baySeq

April 19, 2010

---

baySeq-classes *baySeq - classes*

Description

The countData class defines a class for summarising count data and estiablishing prior parameters on distributions defined upon the count data. The countDataPosterior extends this class by including posterior likelihood estimates on the data. Only countData objects should be established by the user directly. countDataPosterior objects are established by the functions described in getLikelihoods.

Slots

Objects of these class should contain the following list components:

data:	Count data (matrix).
libsizes:	Vector of library size for each sample.
groups:	Group (model) structure to test on the data (list).
annotation:	Annotation data for each count (data.frame).
priors:	Prior parameter information.
posteriors:	Estimated posterior likelihoods for each group (matrix). 'countDataPosterior' class only.
estProps:	Estimated proportion of tags belonging to each group (numeric). 'countDataPosterior' class only.

Methods

Methods 'dim', '[' and 'show' have been defined for these classes.

Author(s)

Thomas J. Hardcastle

Examples

```
library(baySeq)

data(simCount)
data(libsizes)
```

```
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))

#create new 'countData' object
CD <- new("countData", data = simCount, libsizes = libsizes, groups = groups)

CD[1:10,]
dim(CD)
```

`baySeq-package` *Empirical Bayesian analysis of patterns of differential expression in count data.*

Description

This package is intended to identify differential expression in high-throughput 'count' data, such as that derived from next-generation sequencing machines. We achieve this by empirical bayesian methods, first bootstrapping to estimate prior parameters from the data and then assessing posterior likelihoods of the models proposed.

Details

Package:	baySeq
Type:	Package
Version:	1.0
Date:	2009-16-05
License:	GPL-3
LazyLoad:	yes

To use the package, construct a `countData` object and use the functions documented in getPriors to empirically determine priors on the data. Then use the functions documented in getLikelihoods to establish posterior likelihoods for the models proposed. A few convenience functions, `getTPs` and `topCounts` are also included.

The package (optionally) makes use of the 'snow' package for parallelisation of computationally intensive functions. This is highly recommended for large data sets.

See the vignette for more details.

Author(s)

Thomas J. Hardcastle

Maintainer: Thomas J. Hardcastle <tjh48@cam.ac.uk>

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

Examples

```
# See vignette for more examples.

# load test data
data(simCount)
data(libsizes)

# define hypotheses on data
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))

# construct 'countData' object
CD <- new("countData", data = simCount, libsizes = libsizes, groups = groups)
CD[1:10,]

# estimate prior distributions on 'countData' object using Poisson
# method. Other methods are available - see getPriors
CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000, takemean = TRUE)

# estimate posterior likelihoods for each row of data belonging to each hypothesis
CDPost.Poi <- getLikelihoods.Pois(CDP.Poi, prs = c(0.5, 0.5), estimatePriors = TRUE, cl =

# display the rows of data showing greatest association with the second
# hypothesis (differential expression)
topCounts(CDPost.Poi, group = 2, number = 10)

# find true positive selection rate
getTPs(CDPost.Poi, group = 2, TPs = 1:100)[1:100]
```

`factCount` *Simulated data for testing the baySeq package methods; simulated counts from a factorial design differential expression analysis*

Description

This data set is a matrix of simulated counts from a simple pairwise expression analysis. It is simulated according to a negative binomial distribution with varying parameters for each row. The first hundred rows of the data are truly differentially expressed between the first four samples and the second four samples. The second hundred rows of the data are truly differentially expressed between samples 1,2,5,6 and samples 3,4,7,8.

Usage

```
factCount
```

Format

A matrix of which each of the eight columns represents a sample, and each row some discrete data (acquired by sequencing).

Source

Simulation.

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

factlibsizes

`factlibsizes` *Simulated data for testing the baySeq package methods; simulated library sizes from a pairwise differential expression analysis*

Description

This data set is a vector of library sizes for the `factCount` matrix.

Usage

```
factlibsizes
```

Format

A vector containing library sizes for the ten libraries whose data is given in the `factCount` matrix.

Source

Simulation.

