Package 'BASiCS'

August 6, 2017

Type Package

Title Bayesian Analysis of Single-Cell Sequencing data

Version 1.0.10

Date : 2017-08-06

Author Catalina Vallejos, Nils Eling, Sylvia Richardson, John Marioni

Maintainer Catalina A. Vallejos <cnvallej@uc.cl>

Description Single-cell mRNA sequencing can uncover novel cell-to-cell heterogeneity in gene expression levels in seemingly homogeneous populations of cells. However, these experiments are prone to high levels of unexplained technical noise, creating new challenges for identifying genes that show genuine heterogeneous expression within the population of cells under study. BASiCS (Bayesian Analysis of Single-Cell Sequencing data) is an integrated Bayesian hierarchical model to perform statistical analyses of single-cell RNA sequencing datasets in the context of supervised experiments (where the groups of cells of interest are known a priori, e.g. experimental conditions or cell types). BASiCS performs built-in data normalisation (global scaling) and technical noise quantification (based on spike-in genes). BASiCS provides an intuitive detection criterion for highly (or lowly) variable genes within a single group of cells. Additionally, BASiCS can compare gene expression patterns between two or more pre-specified groups of cells. Unlike traditional differential expression tools, BASiCS quantifies changes in expression that lie beyond comparisons of means, also allowing the study of changes in cell-to-cell heterogeneity. The latter are quantified via a biological over-dispersion parameter that measures residual over-dispersion (with respect to Poisson sampling) after normalisation and the removal of technical variation.

License GPL (>= 2)

Depends R (>= 3.1.0), S4Vectors, SummarizedExperiment

Imports BiocGenerics, Rcpp (>= 0.11.3), methods, coda, scran, testthat, data.table, matrixStats, graphics, KernSmooth, grDevices

Suggests knitr

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

RemoteType github

RemoteHost https://api.github.com

RemoteRepo BASiCS

RemoteUsername catavallejos

RemoteRef BioC

RemoteSha 9261b72af9c508be85264911d35c04b5c50b1f4a

GithubRepo BASiCS

GithubUsername catavallejos

GithubRef BioC

GithubSHA1 9261b72af9c508be85264911d35c04b5c50b1f4a

R topics documented:

`BASiCS_Chain-class` *The BASiCS_Chain class*

Description

Container of an MCMC sample of the BASiCS' model parameters (see Vallejos et al, 2015) as generated by the function `BASiCS_MCMC`.

Slots

`mu` MCMC chain for gene-specific expression levels $\mu[i]$, defined as true input molecules in case of technical genes (matrix with `q` columns, technical genes located at the end of the matrix, all elements must be positive numbers)

`delta` MCMC chain for gene-specific biological cell-to-cell heterogeneity hyper-parameters $\delta[i]$, biological genes only (matrix with `q.bio` columns, all elements must be positive numbers)

`phi` MCMC chain for cell-specific mRNA content normalising constants $\phi[j]$ (matrix with `n` columns, all elements must be positive numbers and the sum of its elements must be equal to `n`)

`s` MCMC chain for cell-specific capture efficiency (or amplification biases if not using UMI based counts) normalising constants $s[j]$ (matrix with `n` columns, all elements must be positive numbers)

`nu` MCMC chain for cell-specific random effects $\nu[j]$ (matrix with `n` columns, all elements must be positive numbers)

`theta` MCMC chain for technical variability hyper-parameter(s) θ (matrix, all elements must be positive, each colum represents 1 batch)

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data. PLoS Computational Biology.

Examples

```
# A BASiCS_Chain object created by the BASiCS_MCMC function.
Data = makeExampleBASiCS_Data()
MCMC_Output <- BASiCS_MCMC(Data, N = 100, Thin = 2, Burn = 2)
```

BASiCS_Chain-methods *'show' method for BASiCS_Chain objects*

Description

'show' method for BASiCS_Chain-class objects.

Usage

```
## S4 method for signature 'BASiCS_Chain'
show(object)
```

Arguments

object	A BASiCS_Chain object.

Value

Prints a summary of the properties of object.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

Examples

```
Data = makeExampleBASiCS_Data()
MCMC_Output <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 2)
```

BASiCS_DetectHVG *Detection method for highly and lowly variable genes*

Description

Functions to detect highly and lowly variable genes

Usage

```
BASiCS_DetectHVG(Chain, VarThreshold, ProbThreshold = NULL, EFDR = 0.05,
  OrderVariable = "Prob", Plot = FALSE, ...)

BASiCS_DetectLVG(Chain, VarThreshold, ProbThreshold = NULL, EFDR = 0.05,
  OrderVariable = "Prob", Plot = FALSE, ...)
```

Arguments

`Chain`	an object of class `BASiCS_Chain-class`
`VarThreshold`	Variance contribution threshold (must be a positive value, between 0 and 1)
`ProbThreshold`	Optional parameter. Posterior probability threshold (must be a positive value, between 0 and 1)
`EFDR`	Target for expected false discovery rate related to HVG/LVG detection (default = 0.05)
`OrderVariable`	Ordering variable for output. Must take values in `c("GeneIndex", "Mu", "Delta", "Sigma", "Pr`
`Plot`	If `Plot = TRUE` a plot of the gene specific expression level against HVG or LVG is generated.
`...`	Graphical parameters (see `par`).

Details

See vignette

Value

`BASiCS_DetectHVG` returns a list of 4 elements:

`Table` Matrix whose columns contain

- `GeneIndex` Vector of length `q.bio`. Gene index as in the order present in the analysed `SummarizedExperiment`
- `GeneNames` Vector of length `q.bio`. Gene name as in the order present in the analysed `SummarizedExperiment`
- `Mu` Vector of length `q.bio`. For each biological gene, posterior median of gene-specific expression levels $\mu[i]$
- `Delta` Vector of length `q.bio`. For each biological gene, posterior median of gene-specific biological cell-to-cell heterogeneity hyper-parameter $\delta[i]$
- `Sigma` Vector of length `q.bio`. For each biological gene, proportion of the total variability that is due to a cell-to-cell biological heterogeneity component.
- `Prob` Vector of length `q.bio`. For each biological gene, probability of being highly variable according to the given thresholds.
- `HVG` Vector of length `q.bio`. For each biological gene, indicator of being detected as highly variable according to the given thresholds.

