

# Package ‘DNAtools’

January 20, 2025

**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 <mik1@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. See Tvedebrink et al. (2012) <[doi:10.1016/j.fsigen.2011.08.001](https://doi.org/10.1016/j.fsigen.2011.08.001)>. Compute the distribution of the numbers of alleles in DNA mixtures. See Tvedebrink (2013) <[doi:10.1016/j.fsigs.2013.10.142](https://doi.org/10.1016/j.fsigs.2013.10.142)>.

**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

## Contents

DNAtools-package	2
dbCollapse	3
dbCompare	4
dbExample	5
dbExpect	6
dbSimulate	8
dbVariance	9
estimatePD	10
freqEst	11
genTypeRec	12
genTypeRec	13
optim.relatedness	13
pContrib	15
pContrib_locus	16
plot.dbcompare	17
plot.dbOptim	18
Pnm_all	19
print.dbcompare	21
print.dbOptim	22
simAlleleFreqs	23
<b>Index</b>	<b>24</b>

---

DNAtools-package	<i>Tools for analysing forensic genetic DNA databases</i>
------------------	---

---

## 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

Package: DNAtools  
 Type: Package  
 Version: 0.1  
 Date: 2014-08-25  
 License: GPL (>= 2)

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 matrix.

**Author(s)**

Torben Tvedebrink <tvede@math.aau.dk>, James Curran <j.curran@auckland.ac.nz> and Mikkel Meyer Andersen <mikl@math.aau.dk>.

**References**

Tvedebrink T, JM Curran, PS Eriksen, HS Mogensen and N Morling (2012). Analysis of matches and partial-matches in a Danish STR data set. *Forensic Science International: Genetics*, 6(3): 387-392.

Read the vignette: `vignette('DNAtools')`

**Examples**

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

## End(Not run)
```

---

dbCollapse

*Collapse m/p output to vector*

---

**Description**

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

**Usage**

```
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 being the sum of entries i in m/p-matrix satisfying  $i=2*m+p$ .

**Author(s)**

Torben Tvedebrink

**Examples**

```
## 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**

```
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 <code>data(dbExample)</code> 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.
hit	The number of matching loci for further investigation
trace	Shows a progress bar
vector	Logical. Whether the result should be returned as vector or a matrix. 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	Use the wildcard comparing.
wildcard.effect	Compare result of wildcard and no wildcard.
wildcard.impose	Force homozygous profiles (aa) to have wildcard (aF).
Rallele	Implementation of 'Rare allele' designation matching.
threads	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**

```
## 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  $0, \dots, L$  and partial number of matches is  $0, \dots, L-m$ , where  $m$  is the number of matching loci.

**Usage**

```
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 coancestry coefficient
k	The vector of identical-by-descent probabilities, $k=(k_2,k_1,k_0)$ , where for full-siblings $k=c(1,2,1)/4$ . The default is $k=c(0,0,1)$ 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 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 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?
no.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

T Tvedebrink, PS Eriksen, J Curran, HS Mogensen, N Morling. 'Analysis of matches and partial-matches in Danish DNA reference profile database'. Forensic Science International: Genetics, 2011.

### Examples

```
## 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)
```

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 $2 * \text{sum}(\text{relatives}) \leq n$ .

**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

**Author(s)**

James Curran and Torben Tvedebrink

**Examples**

```
## Not run:
## Simulate some allele frequencies:

freq <- replicate(10, { g = rgamma(n=10, scale=4, shape=3); g/sum(g)},
  simplify=FALSE)
## Simulate a single database with 5000 DNA profiles:
simdb <- dbSimulate(freq, theta=0, n=5000)
```



```

## Simulate a number of databases, say N=50. For each database compute
## the summary statistic using dbCompare:
N <- 50
Msummary <- matrix(0,N,(length(freq)+1)*(length(freq)+2)/2)
for(i in 1:N)
  Msummary[i,] <- dbCompare(dbSimulate(freq,theta=0,n=1000),
                           vector=TRUE,trace=FALSE)$m
## Give the columns representative names:
dimnames(Msummary)[[2]] <- DNAtools:::dbCats(length(freq),vector=TRUE)
## Plot the simulations using a boxplot
boxplot(log10(Msummary))
## There might come some warnings due to taking log10 to zero-values (no counts)
## Add the expected number to the plot:
points(1:ncol(Msummary),log10(dbExpect(freq,theta=0,n=1000,vector=TRUE)),
      col=2,pch=16)

## 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  $0, \dots, L$  and partial number of matches is  $0, \dots, L-m$ , where  $m$  is the number of matching loci. The expression is given by:

*latex*

## Usage

