` a warning will be given but computations performed using single core.
The number of matching loci for further investigation. 

A progress bar to show the progress.

Logical. Whether the result should be returned as vector or a matrix. Note if 'collapse' is TRUE vector is ignored.

Logical (default FALSE). If TRUE the (m,p)-matrix will be collapsed into a (2*m+p)-vector containing the total number of matching alleles.

Use the wildcard comparing.

Compare result of wildcard and no wildcard.

Force homozygous profiles (aa) to have wildcard (aF).

Implementation of 'Rare allele' designation matching.

The number of threads to use for performing comparisons in parallel for increased computation time. Use 0 for using the same number as the computer has CPU cores. NOTE: Only available on Linux and MacOS operating systems.

Details

Computes the distance between DNA profiles in terms of matching and partially-matching STR loci.

Value

Returns a matrix with the number of pairs matching/partially-matching at (i,j)-loci.

Author(s)

James Curran and Torben Tvedebrink. The multicore/CPU implementation was provided by Mikkel Meyer Andersen.

Examples

```r
## Not run:
data(dbExample)
dbCompare(dbExample, hit=5, trace=TRUE)

## End(Not run)
```
**dbExample**

*Simulated database with 1,000 individuals*

**Description**

Database containing 1,000 simulated DNA profiles typed on ten autosomal markers.

**Format**

A data frame with each row being a DNA profile and each column a part of a genetic marker. Note that homozygote profiles has the same allelic value in the two columns associated to the same marker.

**dbExpect**

*Expected value of cell counts in DNA database comparison*

**Description**

Computes the expected number of cell counts when comparing DNA profiles in a DNA database. For every pair of DNA profiles in a database the number of matching and partial matching loci is recorded. A match is declared if the two DNA profiles coincide for both alleles in a locus and a partial-match is recorded if only one allele is shared between the profiles. With a total of \( L \) loci the number of matching loci is 0,...,\( L \) and partial number of matches is 0,...,\( L-m \), where \( m \) is the number of matching loci.

**Usage**

```r
dbExpect(
  probs, 
  theta = 0, 
  k = c(0, 0, 1),
  n = 1, 
  r = 0, 
  R = 0, 
  round = FALSE, 
  na = TRUE, 
  vector = FALSE, 
  collapse = FALSE, 
  wildcard = FALSE, 
  no.wildcard = NULL, 
  rare.allele = FALSE, 
  no.rare.allele = NULL 
)`
Arguments

probs  List of vectors with allele probabilities for each locus
theta  The coancestercy coefficient
k      The vector of identical-by-descent probabilities, \( k = (k_2, k_1, k_0) \), where for full-siblings \( k = (1, 2, 1)/4 \). The default is \( k = (0, 0, 1) \) referring to unrelated individuals.
n      Number of DNA profiles in the database
r      The probability assigned to the rare alleles (see rare allele matching). If a vector must be of same length as \( \text{probs} \).
R      The probability assigned to alleles shorter or longer than allelic ladder (see rare allele matching). If a vector must be of length 1 or 2, and if a list it must be same length as \( \text{probs} \).
round  Whether or not the results should be rounded or not
na     Whether or not the off-elements should be returned as 0 or NA
vector Whether or not the result should be returned as a matrix or vector. Note if 'collapse' is TRUE vector is ignored.
collapse Logical (default FALSE). If TRUE the (m,p)-matrix will be collapsed into a 
(2*m+p)-vector containing the total number of matching alleles.
wildcard Should wildcards be used?
o wildcard Should 'w' wildcards be used?
rare.allele Should rare allele matching be used?
no.rare.allele Should 'r' rare allele loci be used?

Details

Computes the expected cell counts using a recursion formula. See Tvedebrink et al (2011) for details.

Value

Returns a matrix (or vector, see above) of expected cell counts.

Author(s)

James Curran and Torben Tvedebrink

References

dbSimulate

Simulate a DNA database

Description

Simulates a DNA database given a set of allele probabilities and theta value. It is possible to have close relatives in the database simulated in pairs, such that within each pair the profiles are higher correlated due to close familial relationship, but between pairs of profiles the correlation is only modelled by theta.

Usage

dbSimulate(probs, theta = 0, n = 1000, relatives = NULL)

Arguments

- **probs**: List of allele probabilities, where each element in the list is a vector of allele probabilities.
- **theta**: The coancestry coefficient
- **n**: The number of profiles in the database
- **relatives**: A vector of length 4. Determining the number of PAIRS of profiles in the database: (FULL-SIBLINGS; FIRST-COUSINS; PARENT-CHILD, AVUNCULAR). They should obey that 2*sum(relatives)<=n.

Details

Simulates a DNA database with a given number of DNA profiles (and possibly relatives) with a correlation between profiles governed by theta.

Value

A data frame where each row represents a DNA profile. The first column is a profile identifier (id) and the next 2*L columns contains the simulated genotype for each of the L loci. L is determined by the length of the list 'probs' with allele probabilities.

Examples

