Package ‘disclapmix’

April 22, 2018

Type Package
Title Discrete Laplace Mixture Inference using the EM Algorithm
Version 1.7
Date 2018-04-23
Author Mikkel Meyer Andersen [aut, cre],
Poul Svante Eriksen [aut]
Maintainer Mikkel Meyer Andersen <mikl@math.aau.dk>
Description Make inference in a mixture of discrete Laplace distributions using the EM algo-
rithm. This can e.g. be used for modelling the distribution of Y chromosomal haplotypes as de-
scribed in [1, 2] (refer to the URL section).
License GPL (>= 2) | file LICENSE
LinkingTo Rcpp, RcppProgress
Imports Rcpp (>= 0.11), disclap (>= 1.4), cluster (>= 1.14.4), MASS,
stats, graphics, methods, utils
Suggests knitr, ggplot2, gridExtra, ggdendro, scales, seriation,
fwsm, testthat
LazyLoad yes
BugReports https://github.com/mikldk/disclapmix/issues
VignetteBuilder knitr
SystemRequirements C++11
URL http://dx.doi.org/10.1016/j.jtbi.2013.03.009
http://arxiv.org/abs/1304.2129
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-04-22 21:51:46 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterdist</td>
<td>2</td>
</tr>
<tr>
<td>clusterprob</td>
<td>3</td>
</tr>
<tr>
<td>contributor_pairs</td>
<td>3</td>
</tr>
<tr>
<td>danes</td>
<td>4</td>
</tr>
<tr>
<td>disclapmix</td>
<td>5</td>
</tr>
<tr>
<td>generate_mixture</td>
<td>7</td>
</tr>
<tr>
<td>get_rank</td>
<td>8</td>
</tr>
<tr>
<td>haplotype_diversity</td>
<td>9</td>
</tr>
<tr>
<td>plot.disclapmixfit</td>
<td>9</td>
</tr>
<tr>
<td>plot.ranked_contrib_pairs</td>
<td>10</td>
</tr>
<tr>
<td>predict.disclapmixfit</td>
<td>11</td>
</tr>
<tr>
<td>print.contrib_pairs</td>
<td>11</td>
</tr>
<tr>
<td>print.disclapmixfit</td>
<td>12</td>
</tr>
<tr>
<td>print.ranked_contrib_pairs</td>
<td>12</td>
</tr>
<tr>
<td>rank_contributor_pairs</td>
<td>13</td>
</tr>
<tr>
<td>simulate.disclapmixfit</td>
<td>14</td>
</tr>
<tr>
<td>summary.disclapmixfit</td>
<td>15</td>
</tr>
</tbody>
</table>

### Index

16

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterdist</td>
<td>2</td>
</tr>
</tbody>
</table>

#### clusterdist

**Calculate distance between clusters**

**Description**

clusterdist calculates the distance between each pair of clusters. The distance measure is based on a symmetric Kullback-Leibler divergence.

**Usage**

```r
clusterdist(fit, ...)
```

**Arguments**

- `fit`  
  A `disclapmixfit` object.

- `...`  
  Not used

**Value**

A distance matrix

**See Also**

- `disclapmix-package`
- `disclapmix`
- `disclapmixfit`
- `clusterprob`
- `predict.disclapmixfit`
- `print.disclapmixfit`
- `summary.disclapmixfit`
- `simulate.disclapmixfit`
clusterprob

Cluster origin probabilities for haplotypes

Description

clusterprob calculates the cluster origin probabilities for haplotypes.

Usage

clusterprob(fit, newdata, ...)

Arguments

fit A disclapmixfit object.
newdata The haplotypes to predict the cluster origin probabilities for.
... Not used

Value

A matrix where the rows correspond to the rows in newdata and the sum of each row is 1.

See Also

disclapmix-package disclapmix disclapmixfit clusterdist predict.disclapmixfit print.disclapmixfit summary.disclapmixfit simulate.disclapmixfit disclap

contributor_pairs Contributor pairs from a 2 person mixture

Description

Get all possible contributor pairs from a 2 person mixture

Usage

contributor_pairs(mixture)

Arguments

mixture A list of integer vectors. The k’th element in the list is an integer vector with the alleles in the mixture at locus k.
Value

A contrib_pairs object that is a unordered list of pairs. Note, that contributor order is disregarded so that each contributor pair is only present once (and not twice as would be the case if taking order into consideration). See example usage at rank_contributor_pairs.