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

factCount

`getLikelihoods` *Finds posterior likelihoods for each count as belonging to some hypothesis.*

Description

These functions calculate posterior probabilities for each of the 'counts' in the countDP object belonging to each of the groups specified. The choice of function depends on the prior belief about the underlying distribution of the data. It is essential that the method used for calculating priors matches the method used for calculating the posterior probabilites.

For a comparison of the methods, see Hardcastle & Kelly, 2009.

Usage

```
getLikelihoods.Dirichlet(cDP, prs, estimatePriors = TRUE, subset = NULL, cl)
getLikelihoods.Pois(cDP, prs, estimatePriors = TRUE, subset = NULL,  distpriors
getLikelihoods.NBboot(cDP, prs, estimatePriors = TRUE, subset = NULL,
bootStraps = 2, conv = 1e-4, cl)
```

Arguments

cDP	An object of type countData.
prs	(Initial) prior probabilities for each of the groups in the 'countDP' object.
estimatePriors	Should the prior probabilities on each of the groups be estimated by bootstrap from the data? Defaults to TRUE.
subset	Numeric vector giving the subset of counts for which posterior likelihoods should be estimated.
distpriors	Should the Poisson method use an empirically derived distribution on the prior parameters of the Poisson distribution, or use the mean of the maximum likelihood estimates (default).
bootStraps	How many iterations of bootstrapping should be used in the (re)estimation of priors in the negative binomial method.
conv	If not null, bootstrapping iterations will cease if the mean squared difference between posterior likelihoods of consecutive bootstraps drops below this value.
cl	A SNOW cluster object.

Details

These functions estimate, under the assumption of various distributions, the (log) posterior likelihoods that each count belongs to a group defined by the @group slot of the countData object. The posterior likelihoods are stored on the natural log scale in the @posteriors slot of the countDataPosterior object generated by this function. This is because the posterior likelihoods are calculated in this form, and ordering of the counts is better done on these log-likelihoods than on the likelihoods.

The Dirichlet and Poisson methods produce almost identical results in simulation. The Negative Binomial method produces results with much lower false discovery rates, but takes considerably longer to run. The quality of the results of the Negative Binomial is further improved by increasing the amount of bootstrapping. However, this further increases the run time.

Filtering the data may be extremely advantageous in reducing run time. This can be done by passing a numeric vector to 'subset' defining a subset of the data for which posterior likelihoods are required.

See Hardcastle & Kelly (2009) for a full comparison of the methods.

A 'cluster' object is strongly recommended in order to parallelise the estimation of posterior likelihoods, particularly for the negative binomial method. However, passing NULL to the cl variable will allow the functions to run in non-parallel mode.

Value

A countDataPosterior object.

Author(s)

Thomas J. Hardcastle

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

countData, getPriors, topCounts, getTPs

Examples

```
library(baySeq)

# See vignette for more examples.

# Create a {countData} object and estimate priors for the
# Poisson methods.
data(simCount)
data(libsizes)
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))
CD <- new("countData", data = simCount, libsizes = libsizes, groups = groups)
CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000,
takemean = TRUE)

# Get likelihoods for data with Poisson method
CDPost.Poi <- getLikelihoods.Pois(CDP.Poi, prs = c(0.5, 0.5),
estimatePriors = TRUE, cl = NULL)

## Not run:

# Alternatively, get priors for negative binomial method
CDP.NBML <- getPriors.NB(CD, samplesize = 10^5, estimation = "ML", cl = NULL)

# Get likelihoods for data with negative binomial method with bootstrapping

CDPost.NBML <- getLikelihoods.NBboot(CDP.NBML, pres = c(0.5, 0.5),
estimatePriors = TRUE, bootStraps = 2, cl = NULL)

# Alternatively, if we have the 'snow' package installed we
# can parallelise the functions. This will usually (not always) offer
# significant performance gain.


library(snow)
cl <- makeCluster(4, 'SOCK')

CDP.NBML <- getPriors.NB(CD, samplesize = 10^5, estimation = "ML", cl = cl)
CDPost.NBML <- getLikelihoods.NBboot(CDP.NBML, pres = c(0.5, 0.5),
estimatePriors = TRUE, bootStraps = 2, cl = cl)

## End(Not run)
```

getPosteriors *An internal function in the baySeq package for calculating posterior likelihoods given likelihoods of the data.*

Description

For likelihoods of the data given a set of models, this function calculates the posterior likelihoods of the models given the data. An internal function of baySeq, which should not in general be called by the user.

