

# Package ‘scMultiSim’

November 29, 2024

**Title** Simulation of Multi-Modality Single Cell Data Guided By Gene  
Regulatory Networks and Cell-Cell Interactions

**Version** 1.2.0

## Description

scMultiSim simulates paired single cell RNA-seq, single cell ATAC-seq and RNA velocity data, while incorporating mechanisms of gene regulatory networks, chromatin accessibility and cell-cell interactions. It allows users to tune various parameters controlling the amount of each biological factor, variation of gene-expression levels, the influence of chromatin accessibility on RNA sequence data, and so on. It can be used to benchmark various computational methods for single cell multi-omics data, and to assist in experimental design of wet-lab experiments.

**License** Artistic-2.0

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Depends** R (>= 4.4.0)

**Imports** foreach, rlang, dplyr, ggplot2, Rtsne, ape, MASS, matrixStats,  
phytools, KernelKnn, gplots, zeallot, crayon, assertthat,  
igraph, methods, grDevices, graphics, stats, utils, markdown,  
SummarizedExperiment, BiocParallel

**Suggests** knitr, rmarkdown, roxygen2, shiny, testthat (>= 3.0.0)

**biocViews** SingleCell, Transcriptomics, GeneExpression, Sequencing,  
ExperimentalDesign

**VignetteBuilder** knitr

**Roxygen** list(markdown = TRUE)

**BugReports** <https://github.com/ZhangLabGT/scMultiSim/issues>

**URL** <https://zhanglabgt.github.io/scMultiSim/>

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/scMultiSim>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** 92f9425

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-11-28

**Author** Hechen Li [aut, cre] (<<https://orcid.org/0000-0003-4907-429X>>),  
 Xiuwei Zhang [aut],  
 Ziqi Zhang [aut],  
 Michael Squires [aut]

**Maintainer** Hechen Li <hli691@gatech.edu>

## Contents

.amplifyOneCell . . . . .	3
.calAmpBias . . . . .	4
.continuousCIF . . . . .	4
.divideBatchesImpl . . . . .	5
.expandToBinary . . . . .	6
.getCountCorrMatrix . . . . .	6
.getParams . . . . .	7
.normalizeGRNParams . . . . .	7
.rnormTrunc . . . . .	8
add_expr_noise . . . . .	8
add_outliers . . . . .	9
cci_cell_type_params . . . . .	9
dens_nonzero . . . . .	10
divide_batches . . . . .	11
gene_corr_cci . . . . .	11
gene_corr_regulator . . . . .	12
gene_len_pool . . . . .	13
gen_lbranch . . . . .	13
gen_clutter . . . . .	14
Get_lregion_ATAC_correlation . . . . .	15
Get_ATAC_correlation . . . . .	16
GRN_params_100 . . . . .	16
GRN_params_1139 . . . . .	17
len2nfrag . . . . .	17
match_params . . . . .	18
OP . . . . .	18
Phyla1 . . . . .	19
Phyla3 . . . . .	19
Phyla5 . . . . .	20
plot_cell_loc . . . . .	20
plot_gene_module_cor_heatmap . . . . .	21
plot_grid . . . . .	22
plot_grn . . . . .	22
plot_phyla . . . . .	23
plot_rna_velocity . . . . .	23
plot_tsne . . . . .	24
run_shiny . . . . .	25
SampleDen . . . . .	25
scmultisim_help . . . . .	26
sim_example . . . . .	26
sim_example_spatial . . . . .	27
sim_true_counts . . . . .	27
spatialGrid-class . . . . .	29

<code>.amplifyOneCell</code>	3
True2ObservedATAC	29
True2ObservedCounts	30

**Index** **32**

`.amplifyOneCell`      *This function simulates the amplification, library prep, and the sequencing processes.*

**Description**

This function simulates the amplification, library prep, and the sequencing processes.

**Usage**

```
.amplifyOneCell(
  true_counts_1cell,
  protocol,
  rate_2cap,
  gene_len,
  amp_bias,
  rate_2PCR,
  nPCR1,
  nPCR2,
  LinearAmp,
  LinearAmp_coef,
  N_molecules_SEQ
)
```

**Arguments**

<code>true_counts_1cell</code>	the true transcript counts for one cell (one vector)
<code>protocol</code>	a string, can be "nonUMI" or "UMI"
<code>rate_2cap</code>	the capture efficiency for this cell
<code>gene_len</code>	gene lengths for the genes/transcripts, sampled from real human transcript length
<code>amp_bias</code>	amplification bias for each gene, a vector of length ngenes
<code>rate_2PCR</code>	PCR efficiency, usually very high
<code>nPCR1</code>	the number of PCR cycles
<code>nPCR2</code>	the number of PCR cycles
<code>LinearAmp</code>	if linear amplification is used for pre-amplification step, default is FALSE
<code>LinearAmp_coef</code>	the coefficient of linear amplification, that is, how many times each molecule is amplified by
<code>N_molecules_SEQ</code>	number of molecules sent for sequencing; sequencing depth

