

# Package ‘MOSim’

January 23, 2025

**Title** Multi-Omics Simulation (MOSim)

**Version** 2.3.0

**Description** MOSim package simulates multi-omic experiments that mimic regulatory mechanisms within the cell, allowing flexible experimental design including time course and multiple groups.

**Encoding** UTF-8

**Depends** R (>= 4.2.0)

**License** GPL-3

**LazyData** false

**biocViews** Software, TimeCourse, ExperimentalDesign, RNASeq

**BugReports** <https://github.com/ConesaLab/MOSim/issues>

**URL** <https://github.com/ConesaLab/MOSim>

**Imports** HiddenMarkov, zoo, IRanges, S4Vectors, dplyr, ggplot2, lazyeval, matrixStats, methods, rlang, stringi, stringr, scran, Seurat, Signac, edgeR, Rcpp

**Suggests** testthat, knitr, rmarkdown, codetools, BiocStyle, stats, utils, purrr, scales, tibble, tidyr, Biobase, scater, SingleCellExperiment, decor, markdown, Rsamtools, igraph, leiden, bluster

**Collate** 'AllClass.R' 'AllGeneric.R' 'Simulator.R' 'SimulatorRegion.R' 'ChIP-seq.R' 'DNase-seq.R' 'MOSim-package.R' 'functions.R' 'Simulation.R' 'MOSim.R' 'RNA-seq.R' 'data.R' 'simulate\_WGBS\_functions.R' 'methyl-seq.R' 'miRNA-seq.R' 'sc\_MOSim.R' 'sc\_coexpression.R' 'sparsim\_functions.R' 'zzz.R'

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**LinkingTo** cpp11, Rcpp

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

**git\_branch** devel

**git\_last\_commit** 8e78c32

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

**Repository** Bioconductor 3.21

**Date/Publication** 2025-01-22

**Author** Carolina Monzó [aut],  
 Carlos Martínez [aut],  
 Sonia Tarazona [cre, aut]

**Maintainer** Sonia Tarazona <sotacam@gmail.com>

## Contents

MOSim-package	3
associationList	3
calculate_mean_per_list_df	4
check_patterns	4
discretize	5
experimentalDesign	6
is.declared	6
make_association_dataframe	7
make_cluster_patterns	8
match_gene_regulator	8
match_gene_regulator_cluster	9
mosim	10
MOSimulation-class	12
MOSimulator-class	13
MOSimulatorRegion-class	14
omicData	15
omicResults	16
omicSettings	17
omicSim	18
plotProfile	19
random_unif_interval	20
sampleData	20
scatac	21
scrna	21
sc_mosim	22
sc_omicData	24
sc_omicResults	25
sc_omicSettings	25
sc_param_estimation	26
shuffle_group_matrix	27
simulate_coexpression	28
simulate_hyper	29
sparsim_create_simulation_parameter	30
sparsim_estimate_intensity	31
sparsim_estimate_library_size	32
sparsim_estimate_parameter_from_data	32

*MOSim-package* 3

sparsim_estimate_variability . . . . .	33
sparsim_simulation . . . . .	34
TF_human . . . . .	35

**Index** 36

---

*MOSim-package*      *MOSim*

---

### Description

Multiomics simulation package.

### Author(s)

**Maintainer:** Sonia Tarazona <sotacam@gmail.com>

Authors:

- Carolina Monzó <carolmonzoc@gmail.com>
- Carlos Martínez <cmarmir@gmail.com>

### See Also

Useful links:

- <https://github.com/ConesaLab/MOSim>
- Report bugs at <https://github.com/ConesaLab/MOSim/issues>

---

*associationList*      *Data to showcase scRNA and scATAC-seq association*

---

### Description

Data to showcase scRNA and scATAC-seq association

### Usage

```
data("associationList")
```

### Format

A dataframe with two columns and rows according to gene/feature relationships

**Peak\_ID** ATAC chromosomic positions associated to genes

**Gene\_ID** RNA genes associated to peaks

@source Created in-house to serve as an example

---

```
calculate_mean_per_list_df  
  calculate_mean_per_list_df
```

---

**Description**

Helper function to calculate mean expression per celltype

**Usage**

```
calculate_mean_per_list_df(df, named_lists)
```

**Arguments**

`df` dataframe of expression where columns are cells  
`named_lists` list of which cells belong to each celltype