Usage

```
getPosteriors(ps, prs, estimatePriors = FALSE, maxit = 100, accuracy = 1e-5)
```

Arguments

ps	A matrix containing likelihoods of the data for each count (rows) under each model (columns).
prs	(Initial) prior probabilities for each of the models.
estimatePriors	Should the prior probabilities on each of the groups be estimated by bootstrap from the data? Defaults to FALSE.
maxit	What is the maximum number of iterations that should be tried if we are bootstrapping prior probabilities from the data?
accuracy	How small should the difference in estimated priors be before we stop bootstrapping.

Details

An internal function, that will not in general be called by the user. It takes the log-likelihoods of the data given the models being tested and returns the posterior likelihoods of the models. An initial estimate for the prior likelihoods of the models is required but can be iteratively re-estimated from the data by taking the mean of the posterior likelihoods across all data.

Value

A list containing posteriors: estimated posterior likelihoods of the model for each count (log-scale) priors: estimated (or given) prior probabilities of the model

Author(s)

Thomas J. Hardcastle

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

getLikelihoods

Examples

```
# Simulate some log-likeihoods of data given models (each model
# describes one column of the 'ps' object).
ps <- log(rbind(
                cbind(runif(10000, 0, 0.1), runif(10000, 0.3, 0.9)),
                cbind(runif(10000, 0.4, 0.9), runif(1000, 0, 0.2))))

# get posterior log-likelihoods of model, estimating prior likelihoods
# of each model from the data.

pps <- getPosteriors(ps, prs <- c(0.5, 0.5), estimatePriors = TRUE)

pps$priors

pps$posteriors[,1:10]
```

`getPriors` *Estimates prior parameters for the underlying distributions of 'count' data.*

Description

These functions estimate, via maximum likelihood methods, the parameters of the underlying distributions for the different methods of describing the 'count' data.

Usage

```
getPriors.Dirichlet(cD, samplesize = 10^5, iterations = 10^3)
getPriors.Pois(cD, samplesize = 10^5, iterations = 10^3, takemean = TRUE)
getPriors.NB(cD, samplesize = 10^5, estimation = "ML", cl)
```

Arguments

`cD`	A `countData` object.
`samplesize`	How large a sample should be taken in estimating the priors?
`iterations`	Over how many iterations should we take samples and re-estimate the priors?
`takemean`	If TRUE (recommended), we take the mean of the estimated priors to define a gamma distribution. If FALSE, we use all estimated priors to define an empirical distribtion on the parameters of the gamma distribution.
`estimation`	Defaults to "ML", indicating maximum likelihood estimation of priors. Currently, the only other possibility is "QL", a quasi-likelihood method.
`cl`	A SNOW cluster object.

Details

These functions empirically estimate prior parameters for different distributions used in estimating posterior likelihoods of each count belonging to a particular group. The choice of which function to use for estimating the prior parameters will depend on the choice of which method is being used to estimate the posterior likeihoods (see getLikelihoods).

A 'cluster' object is recommended in order to estimate the priors for the negative binomial distribution. Passing NULL to this variable will cause the function to run in non-parallel mode.

Value

A `countData` object.

Author(s)

Thomas J. Hardcastle

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

`countData`, `getLikelihoods`

Examples

```
# See vignette for more examples.

# Create a {countData} object.
data(simCount)
data(libsizes)
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))
CD <- new("countData", data = simCount, libsizes = libsizes, groups =
groups)

# Estimate priors using Poisson method.
CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000,
takemean = TRUE)

## Not run:

# Alternatively, get priors for negative binomial method

CDP.NBML <- getPriors.NB(CD, samplesize = 10^5, estimation = "ML", cl = NULL)

# Alternatively, if we have the 'snow' package installed we
# can parallelise the prior estimation (for the negative binomial
# methods only). This will usually (not always) offer
# significant performance gain.

library(snow)
cl <- makeCluster(4, 'SOCK')

CDP.NBML <- getPriors.NB(CD, samplesize = 10^5, estimation = "ML", cl = cl)
```

```
## End(Not run)
```

`getTPs` *Gets the number of true positives in the top n counts selected by ranked posterior likelihoods*

Description

If the true positives are known, this function will return a vector, the ith member of which gives the number of true positives identified if the top i counts, based on estimated posterior likelihoods, are chosen.