**Value**

read counts (if protocol="nonUMI") or UMI counts (if protocol="UMI")

---

.calAmpBias	<i>Simulate technical biases</i>
-------------	----------------------------------

---

**Description**

Simulate technical biases

**Usage**

```
.calAmpBias(lenslope, nbins, gene_len, amp_bias_limit)
```

**Arguments**

lenslope	amount of length bias. This value should be less than $2 * \text{amp\_bias\_limit}[2] / (\text{nbins} - 1)$
nbins	number of bins for gene length
gene_len	transcript length of each gene
amp_bias_limit	range of amplification bias for each gene, a vector of length ngenes

**Value**

a vector

---

.continuousCIF	<i>Generates cifs for cells sampled along the trajectory of cell development</i>
----------------	--

---

**Description**

Generates cifs for cells sampled along the trajectory of cell development

**Usage**

```
.continuousCIF(
  seed,
  N,
  options,
  ncell_key = "cell",
  is_spatial = FALSE,
  spatial_params = NULL,
  .plot = FALSE,
  .plot.name = "cont_cif.pdf"
)
```

**Arguments**

seed	random seed
N	the number list
options	the option list
ncell_key	the key for the number of cells in N
is_spatial	return a list of cifs for spatial
spatial_params	the spatial parameters
.plot	save the CIF plot
.plot.name	plot name

**Value**

a list containing the cif and meta data

---

.divideBatchesImpl     *Divide the observed counts into multiple batches by adding batch effect to each batch*

---

**Description**

Divide the observed counts into multiple batches by adding batch effect to each batch

**Usage**

```
.divideBatchesImpl(  
  counts,  
  meta_cell,  
  nbatch,  
  batch_effect_size = 1,  
  randseed = 0  
)
```

**Arguments**

counts	gene cell matrix
meta_cell	the meta information related to cells, will be combined with technical cell level information and returned
nbatch	number of batches
batch_effect_size	amount of batch effects. Larger values result in bigger differences between batches. Default is 1.
randseed	random seed

**Value**

a list with two elements: counts and meta\_cell

---

<code>.expandToBinary</code>	<i>expand transcript counts to a vector of binaries of the same length of as the number of transcripts</i>
------------------------------	--

---

**Description**

expand transcript counts to a vector of binaries of the same length of as the number of transcripts

**Usage**

```
.expandToBinary(true_counts_1cell)
```

**Arguments**

`true_counts_1cell`  
number of transcript in one cell

**Value**

a list of two vectors, the first vector is a vector of 1s, the second vector is the index of transcripts

---

<code>.getCountCorrMatrix</code>	<i>This function finds the correlation between every pair of genes</i>
----------------------------------	--

---

**Description**

This function finds the correlation between every pair of genes

**Usage**

```
.getCountCorrMatrix(counts)
```

**Arguments**

`counts`            rna seq counts

**Value**

the correlation matrix

---

.getParams                      *Get Kinetic Parameters for all cells and genes*

---

**Description**

Get Kinetic Parameters for all cells and genes

**Usage**

.getParams(seed, sim, sp\_cell\_i = NULL, sp\_path\_i = NULL)

**Arguments**

seed	random seed
sim	the simulation environment
sp_cell_i	spatial cell index
sp_path_i	the pre-sampled path along the tree for this cell

**Value**

the kinetic parameters

---

.normalizeGRNParams            *Rename the original gene IDs in the GRN table to integers.*

---

**Description**

Rename the original gene IDs in the GRN table to integers.

**Usage**

.normalizeGRNParams(params)

**Arguments**

params	GRN parameters.
--------	-----------------

**Value**

list

---

`.rnormTrunc`                      *sample from truncated normal distribution*

---

**Description**

sample from truncated normal distribution

**Usage**

```
.rnormTrunc(n, mean, sd, a, b)
```

**Arguments**

<code>n</code>	number of values to create
<code>mean</code>	mean of the normal distribution
<code>sd</code>	standard deviation of the normal distribution
<code>a</code>	the minimum value allowed
<code>b</code>	the maximum value allowed

