

Package ‘BASiCS’

March 29, 2021

Type Package

Title Bayesian Analysis of Single-Cell Sequencing data

Version 2.2.2

Date 2021-01-29

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 (≥ 4.0), SingleCellExperiment

Imports Biobase, BiocGenerics, coda, cowplot, ggExtra, ggplot2, graphics, grDevices, MASS, methods, Rcpp ($\geq 0.11.3$), S4Vectors, scran, stats, stats4, SummarizedExperiment, viridis, utils, Matrix, matrixStats, assertthat, reshape2, BiocParallel, hexbin

Suggests BiocStyle, knitr, rmarkdown, testthat, magick

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 7.1.1

Encoding UTF-8

LazyData true

Collate 'AllClasses.R' 'AllGenerics.R' 'BASiCS_CorrectOffset.R'
 'BASiCS_DenoisedCounts.R' 'BASiCS_DenoisedRates.R'
 'BASiCS_DetectHVG_LVG.R' 'BASiCS_DiagHist.R'
 'BASiCS_DiagPlot.R' 'BASiCS_DivideAndConquer.R' 'BASiCS_Draw.R'
 'BASiCS_EffectiveSize.R' 'BASiCS_Filter.R' 'BASiCS_LoadChain.R'
 'BASiCS_MCMC.R' 'BASiCS_MockSCE.R' 'BASiCS_Package.R'
 'BASiCS_PlotDE.R' 'BASiCS_PlotOffset.R' 'BASiCS_PriorParam.R'
 'BASiCS_ShowFit.R' 'BASiCS_Sim.R' 'BASiCS_TestDE.R'
 'BASiCS_VarThresholdSearchHVG_LVG.R' 'BASiCS_VarianceDecomp.R'
 'HiddenBASiCS_Sim.R' 'HiddenHeaderBASiCS_Sim.R'
 'HiddenHeaderTest_DE.R' 'HiddenVarDecomp.R' 'utils_Misc.R'
 'Methods.R' 'RcppExports.R' 'data.R' 'makeExampleBASiCS_Data.R'
 'newBASiCS_Chain.R' 'newBASiCS_Data.R' 'utils_Data.R'
 'utils_DivideAndConquer.R' 'utils_MCMC.R' 'utils_Store.R'
 'utils_Tests.R' 'utils_VG.R' 'welcome.R'

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

git_branch RELEASE_3_12

git_last_commit ec86e47

git_last_commit_date 2021-01-28

Date/Publication 2021-03-29

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:

.generateSubsets	4
as.data.frame,BASiCS_ResultsDE-method	4
BASiCS-defunct	5
BASiCS_Chain	6
BASiCS_Chain-methods	7
BASiCS_CorrectOffset	8
BASiCS_DenoisedCounts	9

BASiCS_DenoisedRates	10
BASiCS_DetectVG	11
BASiCS_DiagHist	13
BASiCS_DiagPlot	14
BASiCS_DivideAndConquer	15
BASiCS_Draw	17
BASiCS_EffectiveSize	18
BASiCS_Filter	19
BASiCS_LoadChain	20
BASiCS_MCMC	21
BASiCS_MockSCE	25
BASiCS_PlotDE	26
BASiCS_PlotOffset	27
BASiCS_PlotVG	28
BASiCS_PriorParam	29
BASiCS_Result	31
BASiCS_ResultDE	31
BASiCS_ResultsDE	32
BASiCS_ResultVG	32
BASiCS_ShowFit	33
BASiCS_Sim	34
BASiCS_Summary	35
BASiCS_Summary-methods	36
BASiCS_TestDE	37
BASiCS_VarianceDecomp	40
BASiCS_VarThresholdSearchHVG	41
ChainRNA	43
ChainRNAREg	43
ChainSC	44
ChainSCReg	44
dim	45
dimnames	45
displayChainBASiCS-BASiCS_Chain-method	46
displaySummaryBASiCS-BASiCS_Summary-method	47
format,BASiCS_ResultsDE-method	47
makeExampleBASiCS_Data	48
newBASiCS_Chain	49
newBASiCS_Data	51
plot-BASiCS_Chain-method	52
plot-BASiCS_Summary-method	53
rowData,BASiCS_ResultsDE-method	54
show,BASiCS_ResultDE-method	55
show,BASiCS_ResultsDE-method	56
show,BASiCS_ResultVG-method	56
subset	57
Summary	58
[,BASiCS_ResultsDE,ANY,ANY,ANY-method	58

`.generateSubsets` *Generate balanced subsets for divide and conquer BASiCS*

Description

Partitions data based on either cells or genes. Attempts to find a partitioning scheme which is "balanced" for either total reads per cell across all genes (partitioning by gene) or total expression per gene across all cells (partitioning by gene). When partitioning by cell, at least 20 cells must be in each partition or BASiCS_MCMC will fail. If this partitioning fails, it will continue recursively up to a maximum number of iterations (20 by default).

Usage

```
.generateSubsets(
  Data,
  NSubsets,
  SubsetBy = c("cell", "gene"),
  Alpha = 0.05,
  WithSpikes = FALSE,
  MaxDepth = 20,
  .Depth = 1
)
```

Arguments

<code>Data</code>	a SingleCellExperiment object
<code>NSubsets</code>	Integer specifying the number of batches into which to divide Data for divide and conquer inference.
<code>SubsetBy</code>	Partition by "cell" or by "gene".
<code>Alpha</code>	p-value threshold for ANOVA testing of "balance"
<code>WithSpikes</code>	Similar to argument for BASiCS_MCMC - do the Data contain spikes?
<code>MaxDepth</code>	Maximum number of recursive
<code>.Depth</code>	Internal parameter. Do not set.

Value

A list of SingleCellExperiment objects

as.data.frame,BASiCS_ResultsDE-method
Converting BASiCS results objects to data.frames

Description

Converting BASiCS results objects to data.frames

Usage

```
## S4 method for signature 'BASiCS_ResultsDE'
as.data.frame(x, Parameter, Filter = TRUE, ProbThreshold = NULL)

## S4 method for signature 'BASiCS_ResultDE'
as.data.frame(x, Filter = TRUE, ProbThreshold = NULL)

## S4 method for signature 'BASiCS_ResultVG'
as.data.frame(x, Filter = TRUE, ProbThreshold = NULL)
```

Arguments

x	An object of class BASiCS_ResultVG , BASiCS_ResultDE , or BASiCS_ResultsDE .
Parameter	For BASiCS_ResultsDE objects only. Character scalar specifying which table of results to output. Available options are "Mean", (mu, mean expression), "Disp" (delta, overdispersion) and "ResDisp" (epsilon, residual overdispersion).
Filter	Logical scalar. If TRUE, output only entries corresponding to genes that pass the decision rule used in the probabilistic test.
ProbThreshold	Only used if Filter=TRUE. Numeric scalar specifying the probability threshold to be used when filtering genes. Default is to use the threshold used in the original decision rule when the test was performed.

