Package 'DNAmixturesLite'

July 21, 2025

Type Package

Title Statistical Inference for Mixed Traces of DNA (Lite-Version)

Version 0.0-1

Date 2023-03-15

Description Statistical methods for DNA mixture analysis. This package is a lite-version of the 'DNAmixtures' package to allow users without a 'HUGIN' software license to experiment with the statistical methodology. While the lite-version aims to provide the full functionality it is noticeably less efficient than the original 'DNAmixtures' package. For details on implementation and methodology see <https://dnamixtures.r-forge.r-project.org/>.

Depends gRaven

Imports methods, Rsolnp, numDeriv, Matrix, gRbase

License GPL (>= 2)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Author Therese Graversen [aut, cre]

Maintainer Therese Graversen <theg@itu.dk>

Repository CRAN

Date/Publication 2023-03-16 11:30:08 UTC

Contents

`DNAmixturesLite-package`

Statistical Inference for Mixed Samples of DNA (Lite-Version)

Description

Tools for statistical inference for one or multiple DNA mixtures.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Details

The package implements a statistical model for analysis of one or more mixed samples of DNA in the possible presence of dropout and stutter. Details of the model can be found in Cowell et. al (2013), and details on the model checking tools and Bayesian network structure can be found in Graversen and Lauritzen (2014).

Any hypothesis involving unknown contributors relies on computations in a Bayesian network. For performing such computations, **DNAmixtures** package relies on Hugin (https://www.hugin.com) through the R-package **RHugin**. For an installation guide, see the package webpage https://dnamixtures.r-forge.r-project.org.

Although **DNAmixtures** can be installed with only the free version of Hugin, the size of the networks will in practice require the full licence. In theory, the implementation allows analysis with an arbitrary number of unknown contributors. However, in practice, depending on hardware and time-constraints working with up to 5 or 6 unknown contributors seems realistic.

Summary of the statistical model

The statistical model jointly models the observed peak heights and the set of contributors to the DNA sample(s). In the event of analysing multiple DNA mixtures, the union of the contributors is used as the contributor set for each mixture. By allowing a contribution of zero, we cover the case of a contributor not having contributed to a particular mixture.

Genotypes for unknown contributors are modelled using allele-frequencies from a database specified by the user. The database is also used to define the range of alleles at each marker. A genotype for an unknown contributor is represented by a vector of allele counts n_{ia}, counting for each allele a the number of alleles i that a person possesses; in the network for a marker, the allele count n_{ia} is represented by a variable `n_i_a`. The vector of allele counts follows a multinomial distribution with $\sum_i n_{ia} = 2$ and the specified allele frequencies. It is assumed that genotypes are independent across markers and between contributors. If desired, the database of allele frequencies may be corrected for F_st or sampling adjustment before use.

Peak heights are assumed mutually independent and their distributions for a fixed set of DNA profiles are modelled using gamma distributions. The peak height for allele a in EPG r is assumed to follow a gamma distribution with scale parameter η_r and shape parameter

$$\rho_r \sum_a ((1 - \xi_{ra}) n_{ia} + \xi_{r,a+1} n_{i,a+1}) \phi_{ri}.$$

Applying a detection threshold $C_r \geq 0$, any peak height falling below the threshold is considered to be 0. The peak heights are denoted by `height1, ..., heightR`.

The model parameters are for each DNA mixture

ϕ The proportions of DNA from each contributor.

ρ Amplification parameter, which will be larger for larger amounts of DNA amplified.

η Scale parameter for the gamma distribution.

ξ Mean stutter percentage. Allele a uses stutter parameter $\xi_a = \xi$ if the allele $a - 1$ is included in the model, and $\xi_a = 0$ otherwise

An alternative parametrisation uses $\mu = \rho\eta$ and $\sigma = 1/\sqrt{\rho}$, which can be interpreted as the mean peak height and the coefficient of variation respectively. Besides being interpretable, an advantage of this reparametrisation is that the parameters are fairly orthogonal.

The model assumes the model parameters to be the same across markers. Relaxations of these assumptions are not implemented here.

Computation by auxiliary variables

The computational approach of the implementation of this package is discussed in Graversen and Lauritzen (2014).

The Bayesian networks include three types of auxiliary variables `O`, `D`, and `Q`; these can be thought of as representing the observed peak heights, the absence/presence of peaks, and the peak height distribution function. Note that if invalid tables are set – for instance if very extreme parameter values are used, or if the vector of mixture proportions is mis-labeled – then any subsequent propagation will fail. No roll-back functionality has so far been implemented to fix this, and the easiest solution is to re-fit the mixture model.

The workhorses of this package are the functions `setCPT.O`, `setCPT.D` and `setCPT.Q` for setting the conditional probability tables for the three types of auxiliary variables according to specified peak heights and model parameters.

Amelogenin

As an experiment, it is possible to add the marker Amelogenin, provided that the marker is named "AMEL" and that the coding of alleles X and Y is of a particular form. One example of a suitable form is the coding X = 0 and Y = 1. The allele frequencies used should then also contain a marker "AMEL", and here frequencies have a slightly different interpretation than for the rest of the markers; as all people possess one X, the frequencies of X and Y denote the presence of an additional X or Y respectively, and thus the frequencies correspond directly to the proportions of the two genders.

Author(s)

Therese Graversen <theg@itu.dk>

References

Details on the implemented model may be found in

Cowell, R. G., Graversen, T., Lauritzen, S., and Mortera, J. (2015). *Analysis of Forensic DNA Mixtures with Artefacts*. With supplementary material documenting the analyses using **DNAmixtures**. Journal of the Royal Statistical Society: Series C (Applied Statistics). Volume 64, Issue 1, pages 1-48.

Graversen, T. (2014) *Statistical and Computational Methodology for the Analysis of Forensic DNA Mixtures with Artefacts*. DPhil. University of Oxford. https://ora.ox.ac.uk/objects/uuid:4c3bfc88-25e7-4c5b-968f-10a35f5b82b0.

Graversen, T. and Lauritzen, S. (2014). *Computational aspects of DNA mixture analysis*. Statistics and Computing, DOI: 10.1007/s11222-014-9451-7.