```r
## Not run:
## Simulate some allele frequencies:
freqs <- replicate(10, { g = rgamma(n=10,scale=4,shape=3); g/sum(g)},
                    simplify=FALSE)
## Compute the expected number for a DB with 10000 profiles:
dbExpect(freqs,theta=0,n=10000)
## End(Not run)
```
**dbVariance**

**Covariance matrix of cell counts in DNA database comparison**

**Description**

Computes the covariance matrix for the cell counts when comparing DNA profiles in a DNA database. For every pair of DNA profiles in a database the number of matching and partial matching loci is recorded. A match is declared if the two DNA profiles coincide for both alleles in a locus and a partial-match is recorded if only one allele is shared between the profiles. With a total of \(L\) loci the number of matching loci is \(0,...,L\) and partial number of matches is \(0,...,L-m,\) where \(m\) is the number of matching loci. The expression is given by:

\[
\text{latex}
\]

**Usage**

\[
\text{dbVariance(probs, theta = 0, n = 1, collapse = FALSE)}
\]
Arguments

- **probs**: List of vectors with allele probabilities for each locus
- **theta**: The coancestry coefficient. If a vector of different theta values are supplied a list of covariance matrices is returned. Note it is faster to give a vector of theta values as argument than calculating each matrix at the time.
- **n**: Number of DNA profiles in the database. If n=1 is supplied a list of the components for computing the variance is returned. That is, the variance and two covariances on the right hand side of the equation above.
- **collapse**: Logical, default FALSE. If TRUE the covariance matrix is collapsed such that it relates to \((2*m+p)\)-vectors of total number of matching alleles rather than \((m,p)\)-matrix.

Details

Computes the covariance matrix of the cell counts using a recursion formula. See Tvedebrink et al (2011) for details.

Value

Returns a covariance matrix for the cell counts.

Author(s)

James Curran and Torben Tvedebrink

References


Examples

```r
## Not run:
## Simulate some allele frequencies:
freqs <- replicate(10, { g = rgamma(n=10,scale=4,shape=3); g/sum(g) }, simplify=FALSE)
## List of elements needed to compute the covariance matrix.
## Useful option when the covariance needs to be computed for varying
## database sizes but for identical theta-value.
comps <- dbVariance(freqs,theta=0,n=1)
## Covariance for a DB with 1000 DNA profiles
cov1000 <- dbVariance(freqs,theta=0,n=1000)
## The result is the same as:
comps1000 <- choose(1000,2)*comps$V1 + 6*choose(1000,3)*comps$V2 + 6*choose(1000,4)*comps$V3

## End(Not run)
```
Estimate the drop-out probability based on number of alleles

Description
An inferior may to estimate the drop-out probability compared to using the peak heights from the electropherogram. However, to compare the performance with Gill et al. (2007) this implements a theoretical approach based on their line of arguments.

Usage
estimatePD(n0, m, pnoa = NULL, probs = NULL, theta = 0, locuswise = FALSE)

Arguments
n0    Vector of observed allele counts - same length as the number of loci
m     The number of contributors
pnoa  The vector of \( \Phi(N(m) = n) \) for \( n = 1, \ldots, 2Lm \), where \( L \) is the number of loci and \( m \) is the number of contributors OR
probs List of vectors with allele probabilities for each locus
theta The coancestry coefficient
locuswise Logical. Indicating whether computations should be done locuswise.

Details
Computes the \( \Pr(D) \) that maximises equation (10) in Tvedebrink (2014).

Value
Returns the MLE of \( \Pr(D) \) based on equation (10) in Tvedebrink (2014)

Author(s)
Torben Tvedebrink

References
Examples