**Value**

a vector of length `n`

---

`add_expr_noise`                      *Add experimental noise to true counts*

---

**Description**

Add experimental noise to true counts

**Usage**

```
add_expr_noise(results, ...)
```

**Arguments**

<code>results</code>	The <code>scMultisim</code> result object
<code>...</code>	<code>randseed</code> : The random seed protocol: UMI or non-UMI <code>gene_len</code> : A vector with lengths of all genes <code>alpha_mean</code> , <code>alpha_sd</code> : rate of subsampling of transcripts during capture step <code>depth_mean</code> , <code>depth_sd</code> : The sequencing depth

**Value**

none

**See Also**

The underlying methods [True2ObservedCounts](#) and [True2ObservedATAC](#)



**Examples**

```
results <- sim_example(ncells = 10)
add_expr_noise(results)
```

---

add_outliers	<i>Add outliers to the observed counts</i>
--------------	--

---

**Description**

Add outliers to the observed counts

**Usage**

```
add_outliers(
  res,
  prob = 0.01,
  factor = 2,
  sd = 0.5,
  cell.num = 1,
  max.var = Inf
)
```

**Arguments**

res	The scMultisim result object
prob	The probability of adding outliers for each gene
factor	The factor of the outliers
sd	The standard deviation of the outliers
cell.num	For a gene, the number of cells chosen to add outliers
max.var	The maximum variance allowed

**Value**

none

---

cci_cell_type_params	<i>Generate cell-type level CCI parameters</i>
----------------------	--

---

**Description**

See the return value if you want to specify the cell-type level ground truth.

**Usage**

```

cci_cell_type_params(
  tree,
  total.lr,
  ctype.lr = 4:6,
  step.size = 1,
  rand = TRUE,
  discrete = FALSE
)

```

**Arguments**

tree	Use the same value for <code>sim_true_counts()</code> .
total.lr	Total number of LR pairs in the database. Use the same value for <code>sim_true_counts()</code> .
ctype.lr	If <code>rand</code> is TRUE, how many LR pairs should be enabled between each cell type pair. Should be a range, e.g. 4:6.
step.size	Use the same value for <code>sim_true_counts()</code> .
rand	Whether fill the matrix randomly
discrete	Whether the cell population is discrete. Use the same value for <code>sim_true_counts()</code> .

**Value**

A 3D matrix of (n\_cell\_type, n\_cell\_type, n\_lr). The value at (i, j, k) is 1 if there exist CCI of LR-pair k between cell type i and cell type j.

**Examples**

```

cci_cell_type_params(Phyla3(), 100, 4:6, 0.5, TRUE, FALSE)

```

---

dens_nonzero	<i>this is the density function of <math>\log(x+1)</math>, where <math>x</math> is the non-zero values for ATAC-SEQ data</i>
--------------	--

---

**Description**

this is the density function of  $\log(x+1)$ , where  $x$  is the non-zero values for ATAC-SEQ data

**Usage**

```

data(dens_nonzero)

```

**Format**

a vector.

**Value**

a vector.

**Examples**

```
data(dens_nonzero)
```

---

divide_batches	<i>Divide batches for observed counts</i>
----------------	---

---

**Description**

Divide batches for observed counts

**Usage**

```
divide_batches(results, nbatch = 2, effect = 3, randseed = 0)
```

**Arguments**

results	The scMultisim result object, after running addExprNoise()
nbatch	Number of batches
effect	Batch effect size, default is 3
randseed	Random seed

**Value**

none

**Examples**

```
results <- sim_example(ncells = 10)
add_expr_noise(results)
divide_batches(results)
```

---

gene_corr_cci	<i>Plot the ligand-receptor correlation summary</i>
---------------	---

---

**Description**

Plot the ligand-receptor correlation summary

**Usage**

```
gene_corr_cci(
  results = .getResultsFromGlobal(),
  all.genes = FALSE,
  .pair = NULL,
  .exclude.same.types = TRUE
)
```

**Arguments**

<code>results</code>	The scMultisim result object
<code>all.genes</code>	Whether to use all genes or only the ligand/receptor genes
<code>.pair</code>	Return the raw data for the given LR pair
<code>.exclude.same.types</code>	Whether to exclude neighbor cells with same cell type

**Value**

none

**Examples**

```
results <- sim_example_spatial(ncells = 10)
gene_corr_cci(results)
```

---

`gene_corr_regulator`     *Print the correlations between targets of each regulator*

---

**Description**

Print the correlations between targets of each regulator

**Usage**

```
gene_corr_regulator(results = .getResultsFromGlobal(), regulator)
```

**Arguments**

<code>results</code>	The scMultisim result object
<code>regulator</code>	The regulator ID in the GRN params

**Value**

none

**Examples**

```
results <- sim_example(ncells = 10)
gene_corr_regulator(results, 2)
```

---

gene_len_pool	<i>a pool of gene lengths to sample from</i>
---------------	--

---

**Description**

a pool of gene lengths to sample from

**Usage**

```
data(gene_len_pool)
```

**Format**

a vector.