`ProbThreshold` Posterior probability threshold.

`EFDR` Expected false discovery rate for the given thresholds.

`EFNR` Expected false negative rate for the given thresholds.

`BASiCS_DetectLVG` produces a similar output, replacing the element `HVG` by `LVG`, an indicator of a gene being detected as lowly variable according to the given thresholds.

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data. PLoS Computational Biology.

See Also

BASiCS_Chain-class

Examples

```
# See
help(BASiCS_MCMC)
```

BASiCS_D_TestDE *Detection of genes with changes in expression.*

Description

Function to assess changes in expression (mean and over-dispersion). This function is no longer in use and will be removed in future releases. Please use BASiCS_TestDE instead.

Usage

```
BASiCS_D_TestDE(Data = NULL, object = NULL, GeneNames = NULL,
  EpsilonM = 0.1, EpsilonD = 0.1, EviThresholdM = NULL,
  EviThresholdD = NULL, OrderVariable = "ProbDiffExp",
  GroupLabelRef = "Ref", GroupLabelTest = "Test", Plot = FALSE,
  OffSet = FALSE, EFDR_M = 0.05, EFDR_D = 0.05, GenesSelect = NULL, ...)
```

Arguments

`Data`	an object of class `BASiCS_D_Data-class` (class no longer in use)
`object`	an object of class `BASiCS_D_Data-class` (class no longer in use)
`GeneNames`	Vector containing gene names to be used in results table (argument to be removed as 'GeneNames' will be an slot of 'BASiCS_D_Data' object)
`EpsilonM`	Minimum fold change tolerance threshold for detecting changes in overall expression (must be a positive real number)
`EpsilonD`	Minimum fold change tolerance threshold for detecting changes in cell-to-cell biological over dispersion (must be a positive real number)
`EviThresholdM`	Optional parameter. Evidence threshold for detecting changes in overall expression (must be a positive value, between 0 and 1)
`EviThresholdD`	Optional parameter. Evidence threshold for detecting changes in cell-to-cell biological over dispersion (must be a positive value, between 0 and 1)
`OrderVariable`	Ordering variable for output. Must take values in `c("GeneIndex", "GeneNames", "ProbDiffExp",`
`GroupLabelRef`	Label assigned to reference group. Default: `GroupLabelRef = "Ref"`
`GroupLabelTest`	Label assigned to reference group. Default: `GroupLabelRef = "Test"`
`Plot`	If `Plot = T`, error rates control rates and volcano plots are generated.
`OffSet`	Optional argument to remove a fix offset effect (if not previously removed from the MCMC chains). This argument will be removed shorly, once offset removal is built as an internal step.

`EFDR_M`	Target for expected false discovery rate related to the comparison of means (default = 0.05)
`EFDR_D`	Target for expected false discovery rate related to the comparison of dispersions (default = 0.05)
`GenesSelect`	Optional argument to provide a user-defined list of genes to be considered for the comparison (default = NULL). When used, this argument must be a vector of 'TRUE' (include gene) / 'FALSE' (exclude gene) indicator, with the same length as the number of intrinsic genes and following the same order as how genes are displayed in the table of counts. This argument is necessary in order to have a meaningful EFDR calibration when the user decides to exclude some genes from the comparison.
`...`	Graphical parameters (see `par`).

Details

This function is no longer in use and will be removed in future releases.

Value

This function is no longer in use and will be removed in future releases. Please use `BASiCS_TestDE` instead.

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

Examples

```
# This function is no longer in use and will be removed in future releases.
```

`BASiCS_Filter` *Filter for input datasets*

Description

`BASiCS_Filter` indicates which transcripts and cells pass a pre-defined inclusion criteria. The output of this function can be combined with `newBASiCS_Data` to generate a the `SummarizedExperiment` object required to run BASiCS.

Usage

```
BASiCS_Filter(Counts, Tech, SpikeInput, BatchInfo = NULL,
  MinTotalCountsPerCell = 2, MinTotalCountsPerGene = 2,
  MinCellsWithExpression = 2, MinAvCountsPerCellsWithExpression = 2)
```

Arguments

`Counts`	Matrix of dimensions `q` times `n` whose elements corresponds to the simulated expression counts. First `q.bio` rows correspond to biological genes. Last `q-q.bio` rows correspond to technical spike-in genes.
`Tech`	Logical vector of length `q`. If `Tech = F` the gene is biological; otherwise the gene is spike-in.
`SpikeInput`	Vector of length `q-q.bio` whose elements indicate the simulated input concentrations for the spike-in genes.
`BatchInfo`	Vector of length `n` whose elements indicate batch information. Not required if a single batch is present on the data. Default value: `BatchInfo = NULL`.
`MinTotalCountsPerCell`	Minimum value of total expression counts required per cell (biological and technical). Default value: `MinTotalCountsPerCell = 2`.
`MinTotalCountsPerGene`	Minimum value of total expression counts required per transcript (biological and technical). Default value: `MinTotalCountsPerGene = 2`.
`MinCellsWithExpression`	Minimum number of cells where expression must be detected (positive count). Criteria applied to each transcript. Default value: `MinCellsWithExpression = 2`.
`MinAvCountsPerCellsWithExpression`	Minimum average number of counts per cells where expression is detected. Criteria applied to each transcript. Default value: `MinAvCountsPerCellsWithExpression = 2`.

Value

A list of 2 elements

`Counts` Filtered matrix of expression counts

`Tech` Filtered vector of spike-in indicators

`SpikeInput` Filtered vector of spike-in genes input molecules

`BatchInfo` Filtered vector of the 'BatchInfo' argument

`IncludeGenes` Inclusion indicators for transcripts

`IncludeCells` Inclusion indicators for cells

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data. PLoS Computational Biology.

Vallejos, Marioni and Richardson (2016). Beyond comparisons of means: understanding changes in gene expression at the single-cell level. Genome Biology.