Value

A data.frame of test results.

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 column 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 if [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

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 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, WithSpikes = TRUE)
```

Arguments

Data	An object of class SingleCellExperiment
Chain	An object of class BASiCS_Chain
WithSpikes	A logical scalar specifying whether denoised spike-in genes should be generated as part of the output value. This only applies when the BASiCS_Chain object was generated with the setting <code>WithSpikes=TRUE</code> .

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 <cvallej@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 propensities 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_DetectVG	<i>Detection method for highly (HVG) and lowly (LVG) variable genes</i>
-----------------	-------------------------------------------------------------------------

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_DetectVG(
  Chain,
  Task = c("HVG", "LVG"),
  PercentileThreshold = NULL,
  VarThreshold = NULL,
  ProbThreshold = 2/3,
  EpsilonThreshold = NULL,
  EFDR = 0.1,
  OrderVariable = c("Prob", "GeneIndex", "GeneName"),
  Plot = FALSE,
  MinESS = 100,
  ...
)

BASiCS_DetectLVG(Chain, ...)
```

```
BASiCS_DetectHVG(Chain, ...)
```

Arguments

Chain	an object of class BASiCS_Chain
Task	Search for highly variable genes (Task="HVG") or lowly variable genes (Task="LVG").
PercentileThreshold	Threshold to detect a percentile of variable genes (must be a positive value, between 0 and 1). Default: PercentileThreshold = NULL.

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. Default: VarThreshold = NULL.
ProbThreshold	Optional parameter. Posterior probability threshold (must be a positive value, between 0 and 1). If EFDR = NULL, the posterior probability threshold for the test will be set to ProbThreshold
EpsilonThreshold	Threshold for residual overdispersion above which
EFDR	Target for expected false discovery rate related to HVG/LVG detection. If EFDR = NULL, EFDR calibration is not performed and the posterior probability threshold is set equal to ProbThreshold. Default EFDR = 0.10.
OrderVariable	Ordering variable for output. Possible values: 'GeneIndex', 'GeneName' and 'Prob'. Default ProbThreshold = 'Prob'
Plot	If Plot = TRUE error control and expression versus HVG/LVG probability plots are generated.
MinESS	The minimum effective sample size for a gene to be included in the HVG or LVG tests. This helps to remove genes with poor mixing from detection of HVGs/LVGs. Default is 100. If set to NA, genes are not checked for effective sample size the tests are performed.
...	Graphical parameters (see par).

Details

See vignette

Value

An object of class [BASiCS_ResultVG](#).

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)

## Highly and lowly variable genes detection based on residual overdispersion
## threshold
DetectHVG <- BASiCS_DetectHVG(ChainSCReg, EpsilonThreshold = log(2), Plot = TRUE)
DetectLVG <- BASiCS_DetectLVG(ChainSCReg, EpsilonThreshold = -log(2), 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,
  Parameter = NULL,
  Measure = c("ess", "geweke.diag"),
  na.rm = TRUE
)

BASiCS_diagHist(...)

```

Arguments

object	an object of class BASiCS_Summary
Parameter	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 Parameter = NULL.
Measure	Character scalar specifying the diagnostic measure to plot. Current options are effective sample size and the Geweke diagnostic criterion.
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

References

Geweke, J. Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In *_Bayesian Statistics 4_* (ed JM Bernardo, JO Berger, AP Dawid and AFM Smith). Clarendon Press, Oxford, 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, Parameter = "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,
  Parameter = "mu",
  Measure = c("ess", "geweke.diag"),
  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
Parameter	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 Parameter = 'mu'
Measure	Character scalar specifying the diagnostic measure to plot. Current options are effective sample size and the Geweke diagnostic criterion.
x, y	Optional MCMC parameter values to be plotted on the x or y axis, respectively. If neither is supplied, Parameter will be plotted on the x axis and effective sample size 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

See Also

[BASiCS_Chain](#)

Examples

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

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

BASiCS_DivideAndConquer

Run divide and conquer MCMC with BASiCS

Description

Performs MCMC inference on batches of data. Data is divided into `NSubsets` batches, and `BASiCS_MCMC` is run on each batch separately.

Usage

```

BASiCS_DivideAndConquer(
  Data,
  NSubsets = 5,
  SubsetBy = c("cell", "gene"),
  Alpha = 0.05,
  WithSpikes,
  Regression,
  BPPARAM = BiocParallel::bpparam(),
  PriorParam = BASiCS_PriorParam(Data, PriorMu = "EmpiricalBayes"),
  ...
)

```

Arguments

Data	SingleCellExperiment object
NSubsets	The number of batches to create and perform MCMC inference with.
SubsetBy	A character value specifying whether batches should consist of a subset of the cells in Data (when SubsetBy="cell") or a subset of the genes in Data (when SubsetBy="gene").
Alpha	A numeric value specifying the statistical significance level used to determine whether the average library size or average count are significantly different between batches.
WithSpikes, Regression, PriorParam	See BASiCS_MCMC .
BPPARAM	A BiocParallelParam instance.
...	Passed to BASiCS_MCMC . All arguments required by BASiCS_MCMC must be supplied here, for example N, Thin, Burn.

Details

Subsets are chosen such that the average library size (when partitioning by cells) or average count (when partitioning by genes) is not significantly different between batches, at a significance level Alpha.

Value

A list of [BASiCS_Chain](#) objects.

References

Simple, Scalable and Accurate Posterior Interval Estimation Cheng Li and Sanvesh Srivastava and David B. Dunson arXiv (2016)

Examples

```

bp <- BiocParallel::SnowParam()
Data <- BASiCS_MockSCE()
BASiCS_DivideAndConquer(
  Data,
  NSubsets = 2,
  SubsetBy = "gene",

```



```

N = 8,
Thin = 2,
Burn = 4,
WithSpikes = TRUE,
Regression = TRUE,
BPPARAM = bp
)

```

BASiCS_Draw

Generate a draw from the posterior of BASiCS using the generative model.

Description

BASiCS_Draw creates a simulated dataset from the posterior of a fitted model implemented in BASiCS.

Usage

```

BASiCS_Draw(
  Chain,
  BatchInfo = gsub(".*_Batch([0-9a-zA-Z])", "\\1",
    colnames(Chain@parameters[["nu"]])),
  N = sample(nrow(Chain@parameters[["nu"]]), 1)
)

```

Arguments

Chain	An object of class BASiCS_Chain .
BatchInfo	Vector of batch information from the <code>SingleCellExperiment</code> object used as input to <code>BASiCS_MCMC</code> .
N	The integer index for the draw to be used to sample from the posterior predictive distribution. If not supplied, a random value is chosen.

Value

An object of class [SingleCellExperiment](#), including synthetic data generated by the model implemented in BASiCS.

Author(s)

Alan O'Callaghan

References

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

Examples

