Package ‘HTSCluster’

December 10, 2013

Type Package

Title Clustering high throughput sequencing (HTS) data

Version 2.0.1

Date 2013-10-18

Author Andrea Rau, Gilles Celeux, Marie-Laure Martin-Magniette, Cathy Maugis-Rabusseau

Maintainer Andrea Rau `<andrea.rau@jouy.inra.fr>`

Depends R (>= 2.10.0), plotrix, ggplot2, RColorBrewer

Imports edgeR

Description This package implements a Poisson mixture model to cluster observations (e.g., genes) in high throughput sequencing data. Parameter estimation is performed using either the EM or CEM algorithm, and the ICL criterion is used for model selection (i.e., to choose the number of clusters).

License GPL (>= 3)

LazyLoad yes

Repository CRAN

Repository/R-Forge/Project htsfilter

Repository/R-Forge/Revision 63

Repository/R-Forge/DateTimeStamp 2013-10-18 09:39:24

Date/Publication 2013-12-10 14:53:44

NeedsCompilation no

R topics documented:

`HTSCluster-package` *Clustering high throughput sequencing (HTS) data*

Description

This package implements estimation of a Poisson mixture model to cluster observations (e.g., genes) in high throughput sequencing data. Parameter estimation is performed using either the EM or CEM algorithm, and the BIC or ICL criteria are used for model selection (i.e., to choose the number of clusters).

Details

Package:	HTSCluster
Type:	Package
Version:	2.0.1
Date:	2013-10-18
License:	GPL (>=3)
LazyLoad:	yes

Author(s)

Andrea Rau, Gilles Celeux, Marie-Laure Martin-Magniette, Cathy Maugis-Rabusseau

Maintainer: Andrea Rau <`andrea.rau@jouy.inra.fr`>

References

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at `http://hal.inria.fr/inria-00638082`.

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 2000 observations

simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions

## Run the PMM model for g = 3
## "TC" library size estimate, EM algorithm

run <- PoisMixClus(y, g = 3, conds = conds, lib.type = "TC")

## Estimates of pi and lambda for the selected model

pi.est <- run$pi
lambda.est <- run$lambda


## Not run: PMM for 4 total clusters, with one fixed class
## "TC" library size estimate, EM algorithm
##
## run <- PoisMixClus(y, g = 3, lib.type = "TC", conds = conds,
##    fixed.lambda = list(c(1,1,1)))
##
##
## Not run: PMM model for 4 clusters, with equal proportions
## "TC" library size estimate, EM algorithm
##
## run <- PoisMixClus(y, g = 4, lib.type = "TC", conds = conds,
##     equal.proportions = TRUE)
##
##
## Not run: PMM model for g = 1, ..., 10 clusters, Split Small-EM init
##
## run1.10 <- PoisMixClusWrapper(y, gmin = 1, gmax = 10, conds = conds,
## lib.type = "TC")
##
##
## Not run: PMM model for g = 1, ..., 10 clusters, Small-EM init
##
## run1.10bis <-  <- PoisMixClusWrapper(y, gmin = 1, gmax = 10, conds = conds,
## lib.type = "TC", split.init = FALSE)
##
##
## Not run: previous model equivalent to the following
##
## for(K in 1:10) {
```

```
## run <- PoisMixClus(y, g = K, conds = conds, lib.type = "TC")
## }
```

highDimensionARI *Calculate ARI for high-dimensional data via data splits*

Description

This function is used to calculate Adjusted Rand Index (ARI) values for high-dimensional data.

Usage

```
highDimensionARI(x, y, splits = 2, verbose = FALSE)
```

Arguments

`x`	Vector of classification labels
`y`	Vector of classification labels
`splits`	Number of subsets data should be split into
`verbose`	`TRUE` if verbose output is desired

Value

Value of Adjusted Rand Index for samples `x` and `y`

Author(s)

Andrea Rau <`andrea.rau@jouy.inra.fr`>

References

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C. (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at `http://hal.inria.fr/inria-00638082`.

`Init` *Parameter initialization for a Poisson mixture model.*

Description

These functions implement a variety of initialization methods for the parameters of a Poisson mixture model: the Small EM initialization strategy (`emInit`) described in Rau et al. (2011), a K-means initialization strategy (`kmeanInit`) that is itself used to initialize the small EM strategy, the splitting small-EM initialization strategy (`splitEMInit`) described in Papastamoulis et al. (2012), and a function to initialize a small-EM strategy using the posterior probabilities (`probaPostInit`) obtained from a previous run with one fewer cluster following the splitting strategy.

Usage