Examples

```
## Analysing trace MC18, using USCaucasian allele-frequencies.
data(MC18, USCaucasian, SGMplusDyes)
## The prosecution hypothesis could be that K1, K2, and K3 have
## contributed to the trace. Setting detection threshold to 50rfu.
## For layout as an EPG we follow the layout of SGMplus.
mixHp <- DNAmixture(list(MC18), k = 3, K = c("K1", "K2", "K3"), C = list(50),
                  database = USCaucasian, dyes = list(SGMplusDyes))
plot(mixHp, epg = TRUE)
## The yellow is a bit difficult to see; we can
## change the colors representing the dyes
plot(mixHp, epg = TRUE, dyecol = list(c("blue", "green", "black")))

## Fit by maximum likelihood using p as the initial value for optimisation
p <- mixpar(rho = list(30), eta = list(34), xi = list(0.08),
            phi = list(c(K1 = 0.71, K3 = 0.1, K2 = 0.19)))
mlHp <- mixML(mixHp, pars = p)
mlHp$mle

## Find the estimated covariance matrix of the MLE
V.Hp <- varEst(mixHp, mlHp$mle, npars = list(rho=1,eta=1,xi=1,phi=1))
V.Hp$cov ## using (rho, eta)
## The summary is a table containing the MLE and their standard errors
summary(V.Hp)

## Assess the fit of the distribution of peak heights
qqpeak(mixHp, pars = mlHp$mle, dist = "conditional")

## Assess the prediction of allele presence
pr <- prequential.score(mixHp, pars = mlHp$mle)
plot(pr)

## Compare observed peak heights to peak heights simulated under the model
sims <- rPeakHeight(mixHp, mlHp$mle, nsim = 100, dist = "conditional")
oldpar <- par(mfrow = c(2,5), mar = c(2,2,2,0))
boxplot(mixHp, sims)
par(oldpar)


data(MC18, USCaucasian, SGMplusDyes)

## A defence hypothesis could be that one unknown person has
## contributed along with K1 and K3.
mixHd <- DNAmixture(list(MC18), k = 3, K = c("K1", "K3"), C = list(50),
                  database = USCaucasian, dyes = list(SGMplusDyes))

## Fit by ML
mlHd <- mixML(mixHd, pars = mixpar(rho = list(30), eta = list(34), xi = list(0.08),
                       phi = list(c(K1 = 0.71, K3 = 0.1, U1 = 0.19))))
mlHd$mle

## Find the estimated covariance matrix of the MLE
```

```
V <- varEst(mixHd, mlHd$mle, npars = list(rho=1,eta=1,xi=1,phi=1))
summary(V)

## Assess the peak height distribution
qq <- qqpeak(mixHd, pars = mlHd$mle, dist = "conditional")

## Assess the prediction of allele presence
pr <- prequential.score(mixHd, pars = mlHd$mle)
plot(pr, col = pr$trace)

## Compare observed peak heights to peak heights simulated under the model
sims <- rPeakHeight(mixHd, mlHd$mle, nsim = 100, dist = "conditional")
par(mfrow = c(2,5), mar = c(2,2,2,0))
boxplot(mixHd, sims)


## The log10 of likelihood-ratio can be found as
(mlHp$lik - mlHd$lik)/log(10)


## We can find the most probable genotypes for U1 under the hypothesis
## Hd : K1, K2, and U1.

## Include the peak heights
setPeakInfo(mixHd, mlHd$mle)
## Jointly best for all unknown contributors
mp <- map.genotypes(mixHd, pmin = 0.01, type = "seen")
## See the genotypes rather than allelecounts
print(summary(mp), marker = "D2S1338")

## Include only information on presence and absence of alleles
setPeakInfo(mixHd, mlHd$mle, presence.only = TRUE)
## Jointly best for all unknown contributors
mp2 <- map.genotypes(mixHd, pmin = 0.01, type = "seen")
## See the genotypes rather than allelecounts
print(summary(mp2), marker = "D2S1338")
```

`boxplot.DNAmixture` *Plot simulated peak heights for multiple DNA mixtures.*

Description

A plot will be made for each combination of samples and markers specified, and it is up to the user to specify a layout for the plots (e.g. via calls to `par`)

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
## S3 method for class 'DNAmixture'
boxplot(
  x,
  sims,
  traces = 1:x$ntraces,
  markers = x$markers,
  pw = 0.4,
  ylim = NULL,
  border = "grey",
  ...
)
```

Arguments

x	A DNAmixture.
sims	A set of simulated peak heights. See also rPeakHeight.
traces	Selected traces to plot.
markers	Selected markers to plot.
pw	Peaks are 2*pw wide.
ylim	Range of the y-axis.
border	Color of the boxes.
...	Arguments passed on to plot.DNAmixture.

Value

Invisibly the peak heights as used for the boxplots.

Author(s)

Therese Graversen

Examples

```
data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = c("K1", "K3"),
                  database = USCaucasian)
p <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = c(0.08,0.08),
           phi = list(c(U1 = 0.1, K3 = 0.2, K1 = 0.7), c(K1 = 0.9, K3 = 0.05, U1 = 0.05)))
rpm <- rPeakHeight(mix, p, nsim = 1000, dist = "joint")
oldpar <- par("mfrow")
par(mfrow = c(1,2))
```

```
boxplot(mix, rpm, traces = 1:2, markers = "VWA")
par(mfrow = oldpar)


data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = "K3", database = USCaucasian)
p <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = c(0.08,0.08),
          phi = list(c(U2 = 0.1, K3 = 0.2, U1 = 0.7), c(U1 = 0.9, K3 = 0.05, U2 = 0.05)))
rpm <- rPeakHeight(mix, p, nsim = 50, dist = "joint")
rpc <- rPeakHeight(mix, p, nsim = 50, dist = "conditional")
oldpar <- par("mfrow")
par(mfrow = c(2,2))
boxplot(mix, rpm, traces = 1:2, markers = "VWA") ## First row of plots
boxplot(mix, rpc, traces = 1:2, markers = "VWA") ## Second row of plots
par(mfrow = oldpar)
mixK <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = c("K1", "K2", "K3"),
                   database = USCaucasian)
pK <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08,0.08),
            phi = list(c(K2 = 0.1, K3 = 0.2, K1 = 0.7), c(K2 = 0.1, K3 = 0.2, K1 = 0.7)))
rpmK <- rPeakHeight(mixK, pK, nsim = 1000, markers = "D2S1338")
oldpar <- par(mfrow = c(1,2))
boxplot(mixK, rpmK, markers = "D2S1338")
par(oldpar)
```

`buildMixtureDomains` *Create RHugin domains for computation.*

Description

The function is intended for internal use in `DNAmixture`, and it creates one network per marker.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
buildMixtureDomains(
  n.unknown,
  data,
  domainnamelist,
  write,
  dir = NULL,
```

```
  ntraces = 1,
  triangulate = TRUE,
  compile = TRUE,
  compress = TRUE,
  use.order = TRUE
)
```

Arguments

n.unknown	Number of unknown contributors
data	A list of data.frames on the form returned by DNAmixtureData.
domainnamelist	List of names for networkfiles.
write	[not available in lite-version] Save networkfiles? Defaults to FALSE.
dir	Path to the networkfiles.
ntraces	Number of traces (EPG's)
triangulate	Triangulate the networks? Default is to triangulate the network using a good elimination order.
compile	Compile the networks? Defaults to TRUE.
compress	Compress the network? Defaults to TRUE and is strongly recommended for models with a large number of contributors.
use.order	Should the default elimination order be used for triangulation? Otherwise the "total.weight" heuristic for triangulation in HUGIN is used.

Details

For one marker, the network contains variables

- n_i_a The number of alleles a that unknown contributor Ui possesses.
- S_i_a Cumulative allele counts, summarising how many alleles amongst types $1, \ldots, a$ that contributor i possesses.
- O_r_a Binary variable representing observed peak height for allele a in EPG r. See setCPT.O for details.
- D_r_a Binary variable representing the event of a peak for allele a falling below threshold in EPG r.
- Q_r_a Binary variable representing the event of a peak for allele a being smaller than the peak observed in EPG r.

