Package 'BASiCS'

April 15, 2020

Type Package

Title Bayesian Analysis of Single-Cell Sequencing data

Version 1.8.1

Date 2020-02-10

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 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 can be quantified via a biological over-dispersion parameter that measures the excess of variability that is observed with respect to Poisson sampling noise, after normalisation and technical noise removal. Due to the strong mean/over-dispersion confounding that is typically observed for scRNA-seq datasets, BASiCS also tests for changes in residual over-dispersion, defined by residual values with respect to a global mean/over-dispersion trend.

License GPL (>= 2)

Depends R (>= 3.6), SingleCellExperiment

Imports Biobase, BiocGenerics, coda, cowplot, data.table, ggExtra, ggplot2, graphics, grDevices, KernSmooth, MASS, matrixStats, methods, Rcpp (>= 0.11.3), S4Vectors, scran, stats, stats4, SummarizedExperiment, viridis, utils, Matrix

Suggests BiocStyle, knitr, rmarkdown, testthat

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

biocViews ImmunoOncology, Normalization, Sequencing, RNASeq, Software, GeneExpression, Transcriptomics, SingleCell, DifferentialExpression, Bayesian, CellBiology, ImmunoOncology

SystemRequirements C++11

NeedsCompilation yes

URL https://github.com/catavallejos/BASiCS

BugReports https://github.com/catavallejos/BASiCS/issues

RoxygenNote 7.0.2

Encoding UTF-8

git_url https://git.bioconductor.org/packages/BASiCS

git_branch RELEASE_3_10

git_last_commit 42dc339

git_last_commit_date 2020-02-10

Date/Publication 2020-04-14

Author Catalina Vallejos [aut, cre],
Nils Eling [aut],
Alan O'Callaghan [aut],
Sylvia Richardson [ctb],
John Marioni [ctb]

Maintainer Catalina Vallejos <catalina.vallejos@igmm.ed.ac.uk>

R topics documented:

`BASiCS-defunct` *Defunct functions in package 'BASiCS'*

Description

The functions listed here are no longer part of `BASiCS`.

Details

Removed

- `BASiCS_D_TestDE` has been replaced by `BASiCS_TestDE`.

usage

Removed

- BASiCS_D_TestDE()

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

See Also

- `BASiCS_TestDE`

BASiCS_Chain *The BASiCS_Chain class*

Description

Container of an MCMC sample of the BASiCS' model parameters as generated by the function BASiCS_MCMC.

Slots

`parameters` List of matrices containing MCMC chains for each model parameter. Depending on the mode in which BASiCS was run, the following parameters can appear in the list:

- `mu` MCMC chain for gene-specific mean expression parameters μ_i, biological genes only (matrix with `q.bio` columns, all elements must be positive numbers)
- **delta** MCMC chain for gene-specific biological over-dispersion parameters δ_i, biological genes only (matrix with `q.bio` columns, all elements must be positive numbers)
- **phi** MCMC chain for cell-specific mRNA content normalisation parameters ϕ_j (matrix with `n` columns, all elements must be positive numbers and the sum of its elements must be equal to `n`) This parameter is only used when spike-in genes are available.
- **s** MCMC chain for cell-specific technical normalisation parameters s_j (matrix with `n` columns, all elements must be positive numbers)
- **nu** MCMC chain for cell-specific random effects ν_j (matrix with `n` columns, all elements must be positive numbers)
- **theta** MCMC chain for technical over-dispersion parameter(s) θ (matrix, all elements must be positive, each colum represents 1 batch)
- `beta` Only relevant for regression BASiCS model (Eling et al, 2017). MCMC chain for regression coefficients (matrix with `k` columns, where `k` represent the number of chosen basis functions + 2)
- `sigma2` Only relevant for regression BASiCS model (Eling et al, 2017). MCMC chain for the residual variance (matrix with one column, sigma2 represents a global parameter)
- `epsilon` Only relevant for regression BASiCS model (Eling et al, 2017). MCMC chain for the gene-specific residual over-dispersion parameter (matrix with `q` columns)
- `RefFreq` Only relevant for no-spikes BASiCS model (Eling et al, 2017). For each biological gene, this vector displays the proportion of times for which each gene was used as a reference (within the MCMC algorithm), when using the stochastic reference choice described in (Eling et al, 2017). This information has been kept as it is useful for the developers of this library. However, we do not expect users to need it.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Examples

```
# A BASiCS_Chain object created by the BASiCS_MCMC function.
Data <- makeExampleBASiCS_Data()
```

```
# To run the model without regression
Chain <- BASiCS_MCMC(Data, N = 100, Thin = 2, Burn = 2, Regression = FALSE)

# To run the model using the regression model
ChainReg <- BASiCS_MCMC(Data, N = 100, Thin = 2, Burn = 2, Regression = TRUE)
```

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

Description

'show' method for BASiCS_Chain objects.

'updateObject' method for BASiCS_Chain objects. It is used to convert outdated BASiCS_Chain objects into a version that is compatible with the Bioconductor release of BASiCS. Do not use this method is BASiCS_Chain already contains a parameters slot.

Usage

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

## S4 method for signature 'BASiCS_Chain'
updateObject(object, ..., verbose = FALSE)
```

Arguments

object	A BASiCS_Chain object.
...	Additional arguments of updateObject generic method. Not used within BASiCS.
verbose	Additional argument of updateObject generic method. Not used within BASiCS.

Value

Prints a summary of the properties of object.

Returns an updated BASiCS_Chain object that contains all model parameters in a single slot object (list).

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Examples

```
Data <- makeExampleBASiCS_Data()
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 2, Regression = FALSE)


# Not run
# New_Chain <- updateObject(Old_Chain)
```

BASiCS_CorrectOffset *Remove global mean expression offset*

Description

Remove global offset in mean expression between two BASiCS_Chain objects.

Usage