```
data(ChainSC)
BASiCS_Draw(ChainSC)
```

```
data(ChainSC)
BASiCS_Draw(ChainSC)
```

BASiCS_EffectiveSize *Calculate effective sample size for BASiCS_Chain parameters*

Description

A function to calculate effective sample size [BASiCS_Chain](#) objects.

Usage

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

```
BASiCS_effectiveSize(...)
```

Arguments

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

Value

A vector with effective sample sizes for all the elements of Parameter

Examples

```
data(ChainSC)
BASiCS_EffectiveSize(ChainSC, Parameter = "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 used to generate a [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),]
```

BASiCS_LoadChain	<i>Loads pre-computed MCMC chains generated by the BASiCS_MCMC function</i>
------------------	---------------------------------------------------------------------------------------------

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,
  PriorParam = BASiCS_PriorParam(Data, PriorMu = "EmpiricalBayes"),
  SubsetBy = c("none", "gene", "cell"),
  NSubsets = 1,
  CombineMethod = c("pie", "consensus"),
  Weighting = c("naive", "n_weight", "inverse_variance"),
  Threads = getOption("Ncpus", default = 1L),
  BPPARAM = BiocParallel::bpparam(),
  ...
)

```

Arguments

Data	A SingleCellExperiment object. If <code>WithSpikes = TRUE</code> , 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 $N \geq \max(4, \text{Thin})$, N being a multiple of Thin.
Thin	Thinning period for the MCMC sampler. Use $\text{Thin} \geq 2$.
Burn	Burn-in period for the MCMC sampler. Use $\text{Burn} \geq 1$, $\text{Burn} < N$, Burn being a multiple of Thin.
Regression	If <code>Regression = TRUE</code> , 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 <code>Regression = TRUE</code> .
WithSpikes	If <code>WithSpikes = TRUE</code> , BASiCS will use reads from added spike-ins to estimate technical variability. If <code>WithSpikes = FALSE</code> , BASiCS depends on replicated experiments (batches) to estimate technical variability. In this case, please supply the <code>BatchInfo</code> vector in <code>colData(Data)</code> . Default: <code>WithSpikes = TRUE</code> .
PriorParam	List of prior parameters for BASiCS_MCMC. Should be created using BASiCS_PriorParam .
SubsetBy	Character value specifying whether a divide and conquer inference strategy should be used. When this is set to "gene", inference is performed on batches of genes separately, and when it is set to "cell", inference is performed on batches of cells separately. Posterior distributions are combined using posterior interval estimation (see Li et al., 2016).
NSubsets	If <code>SubsetBy="gene"</code> or <code>SubsetBy="cell"</code> , <code>NSubsets</code> specifies the number of batches to create and perform divide and conquer inference with.
CombineMethod	The method used to combine subposteriors if <code>SubsetBy</code> is set to "gene" or "cell". Options are "pie" corresponding to posterior interval estimation (see Li et al., 2016) or "consensus" (see Scott et al., 2016). Both of these methods use a form of weighted average to combine subposterior draws into the final posterior.

Weighting	The weighting method used in the weighted average chosen using <code>CombineMethod</code> . Available options are "naive" (unweighted), "n_weight" (weights are chosen based on the size of each partition) and "inverse_variance" (subposteriors are weighted based on the inverse of the variance of the subposterior for each parameter).
Threads	Integer specifying the number of threads to be used to parallelise parameter updates. Default value is the globally set "Ncpus" option, or 1 if this option is not set.
BPPARAM	A <code>BiocParallelParam</code> instance, used for divide and conquer inference.
...	Optional parameters.
	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 <code>StopAdapt >= 1</code> . Default: <code>StopAdapt = Burn</code> .
	StoreChains If <code>StoreChains = TRUE</code> , the generated <code>BASiCS_Chain</code> object is stored as a '.Rds' file (<code>RunName</code> argument used to index the file name). Default: <code>StoreChains = FALSE</code> .
	StoreAdapt If <code>StoreAdapt = TRUE</code> , trajectory of adaptive proposal variances (in log-scale) for all parameters is stored as a list in a '.Rds' file (<code>RunName</code> argument used to index file name). Default: <code>StoreAdapt = FALSE</code> .
	StoreDir Directory where output files are stored. Only required if <code>StoreChains = TRUE</code> and/or <code>StoreAdapt = TRUE</code> . Default: <code>StoreDir = getwd()</code> .
	RunName String used to index '.Rds' files storing chains and/or adaptive proposal variances.
	PrintProgress If <code>PrintProgress = FALSE</code> , 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: <code>mu0</code> , <code>delta0</code> , <code>phi0</code> , <code>s0</code> , <code>nu0</code> , <code>theta0</code> , <code>ls.mu0</code> , <code>ls.delta0</code> , <code>ls.phi0</code> , <code>ls.nu0</code> and <code>ls.theta0</code>
	GeneExponent/CellExponent Exponents applied to the prior for MCMC updates. Intended for use only when performing divide & conquer MCMC strategies.

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

Simple, Scalable and Accurate Posterior Interval Estimation Cheng Li and Sanvesh Srivastava and David B. Dunson arXiv (2016)

Bayes and Big Data: The Consensus Monte Carlo Algorithm Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George and Robert E. McCulloch International Journal of Management Science and Engineering Management (2016)

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 might 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 contrast 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 contrast 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_MockSCE

Create a mock [SingleCellExperiment](#) object.

Description

Creates a [SingleCellExperiment](#) object of Poisson-distributed approximating a homogeneous cell population.

Usage

```
BASiCS_MockSCE(NGenes = 100, NCells = 100, NSpikes = 20, WithBatch = TRUE)
```

Arguments

NGenes	Integer value specifying the number of genes that will be present in the output.
NCells	Integer value specifying the number of cells that will be present in the output.
NSpikes	Integer value specifying the number of spike-in genes that will be present in the output.
WithBatch	Logical value specifying whether a dummy BatchInfo is included in the output.

Value

A [SingleCellExperiment](#) object.