The network is by default triangulated, compiled, compressed, and, optionally, saved as a hkb-file. If the networks are large – if there are many unknown contributors – it is worth considering saving the networks rather than re-building them every time DNAmixture is called.

Value

A list containing a list of pointers for RHugin domains. Additionally a list of the total size of the compressed junction tree in percent of the uncompressed size.

Note

If amelogenin is included as a marker, use integer codes for the alleles with X preceding Y, e.g. X=0, Y=1.

Author(s)

Therese Graversen

`DNAmixture` *Create a DNA mixture model*

Description

A model object for analysis of one or more DNA mixtures. For a brief overview of the package functionality, see in particular `DNAmixtures`.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
DNAmixture(
  data,
  k,
  C,
  database,
  K = character(0),
  reference.profiles = NULL,
  dir = character(0),
  domainnamelist = NULL,
  load = FALSE,
  write = FALSE,
  dyes = NULL,
  triangulate = TRUE,
  compile = TRUE,
  compress = TRUE,
  use.order = TRUE
)

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

Arguments

data	A list containing one data.frame for each DNA mixture. Note, that in the special case of analysing just one mixture, this still has to be specified as list(data). Each dataset should contain variables marker, allele, and frequency. Optionally, also a column for each reference profile specified in K.
k	Number of contributors.
C	A list of thresholds, one for each mixture.
database	A data.frame containing at least variables marker, allele, frequency.
K	Names of reference profiles; these can be chosen freely, but should match (possibly only a subset of) the names specified by the reference profiles.
reference.profiles	A data.frame containing allele counts for each reference profile, if not specified in data.
dir	Location of network files if loading or saving the networks.
domainnamelist	Names of marker-wise network files (without hkb-extension). Default is the set of markers.
load	Read networks from disk?
write	Save networks as hkb files?
dyes	A list containing a list of dyes indexed by markers
triangulate	Triangulate the networks? Default is to triangulate the network using a good elimination order.
compile	Compile the networks?
compress	Compress the network? Defaults to TRUE and is strongly recommended for models with a large number of contributors.
use.order	Should the default elimination order be used for triangulation? Otherwise the "total.weight" heuristic for triangulation in Hugin is used.
x	An object of class DNAmixture.
...	not used.

Details

The names for known contributors can be chosen freely, whereas unknown contributors are always termed U1,U2, ...

We allow for stutter to an allele one repeat number shorter. The range of alleles at a marker is defined by the union of alleles specified though the peak heights, the reference profiles, and the allele frequencies. Any alleles that are included, but not found in the database, will be assigned frequency NA, and it is then up to the user to decide on further actions. If a particular mixture has no observations at a marker, the heights are set to NA, but if the mixture has some peaks at that marker, then missing heights are all set to 0. Note that we hereby cover the possibility that mixtures are analysed with different kits, and so are observed at different markers. We do not (readily) allow kits to have different ranges of possible alleles at one marker.

If amelogenin is included in the analysis, the marker should be named "AMEL" and an integer coding such as X=0, Y=1, where X is assigned a lower number than Y, should be used. Note that in terms of

amelogenin, the allele frequencies have a slighly different interpretation to that for other markers, in that they denote the probability of having an *additional* X or Y to the X that all people have. Thus, a natural choice will be $p(X) = p(Y) = 0.5$, denoting equal probability of a male or female contributor.

Value

An object of class `DNAmixture`. This contains amongst other things

`markers`	The joint set of markers used for the mixtures specified.
`domains`	For models involving unknown contributors, a list containing one Bayesian network (`hugin.domain`) per marker; see `buildMixtureDomains` for details on the networks
`data`	A list containing for each marker the combined allele frequencies, peak heights, and reference profiles as produced by `DNAmixtureData`.

Examples

```
data(MC15, MC18, USCaucasian)
DNAmixture(list(MC15, MC18), C = list(50,50), k = 3, K = c("K1", "K2"),
           database = USCaucasian)
DNAmixture(list(MC15, MC18), C = list(50,50), k = 3, K = c("K3", "K1", "K2"),
           database = USCaucasian)
```

DNAmixtureData *Create a data set for a DNA mixture model*

Description

The function is intended for internal use in `DNAmixture` and its purpose is to combine peak height data, any reference profiles, and allele frequencies.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* `https://dnamixtures.r-forge.r-project.org/` *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
DNAmixtureData(data, database, K = character(0), reference.profiles = NULL)
```

Arguments

data	Either one data.frame containing variables markers and allele, height, or a list of one or more of such data.frames, corresponding to each EPG. If a data.frame with reference profiles are not specified separately, these should be found amongst the datasets.
database	A data.frame with variables marker, allele, and allele frequency.
K	Names for reference profiles.
reference.profiles	Optionally, a data.frame containing allele counts for each reference profile specified in K. There should be one variable for each name in K as well as variables marker and allele.

Value

list of data.frame's indexed by markers and containing variables

marker	The STR marker
allele	Repeat number of the allele
frequency	Allele frequency
height1, ..., heightN	Peak heights for each of the N mixtures analysed; their order follows the order in which the mixtures are specified in data.
stutter.from	For internal use. Where do the alleles get stutter from? A value of -1 means that the allele cannot receive stutter, corresponding to gets_stutter being false. allele[i] receives stutter from allele[stutter.from[i]]
stutter.to	As above. allele[i] can stutter to allele[stutter.to[i]]

as well as known DNA profiles as labelled by K.

Author(s)

Therese Graversen

Examples

```
## Create a dataset for two markers with each 3 observed alleles
epgdf <- data.frame(marker = rep(c("FGA", "TH01"), each = 3),
                    allele = c(18, 23, 27, 7, 8, 9.3), ## Observed alleles
                    height = c(100, 100, 200, 200, 100, 100), ## Peak heights
                    Anna = c(0,0,2,1,0,1),             ## Anna's profile
                    Peter = c(1,1,0,1,1,0))            ## Peter's profile

data(USCaucasian)
dat <- DNAmixtureData(epgdf, K = c("Anna", "Peter"), database = USCaucasian)
```

dyes *Dyes for a DNA mixture*

Description

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
dyes(mixture)

dyes(mixture) <- value
```

Arguments

mixture	A DNA mixture
value	A list with one item per dye, each containing a vector of marker names.

