Package 'SCRIP'

August 25, 2021

Type Package

Title SCRIP: an accurate simulator for single-cell RNA sequencing data

Version 1.2.1

Date 2021-08-17

Author Fei Qin, Xizhi Luo, Feifei Xiao, Guoshuai Cai

Maintainer Fei Qin <`fqin@email.sc.edu`>

Description SCRIP provides a comprehensive scheme that is capable of simulating scRNA-seq data for various parameters of Biological Coefficient of Variation (BCV), busting kinetics, differential expression (DE), cell or sample groups, cell trajectory, batch effect and other experimental designs. SCRIP proposed and compared two frameworks with Gamma-Poisson and Beta-Gamma-Poisson models for simulating scRNA-seq data.

License GPL-3 + file LICENSE

LazyData TRUE

Depends R (¿= 4.0)

Imports splatter,
S4Vectors,
ggplot2,
methods,
SummarizedExperiment,
SingleCellExperiment,
mgcv,
edgeR,
knitr,
BiocManager,
BiocGenerics,
Seurat,
akima,
car,
crayon,
fitdistrplus,
checkmate (¿= 2.0.0),
SCALE

URL `https://https://github.com/thecailab/SCRIP`

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

R topics documented:

`bridge` *Brownian bridge*

Description

Calculate a smoothed Brownian bridge between two points. A Brownian bridge is a random walk with fixed end points.

Usage

```
bridge(x = 0, y = 0, N = 5, n = 100, sigma.fac = 0.8)
```

Arguments

`x`	starting value.
`y`	end value.
`N`	number of steps in random walk.
`n`	number of points in smoothed bridge.
`sigma.fac`	multiplier specifying how extreme each step can be.

Value

Vector of length n following a path from x to y.

`bringItemsForward` *Bring items forward*

Description

Move selected items to the start of a list.

Usage

```
bringItemsForward(ll, items)
```

Arguments

`ll`	list to adjust item order.
`items`	vector of items to bring to the front. Any not in the list will be ignored.

Value

list with selected items first

co.var *Calculate coefficient of variation*

Description

Implementation of the coefficient of variation

Usage

```
co.var(x)
```

Arguments

`x`	vector of values.

Value

Value of coefficient of variation for vector

expandParams *Expand parameters*

Description

Expand the parameters that can be vectors so that they are the same length as the number of groups.

Usage

```
expandParams(object, ...)

## S4 method for signature 'SplatParams'
expandParams(object)
```

Arguments

`object`	object to expand.
`...`	additional arguments.

Value

Expanded object.

`getLNormFactors` *Get log-normal factors*

Description

Randomly generate multiplication factors from a log-normal distribution.

Usage

```
getLNormFactors(n.facs, sel.prob, neg.prob, fac.loc, fac.scale)
```

Arguments

`n.facs`	Number of factors to generate.
`sel.prob`	Probability that a factor will be selected to be different from 1.
`neg.prob`	Probability that a selected factor is less than one.
`fac.loc`	Location parameter for the log-normal distribution.
`fac.scale`	Scale factor for the log-normal distribution.

Value

Vector containing generated factors.

`getParam` *Get a parameter*

Description

Accessor function for getting parameter values.

Usage

```
getParam(object, name)

## S4 method for signature 'Params'
getParam(object, name)
```

Arguments

`object`	object to get parameter from.
`name`	name of the parameter to get.

Value

The extracted parameter value

`getParams` *Get parameters*

Description

Get multiple parameter values from a Params object.

Usage

```
getParams(params, names)
```

Arguments

`params`	Params object to get values from.
`names`	vector of names of the parameters to get.

Value

List with the values of the selected parameters.

`getPathOrder` *Get path order*

Description

Identify the correct order to process paths so that preceding paths have already been simulated.

Usage

```
getPathOrder(path.from)
```

Arguments

`path.from`	vector giving the path endpoints that each path originates from.

Value

Vector giving the order to process paths in.

`logistic` *Logistic function*

Description

Implementation of the logistic function

Usage

```
logistic(x, x0, k)
```

Arguments

`x`	value to apply the function to.
`x0`	midpoint parameter. Gives the centre of the function.
`k`	shape parameter. Gives the slope of the function.

Value

Value of logistic function with given parameters

`newParams` *New Params*

Description

Create a new Params object. Functions exist for each of the different Params subtypes.