Examples

```
BASiCS_MockSCE()
```

BASiCS_PlotDE

*Produce plots assessing differential expression results***Description**

Produce plots assessing differential expression results

Usage

```

BASiCS_PlotDE(object, ...)

## S4 method for signature 'BASiCS_ResultsDE'
BASiCS_PlotDE(
  object,
  Plots = c("MA", "Volcano", "Grid"),
  Parameters = intersect(c("Mean", "Disp", "ResDisp"), names(object@Results)),
  MuX = TRUE,
  ...
)

## S4 method for signature 'BASiCS_ResultDE'
BASiCS_PlotDE(object, Plots = c("Grid", "MA", "Volcano"), Mu = NULL)

## S4 method for signature 'missing'
BASiCS_PlotDE(
  GroupLabel1,
  GroupLabel2,
  ProbThresholds = seq(0.5, 0.9995, by = 0.00025),
  Epsilon,
  EFDR,
  Table,
  Measure,
  EFDRgrid,
  EFNRgrid,
  ProbThreshold,
  Mu,
  Plots = c("Grid", "MA", "Volcano")
)

```

Arguments

object	A BASiCS_ResultsDE or BASiCS_ResultDE object.
...	Passed to methods.
Plots	Plots plot to produce? Options: "MA", "Volcano", "Grid".
Parameters	Character vector specifying the parameter(s) to produce plots for, Available options are "Mean", (mu, mean expression), "Disp" (delta, overdispersion) and "ResDisp" (epsilon, residual overdispersion).
MuX	Use Mu (mean expression across both chains) as the X-axis for all MA plots? Default: TRUE.
Mu, GroupLabel1, GroupLabel2, ProbThresholds, Epsilon, EFDR, Table, Measure, EFDRgrid, EFNRgrid, Prob	Internal arguments.

Value

A plot (possibly several combined using `plot_grid`).

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Alan O'Callaghan

Examples

```
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)

BASiCS_PlotDE(Test)
```

BASiCS_PlotOffset	<i>Visualise global offset in mean expression between two chains.</i>
-------------------	-----------------------------------------------------------------------

Description

Visualise global offset in mean expression between two BASiCS_Chain objects.

Usage

```
BASiCS_PlotOffset(
  Chain1,
  Chain2,
  Type = c("offset estimate", "before-after", "MAPlot"),
  GroupLabel1 = "Group 1",
  GroupLabel2 = "Group 2"
)
```

Arguments

Chain1, Chain2 [BASiCS_Chain](#) objects to be plotted.

Type The type of plot generated. "offset estimate" produces a boxplot of the offset alongside an estimate of the global offset. "before-after" produces MA plots of Mean expression against $\log_2(\text{fold-change})$ before and after offset correction. "MA plot" produces an MA plot of Mean expression against $\log_2(\text{fold-change})$.

GroupLabel1, GroupLabel2 Labels for Chain1 and Chain2 in the resulting plot(s).

Value

Plot objects.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Nils Eling <eling@ebi.ac.uk>

Alan O'Callaghan

Examples

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

BASiCS_PlotOffset(ChainSC, ChainRNA)
```

BASiCS_PlotVG	<i>Plots of HVG/LVG search.</i>
---------------	---------------------------------

Description

Plots of HVG/LVG search.

Usage

```
BASiCS_PlotVG(object, Plot = c("Grid", "VG"), ...)
```

Arguments

object	BASiCS_ResultVG object.
Plot	Character scalar specifying the type of plot to be made. Options are "Grid" and "VG".
...	Optional graphical parameters passed to <code>.VGPlot</code> (internal function).

Value

A plot.

Examples

```
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)

BASiCS_PlotVG(DetectHVG)
```

BASiCS_PriorParam *Prior parameters for BASiCS_MCMC*

Description

This is a convenience function to allow partial specification of prior parameters, and to ensure default parameters are consistent across usage within the package.

Usage

```
BASiCS_PriorParam(
  Data,
  k = 12,
  mu.mu = NULL,
  s2.mu = 0.5,
  s2.delta = 0.5,
  a.delta = 1,
  b.delta = 1,
  p.phi = rep(1, times = ncol(Data)),
  a.s = 1,
  b.s = 1,
  a.theta = 1,
  b.theta = 1,
  RBFMinMax = TRUE,
  FixLocations = !is.null(RBFLocations) | !is.na(MinGenesPerRBF),
  RBFLocations = NULL,
  MinGenesPerRBF = NA,
  variance = 1.2,
  m = numeric(k),
  V = diag(k),
  a.sigma2 = 2,
  b.sigma2 = 2,
  eta = 5,
  PriorMu = c("default", "EmpiricalBayes"),
  PriorDelta = c("log-normal", "gamma"),
  StochasticRef = TRUE,
  ConstrainProp = 0.2,
  GeneExponent = 1,
  CellExponent = 1
)
```

Arguments

Data	SingleCellExperiment object (required).
k	Number of regression terms, including k - 2 Gaussian radial basis functions (GRBFs).
mu.mu, s2.mu	Mean and variance parameters for lognormal prior on mu.
s2.delta	Variance parameter for lognormal prior on delta when PriorDelta="lognormal".
a.delta, b.delta	Parameters for gamma prior on delta when PriorDelta="gamma".

p.phi	Parameter for dirichlet prior on phi.
a.s, b.s	Parameters for gamma prior on s.
a.theta, b.theta	Parameters for gamma prior on theta.
RBFMinMax	Should GRBFs be placed at the minimum and maximum of $\log(\mu)$?
FixLocations	Should RBFLocations be fixed throughout MCMC, or adaptive during burn-in? By default this is FALSE, but it is set to TRUE if RBFLocations or MinGenesPerRBF are specified.
RBFLocations	Numeric vector specifying locations of GRBFs in units of $\log(\mu)$.
MinGenesPerRBF	Numeric scalar specifying the minimum number of genes for GRBFs to be retained. If fewer than MinGenesPerRBF genes have values of $\log(\mu)$ within the range of an RBF, it is removed. The range covered by each RBF is defined as centre of the RBF plus or minus half the distance between RBFs.
variance	Variance of the GRBFs.
m, V	Mean and (co)variance priors for regression coefficients.
a.sigma2, b.sigma2	Priors for inverse gamma prior on regression scale.
eta	Degrees of freedom for t distribution of regression errors.
PriorMu	Indicates if the original prior (PriorMu = 'default') or an empirical Bayes approach (PriorMu = 'EmpiricalBayes') will be assigned to gene-specific mean expression parameters.
PriorDelta	Scalar character specifying the prior type to use for delta overdispersion parameter. Options are "log-normal" (recommended) and "gamma" (used in Vallejos et al. (2015)).
StochasticRef	Logical scalar specifying whether the reference gene for the no-spikes version should be chosen randomly at MCMC iterations.
ConstrainProp	Proportion of genes to be considered as reference genes if StochasticRef=TRUE.
GeneExponent, CellExponent	Exponents for gene and cell-specific parameters. These should not be outside of divide and conquer MCMC applications.

Value

A list containing the prior hyper-parameters that are required to run the algorithm implemented in [BASiCS_MCMC](#).

Examples