Examples

```
set.seed(1)
Counts = Counts = matrix(rpois(50*10, 2), ncol = 10)
rownames(Counts) <- c(paste0("Gene", 1:40), paste0("Spike", 1:10))
Tech = c(rep(FALSE,40),rep(TRUE,10))
set.seed(2)
SpikeInput = rgamma(10,1,1)
SpikeInfo <- data.frame("SpikeID" = paste0("Spike", 1:10), "SpikeInput" = SpikeInput)

Filter = BASiCS_Filter(Counts, Tech, SpikeInput,
                       MinTotalCountsPerCell = 2, MinTotalCountsPerGene = 2,
                      MinCellsWithExpression = 2, MinAvCountsPerCellsWithExpression = 2)
SpikeInfoFilter = SpikeInfo[SpikeInfo$SpikeID %in%
         names(Filter$IncludeGenes)[Filter$IncludeGenes == TRUE],]
FilterData = newBASiCS_Data(Filter$Counts, Filter$Tech, SpikeInfoFilter)
```

`BASiCS_LoadChain` *Loads pre-computed MCMC chains generated by the* `BASiCS_MCMC` *function*

Description

Loads pre-computed MCMC chains generated by the `BASiCS_MCMC` function, creating a `BASiCS_Chain-class` object

Usage

```
BASiCS_LoadChain(RunName, StoreDir = getwd(), WithSpikes = TRUE)
```

Arguments

`RunName`	String used to index '.Rds' files storing the MCMC chains (produced by the `BASiCS_MCMC` function)
`StoreDir`	Directory where '.Rds' files is stored. Default: `StoreDir = getwd()`
`WithSpikes`	`TRUE/FALSE` indicator of spike-ins usage. Default: `WithSpikes = TRUE`

Value

An object of class `BASiCS_Chain-class`.

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

See Also

`BASiCS_Chain-class`

Examples

```
Data = makeExampleBASiCS_Data()
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 5, Burn = 5,
                           StoreChains = TRUE, StoreDir = tempdir(),
                           RunName = "Test")
ChainLoad <- BASiCS_LoadChain(RunName = "Test", StoreDir = tempdir())
```

`BASiCS_MCMC` *BASiCS MCMC sampler*

Description

MCMC sampler to perform Bayesian inference for single-cell mRNA sequencing datasets using the model described in Vallejos et al (2015).

Usage

```
BASiCS_MCMC(Data, N, Thin, Burn, ...)
```

Arguments

`Data`	A `SummarizedExperiment` object. This MUST be formatted to include the spike-ins information. See vige
`N`	Total number of iterations for the MCMC sampler. Use `N>=max(4,Thin)`, `N` being a multiple of `Thin`.
`Thin`	Thining period for the MCMC sampler. Use `Thin>=2`.
`Burn`	Burn-in period for the MCMC sampler. Use `Burn>=1`, `Burn<N`, `Burn` being a multiple of `Thin`.
`...`	Optional parameters.

- `PriorDelta` Specifies the prior used for `delta`. Possible values are 'gamma' (Gamma(`a.theta`,`b.theta`) prior) and 'log-normal' (log-Normal(`0`,`s2.delta`) prior) .. Default value: `PriorDelta = 'log-normal'`.
- `PriorParam` List of 7 elements, containing the hyper-parameter values required for the adopted prior (see Vallejos et al, 2015). All elements must be positive real numbers.
  - `s2.mu` Scale hyper-parameter for the log-Normal(`0`,`s2.mu`) prior that is shared by all gene-specific expression rate parameters $\mu[i]$. Default: `s2.mu = 0.5`.
  - `s2.delta` Only used when 'PriorDelta == 'log-normal''. Scale hyper-parameter for the log-Normal(`0`,`s2.delta`) prior that is shared by all gene-specific expression rate parameters $\delta[i]$. Default: `s2.delta = 0.5`.
  - `a.delta` Only used when 'PriorDelta == 'gamma''. Shape hyper-parameter for the Gamma(`a.delta`,`b.delta`) prior that is shared by all gene-specific biological cell-to-cell heterogeneity hyper-parameters $\delta[i]$. Default: `a.delta = 1`.
  - `b.delta` Only used when 'PriorDelta == 'gamma''. Rate hyper-parameter for the Gamma(`a.delta`,`b.delta`) prior that is shared by all gene-specific biological cell-to-cell heterogeneity hyper-parameters $\delta[i]$. Default: `b.delta = 1`.

- p.phi Dirichlet hyper-parameter for the joint of all (scaled by n) cell-specific mRNA content normalising constants $\phi[j]/n$. Default: p.phi = rep(1, n).
- a.s Shape hyper-parameter for the Gamma(a.s,b.s) prior that is shared by all cell-specific capture efficiency normalising constants $s[j]$. Default: a.s = 1.
- b.s Rate hyper-parameter for the Gamma(a.s,b.s) prior that is shared by all cell-specific capture efficiency normalising constants $s[j]$. Default: b.s = 1.
- a.theta Shape hyper-parameter for the Gamma(a.theta,b.theta) prior for technical noise hyper-parameter θ. Default: a.theta = 1.
- b.theta Rate hyper-parameter for the Gamma(a.theta,b.theta) prior for technical noise hyper-parameter θ. Default: b.theta = 1.

AR Optimal acceptance rate for adaptive Metropolis Hastings updates. It must be a positive number between 0 and 1. Default (and recommended): AR = 0.44.

StopAdapt Iteration at which adaptive proposals are not longer adapted. Use StopAdapt>=1. Default: StopAdapt = Burn.

StoreChains If StoreChains = TRUE, the slots of the generated BASiCS_Chain object are stored in separate .txt files. Each row of the output file containing an interation (RunName argument used to index file names). Default: StoreChains = FALSE.

StoreAdapt If StoreAdapt = TRUE, trajectory of adaptive proposal variances (in log-scale) for each parameter are stored in separate .txt files. Each row of the output file containing an interation (RunName argument used to index file names). Default: StoreAdapt = FALSE.