See Also

rank_contributor_pairs generate_mixture disclapmix-package disclapmix disclapmixfit clusterprob predict.disclapmixfit print.disclapmixfit summary.disclapmixfit simulate.disclapmixfit disclap

---

danes  

Y-STR haplotypes

Description

185 Y-STR 10 loci haplotypes

Format

A data frame with 185 observations on the following 10 loci (n is the number of times each haplotype has been observed)

DYS19
DYS389I
DYS389II
DYS390
DYS391
DYS392
DYS393
DYS437
DYS438
DYS439
n

Source

Description

Discrete Laplace Mixture Inference using the EM Algorithm

disclapmix makes inference in a mixture of Discrete Laplace distributions using the EM algorithm. After the EM algorithm has converged, the centers are moved if the marginal likelihood increases by doing so. And then the EM algorithm is run again. This continues until the centers are not moved.

Usage

\[
\text{disclapmix}(x, \text{clusters}, \text{init}_y = \text{NULL}, \text{iterations} = 100L, \text{eps} = 0.001, \\
\text{verbose} = 0L, \text{glm.method} = \text{"internal_coef"}, \text{glm.control}_\text{maxit} = 50L, \\
\text{glm.control}_\text{eps} = 1e-06, \text{init}_y\text{.method} = \text{"pam"}, \ldots)
\]

Arguments

- **x**: Dataset.
- **clusters**: The number of clusters/components to fit the model for.
- **init_y**: Initial central haplotypes, if NULL, these will be estimated as described under the **init_y_method** argument.
- **iterations**: Maximum number of iterations in the EM-algorithm.
- **eps**: Convergence stop criteria in the EM algorithm which is compared to \[
\frac{\max\{v_{\text{new}} - v_{\text{old}}\}}{\max\{v_{\text{old}}\}},
\]
where \(v\) is a matrix of each observation’s probability of belonging to a certain center.
- **verbose**: from 0 to 2 (both including): 0 for silent, 2 for extra verbose.
- **glm_method**: internal_coef, internal_dev or glm.fit. Please see details.
- **glm_control_maxit**: Integer giving the maximal number of IWLS iterations.
- **glm_control_eps**: Positive convergence tolerance epsilon; the iterations converge when \[
|x - x_{\text{old}}|/(|x| + 0.1) < \text{eps},
\]
where \(x\) = beta_correction for internal_coef and \(x\) = deviance otherwise.
- **init_y_method**: Which cluster method to use for finding initial central haplotypes, \(y\): pam (recommended) or clara. Ignored if init_y is supplied.
- **...**: Used to detect obsolete usage (when using parameters centers, use_parallel, calculate_logls or plots_prefix).
Details

**glm_method**: `internal_coef` is the fastest as it uses the relative changes in the coefficients as a stopping criterium, hence it does not need to compute the deviance until the very end. In normal situations, it would not be a problem to use this method. `internal_dev` is the reasonably fast method that uses the deviance as a stopping criterium (like `glm.fit`). `glm.fit` to use the traditional glm.fit IWLS implementation and is slow compared to the other two methods.

**init_y_method**: For `init_y_method = 'clara'`, the sampling parameters are: `samples = 100`, `sampsize = min(ceiling(nrow(x)/2), 100 + 2*clusters)` and the random number generator in R is used.

Value

A disclapmixfit object:

- **list("glm_method")**: The supplied GLM method.
- **list("init_y")**: The supplied initial central haplotypes, `init_y`.
- **list("init_y_method")**: The supplied method for choosing initial central haplotypes (only used if `init_y` is NULL).
- **list("converged")**: Whether the estimation converged or not.
- **list("x")**: Dataset used to fit the model.
- **list("y")**: The central haplotypes, `y`.
- **list("tau")**: The prior probabilities of belonging to a cluster, `tau`.
- **list("v_matrix")**: The matrix `v` of each observation’s probability of belonging to a certain cluster. The rows are in the same order as the observations in `x` used to generate this fit.
- **list("disclap_parameters")**: A matrix with the estimated discrete Laplace parameters.
- **list("glm_coef")**: The coefficients from the last GLM fit (used to calculate disclap_parameters).
- **list("model_observations")**: Number of observations.
- **list("model_parameters")**: Number of parameters in the model.
- **list("iterations")**: Number of iterations performed in total (including moving centers and re-estimating using the EM algorithm).
- **list("logL_full")**: Full log likelihood of the final model.
- **list("logL_marginal")**: Marginal log likelihood of the final model.
- **list("BIC_full")**: BIC based on the full log likelihood of the final model.
- **list("BIC_marginal")**: BIC based on the marginal log likelihood of the final model.
- **list("v_gain_iterations")**: The gain $\frac{\max\{v_{new} - v_{old}\}}{\max\{v_{old}\}}$, where `v` is `v_matrix` mentioned above, during the iterations.
- **list("tau_iterations")**: The prior probability of belonging to the centers during the iterations.
- **list("logL_full_iterations")**: Full log likelihood of the models during the iterations (only calculated when `verbose = 2L`).
- **list("logL_marginal_iterations")**: Marginal log likelihood of the models during the iterations (only calculated when `verbose = 2L`).
generate_mixture

**list**("BIC_full_iterations") BIC based on full log likelihood of the models during the iterations (only calculated when verbose = 2L).

**list**("BIC_marginal_iterations") BIC based on marginal log likelihood of the models during the iterations (only calculated when verbose = 2L).

**Author(s)**

Mikkel Meyer Andersen <mikl@math.aau.dk>

**See Also**

disclapmix-package disclapmix disclapmixfit predict.disclapmixfit print.disclapmixfit summary.disclapmixfit simulate.disclapmixfit clusterdist clusterprob glm.fit disclap pam clara

**Examples**

```r
# Generate sample database
db <- matrix(disclap::rdisclap(1000, 0.3), nrow = 250, ncol = 4)

# Add location parameters
db <- sapply(1:ncol(db), function(i) as.integer(db[, i]+13+i))

head(db)

fit1 <- disclapmix(db, clusters = 1L, verbose = 1L, glm_method = "glm.fit")
fit1$disclap_parameters
fit1$y

fit1b <- disclapmix(db, clusters = 1L, verbose = 1L, glm_method = "internal_coef")
fit1b$disclap_parameters
fit1b$y

max(abs(fit1$disclap_parameters - fit1b$disclap_parameters))

# Generate another type of database
db2 <- matrix(disclap::rdisclap(2000, 0.1), nrow = 500, ncol = 4)
db2 <- sapply(1:ncol(db2), function(i) as.integer(db2[, i]+14+i))
fit2 <- disclapmix(rbind(db, db2), clusters = 2L, verbose = 1L)
fit2$disclap_parameters
fit2$y
fit2$tau
```

---

**generate_mixture** Generate a mixture
**get_rank**

**Description**

This function can generate a mixture given a list of contributors.

**Usage**

```r
generate_mixture(profiles)
```

**Arguments**

- `profiles`: A list with profiles to mix.

**Value**

A list, e.g. for use with `contributor_pairs`. See example usage at `rank_contributor_pairs`.

**See Also**

- `contributor_pairs`
- `rank_contributor_pairs`
- `disclapmix-package`
- `disclapmix`
- `disclapmixfit`
- `clusterprob`
- `predict.disclapmixfit`
- `print.disclapmixfit`
- `summary.disclapmixfit`
- `simulate.disclapmixfit`
- `disclap`

---

**get_rank**  

*Get rank of pair*

**Description**

Get rank of pair

**Usage**

```r
get_rank(x, haplotype)
```

**Arguments**

- `x`: A ranked_contrib_pairs object.
- `haplotype`: A haplotype.
haplotype_diversity  
Calculate haplotype diversity from a disclapmixfit

Description
Calculate haplotype diversity from a disclapmixfit object. The method is based on simulating a huge database that approximates the population.

Usage
haplotype_diversity(object, nsim = 10000L)

Arguments
object  
a disclapmixfit object, usually from a result of a call to disclapmix.
nsim  
number of haplotypes to generate for calculating the haplotype diversity.