```
BASiCS_PriorParam(makeExampleBASiCS_Data(), k = 12)
```

BASiCS_Result	<i>The BASiCS_Result class</i>
---------------	--------------------------------

Description

Container of results for a single test (HVG/LVG/DE). This should be an abstract class (but this is R so no) and shouldn't be directly instantiated. Defines a very small amount of common behaviour for [BASiCS_ResultDE](#) and [BASiCS_ResultVG](#).

Slots

Table Tabular results for each gene.

Name The name of the test performed (typically "Mean", "Disp" or "ResDisp")

ProbThreshold Posterior probability threshold used in differential test.

EFDR, EFNR Expected false discovery and expected false negative rates for differential test.

Extra Additional objects for class flexibility.

BASiCS_ResultDE	<i>The BASiCS_ResultDE class</i>
-----------------	----------------------------------

Description

Container of results for a single differential test.

Slots

Table Tabular results for each gene.

Name The name of the test performed (typically "Mean", "Disp" or "ResDisp")

GroupLabel1, GroupLabel2 Group labels.

ProbThreshold Posterior probability threshold used in differential test.

EFDR, EFNR Expected false discovery and expected false negative rates for differential test.

EFDRgrid, EFNRgrid Grid of EFDR and EFNR values calculated before thresholds were fixed.

Epsilon Minimum fold change or difference threshold.

Extra objects for class flexibility.

BASiCS_ResultsDE *The BASiCS_ResultsDE class*

Description

Results of BASiCS_TestDE

Slots

Results [BASiCS_ResultDE](#) objects

Chain1,Chain2 [BASiCS_Chain](#) objects.

GroupLabel1,GroupLabel2 Labels for Chain1 and Chain2

Offset Ratio between median of chains

RowData Annotation for genes

Extras Slot for extra information to be added later

BASiCS_ResultVG *The BASiCS_ResultVG class*

Description

Container of results for a single HVG/LVG test.

Slots

Method Character value detailing whether the test performed using a threshold directly on epsilon values (Method="Epsilon"), variance decomposition (Method="Variance") or percentiles of epsilon (Method="Percentile").

RowData Optional [DataFrame](#) containing additional information about genes used in the test.

EFDRgrid,EFNRgrid Grid of EFDR and EFNR values calculated before thresholds were fixed.

Threshold Threshold used to calculate tail posterior probabilities for the HVG or LVG decision rule.

ProbThresholds Probability thresholds used to calculate EFDRGrid and EFNRGrid.

ProbThreshold Posterior probability threshold used in the HVG/LVG decision rule.

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 whether the <code>smoothScatter</code> 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>

ReferencesEling et al (2018). Cell Systems <https://doi.org/10.1016/j.cels.2018.06.011>**Examples**

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

BASiCS_Sim	<i>Generates synthetic data according to the model implemented in BASiCS</i>
------------	------------------------------------------------------------------------------

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 <code>q.bio</code> , all elements must be positive numbers)
Mu_spikes	μ_i for all technical genes defined as true input molecules (vector of length <code>q-q.bio</code> , all elements must be positive numbers). If <code>Mu_spikes = NULL</code> , the generated data will not contain spike-ins. If <code>Phi = NULL</code> , <code>Mu_spikes</code> will be ignored. Default: <code>Mu_spikes = NULL</code> .
Delta	Gene-specific biological over-dispersion parameters δ_i , biological genes only (vector of length <code>q.bio</code> , all elements must be positive numbers)
Phi	Cell-specific mRNA content normalising parameters ϕ_j (vector of length <code>n</code> , all elements must be positive numbers and the sum of its elements must be equal to <code>n</code>). <code>Phi</code> must be set equal to <code>NULL</code> when generating data without spike-ins. If <code>Mu_spikes = NULL</code> , <code>Phi</code> will be ignored. Default: <code>Phi = NULL</code>
S	Cell-specific technical normalising parameters s_j (vector of length <code>n</code> , all elements must be positive numbers)
Theta	Technical variability parameter θ (must be positive). <code>Theta</code> can be a scalar (single batch of samples), or a vector (multiple batches of samples). If a value for <code>BatchInfo</code> is provided, the length of <code>Theta</code> must match the number of unique values in <code>BatchInfo</code> .
BatchInfo	Vector detailing which batch each cell should be simulated from. If spike-ins are not in use, the number of unique values contained in <code>BatchInfo</code> 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 biological 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))
altExp(Data)
rowData(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)
```

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 = c("GeneIndex", "GeneName", "Mu"),
  GroupLabel1 = "Group1",
  GroupLabel2 = "Group2",
  Plot = TRUE,
  PlotOffset = TRUE,
  PlotOffsetType = c("offset estimate", "before-after", "MA plot"),
  Offset = TRUE,
  EFDR_M = 0.05,
  EFDR_D = 0.05,
  EFDR_R = 0.05,
  GenesSelect = rep(TRUE, ncol(Chain1@parameters[["mu"]])),
  min.mean = 1,
  MinESS = 100,
  ...
)
```

Arguments

Chain1	an object of class <code>BASiCS_Chain</code> containing parameter estimates for the first group of cells
Chain2	an object of class <code>BASiCS_Chain</code> 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: $\text{EpsilonM} = \log_2(1.5)$ (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: $\text{EpsilonM} = \log_2(1.5)$ (i.e. 50% increase).
EpsilonR	Minimum distance threshold for detecting changes in residual over-dispersion (must be a positive real number). Default value: $\text{EpsilonR} = \log_2(1.5)/\log_2(\exp(1))$ (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 <code>EFDR_M = NULL</code> , the posterior probability threshold for the differential mean expression test will be set to <code>ProbThresholdM</code> . If a value for <code>EFDR_M</code> is provided, the posterior probability threshold is chosen to achieve an EFDR equal to <code>EFDR_M</code> and <code>ProbThresholdM</code> defines a minimum probability threshold for this calibration (this avoids low values of <code>ProbThresholdM</code> to be chosen by the EFDR calibration). Default value <code>ProbThresholdM = 2/3</code> , i.e. the probability of observing a \log_2 -FC above <code>EpsilonM</code> must be at least twice the probability of observing the complementary event (\log_2 -FC below <code>EpsilonM</code>).
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 <code>ProbThresholdM</code> , depending on the value provided for <code>EFDR_D</code> . Default value <code>ProbThresholdD = 2/3</code> .
ProbThresholdR	Optional parameter. Probability threshold for detecting changes in residual over-dispersion (must be a positive value, between 0 and 1). Same usage as <code>ProbThresholdM</code> , depending on the value provided for <code>EFDR_R</code> . Default value <code>ProbThresholdR = 2/3</code> .
OrderVariable	Ordering variable for output. Possible values: 'GeneIndex' (default), 'GeneName' and 'Mu' (mean expression).
GroupLabel1	Label assigned to reference group. Default: <code>GroupLabel1 = 'Group1'</code>
GroupLabel2	Label assigned to reference group. Default: <code>GroupLabel2 = 'Group2'</code>
Plot	If <code>Plot = TRUE</code> , MA and volcano plots are generated.
PlotOffset	If <code>Plot = TRUE</code> , the offset effect is visualised.
PlotOffsetType	See argument <code>Type</code> in <code>BASiCS_PlotOffset</code> for more information.
Offset	Optional argument to remove a fix offset effect (if not previously removed from the MCMC chains). Default: <code>Offset = TRUE</code> .
EFDR_M	Target for expected false discovery rate related to the comparison of means. If <code>EFDR_M = NULL</code> , EFDR calibration is not performed and the posterior probability threshold is set equal to <code>ProbThresholdM</code> . Default <code>EFDR_M = 0.05</code> .
EFDR_D	Target for expected false discovery rate related to the comparison of dispersions. If <code>EFDR_D = NULL</code> , EFDR calibration is not performed and the posterior probability threshold is set equal to <code>ProbThresholdD</code> . Default <code>EFDR_D = 0.05</code> .

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 = rep(TRUE, nGene) 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'.
MinESS	The minimum effective sample size for a gene to be included in the tests for differential expression. This helps to remove genes with poor mixing from differential expression tests. Default is 100. If set to NA, genes are not checked for effective sample size before differential expression tests are performed.
...	Optional parameters.

Value

BASiCS_TestDE returns an object of class `BASiCS_ResultsDE`

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(as.data.frame(Test, Parameter = "Mean"))

# Results for the differential over-dispersion test
# This only includes genes marked as 'NoDiff' in Test$TableMean
head(as.data.frame(Test, Parameter = "Disp"))

# 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
)

## Plotting the results of these tests
BASiCS_PlotDE(Test)