StoreDir Directory where output files are stored. Only required if StoreChains = TRUE and/or StoreAdapt = TRUE). Default: StoreDir = getwd().

RunName String used to index '.txt' files storing chains and/or adaptive proposal variances.

PrintProgress If PrintProgress = FALSE, console-based progress report is suppressed.

ls.phi0 Starting value for the adaptive concentration parameter of the Metropolis proposals for phi.

Start In general, we do not advise to specify this argument. Default options have been tuned to facilitate convergence. It can be used to set user defined starting points for the MCMC algorithm. If used, it must be a list containing the following elements: mu0, delta0, phi0, s0, nu0, theta0, ls.mu0, ls.delta0, ls.phi0, ls.nu0, ls.theta0

Value

An object of class BASiCS_Chain-class.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl> and Nils Eling

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data. PLoS Computational Biology.

Vallejos, Marioni and Richardson (2016). Beyond comparisons of means: understanding changes in gene expression at the single-cell level. Genome Biology.

Examples

```
# Built-in simulated dataset
Data = makeExampleBASiCS_Data()
# To analyse real data, please refer to the instructions in:
# https://github.com/catavallejos/BASiCS/wiki/2.-Input-preparation

# Only a short run of the MCMC algorithm for illustration purposes
# Longer runs migth be required to reach convergence
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 10, PrintProgress = FALSE)

# For illustration purposes we load a built-in 'BASiCS_Chain' object
# (obtained using the 'BASiCS_MCMC' function)
data(ChainSC)

# `displayChainBASiCS` can be used to extract information from this output. For example:
head(displayChainBASiCS(ChainSC, Param = "mu"))

# Traceplot (examples only)
plot(ChainSC, Param = "mu", Gene = 1)
plot(ChainSC, Param = "phi", Cell = 1)
plot(ChainSC, Param = "theta", Batch = 1)

# Calculating posterior medians and 95% HPD intervals
ChainSummary <- Summary(ChainSC)

# `displaySummaryBASiCS` can be used to extract information from this output. For example:
head(displaySummaryBASiCS(ChainSummary, Param = "mu"))

# Graphical display of posterior medians and 95% HPD intervals (examples only)
plot(ChainSummary, Param = "mu", main = "All genes")
plot(ChainSummary, Param = "mu", Genes = 1:10, main = "First 10 genes")
plot(ChainSummary, Param = "phi", main = "All cells")
plot(ChainSummary, Param = "phi", Cells = 1:5, main = "First 5 cells")
plot(ChainSummary, Param = "theta")

# To constrast posterior medians of cell-specific parameters (example only)
par(mfrow = c(1,2))
plot(ChainSummary, Param = "phi", Param2 = "s", SmoothPlot = FALSE)
# Recommended for large numbers of cells
plot(ChainSummary, Param = "phi", Param2 = "s", SmoothPlot = TRUE)

# To constrast posterior medians of gene-specific parameters
par(mfrow = c(1,2))
plot(ChainSummary, Param = "mu", Param2 = "delta", log = "x", SmoothPlot = FALSE)
# Recommended
plot(ChainSummary, Param = "mu", Param2 = "delta", log = "x", SmoothPlot = TRUE)

# Highly and lowly variable genes detection (within a single group of cells)
DetectHVG <- BASiCS_DetectHVG(ChainSC, VarThreshold = 0.60, EFDR = 0.10, Plot = TRUE)
DetectLVG <- BASiCS_DetectLVG(ChainSC, VarThreshold = 0.40, EFDR = 0.10, Plot = TRUE)

plot(ChainSummary, Param = "mu", Param2 = "delta", log = "x", col = 8)
with(DetectHVG$Table, points(Mu[HVG == TRUE], Delta[HVG == TRUE],
       pch = 16, col = "red", cex = 1))
with(DetectLVG$Table, points(Mu[LVG == TRUE], Delta[LVG == TRUE],
        pch = 16, col = "blue", cex = 1))

# If variance thresholds are not fixed
BASiCS_VarThresholdSearchHVG(ChainSC, VarThresholdsGrid = seq(0.55,0.65,by=0.01), EFDR = 0.10)
BASiCS_VarThresholdSearchLVG(ChainSC, VarThresholdsGrid = seq(0.35,0.45,by=0.01), EFDR = 0.10)

# For examples of differential analyses between 2 populations of cells:
?BASiCS_TestDE
```

`BASiCS_Sim` *Simulates expression counts according to the model implemented in BASiCS*

Description

`BASiCS_Sim` creates a simulated dataset from the model implemented in BASiCS. This function is used in order to illustrate the performance of the `BASiCS` library.

Usage

```
BASiCS_Sim(mu, mu_spikes, delta, phi, s, theta)
```

Arguments

`mu`	Gene-specific expression levels $\mu[i]$ for all biological genes (vector of length `q.bio`, all elements must be positive numbers)
`mu_spikes`	$\mu[i]$ for all technical genes defined as true input molecules (SpikeInfo) (vector of length `q-q.bio`, all elements must be positive numbers)
`delta`	Gene-specific biological cell-to-cell heterogeneity hyper-parameters $\delta[i]$, biological genes only (vector of length `q.bio`, all elements must be positive numbers)
`phi`	Cell-specific mRNA content normalising constants $\phi[j]$ (vector of length `n`, all elements must be positive numbers and the sum of its elements must be equal to `n`)
`s`	Cell-specific capture efficiency (or amplification biases if not using UMI based counts) normalising constants $s[j]$ (vector of length `n`, all elements must be positive numbers)
`theta`	Technical variability hyper-parameter θ (must be positive)

Value

An object of class `SummarizedExperiment`, simulated from the model implemented in BASiCS.