Usage

```
newSplatParams(...)
```

Arguments

`...`	additional parameters passed to `setParams`.

Value

New Params object.

`Params` *The Params virtual class*

Description

Virtual S4 class that all other Params classes inherit from.

Parameters

The Params class defines the following parameters:

`nGenes`	The number of genes to simulate.
`nCells`	The number of cells to simulate.
`[seed]`	Seed to use for generating random numbers.

The parameters not shown in brackets can be estimated from real data.

`rbindMatched` *Bind rows (matched)*

Description

Bind the rows of two data frames, keeping only the columns that are common to both.

Usage

```
rbindMatched(df1, df2)
```

Arguments

`df1`	first data.frame to bind.
`df2`	second data.frame to bind.

Value

data.frame containing rows from `df1` and `df2` but only common columns.

`SCRIPsimBatchCellMeans` *Simulate batch means*

Description

Simulate a mean for each gene in each cell incorporating batch effect factors.

Usage

```
SCRIPsimBatchCellMeans(sim, params)
```

Arguments

`sim`	SingleCellExperiment to add batch means to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with simulated batch means.

`SCRIPsimBatchEffects` *Simulate batch effects*

Description

Simulate batch effects. Batch effect factors for each batch are produced using `getLNormFactors` and these are added along with updated means for each batch.

Usage

```
SCRIPsimBatchEffects(sim, params)
```

Arguments

`sim`	SingleCellExperiment to add batch effects to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with simulated batch effects.

`SCRIPsimBCVMeans` *Simulate BCV means Simulate means for each gene in each cell that are adjusted to follow a mean-variance trend using Biological Coefficient of Variation taken from and inverse gamma distribution.*

Description

Simulate BCV means Simulate means for each gene in each cell that are adjusted to follow a mean-variance trend using Biological Coefficient of Variation taken from and inverse gamma distribution.

Usage

```
SCRIPsimBCVMeans(data, sim, params)
```

Arguments

`data`	data are used to fit the mean-BCV trend for simulation
`sim`	SingleCellExperiment to add BCV means to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with simulated BCV means.

`SCRIPsimDE` *Simulate group differential expression*

Description

Simulate differential expression. Differential expression factors for each group are produced using `getLNormFactors` and these are added along with updated means for each group. For paths care is taken to make sure they are simulated in the correct order.

Usage

```
SCRIPsimGroupDE(sim, params)
```

Arguments

`sim`	SingleCellExperiment to add differential expression to.
`params`	splatParams object with simulation parameters.

Value

SingleCellExperiment with simulated differential expression.

`SCRIPsimDropout` *Simulate dropout*

Description

A logistic function is used to form a relationship between the expression level of a gene and the probability of dropout, giving a probability for each gene in each cell. These probabilities are used in a Bernoulli distribution to decide which counts should be dropped.

Usage

```
SCRIPsimDropout(sim, params)
```

Arguments

`sim`	SingleCellExperiment to add dropout to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with simulated dropout and observed counts.

`SCRIPsimGeneMeans` *Simulate gene means*

Description

Simulate gene means from a gamma distribution. Also simulates outlier expression factors. Genes with an outlier factor not equal to 1 are replaced with the median mean expression multiplied by the outlier factor.

Usage

```
SCRIPsimGeneMeans(data, sim, params)
```

Arguments

`data`	raw dataset.
`sim`	SingleCellExperiment to add gene means to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with simulated gene means.

`SCRIPsimLibSizes` *Simulate library sizes*

Description

Simulate expected library sizes. Typically a log-normal distribution is used but there is also the option to use a normal distribution. In this case any negative values are set to half the minimum non-zero value.

Usage

```
SCRIPsimLibSizes(sim, params, libsize)
```

Arguments

`sim`	SingleCellExperiment to add library size to.
`params`	SplatParams object with simulation parameters.
`libsize`	Provide the library size directly instread of using parameters to estimate

Value

SingleCellExperiment with simulated library sizes.

`SCRIPsimSingleCellMeans` *Simulate cell means*

Description

Simulate a gene by cell matrix giving the mean expression for each gene in each cell. Cells start with the mean expression for the group they belong to (when simulating groups) or cells are assigned the mean expression from a random position on the appropriate path (when simulating paths). The selected means are adjusted for each cell's expected library size.