**Value**

a vector of gene lengths.

**Examples**

```
data(gene_len_pool)
```

---

gen_1branch	<i>Generate true transcript counts for linear structure</i>
-------------	---

---

**Description**

Generate true transcript counts for linear structure

**Usage**

```
gen_1branch(  
  kinet_params,  
  start_state,  
  start_s,  
  start_u,  
  randpoints1,  
  ncells1,  
  ngenes,  
  beta_vec,  
  d_vec,  
  cycle_length_factor,  
  cell  
)
```

**Arguments**

kinet_params	kinetic parameters, include k_on, k_off, s and beta
start_state	the starting state: on or off of each gene
start_s	spliced count of the root cell in the branch
start_u	unspliced count of the root cell in the branch
randpoints1	the value which evf mean is generated from
ncells1	number of cells in the branch
ngenes	number of genes
beta_vec	splicing rate of each gene
d_vec	degradation rate of each gene
cycle_length_factor	for generating velocity data, a factor which is multiplied by the expected time to transition from kon to koff and back to to form the the length of a cycle
cell	the cell number currently having counts generated

**Value**

a list of 4 elements, the first element is true counts, second is the gene level meta information, the third is cell level meta information, including a matrix of evf and a vector of cell identity, and the fourth is the parameters kon, koff and s used to simulation the true counts

---

gen_clutter	<i>generate a clutter of cells by growing from the center</i>
-------------	---

---

**Description**

generate a clutter of cells by growing from the center

**Usage**

```
gen_clutter(
  n_cell,
  grid_size = NA,
  center = c(0, 0),
  existing_loc = NULL,
  existing_grid = NULL
)
```

**Arguments**

n_cell	the number of cells
grid_size	the width and height of the grid
center	the center of the grid
existing_loc	only place cells on the specified existing locations
existing_grid	manually specify what locations are in the grid

**Value**

a matrix of locations

**Examples**

```
gen_clutter(10, 10, c(5, 5))
```

---

Get\_1region\_ATAC\_correlation

*This function gets the average correlation rna seq counts and region effect on genes for genes which are only associated with 1 chromatin region*

---

**Description**

This function gets the average correlation rna seq counts and region effect on genes for genes which are only associated with 1 chromatin region

**Usage**

```
Get_1region_ATAC_correlation(counts, atacseq_data, region2gene)
```

**Arguments**

counts	rna seq counts
atacseq_data	atac seq data
region2gene	a 0 1 coupling matrix between regions and genes of shape (nregions) x (num_genes), where a value of 1 indicates the gene is affected by a particular region

**Value**

the correlation value

**Examples**

```
results <- sim_example(ncells = 10)
Get_1region_ATAC_correlation(results$counts, results$atacseq_data, results$region_to_gene)
```

---

Get\_ATAC\_correlation *This function gets the average correlation rna seq counts and chromatin region effect on genes*

---

### Description

This function gets the average correlation rna seq counts and chromatin region effect on genes

### Usage

```
Get_ATAC_correlation(counts, atacseq_data, num_genes)
```

### Arguments

counts	rna seq counts
atacseq_data	atac seq data
num_genes	number of genes

### Value

the correlation value

### Examples

```
results <- sim_example(ncells = 10)
Get_ATAC_correlation(results$counts, results$atacseq_data, results$num_genes)
```

---

GRN\_params\_100 *100\_gene\_GRN is a matrix of GRN params consisting of 100 genes where: # - column 1 is the target gene ID, # - column 2 is the gene ID which acts as a transcription factor for the target (regulated) gene # - column 3 is the effect of the column 2 gene ID on the column 1 gene ID*

---

### Description

100\_gene\_GRN is a matrix of GRN params consisting of 100 genes where: # - column 1 is the target gene ID, # - column 2 is the gene ID which acts as a transcription factor for the target (regulated) gene # - column 3 is the effect of the column 2 gene ID on the column 1 gene ID

### Usage

```
data(GRN_params_100)
```

### Format

a data frame.

### Value

a data frame with three columns: target gene ID, TF gene ID, and the effect of TF on target gene.