Usage

```
getTPs(cDP, group, decreasing = TRUE, TPs)
```

Arguments

`cDP`	`countDataPosterior` object, containing posterior likelihoods for each group.
`group`	Which group should we give the counts for?
`decreasing`	Ordering on posterior likelihoods.
`TPs`	Known true positives.

Details

In the rare (or simulated) cases where the true positives are known, this function will calculate the number of true positives selected at any cutoff.

Value

A vector, the ith member of which gives the number of true positives identified if the top i counts are chosen.

Author(s)

Thomas J. Hardcastle

See Also

`countDataPosterior`

Examples

```
# Create a {countData} object and estimate priors for the Poisson methods.
data(simCount)
data(libsizes)
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))
CD <- new("countData", data = simCount, libsizes = libsizes, groups = groups)
CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000,
takemean = TRUE)

# Get likelihoods for data with Poisson method
CDPost.Poi <- getLikelihoods.Pois(CDP.Poi, prs = c(0.5, 0.5),
estimatePriors = TRUE, cl = NULL)

# If the first hundred rows in the 'simCount' matrix are known to be
# truly differentially expressed (the second hypothesis defined in the
# 'groups' list) then we find the number of true positives for the top n
# genes selected as the nth member of

getTPs(CDPost.Poi, group = 2, decreasing = TRUE, TPs = 1:100)
```

`libsizes` *Simulated data for testing the baySeq package methods; simulated library sizes from a pairwise differential expression analysis*

Description

This data set is a vector of library sizes for the `simCount` matrix.

Usage

```
libsizes
```

Format

A vector containing library sizes for the ten libraries whose data is given in the `simCount` matrix.

Source

Simulation.

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

`simCount`

`logsum` *An internal function for the baySeq package that calculates the log of the sum of the exponential of two variables*

Description

If x = log(y), where y is a vector, this function returns log(sum(y)) given x.

Usage

```
logsum(x)
```

Arguments

`x`	Vector of log-values.

Value

Numeric.

Author(s)

Thomas J. Hardcastle

See Also

`countDataPosterior`

Examples

```
x <- logsum(log(c(3, 4, 5)))
exp(x) == sum(c(3,4,5))
```

`PDgivenr` *Functions for estimating the likelihood of the data given an equivalence relation for some assumption on the distribution.*

Description

These functions aim to calculate the likelihood of the data given an equivalence relation R for some assumed distribution. They are internal functions to baySeq and should not be called by the user.

Usage

```
PDgivenr.Dirichlet(us, prior, group)
PDgivenr.Pois(us, ns, prior, group)
PDgivenr.PoisIndie(us, ns, prior, group)
PDgivenr.NBIndie(us, ns, prior, group, priorWeights)
```

Arguments

us	Number of counts.
ns	Library size.
prior	Prior parameters on distribution.
group	Hypothesised grouping of the data based on equivalence relation.
priorWeights	Weightings for each prior to be used in numerical integration.

Details

Not intended to be called by the user.

Value

Numeric value giving log-likelihood of data given distribution and grouping.

Author(s)

Thomas J. Hardcastle

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

Examples

```
data(simCount)
data(libsizes)
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))
CD <- new("countData", data = simCount, libsizes = libsizes, groups =
groups)

# Estimate priors using Poisson method.
CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000,
takemean = TRUE)

# Calculate likelihood of first row of data given the first hypothesis
# defined by the 'groups' slot of CDP.Poi by Poisson methods

PDgivenr.Pois(us = CDP.Poi@data[1,], ns = CDP.Poi@libsizes, prior =
CDP.Poi@priors$priors[[1]], group = CDP.Poi@groups[[1]])

# Calculate likelihood of each row of data given the first hypothesis
# defined by the 'groups' slot of CDP.Poi by Poisson methods

apply(CDP.Poi@data, 1, PDgivenr.Pois, ns = CDP.Poi@libsizes, prior =
CDP.Poi@priors$priors[[1]], group = CDP.Poi@groups[[1]])
```

simCount *Simulated data for testing the baySeq package methods; simulated counts from a pairwise differential expression analysis*

Description

This data set is a matrix of simulated counts from a simple pairwise expression analysis. It is simulated according to a negative binomial distribution with varying parameters for each row. The first hundred rows of the data are truly differentially expressed, the remainder have no differential expression. Library sizes for these data sets are given in libsizes.