**Examples**

```
rna <- data.frame(c1 = c(1.5, 15.5, 3.5, 20.5), c2 = c(2, 15, 4, 20),  
                 c3 = c(10, 1, 12, 13), c4 = c(11, 1, 13, 14))  
cell_types <- list("ct1" = c(1,2), "ct2" = c(3, 4))  
calculate_mean_per_list_df(rna, cell_types)
```

---

```
check_patterns  check_patterns
```

---

**Description**

Function to check if the TRUE FALSE patterns have at least two rows that are opposite, we need this to be able to generate repressor regulators

**Usage**

```
check_patterns(patterns)
```

**Arguments**

`patterns` tibble of TRUE FALSE values

**Value**

list of indices where the rows are opposite

**Examples**

```
patterns <- tibble::tibble(one = c(TRUE, FALSE, TRUE, FALSE),
  two = c(TRUE, TRUE, TRUE, TRUE),
  three = c(FALSE, TRUE, FALSE, TRUE),
  four = c(FALSE, TRUE, TRUE, TRUE))
opposite_indices <- check_patterns(patterns)
```

---

**discretize***Discretize ChIP-Seq counts to simulate a binary dataset*

---

**Description**

Discretize ChIP-Seq counts to simulate a binary dataset

**Usage**

```
discretize(df, omic)
```

**Arguments**

df	A MOSimulated object
omic	Character string of the omic to transform into binary data

**Value**

A regulator dataframe of 0 and 1

**Examples**

```
omic_list <- c("RNA-seq", "ChIP-seq")
rnaseq_simulation <- mosim(omics = omic_list,
  omicsOptions = c(omicSim("ChIP-seq", totalFeatures = 2500)))
rnaseq_simulated <- omicResults(rnaseq_simulation, omic_list)
discrete_ChIP <- discretize(rnaseq_simulated, "ChIP-seq")
```

---

experimentalDesign      *Retrieves the experimental design*

---

**Description**

Retrieves the experimental design

**Usage**

```
experimentalDesign(simulation)
```

**Arguments**

simulation      A MOSimulation object

**Value**

A data frame containing the experimental design used to simulate the data.

**Examples**

```
omic_list <- c("RNA-seq")
rnaseq_simulation <- mosim(omics = omic_list)
# This will be a data frame with RNA-seq counts

design_matrix <- experimentalDesign(rnaseq_simulation)
```

---

is.declared      *Check if a variable is declared.*

---

**Description**

Check if a variable is declared.

**Usage**

```
is.declared(object, key = NULL)
```

**Arguments**

object      Variable name to check  
key      Optional key to check inside object.

**Value**

TRUE or FALSE indicating if the variable is initialized & non-empty.

---

```
make_association_dataframe  
    make_association_dataframe
```

---

## Description

This function generates a dataframe containing the information of the relationship between ATAC and RNA, based on the cluster groups, and then tells the order the genes and peaks should be in the simulated dataframe of the group

## Usage

```
make_association_dataframe(  
  group,  
  generegroup,  
  numtotalgenes,  
  numtotalpeaks,  
  minFC,  
  maxFC  
)
```

## Arguments

group	Group from which we are generating the association dataframe
generegroup	list of elements to generate the association dataframe such as clusters of each omic, indices of opposite clusters, which genes are activated, repressed, behavior of the features etc.
numtotalgenes	total number of genes
numtotalpeaks	total number of peaks
minFC	FC below which is downregulated
maxFC	FC above which is upregulated

## Value

a dataframe with all the information the user needs about each gene and the order of gene and peak names to rename them in the simulated datasets of the group

---

make\_cluster\_patterns *make\_cluster\_patterns*

---

### Description

Function to make the tibble with cluster combinations for the gene expression patterns along the cells. This function is a slightly modified copy of the 'make\_cluster\_patterns' function from the 'Acorde' package (v1.0.0), originally developed by Arzalluz-Luque A, Salguero P, Tarazona S, Conesa A. (2022). acorde unravels functionally interpretable networks of isoform co-usage from single cell data. Nature communications 1828. DOI: 10.1038/s41467-022-29497-w. The original package is licensed under the GPL-3 license.