Value

The dyes used for the DNA mixture

See Also

DNAmixture

getShapes *Shape parameters for a mixture with known contributors only*

Description

The function is mainly for internal use. It calculates the shape parameters, or contribution to the shape parameter, that correspond to the known contributors in the mixture.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
getShapes(mixture, pars, allelecounts = NULL)
```

Arguments

`mixture`	A `DNAmixture`.
`pars`	Model parameters.
`allelecounts`	This argument is possibly redundant.

Value

For each marker a matrix of shape-parameters.

Author(s)

Therese Graversen

See Also

predict

Examples

```
data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50,50), k = 3, K = c("K1", "K3", "K2"),
database = USCaucasian)
p <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08,0.08),
           phi = list(c(K2 = 0.1, K3 = 0.2, K1 = 0.7), c(K2 = 0.1, K3 = 0.2, K1 = 0.7)))
sh <- getShapes(mix, p)
sh$VWA
```

---

logL *Loglikelihood function for DNA mixture analysis.*

Description

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware*

that the present documentation is copied from **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

The function `logL` is used to produce a log likelihood function for one or more traces of DNA. `logL` calls one of four other likelihood functions according to whether peak heights or only peak presence/absence is used as observations, and also whether there are any unknown contributors in the model.

Usage

```
logL(mixture, presence.only = FALSE, initialize = TRUE)

logL.UK(mixture, initialize = TRUE)

logL.K(mixture)

logLpres.UK(mixture, initialize = TRUE)

logLpres.K(mixture)
```

Arguments

`mixture`	A `DNAmixture` model.
`presence.only`	Set to TRUE to ignore peak heights and evaluate the likelihood function considering peak presence and absence (heights above and below threshold) only. Defaults to FALSE.
`initialize`	By default all entered evidence is removed from the networks in `object`. Setting `initialize = FALSE` should be done with care, and it is up to the user to ensure that the likelihood being computed is meaningful.

Value

A function, which takes a `mixpar` model parameter as argument.

Note

In the presence of unknown contributors, the returned likelihood function relies on the Bayesian networks in the `DNAmixture` object. If any evidence is entered or propagated in these networks after creating the likelihood function, the function will no longer compute the correct likelihood. In particular, be ware of calling other functions in between calls to the likelihood function, as they may change the state of the networks.

Author(s)

Therese Graversen

Examples

```
data(MC18, USCaucasian)
mixHp <- DNAmixture(list(MC18), k = 3, K = c("K1", "K2", "K3"), C = list(50),
                    database = USCaucasian)
p <- mixpar(rho = list(30), eta = list(34), xi = list(0.08),
            phi = list(c(K1 = 0.71, K3 = 0.1, K2 = 0.19)))
l <- logL(mixHp)
l(p)
```

---

map.genotypes *Maximum posterior genotypes of unknown contributors*

Description

For each marker, a ranked list of configurations of genotypes for some or all unknown contributors is returned. The list contains all configurations with posterior probability higher than some specified `pmin`.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
map.genotypes(
  mixture,
  pmin,
  U = seq_along(mixture$U),
  markers = mixture$markers,
  type = c("seen", "all", "unseen")
)
```

Arguments

`mixture`	a DNA mixture
`pmin`	A list of the minimum probability to consider for each marker.
`U`	Optionally the indices of the unknown contributors of interest, specified as an integer vector.
`markers`	Optionally, a subset of markers.
`type`	It may be of interest to consider only the prediction of alleles in some subset of alleles. We allow

"seen" Consider only alleles that are seen in at least one EPG

"all" Consider the entire allelic range

"unseen" Consider only alleles that are not seen in any EPG (possibly redundant)

Details

Note that an error occurs if there are no configurations with probability higher than pmin. In this case, try a smaller pmin.

The function makes use of map.configurations, which localises the configurations of highest high posterior probability by simulating from the Bayesian networks until enough (at least mass 1-pmin) of the state space has been explored – the computation time is thus dependent on how flat the posterior is, and how small pmin is. The simulation is used only to locate the relevant configurations; the computed probabilities are exact.

Value

A list, which for each marker contains the maximum posterior configurations of allele counts (genotypes) above the specified probabilities pmin.

See Also

summary.map.genotypes

Examples

```
data(MC18, USCaucasian, SGMplusDyes)
mix <- DNAmixture(list(MC18), k = 3, K = "K1", C = list(50), database = USCaucasian)
p <- mixpar(rho = list(30), eta = list(30), xi = list(0.07),
            phi = list(c(K1 = 0.7, U1 =0.2, U2 = 0.1)))
## Inlude the peak height information
setPeakInfo(mix, p)
## Marginally best genotypes for contributor U1
mpU1 <- map.genotypes(mix, pmin = 0.01, U = 1, type = "seen", markers = "D16S539")
summary(mpU1)

## Jointly best genotypes for all unknown contributors
mp <- map.genotypes(mix, pmin = 0.01, type = "seen")
summary(mp) ## Profiles as genotypes rather than allelecounts
```

MC15 *The MC15 DNA mixture*

Description

Peak heights from analysis of a bloodstain with DNA from an unknown number of contributors. There are three reference profiles K1, K2, and K3 for individuals related to the case.

Format

A `data.frame`.

Source

Peter Gill et. al. (2008) *Interpretation of complex DNA profiles using empirical models and a method to measure their robustness*. Forensic Science International: Genetics, 2(2):91–103.

`MC18` *The MC18 DNA mixture*

Description

Peak heights from analysis of a bloodstain with DNA from an unknown number of contributors. There are three reference profiles `K1`, `K2`, and `K3` for individuals related to the case.

Format

A `data.frame`.

Source

Peter Gill et. al. (2008) *Interpretation of complex DNA profiles using empirical models and a method to measure their robustness*. Forensic Science International: Genetics, 2(2):91–103.

`mixML` *Maximisation of the likelihood for one or more mixed traces of DNA*

Description

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* `https://dnamixtures.r-forge.r-project.org/` *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
mixML(
  mixture,
  pars,
  constraints = NULL,
  phi.eq = FALSE,
  val = NULL,
  trace = FALSE,
  order.unknowns = TRUE,
  ...
)
```

Arguments

mixture	A DNAmixture object.
pars	A mixpar parameter used as a starting value for the optimisation.
constraints	Equality constraint function as function of an array of parameters.
phi.eq	Should the mixture proportions be the same for all traces? Defaults to FALSE.
val	Vector of values to be satisfied for the equality constraints.
trace	Print the evaluations of the likelihood-function during optimisation?
order.unknowns	Should unknown contributors be ordered according to decreasing contributions? Defaults to TRUE.
...	Further arguments to be passed on to solnp.

Details

The pre-specified constraints for the model parameter pars are for each mixture r

- 0 <= pars[[r, "rho"]] < Inf
- 0 <= pars[[r, "eta"]] < Inf
- 0 <= pars[[r, "xi"]] <= 1
- 0 <= pars[[r, "phi"]][i] <= 1
- sum(pars[[r, "phi"]]) == 1.
- If there are 2 or more unknown contributors, then the mixture proportions for the unknown contributors are ordered decreasingly with the first DNA mixture determining the order.

Further constraints can be specified by the user; for this see examples below.

Value

A list containing

mle	The (suggested) MLE.
lik	The log of the likelihood (log e).

as well as the output from the optimisation.

Author(s)

Therese Graversen

See Also

DNAmixture

Examples

```
data(MC15, USCaucasian)
mix <- DNAmixture(list(MC15), C = list(50), k = 3, K = c("K1", "K2", "K3"), database = USCaucasian)
startpar <- mixpar(rho = list(24), eta = list(37), xi = list(0.08),
                   phi = list(c(K3 = 0.15, K1 = 0.8, K2 = 0.05)))
ml <- mixML(mix, startpar)
ml$mle


data(MC15, USCaucasian)
mix <- DNAmixture(list(MC15), C = list(50), k = 3, K = "K1", database = USCaucasian)
startpar <- mixpar(rho = list(24), eta = list(37), xi = list(0.08),
phi = list(c(U1 = 0.05, K1 = 0.8, U2 = 0.15)))
ml <- mixML(mix, startpar)
ml$mle