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

Examples

```
# Simulated parameter values for 10 genes
# (7 biogical and 3 spike-in) measured in 5 cells
Mu =  c(8.36, 10.65, 4.88, 6.29, 21.72, 12.93, 30.19)
Mu_spike =  c(1010.72, 7.90, 31.59)
Delta = c(1.29, 0.88, 1.51, 1.49, 0.54, 0.40, 0.85)
Phi = c(1.00, 1.06, 1.09, 1.05, 0.80)
S = c(0.38, 0.40, 0.38, 0.39, 0.34)
Theta = 0.39

Data = BASiCS_Sim(Mu, Mu_spike, Delta, Phi, S, Theta)
head(assay(Data))
dim(assay(Data))
metadata(Data)$SpikeInput
rowData(Data)$Tech
```

`BASiCS_Summary-class` *The BASiCS_Summary class*

Description

Container of a summary of a `BASiCS_Chain-class` object. In each slot, first column contains posterior medians, second column contains the lower limits of an high posterior density interval and third column contains the upper limits of high posterior density intervals.

Slots

- `mu` Posterior medians (first column), lower (second column) and upper (third column) limits of gene-specific expression levels $\mu[i]$.
- `delta` Posterior medians (first column), lower (second column) and upper (third column) limits of gene-specific biological cell-to-cell heterogeneity hyper-parameters $\delta[i]$, biological genes only
- `phi` Posterior medians (first column), lower (second column) and upper (third column) limits of cell-specific mRNA content normalising constants $\phi[j]$
- `s` Posterior medians (first column), lower (second column) and upper (third column) limits of cell-specific capture efficiency (or amplification biases if not using UMI based counts) normalising constants $s[j]$
- `nu` Posterior medians (first column), lower (second column) and upper (third column) limits of cell-specific random effects $\nu[j]$
- `theta` Posterior median (first column), lower (second column) and upper (third column) limits of technical variability hyper-parameter θ (each row represents one batch)

Examples

```
# A BASiCS_Summary object created by the Summary method.
Data = makeExampleBASiCS_Data()
MCMC_Output <- BASiCS_MCMC(Data, N = 100, Thin = 2, Burn = 2)
MCMC_Summary <- Summary(MCMC_Output)
```

BASiCS_Summary-methods *'show' method for BASiCS_Summary objects*

Description

'show' method for BASiCS_Summary-class objects.

Usage

```
## S4 method for signature 'BASiCS_Summary'
show(object)
```

Arguments

object	A BASiCS_Summary object.

Value

Prints a summary of the properties of object.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

Examples

```
Data = makeExampleBASiCS_Data()
MCMC_Output <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 2)
Summary(MCMC_Output)
```

BASiCS_TestDE *Detection of genes with changes in expression*

Description

Function to assess changes in expression (mean and over-dispersion)

Usage

```
BASiCS_TestDE(Chain1, Chain2, EpsilonM = log2(1.5), EpsilonD = log2(1.5),
  EviThresholdM = NULL, EviThresholdD = NULL, OrderVariable = "Prob",
  GroupLabel1 = "Group1", GroupLabel2 = "Group2", Plot = TRUE,
  PlotOffset = TRUE, OffSet = TRUE, EFDR_M = 0.05, EFDR_D = 0.05,
  GenesSelect = NULL, ...)
```

Arguments

`Chain1`	an object of class `BASiCS_Chain-class` containing parameter estimates for the first group of cells
`Chain2`	an object of class `BASiCS_Chain-class` containing parameter estimates for the second group of cells
`EpsilonM`	Minimum fold change tolerance threshold for detecting changes in overall expression (must be a positive real number). Default value: `EpsilonM = log2(1.5)` (i.e. 50% increase).
`EpsilonD`	Minimum fold change tolerance threshold for detecting changes in cell-to-cell biological over dispersion (must be a positive real number). Default value: `EpsilonM = log2(1.5)` (i.e. 50% increase).
`EviThresholdM`	Optional parameter. Evidence threshold for detecting changes in overall expression (must be a positive value, between 0 and 1)
`EviThresholdD`	Optional parameter. Evidence threshold for detecting changes in cell-to-cell biological over dispersion (must be a positive value, between 0 and 1)
`OrderVariable`	Ordering variable for output. Must take values in `c("GeneIndex", "GeneNames", "Prob")`.
`GroupLabel1`	Label assigned to reference group. Default: `GroupLabel1 = "Group1"`
`GroupLabel2`	Label assigned to reference group. Default: `GroupLabel2 = "Group2"`
`Plot`	If `Plot = TRUE`, MA and volcano plots are generated.
`PlotOffset`	If `Plot = TRUE`, the offset effect is visualised.
`OffSet`	Optional argument to remove a fix offset effect (if not previously removed from the MCMC chains). This argument will be removed shorly, once offset removal is built as an internal step.
`EFDR_M`	Target for expected false discovery rate related to the comparison of means (default = 0.05)
`EFDR_D`	Target for expected false discovery rate related to the comparison of dispersions (default = 0.05)
`GenesSelect`	Optional argument to provide a user-defined list of genes to be considered for the comparison (default = NULL). When used, this argument must be a vector of 'TRUE' (include gene) / 'FALSE' (exclude gene) indicator, with the same length as the number of intrinsic genes and following the same order as how genes are displayed in the table of counts. This argument is necessary in order to have a meaningful EFDR calibration when the user decides to exclude some genes from the comparison.
`...`	Graphical parameters (see `par`).

Value

`BASiCS_TestDE` returns a list of 4 elements:

`TableMean` A `data.frame` containing the results of the differential mean test

- `GeneNames` Gene name
- `MeanOverall` For each gene, the estimated mean expression parameter $\mu[i]$ is averaged across both groups of cells (weighted by sample size).
- `Mean1` Estimated mean expression parameter $\mu[i]$ for each biological gene in the first group of cells.
- `Mean2` Estimated mean expression parameter $\mu[i]$ for each biological gene in the second group of cells.

- `MeanFC` Fold change in mean expression parameters between the first and second groups of cells.
- `MeanLog2FC` Log2-transformed fold change in mean expression between the first and second groups of cells.
- `ProbDiffMean` Posterior probability for mean expression difference between the first and second groups of cells.
- `ResultDiffExp` Indicator if a gene has a higher mean expression in the first or second groups of cells.

`TableDisp` A `data.frame` containing the results of the differential dispersion test (excludes genes for which the mean changes).