### Usage

```
make_cluster_patterns(numcells = 4, clusters = 8)
```

### Arguments

`numcells`            Number of different celltypes we are simulating  
`clusters`            OPTIONAL. Number of co-expression patterns the user wants to simulate

### Value

A tibble with number of columns equal to number of celltypes, rows according to the number of TRUE/FALSE combinations corresponding to the gene expression patterns along the cells

### Examples

```
patterns <- make_cluster_patterns(numcells = 4, clusters = 8)
cell_types <- list('Treg' = c(1:10), 'cDC' = c(11:20), 'CD4_TEM' = c(21:30),
  'Memory_B' = c(31:40))
patterns <- make_cluster_patterns(numcells = length(cell_types),
  clusters = 8)
```

---

match\_gene\_regulator *match\_gene\_regulator*

---

### Description

Helper function to make the most similar profiles possible between gene and regulator

### Usage

```
match_gene_regulator(rna, atac, cell_types, associationList)
```





## Examples

```
rna <- data.frame(c1 = c(1.5, 15.5, 3.5, 20.5), c2 = c(2, 15, 4, 20),
  c3 = c(10, 1, 12, 13), c4 = c(11, 1, 13, 14), c5 = c(7, 0, 0, 0),
  c6 = c(8, 1, 1, 1), c7 = c(8, 1, 1, 1))
rownames(rna) <- c('GenB', 'GenA', 'GenC', 'GenD')
associationList <- data.frame(Gene_ID = c('GenA', 'GenB', 'GenC', 'GenA'),
  Peak_ID = c('PeakA', 'PeakB', 'PeakC', 'PeakD'),
  Gene_cluster = c(1, 2, 1, 2), Peak_cluster = c(1, 2, 1, 2))
cell_types <- list("ct1" = c(1,2), "ct2" = c(3, 4), "ct3" = c(5, 6), "ct4" = c(7))
atac <- data.frame(c1 = c(3,20, 1,15, 1, 7, 1), c2 = c(4,20, 2,15, 0, 5, 1.5),
  c3 = c(10, 13, 1, 12, 1, 14, 9), c4 = c(11, 14, 1, 13, 1, 4, 12),
  c5 = c(0, 0, 0, 7, 1, 6, 6), c6 = c(1, 1, 1, 8, 0, 5, 8),
  c7 = c(1, 1, 1, 8, 1, 5, 7))
rownames(atac) <- c('PeakB', "PeakC", "PeakF", "PeakD", "PeakE", "PeakA", "PeakG")
match_gene_regulator_cluster(rna, atac, cell_types, associationList)
```

---

 mosim

 mosim
 

---

## Description

Performs a multiomic simulation by chaining two actions: 1) Creating the "MOSimulation" class with the provided params. 2) Calling "simulate" method on the initialized object.

## Usage

```
mosim(
  omics,
  omicsOptions,
  diffGenes,
  numberReps,
  numberGroups,
  times,
  depth,
  profileProbs,
  minMaxFC,
  TFtoGene
)
```

## Arguments

**omics** Character vector containing the names of the omics to simulate, which can be "RNA-seq", "miRNA-seq", "DNase-seq", "ChIP-seq" or "Methyl-seq" (e.g. c("RNA-seq", "miRNA-seq")). It can also be a list with the omic names as names and their options as values, but we recommend to use the argument omicSim to provide the options to simulated each omic.

omicsOptions	<p>List containing the options to simulate each omic. We recommend to apply the helper method <code>omicSim</code> to create this list in a friendly way, and the function <code>omicData</code> to provide custom data (see the related sections for more information). Each omic may have different configuration parameters, but the common ones are:</p> <p><b>simuData/idToGene</b> Seed sample and association tables for regulatory omics. The helper function <code>omicData</code> should be used to provide this information (see the following section).</p> <p><b>regulatorEffect</b> For regulatory omics. List containing the percentage of effect types (repressor, activator or no effect) over the total number of regulators. See vignette for more information.</p> <p><b>totalFeatures</b> Number of features to simulate. By default, the total number of features in the seed dataset.</p> <p><b>depth</b> Sequencing depth in millions of reads. If not provided, it takes the global parameter passed to <code>mosim</code> function.</p> <p><b>replicateParams</b> List with parameters <i>a</i> and <i>b</i> for adjusting the variability in the generation of replicates using the negative binomial. See vignette for more information.</p>
diffGenes	Number of differentially expressed genes to simulate, given in percentage (0 - 1) or in absolute number (> 1). By default 0.15
numberReps	Number of replicates per experimental condition (and time point, if time series are to be generated). By default 3.
numberGroups	Number of experimental groups or conditions to simulate.
times	Vector of time points to consider in the experimental design.
depth	Sequencing depth in millions of reads.
profileProbs	Numeric vector with the probabilities to assign each of the patterns. Defaults to 0.2 for each.
minMaxFC	Numeric vector of length 2 with minimum and maximum fold-change for differentially expressed features, respectively.
TFtoGene	A logical value indicating if default transcription factors data should be used (TRUE) or not (FALSE), or a 3 column data frame containing custom associations. By default FALSE.