**Examples**

```
data(GRN_params_100)
```

---

GRN_params_1139	<i>GRN_params_1139 is a matrix of GRN params consisting of 1139 genes where: # - column 1 is the target gene ID, # - column 2 is the gene ID which acts as a transcription factor for the target (regulated) gene # - column 3 is the effect of the column 2 gene ID on the column 1 gene ID</i>
-----------------	--

---

**Description**

GRN\_params\_1139 is a matrix of GRN params consisting of 1139 genes where: # - column 1 is the target gene ID, # - column 2 is the gene ID which acts as a transcription factor for the target (regulated) gene # - column 3 is the effect of the column 2 gene ID on the column 1 gene ID

**Usage**

```
data(GRN_params_1139)
```

**Format**

a data frame.

**Value**

a data frame with three columns: target gene ID, TF gene ID, and the effect of TF on target gene.

**Examples**

```
data(GRN_params_1139)
```

---

len2nfrag	<i>from transcript length to number of fragments (for the nonUMI protocol)</i>
-----------	--

---

**Description**

from transcript length to number of fragments (for the nonUMI protocol)

**Usage**

```
data(len2nfrag)
```

**Format**

a vector.

**Value**

a vector.

**Examples**

```
data(len2nfrag)
```

---

match_params	<i>distribution of kinetic parameters learned from the Zeisel UMI cortex datasets</i>
--------------	---

---

**Description**

distribution of kinetic parameters learned from the Zeisel UMI cortex datasets

**Usage**

```
data(param_realdata.zeisel.imputed)
```

**Format**

a data frame.

**Value**

a data frame.

**Examples**

```
data(param_realdata.zeisel.imputed)
```

---

OP	<i>Get option from an object in the current environment</i>
----	---

---

**Description**

Get option from an object in the current environment

**Usage**

```
OP(..., .name = "options")
```

**Arguments**

...	the parameter name
.name	get option from this object

**Value**

the parameter value

---

Phyla1

*Creating a linear example tree*

---

**Description**

Creating a linear example tree

**Usage**

```
Phyla1(len = 1)
```

**Arguments**

len                    length of the tree

**Value**

a R phylo object

**Examples**

```
Phyla1(len = 1)
```

---

Phyla3

*Creating an example tree with 3 tips*

---

**Description**

Creating an example tree with 3 tips

**Usage**

```
Phyla3(plotting = FALSE)
```

**Arguments**

plotting                True for plotting the tree on console, False for no plot

**Value**

a R phylo object

**Examples**

```
Phyla3()
```

---

Phyla5 *Creating an example tree with 5 tips*

---

**Description**

Creating an example tree with 5 tips

**Usage**

```
Phyla5(plotting = FALSE)
```

**Arguments**

plotting            True for plotting the tree on console, False for no plot

**Value**

a R phylo object

**Examples**

```
Phyla5()
```

---

plot\_cell\_loc *Plot cell locations*

---

**Description**

Plot cell locations

**Usage**

```
plot_cell_loc(
  results = .getResultsFromGlobal(),
  size = 4,
  show.label = FALSE,
  show.arrows = TRUE,
  lr.pair = 1,
  .cell.pop = NULL,
  .locs = NULL
)
```

**Arguments**

results	The scMultisim result object
size	Fig size
show.label	Show cell numbers
show.arrows	Show arrows representing cell-cell interactions
lr.pair	The ligand-receptor pair used to plot CCI arrows results\$cci_cell_type_param[lr.pair]
.cell.pop	Specify the cell population metadata
.locs	Manually specify the cell locations as a 2xncells matrix

**Value**

none

**Examples**

```
results <- sim_example_spatial(ncells = 10)
plot_cell_loc(results)
```

---

plot\_gene\_module\_cor\_heatmap

*Plot the gene module correlation heatmap*

---

**Description**

Plot the gene module correlation heatmap

**Usage**

```
plot_gene_module_cor_heatmap(
  results = .getResultsFromGlobal(),
  seed = 0,
  grn.genes.only = TRUE,
  save = FALSE
)
```

**Arguments**

results	The scMultisim result object
seed	The random seed
grn.genes.only	Plot the GRN gens only
save	save the plot as pdf

**Value**

none

**Examples**

```
results <- sim_example(ncells = 10)
plot_gene_module_cor_heatmap(results)
```

---

plot_grid	<i>Plot the CCI grid</i>
-----------	--------------------------

---

**Description**

In normal cases, please use plotCellLoc instead.