```
BASiCS_CorrectOffset(Chain, ChainRef, min.mean = 1)
```

Arguments

Chain	a 'BASiCS_MCMC' object to which the offset correction should be applied (with respect to 'ChainRef').
ChainRef	a 'BASiCS_MCMC' object to be used as the reference in the offset correction procedure.
min.mean	Minimum mean expression threshold required for inclusion in offset calculation. Similar to 'min.mean' in 'scran::computeSumFactors'.

Value

A list whose first element is an offset corrected version of 'Chain' (using 'ChainRef' as a reference), whose second element is the point estimate for the offset and whose third element contains iteration-specific offsets.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Alan O'Callaghan <a.b.o'callaghan@sms.ed.ac.uk>

Examples

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

A <- BASiCS_CorrectOffset(ChainSC, ChainRNA)
```

```
# Offset corrected versions for ChainSC (with respect to ChainRNA).
A$Chain
A$Offset
```

`BASiCS_DenoisedCounts` *Calculates denoised expression expression counts*

Description

Calculates denoised expression counts by removing cell-specific technical variation. The latter includes global-scaling normalisation and therefore no further normalisation is required.

Usage

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

Arguments

`Data`	an object of class `SingleCellExperiment`
`Chain`	an object of class `BASiCS_Chain`

Details

See vignette `browseVignettes("BASiCS")`

Value

A matrix of denoised expression counts. In line with global scaling normalisation strategies, these are defined as $X_{ij}/(\phi_j \nu_j)$ for biological genes and $X_{ij}/(\nu_j)$ for spike-in genes. For this calculation ϕ_j ν_j are estimated by their corresponding posterior medians. If spike-ins are not used, ϕ_j is set equal to 1.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

`BASiCS_Chain`

Examples

```
Data <- makeExampleBASiCS_Data(WithSpikes = TRUE)
## The N and Burn parameters used here are optimised for speed
## and should not be used in regular use.
## For more useful parameters,
## see the vignette (\code{browseVignettes("BASiCS")})
Chain <- BASiCS_MCMC(Data, N = 1000, Thin = 10, Burn = 500,
                     Regression = FALSE, PrintProgress = FALSE)
```

```
DC <- BASiCS_DenoisedCounts(Data, Chain)
```

`BASiCS_DenoisedRates` *Calculates denoised expression rates*

Description

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

Usage

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

Arguments

`Data`	an object of class `SingleCellExperiment`
`Chain`	an object of class `BASiCS_Chain`
`Propensities`	If `TRUE`, returns underlying expression propensitites ρ_{ij}. Otherwise, denoised rates $\mu_i \rho_{ij}$ are returned. Default: `Propensities = FALSE`.

Details

See vignette

Value

A matrix of denoised expression rates (biological genes only)

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

`BASiCS_Chain`

Examples

```
Data <- makeExampleBASiCS_Data(WithSpikes = TRUE)
## The N and Burn parameters used here are optimised for speed
## and should not be used in regular use.
## For more useful parameters,
## see the vignette (\code{browseVignettes("BASiCS")})
Chain <- BASiCS_MCMC(Data, N = 1000, Thin = 10, Burn = 500,
                     Regression = FALSE, PrintProgress = FALSE)

DR <- BASiCS_DenoisedRates(Data, Chain)
```

BASiCS_DetectHVG *Detection method for highly (HVG) and lowly (LVG) variable genes*

Description

Functions to detect highly and lowly variable genes. If the BASiCS_Chain object was generated using the regression approach, BASiCS finds the top highly variable genes based on the posteriors of the epsilon parameters. Otherwise, the old approach is used, which initially performs a variance decomposition.

Usage

```
BASiCS_DetectHVG(
  Chain,
  PercentileThreshold = 0.9,
  VarThreshold = NULL,
  ProbThreshold = NULL,
  EFDR = 0.1,
  OrderVariable = c("Prob", "GeneIndex", "GeneName"),
  Plot = FALSE,
  ...
)

BASiCS_DetectLVG(
  Chain,
  PercentileThreshold = 0.1,
  VarThreshold = NULL,
  ProbThreshold = NULL,
  EFDR = 0.1,
  OrderVariable = c("Prob", "GeneIndex", "GeneName"),
  Plot = FALSE,
  ...
)
```

Arguments

`Chain`	an object of class `BASiCS_Chain`
`PercentileThreshold`	Threshold to detect a percentile of variable genes (must be a positive value, between 0 and 1). Defaults: 0.9 for HVG (top 10 percent), 0.1 for LVG (bottom 10 percent)
`VarThreshold`	Variance contribution threshold (must be a positive value, between 0 and 1). This is only used when the BASiCS non-regression model was used to generate the Chain object.
`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 = 0.10)
`OrderVariable`	Ordering variable for output. Possible values: `'GeneIndex'`, `'GeneName'` and `'Prob'`.

Plot	If Plot = TRUE error control and expression versus HVG/LVG probability plots are generated
...	Graphical parameters (see par).

Details

See vignette

Value

BASiCS_DetectHVG returns a list of 4 elements:

Table Matrix whose columns can contain

- GeneIndex Vector of length q.bio. Gene index as in the order present in the analysed SingleCellExperiment
- GeneName Vector of length q.bio. Gene name as in the order present in the analysed SingleCellExperiment
- Mu Vector of length q.bio. For each biological gene, posterior median of gene-specific mean expression parameters μ_i
- Delta Vector of length q.bio. For each biological gene, posterior median of gene-specific biological over-dispersion parameter δ_i
- Sigma Vector of length q.bio. For each biological gene, proportion of the total variability that is due to a biological heterogeneity component.
- Epsilon Vector of length q.bio. For each biological gene, posterior median of gene-specific residual over-dispersion parameter ϵ_i.
- 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.
- LVG Vector of length q.bio. For each biological gene, indicator of being detected as lowly 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.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

References

Vallejos, Marioni and Richardson (2015). PLoS Computational Biology.