Usage

```
SCRIPsimSingleCellMeans(sim, params)
```

Arguments

`sim`	SingleCellExperiment to add cell means to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with added cell means.

`SCRIPsimTrueCounts` *Simulate true counts*

Description

Simulate a true counts matrix. Counts are simulated from a poisson distribution where Each gene in each cell has it's own mean based on the group (or path position), expected library size and BCV.

Usage

```
SCRIPsimTrueCounts(sim, params)
```

Arguments

`sim`	SingleCellExperiment to add true counts to.
`params`	SplatParams object with simulation parameters.

Value

SingleCellExperiment with simulated true counts.

`SCRIPsimu` *SCRIP simulation*

Description

Simulate count data for single cell RNA-sequencing using SCIRP method

Usage

```
SCRIPsimu(
  data,
  params,
  method = "single",
  base_allcellmeans_SC = NULL,
  pre.bcv.df = NULL,
  libsize = NULL,
  bcv.shrink = 1,
  Dropout_rate = NULL,
  mode = "GP-trendedBCV",
  de.prob = NULL,
  de.downProb = NULL,
  de.facLoc = NULL,
  de.facScale = NULL,
  path.skew = NULL,
  batch.facLoc = NULL,
  batch.facScale = NULL,
  path.nSteps = NULL,
  ...
)
```

Arguments

`data`	data matrix required to fit the mean-BCV trend for simulation
`params`	SplatParams object containing parameters for the simulation
`method`	"single", "groups" or "paths"
`base_allcellmeans_SC`	base mean vector provided to help setting DE analysis
`pre.bcv.df`	BCV.df enables us to change the variation of BCV values
`libsize`	library size can be provided directly
`bcv.shrink`	factor to control the BCV levels
`Dropout_rate`	factor to control the dropout rate directly
`mode`	"GP-commonBCV", "BP-commonBCV", "BP", "BGP-commonBCV" and "BGP-trendedBCV"
`de.prob`	the proportion of DE genes
`de.downProb`	the proportion of down-regulated DE genes
`de.facLoc`	DE location factor
`de.facScale`	DE scale factor
`path.skew`	Controls how likely cells are from the start or end point of the path
`batch.facLoc`	DE location factor in batch
`batch.facScale`	DE scale factor in batch
`path.nSteps`	number of steps between the start point and end point for each path
`...`	Other parameters #SCRIP simulation function

`setParam` *Set a parameter*

Description

Function for setting parameter values.

Usage

```
setParam(object, name, value)

## S4 method for signature 'Params'
setParam(object, name, value)

## S4 method for signature 'SplatParams'
setParam(object, name, value)
```

Arguments

`object`	object to set parameter in.
`name`	name of the parameter to set.
`value`	value to set the parameter to.

Value

Object with new parameter value.

`setParams` *Set parameters*

Description

Set multiple parameters in a Params object.

Usage

```
setParams(object, update = NULL, ...)

## S4 method for signature 'Params'
setParams(object, update = NULL, ...)

## S4 method for signature 'SplatParams'
setParams(object, update = NULL, ...)
```

Arguments

`object`	Params object to set parameters in.
`update`	list of parameters to set where `names(update)` are the names of the parameters to set and the items in the list are values.
`...`	additional parameters to set. These are combined with any parameters specified in `update`.

Details

Each parameter is set by a call to `setParam`. If the same parameter is specified multiple times it will be set multiple times. Parameters can be specified using a list via `update` (useful when collecting parameter values in some way) or individually (useful when setting them manually), see examples.

Value

Params object with updated values.

`setParamsUnchecked` *Set parameters UNCHECKED*

Description

Set multiple parameters in a Params object.

Usage

```
setParamsUnchecked(params, update = NULL, ...)
```

Arguments

`params`	Params object to set parameters in.
`update`	list of parameters to set where `names(update)` are the names of the parameters to set and the items in the list are values.
`...`	additional parameters to set. These are combined with any parameters specified in `update`.

Details

Each parameter is set by a call to `setParam`. If the same parameter is specified multiple times it will be set multiple times. Parameters can be specified using a list via `update` (useful when collecting parameter values in some way) or individually (useful when setting them manually), see examples. THE FINAL OBJECT IS NOT CHECKED FOR VALIDITY!

Value

Params object with updated values.