```
emInit(y, g, conds, lib.size, lib.type, alg.type = "EM",
    init.runs, init.iter, fixed.lambda, equal.proportions, verbose)

kmeanInit(y, g, conds, lib.size, lib.type, fixed.lambda,
    equal.proportions)

splitEMInit(y, g, conds, lib.size, lib.type, alg.type, fixed.lambda,
    equal.proportions, prev.labels, prev.probaPost, init.runs,
    init.iter, verbose)

probaPostInit(y, g, conds, lib.size, lib.type, alg.type = "EM",
    fixed.lambda, equal.proportions, probaPost.init, init.iter,
    verbose)
```

Arguments

`y`	(*n* x *q*) matrix of observed counts for *n* observations and *q* variables
`g`	Number of clusters. If `fixed.lambda` contains a list of lambda values to be fixed, *g* corresponds to the number of clusters in addition to those fixed.
`conds`	Vector of length *q* defining the condition (treatment group) for each variable (column) in `y`
`lib.size`	If `FALSE`, the library size parameter is not included in the model (i.e., the PMM-I model). If `TRUE`, the library size parameter is included in the Poisson mixture model (i.e., the PMM-II model)
`lib.type`	If `lib.size = TRUE`, the estimator to be used for the library size parameter: (“`TC`” for total count, “`UQ`” for upper quantile, “`Med`” for median, “`DESeq`” for the normalization method in the DESeq package, and “`TMM`” for the TMM normalization method (Robinson and Oshlack, 2010)
`alg.type`	Algorithm to be used for parameter estimation (“`EM`” or “`CEM`” for the EM or CEM algorithms, respectively)

`init.runs`	In the case of the Small-EM algorithm, the number of independent runs to be performed. In the case of the splitting Small-EM algorithm, the number of cluster splits to be performed in the splitting small-EM initialization. For values less than 1, an exhaustive set of splits (i.e., splitting each of the g clusters one time) is performed
`init.iter`	The number of iterations to run within each Small-EM algorithm
`fixed.lambda`	If one (or more) clusters with fixed values of lambda is desires, a list containing vectors of length d (the number of conditions). Note that the values of lambda chosen must satisfy the constraint noted in the technical report.
`equal.proportions`	If `TRUE`, the cluster proportions are set to be equal for all clusters. Default is `FALSE` (unequal cluster proportions)
`prev.labels`	A vector of length n of cluster labels obtained from the previous run (g-1 clusters)
`prev.probaPost`	An n x (g-1) matrix of the conditional probabilities of each observation belonging to each of the g-1 clusters from the previous run
`probaPost.init`	An n x (g) matrix of the conditional probabilities of each observation belonging to each of the g clusters following the splitting strategy in the `splitEMInit` function
`verbose`	If `TRUE`, include verbose output

Details

In practice, the user will not directly call the initialization functions described here; they are indirectly called for a single number of clusters through the `PoisMixClus` function (via `init.type`) or via the `PoisMixClusWrapper` function for a sequence of cluster numbers (via `gmin.init.type` and `split.init`).

To initialize parameter values for the EM and CEM algorithms, for the Small-EM strategy (Biernacki et al., 2003) we use the `emInit` function as follows. For a given number of independent runs (given by `init.runs`), the following procedure is used to obtain parameter values: first, a K-means algorithm (MacQueen, 1967) is run to partition the data into g clusters ($\hat{\boldsymbol{z}}^{(0)}$). Second, initial parameter values $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\lambda}^{(0)}$ are calculated (see Rau et al. (2011) for details). Third, a given number of iterations of an EM algorithm are run (defined by `init.iter`), using $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\lambda}^{(0)}$ as initial values. Finally, among the `init.runs` sets of parameter values, we use $\hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\pi}}$ corresponding to the highest log likelihood or completed log likelihood to initialize the subsequent full EM or CEM algorithms, respectively.

For the splitting small EM initialization strategy, we implement the approach described in Papastamoulis et al. (2012), where a cluster from a previous run (with g-1 clusters) is randomly chosen to be split into two new clusters, followed by a small EM run as described above.

Value

`pi.init`	Vector of length g containing the estimate for $\hat{\boldsymbol{\pi}}$ corresponding to the highest log likelihood (or completed log likelihood) from the chosen inialization strategy.
`lambda.init`	(d x g) matrix containing the estimate of $\hat{\boldsymbol{\lambda}}$ corresponding to the highest log likelihood (or completed log likelihood) from the chosen initialization strategy, where d is the number of conditions and g is the number of clusters.

`lambda`	(d x g) matrix containing the estimate of $\hat{\boldsymbol{\lambda}}$ arising from the splitting initialization and small EM run for a single split, where d is the number of conditions and g is the number of clusters.
`pi`	Vector of length g containing the estimate for $\hat{\boldsymbol{\pi}}$ arising from the splitting initialization and small EM run for a single split, where g is the number of clusters.
`log.like`	Log likelihood arising from the splitting initialization and small EM run for a single split.

Author(s)

Andrea Rau <`andrea.rau@jouy.inra.fr`>

References

Anders, S. and Huber, W. (2010) Differential expression analysis for sequence count data. *Genome Biology*, **11**(R106), 1-28.

Biernacki, C., Celeux, G., Govaert, G. (2003) Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Computational Statisitcs and Data Analysis*, **41**(1), 561-575.

MacQueen, J. B. (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, number 1, pages 281-297. Berkeley, University of California Press.

Papastamoulis, P., Martin-Magniette, M.-L., and Maugis-Rabusseau, C. (2012). Efficient estimation of high dimensional mixtures of Poisson generalized linear models via the EM algorithm (submitted).

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C. (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at `http://hal.inria.fr/inria-00638082`.

Robinson, M. D. and Oshlack, A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, **11**(R25).

See Also

`PoisMixClus` for Poisson mixture model estimation for a given number of clusters, `PoisMixClusWrapper` for Poisson mixture model estimation and model selection for a sequence of cluster numbers.

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 500 observations

simulate <- PoisMixSim(n = 500, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions
```

```
## Calculate initial values for lambda and pi using the Small-EM
## initialization (4 classes, PMM-II model with "TC" library size)

init.values <- emInit(y, g = 4, conds, lib.size = TRUE,
    lib.type = "TC", alg.type = "EM",
    init.runs = 50, init.iter = 10, fixed.lambda = NA,
    equal.proportions = FALSE, verbose = FALSE)
pi.init <- init.values$pi.init
lambda.init <- init.values$lambda.init
```