See Also

BASiCS_Chain

Examples

```
# Loads short example chain (non-regression implementation)
data(ChainSC)

# 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)

# Loads short example chain (regression implementation)
data(ChainSCReg)

# Highly and lowly variable genes detection (within a single group of cells)
DetectHVG <- BASiCS_DetectHVG(ChainSCReg, PercentileThreshold = 0.90,
                              EFDR = 0.10, Plot = TRUE)
DetectLVG <- BASiCS_DetectLVG(ChainSCReg, PercentileThreshold = 0.10,
                              EFDR = 0.10, Plot = TRUE)
```

`BASiCS_DiagHist` *Create diagnostic plots of MCMC parameters*

Description

Plot a histogram of effective sample size or Geweke's diagnostic z-statistic. See effectiveSize and geweke.diag for more details.

Usage

```
BASiCS_DiagHist(object, Param = NULL, na.rm = TRUE)

BASiCS_diagHist(...)
```

Arguments

`object`	an object of class `BASiCS_Summary`
`Param`	Optional name of a chain parameter to restrict the histogram; if not supplied, all parameters will be assessed. Possible values: `'mu'`, `'delta'`, `'phi'`, `'s'`, `'nu'`, `'theta'`, `'beta'`, `'sigma2'` and `'epsilon'`. Default `Param = NULL`.
`na.rm`	Logical value indicating whether NA values should be removed before calculating effective sample size.
`...`	Unused.

Value

A ggplot object.

Author(s)

Alan O'Callaghan <a.b.ocallaghan@sms.ed.ac.uk>

See Also

BASiCS_Chain

Examples

```
# Built-in example chain
data(ChainSC)

# See effective sample size distribution across all parameters
BASiCS_DiagHist(ChainSC)
# For mu only
BASiCS_DiagHist(ChainSC, Param = "mu")
```

BASiCS_DiagPlot *Create diagnostic plots of MCMC parameters*

Description

Plot parameter values and effective sample size. See effectiveSize for more details on this diagnostic measure.

Usage

```
BASiCS_DiagPlot(
  object,
  Param = "mu",
  x = NULL,
  y = NULL,
  LogX = isTRUE(x %in% c("mu", "delta")),
  LogY = isTRUE(y %in% c("mu", "delta")),
  Smooth = TRUE,
  na.rm = TRUE
)

BASiCS_diagPlot(...)
```

Arguments

object	an object of class BASiCS_Summary
Param	Optional name of a chain parameter to restrict the histogram; if not supplied, all parameters will be assessed. Possible values: 'mu', 'delta', 'phi', 's', 'nu', 'theta', 'beta', 'sigma2' and 'epsilon'. Default Param = 'mu'
x, y	Optional MCMC parameter values to be plotted on the x or y axis, respectively. If neither is supplied, Param will be plotted on the x axis and coda::effectiveSize(Param) will be plotted on the y axis as a density plot.
LogX, LogY	A logical value indicating whether to use a log10 transformation for the x or y axis, respectively.
Smooth	A logical value indicating whether to use smoothing (specifically hexagonal binning using geom_hex).

na.rm	Logical value indicating whether NA values should be removed before calculating effective sample size.
...	Unused.

Value

A ggplot object.

Author(s)

Alan O'Callaghan <a.b.ocallaghan@sms.ed.ac.uk>

See Also

BASiCS_Chain

Examples

```
# Built-in example chain
data(ChainSC)

# Point estimates versus effective sample size
BASiCS_DiagPlot(ChainSC, Param = "mu")
# Effective sample size as colour, mu as x, delta as y.
BASiCS_DiagPlot(ChainSC, x = "mu", y = "delta")
```

BASiCS_EffectiveSize *Calculate effective sample size for BASiCS_Chain parameters*

Description

A wrapper of coda::effectiveSize to be used with BASiCS_Chain objects.

Usage

```
BASiCS_EffectiveSize(object, Param, na.rm = TRUE)

BASiCS_effectiveSize(...)
```

Arguments

object	an object of class BASiCS_Chain.
Param	The parameter to use to calculate effectiveSize. Possible values: 'mu', 'delta', 'phi', 's', 'nu', 'theta', 'beta', 'sigma2' and 'epsilon'.
na.rm	Remove NA values before calculating effectiveSize. Only relevant when Param = "epsilon" (genes with very low expression are excluding when infering the mean/over-dispersion trend. Default: na.rm = TRUE.
...	Unused.

Value

A vector with effective sample sizes for all the elements of `Param`

Examples

```
data(ChainSC)
BASiCS_EffectiveSize(ChainSC, Param = "mu")
```

`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 `SingleCellExperiment` object required to run BASiCS. For more systematic tools for quality control, please refer to the `scater` Bioconductor package.

Usage

```
BASiCS_Filter(
  Counts,
  Tech = rep(FALSE, nrow(Counts)),
  SpikeInput = NULL,
  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 = FALSE` 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: `BatchInfo = NULL`.
`MinTotalCountsPerCell`	Minimum value of total expression counts required per cell (biological and technical). Default: `MinTotalCountsPerCell = 2`.
`MinTotalCountsPerGene`	Minimum value of total expression counts required per transcript (biological and technical). Default: `MinTotalCountsPerGene = 2`.

`MinCellsWithExpression`
: Minimum number of cells where expression must be detected (positive count). Criteria applied to each transcript. Default: `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>

Examples

```
set.seed(1)
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$SpikeID %in% rownames(Filter$Counts),]
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` object

Usage

```
BASiCS_LoadChain(RunName, StoreDir = getwd(), StoreUpdatedChain = FALSE)
```

Arguments

RunName	String used to index '.Rds' file containing the MCMC chain (produced by the BASiCS_MCMC function, with StoreChains = TRUE)
StoreDir	Directory where '.Rds' file is stored. Default: StoreDir = getwd()
StoreUpdatedChain	Only required when the input files contain an outdated version of a BASiCS_Chain object. If StoreUpdatedChain = TRUE, an updated object is saved (this overwrites original input file, if it was an '.Rds' file).