```
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**

T Tvedebrink, PS Eriksen, J Curran, HS Mogensen, N Morling. 'Analysis of matches and partial-matches in Danish DNA reference profile database'. Forensic Science International: Genetics, 2011.

**Examples**

```
## 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)
```

---

estimatePD

*Estimate the drop-out probability based on number of alleles*

---

**Description**

An inferior way 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 $\mathbb{P}(N(m) = n)$ for $n = 1, \dots, 2Lm$ , 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**

Gill, P., A. Kirkham, and J. Curran (2007). LoComatioN: A software tool for the analysis of low copy number DNA profiles. *Forensic Science International* 166(2-3): 128 - 138.

T. Tvedebrink (2014). 'On the exact distribution of the number of alleles in DNA mixtures', *International Journal of Legal Medicine*; 128(3):427–37. <<https://doi.org/10.1007/s00414-013-0951-3>>

**Examples**

```
## 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**

```
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**

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

**Arguments**

x                    Allele probabilities  
t                    theta correction  
k                    Relatedness vector  
n                    Number of probes  
print                Print information

---

genTypeRec	<i>Generates DNA profiles of n unrelated individuals for a locus</i>
------------	--

---

**Description**

Generates DNA profiles of n unrelated individuals for a locus

**Usage**

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

**Arguments**

x	Allele probabilities
t	theta correction
n	Number of probes
z	FIXME

---

optim.relatedness	<i>Estimate theta and the fraction of comparisons between close relatives</i>
-------------------	---

---

**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**

<code>obs</code>	The matrix or vector of observed matches/partial-matches as returned by the <code>dbCompare()</code> -function
<code>theta0</code>	The left value of the interval in which a bisection-like search is performed for <code>theta</code>
<code>theta1</code>	Right value of interval (see <code>theta0</code> )
<code>theta.tol</code>	A stopping criterion for the search. If the search narrows within <code>theta.tol</code> the function terminates
<code>theta.step</code>	Default is NULL. If not a grid search will be performed on <code>seq(from = theta0, to = theta1, by = theta.step)</code>
<code>max.bisect</code>	The maximum number of bisectional iterations perform prior to termination
<code>probs</code>	List of vectors with allele probabilities for each locus
<code>var.list</code>	A named list of components for computing variances, see <code>dbVariance</code> . The names of the elements are the associated <code>theta</code> -values, and each component is a list of (V1,V2,V3) - see <code>dbVariance</code> with <code>n=1</code>
<code>init.alpha</code>	Initial values for <code>alpha</code> , where the order is (First-cousins, Avuncular, Parent-child, Full-siblings). The value for Unrelated is computed as <code>1-sum(init.alpha)</code>
<code>init.keep</code>	Whether the initial values should be used in successive steps for the current optimum should be used.
<code>objFunction</code>	Which of the five different object functions should be used to compare observed and expected
<code>collapse</code>	Not yet implemented
<code>trace</code>	Should iteration steps and other process indicators be printed
<code>solnp.ctrl</code>	See <code>solnp</code> 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

T Tvedebrink, PS Eriksen, J Curran, HS Mogensen, N Morling. 'Analysis of matches and partial-matches in Danish DNA reference profile database'. Forensic Science International: Genetics, 2011.