```r
## Simulate some allele frequencies:
freqs <- simAlleleFreqs()
## Assume 15 alleles are observed in a 2-person DNA mixture with 10 loci:
estimatePD(n0 = 15, m = 2, probs = freqs)
```

freqEst  

### Simple allele frequency estimation

**Description**

Estimates allele frequencies from a database with DNA profiles

**Usage**

`freqEst(x)`

**Arguments**

- `x`  
  A database of the form `[id,’locus1 allele1’,’locus1 allele2’,...,’locusN allele1’,’locusN allele2’].`

**Details**

Computes the allele frequencies for a given database.

**Value**

Returns a list of probability vectors - one vector for each locus.

**Author(s)**

James Curran and Torben Tvedebrink

**Examples**

```r
data(dbExample)
freqEst(dbExample)
```
**genRypeRec**

*Generates DNA profiles of n individuals.*

**Description**

These are formed as n/2 pairs for relatives with a IDB-vector given by k. I.e. the profiles are mutually unrelated between pairs.

**Usage**

```r
genRypeRec(x, t, k, n, print = FALSE)
```

**Arguments**

- `x`: Allele probabilities
- `t`: theta correction
- `k`: Relatedness vector
- `n`: Number of probles
- `print`: Print information

---

**genTypeRec**

*Generates DNA profiles of n unrelated individuals for a locus*

**Description**

Generates DNA profiles of n unrelated individuals for a locus

**Usage**

```r
genTypeRec(x, t, n, z = rep(0, 1x <- length(x)))
```

**Arguments**

- `x`: Allele probabilities
- `t`: theta correction
- `n`: Number of probles
- `z`: FIXME
optim.relatedness

Estimate theta and the fraction of comparisons between close relatives

Description

Estimates the fraction of comparisons between pairs of close relatives while fitting the theta parameter minimising the object function. The function makes use of the R-package 'Rsolnp' which is an implementation of an solver for non-linear minimisation problems with parameter constraints.

Usage

optim.relatedness(
  obs,
  theta0 = 0,
  theta1 = 0.03,
  theta.tol = 10^(-7),
  theta.step = NULL,
  max.bisect = 15,
  probs,
  var.list = NULL,
  init.alpha = 10^c(-4, -6, -8, -10),
  init.keep = FALSE,
  objFunction = c("T2", "T1", "C3", "C2", "C1"),
  collapse = FALSE,
  trace = FALSE,
  solnp.ctrl = list(tol = 10^(-9), rho = 10, delta = min(init.alpha) * 0.01, trace = FALSE)
)

Arguments

obs The matrix or vector of observed matches/partial-matches as returned by the dbCompare()-function
theta0 The left value of the interval in which a bisection-like search is performed for theta
theta1 Right value of interval (see theta0)
theta.tol A stopping criterion for the search. If the search narrows within theta.tol the function terminates
theta.step Default is NULL. If not a grid search will be performed on seq(from = theta0, to = theta1, by = theta.step)
max.bisect The maximum number of bisectional iterations perform prior to termination
probs List of vectors with allele probabilities for each locus
var.list A named list of components for computing variances, see dbVariance. The names of the elements are the associated theta-values, and each component is a list of (V1,V2,V3) - see dbVariance with n=1
init.alpha  Initial values for alpha, where the order is (First-cousins, Avuncular, Parent-child, Full-siblings). The value for Unrelated is computed as 1-sum(init.alpha)

init.keep  Whether the initial values should be used in successive steps for the current optimum should be used.

objFunction  Which of the five different object functions should be used to compare observed and expected

collapse  Not yet implemented

trace  Should iteration steps and other process indicators be printed

solnp.ctrl  See solnp for details

Details

Computes the proportion of comparisons between close relatives in a database matching exercise for each theta value under investigation.

Value

Returns a list of three components: value, solution and var.list. The first element, value, is a dataframe with the value of the objection function for each of the theta values investigated. Solution is the estimated alpha-vector where the objection function was minimised. Finally, var.list is a names list of components for computing variances. May be reused in later computations for increased speed in some iterations.

Author(s)

James Curran and Torben Tvedebrink

References


Examples