Value

An object of class BASiCS_Chain.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

BASiCS_Chain

Examples

```
Data <- makeExampleBASiCS_Data()
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 5, Burn = 5, Regression = FALSE,
                     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, Regression, WithSpikes = TRUE, ...)
```

Arguments

Data	A SingleCellExperiment object. If WithSpikes = TRUE, this MUST be formatted to include the spike-ins and/or batch information (see vignette).
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.
Regression	If Regression = TRUE, BASiCS exploits a joint prior formulation for mean and over-dispersion parameters to estimate a measure of residual over-dispersion is not confounded by mean expression. Recommended setting is Regression = TRUE.
WithSpikes	If WithSpikes = TRUE, BASiCS will use reads from added spike-ins to estimate technical variability. If WithSpikess = FALSE, BASiCS depends on replicated experiments (batches) to estimate technical variability. In this case, please supply the BatchInfo vector in colData(Data). Default: WithSpikes = TRUE.
...	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, 2016). 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 μ_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 over-dispersion parameters δ_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 over-dispersion parameters δ_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 over-dispersion hyper-parameters δ_i. Default: b.delta = 1.
- p.phi Dirichlet hyper-parameter for the joint of all (scaled by n) cell-specific mRNA content normalising constants ϕ_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 parameter θ. Default: a.theta = 1.

b.theta Rate hyper-parameter for the Gamma(a.theta,b.theta) prior for technical noise parameter θ. Default: b.theta = 1.

eta Only used when Regression = TRUE. eta specifies the degress of freedom for the residual term. Default: eta = 5..

k Only used when Regression = TRUE. k specifies the number of regression Gaussian Radial Basis Functions (GRBF) used within the correlated prior adopted for gene-specific over-dispersion and mean expression paramters. Default: k = 12.

Var Only used when Regression = TRUE. Var specifies the GRBF scaling parameter. Default: Var = 1.2.

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 generated BASiCS_Chain object is stored as a '.Rds' file (RunName argument used to index the file name). Default: StoreChains = FALSE.

StoreAdapt If StoreAdapt = TRUE, trajectory of adaptive proposal variances (in log-scale) for all parameters is stored as a list in a '.Rds' file (RunName argument used to index file name). 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 '.Rds' files storing chains and/or adaptive proposal variances.

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

Start Starting values for the MCMC sampler. We do not advise to use this argument. Default options have been tuned to facilitate convergence. If changed, it must be a list containing the following elements: mu0, delta0, phi0, s0, nu0, theta0, ls.mu0, ls.delta0, ls.phi0, ls.nu0 and ls.theta0

Value

An object of class BASiCS_Chain.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

References

Vallejos, Marioni and Richardson (2015). PLoS Computational Biology.

Vallejos, Richardson and Marioni (2016). Genome Biology.

Eling et al (2018). Cell Systems

Examples

```
# Built-in simulated dataset
set.seed(1)
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, Regression = FALSE,
                     PrintProgress = FALSE, WithSpikes = TRUE)

# To run the regression version of BASiCS, use:
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 10, Regression = TRUE,
                     PrintProgress = FALSE, WithSpikes = TRUE)

# To run the non-spike version BASiCS requires the data to contain at least
# 2 batches:
set.seed(2)
Data <- makeExampleBASiCS_Data(WithBatch = TRUE)
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 2, Burn = 10, Regression = TRUE,
                     PrintProgress = FALSE, WithSpikes = 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
# For example:
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
# For example:
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)

# To obtain denoised rates / counts, see:
# help(BASiCS_DenoisedRates)
# and
# help(BASiCS_DenoisedCounts)

# For examples of differential analyses between 2 populations of cells see:
# help(BASiCS_TestDE)
```

BASiCS_ShowFit *Plotting the trend after Bayesian regression*

Description

Plotting the trend after Bayesian regression using a BASiCS_Chain object

Usage

```
BASiCS_ShowFit(
  object,
  xlab = "log(mu)",
  ylab = "log(delta)",
  pch = 16,
  smooth = TRUE,
  variance = 1.2,
  colour = "dark blue",
  markExcludedGenes = TRUE,
  GenesSel = NULL,
  colourGenesSel = "dark red",
  Uncertainty = TRUE
)
```

Arguments

object	an object of class BASiCS_Chain
xlab	As in par.
ylab	As in par.
pch	As in par. Default value pch = 16.
smooth	Logical to indicate wether the smoothScatter function is used to plot the scatter plot. Default value smooth = TRUE.

variance	Variance used to build GRBFs for regression. Default value variance = 1.2
colour	colour used to denote genes within the scatterplot. Only used when smooth = TRUE. Default value colour = "dark blue".
markExcludedGenes	Whether or not lowly expressed genes that were excluded from the regression fit are included in the scatterplot. Default value markExcludedGenes = TRUE.
GenesSel	Vector of gene names to be highlighted in the scatterplot. Only used when smooth = TRUE. Default value GenesSel = NULL.
colourGenesSel	colour used to denote the genes listed in GenesSel within the scatterplot. Default value colourGenesSel = "dark red".
Uncertainty	logical indicator. If true, statistical uncertainty around the regression fit is shown in the plot.

Value

A ggplot2 object

Author(s)

Nils Eling <eling@ebi.ac.uk>

Catalina Vallejos <cnvallej@uc.cl>

References

Eling et al (2018). Cell Systems https://doi.org/10.1016/j.cels.2018.06.011

Examples

```
data(ChainRNAReg)
BASiCS_ShowFit(ChainRNAReg)
```

BASiCS_Sim *Generates synthetic data according to the model implemented in BASiCS*

Description

BASiCS_Sim creates a simulated dataset from the model implemented in BASiCS.