- `GeneNames` Gene name
- `MeanOverall` For each gene, the estimated mean expression parameter $\mu[i]$ is averaged across both groups of cells (weighted by sample size).
- `DispOverall` For each gene, the estimated over-dispersion parameter $\delta[i]$ is averaged across both groups of cells (weighted by sample size).
- `Disp1` Estimated over-dispersion parameter $\delta[i]$ for each biological gene in the first group of cells.
- `Disp2` Estimated over-dispersion parameter $\delta[i]$ for each biological gene in the second group of cells.
- `DispFC` Fold change in over-dispersion parameters between the between the first and second groups of cells.
- `DispLog2FC` Log-transformed fold change in over-dispersion between the first and second groups of cells.
- `ProbDiffDisp` Posterior probability for over-dispersion difference between the first and second groups of cells.
- `ResultDiffDisp` Indicator if a gene has a higher over-dispersion in the first or second groups of cells.

`DiffExpSummary` A list containing the following information for the differential mean expression test:

- `EviThreshold` Evidence thresholds.
- `EFDR` Expected false discovery rate for the given thresholds.
- `EFNR` Expected false negative rate for the given thresholds.

`DiffOverDispSummary` A list containing the following information for the differential over-dispersion test:

- `EviThreshold` Evidence thresholds.
- `EFDR` Expected false discovery rate for the given thresholds.
- `EFNR` Expected false negative rate for the given thresholds.

`Chain1_offset` an `BASiCS_Chain-class` object: `Chain1` after offset removal.

`Chain2_offset` an `BASiCS_Chain-class` object: `Chain2` after offset removal (this is only provided for completeness; `Chain2` is not affected by the offset).

`OffsetChain` MCMC chain calculated for the offset effect.

`Offset` Estimated offset (posterior median of `OffsetChain`). Default value set equal to 1 when offset correction is not performed.

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`> and Nils Eling

Examples

```
# Loading two 'BASiCS_Chain' objects (obtained using the 'BASiCS_MCMC' function)
data(ChainSC)
data(ChainRNA)

Test <- BASiCS_TestDE(Chain1 = ChainSC, Chain2 = ChainRNA,
                      GroupLabel1 = "SC", GroupLabel2 = "P&S",
                      EpsilonM = log2(1.5), EpsilonD = log2(1.5), OffSet = TRUE)

# Results for the differential mean test
head(Test$TableMean)

# Results for the differential over-dispersion test
# This only includes genes marked as 'NoDiff' in Test$TableMean
head(Test$TableDisp)
```

BASiCS_VarianceDecomp *Decomposition of gene expression variability according to BASiCS*

Description

Function to decompose total variability of gene expression into biological and technical components.

Usage

```
BASiCS_VarianceDecomp(Chain, OrderVariable = "BioVarGlobal", Plot = TRUE,
  ...)
```

Arguments

`Chain`	an object of class `BASiCS_Chain-class`
`OrderVariable`	Ordering variable for output. Must take values in `c("GeneNames", "BioVarGlobal", "TechVarGlo`
`Plot`	If `TRUE`, a barplot of the variance decomposition (global and by batches, if any) is generated
`...`	Other arguments to be passed to `barplot`

Details

See vignette

Value

A `data.frame` whose first 4 columns correspond to

`GeneName` Gene name (as indicated by user)

`BioVarGlobal` Percentage of variance explained by a biological cell-to-cell heterogeneity component (overall across all cells)

`TechVarGlobal` Percentage of variance explained by the technical cell-to-cell heterogeneity component (overall across all cells)

`ShotNoiseGlobal` Percentage of variance explained by the shot noise component (baseline, overall across all cells)

If more than 1 batch of cells are being analysed, the remaining columns contain the corresponding variance decomposition calculated within each batch.

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

See Also

`BASiCS_Chain-class`

Examples

```
# See
help(BASiCS_MCMC)
```

`BASiCS_VarThresholdSearchHVG`

Detection method for highly and lowly variable genes using a grid of variance contribution thresholds

---

Description

Detection method for highly and lowly variable genes using a grid of variance contribution thresholds

Usage

```
BASiCS_VarThresholdSearchHVG(Chain, VarThresholdsGrid, EFDR = 0.1,
  Progress = TRUE)

BASiCS_VarThresholdSearchLVG(Chain, VarThresholdsGrid, EFDR = 0.1,
  Progress = TRUE)
```

Arguments

`Chain`	an object of class `BASiCS_Chain-class`
`VarThresholdsGrid`	Grid of values for the variance contribution threshold (they must be contained in (0,1))
`EFDR`	Target for expected false discovery rate related to HVG/LVG detection (default = 0.10)
`Progress`	If `Progress = TRUE`, partial output is printed in the console.

Details

See vignette

Value

BASiCS_VarThresholdSearchHVG A table displaying the results of highly variable genes detecting for different variance contribution thresholds.

BASiCS_VarThresholdSearchLVG A table displaying the results of lowly variable genes detecting for different variance contribution thresholds.oo

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

See Also

BASiCS_Chain-class

Examples

```
# See
help(BASiCS_MCMC)
```

ChainRNA *Extract from the chain Vallejos et al (2016): pool-and-split samples*

Description

Small extract (100 MCMC iterations, 500 randomly selected genes) from the chain obtained in Vallejos et al (2016), related to pool-and-split samples (this corresponds to the RNA 2i samples in Grun et al, 2014).

Usage

```
ChainRNA
```

Format

BASiCS_Chain-class containing 10 MCMC iterations

References

Vallejos et al (2016) - Genome Biology Grun et al (2014) - Nature Methods

`ChainSC` *Extract from the chain Vallejos et al (2016): single-cell samples*

Description

Small extract (100 MCMC iterations, 500 randomly selected genes) from the chain obtained in Vallejos et al (2016), related to single-cell samples (this corresponds to the SC 2i samples in Grun et al, 2014).

Usage

```
ChainSC
```

Format

`BASiCS_Chain-class` containing 10 MCMC iterations

References

Vallejos et al (2016) - Genome Biology Grun et al (2014) - Nature Methods

`DenoisedCounts` *Calculates normalised and denoised expression expression counts*

Description

Calculates normalised and denoised expression counts, by removing the effect of technical variation.