## The following advanced example has a model with two DNA samples
## and various parameter restrictions.
## Be aware that the computation is rather demanding and takes
## a long time to run with this lite-version of DNAmixtures.
## With DNAmixtures (based on HUGIN), it takes only about one minute.
data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = "K1", database = USCaucasian)
startpar <- mixpar(rho = list(24, 25), eta = list(37, 40), xi = list(0.08, 0.1),
                   phi = list(c(U1 = 0.05, K1 = 0.7, U2 = 0.25),
                              c(K1 = 0.7, U2 = 0.1, U1 = 0.2)))
eqxis <- function(x){ diff(unlist(x[,"xi"])) }
## Note that for these two traces, we do not expect phi to be equal.
## Here we set stutter equal for all traces
ml.diff <- mixML(mix, startpar, eqxis, val = 0, phi.eq = FALSE)
## Equal mixture proportions across traces
ml.eqphi <- mixML(mix, startpar, eqxis, val = 0, phi.eq = TRUE)
## Likelihood ratio test for equal mixture proportions
pchisq(-2*(ml.eqphi$lik - ml.diff$lik), df = 1, lower.tail = FALSE)
```

---

mixpar *Parameters for DNA mixture models*

Description

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
mixpar(rho = NULL, eta = NULL, xi = NULL, phi = NULL, parlist = NULL)

## S3 method for class 'mixpar'
print(x, digits = max(3L, getOption("digits") - 3L), scientific = FALSE, ...)
```

Arguments

rho	Amplification factor.
eta	Scale parameter in the gamma distribution.
xi	Stutter parameter.
phi	Named vector of the fraction of DNA contributed by each contributor.
parlist	A list of parameters of class mixpar
x	An object of class mixpar.
digits	The number of digits to print
scientific	Should scientific notation be used?
...	arguments passed to print

Details

The mixture parameter is a two-way array of lists; columns correspond to the four model parameters rho, eta, xi, and phi, and rows correspond to the mixtures included in the model.

The print method is currently somewhat specialised, in that it assumes that rho, eta, and xi are merely real numbers. phi is assumed to consist of a named vector per mixture; the names, or order of names, can differ between mixtures.

Value

An object of class "mixpar".

Author(s)

Therese Graversen

Examples

```
## A parameter for two mixtures
q <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08, 0.08),
            phi = list(c(Anna = 0.5, Peter = 0.2, U1 = 0.3),
                       c(U1 = 0.5, Anna = 0.2, Peter = 0.3)))
## Equivalent to specifying the parameter for each mixture and then combining.
p1 <- mixpar(rho = list(30), eta = list(30), xi = list(0.08),
             phi = list(c(Anna = 0.5, Peter = 0.2, U1 = 0.3)))
p2 <- mixpar(rho = list(30), eta = list(30), xi = list(0.08),
             phi = list(c(U1 = 0.5, Anna = 0.2, Peter = 0.3)))
p <- mixpar(parlist = list(p1, p2))
```

NGM *NGM allele frequencies*

Description

NGM allele frequencies.

Format

A list containing

`USCaucasian` A `data.frame` with, for each marker, the set of possible alleles as well as their corresponding frequency.

Source

Budowle et. al (2011) *Population Genetic Analyses of the NGM STR loci*. International Journal of Legal Medicine, 125(1):101-109.

NGMDyes *Dyes used for NGM*

Description

Dyes used for the AmpFlSTR NGM PCR Amplification Kit

Format

A list with one item per dye, each containing a vector of marker-names specifying the order in which they occur in an EPG.

Source

Applied Biosystems

`plot.DNAmixture` *Plot a DNA mixture model*

Description

Plot of peak heights against repeat numbers for each marker.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
## S3 method for class 'DNAmixture'
plot(
  x,
  traces = seq_len(x$ntraces),
  markers = x$markers,
  epg = FALSE,
  dyecol = lapply(dyes(x), names),
  pw = 0.4,
  threshold = TRUE,
  panel = NULL,
  add = FALSE,
  ask = NULL,
  ...
)
```

Arguments

`x`	A `DNAmixture`.
`traces`	Indices giving the mixtures, for which plots should be made.
`markers`	Vector of names giving the markers, for which plots should be made.
`epg`	Should a stylized EPG be produced? This requires a list of dyes to be specified for the mixtures, possibly through `dyes`.
`dyecol`	List containing for each mixture a vector of dye names. The default is to use the names in `dyes(x)`. Set to `NULL` to ignore.
`pw`	Peaks are `2*pw` wide.
`threshold`	Should the detection thresholds be indicated on the plot?
`panel`	Alternative function for drawing the peaks. For instance `lines` can be used for making triangular peaks. This functionality will be removed.

add	Add to existing plot? (not meaningful if EPG = TRUE.)
ask	Should the user be prompted for page changes? The default is TRUE, when the device is dev.interactive() and there is a need for multiple pages.
...	Other parameters to be passed on to plot

Value

A data.frame containing the plotted data points.

Examples

```
data(MC15, MC18, USCaucasian, SGMplusDyes)
mix <- DNAmixture(list(MC15, MC18), C = list(50,50), k = 3, K = c("K1", "K3"),
                  database = USCaucasian)
## Plot as a stylized EPG, using "orange" for the "yellow" dye
names(SGMplusDyes) <- c("blue", "green", "orange")
dyes(mix) <- list(SGMplusDyes, SGMplusDyes)
plot(mix, epg = TRUE)
## We can also supress the dye colors
plot(mix, traces = 1, epg = TRUE, dyecol = NULL)
## Create a user specified layout
op <- par(mfrow = c(2,3))
plot(mix, markers = c("VWA", "D19S433", "TH01"), col = "red")
par(mfrow = op)
plot(mix, traces = 1, markers = "D19S433", col = "orange",
     main = "A single marker", cex.main = 2, ylim = c(0,200))
```

---

predict.DNAmixture *Various probabilities in a fitted DNA mixture model*

Description

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
## S3 method for class 'DNAmixture'
predict(
  object,
  pars,
  dist = c("joint", "conditional", "prequential"),
```

```
  markers = object$markers,
  by.allele = TRUE,
  initialize = TRUE,
  ...
)
```

Arguments

object	A DNAmixture object
pars	Array of model parameters
dist	One of "joint", "conditional", and "prequential". If there are only known contributors, these are all the same since, under the model, peak heights are conditionally independent given profiles of the contributors.
markers	The set of markers of interest
by.allele	If dist = "prequential" then the order in which we condition on mixtures and alleles matters. by.allele = TRUE will proceed through alleles in increasing repeat number, and for each allele condition on one mixture at the time. If FALSE, the conditioning is done by mixtures and then alleles within these.
initialize	By default predict removes all entered evidence from the networks in object. Setting initialize = FALSE should be done with care, and it is up to the user to ensure that the returned probabilities are meaningful.
...	Not used

Details

For a mixture with unknown contributors, the probabilities are computed with respect to one of three distributions. Let height be the matrix of peak heights with columns height1, ..., heightR. For a peak at allele a in the mixture r, the three choices of distributions are

"joint" Default. No conditioning on observed peak heights.

"conditional" Conditional on height[-a, -r], i.e. on heights for all peaks, except the one under consideration.