Usage

```
BASiCS_Sim(Mu, Mu_spikes = NULL, Delta, Phi = NULL, S, Theta, BatchInfo = NULL)
```

Arguments

Mu	Gene-specific mean expression parameters μ_i for all biological genes (vector of length q.bio, all elements must be positive numbers)
Mu_spikes	μ_i for all technical genes defined as true input molecules (vector of length q-q.bio, all elements must be positive numbers). If Mu_spikes = NULL, the generated data will not contain spike-ins. If Phi = NULL, Mu_spikes will be ignored. Default: Mu_spikes = NULL.
Delta	Gene-specific biological over-dispersion parameters δ_i, biological genes only (vector of length q.bio, all elements must be positive numbers)
Phi	Cell-specific mRNA content normalising parameters ϕ_j (vector of length n, all elements must be positive numbers and the sum of its elements must be equal to n). Phi must be set equal to NULL when generating data without spike-ins. If Mu_spikes = NULL, Phi will be ignored. Default: Phi = NULL
S	Cell-specific technical normalising parameters s_j (vector of length n, all elements must be positive numbers)
Theta	Technical variability parameter θ (must be positive). Theta can be a scalar (single batch of samples), or a vector (multiple batches of samples). If a value for BatchInfo is provided, the length of Theta must match the number of unique values in BatchInfo.
BatchInfo	same as in newBASiCS_Data. If spike-ins, are not in use, the number of unique values contained in BatchInfo must be larger than 1 (i.e. multiple batches are present).

Value

An object of class SingleCellExperiment, including synthetic data generated by the model implemented in BASiCS.

Author(s)

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

References

Vallejos, Marioni and Richardson (2015). PLoS Computational Biology.

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_spikes <-  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 with spike-ins, single batch
Data <- BASiCS_Sim(Mu, Mu_spikes, Delta, Phi, S, Theta)
head(SingleCellExperiment::counts(Data))
dim(SingleCellExperiment::counts(Data))
```

```
metadata(Data)$SpikeInput
altExp(Data)

# Data with spike-ins, multiple batches
BatchInfo <- c(1,1,1,2,2)
Theta2 <- rep(Theta, times = 2)
Data <- BASiCS_Sim(Mu, Mu_spikes, Delta, Phi, S, Theta2, BatchInfo)

# Data without spike-ins, multiple batches
Data <- BASiCS_Sim(Mu, Mu_spikes = NULL, Delta,
                   Phi = NULL, S, Theta2, BatchInfo)
```

`BASiCS_Summary` *The BASiCS_Summary class*

Description

Container of a summary of a `BASiCS_Chain` object. In each element of the `parameters` slot, first column contains posterior medians; second and third columns respectively contain the lower and upper limits of an high posterior density interval (for a given probability).

Slots

`parameters` List of parameters in which each entry contains a matrix: 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.

- **mu** Posterior medians (1st column), lower (2nd column) and upper (3rd column) limits of gene-specific mean expression parameters μ_i.
- **delta** Posterior medians (1st column), lower (2nd column) and upper (3rd column) limits of gene-specific biological over-dispersion parameters δ_i, biological genes only
- **phi** Posterior medians (1st column), lower (2nd column) and upper (3rd column) limits of cell-specific mRNA content normalisation parameters ϕ_j
- **s** Posterior medians (1st column), lower (2nd column) and upper (3rd column) limits of cell-specific technical normalisation parameters $s[j]$
- **nu** Posterior medians (1st column), lower (2nd column) and upper (3rd column) limits of cell-specific random effects ν_j
- **theta** Posterior median (1st column), lower (2nd column) and upper (3rd column) limits of technical over-dispersion parameter(s) θ (each row represents one batch)
- `beta` Posterior median (first column), lower (second column) and upper (third column) limits of regression coefficients β
- `sigma2` Posterior median (first column), lower (second column) and upper (third column) limits of residual variance σ^2
- `epsilon` Posterior median (first column), lower (second column) and upper (third column) limits of gene-specific residual over-dispersion parameter ϵ

Examples

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

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

Description

'show' method for BASiCS_Summary 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>

Nils Eling <eling@ebi.ac.uk>

Examples

```
data(ChainSC)
show(ChainSC)
```

BASiCS_TestDE *Detection of genes with changes in expression*

Description

Function to assess changes in expression between two groups of cells (mean and over-dispersion)

Usage