**Usage**

```
plot_grid(results = .getResultsFromGlobal())
```

**Arguments**

results            The scMultisim result object

**Value**

none

**Examples**

```
results <- sim_example_spatial(ncells = 10)
plot_grid(results)
```

---

plot_grn	<i>Plot the GRN network</i>
----------	-----------------------------

---

**Description**

Plot the GRN network

**Usage**

```
plot_grn(params)
```

**Arguments**

params            The GRN params data frame

**Value**

none

**Examples**

```
data(GRN_params_100, envir = environment())
plot_grn(GRN_params_100)
```

---

plot_phyla	<i>Plot a R phylogenetic tree</i>
------------	-----------------------------------

---

**Description**

Plot a R phylogenetic tree

**Usage**

```
plot_phyla(tree)
```

**Arguments**

tree	The tree
------	----------

**Value**

none

**Examples**

```
plot_phyla(Phyla5())
```

---

plot_rna_velocity	<i>Plot RNA velocity as arrows on tSNE plot</i>
-------------------	---

---

**Description**

Plot RNA velocity as arrows on tSNE plot

**Usage**

```
plot_rna_velocity(
  results = .getResultsFromGlobal(),
  velocity = results$velocity,
  perplexity = 70,
  arrow.length = 1,
  save = FALSE,
  randseed = 0,
  ...
)
```

**Arguments**

results	The scMultiSim result object
velocity	The velocity matrix, by default using the velocity matrix in the result object
perplexity	The perplexity for tSNE
arrow.length	The length scaler of the arrow
save	Whether to save the plot
randseed	The random seed
...	Other parameters passed to ggplot

**Value**

The plot

**Examples**

```
results <- sim_example(ncells = 10, velocity = TRUE)
plot_rna_velocity(results, perplexity = 3)
```

---

plot\_tsne

*Plot t-SNE visualization of a data matrix*

---

**Description**

Plot t-SNE visualization of a data matrix

**Usage**

```
plot_tsne(
  data,
  labels,
  perplexity = 60,
  legend = "",
  plot.name = "",
  save = FALSE,
  rand.seed = 0,
  continuous = FALSE,
  labels2 = NULL,
  lim = NULL,
  runPCA = FALSE,
  alpha = 1
)
```

**Arguments**

data	The dxn matrix
labels	A vector of length n, usually cell clusters
perplexity	Perplexity value used for t-SNE
legend	A list of colors for the labels
plot.name	The plot title
save	If TRUE, save as plot.name.pdf
rand.seed	The random seed
continuous	Whether labels should be treated as continuous, e.g. pseudotime
labels2	Additional label
lim	Specify the xlim and y lim c(x_min, x_max, y_min, y_max)
runPCA	Whether to run PCA before t-SNE
alpha	The alpha value for the points



**Value**

the figure if not save, otherwise save the figure as plot.name.pdf

**Examples**

```
results <- sim_example(ncells = 10)
plot_tsne(log2(results$counts + 1), results$cell_meta$pop, perplexity = 3)
```

---

run\_shiny

*Launch the Shiny App to configure the simulation*

---

**Description**

Launch the Shiny App to configure the simulation

**Usage**

```
run_shiny()
```

---

SampleDen

*sample from smoothed density function*

---

**Description**

sample from smoothed density function

**Usage**

```
SampleDen(nsamples, den_fun, reduce.mem = FALSE)
```

**Arguments**

nsamples	number of samples needed
den_fun	density function estimated from density() from R default
reduce.mem	use alternative implementation to reduce memory usage

**Value**

a vector of samples

---

scmultisim_help	<i>Show detailed documentations of scMultiSim's parameters</i>
-----------------	--

---

**Description**

Show detailed documentations of scMultiSim's parameters

**Usage**

```
scmultisim_help(topic = NULL)
```

**Arguments**

topic	Can be options, dynamic.GRN, or cci
-------	-------------------------------------

**Value**

none

**Examples**

```
scmultisim_help()
```

---

sim_example	<i>Simulate a small example dataset with 200 cells and the 100-gene GRN</i>
-------------	---

---

**Description**

Simulate a small example dataset with 200 cells and the 100-gene GRN

**Usage**

```
sim_example(ncells = 10, velocity = FALSE)
```

**Arguments**

ncells	number of cells, please increase this number on your machine
velocity	whether to simulate RNA velocity

**Value**

the simulation result

**Examples**

```
sim_example(ncells = 10)
```

---

sim_example_spatial	<i>Simulate a small example dataset with 200 cells and the 100-gene GRN, with CCI enabled</i>
---------------------	---

---

**Description**

Simulate a small example dataset with 200 cells and the 100-gene GRN, with CCI enabled

**Usage**

```
sim_example_spatial(ncells = 10)
```

**Arguments**

ncells                    number of cells, please increase this number on your machine

**Value**

the simulation result

**Examples**

```
sim_example_spatial(ncells = 10)
```

---

sim_true_counts	<i>Simulate true scRNA and scATAC counts from the parameters</i>
-----------------	--

---

**Description**

Simulate true scRNA and scATAC counts from the parameters

**Usage**

```
sim_true_counts(options, return_summarized_exp = FALSE)
```

**Arguments**

options                    See scMultiSim\_help().  
return\_summarized\_exp        Whether to return a SummarizedExperiment object.