logLikePoisMix *Calculate the log likelihood of a Poisson mixture model*

Description

Functions to calculate the log likelihood of a Poisson mixture model, given a set of parameters, and the difference in log likelihoods for two different sets of parameters of a Poisson mixture model.

Usage

```
logLikePoisMix(y, mean, pi)
logLikePoisMixDiff(y, mean.new, pi.new, mean.old, pi.old)
```

Arguments

`y`	(n x q) matrix of observed counts for n observations and q variables
`mean`	List of length g containing the (n x q) matrices of conditional mean expression for all observations as calculated by the `PoisMixMean` function, where g represents the number of clusters
`pi`	Vector of length g containing an estimate for $\hat{\pi}$
`mean.new`	List of length g containing the (n x q) matrices of conditional mean expression for all observations for one set of parameters, as calculated by the `PoisMixMean` function, where g represents the number of clusters
`mean.old`	List of length g containing the (n x q) matrices of conditional mean expression for all observations for another set of parameters, as calculated by the `PoisMixMean` function, where g represents the number of clusters
`pi.new`	Vector of length g containing one estimate for $\hat{\pi}$
`pi.old`	Vector of length g containing another estimate for $\hat{\pi}$

Details

The `logLikePoisMix` function calculates the log likelihood for a given set of parameters in a Poisson mixture model and is used in the `PoisMixClus` function for the calculation of the BIC and ICL. The `logLikePoisMixDiff` function is used to calculate the difference in log likelihood for two different sets of parameters in a Poisson mixture model; it is used to determine convergence in the EM algorithm run by the `PoisMixClus` function.

Value

`ll`	Log likelihood for a given set of parameters (`logLikePoisMix` function) or difference in log likelihoods for two different sets of parameters (in the case of the `logLikePoisMixDiff` function)

Note

In the `logLikePoisMixDiff` function, we make use of the alternative mass function for a Poisson density proposed by Loader (2000) to avoid computational difficulties. The `logLikePoisMixDiff` function returns a default value of 100 if one or both of the log likelihoods associated with the two parameter sets takes on a value of $-\infty$.

Author(s)

Andrea Rau <andrea.rau@jouy.inra.fr>

References

Loader, C. (2000) Fast and accurate computation of binomial probabilities. Available at http://projects.scipy.org/scipy/raw-attachment/ticket/620/loader2000Fast.pdf.

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at http://hal.inria.fr/inria-00638082.