```
BASiCS_TestDE(
  Chain1,
  Chain2,
  EpsilonM = log2(1.5),
  EpsilonD = log2(1.5),
  EpsilonR = log2(1.5)/log2(exp(1)),
  ProbThresholdM = 2/3,
  ProbThresholdD = 2/3,
  ProbThresholdR = 2/3,
  OrderVariable = "GeneIndex",
  GroupLabel1 = "Group1",
  GroupLabel2 = "Group2",
  Plot = TRUE,
  PlotOffset = TRUE,
  Offset = TRUE,
  EFDR_M = 0.05,
  EFDR_D = 0.05,
  EFDR_R = 0.05,
  GenesSelect = NULL,
  min.mean = 1,
  ...
)
```

Arguments

Chain1	an object of class BASiCS_Chain containing parameter estimates for the first group of cells
Chain2	an object of class BASiCS_Chain 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 biological over-dispersion (must be a positive real number). Default value: EpsilonM = log2(1.5) (i.e. 50% increase).
EpsilonR	Minimum distance threshold for detecting changes in residual over-dispersion (must be a positive real number). Default value: EpsilonR= log2(1.5)/log2(exp(1)) (i.e. 50% increase).
ProbThresholdM	Optional parameter. Probability threshold for detecting changes in overall expression (must be a positive value, between 0 and 1). If EFDR_M = NULL, the posterior probability threshold for the differential mean expression test will be set

to ProbThresholdM. If a value for EFDR_M is provided, the posterior probability threshold is chosen to achieve an EFDR equal to EFDR_M and ProbThresholdM defines a minimum probability threshold for this calibration (this avoids low values of ProbThresholdM to be chosen by the EFDR calibration. Default value ProbThresholdM = 2/3, i.e. the probability of observing a log2-FC above EpsilonM must be at least twice the probality of observing the complementary event (log2-FC below EpsilonM).

ProbThresholdD — Optional parameter. Probability threshold for detecting changes in cell-to-cell biological over-dispersion (must be a positive value, between 0 and 1). Same usage as ProbThresholdM, depending on the value provided for EFDR_D. Default value ProbThresholdD = 2/3.

ProbThresholdR — Optional parameter. Probability threshold for detecting changes in residual over-dispersion (must be a positive value, between 0 and 1). Same usage as ProbThresholdM, depending on the value provided for EFDR_R. Default value ProbThresholdR = 2/3.

OrderVariable — Ordering variable for output. Possible values: 'GeneIndex' (default), 'GeneName' and 'Mu' (mean expression).

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). Default: Offset = TRUE.

EFDR_M — Target for expected false discovery rate related to the comparison of means. If EFDR_M = NULL, EFDR calibration is not performed and the posterior probability threshold is set equal to ProbThresholdM. Default EFDR_M = 0.05.

EFDR_D — Target for expected false discovery rate related to the comparison of dispersions. If EFDR_D = NULL, EFDR calibration is not performed and the posterior probability threshold is set equal to ProbThresholdD.Default EFDR_D = 0.05.

EFDR_R — Target for expected false discovery rate related to the comparison of residual over-dispersions. If EFDR_R = NULL, EFDR calibration is not performed and the posterior probability threshold is set equal to ProbThresholdR.Default EFDR_D = 0.05.

GenesSelect — Optional argument to provide a user-defined list of genes to be considered for the comparison. Default: GenesSelect = 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.

min.mean — Minimum mean expression threshold required for inclusion in offset calculation. Similar to ‘min.mean‘ in ‘scran::computeSumFactors‘. This parameter is only relevant with ‘Offset = TRUE‘.

... — Optional parameters.

Value

BASiCS_TestDE returns a list of 4 elements:

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

- `GeneName` Gene name
- `MeanOverall` For each gene, the estimated mean expression parameter μ_i is averaged across both groups of cells (weighted by sample size).
- `Mean1` Estimated mean expression parameter μ_i for each biological gene in the first group of cells.
- `Mean2` Estimated mean expression parameter μ_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 does not changes).

- `GeneName` Gene name
- `MeanOverall` For each gene, the estimated mean expression parameter μ_i is averaged across both groups of cells (weighted by sample size).
- `DispOverall` For each gene, the estimated over-dispersion parameter δ_i is averaged across both groups of cells (weighted by sample size).
- `Disp1` Estimated over-dispersion parameter δ_i for each biological gene in the first group of cells.
- `Disp2` Estimated over-dispersion parameter δ_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. Genes labelled with "ExcludedFromTest" were detected as showing differential mean expression.

`TableResDisp` A `data.frame` containing the results of the differential residual over-dispersion test.

- `GeneName` Gene name
- `MeanOverall` For each gene, the estimated mean expression parameter μ_i is averaged across both groups of cells (weighted by sample size).
- `ResDispOverall` For each gene, the estimated residual over-dispersion parameter δ_i is averaged across both groups of cells (weighted by sample size).
- `ResDisp1` Estimated residual over-dispersion parameter ϵ_i for each biological gene in the first group of cells.
- `ResDisp2` Estimated residual over-dispersion parameter ϵ_i for each biological gene in the second group of cells.

`ResDispDistance` Difference in residual over-dispersion between the first and second groups of cells.

`ProbDiffResDisp` Posterior probability for residual over-dispersion difference between the first and second groups of cells.

`ResultDiffResDisp` Indicator if a gene has a higher residual over-dispersion in the first or second groups of cells. Genes labelled with "ExcludedFromTest" were not expressed in at least 2 cells per condition.

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

`ProbThreshold` Posterior probability threshold.

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

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

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

`ProbThreshold` Posterior probability threshold.

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

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

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

`ProbThreshold` Posterior probability threshold.

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

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

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

`Chain2_offset` an `BASiCS_Chain` 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>

Nils Eling <eling@ebi.ac.uk>

Examples

```
# Loading two 'BASiCS_Chain' objects (obtained using 'BASiCS_MCMC')
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)

# For testing differences in residual over-dispersion, two chains obtained
# via 'BASiCS_MCMC(Data, N, Thin, Burn, Regression=TRUE)' need to be provided
data(ChainSCReg)
data(ChainRNAReg)

Test <- BASiCS_TestDE(Chain1 = ChainSCReg, Chain2 = ChainRNAReg,
                      GroupLabel1 = 'SC', GroupLabel2 = 'P&S',
                      EpsilonM = log2(1.5), EpsilonD = log2(1.5),
                      EpsilonR = log2(1.5)/log2(exp(1)),
                      OffSet = TRUE)
```

`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,
  main = "Overall variance decomposition",
  ylab = "% of variance",
  beside = FALSE,
  col = c("lightblue", "mistyrose", "lightcyan"),
  legend = c("Biological", "Technical", "Shot noise"),
  args.legend = list(x = "bottomright", bg = "white"),
 names.arg = if (nBatch > 1) {    c("Overall", paste("Batch ", seq_len(nBatch))) }
    else "Overall"
)
```

Arguments

`Chain`	an object of class `BASiCS_Chain`
`OrderVariable`	Ordering variable for output. Possible values: `'GeneName'`, `'BioVarGlobal'`, `'TechVarGlobal'` and `'ShotNoiseGlobal'`. Default: `OrderVariable = "BioVarGlobal"`.
`Plot`	If `TRUE`, a barplot of the variance decomposition (global and by batches, if any) is generated. Default: `Plot = TRUE`.
`main, ylab, beside, col, legend, args.legend, names.arg`	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 component (overall across all cells)

TechVarGlobal Percentage of variance explained by the technical component (overall across all cells)

ShotNoiseGlobal Percentage of variance explained by the shot noise component (baseline Poisson noise, 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). PLoS Computational Biology.