Value
The calculated haplotype diversity.

See Also
disclapmix disclapmixfit predict.disclapmixfit print.disclapmixfit summary.disclapmixfit simulate.disclapmixfit

plot.disclapmixfit  
Plot a disclapmixfit

Description
Plot a disclapmixfit object.

Usage
## S3 method for class 'disclapmixfit'
plot(x, which = 1L, clusdist = clusterdist(x), ...)

Arguments
x  
a disclapmixfit object, usually from a result of a call to disclapmix.
which  
What plot to make. 1L = clusters and their distances.
clusdist  
To use previously computed cluster distances to avoid doing the same computations twice.
...  
not used
Value

A data frame with discrete Laplace distributions for each cluster and locus. Side effect: A plot.

See Also

disclapmix disclapmixfit predict.disclapmixfit print.disclapmixfit simulate.disclapmixfit summary.disclapmixfit

Examples

data(danes)
db <- as.matrix(danes[,rep(1:nrow(danes), danes$n), 1:(ncol(danes)-1)])
fit <- disclapmix(db, clusters = 4L)
plot(fit)

plot.ranked_contrib_pairs

Plot ranked contributor pairs

Description

Plot ranked contributor pairs

Usage

## S3 method for class 'ranked_contrib_pairs'
plot(x, top = NULL, ...,
  xlab = "Rank",
  ylab = "P(H1)P(H2)")

Arguments

x A ranked_contrib_pairs object.

top The top ranked number of pairs to print. NULL for all.

... Delegated to the generic plot function.

xlab Graphical parameter.

ylab Graphical parameter.
**predict.disclapmixfit**  *Predict from a disclapmixfit*

---

**Description**

Is able to predict haplotype frequencies using a *disclapmixfit* object.

**Usage**

```r
## S3 method for class 'disclapmixfit'
predict(object, newdata, ...)  
```

**Arguments**

- `object`  
  A *disclapmixfit* object
- `newdata`  
  the haplotypes in matrix format to estimate haplotype probabilities for
- `...`  
  not used

**See Also**

- `disclapmix`
- `disclapmixfit`
- `print.disclapmixfit`
- `summary.disclapmixfit`
- `simulate.disclapmixfit`
- `plot.disclapmixfit`
- `clusterprob`

---

**print.contrib_pairs**  *Print contributor pairs*

---

**Description**

Print contributor pairs

**Usage**

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

**Arguments**

- `x`  
  A *contrib_pairs* object.
- `...`  
  Ignored
print.disclapmixfit  Print a disclapmixfit

Description

Prints a disclapmixfit object.

Usage

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

Arguments

x  a disclapmixfit object, usually from a result of a call to disclapmix.
...
not used

See Also

disclapmix disclapmixfit predict.disclapmixfit summary.disclapmixfit simulate.disclapmixfit plot.disclapmixfit

print.ranked_contrib_pairs  Print ranked contributor pairs

Description

Print ranked contributor pairs

Usage

## S3 method for class 'ranked_contrib_pairs'
print(x, top = 5L,
      hide_non_varying_loci = TRUE, ...)

Arguments

x  A ranked_contrib_pairs object.
top  The top ranked number of pairs to print/plot. NULL for all.
hide_non_varying_loci  Whether to hide alleles on loci that do not vary.
...
Ignored
Separate a 2 person mixture by ranking the possible contributor pairs.

**Usage**

```
rank_contributor_pairs(contri_pairs, fit, max_rank = NULL)
```

**Arguments**

- `contri_pairs`: A `contri_pairs` object obtained from `contributor_pairs`.
- `fit`: A `disclapmixfit` object.
- `max_rank`: Not used. Reserved for future use.

**Value**

A `ranked_contri_pairs` object that is basically an order vector and the probabilities for each pair (in the same order as given in `contri_pairs`), found by using `fit`. Note, that contributor order is disregarded so that each contributor pair is only present once (and not twice as would be the case if taking order into consideration).