See Also

`PoisMixClus` for Poisson mixture model estimation and model selection; `PoisMixMean` to calculate the per-cluster conditional mean of each observation

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", low cluster separation
## n = 200 observations

simulate <- PoisMixSim(n = 200, libsize = "A", separation = "low")
y <- simulate$y
conds <- simulate$conditions
w <- rowSums(y)               ## Estimate of w
r <- table(conds)             ## Number of replicates per condition
d <- length(unique(conds))    ## Number of conditions
s <- colSums(y) / sum(y)      ## TC estimate of lib size
s.dot <- rep(NA, d)           ## Summing lib size within conditions
for(j in 1:d) s.dot[j] <- sum(s[which(conds == unique(conds)[j])]);

## Initial guess for pi and lambda
g.true <- 4
```

```
pi.guess <- simulate$pi
## Recalibrate so that (s.dot * lambda.guess) = 1
lambda.sim <- simulate$lambda
lambda.guess <- matrix(NA, nrow = d, ncol = g.true)
for(k in 1:g.true) {
    tmp <- lambda.sim[,k]/sum(lambda.sim[,k])
    lambda.guess[,k] <- tmp/s.dot
}

## Run the PMM-II model for g = 4
## with EM algorithm and "TC" library size parameter
run <- PoisMixClus(y, g = 4, lib.size = TRUE,
   lib.type = "TC", conds = conds)
pi.est <- run$pi
lambda.est <- run$lambda

## Mean values for each of the parameter sets
mean.guess <- PoisMixMean(y, 4, conds, s, lambda.guess)
mean.est <- PoisMixMean(y, 4, conds, s, lambda.est)

## Log likelihood of each parameter set,
## and difference in log likelihoods
LL.guess <- logLikePoisMix(y, mean.guess, pi.guess)$ll
LL.est <- logLikePoisMix(y, mean.est, pi.est)$ll
LL.diff <- logLikePoisMixDiff(y, mean.guess, pi.guess, mean.est, pi.est)

LL.guess - LL.est   ## -12841.11
LL.diff             ## -12841.11
```

plot.HTSCluster   *Visualize results from clustering using a Poisson mixture model*

Description

A function to visualize the clustering results obtained from a Poisson mixture model.

Usage

```
## S3 method for class 'HTSCluster'
plot(x, data, file.name = FALSE,
    graphs = c("map", "map.bycluster", "lambda"),
    lambda.plot = c("bar"), ...)
## S3 method for class 'HTSClusterWrapper'
plot(x, file.name = FALSE,
    graphs = c("ICL", "BIC"), ...)
```

Arguments

`x`	An object of class `"HTSCluster"` or `"HTSClusterWrapper"`
`data`	(*n* x *q*) matrix of observed counts for *n* observations and *q* variables
`file.name`	Optional file name if plots are to be saved in a PDF file.
`graphs`	Type of graph to be included in plots. May be equal to `"map"`, `"may.bycluster"`, `"weighted.histograms"`, and/or `"lambda"` for objects of class `"HTSCluster"` and `c("ICL", "BIC")` for objects of class `"HTSClusterWrapper"`
`lambda.plot`	Type of plot to be used for the `lambda` plot, `"bar"` (default), `"pie"` or `"stars"`
`...`	Additional arguments (mainly useful for plotting)

Details

For objects of class `"HTSCluster"`, the plotting function provides the possibility for the following visualizations:

1) A histogram of maximum conditional probabilities across all clusters.

2) Per-cluster boxplots of maximum conditional probabilities.

3) Weighted histograms of observation profiles (with weights equal to the corresponding conditional probability for each observation in each cluster), plotted independently for each variable. Fitted densities after fitting the Poisson mixture model are overlaid in red.

4) A global view of $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$ values for the selected model. When the number of conditions <= 2, bar heights represent the value of $\boldsymbol{\lambda}_k$ for each cluster, and bar width corresponds to the value of $\boldsymbol{\pi_k}$. When the number of conditions > 2, either bar, pie or star plots are drawn for each cluster to illustrate the relative value of $\boldsymbol{\lambda}_k$ in each condition, where for the star plots, half-circle radiuses reflect the corresponding value of $\boldsymbol{\pi_k}$.

For objects of class `"HTSClusterWrapper"`, the plotting function provides the possibility for one or both of the following visualizations:

1) ICL plot for all fitted models.

2) BIC plot for all fitted models.

Author(s)

Andrea Rau

References

Andrea Rau, Gilles Celeux, Marie-Laure Martin-Magniette, and Cathy Maugis-Rabusseau (2011). Clustering high-throughput sequencing data with Poisson mixture models. *Technical report* RR-7786, Inria Saclay – Ile-de-France.

See Also

`PoisMixClus`, `PoisMixClusWrapper`

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 2000 observations
simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions

## Run the PMM-II model for g = 3
## "TC" library size estimate, EM algorithm
run <- PoisMixClus(y, g = 3, lib.size = TRUE,
    lib.type = "TC", conds = conds, init.type = "small-em")

## Visualization of results (not run):
## plot(run)
```

PoisMixClus *Poisson mixture model estimation and model selection*

Description

These functions implement the EM and CEM algorithms for parameter estimation in a Poisson mixture model for clustering high throughput sequencing observations (e.g., genes) for a single number of clusters (`PoisMixClus`) or a sequence of cluster numbers (`PoisMixClusWrapper`). Parameters are initialized using a Small-EM strategy as described in Rau et al. (2011) or the splitting small-EM strategy described in Papastamoulis et al. (2012), and model selection is performed using the ICL criteria. Note that these functions implement the PMM-I and PMM-II models described in Rau et al. (2011).