See Also

BASiCS_Chain

Examples

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

VD <- BASiCS_VarianceDecomp(ChainSC)
```

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. Only used when HVG/LVG are found based on the variance decomposition.

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`
`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: `EFDR = 0.10`.
`Progress`	If `Progress = TRUE`, partial output is printed in the console. Default: `Progress = TRUE`.

Details

See vignette

Value

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

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

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

References

Vallejos, Marioni and Richardson (2015). PLoS Computational Biology.

See Also

`BASiCS_Chain`

Examples

```
data(ChainSC)

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)
```

`ChainRNA` *Extract from the chain obtained for the Grun et al (2014) data: pool-and-split samples*

Description

Small extract (75 MCMC iterations, 350 randomly selected genes) from the chain obtained for the pool-and-split samples (this corresponds to the RNA 2i samples in Grun et al, 2014).

Usage

```
ChainRNA
```

Format

An object of class `BASiCS_Chain` containing 75 MCMC iterations.

References

Grun, Kester and van Oudenaarden (2014). Nature Methods.

`ChainRNAReg` *Extract from the chain obtained for the Grun et al (2014) data: pool-and-split samples (regression model)*

Description

Small extract (75 MCMC iterations, 350 randomly selected genes) from the chain obtained for the pool-and-split samples (this corresponds to the RNA 2i samples in Grun et al, 2014).

Usage

```
ChainRNAReg
```

Format

An object of class `BASiCS_Chain` containing 75 MCMC iterations.

References

Grun, Kester and van Oudenaarden (2014). Nature Methods.

ChainSC — *Extract from the chain obtained for the Grun et al (2014) data: single-cell samples*

Description

Small extract (75 MCMC iterations, 350 randomly selected genes) from the chain obtained for the pool-and-split samples (this corresponds to the SC 2i samples in Grun et al, 2014).

Usage

```
ChainSC
```

Format

An object of class `BASiCS_Chain` containing 75 MCMC iterations.

References

Grun, Kester and van Oudenaarden (2014). Nature Methods.

ChainSCReg — *Extract from the chain obtained for the Grun et al (2014) data: single-cell samples (regression model)*

Description

Small extract (75 MCMC iterations, 350 randomly selected genes) from the chain obtained for the pool-and-split samples (this corresponds to the SC 2i samples in Grun et al, 2014).

Usage

```
ChainSCReg
```

Format

An object of class `BASiCS_Chain` containing 75 MCMC iterations.

References

Grun, Kester and van Oudenaarden (2014). Nature Methods.

colnames *'colnames' method for BASiCS_Chain objects*

Description

Returns the labels of cell-specific BASiCS parameters

Usage

```
## S4 method for signature 'BASiCS_Chain'
colnames(x)
```

Arguments

x	A BASiCS_Chain object.

Value

An vector of labels

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Examples

```
data(ChainSC)
colnames(ChainSC)
```

displayChainBASiCS-BASiCS_Chain-method *Accessors for the slots of a BASiCS_Chain object*

Description

Accessors for the slots of a BASiCS_Chain

Usage

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

Arguments

object	an object of class BASiCS_Chain
Param	Name of the slot to be used for the accessed. Possible values: 'mu', 'delta', 'phi', 's', 'nu', 'theta', 'beta', 'sigma2' and 'epsilon'.

Value

The requested slot of a BASiCS_Chain object

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

BASiCS_Chain

Examples

```
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 object

Usage

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

Arguments

object	an object of class BASiCS_Summary
Param	Name of the slot to be used for the accessed. Possible values: 'mu', 'delta', 'phi', 's', 'nu', 'theta', 'beta', 'sigma2' and 'epsilon'.

Value

The requested slot of a BASiCS_Summary object

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

BASiCS_Summary

Examples

```
help(BASiCS_MCMC)
```

`makeExampleBASiCS_Data` *Create a synthetic SingleCellExperiment example object with the format required for BASiCS*

Description

A synthetic `SingleCellExperiment` object is generated by simulating a dataset from the model underlying BASiCS. This is used to illustrate BASiCS in some of the package and vignette examples.

Usage

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

Arguments

`WithBatch`	If `TRUE`, 2 batches are generated (each of them containing 15 cells). Default: `WithBatch = FALSE`.
`WithSpikes`	If `TRUE`, the simulated dataset contains 20 spike-in genes. If `WithSpikes = FALSE`, `WithBatch` is automatically set to `TRUE`. Default: `WithSpikes = TRUE`

Details

Note: In BASiCS versions < 1.5.22, `makeExampleBASiCS_Data` used a fixed seed within the function. This has been removed to comply with Bioconductor policies. If a reproducible example is required, please use `set.seed` prior to calling `makeExampleBASiCS_Data` .

Value

An object of class `SingleCellExperiment`, with synthetic data simulated from the model implemented in BASiCS. If `WithSpikes = TRUE`, it contains 70 genes (50 biological and 20 spike-in) and 30 cells. Alternatively, it contains 50 biological genes and 30 cells.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Examples

```
Data <- makeExampleBASiCS_Data()
is(Data, 'SingleCellExperiment')
```

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

Description

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

Usage

```
newBASiCS_Chain(parameters)
```

Arguments

parameters List of matrices containing MCMC chains for each model parameter.

mu MCMC chain for gene-specific mean expression parameters μ_i, biological genes only (matrix with q.bio columns, all elements must be positive numbers)

delta MCMC chain for gene-specific biological over-dispersion parameters δ_i, biological genes only (matrix with q.bio columns, all elements must be positive numbers)

phi MCMC chain for cell-specific mRNA content normalisation parameters ϕ_j (matrix with n columns, all elements must be positive numbers and the sum of its elements must be equal to n). This parameter is only used when spike-in genes are available.

s MCMC chain for cell-specific technical normalisation parameters s_j (matrix with n columns, all elements must be positive numbers)

nu MCMC chain for cell-specific random effects ν_j (matrix with n columns, all elements must be positive numbers)

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

beta Only used for regression model. MCMC chain for regression coefficients (matrix with k columns, where k represent the number of chosen basis functions + 2)

sigma2 Only used for regression model. MCMC chain for the residual variance (matrix with one column, sigma2 represents a global parameter)

epsilon Only used for regression model. MCMC chain for the gene specific residual over-dispersion parameter (mean corrected vraribility) (matrix with q columns)

Value

An object of class BASiCS_Chain.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