**Value**

scMultiSim returns an environment with the following fields:

- counts: Gene-by-cell scRNA-seq counts.
- atac\_counts: Region-by-cell scATAC-seq counts.
- region\_to\_gene: Region-by-gene 0-1 matrix indicating the corresponding relationship between chromatin regions and genes.
- atacseq\_data: The "clean" scATAC-seq counts without added intrinsic noise.

- `cell_meta`: A dataframe containing cell type labels and pseudotime information.
- `cif`: The CIF used during the simulation.
- `giv`: The GIV used during the simulation.
- `kinetic_params`: The kinetic parameters used during the simulation.
- `.grn`: The GRN used during the simulation.
- `.grn$regulators`: The list of TFs used by all gene-by-TF matrices.
- `.grn$geff`: Gene-by-TF matrix representing the GRN used during the simulation.
- `.n`: Other metadata, e.g. `.n$cells` is the number of cells.

If `do.velocity` is enabled, it has these additional fields:

- `unspliced_counts`: Gene-by-cell unspliced RNA counts.
- `velocity`: Gene-by-cell RNA velocity ground truth.
- `cell_time`: The pseudotime at which the cell counts were generated.

If dynamic GRN is enabled, it has these additional fields:

- `cell_specific_grn`: A list of length `n_cells`. Each element is a gene-by-TF matrix, indicating the cell's GRN.

If cell-cell interaction is enabled, it has these additional fields:

- `grid`: The grid object used during the simulation.
  - `grid$get_neighbours(i)`: Get the neighbour cells of cell `i`.
- `cci_locs`: A dataframe containing the X and Y coordinates of each cell.
- `cci_cell_type_param`: A dataframe containing the CCI network ground truth: all ligand-receptor pairs between each pair of cell types.
- `cci_cell_types`: For continuous cell population, the sub-divided cell types along the trajectory used when simulating CCI.

If it is a debug session (`debug = TRUE`), a `sim` field is available, which is an environment contains all internal states and data structures.

## Examples

```
data(GRN_params_100, envir = environment())
sim_true_counts(list(
  rand.seed = 0,
  GRN = GRN_params_100,
  num.cells = 100,
  num.cifs = 50,
  tree = Phyla5()
))
```

---

spatialGrid-class      *The class for spatial grids*

---

**Description**

The class for spatial grids

**Value**

a spatialGrid object

**Fields**

method the method to generate the cell layout  
 grid\_size the width and height of the grid  
 ncells the number of cells  
 grid the grid matrix  
 locs a list containing the locations of all cells  
 loc\_order deprecated, don't use; the order of the locations  
 cell\_types a map to save the cell type of each allocated cell  
 same\_type\_prob the probability of a new cell placed next to a cell with the same type  
 max\_nbs the maximum number of neighbors for each cell  
 nb\_map a list containing the neighbors for each cell  
 nb\_adj adjacency matrix for neighbors  
 nb\_radius the radius of neighbors  
 final\_types the final cell types after the final time step  
 pre\_allocated\_pos the pre-allocated positions for each cell, if any  
 method\_param additional parameters for the layout method

---

True2ObservedATAC      *Simulate observed ATAC-seq matrix given technical noise and the true counts*

---

**Description**

Simulate observed ATAC-seq matrix given technical noise and the true counts

**Usage**

```
True2ObservedATAC(
  atacseq_data,
  randseed,
  observation_prob = 0.3,
  sd_frac = 0.1
)
```

**Arguments**

`atacseq_data` true ATAC-seq data  
`randseed` (should produce same result if `nregions`, `nevf` and `randseed` are all the same)  
`observation_prob` for each integer count of a particular region for a particular cell, the probability the count will be observed  
`sd_frac` the fraction of ATAC-seq data value used as the standard deviation of added normally distributed noise

**Value**

a matrix of observed ATAC-seq data

**Examples**

```

results <- sim_example(ncells = 10)
True2ObservedATAC(results$atac_counts, randseed = 1)

```

---

True2ObservedCounts	<i>Simulate observed count matrix given technical biases and the true counts</i>
---------------------	--

---

**Description**

Simulate observed count matrix given technical biases and the true counts

**Usage**

```

True2ObservedCounts(
  true_counts,
  meta_cell,
  protocol,
  randseed,
  alpha_mean = 0.1,
  alpha_sd = 0.002,
  alpha_gene_mean = 1,
  alpha_gene_sd = 0,
  gene_len,
  depth_mean,
  depth_sd,
  lenslope = 0.02,
  nbins = 20,
  amp_bias_limit = c(-0.2, 0.2),
  rate_2PCR = 0.8,
  nPCR1 = 16,
  nPCR2 = 10,
  LinearAmp = FALSE,
  LinearAmp_coef = 2000
)