Usage

```
PoisMixClus(y, g, conds, lib.size = TRUE, lib.type = "TMM",
    init.type = "small-em", init.runs = 20, init.iter = 10,
    alg.type = "EM", cutoff = 10e-6, iter = 1000, fixed.lambda = NA,
    equal.proportions = FALSE, prev.labels = NA,
    prev.probaPost = NA, verbose = FALSE, interpretation = "sum")

PoisMixClusWrapper(y, gmin = 1, gmax, conds, lib.size = TRUE,
    lib.type = "TMM", gmin.init.type = "small-em",
    init.runs = 20, init.iter = 10, split.init = TRUE, alg.type = "EM",
    cutoff = 10e-6, iter = 1000, fixed.lambda = NA,
    equal.proportions = FALSE, verbose = FALSE, interpretation = "sum")
```

Arguments

`y`	(*n* x *q*) matrix of observed counts for *n* observations and *q* variables
`g`	Number of clusters (a single value). If `fixed.lambda` contains a list of lambda values to be fixed, `g` corresponds to the number of clusters in addition to those fixed.
`gmin`	The minimum number of clusters in a sequence to be tested, where `gmin` corresponds to the minimum number of clusters in addition to those fixed.
`gmax`	The maximum number of clusters in a sequence to be tested, where `gmax` corresponds to the maximum number of clusters in addition to those fixed.
`conds`	Vector of length *q* defining the condition (treatment group) for each variable (column) in `y`
`lib.size`	If `FALSE`, the library size parameter is not included in the model (i.e., the PMM-I model). If `TRUE`, the library size parameter is included in the Poisson mixture model (i.e., the PMM-II model)
`lib.type`	If `lib.size = TRUE`, the estimator to be used for the library size parameter: ("`TC`" for total count, "`UQ`" for upper quantile, "`Med`" for median, "`DESeq`" for the normalization method in the DESeq package, and "`TMM`" for the TMM normalization method.
`init.type`	Type of initialization strategy to be used ("`small-em`" for the Small-EM strategy described in Rau et al. (2011), and "`kmeans`" for a simple *K*-means initialization)
`gmin.init.type`	Type of initialization strategy to be used for the minimum number of clusters in a sequence (`gmin`): ("`small-em`" for the Small-EM strategy described in Rau et al. (2011), and "`kmeans`" for a simple *K*-means initialization)
`init.runs`	Number of iterations to be used for the Small-EM strategy described in Rau et al. (2011), with a default value of 20
`init.iter`	Number of iterations to be used within each run for the Small-EM strategry, with a default value of 10
`split.init`	If `TRUE`, the splitting initialization strategy of Papastamoulis et al. (2012) will be used for cluster sizes (`gmin`+1, ..., `gmax`). If `FALSE`, the initialization strategy specified in `gmin.init.type` is used for all cluster sizes in the sequence.
`alg.type`	Algorithm to be used for parameter estimation ("`EM`" or "`CEM`")
`cutoff`	Cutoff to declare algorithm convergence (in terms of differences in log likelihoods from one iteration to the next)
`iter`	Maximum number of iterations to be run for the chosen algorithm
`fixed.lambda`	If one (or more) clusters with fixed values of lambda is desired, a list containing vectors of length *d* (the number of conditions). Note that the values of lambda chosen must satisfy the constraint noted in the technical report.
`equal.proportions`	If `TRUE`, the cluster proportions are set to be equal for all clusters. Default is `FALSE` (unequal cluster proportions).

`prev.labels`	A vector of length *n* of cluster labels obtained from the previous run (g-1 clusters) to be used with the splitting small-EM strategy described in described in Papastamoulis et al. (2012). For other initialization strategies, this parameter takes the value NA
`prev.probaPost`	An *n* x (*g*-1) matrix of the conditional probabilities of each observation belonging to each of the *g*-1 clusters from the previous run, to be used with the splitting small-EM strategy of described in Papastamoulis et al. (2012). For other initialization strategies, this parameter takes the value NA
`verbose`	If `TRUE`, include verbose output
`interpretation`	If `"sum"`, cluster behavior is interpreted with respect to overall gene expression level (sums per gene), otherwise for `"mean"`, cluster behavior is interpreted with respect to mean gene expression (means per gene).