BASiCS_Chain

Examples

```
Data <- makeExampleBASiCS_Data()

# No regression model
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 5, Burn = 5, Regression = FALSE)

ChainMu <- displayChainBASiCS(Chain, 'mu')
ChainDelta <- displayChainBASiCS(Chain, 'delta')
ChainPhi <- displayChainBASiCS(Chain, 'phi')
ChainS <- displayChainBASiCS(Chain, 's')
ChainNu <- displayChainBASiCS(Chain, 'nu')
ChainTheta <- displayChainBASiCS(Chain, 'theta')

ChainNew <- newBASiCS_Chain(parameters = list(mu = ChainMu,
                                              delta = ChainDelta,
                                              phi = ChainPhi,
                                              s = ChainS,
                                              nu = ChainNu,
                                              theta = ChainTheta))


# No regression model
Chain <- BASiCS_MCMC(Data, N = 50, Thin = 5, Burn = 5, Regression = TRUE)

ChainMu <- displayChainBASiCS(Chain, 'mu')
ChainDelta <- displayChainBASiCS(Chain, 'delta')
ChainPhi <- displayChainBASiCS(Chain, 'phi')
ChainS <- displayChainBASiCS(Chain, 's')
ChainNu <- displayChainBASiCS(Chain, 'nu')
ChainTheta <- displayChainBASiCS(Chain, 'theta')
ChainBeta <- displayChainBASiCS(Chain, 'beta')
ChainSigma2 <- displayChainBASiCS(Chain, 'sigma2')
ChainEpsilon <- displayChainBASiCS(Chain, 'epsilon')

ChainNew <- newBASiCS_Chain(parameters = list(mu = ChainMu,
                                              delta = ChainDelta,
                                              phi = ChainPhi,
                                              s = ChainS,
                                              nu = ChainNu,
                                              theta = ChainTheta,
                                              beta = ChainBeta,
                                              sigma2 = ChainSigma2,
                                              epsilon = ChainEpsilon))
```

---

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

Description

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

Usage

```
newBASiCS_Data(
  Counts,
  Tech = rep(FALSE, nrow(Counts)),
  SpikeInfo = NULL,
  BatchInfo = NULL,
  SpikeType = "ERCC"
)
```

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. Defaul value: Tech = rep(FALSE,nrow(Counts)).
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. If SpikeInfo = NULL, only the horizontal integration implementation (no spikes) can be run. Default value: SpikeInfo = NULL.
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.
SpikeType	Character to indicate what type of spike-ins are in use. Default value: SpikeType = "ERCC" (parameter is no longer used).

Value

An object of class SingleCellExperiment.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

See Also

SingleCellExperiment

Examples

```
## Data with spike-ins

# Expression counts
set.seed(1)
Counts <- matrix(rpois(50*10, 2), ncol = 10)
```

```
rownames(Counts) <- c(paste0('Gene', 1:40), paste0('ERCC', 1:10))
# Technical information
Tech <- grepl("ERCC", rownames(Counts))
# 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 = Tech, SpikeInfo = SpikeInfo)

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

## Data without spike-ins (BatchInfo is required)

# Expression counts
set.seed(1)
Counts <- matrix(rpois(50*10, 2), ncol = 10)
rownames(Counts) <- paste0('Gene', 1:50)
BatchInfo <- c(rep(1, 5), rep(2, 5))

# Creating a BASiCS_Data object (with batch effect)
DataExample <- newBASiCS_Data(Counts, BatchInfo = BatchInfo)
```

---

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,
  RegressionTerm = NULL,
  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', 'beta', 'sigma2' and 'epsilon'.
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'
RegressionTerm	Specifies which regression coefficient is requested. Required only if Param = 'beta'
ylab	As in par.
xlab	As in par.
...	Other graphical parameters (see par).

Value

A plot object

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Examples

```
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,
  RegressionTerms = NULL,
  xlab = "",
```

```
  ylab = "",
  xlim = "",
  ylim = NULL,
  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', 'beta', 'sigma2' and 'epsilon'.
Param2	Name of the second slot to be used for the plot. Possible values: 'mu', 'delta', 'epsilon', 'phi', 's' and 'nu' (combinations between gene-specific and cell-specific parameters are not admitted).
Genes	Specifies which genes are requested. Required only if Param = 'mu', 'delta' or 'epsilon'.
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'
RegressionTerms	Specifies which regression coefficients are requested. Required only if Param = 'beta'
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).

Value

A plot object

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Examples

```
help(BASiCS_MCMC)
```

rownames *'rownames' method for BASiCS_Chain objects*

Description

Returns the labels of gene-specific BASiCS parameters

Usage

```
## S4 method for signature 'BASiCS_Chain'
rownames(x)
```

Arguments

x	A BASiCS_Chain object.

Value

An vector of labels

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Examples

```
data(ChainSC)
rownames(ChainSC)
```

subset *A 'subset' method for 'BASiCS_Chain" objects*

Description

This can be used to extract a subset of a 'BASiCS_Chain' object. The subset can contain specific genes, cells or MCMC iterations

Usage

```
## S4 method for signature 'BASiCS_Chain'
subset(x, Genes = NULL, Cells = NULL, Iterations = NULL)
```

Arguments

x	A BASiCS_Chain object.
Genes	A vector of characters indicating what genes will be extracted.
Cells	A vector of characters indicating what cells will be extrated.
Iterations	Numeric vector of positive integers indicating which MCMC iterations will be extracted. The maximum value in Iterations must be less or equal than the total number of iterations contained in the original BASiCS_Chain object.

Value

An object of class BASiCS_Chain.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Examples

```
data(ChainSC)

# Extracts 3 first genes
ChainSC1 <- subset(ChainSC, Genes = rownames(ChainSC)[1:3])
# Extracts 3 first cells
ChainSC2 <- subset(ChainSC, Cells = colnames(ChainSC)[1:3])
# Extracts 10 first iterations
ChainSC3 <- subset(ChainSC, Iterations = 1:10)
```

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 (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.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Examples

```
data(ChainSC)
SummarySC <- Summary(ChainSC)
```

Index