```r
## Not run:
## Simulate some allele frequencies:
freqs <- replicate(10, { g = rgamma(n=10,scale=4,shape=3); g/sum(g)}, simplify=FALSE)
## Load the sample database:
data(dbExample)
obs <- dbCompare(dbExample,trace=FALSE)$m
C3 <- optim.relatedness(obs,theta0=0.0,theta1=0.03,probs=freqs,
                         objFunction="C3",max.bisect=30,trace=TRUE)
## End(Not run)
```
Compute the posterior probabilities for \( P(m|n_0) \) for a given prior \( P(m) \) and observed vector \( n_0 \) of locus counts

Description

where \( m \) ranges from 1 to \( m_{\text{max}} \) and \( n_0 \) is the observed locus counts.

Usage

\[
p\text{Contrib}(n_0, \text{probs} = \text{NULL}, \text{m.prior} = \text{rep}(1/m_{\text{max}}, m_{\text{max}}), m_{\text{max}} = 8, \theta = 0)
\]

Arguments

- \( n_0 \): Vector of observed allele counts - same length as the number of loci.
- \( \text{probs} \): List of vectors with allele probabilities for each locus
- \( \text{m.prior} \): A vector with prior probabilities (summing to 1), where the length of \( \text{m.prior} \) determines the plausible range of \( m \)
- \( \text{m.max} \): Derived from the length of \( \text{m.prior} \), and if \( \text{m.prior} = \text{NULL} \) a uniform prior is specified by \( \text{m.max} \): \( \text{m.prior} = \text{rep}(1/m_{\text{max}}, m_{\text{max}}) \).
- \( \theta \): The coancestry coefficient

Details

Computes a vector \( P(m|n_0) \) evaluated over the plausible range 1,...,\( m_{\text{max}} \).

Value

Returns a vector \( P(m|n_0) \) for \( m=1,...,m_{\text{max}} \)

Author(s)

Torben Tvedebrink, James Curran

References


Examples

```r
## Simulate some allele frequencies:
freqs <- simAlleleFreqs()
m <- 2
n0 <- sapply(freqs, function(px)
  peaks = unique(sample(length(px),
```

### Notes

- The function `pContrib` computes the posterior probabilities for the number of contributors given the prior and the observed data. The function takes into account the allele probabilities and prior probabilities to calculate the posterior distribution.
- The `m.prior` argument specifies the prior probabilities for the number of contributors, which is critical for the computation. If `m.prior` is not specified, a uniform prior is used.
- The `theta` argument represents the coancestry coefficient, which is a measure of relatedness among contributors.
- The `Value` section describes that the function returns a vector of posterior probabilities for each possible number of contributors.
- The `Examples` section provides a usage example, demonstrating how to simulate allele frequencies and calculate the posterior probabilities.

### Implementation

The implementation details are not provided in the image, but they typically involve vectorized operations and possibly a loop to iterate through the range of plausible numbers of contributors. The calculation of posterior probabilities involves Bayes' theorem, applying the observed data likelihood and the prior distribution.
Contrib_locus}

```r
size = 2 * m,
replace = TRUE,
prob = px))

return(length(peaks))
```

## Compute P(m|n0) for m=1,...,4 and the sampled n0
pContrib(n0=n0,probs=freqs,m.max=4)

---

**pContrib_locus**

*Compute the posterior probabilities for Pr(m|n_0) for a given prior Pr(m).*

### Description

Compute a matrix of posterior probabilites Pr(m|n_0) where m ranges from 1 to m_{max}, and n_0 is 0,...,2m_{max}. This is done by evaluating Pr(m|n_0) = Pr(n_0|m)Pr(m)/Pr(n), where Pr(n_0|m) is evaluated by *pNoA.*

### Usage