Details

Output of `PoisMixClus` is an S3 object of class `HTSCluster`, and output of `PoisMixClusWrapper` is an S3 object of class `HTSClusterWrapper`.

In a Poisson mixture model, the data $\mathbf{y}$ are assumed to come from *g* distinct subpopulations (clusters), each of which is modeled separately; the overall population is thus a mixture of these subpopulations. In the case of a Poisson mixture model with *g* components, the model may be written as

$$f(\mathbf{y}; g, \mathbf{\Psi}_g) = \prod_{i=1}^{n} \sum_{k=1}^{g} \pi_k \prod_{j=1}^{d} \prod_{l=1}^{r_j} P(y_{ijl}; \boldsymbol{\theta}_k)$$

for $i = 1, \ldots, n$ observations in $l = 1, \ldots, r_j$ replicates of $j = 1, \ldots, d$ conditions (treatment groups), where $P(\cdot)$ is the standard Poisson density, $\mathbf{\Psi}_g = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\theta}')$, $\boldsymbol{\theta}'$ contains all of the parameters in $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_g$ assumed to be distinct, and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_g)'$ are the mixing proportions such that π_k is in (0,1) for all *k* and $\sum_k \pi_k = 1$.

We consider two possible parameterizations for the mean $\boldsymbol{\theta}_k = (\mu_{ijlk})$. In the first, called the PMM-I, we consider

$$\mu_{ijlk} = w_i \lambda_{jk}$$

where w_i corresponds to the expression level of observation *i* and $\boldsymbol{\lambda}_k = (\lambda_{1k}, \ldots, \lambda_{dk})$ corresponds to the clustering parameters that define the profiles of the genes in cluster *k* across all variables. In the second parameterization, called the PMM-II, we consider

$$\mu_{ijlk} = w_i s_{jl} \lambda_{jk}$$

where w_i and $\boldsymbol{\lambda}_k$ are as before and s_{jl} is the normalized library size (a fixed constant) for replicate *l* of condition *j*. See Rau et al. (2011) for more details on the PMM-I and PMM-II.

There are two approaches to estimating the parameters of a finite mixture model and obtaining a clustering of the data: the estimation approach (via the EM algorithm) and the clustering approach (via the CEM algorithm). Parameter initialization is done using a Small-EM strategy as described in Rau et al. (2011) via the `emInit` function. Model selection may be performed using the BIC or ICL criteria.

Value

`lambda`	(d x g) matrix containing the estimate of $\hat{\lambda}$
`pi`	Vector of length g containing the estimate of $\hat{\pi}$
`labels`	Vector of length n containing the cluster assignments of the n observations
`probaPost`	Matrix containing the conditional probabilities of belonging to each cluster for all observations
`log.like`	Value of log likelihood
`BIC`	Value of BIC criterion
`ICL`	Value of ICL criterion
`alg.type`	Estimation algorithm used; matches the argument `alg.type` above)
`lib.size`	TRUE if library size included in the model (matches the argument `alg.type` above)
`lib.type`	Type of library size normalization used (if `lib.size = TRUE`; matches the argument `lib.type` above)
`s`	Library size normalization factors used (if `lib.size = TRUE`)
`conds`	Conditions specified by user
`loglike.all`	Log likelihoods calculated for each of the fitted models for cluster sizes `gmin`, ..., `gmax`
`ICL.all`	ICL values calculated for each of the fitted models for cluster sizes `gmin`, ..., `gmax`
`select.results`	Object of class `HTSCluster` giving the results from the model chosen via the ICL criterion
`all.results`	List of objects of class `HTSCluster` giving the results for all models for cluster sizes `gmin`, ..., `gmax`

Author(s)

Andrea Rau <andrea.rau@jouy.inra.fr>

References

Anders, S. and Huber, W. (2010) Differential expression analysis for sequence count data. *Genome Biology*, **11**(R106), 1-28.

Papastamoulis, P., Martin-Magniette, M.-L., and Maugis-Rabusseau, C. (2012). Efficient estimation of high dimensional mixtures of Poisson generalized linear models via the EM algorithm (submitted).

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at http://hal.inria.fr/inria-00638082.