## Examples

```
## 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)
```

---

pContrib	<i>Compute the posterior probabilities for <math>P(m n_0)</math> for a given prior <math>P(m)</math> and observed vector <math>n_0</math> of locus counts</i>
----------	---

---

## Description

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

## Usage

```
pContrib(n0, probs = NULL, m.prior = rep(1/m.max, m.max), m.max = 8, theta = 0)
```

## Arguments

n0	Vector of observed allele counts - same length as the number of loci.
probs	List of vectors with allele probabilities for each 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).
theta	The coancestry coefficient

## Details

Computes a vector  $P(m|n_0)$  evaluated over the plausible range 1,...m.max.

**Value**

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

**Author(s)**

Torben Tvedebrink, James Curran

**References**

T. Tvedebrink (2014). 'On the exact distribution of the number of alleles in DNA mixtures', International Journal of Legal Medicine; 128(3):427–37. <<https://doi.org/10.1007/s00414-013-0951-3>>

**Examples**

```
## Simulate some allele frequencies:
freqs <- simAlleleFreqs()
m <- 2
n0 <- sapply(freqs, function(px){
  peaks = unique(sample(length(px),
                        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	<i>Compute the posterior probabilities for <math>\Pr(m n_0)</math> for a given prior <math>\Pr(m)</math>.</i>
----------------	---

---

**Description**

Compute a matrix of posterior probabilities  $\Pr(m|n_0)$  where  $m$  ranges from 1 to  $m_{\max}$ , and  $n_0$  is  $0, \dots, 2m_{\max}$ . 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**

```
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 $P(N(m) = n), n = 1, \dots, 2m$ .
theta	The coancestry coefficient

**Details**

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

**Value**

Returns a matrix  $[\Pr(m|n_0)]$  for  $m = 1, \dots, m.max$  and  $n_0 = 1, \dots, 2m.max$ .

**Author(s)**

Torben Tvedebrink, James Curran

**References**

T. Tvedebrink (2014). 'On the exact distribution of the number of alleles in DNA mixtures', International Journal of Legal Medicine; 128(3):427–37. <<https://doi.org/10.1007/s00414-013-0951-3>>

**Examples**

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

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

---

plot.dbcompare      *Plots the summary matrix*

---

**Description**

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

**Usage**

```
## 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

**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**

```
## 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  $1, \dots, 2 \times m \times L$  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**

T. Tvedebrink (2014). 'On the exact distribution of the number of alleles in DNA mixtures', International Journal of Legal Medicine; 128(3):427–37. <<https://doi.org/10.1007/s00414-013-0951-3>>

**Examples**

```
## 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  $\Pr(N(m = 3) = n)$ ,  $n = 1, \dots, 2 * L * m$ , 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	<i>Prints the summary matrix</i>
-----------------	----------------------------------

---

**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**

```
## 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

```
## 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

```
## 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)
```

---

simAlleleFreqs	<i>Simulate Allele Frequencies</i>
----------------	------------------------------------

---

### Description

Simulate some allele frequencies using Dirichlet Random variables

### Usage

```
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  $l = 1, \dots, L$ , each of which are vectors of length `allelesPerLocus[1]`, consisting of allele frequencies for that locus

### Examples

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

# Index

## \* **Forensic**

DNAtools-package, [2](#)

## \* **genetics**

DNAtools-package, [2](#)

convolve (Pnm\_all), [19](#)

dbCollapse, [3](#)

dbCompare, [4](#)

dbExample, [5](#)

dbExpect, [6](#)

dbSimulate, [8](#)

dbVariance, [9](#)

DNAtools (DNAtools-package), [2](#)

DNAtools-package, [2](#)

estimatePD, [10](#)

freqEst, [11](#)

genRypeRec, [12](#)

genTypeRec, [13](#)

lines.dbOptim (plot.dbOptim), [18](#)

optim.relatedness, [13](#)

p.numberofalleles (Pnm\_all), [19](#)

pContrib, [15](#)

pContrib\_locus, [16](#)

plot.dbcompare, [17](#)

plot.dbOptim, [18](#)

Pnm\_all, [19](#)

Pnm\_locus (Pnm\_all), [19](#)

pNoA, [16](#)

pNoA (Pnm\_all), [19](#)

points.dbOptim (plot.dbOptim), [18](#)

print.dbcompare, [21](#)

print.dbOptim, [22](#)

simAlleleFreqs, [23](#)