```

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 = c("BioVarGlobal", "GeneName", "TechVarGlobal", "ShotNoiseGlobal"),
  Plot = TRUE,
  main = "Overall variance decomposition",
  ylab = "% of variance",
  beside = FALSE,
  palette = "Set1",
  legend = c("Biological", "Technical", "Shot noise"),
  names.arg = if (nBatch == 1) "Overall" else c("Overall", paste("Batch",
    seq_len(nBatch)))
)

```

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	Plot title.
ylab	y axis label.
beside	If TRUE, bars are placed beside each other. If FALSE, bars are stacked.
palette	Palette to be passed to scale_fill_brewer to create a discrete colour mapping.
legend	Labels for variance components.
names.arg	X axis labels.

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_VarThresholdSearchVG(
  Chain,
  Task = c("HVG", "LVG"),
  VarThresholdsGrid,
  EFDR = 0.1,
  Progress = TRUE
)

BASiCS_VarThresholdSearchHVG(...)

BASiCS_VarThresholdSearchLVG(...)

```

Arguments

Chain	an object of class BASiCS_Chain
Task	See ?BASiCS_DetectVG.
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.
...	Passed to methods.

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	<i>Extract from the chain obtained for the Grun et al (2014) data: pool-and-split samples</i>
----------	-----------------------------------------------------------------------------------------------

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	<i>Extract from the chain obtained for the Grun et al (2014) data: pool-and-split samples (regression model)</i>
-------------	------------------------------------------------------------------------------------------------------------------

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	<i>Extract from the chain obtained for the Grun et al (2014) data: single-cell samples</i>
---------	--------------------------------------------------------------------------------------------

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	<i>Extract from the chain obtained for the Grun et al (2014) data: single-cell samples (regression model)</i>
------------	---------------------------------------------------------------------------------------------------------------

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.

dim	<i>'dim' method for BASiCS_Chain objects</i>
-----	----------------------------------------------

Description

Returns the dimensions (genes x cells) of a [BASiCS_Chain](#)

Usage

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

Arguments

x A [BASiCS_Chain](#) object.

Value

An vector of dimensions

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Examples

```
data(ChainSC)  
dimnames(ChainSC)
```

dimnames	<i>'dimnames' method for BASiCS_Chain objects</i>
----------	---------------------------------------------------

Description

Returns the dimension names (genes x cells) of a [BASiCS_Chain](#)

Usage

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

Arguments

x A [BASiCS_Chain](#) object.

Value

A list of two elements: (1) a vector of gene names and (2) a vector of cell names.

Author(s)

Catalina A. Vallejos <cnvallej@uc.cl>

Examples

```
data(ChainSC)
dimnames(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, Parameter = "mu")
```

Arguments

object	an object of class BASiCS_Chain
Parameter	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, Parameter = "mu")
```

Arguments

object an object of class [BASiCS_Summary](#)

Parameter 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)
```

format,BASiCS_ResultsDE-method

Methods for formatting [BASiCS_Result](#) and [BASiCS_ResultsDE](#) objects.

Description

Methods for formatting [BASiCS_Result](#) and [BASiCS_ResultsDE](#) objects.

Usage

```
## S4 method for signature 'BASiCS_ResultsDE'
format(x, Parameter, Filter = TRUE, ProbThreshold = NULL, ...)

## S4 method for signature 'BASiCS_ResultDE'
format(x, Filter = TRUE, ProbThreshold = NULL, ...)

## S4 method for signature 'BASiCS_ResultVG'
format(x, Filter = TRUE, ProbThreshold = NULL, ...)
```

Arguments

x	Object being subsetted.
Parameter	Character scalar indicating which of the BASiCS_Result should be formatted.
Filter	Logical scalar indicating whether results should be filtered based on differential expression or HVG/LVG status if ProbThreshold=NULL, or a probability threshold if ProbThreshold=NULL
ProbThreshold	Probability threshold to be used if Filter=TRUE
...	Passed to format .

Value

A data.frame.

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')
```

<code>newBASiCS_Chain</code>	<i>Creates a <code>BASiCS_Chain</code> object from pre-computed MCMC chains</i>
------------------------------	---------------------------------------------------------------------------------

Description

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

Usage