```r
pContrib_locus(
  prob = NULL,
  m.prior = NULL,
  m.max = 8,
  pnoa.locus = NULL,
  theta = 0
)
```

### Arguments

- **prob** Vectors with allele probabilities for the specific locus
- **m.prior** A vector with prior probabilities (summing to 1), where the length of m.prior determines the plausible range of m
- **m.max** Derived from the length of m.prior, and if m.prior=NULL a uniform prior is specified by m.max: m.prior = rep(1/m.max,m.max).
- **pnoa.locus** A named vector of locus specific probabilities P(N(m) = n), n = 1,...,2m.
- **theta** The coancestery coefficient

### Details

Computes a matrix of Pr(m|n_0) values for a specific locus.

### Value

Returns a matrix [Pr(m|n_0)] for m = 1,...,m.max and n_0 = 1,...,2m.max.
Author(s)
Torben Tvedebrink, James Curran

References

Examples

```r
## Simulate some allele frequencies:
freqs <- simAlleleFreqs()

## Compute Pr(m|n0) for m = 1, ..., 5 and n0 = 1, ..., 10 for the first locus:
pContribution(prob = freqs[[1]], m.max = 5)
```

---

**plot.dbcompare**

Plots the summary matrix with counts on y-axis and classification on x-axis.

**Usage**

```r
## S3 method for class 'dbcompare'
plot(x, log = "y", las = 3, xlab = "Match/Partial", ylab = "Counts", ...)
```

**Arguments**

- `x`: Summary matrix returned from dbcompare
- `log`: Specifies whether log(Counts) should be plotted (default)
- `las`: Direction of the labels on x-axis. Default is 3 which gives perpendicular labels
- `xlab`: Axis label
- `ylab`: Axis label
- `...`: Other plot options

**Value**

A plot of the summary matrix. The counts are on log10 scale and the x-axis is labeled by appropriate matching/partially-matching levels.

**Author(s)**
James Curran and Torben Tvedebrink
plot.dbOptim

See Also

dbCompare, print.dbcompare

Examples

## Not run:
data(dbExample)
M = dbCompare(dbExample, hit=5)
plot(M)

## End(Not run)

plot.dbOptim  Plots the fitted object function for estimated familial relationships in the database and theta.

Description

Plots the minimised object function for included values of theta

Usage

## S3 method for class 'dbOptim'
plot(x, type = "l", ...)

Arguments

x  Object returned by optim.relatedness

type  The type of plot character (’l’=line, ’p’=points, ...), see ’par’ for more details

...  Other plot options

Details

Plots the object function

Value

A plot of the object function

Author(s)

James Curran and Torben Tvedebrink

See Also

optim.relatedness
Examples

```r
## Not run:
## Simulate some allele frequencies:
freqs <- replicate(10, { g = rgamma(n=10, scale=4, shape=3); g/sum(g)},
    simplify=FALSE)
## Load the sample database:
data(dbExample)
obs <- dbCompare(dbExample,trace=FALSE)$m
C3 <- optim.relatedness(obs,theta0=0.0,theta1=0.03,probs=freqs,
    objFunction='C3',max.bisect=30,trace=TRUE)
plot(C3)
## End(Not run)
```

Pnm_all

The exact distribution of the number of alleles in a m-person DNA mixture

Description

Computes the exact distribution of the number of alleles in a \( m \)-person DNA mixture typed with STR loci. For a \( m \)-person DNA mixture it is possible to observe \( 1, \ldots, 2 \times m \times L \) alleles, where \( L \) is the total number of typed STR loci. The method allows incorporation of the subpopulation correction, the so-called \( \theta \)-correction, to adjust for shared ancestry. If needed, the locus-specific probabilities can be obtained using the `locuswise` argument.

Usage

```
Pnm_all(m, theta, probs, locuswise = FALSE)
Pnm_locus(m, theta, alleleProbs)
```

Arguments

- `m` The number of contributors
- `theta` The coancestry coefficient
- `probs` List of vectors with allele probabilities for each locus
- `locuswise` Logical. If `TRUE` the locus-wise probabilities will be returned. Otherwise, the probability over all loci is returned.
- `alleleProbs` Vectors with allele probabilities

Details

Computes the exact distribution of the number of alleles for a m-person DNA mixture.
Value
Returns a vector of probabilities, or a matrix of locuswise probability vectors.

Author(s)
Torben Tvedebrink, James Curran, Mikkel Andersen

References

Examples

```r
## Simulate some allele frequencies:
freqs <- structure(replicate(10, { g = rgamma(n = 10, scale = 4, shape = 3);
g/sum(g)
}),
simplify = FALSE), .Names = paste('locus', 1:10, sep = '.')