`setParamUnchecked` *Set a parameter UNCHECKED*

Description

Function for setting parameter values. THE OUTPUT IS NOT CHECKED FOR VALIDITY!

Usage

```
setParamUnchecked(object, name, value)

## S4 method for signature 'Params'
setParamUnchecked(object, name, value)
```

Arguments

`object`	object to set parameter in.
`name`	name of the parameter to set.
`value`	value to set the parameter to.

Value

Object with new parameter value.

`showDFs` *Show data.frame*

Description

Function used for pretty printing data.frame parameters.

Usage

```
showDFs(dfs, not.default)
```

Arguments

`dfs`	list of data.frames to show.
`not.default`	logical vector giving which have changed from the default.

Value

Print data.frame parameters

`showPP` *Show pretty print*

Description

Function used for pretty printing params object.

Usage

```
showPP(params, pp)
```

Arguments

`params`	object to show.
`pp`	list specifying how the object should be displayed.

Value

Print params object to console

`showValues` *Show values*

Description

Function used for pretty printing scalar or vector parameters.

Usage

```
showValues(values, not.default)
```

Arguments

`values`	list of values to show.
`not.default`	logical vector giving which have changed from the default.

Value

Print values

`SimpleParams` *The SimpleParams class*

Description

S4 class that holds parameters for the simple simulation.

Parameters

The simple simulation uses the following parameters:

`nGenes` The number of genes to simulate.

`nCells` The number of cells to simulate.

`[seed]` Seed to use for generating random numbers.

`mean.shape` The shape parameter for the mean gamma distribution.

`mean.rate` The rate parameter for the mean gamma distribution.

`[count.disp]` The dispersion parameter for the counts negative binomial distribution.

The parameters not shown in brackets can be estimated from real data using `simpleEstimate`. For details of the simple simulation see `simpleSimulate`.

`simu.VEGs` *SCRIP simulation for clustering analysis*

Description

Simulate count data for clustering analysis by preserving variably expressed genes

Usage

```
simu.VEGs(
  counts.matrix,
  params = params,
  base_allcellmeans,
  mode = "GP-trendedBCV",
  nCells,
  nfeatures = 1000
)
```

Arguments

`counts.matrix`	data matrix required for simulation
`params`	SplatParams object containing parameters for the simulation
`base_allcellmeans`	base cell means specified directly for simulating counts
`mode`	"GP-commonBCV", "BP-commonBCV", "BP", "BGP-commonBCV" and "BGP-trendedBCV"
`nCells`	number of cells simulated
`nfeatures`	parameter required for FinalVariable function in Seurat package #SCRIP group simulation function for clustering analysis

`simu_cluster` *SCRIP simulation for clustering analysis with multiple cell types*

Description

Simulate count data for clustering analysis by preserving variably expressed genes with multiple cell types

Usage

```
simu_cluster(expre_data, pheno_data, CTlist, mode, nfeatures, seed = 2021)
```

Arguments

`expre_data`	data matrix required for simulation
`pheno_data`	phenotype data information
`CTlist`	cell types used for simulation
`mode`	"GP-commonBCV", "BP-commonBCV", "BP", "BGP-commonBCV" and "BGP-trendedBCV"
`nfeatures`	parameter required for FinalVariable function in Seurat package
`seed`	seed used for simulation #SCRIP group simulation function for clustering analysis with multiple cell types

---

`simu_DE` *SCRIP simulation for differential expression*

Description

Simulate count data for differential expression analysis using SCRIP

Usage

```
simu_DE(
  expre_data,
  params,
  nGenes = NULL,
  nDE,
  ncells = NULL,
  FC,
  Dropout_rate = NULL,
  libsize = NULL,
  pre.bcv.df = NULL,
  bcv.shrink = 1,
  seed = 2021
)
```

Arguments

`expre_data`	data matrix required for simulation
`params`	SplatParams object containing parameters for the simulation
`nGenes`	number of genes simulated
`nDE`	number of differentially expressed genes simulated
`ncells`	number of cells simulated
`FC`	fold change rate simulated between two groups
`Dropout_rate`	factor to control the dropout rate directly
`libsize`	library size used for simulation
`pre.bcv.df`	BCV.df enables us to change the variation of BCV values
`bcv.shrink`	factor to control the BCV levels
`seed`	seed for simulation #SCRIP group simulation for differential expression analysis

`SplatParams` *The SplatParams class*

Description

S4 class that holds parameters for the Splat simulation.