```
newBASiCS_Chain(parameters)
```

Arguments

<code>parameters</code>	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 <code>q.bio</code> columns, all elements must be positive numbers)
delta	MCMC chain for gene-specific biological over-dispersion parameters δ_i , biological genes only (matrix with <code>q.bio</code> columns, all elements must be positive numbers)
phi	MCMC chain for cell-specific mRNA content normalisation parameters ϕ_j (matrix with <code>n</code> columns, all elements must be positive numbers and the sum of its elements must be equal to <code>n</code>). This parameter is only used when spike-in genes are available.
s	MCMC chain for cell-specific technical normalisation parameters s_j (matrix with <code>n</code> columns, all elements must be positive numbers)
nu	MCMC chain for cell-specific random effects ν_j (matrix with <code>n</code> columns, all elements must be positive numbers)
theta	MCMC chain for technical over-dispersion parameter(s) θ (matrix, all elements must be positive, each column represents 1 batch)
beta	Only used for regression model. MCMC chain for regression coefficients (matrix with <code>k</code> columns, where <code>k</code> 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 variability) (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	<i>Creates a <code>SingleCellExperiment</code> object from a matrix of expression counts and experimental information about spike-in genes</i>
----------------	------------------------------------------------------------------------------------------------------------------------------------------------

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 analysed (including biological and technical spike-in genes). Gene names must be stored as <code>rownames(Counts)</code> .
Tech	Logical vector of length q. If <code>Tech = FALSE</code> the gene is biological; otherwise the gene is spike-in. Default value: <code>Tech = rep(FALSE, nrow(Counts))</code> .
SpikeInfo	<code>data.frame</code> whose first and second columns contain the gene names assigned to the spike-in genes (they must match the ones in <code>rownames(Counts)</code>) and the associated input number of molecules, respectively. If <code>SpikeInfo = NULL</code> , only the horizontal integration implementation (no spikes) can be run. Default value: <code>SpikeInfo = NULL</code> .
BatchInfo	Vector of length n whose elements indicate batch information. Not required if a single batch is present on the data. Default value: <code>BatchInfo = NULL</code> .
SpikeType	Character to indicate what type of spike-ins are in use. Default value: <code>SpikeType = "ERCC"</code> (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](#)

`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,
  Parameter = "mu",
  Gene = NULL,
  Cell = NULL,
  Batch = 1,
  RegressionTerm = NULL,
  ...
)
```

Arguments

<code>x</code>	A BASiCS_Chain object.
<code>Parameter</code>	Name of the slot to be used for the plot. Possible values: 'mu', 'delta', 'phi', 's', 'nu', 'theta', 'beta', 'sigma2' and 'epsilon'.
<code>Gene</code>	Specifies which gene is requested. Required only if <code>Parameter = 'mu'</code> or <code>'delta'</code>
<code>Cell</code>	Specifies which cell is requested. Required only if <code>Parameter = 'phi'</code> , <code>'s'</code> or <code>'nu'</code>
<code>Batch</code>	Specifies which batch is requested. Required only if <code>Parameter = 'theta'</code>
<code>RegressionTerm</code>	Specifies which regression coefficient is requested. Required only if <code>Parameter = 'beta'</code>
<code>...</code>	Unused.

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)
```

rowData, BASiCS_ResultsDE-method

rowData getter and setter for [BASiCS_ResultsDE](#) and [BASiCS_ResultVG](#) objects.

Description

rowData getter and setter for [BASiCS_ResultsDE](#) and [BASiCS_ResultVG](#) objects.

Usage

```
## S4 method for signature 'BASiCS_ResultsDE'
rowData(x)
```

```
## S4 replacement method for signature 'BASiCS_ResultsDE'
rowData(x) <- value
```

```
## S4 method for signature 'BASiCS_ResultVG'
rowData(x)
```

```
## S4 replacement method for signature 'BASiCS_ResultVG'
rowData(x) <- value
```

Arguments

- x [BASiCS_ResultVG](#) or [BASiCS_ResultsDE](#) object.
- value New rowData value for setter method.

Value

For the getter, a [DFrame](#). For setter, the modified x.

show,BASiCS_ResultDE-method

Accessors for the slots of a [BASiCS_ResultDE](#) object

Description

Accessors for the slots of a [BASiCS_ResultDE](#) object

Usage

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

Arguments

- object an object of class [BASiCS_ResultDE](#)

Value

Prints a summary of the properties of object.

See Also

[show](#)

Examples

```
help(BASiCS_MCMC)
```

show,BASiCS_ResultsDE-method

Accessors for the slots of a [BASiCS_ResultsDE](#) object

Description

Accessors for the slots of a [BASiCS_ResultsDE](#) object

Usage

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

Arguments

object an object of class [BASiCS_ResultsDE](#)

Value

Prints a summary of the properties of object.

See Also

[show](#)

Examples

```
help(BASiCS_MCMC)
```

show,BASiCS_ResultVG-method

Accessors for the slots of a [BASiCS_ResultVG](#) object

Description

Accessors for the slots of a [BASiCS_ResultVG](#) object

Usage

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

Arguments

object an object of class [BASiCS_ResultsDE](#)

Value

Prints a summary of the properties of object.

See Also[show](#)**Examples**

```
help(BASiCS_MCMC)
```

subset	<i>A 'subset' method for 'BASiCS_Chain' objects</i>
--------	-----------------------------------------------------

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 extracted.
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	<i>'Summary' method for BASiCS_Chain objects</i>
---------	--------------------------------------------------

Description

For each of the BASiCS parameters (see Vallejos et al 2015), Summary returns the corresponding posterior 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)
```

[,BASiCS_ResultsDE,ANY,ANY,ANY-method	<i>Methods for subsetting BASiCS_Result and BASiCS_ResultsDE objects.</i>
---------------------------------------	-----------------------------------------------------------------------------------------------------------

Description

Methods for subsetting [BASiCS_Result](#) and [BASiCS_ResultsDE](#) objects.

Usage

```
## S4 method for signature 'BASiCS_ResultsDE,ANY,ANY,ANY'  
x[i, j, drop = FALSE]
```

```
## S4 method for signature 'BASiCS_ResultsDE,ANY,ANY'  
x[[i]]
```

```
## S4 method for signature 'BASiCS_Result,ANY,ANY,ANY'  
x[i, j, drop = FALSE]
```

Arguments

x	Object being subsetted.
i	See ?`[, ?`[[`
j, drop	Ignored.

Value

An object of the same class as x.