"prequential" Conditional on height[1:(a-1), 1:(r-1)], i.e. on heights for all peaks "before" the peak under consideration (see argument by.allele for details).

If all contributors are known, the three distributions are the same due to independence of the peak heights.

Value

A list with one data.frame per marker containing various probabilities for diagnostics

unseen	The probability of not seeing a peak, i.e. no peak or a peak falling below the threshold
seen	The probability of seeing the allele
smaller	The probability of seeing a smaller peak than the one observed
larger	The probability of seeing a larger peak than the one observed

Author(s)

Therese Graversen

Examples

```
data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50,50), k = 3, K = c("K1", "K3", "K2"),
database = USCaucasian)
p <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08,0.08),
            phi = list(c(K2 = 0.1, K3 = 0.2, K1 = 0.7), c(K2 = 0.1, K3 = 0.2, K1 = 0.7)))
pred <- predict(mix, p)
pred$VWA
```

---

prequential.score *Calculate prequential scores*

Description

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
prequential.score(mixture, pars, markers = mixture$markers, by.allele = TRUE)

## S3 method for class 'prequential.score'
plot(x, normalise = FALSE, ylab = NULL, ylim = NULL, ...)
```

Arguments

mixture	A DNAmixture model object.
pars	A mixpar model parameter.
markers	An ordering of the markers, possibly a subset of the markers only.
by.allele	Should conditioning be done allele-wise (TRUE) or EPG-wise (FALSE). Defaults to TRUE. For details, see predict.
x	A data.frame containing at least variables marker and score.
normalise	Should the prequential score be normalised? Defaults to FALSE.
ylab	Label for the y-axis.
ylim	Range for the y-axis.
...	Additional arguments to be passed on to plot.

Value

A data.frame, which contains the output from `predict` as well as columns `Y`, `EY`, `VY`, corresponding to the log-score and its mean and variance. Finally the variable `score` is added, which is the normalised cumulative log-score.

Author(s)

Therese Graversen

Examples

```
data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50,50), k = 3, K = c("K1", "K3"),
                  database = USCaucasian)
p <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08,0.08),
            phi = list(c(U1 = 0.1, K3 = 0.2, K1 = 0.7), c(U1 = 0.1, K3 = 0.2, K1 = 0.7)))
preq <- prequential.score(mix, pars = p)
plot(preq, col = preq$trace)
## Annotate using repeat numbers
text(preq$score, labels = preq$allele, pos = c(1,3), col = preq$trace, cex = 0.6)
```

ProfilerDyes *Dyes used for Profiler plus*

Description

Dyes used for the AmpFlSTR Profiler Plus PCR Amplification Kit

Format

A list with one item per dye, each containing a vector of marker-names specifying the order in which they occur in an EPG.

Source

Applied Biosystems

qqpeak *Quantile-Quantile plot for assessing the distribution of observed peak heights.*

Description

Given $Z_a \geq C$, the peak height Z_a follows a continuous distribution. The probability transform $P(Z_a \leq z_a | Z_a \geq C)$ thus follows a uniform distribution. The function qqpeak computes the quantiles of the probability transform and produces a quantile-quantile plot. Information about the other peak heights in the EPG may be taken into account in the distribution of a peak.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
qqpeak(
  x,
  pars,
  dist = c("joint", "conditional", "prequential"),
  by.allele = TRUE,
  plot = TRUE,
  xlab = "Uniform quantiles",
  ylab = NULL,
  xlim = NULL,
  ylim = xlim,
  ...
)
```

Arguments

x	A DNAmixture model.
pars	A mixpar parameter.
dist	"joint", "conditional", or "prequential". See predict.DNAmixture for details.
by.allele	Order of conditioning when dist="prequential". See predict.DNAmixture
plot	Should a plot be produced? Defaults to TRUE.
xlab	Legend for the x-axis.
ylab	Legend for y-axis.

xlim	Range of x-axis. Default is a plot with equal ranges on the x- and y-axis.
ylim	Range of y-axis.
...	Other arguments to be passed on to plot.

Value

A data.frame containg quantiles q corresponding to $P(Z < z_{obs}|Z > 0)$ in the specified distribution, together with other quantities computed by predict.DNAmixture. The data are ordered according to q.

Author(s)

Therese Graversen

Examples

```
data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = c("K1", "K3"),
                  database = USCaucasian)
p <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08,0.08),
            phi = list(c(U1 = 0.1, K3 = 0.2, K1 = 0.7)))
qqpeak(mix, pars = p, dist = "conditional")

## If desired, we can make a customised plot -- here we color according to the two mixtures
qq <- qqpeak(mix, pars = p, dist = "conditional", plot = FALSE)
plot(ppoints(qq$q), qq$q, xlab = "Uniform quantiles", ylab = "Probability transform",
     col = qq$trace, pch = qq$trace)
```

---

rPeakHeight *Simulate peak heights from a DNA mixture model.*

Description

A set of peak heights is sampled for each mixture in the model. Note that if the mixtures cover different sets of markers, so will the simulated peak heights. If there are unknown contributors, their genotypes are sampled from the networks in mixture$domains, but these genotypes are not stored.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
rPeakHeight(
  mixture,
  pars,
  nsim = 1,
  markers = mixture$markers,
  dist = c("joint", "conditional"),
  use.threshold = TRUE,
  initialize = TRUE,
  ...
)
```

Arguments

mixture	A DNAmixture.
pars	A mixpar model parameter.
nsim	Number of simulations for each allele.
markers	Subset of markers to simulate
dist	How should observed peak heights be taken into account? Possible choices are "joint" which does not take peak heights into account. "conditional" which takes all peak heights into account, except that for the peak under consideration.
use.threshold	Should peak heights under the detection threshold C be set to 0? Defaults to TRUE, corresponding to simulating under the model fitted.
initialize	Should all propagated evidence be removed from the network? Defaults to TRUE. If set to FALSE, the user should make sure to check whether the networks represent the distribution of interest.
...	Not used.

Value

A list of one three-way array per marker; the three margins correspond to traces, alleles, and simulations.

Author(s)

Therese Graversen

Examples

```
data(MC15, MC18, USCaucasian)

mixP <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = c("K1", "K3", "K2"),
                   database = USCaucasian)
pP <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08, 0.08),
            phi = list(c(K2 = 0.1, K3 = 0.2, K1 = 0.7), c(K2 = 0.1, K3 = 0.2, K1 = 0.7)))
rpk <- rPeakHeight(mixP, pP, nsim = 5)
```

```
mixD <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = c("K1", "K3"),
                   database = USCaucasian)
pD <- mixpar(rho = list(30, 30), eta = list(30, 30), xi = list(0.08, 0.08),
             phi = list(c(U1 = 0.1, K3 = 0.2, K1 = 0.7), c(U1 = 0.1, K3 = 0.2, K1 = 0.7)))
rpj <- rPeakHeight(mixD, pD, nsim = 5, dist = "joint")
rpc <- rPeakHeight(mixD, pD, nsim = 5, dist = "conditional")
```

`setCPT` *Set conditional probability tables for all auxiliary variables*

Description

A wrapper-function for the case where the conditional probability tables are to be set in a DNA mixture model for all the auxiliary variables (O, D and Q, at all markers and all alleles).

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
setCPT(mixture, pars, markers = mixture$markers)
```

Arguments

`mixture`	A DNA mixture
`pars`	A list of parameters
`markers`	Optionally, a list of markers for which to set the tables

Details

The conditional probability tables are set according to a list of specified parameters and the observed peak heights. The function returns evidence for future use on the nodes O.