See Also

probaPost for the calculation of the conditional probability of belonging to a cluster; PoisMixMean for the calculation of the per-cluster conditional mean of each observation; logLikePoisMix for the calculation of the log likelihood of a Poisson mixture model; emInit and kmeanInit for the Small-EM parameter initialization strategy

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 200 observations

simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions

## Run the PMM model for g = 3
## "TC" library size estimate, EM algorithm

run <- PoisMixClus(y, g = 3, conds = conds, lib.type = "TC")

## Estimates of pi and lambda for the selected model

pi.est <- run$pi
lambda.est <- run$lambda


## Not run: PMM for 4 total clusters, with one fixed class
## "TC" library size estimate, EM algorithm
##
## run <- PoisMixClus(y, g = 3, lib.type = "TC", conds = conds,
##    fixed.lambda = list(c(1,1,1)))
##
##
## Not run: PMM model for 4 clusters, with equal proportions
## "TC" library size estimate, EM algorithm
##
## run <- PoisMixClus(y, g = 4, lib.type = "TC", conds = conds,
##     equal.proportions = TRUE)
##
##
## Not run: PMM model for g = 1, ..., 10 clusters, Split Small-EM init
##
## run1.10 <- PoisMixClusWrapper(y, gmin = 1, gmax = 10, conds = conds,
## lib.type = "TC")
##
##
## Not run: PMM model for g = 1, ..., 10 clusters, Small-EM init
##
```

```
## run1.10bis <-  <- PoisMixClusWrapper(y, gmin = 1, gmax = 10, conds = conds,
## lib.type = "TC", split.init = FALSE)
##
##
## Not run: previous model equivalent to the following
##
## for(K in 1:10) {
## run <- PoisMixClus(y, g = K, conds = conds, lib.type = "TC")
## }
```

PoisMixMean — *Calculate the conditional per-cluster mean of each observation*

Description

This function is used to calculate the conditional per-cluster mean expression for all observations. This value corresponds to $\boldsymbol{\mu} = (\mu_{ijlk}) = (\hat{w}_i \hat{\lambda}_{jk})$ for the PMM-I model and $\boldsymbol{\mu} = (\mu_{ijlk}) = (\hat{w}_i s_{jl} \hat{\lambda}_{jk})$ for the PMM-II model.

Usage

```
PoisMixMean(y, g, conds, s, lambda)
```

Arguments

`y`	(*n* x *q*) matrix of observed counts for *n* observations and *q* variables
`g`	Number of clusters
`conds`	Vector of length *q* defining the condition (treatment group) for each variable (column) in `y`
`s`	Estimate of normalized per-variable library size
`lambda`	(*d* x g) matrix containing the current estimate of lambda, where *d* is the number of conditions (treatment groups) and g is the number of clusters

Value

A list of length g containing the (*n* x *q*) matrices of mean expression for all observations, conditioned on each of the g clusters

Author(s)

Andrea Rau <andrea.rau@jouy.inra.fr>

References

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C. (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at http://hal.inria.fr/inria-00638082.

See Also

PoisMixClus for Poisson mixture model estimation and model selection

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 200 observations

simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions
s <- colSums(y) / sum(y)  ## TC estimate of lib size

## Run the PMM-II model for g = 3
## "TC" library size estimate, EM algorithm

run <- PoisMixClus(y, g = 3, lib.size = TRUE,
    lib.type = "TC", conds = conds)
pi.est <- run$pi
lambda.est <- run$lambda

## Calculate the per-cluster mean for each observation
means <- PoisMixMean(y, g = 3, conds, s, lambda.est)
```

PoisMixSim *Simulate data from a Poisson mixture model*

Description

This function simulates data from a Poisson mixture model, as described by Rau et al. (2011). Data are simulated with varying expression level (w_i) for 4 clusters. Clusters may be simulated with "high" or "low" separation, and three different options are available for the library size setting: "equal", "A", and "B", as described by Rau et al. (2011).

Usage

```
PoisMixSim(n = 2000, libsize, separation)
```

Arguments

n	Number of observations
libsize	The type of library size difference to be simulated ("equal", "A", or "B", as described by Rau et al. (2011))
separation	Cluster separation ("high" or "low", as described by Rau et al. (2011))

Value

`y`	(n x q) matrix of simulated counts for n observations and q variables
`labels`	Vector of length n defining the true cluster labels of the simulated data
`pi`	Vector of length 4 (the number of clusters) containing the true value of $\boldsymbol{\pi}$
`lambda`	(d x 4) matrix of $\boldsymbol{\lambda}$ values for d conditions (3 in the case of `libsize` = “`equal`” or “`A`”, and 2 otherwise) in 4 clusters (see note below)
`w`	Row sums of `y` (estimate of $\hat{w}$)
`conditions`	Vector of length q defining the condition (treatment group) for each variable (column) in `y`

Note

If one or more observations are simulated such that all variables have a value of 0, those rows are removed from the data matrix; as such, in some cases the simulated data `y` may have less than `n` rows.