```

**Arguments**

<code>true_counts</code>	gene cell matrix
<code>meta_cell</code>	the meta information related to cells, will be combined with technical cell level information and returned
<code>protocol</code>	a string, can be "nonUMI" or "UMI"
<code>randseed</code>	(should produce same result if nregions, nev and randseed are all the same)
<code>alpha_mean</code>	the mean of rate of subsampling of transcripts during capture step, default at 10 percent efficiency
<code>alpha_sd</code>	the std of rate of subsampling of transcripts
<code>alpha_gene_mean</code>	the per-gene scale factor of the alpha parameter, default at 1
<code>alpha_gene_sd</code>	the standard deviation of the per-gene scale factor of the alpha parameter, default at 0
<code>gene_len</code>	a vector with lengths of all genes
<code>depth_mean</code>	mean of sequencing depth
<code>depth_sd</code>	std of sequencing depth
<code>lenslope</code>	amount of length bias
<code>nbins</code>	number of bins for gene length
<code>amp_bias_limit</code>	range of amplification bias for each gene, a vector of length ngenes
<code>rate_2PCR</code>	PCR efficiency, usually very high, default is 0.8
<code>nPCR1</code>	the number of PCR cycles in "pre-amplification" step, default is 16
<code>nPCR2</code>	the number of PCR cycles used after fragmentation.
<code>LinearAmp</code>	if linear amplification is used for pre-amplification step, default is FALSE
<code>LinearAmp_coef</code>	the coefficient of linear amplification, that is, how many times each molecule is amplified by

**Value**

if UMI, a list with two elements, the first is the observed count matrix, the second is the metadata;  
if nonUMI, a matrix

**Examples**

```

results <- sim_example(ncells = 10)
data(gene_len_pool)
gene_len <- sample(gene_len_pool, results$num_genes, replace = FALSE)
True2ObservedCounts(
  results$counts, results$cell_meta, protocol = "nonUMI", randseed = 1,
  alpha_mean = 0.1, alpha_sd = 0.05, gene_len = gene_len, depth_mean = 1e5, depth_sd = 3e3
)

```

# Index

## \* datasets

dens\_nonzero, 10  
gene\_len\_pool, 13  
GRN\_params\_100, 16  
GRN\_params\_1139, 17  
len2nfrag, 17  
match\_params, 18

## \* internal

.amplifyOneCell, 3  
.calAmpBias, 4  
.continuousCIF, 4  
.divideBatchesImpl, 5  
.expandToBinary, 6  
.getCountCorrMatrix, 6  
.getParams, 7  
.normalizeGRNParams, 7  
.rnormTrunc, 8  
gen\_1branch, 13  
gene\_len\_pool, 13  
len2nfrag, 17  
match\_params, 18  
OP, 18  
SampleDen, 25  
.SpatialGrid (spatialGrid-class), 29  
.amplifyOneCell, 3  
.calAmpBias, 4  
.continuousCIF, 4  
.divideBatchesImpl, 5  
.expandToBinary, 6  
.getCountCorrMatrix, 6  
.getParams, 7  
.normalizeGRNParams, 7  
.rnormTrunc, 8

add\_expr\_noise, 8  
add\_outliers, 9

cci\_cell\_type\_params, 9

dens\_nonzero, 10  
divide\_batches, 11

gen\_1branch, 13  
gen\_clutter, 14

gene\_corr\_cci, 11  
gene\_corr\_regulator, 12  
gene\_len\_pool, 13  
Get\_1region\_ATAC\_correlation, 15  
Get\_ATAC\_correlation, 16  
GRN\_params\_100, 16  
GRN\_params\_1139, 17

len2nfrag, 17

match\_params, 18

OP, 18

Phyla1, 19  
Phyla3, 19  
Phyla5, 20  
plot\_cell\_loc, 20  
plot\_gene\_module\_cor\_heatmap, 21  
plot\_grid, 22  
plot\_grn, 22  
plot\_phyla, 23  
plot\_rna\_velocity, 23  
plot\_tsne, 24

run\_shiny, 25

SampleDen, 25  
scmultisim\_help, 26  
sim\_example, 26  
sim\_example\_spatial, 27  
sim\_true\_counts, 27  
spatialGrid-class, 29

True2ObservedATAC, 8, 29  
True2ObservedCounts, 8, 30