Value

List containing evidence for conditioning on peak heights.

See Also

For further details, see in particular `setCPT.O`.

setCPT.D *Set tables for auxiliary variables* D

Description

Function for setting the tables on all nodes D_r_a for mixture r in the network corresponding to one marker, using peak heights and a set of parameters.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
setCPT.D(
  domain,
  rho,
  xi,
  eta,
  phi,
  C,
  n.unknown,
  n_K,
  D,
  gets_stutter,
  can_stutter,
  stutter.from
)
```

Arguments

domain	A hugin.domain modelling one marker
rho	rho
xi	xi
eta	eta
phi	phi
C	Detection threshold
n.unknown	Number of unknown contributors
n_K	Allelecounts for known contributors
D	Names for binary nodes

gets_stutter	Does the allele receive stutter?
can_stutter	Can the allele stutter?
stutter.from	From which allele does the stutter come?

Details

The function sets conditional probabilities for auxiliary variables D_r_a, which are used for obtaining probabilities of a peak falling above resp. below the detection threshold under the current model. The conditional probability tables are defined using the gamma c.d.f. as

$$P(\texttt{D[a]} = \texttt{TRUE} | n_{ia}, n_{i,a+1}, i = 1, \ldots, k) = G(\texttt{C}).$$

Value

invisibly NULL

See Also

For further details see setCPT.O.

---

setCPT.O *Set tables for auxiliary variables* O.

Description

Function for setting the tables on all nodes O_r_a for mixture r in the network corresponding to one marker, using peak heights and a set of parameters. The returned evidence can be used for conditioning on observed peak heights.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
setCPT.O(
  domain,
  rho,
  xi,
  eta,
  phi,
  heights,
  C,
```

```
  n.unknown,
  n_K,
  O,
  gets_stutter,
  can_stutter,
  stutter.from
)
```

Arguments

domain	A hugin.domain modelling one marker
rho	rho
xi	xi
eta	eta
phi	phi
heights	Peak heights
C	Detection threshold
n.unknown	Number of unknown contributors
n_K	Allelecounts for known contributors
O	Vector of names for binary nodes
gets_stutter	Does the allele receive stutter? A boolean vector.
can_stutter	Can the allele stutter?
stutter.from	From which allele does the stutter come?

Details

The function sets probabilities in CPT according to whether a peak is observed or not. If a peak is seen at allele a, the conditional distribution is defined using the p.d.f. g for the gamma distribution presented in DNAmixtures as

$$P(\texttt{O[a]} = \texttt{TRUE} | n_{ia}, n_{i,a+1}, i = 1, \ldots, k) = g(\texttt{heights[a]})/k_a$$

and likelihood-evidence $(0, k_a)$ for the states (FALSE, TRUE) is returned. If the allele is not seen, the CPT is set using the c.d.f. G for the gamma distribution as

$$P(\texttt{O[a]} = \texttt{FALSE} | n_{ia}, n_{i,a+1}, i = 1, \ldots, k) = G(\texttt{C}),$$

in which case likelihood-evidence (1,0) is returned for this allele.

Value

A matrix with each column containing the evidence to be set on O[a] when conditioning on the peak heights.

Technical comments

For the sake of speed and saving memory, only probabilities for the CPT of `O[a]` are calculated, and not the entire table. The parent nodes are `n_i_a` and possibly `n_i_(a+1)` for all unknown contributors i; the n_{ia} for known contributors are specified through the argument `n_K`. The layout of the table relies in particular on the fact that all nodes have a statespace 0,1,2, which this is *not* checked anywhere. Thus, if other types of allele-count-nodes `n_i_a` are introduced (such as could be relevant for Amelogenin), the function should be changed accordingly.

If invalid tables are set then any subsequent propagation will fail. No roll-back functionality has so far been implemented to fix this, and the easiest solution is to re-fit the mixture model.

Note that the detection threshold and model parameters can be specified as vectors, and so it is in theory possible to use allele-specific values, if desired. This also holds for the functions `setCPT.D` and `setCPT.Q`.

`setCPT.Q` *Set tables for auxiliary variables* `Q`

Description

Function for setting the tables on all nodes `Q_r_a` for mixture `r` in the network corresponding to one marker, using peak heights and a set of parameters.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
setCPT.Q(
  domain,
  rho,
  xi,
  eta,
  phi,
  heights,
  n.unknown,
  n_K,
  Q,
  gets_stutter,
  can_stutter,
  stutter.from
)
```

Arguments

domain	A hugin.domain modelling one marker
rho	rho
xi	xi
eta	eta
phi	phi ordered as phi[c(U,K)] where U and K are the names of unknown and known contributors.
heights	Observed peak heights
n.unknown	Number of unknown contributors
n_K	Allelecounts for known contributors
Q	Names for binary nodes
gets_stutter	Does the allele receive stutter?
can_stutter	Can the allele stutter?
stutter.from	From which allele does the stutter come?

Details

The function sets conditional probabilities for auxiliary variables Q_r_a, which are used for finding probabilities of observing a smaller or larger peak than the one observed for the DNA mixture. The conditional probability tables are defined using the gamma c.d.f. as

$$P(\texttt{Q[a]} = \texttt{TRUE} | n_{ia}, n_{i,a+1}, i = 1, \ldots, k) = G(\texttt{heights[a]}).$$

Value

Scaling factors to be set as evidence when conditioning on the peak heights

Author(s)

Therese Graversen

See Also

For further details see setCPT.O.

setPeakInfo *Include or exclude peak information in the model*

Description

The function setPeakInfo is used for including in the model either the full peak height information or only information about presence and absence of peaks. After a call to setPeakInfo, the Bayesian networks in mixture$domains will represent the conditional distribution given the specified peak information. For the reverse functionality, i.e. exclusion of any such peak information, use removePeakInfo.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
setPeakInfo(mixture, pars, presence.only = FALSE)

removePeakInfo(mixture)
```

Arguments

mixture	A DNAmixture
pars	A mixpar model parameter
presence.only	Default is FALSE, which means that the full peak height information is taken into consideration. Set to TRUE, which will include only information on the presence and absence of peaks.

Details

The function setPeakInfo sets conditional probability tables using the specified model parameters, and propagates suitable evidence using either observed peak heights or the discrete presence/absence observations of peaks. Any previously entered or propagated evidence on nodes O and D will be retracted in this process.

The function removePeakInfo retracts all evidence on nodes O and D; the conditional probability tables are left unchanged.

Value

invisibly NULL

See Also

setCPT. For use of the Bayesian networks, see map.genotypes.

SGMplusDyes *Dyes used for SGMplus*

Description

Dyes used for the AmpFlSTR SGM Plus PCR Amplification Kit

Format

A list with one item per dye, each containing a vector of marker-names specifying the order in which they occur in an EPG.