Usage

```
BASiCS_DenoisedCounts(Data, Chain)
```

Arguments

`Data`	an object of class `SummarizedExperiment`
`Chain`	an object of class `BASiCS_Chain-class`

Details

See vignette

Value

A matrix of normalised and denoised expression counts.

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`>

See Also

BASiCS_Chain-class

Examples

```
# See
help(BASiCS_MCMC)
```

DenoisedRates — *Calculates normalised and denoised expression rates*

Description

Calculates normalised and denoised expression rates, by removing the effect of technical variation.

Usage

```
BASiCS_DenoisedRates(Data, Chain, PrintProgress = FALSE,
  Propensities = FALSE)
```

Arguments

`Data`	an object of class SummarizedExperiment
`Chain`	an object of class BASiCS_Chain-class
`PrintProgress`	If TRUE, partial progress information is printed in the console.
`Propensities`	If TRUE, returns underlying expression propensitites $\rho[ij]$. Otherwise, denoised rates $\mu[i]\rho[ij]$ are returned.

Details

See vignette

Value

A matrix of normalised and denoised expression counts.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

See Also

BASiCS_Chain-class

Examples

```
# See
help(BASiCS_MCMC)
```

displayChainBASiCS-BASiCS_Chain-method

Accessors for the slots of a BASiCS_Chain object

Description

Accessors for the slots of a `BASiCS_Chain-class`

Usage

```
## S4 method for signature 'BASiCS_Chain'
displayChainBASiCS(object, Param = "mu")
```

Arguments

`object`	an object of class `BASiCS_Chain-class`
`Param`	Name of the slot to be used for the accessed. Possible values: `mu, delta, phi, s, nu, theta`

Value

The requested slot of an object of class `BASiCS_Chain-class`

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

See Also

`BASiCS_Chain-class`

Examples

```
# See
help(BASiCS_MCMC)
```

displaySummaryBASiCS-BASiCS_Summary-method

Accessors for the slots of a BASiCS_Summary object

Description

Accessors for the slots of a `BASiCS_Summary-class`

Usage

```
## S4 method for signature 'BASiCS_Summary'
displaySummaryBASiCS(object, Param = "mu")
```

Arguments

`object`	an object of class `BASiCS_Summary-class`
`Param`	Name of the slot to be used for the accessed. Possible values: `mu, delta, phi, s, nu, theta`

Value

The requested slot of an object of class `BASiCS_Summary-class`

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

See Also

`BASiCS_Summary-class`

Examples

```
# See
help(BASiCS_MCMC)
```

`makeExampleBASiCS_Data` *Create a simple example of a SummarizedExperiment object with random data*

Description

A simple `SummarizedExperiment` object is generated by simulating a dataset from the model in BASiCS (Vallejos et al 2015). This is used to illustrate BASiCS in the package examples and in the vignette.

Usage

```
makeExampleBASiCS_Data(WithBatch = FALSE, WithSpikes = TRUE, Example = 1)
```

Arguments

`WithBatch`	If `TRUE`, 2 batches are generated (each of them containing 10 cells). Default value: `WithBatch = FALSE`.
`WithSpikes`	If `TRUE`, the simulated dataset contains 20 spike-in genes. Default value: `WithSpikes = TRUE`.
`Example`	Choice of example will be generated, defined by different parameter values (possible values = 1, 2). In both cases, expression counts associated to 50 biological genes are generated. This is used to generate two populations for differential testing. Default value `Example = 1`.

Value

An object of class `SummarizedExperiment`, simulated from the model implemented in BASiCS. If `WithSpikes = TRUE`, it contains 70 genes (50 biological and 20 spike-in) and 20 cells. Alternatively, it contains 50 biological genes and 20 cells.

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`> and Nils Eling

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data. PLoS Computational Biology.

Vallejos, Marioni and Richardson (2016). Beyond comparisons of means: understanding changes in gene expression at the single-cell level. Genome Biology.

Examples

```
Data = makeExampleBASiCS_Data()
is(Data, "SummarizedExperiment")
```

`newBASiCS_Chain` *Creates a BASiCS_Chain object from pre-computed MCMC chains*

Description

`BASiCS_Chain` creates a `BASiCS_Chain-class` object from pre-computed MCMC chains.

Usage

```
newBASiCS_Chain(mu, delta, phi, s, nu, theta)
```

Arguments

`mu`	MCMC chain for gene-specific expression levels $\mu[i]$, defined as true input molecules in case of technical genes (matrix with `q` columns, technical genes located at the end of the matrix, all elements must be positive numbers)
`delta`	MCMC chain for gene-specific biological cell-to-cell heterogeneity hyper-parameters $\delta[i]$, biological genes only (matrix with `q.bio` columns, all elements must be positive numbers)
`phi`	MCMC chain for cell-specific mRNA content normalising constants $\phi[j]$ (matrix with `n` columns, all elements must be positive numbers and the sum of its elements must be equal to `n`)
`s`	MCMC chain for cell-specific capture efficiency (or amplification biases if not using UMI based counts) normalising constants $s[j]$ (matrix with `n` columns, all elements must be positive numbers)
`nu`	MCMC chain for cell-specific random effects $\nu[j]$ (matrix with `n` columns, all elements must be positive numbers)
`theta`	MCMC chain for technical variability hyper-parameter θ (vector, all elements must be positive)

Value

An object of class `BASiCS_Chain-class`.

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

See Also

`BASiCS_Chain-class`

Examples

```
# Data = makeExampleBASiCS_Data()
# MCMC_Output <- BASiCS_MCMC(Data, N = 50, Thin = 5, Burn = 5,
#                 StoreChains = TRUE, StoreDir = getwd(), RunName = "Test")

# ChainMu = as.matrix(read.table("chain_mu_Test.txt"))
# ChainDelta = as.matrix(read.table("chain_delta_Test.txt"))
# ChainPhi = as.matrix(read.table("chain_phi_Test.txt"))
# ChainS = as.matrix(read.table("chain_s_Test.txt"))
# ChainNu = as.matrix(read.table("chain_nu_Test.txt"))#
# ChainTheta = read.table("chain_theta_Test.txt")[,1]

# MCMC_Output_Load <- newBASiCS_Chain(mu = ChainMu, delta = ChainDelta,
#   phi = ChainPhi, s = ChainS, nu = ChainNu, theta = ChainTheta)
```

`newBASiCS_Data` *Creates a SummarizedExperiment object from a matrix of expression counts and experimental information about spike-in genes*

Description

`newBASiCS_Data` creates a `SummarizedExperiment` object from a matrix of expression counts and experimental information about spike-in genes.