Usage

```
simCount
```

Format

A matrix of which each of the ten columns represents a sample, and each row some discrete data (acquired by sequencing).

Source

Simulation.

References

Hardcastle T.J., and Kelly, K (2009). Empirical Bayesian methods for differential expression in count data. In submission.

See Also

libsizes

topCounts *Get the top counts corresponding to some group from a 'countData-Posterior' object*

Description

Takes posterior likelihoods and returns the counts with highest (or lowest) likelihood of association with a given group.

Usage

```
topCounts(cDP, group, decreasing = TRUE, number = 10)
```

Arguments

cDP	countDataPosterior object, containing posterior likelihoods for each group.
group	Which group should we give the counts for?
decreasing	Ordering on posterior likelihoods.
number	How many results should be returned?

Value

A dataframe of the top counts associated with some model (group), described by annotation drawn from the '@annotation' slot of the 'cDP' object and the raw data from the '@data' slot, together with the posterior log-likelihoods.

Author(s)

Thomas J. Hardcastle

See Also

countDataPosterior

Examples

```
data(simCount)
data(libsizes)

# Make 'countData' object and calculate posterior likelihoods for each
# item belonging to each hypothesis.
groups <- list(c(1,1,1,1,1,1,1,1,1,1), c(1,1,1,1,1,2,2,2,2,2))
CD <- new("countData", data = simCount, libsizes = libsizes, groups = groups)
CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000, takemean = TRUE)
CDPost.Poi <- getLikelihoods.Pois(CDP.Poi, prs = c(0.5, 0.5), estimatePriors = TRUE, cl =

# Report the top ten rows of data that have highest (log) likelihood of belonging to
# group 2 of the data (i.e., differentially expressed)

topCounts(CDPost.Poi, group = 2, number = 10)
```

Index

∗Topic **arith**
logsum, 12
∗Topic **classes**
baySeq-classes, 1
∗Topic **datasets**
factCount, 3
factlibsizes, 4
libsizes, 11
simCount, 14
∗Topic **distribution**
getLikelihoods, 4
getPriors, 8
PDgivenr, 12
∗Topic **manip**
getTPs, 10
∗Topic **models**
getLikelihoods, 4
getPosteriors, 7
getPriors, 8
PDgivenr, 12
∗Topic **package**
baySeq-package, 2
∗Topic **print**
topCounts, 14
[,countData-method
(baySeq-classes), 1
[,countDataPosterior-method
(baySeq-classes), 1

baySeq, *12*
baySeq *(baySeq-package)*, 2
baySeq-class *(baySeq-classes)*, 1
baySeq-classes, 1
baySeq-package, 2

countData, *2*, *5*, *6*, *8*, *9*
countData *(baySeq-classes)*, 1
countData-class *(baySeq-classes)*, 1
countDataPosterior, *5*, *10*, *12*, *15*
countDataPosterior *(baySeq-classes)*, 1
countDataPosterior-class *(baySeq-classes)*, 1

dim,countData-method *(baySeq-classes)*, 1
dim,countDataPosterior-method *(baySeq-classes)*, 1

factCount, 3, *4*
factlibsizes, 4, *4*

getLikelihoods, *1*, *2*, 4, *7*, *9*
getPosteriors, 7
getPriors, *2*, *6*, 8
getTPs, *2*, *6*, 10

libsizes, 11, *14*
logsum, 12

PDgivenr, 12

show,countData-method *(baySeq-classes)*, 1
show,countDataPosterior-method *(baySeq-classes)*, 1
simCount, *11*, 14

topCounts, *2*, *6*, 14