## Compute $\Pr(N(m = 3) = n), n = 1,\ldots,2 \times L \times m$, where $L = 10$
## here
Pnm_all(m = 2, theta = 0, freqs)
## Same, but locuswise results
Pnm_all(m = 2, theta = 0, freqs, locuswise = TRUE)
```

---

print.dbcompare

Prints the summary matrix

Description
Prints the summary matrix and possible 'big hits'.

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

Arguments
- `x` Summary matrix returned from dbcompare
- `...`

Details
Prints the summary matrix
Value

Prints the summary matrix and data frame with 'big hits'

Author(s)

James Curran and Torben Tvedebrink

See Also

dbCompare, plot.dbcompare

Examples

```r
## Not run:
data(dbExample)
M = dbCompare(dbExample, hit=5)
M
## End(Not run)
```

print.dbOptim

Prints the results from optim.relatedness()

Description

Prints the evaluated functions for the object function, best estimate of alpha and possibly list of variances.

Usage

```r
## S3 method for class 'dbOptim'
print(x, var.list = FALSE, ...)
```

Arguments

- **x**: Object returned by optim.relatedness()
- **var.list**: Logical. Whether the (long) list of variance components should be printed to the screen.
- **...**: ...

Details

Prints the summary details of the fit
**Value**

A dataframe with [theta, value] and a vector of fitted alpha parameters

**Author(s)**

James Curran and Torben Tvedebrink

**See Also**

optim.relatedness

**Examples**

```r
## Not run:
## Simulate some allele frequencies:
freqs <- replicate(10, { g = rgamma(n=10, scale=4, shape=3); g/sum(g)}, simplify=FALSE)
## Load the sample database:
data(dbExample)
obs <- dbCompare(dbExample,trace=FALSE)$m
C3 <- optim.relatedness(obs,theta0=0.0,theta1=0.03,probs=freqs,
objFunction='C3',max.bisect=30,trace=TRUE)
print(C3)
## End(Not run)
```

---

**Description**

Simulate some allele frequencies using Dirichlet Random variables

**Usage**

```r
simAlleleFreqs(  
nLoci = 10,  
allelesPerLocus = rep(10, nLoci),  
shape = rep(3, nLoci)  
)
```

**Arguments**

- `nLoci`  
  $L$ the number of loci in the multiplex
- `allelesPerLocus`  
  the number of alleles per locus
- `shape`  
  the shape parameter
Value

A list with elements `locus[l]` where `l = 1, \ldots, L`, each of which are vectors of length `allelesPerLocus[l]`, consisting of allele frequencies for that locus.

Examples

```r
set.seed(123)
simAlleleFreqs()
```
Index

* Forensic
  DNAtools-package, 2
* genetics
  DNAtools-package, 2

convolve (Pnm_all), 20

dbCollapse, 3
dbCompare, 4
dbExample, 6
dbExpect, 6
dbSimulate, 8
dbVariance, 9
DNAtools (DNAtools-package), 2
DNAtools-package, 2

estimatePD, 11

freqEst, 12

genRypeRec, 13
genTypeRec, 13

lines.dbOptim (plot.dbOptim), 19

optim.relatedness, 14

p.numberofalleles (Pnm_all), 20
pContrib, 16
pContribution locus, 17
plot.dbcompare, 18
plot.dbOptim, 19
Pnm_all, 20
Pnm_locus (Pnm_all), 20
pNoA, 17
pNoA (Pnm_all), 20
points.dbOptim (plot.dbOptim), 19
print.dbcompare, 21
print.dbOptim, 22

simAlleleFreqs, 23