Parameters

The Splat simulation requires the following parameters:

- `nGenes` The number of genes to simulate.
- `nCells` The number of cells to simulate.
- `[seed]` Seed to use for generating random numbers.
- ***Batch parameters*** `[nBatches]` The number of batches to simulate.
  - `[batchCells]` Vector giving the number of cells in each batch.
  - `[batch.facLoc]` Location (meanlog) parameter for the batch effect factor log-normal distribution. Can be a vector.
  - `[batch.facScale]` Scale (sdlog) parameter for the batch effect factor log-normal distribution. Can be a vector.
  - `[batch.rmEffect]` Logical, removes the batch effect and continues with the simulation when TRUE. This allows the user to test batch removal algorithms without having to calculate the new expected cell means with batch removed.
- ***Mean parameters*** `mean.shape` Shape parameter for the mean gamma distribution.
  - `mean.rate` Rate parameter for the mean gamma distribution.
- ***Library size parameters*** `lib.loc` Location (meanlog) parameter for the library size log-normal distribution, or mean parameter if a normal distribution is used.
  - `lib.scale` Scale (sdlog) parameter for the library size log-normal distribution, or sd parameter if a normal distribution is used.
  - `lib.norm` Logical. Whether to use a normal distribution for library sizes instead of a log-normal.
- ***Expression outlier parameters*** `out.prob` Probability that a gene is an expression outlier.
  - `out.facLoc` Location (meanlog) parameter for the expression outlier factor log-normal distribution.
  - `out.facScale` Scale (sdlog) parameter for the expression outlier factor log-normal distribution.
- ***Group parameters*** `[nGroups]` The number of groups or paths to simulate.
  - `[group.prob]` Probability that a cell comes from a group.
- ***Differential expression parameters*** `[de.prob]` Probability that a gene is differentially expressed in a group. Can be a vector.
  - `[de.downProb]` Probability that a differentially expressed gene is down-regulated. Can be a vector.
  - `[de.facLoc]` Location (meanlog) parameter for the differential expression factor log-normal distribution. Can be a vector.
  - `[de.facScale]` Scale (sdlog) parameter for the differential expression factor log-normal distribution. Can be a vector.

Biological Coefficient of Variation parameters `bcv.common` Underlying common dispersion across all genes.

`bcv.df` Degrees of Freedom for the BCV inverse chi-squared distribution.

Dropout parameters `dropout.type` The type of dropout to simulate. "none" indicates no dropout, "experiment" is global dropout using the same parameters for every cell, "batch" uses the same parameters for every cell in each batch, "group" uses the same parameters for every cell in each groups and "cell" uses a different set of parameters for each cell.

`dropout.mid` Midpoint parameter for the dropout logistic function.

`dropout.shape` Shape parameter for the dropout logistic function.

Differentiation path parameters `[path.from]` Vector giving the originating point of each path. This allows path structure such as a cell type which differentiates into an intermediate cell type that then differentiates into two mature cell types. A path structure of this form would have a "from" parameter of c(0, 1, 1) (where 0 is the origin). If no vector is given all paths will start at the origin.

`[path.nSteps]` Vector giving the number of steps to simulate along each path. If a single value is given it will be applied to all paths. This parameter was previously called `path.length`.

`[path.skew]` Vector giving the skew of each path. Values closer to 1 will give more cells towards the starting population, values closer to 0 will give more cells towards the final population. If a single value is given it will be applied to all paths.

`[path.nonlinearProb]` Probability that a gene follows a non-linear path along the differentiation path. This allows more complex gene patterns such as a gene being equally expressed at the beginning an end of a path but lowly expressed in the middle.

`[path.sigmaFac]` Sigma factor for non-linear gene paths. A higher value will result in more extreme non-linear variations along a path.

The parameters not shown in brackets can be estimated from real data using `splatEstimate`. For details of the Splat simulation see `splatSimulate`.

`winsorize` *Winsorize vector*

Description

Set outliers in a numeric vector to a specified percentile.

Usage

```
winsorize(x, q)
```

Arguments

`x`	Numeric vector to winsorize
`q`	Percentile to set from each end

Value

Winsorized numeric vector

Index