Index

- * **datasets**
 - ChainRNA, [43](#)
 - ChainRNAREg, [43](#)
 - ChainSC, [44](#)
 - ChainSCReg, [44](#)
- .generateSubsets, [4](#)
- [,BASiCS_Result,ANY,ANY,ANY-method
 - ([,BASiCS_ResultsDE,ANY,ANY,ANY-method), [58](#)
- [,BASiCS_ResultsDE,ANY,ANY,ANY-method,
 - [58](#)
- [[,BASiCS_ResultsDE,ANY,ANY-method
 - ([,BASiCS_ResultsDE,ANY,ANY,ANY-method), [58](#)
- as.data.frame,BASiCS_ResultDE-method
 - (as.data.frame,BASiCS_ResultsDE-method), [4](#)
- as.data.frame,BASiCS_ResultsDE-method,
 - [4](#)
- as.data.frame,BASiCS_ResultVG-method
 - (as.data.frame,BASiCS_ResultsDE-method), [4](#)
- BASiCS-defunct, [5](#)
- BASiCS_Chain, [6, 7, 9–12, 14–18, 20, 21, 23, 27, 32, 33, 35, 38, 40–46, 49, 50, 52, 57, 58](#)
- BASiCS_Chain-class (BASiCS_Chain), [6](#)
- BASiCS_Chain-methods, [7](#)
- BASiCS_CorrectOffset, [8](#)
- BASiCS_D_TestDE (BASiCS-defunct), [5](#)
- BASiCS_DenoisedCounts, [9](#)
- BASiCS_DenoisedRates, [10](#)
- BASiCS_DetectHVG (BASiCS_DetectVG), [11](#)
- BASiCS_DetectHVG_LVG (BASiCS_DetectVG), [11](#)
- BASiCS_DetectHVLVG (BASiCS_DetectVG), [11](#)
- BASiCS_DetectLVG (BASiCS_DetectVG), [11](#)
- BASiCS_DetectVG, [11](#)
- BASiCS_DiagHist, [13](#)
- BASiCS_diagHist (BASiCS_DiagHist), [13](#)
- BASiCS_DiagPlot, [14](#)
- BASiCS_diagPlot (BASiCS_DiagPlot), [14](#)
- BASiCS_DivideAndConquer, [15](#)
- BASiCS_Draw, [17](#)
- BASiCS_EffectiveSize, [18](#)
- BASiCS_effectiveSize
 - (BASiCS_EffectiveSize), [18](#)
- BASiCS_Filter, [19](#)
- BASiCS_LoadChain, [20](#)
- BASiCS_MCMC, [6, 16, 20, 21, 21, 30](#)
- BASiCS_MockSCE, [25](#)
- BASiCS_PlotDE, [26](#)
- BASiCS_PlotDE,BASiCS_ResultDE-method
 - (BASiCS_PlotDE), [26](#)
- BASiCS_PlotDE,BASiCS_ResultsDE-method
 - (BASiCS_PlotDE), [26](#)
- BASiCS_PlotDE,missing-method
 - (BASiCS_PlotDE), [26](#)
- BASiCS_PlotOffset, [27, 38](#)
- BASiCS_PlotVG, [28](#)
- BASiCS_PriorParam, [22, 29](#)
- BASiCS_Result, [31, 47, 48, 58](#)
- BASiCS_Result-class (BASiCS_Result), [31](#)
- BASiCS_ResultDE, [5, 26, 31, 31, 32, 55](#)
- BASiCS_ResultDE-class
 - (BASiCS_ResultDE), [31](#)
- BASiCS_ResultsDE, [5, 26, 32, 39, 47, 54–56, 58](#)
- BASiCS_ResultsDE-class
 - (BASiCS_ResultsDE), [32](#)
- BASiCS_ResultVG, [5, 12, 28, 31, 32, 54–56](#)
- BASiCS_ResultVG-class
 - (BASiCS_ResultVG), [32](#)
- BASiCS_ShowFit, [33](#)
- BASiCS_showFit (BASiCS_ShowFit), [33](#)
- BASiCS_Sim, [34](#)
- BASiCS_Summary, [13, 15, 35, 36, 47, 53, 58](#)
- BASiCS_Summary-class (BASiCS_Summary), [35](#)
- BASiCS_Summary-methods, [36](#)
- BASiCS_TestDE, [5, 37](#)
- BASiCS_VarianceDecomp, [40](#)
- BASiCS_VarThresholdSearchHVG, [41](#)
- BASiCS_VarThresholdSearchHVG_LVG

- (BASiCS_VarThresholdSearchHVG), 41
- BASiCS_VarThresholdSearchLVG (BASiCS_VarThresholdSearchHVG), 41
- BASiCS_VarThresholdSearchVG (BASiCS_VarThresholdSearchHVG), 41
- BiocParallelParam, 16, 23
- ChainRNA, 43
- ChainRNAReg, 43
- ChainSC, 44
- ChainSCReg, 44
- data.frame, 41
- DataFrame, 32
- DFrame, 55
- dim, 45
- dim, BASiCS_Chain-method (dim), 45
- dimnames, 45
- dimnames, BASiCS_Chain-method (dimnames), 45
- displayChainBASiCS (displayChainBASiCS-BASiCS_Chain-method), 46
- displayChainBASiCS, BASiCS_Chain-method (displayChainBASiCS-BASiCS_Chain-method), 46
- displayChainBASiCS-BASiCS_Chain-method, 46
- displaySummaryBASiCS (displaySummaryBASiCS-BASiCS_Summary-method), 47
- displaySummaryBASiCS, BASiCS_Summary-method (displaySummaryBASiCS-BASiCS_Summary-method), 47
- displaySummaryBASiCS-BASiCS_Summary-method, 47
- effectiveSize, 13, 14
- format, 48
- format, BASiCS_ResultDE-method (format, BASiCS_ResultsDE-method), 47
- format, BASiCS_ResultsDE-method, 47
- format, BASiCS_ResultVG-method (format, BASiCS_ResultsDE-method), 47
- geom_hex, 15
- geweke.diag, 13
- HPDinterval, 58
- makeExampleBASiCS_Data, 48
- newBASiCS_Chain, 49
- newBASiCS_Data, 51
- par, 12, 33, 54
- plot, BASiCS_Chain, ANY-method (plot-BASiCS_Chain-method), 52
- plot, BASiCS_Chain-method (plot-BASiCS_Chain-method), 52
- plot, BASiCS_Summary, ANY-method (plot-BASiCS_Summary-method), 53
- plot, BASiCS_Summary-method, (plot-BASiCS_Summary-method), 53
- plot-BASiCS_Chain-method, 52
- plot-BASiCS_Summary-method, 53
- plot_grid, 27
- rowData, BASiCS_ResultsDE-method, 54
- rowData, BASiCS_ResultVG-method (rowData, BASiCS_ResultsDE-method), 54
- rowData<- , BASiCS_ResultsDE-method (rowData, BASiCS_ResultsDE-method), 54
- rowData<- , BASiCS_ResultVG-method (rowData, BASiCS_ResultsDE-method), 54
- scale_fill_brewer, 40
- show, 55–57
- show, BASiCS_Chain-method (BASiCS_Chain-methods), 7
- show, BASiCS_ResultDE-method, 55
- show, BASiCS_ResultsDE-method, 56
- show, BASiCS_ResultVG-method, 56
- show, BASiCS_Summary-method (BASiCS_Summary-methods), 36
- SingleCellExperiment, 9, 10, 17, 19, 22, 25, 29, 35, 48, 49, 51, 52
- subset, 57
- subset, BASiCS_Chain-method (subset), 57
- Summary, 58
- Summary, BASiCS_Chain-method (Summary), 58
- updateObject, 7
- updateObject, BASiCS_Chain-method (BASiCS_Chain-methods), 7