**Examples**

```
data(danes)
db <- as.matrix(danes[rep(1L:nrow(danes), danes$n), 1L:(ncol(danes) - 1L)])
set.seed(1)
true_contri <- sample(1L:nrow(db), 2L)
h1 <- db[true_contri[1L], ]
h2 <- db[true_contri[2L], ]
db_ref <- db[-true_contri, ]
h1h2 <- c(paste(h1, collapse = ";"), paste(h2, collapse = ";"))
tab_db <- table(apply(db, 1, paste, collapse = ";"))
tab_db_ref <- table(apply(db_ref, 1, paste, collapse = ";"))
tab_db[h1h2]
tab_db_ref[h1h2]
```
rm(db) # To avoid use by accident

mixture <- generate_mixture(list(h1, h2))

possible_contributors <- contributor_pairs(mixture)
possible_contributors

fits <- lapply(1:5L, function(clus) disclapmix(db_ref, clusters = clus))

best_fit_BIC <- fits[[which.min(sapply(fits, function(fit) fit$bic_marginal))]]
best_fit_BIC

ranked Contributors_BIC <- rank_contributor_pairs(possible_contributors, best_fit_BIC)
ranked Contributors_BIC

plot(ranked Contributors_BIC, top = 10L, type = "b")

get_rank(ranked Contributors_BIC, h1)

---

**simulate.disclapmixfit**

*Simulate from a disclapmixfit*

**Description**

Simulate from a **disclapmixfit** object.

**Usage**

```r
## S3 method for class 'disclapmixfit'
simulate(object, nsim = 1L, seed = NULL, ...)
```

**Arguments**

- **object**: a **disclapmixfit** object, usually from a result of a call to disclapmix.
- **nsim**: number of haplotypes to generate.
- **seed**: not used
- **...**: not used

**Value**

A matrix where the rows correspond to the simulated haplotypes.

**See Also**

`disclapmix`, `disclapmixfit`, `predict.disclapmixfit`, `print.disclapmixfit`, `plot.disclapmixfit`, `summary.disclapmixfit`
summary.disclapmixfit

Summary of a disclapmixfit

Description

Summary of a disclapmixfit object.

Usage

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

Arguments

object a disclapmixfit object, usually from a result of a call to disclapmix.

... not used

See Also

disclapmix disclapmixfit predict.disclapmixfit print.disclapmixfit simulate.disclapmixfit clusterdist
Index

*Topic clusters
  clusterdist, 2
  clusterprob, 3
  disclapmix, 5

*Topic datasets
  danes, 4

*Topic deconvolution
  contributor_pairs, 3
  generate_mixture, 7
  rank_contributor_pairs, 13

*Topic disclapmix
  disclapmix, 5

*Topic distance
  clusterdist, 2
  clusterprob, 3

*Topic eps
  disclapmix, 5

*Topic mixture
  contributor_pairs, 3
  generate_mixture, 7
  rank_contributor_pairs, 13

*Topic plot
  plot.disclapmixfit, 9

*Topic predict
  predict.disclapmixfit, 11

*Topic print
  haplotype_diversity, 9
  print.disclapmixfit, 12
  simulate.disclapmixfit, 14
  summary.disclapmixfit, 15

*Topic separation
  contributor_pairs, 3
  generate_mixture, 7
  rank_contributor_pairs, 13

danes, 4
disclap, 2–4, 7, 8, 13
disclapmix, 2–4, 5, 7–15
disclapmix-package (disclapmix), 5
disclapmixfit, 2–4, 6–15
disclapmixfit (disclapmix), 5

  generate_mixture, 4, 7, 13
  get_rank, 8
  glm.fit, 7

  haplotype_diversity, 9

  pam, 7
  plot, 10
  plot.disclapmixfit, 9, 11, 12, 14
  plot.ranked_contrib_pairs, 10
  predict.disclapmixfit, 2–4, 7–10, 11, 12–15
  print.contrib_pairs, 11
  print.disclapmixfit, 2–4, 7–11, 12, 13–15
  print.ranked_contrib_pairs, 12

  rank_contributor_pairs, 4, 8, 13

  simulate.disclapmixfit, 2–4, 7–13, 14, 15
  summary.disclapmixfit, 2–4, 7–14, 15

clara, 7
clusterdist, 2, 3, 7, 15
clusterprob, 2, 3, 4, 7, 8, 11, 13
contributor_pairs, 3, 8, 13