Source

Applied Biosystems

---

summary.map.genotypes *Summary of best genotypes*

Description

The maximum posterior configurations of genotypes as returned by map.genotypes, but in the format of pairs of alleles rather than raw allelecounts. When a subset of alleles are considered, then the value NA is used to denote an allele outside this subset; in particular, if type="seen" is used in map.genotypes, then NA corresponds to the allele having dropped out in all of the mixtures included in the model.

Usage

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

## S3 method for class 'summary.map.genotypes'
print(x, markers = names(x), ...)
```

Arguments

object	An object returned by map.genotypes.
...	arguments passed on to other methods.
x	An object of class "summary.map.genotypes", typically returned by summary.map.genotypes.
markers	Optionally, a subset of markers to print

Value

A `data.frame` with two columns per unknown contributor under consideration and a column `Prob` containing the probability of the configuration.

Author(s)

Therese Graversen

See Also

`map.genotypes`

UKCaucasian *Allele frequencies for UK Caucasians*

Description

Database of allele frequencies for UK Caucasians.

Format

A `data.frame`

Source

The frequencies are produced from the allele counts for one (the UK Caucasian) of three British sub-populations as found on David Balding's homepage under his **likeLTD** software for mixture analysis in R.

References

https://sites.google.com/site/baldingstatisticalgenetics/software/likeltd-r-forensic-dna-r-code

---

USCaucasian *The data base of allele frequencies for 302 US Caucasian profiles.*

Description

Allele frequencies for US Caucasians.

Format

A `data.frame`.

Note

There are a few errata found after publication of the allele frequencies, but we have not updated the allele frequencies accordingly. The frequencies also did not add up to 1, and this is simply corrected by normalising within each marker.

Source

http://www.cstl.nist.gov/strbase/NISTpop.htm

---

varEst *Estimated asymptotic variance matrix for MLE*

Description

Provided that the user specifies the MLE as well as the constraints used in the maximisation, this function computes an estimate of the variance of the MLE based on the observed information. The observed information is obtained by numerically deriving the Hessian of the log-likelihood function.

IMPORTANT: This is the **DNAmixturesLite** *package, which is intended as a service to enable users to try* **DNAmixtures** *without purchasing a commercial licence for Hugin. When at all possible, we strongly recommend the use of* **DNAmixtures** *rather than this lite-version. See* https://dnamixtures.r-forge.r-project.org/ *for details on both packages.*

While the lite-version seeks to provide the full functionality of **DNAmixtures***, note that computations are much less efficient and that there are some differences in available functionality. Be aware that the present documentation is copied from* **DNAmixtures** *and thus may not accurately describe the implementation of this lite-version.*

Usage

```
varEst(mixture, mle, npars, method = "Richardson", ...)

## S3 method for class 'mixVarEst'
summary(object, transform = FALSE, ...)

## S3 method for class 'summary.mixVarEst'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  scientific = FALSE,
  print.gap = 3L,
  ...
)
```

Arguments

mixture	A DNAmixture.
mle	A mixpar, typically obtained by mixML.
npars	A list of integers specifying the number of each of the four parameters rho, eta, xi and phi. Allowed values are 0 The parameter is fixed, but might differ across mixtures. 1 The parameter is equal across mixtures. N There is one parameter for each of the N mixtures in the model.
method	Method for numeric differentiation used in hessian
...	Arguments to be passed on to other methods.
object	An object of class mixVarEst, typically obtained by a call to varEst.
transform	Should the parameterisation (μ, σ) be used? Defaults to FALSE.
x	An object of class "summary.mixVarEst".
digits	Number of significant digits to print
scientific	Should scientific notation be used?
print.gap	Distance between columns in the printing of the summary.

Details

As the user can apply highly customized constraints to the model parameters when maximising with mixML, it is a complicated matter to write a generic function for computing the asymptotic variance matrix. We have thus restricted attention to the case where each of the (multi-dimensional) parameters rho, eta, xi and phi can be either

- fixed at known values
- unknown, but common across traces
- unconstrained

Value

The mle and the estimated covariance matrix in different parametrisations

cov and mle	ρ, η, ξ, ϕ
cov.trans and mle.trans	μ, σ, ξ, ϕ
cov.res	A non-singular covariance matrix for a reparametrisation of ρ, η, ξ, ϕ, collapsing the parameters according to the specified constrains and removing one contributor from ϕ.

An integer suffix is used to indicate which mixture the parameter is associated with. In the restricted covariance matrix, all fixed parameters are left out. If parameters are equal accross mixtures, the suffix for this parameter will be .1. If parameters are unconstrained and there are N mixtures in the model, suffixes are .1, ..., .N

Examples

```
data(MC18, USCaucasian)
mixHp <- DNAmixture(list(MC18), k = 3, K = c("K1", "K2", "K3"), C = list(50),
                    database = USCaucasian)
p <- mixpar(rho = list(30), eta = list(34), xi = list(0.08),
            phi = list(c(K1 = 0.71, K3 = 0.1, K2 = 0.19)))
mlHp <- mixML(mixHp, pars = p)
## Find the estimated covariance matrix of the MLE
V.Hp <- varEst(mixHp, mlHp$mle, npars = list(rho=1,eta=1,xi=1,phi=1))
V.Hp$cov ## using (rho, eta)
V.Hp$cov.trans ## using (mu, sigma)
## The summary is a table containing the MLE and their standard errors
summary(V.Hp)


data(MC18, USCaucasian)
mixmult <- DNAmixture(list(MC18), C = list(50), k = 3, K = c("K1", "K2"), database = USCaucasian)
startpar <- mixpar(rho = list(30), eta = list(28), xi = list(0.08),
                   phi = list(c(U1 = 0.2, K1 = 0.7, K2 = 0.1)))
ml.mult <- mixML(mixmult, startpar)
Vmult <- varEst(mixmult, ml.mult$mle, list(rho=1,eta=1,xi=1,phi=1))
summary(Vmult)



## Be aware that the following two advanced examples are computationally demanding and
## typically have a runtime of several minutes with the lite-version of DNAmixtures.

data(MC15, MC18, USCaucasian)
mix <- DNAmixture(list(MC15, MC18), C = list(50, 38), k = 3, K = c("K1", "K2"),
database = USCaucasian)
startpar <- mixpar(rho = list(30, 30), eta = list(28, 35), xi = list(0.08, 0.1),
                   phi = list(c(U1 = 0.05, K1 = 0.7, K2 = 0.25),
                              c(K1 = 0.7, K2 = 0.1, U1 = 0.2)))
eqxis <- function(x){ diff(unlist(x[,"xi"])) }
## Here we set stutter equal for all traces
ml.diff <- mixML(mix, startpar, eqxis, val = 0, phi.eq = FALSE)
V.diff <- varEst(mix, ml.diff$mle, list(rho=2,eta=2,xi=1,phi=2))
summary(V.diff)

## Fixing stutter to 0.07
xival <- function(x){unlist(x[,"xi"])}
ml.eq <- mixML(mix, startpar, xival, val = c(0.07, 0.07), phi.eq = FALSE)
V.eq <- varEst(mix, ml.eq$mle, list(rho=2,eta=2,xi=0,phi=2))
summary(V.eq)
```

Index