Usage

```
newBASiCS_Data(Counts, Tech, SpikeInfo, BatchInfo = NULL)
```

Arguments

`Counts`	Matrix of dimensions `q` times `n` whose elements contain the expression counts to be analyses (including biological and technical spike-in genes). Gene names must be stored as ‘rownames(Counts)‘.
`Tech`	Logical vector of length `q`. If `Tech = FALSE` the gene is biological; otherwise the gene is spike-in.
`SpikeInfo`	`data.frame` whose first and second columns contain the gene names assigned to the spike-in genes (they must match the ones in ‘rownames(Counts)‘) and the associated input number of molecules, respectively.
`BatchInfo`	Vector of length `n` whose elements indicate batch information. Not required if a single batch is present on the data. Default value: `BatchInfo = NULL`.

Value

An object of class `SummarizedExperiment`.

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`> and Nils Eling

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data. PLoS Computational Biology.

Vallejos, Marioni and Richardson (2016). Beyond comparisons of means: understanding changes in gene expression at the single-cell level. Genome Biology.

See Also

SummarizedExperiment

Examples

```
# Expression counts
set.seed(1)
Counts = Counts = matrix(rpois(50*10, 2), ncol = 10)
rownames(Counts) <- c(paste0("Gene", 1:40), paste0("Spike", 1:10))

# Technical information
Tech = c(rep(FALSE,40),rep(TRUE,10))

# Spikes input number of molecules
set.seed(2)
SpikeInfo <- data.frame(gene=rownames(Counts)[Tech],amount=rgamma(10,1,1))

# Creating a BASiCS_Data object (no batch effect)
DataExample = newBASiCS_Data(Counts, Tech, SpikeInfo)

# Creating a BASiCS_Data object (with batch effect)
BatchInfo = c(rep(1, 5), rep(2, 5))
DataExample = newBASiCS_Data(Counts, Tech, SpikeInfo, BatchInfo)


# Thanks to Simon Andrews for reporting an issue in previous version of this documentation
```

---

plot-BASiCS_Chain-method *'plot' method for BASiCS_Chain objects*

Description

'plot' method for BASiCS_Chain objects

Usage

```
## S4 method for signature 'BASiCS_Chain,ANY'
plot(x, Param = "mu", Gene = NULL,
  Cell = NULL, Batch = 1, ylab = "", xlab = "", ...)
```

Arguments

`x`	A `BASiCS_Chain` object.
`Param`	Name of the slot to be used for the plot. Possible values: `mu, delta, phi, s, nu, theta`
`Gene`	Specifies which gene is requested. Required only if `Param = "mu"` or `"delta"`
`Cell`	Specifies which cell is requested. Required only if `Param = "phi", "s"` or `"nu"`
`Batch`	Specifies which batch is requested. Required only if `Param = "theta"`
`ylab`	As in `par`.
`xlab`	As in `par`.
`...`	Other graphical parameters (see `par`).

Author(s)

Catalina A. Vallejos <`cnvallej@uc.cl`>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

Examples

```
# See
help(BASiCS_MCMC)
```

`plot-BASiCS_Summary-method`
'plot' method for BASiCS_Summary objects

Description

'plot' method for BASiCS_Summary objects

Usage

```
## S4 method for signature 'BASiCS_Summary,ANY'
plot(x, Param = "mu", Param2 = NULL,
  Genes = NULL, Cells = NULL, Batches = NULL, xlab = "", ylab = "",
  xlim = "", ylim = "", pch = 16, col = "blue", bty = "n",
  SmoothPlot = TRUE, ...)
```

Arguments

`x`	A `BASiCS_Summary` object.
`Param`	Name of the slot to be used for the plot. Possible values: `mu, delta, phi, s, nu, theta`
`Param2`	Name of the second slot to be used for the plot. Possible values: `mu, delta, phi, s, nu` (combinations between gene-specific and cell-specific parameters are not admitted)
`Genes`	Specifies which genes are requested. Required only if `Param = "mu"` or `"delta"`
`Cells`	Specifies which cells are requested. Required only if `Param = "phi", "s"` or `"nu"`
`Batches`	Specifies which batches are requested. Required only if `Param = "theta"`
`xlab`	As in `par`.
`ylab`	As in `par`.
`xlim`	As in `par`.
`ylim`	As in `par`.
`pch`	As in `par`.
`col`	As in `par`.
`bty`	As in `par`.
`SmoothPlot`	Logical parameter. If `TRUE`, transparency will be added to the color of the dots.
`...`	Other graphical parameters (see `par`).

Author(s)

Catalina A. Vallejos `<cnvallej@uc.cl>`

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

Examples

```
# See
help(BASiCS_MCMC)
```

`Summary` *'Summary' method for BASiCS_Chain objects*

Description

For each of the BASiCS parameters (see Vallejos et al 2015), `Summary` returns the corresponding postior medians and limits of the high posterior density interval with probabilty equal to `prob`.

Usage

```
## S4 method for signature 'BASiCS_Chain'
Summary(x, prob = 0.95)
```

Arguments

x	A BASiCS_Chain object.
prob	prob argument for HPDinterval function.

Value

An object of class BASiCS_Summary-class.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

References

Vallejos, Marioni and Richardson (2015). Bayesian Analysis of Single-Cell Sequencing data.

Examples

```
Data = makeExampleBASiCS_Data()
MCMC_Output <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 2)
MCMC_Summary <- Summary(MCMC_Output)

# See documentation of function BASiCS_MCMC
```

Index