The PMM-I model includes the parameter constraint $\sum_k \lambda_{jk} r_j = 1$, where r_j is the number of replicates in condition (treatment group) j. Similarly, the parameter constraint in the PMM-II model is $\sum_j \sum_l \lambda_{jk} s_{jl} = 1$, where s_{jl} is the library size for replicate l of condition j. The value of `lambda` corresponds to that used to generate the simulated data, where the library sizes were set as described in Table 2 of Rau et al. (2011). However, due to variability in the simulation process, the actually library sizes of the data `y` are not exactly equal to these values; this means that the value of `lambda` may not be directly compared to an estimated value of $\hat{\boldsymbol{\lambda}}$ as obtained from the `PoisMixClus` function.

Author(s)

Andrea Rau <`andrea.rau@jouy.inra.fr`>

References

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C. (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at `http://hal.inria.fr/inria-00638082`.

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 200 observations

simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions
```

probaPost — *Calculate the conditional probability of belonging to each cluster in a Poisson mixture model*

Description

This function computes the conditional probabilities t_{ik} that an observation i arises from the k^{th} component for the current value of the mixture parameters.

Usage

```
probaPost(y, g, conds, pi, s, lambda)
```

Arguments

`y`	(n x q) matrix of observed counts for n observations and q variables
`g`	Number of clusters
`conds`	Vector of length q defining the condition (treatment group) for each variable (column) in `y`
`pi`	Vector of length `g` containing the current estimate of $\hat{\boldsymbol{\pi}}$
`s`	Vector of length q containing the estimates for the normalized library size parameters for each of the q variables in `y`
`lambda`	(d x `g`) matrix containing the current estimate $\boldsymbol{\lambda}$, where d is the number of conditions (treatment groups)

Value

`t`	(n x `g`) matrix made up of the conditional probability of each observation belonging to each of the `g` clusters

Note

If all values of t_{ik} are 0 (or nearly zero), the observation is assigned with probability one to belong to the cluster with the closest mean (in terms of the Euclidean distance from the observation). To avoid calculation difficulties, extreme values of t_{ik} are smoothed, such that those smaller than 1e-10 or larger than 1-1e-10 are set equal to 1e-10 and 1-1e-10, respectively.

Author(s)

Andrea Rau <andrea.rau@jouy.inra.fr>

References

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C. (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at http://hal.inria.fr/inria-00638082.

See Also

PoisMixClus for Poisson mixture model estimation and model selection; PoisMixMean to calculate the conditional per-cluster mean of each observation

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 200 observations

simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions
s <- colSums(y) / sum(y)     ## TC estimate of lib size

## Run the PMM-II model for g = 3
## "TC" library size estimate, EM algorithm

run <- PoisMixClus(y, g = 3, lib.size = TRUE, lib.type = "TC",
  conds = conds)
pi.est <- run$pi
lambda.est <- run$lambda

## Calculate the conditional probability of belonging to each cluster
proba <- probaPost(y, g = 3, conds = conds, pi = pi.est, s = s,
lambda = lambda.est)

## head(round(proba,2))
```

summary.HTSCluster *Summarize results from clustering using a Poisson mixture model*

Description

A function to summarize the clustering results obtained from a Poisson mixture model.

Usage

```
## S3 method for class 'HTSCluster'
summary(object, ...)
## S3 method for class 'HTSClusterWrapper'
summary(object, ...)
```

Arguments

`object`	An object of class `"HTSCluster"` or `"HTSClusterWrapper"`
`...`	Additional arguments

Details

The summary function for an object of class `"HTSCluster"` provides the following summary of results:

1) Number of clusters, BIC, and ICL values.

2) Number of observations across all clusters with a maximum conditional probability greater than 90 model.

3) Number of observations per cluster with a maximum conditional probability greater than 90 selected model.

4) λ values for the selected model.

5) π values for the selected model.

The summary function for an object of class `"HTSClusterWrapper"` provides the following summary of results:

1) The ICL value and number of clusters for the chosen model.

2) The ICL values associated to each number of clusters for all fitted models.

Author(s)

Andrea Rau

References

Rau, A., Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C. (2011). Clustering high-throughput sequencing data with Poisson mixture models. Inria Research Report 7786. Available at `http://hal.inria.fr/inria-00638082`.

See Also

`PoisMixClus`, `PoisMixClusWrapper`

Examples

```
set.seed(12345)

## Simulate data as shown in Rau et al. (2011)
## Library size setting "A", high cluster separation
## n = 2000 observations
simulate <- PoisMixSim(n = 200, libsize = "A", separation = "high")
y <- simulate$y
conds <- simulate$conditions

## Run the PMM-II model for g = 3
## "TC" library size estimate, EM algorithm
run <- PoisMixClus(y, g = 3, lib.size = TRUE,
    lib.type = "TC", conds = conds, init.type = "small-em")

## Summary of results:
summary(run)
```

Index