### Value

Instance of class "MOSimulation" containing the multiomic simulation data.

### Examples

```

moSimulation <- mosim(
  omics = c("RNA-seq"),
  numberReps = 3,
  times = c(0, 2, 6, 12, 24)
)

# Retrieve simulated count matrix for RNA-seq

```

```
dataRNAseq <- omicResults(moSimulation, "RNA-seq")
```

---

**MOSimulation-class**      *This class manages the global simulation process, like associating genes with gene classes, regulatory programs and other settings. Finally it will initialize the simulators with their options that will use the previously generated settings to simulate the data.*

---

### Description

This class manages the global simulation process, like associating genes with gene classes, regulatory programs and other settings. Finally it will initialize the simulators with their options that will use the previously generated settings to simulate the data.

### Slots

**simulators** Vector containing either S4 initialized classes of simulators or a list with the class name as keys, and its options as value, see example.

**totalGenes** A number with the total number of genes including not expressed. Overwritten if a genome reference is provided. Currently not used as we force to provide real data.

**diffGenes** A number with the total number of differential genes (if value > 1) or % of total genes (if value < 1).

**numberReps** Number of replicates of the experiment.

**numberGroups** Number of samples considered on the experiment.

**times** Numeric vector containing the measured times. If **numberGroups** < 2, the number of times must be at least 2.

**geneNames** Read only. List containing the IDs of the genes. Overwritten by the genome reference if provided. Currently not used as we force to provide real data.

**simSettings** List of settings that overrides initializing the configuration of the simulation by passing a previously generated list. This could be used to tweak by hand the assigned profiles, genes, regulatory programs, etc.

**noiseFunction** Noise function to apply when simulating counts. Must accept the parameter 'n' and return a vector of the same length. Defaults to 'rnorm'

**profiles** Named list containing the patterns with their coefficients.

**profileProbs** Numeric vector with the probabilities to assign each of the patterns. Defaults to 0.2 for each.

**noiseParams** Default noise parameters to be used with noise function.

**depth** Default depth to simulate.

**TFtoGene** Boolean (for default data) or 3 column data frame containing Symbol-TFGene-LinkedGene

**minMaxQuantile** Numeric vector of length 2 indicating the quantiles to use in order to retrieve the absolute minimum and maximum value that a differentially expressed feature can have.

**minMaxFC** Numeric vector of length 2 indicating the minimum and maximum fold-change that a differentially expressed feature can have.

---

MOSimulator-class      *Virtual class containing common methods and slots for child classes.*

---

### Description

Virtual class containing common methods and slots for child classes.

### Slots

`name` Name of the simulator to be used in messages.

`data` Data frame containing the initial sample to be used, with the features IDs as rownames and only one column named "Counts".

`regulator` Boolean flag to indicate if the omic is a regulator or not.

`regulatorEffect` Possible regulation effects of the omic (enhancer, repressor or both).

`idToGene` Data frame with the association table between genes and other features. The structure must be 2 columns, one named "ID" and the other "Gene".

`min` Minimum value allowed in the omic.

`max` Maximum value allowed in the omic.

`depth` Sequencing depth to simulate.

`depthRound` Number of decimal places to round when adjusting depth.

`depthAdjust` Boolean indicating whether to adjust by sequencing depth or not.

`totalFeatures` Number of features to simulate. This will replace the data with a subset.

`noiseFunction` Noise function to apply when simulating counts. Must accept the parameter 'n' and return a vector of the same length. Defaults to 'rnorm'

`increment` Read-only. Minimum value to increase when simulating counts.

`simData` Contains the final simulated data.

`pregenerated` Indicates if the child class will generate the simulated data instead of the general process.

`randData` Auxiliary vector containing the original count data in random order with other adjustments.

`noiseParams` Noise parameters to be used with noise function.

`roundDigits` Number of digits to round the simulated count values.

`minMaxQuantile` Numeric vector of length 2 indicating the quantiles to use in order to retrieve the absolute minimum and maximum value that a differentially expressed feature can have.

`minMaxFC` Numeric vector of length 2 indicating the minimum and maximum fold-change that a differentially expressed feature can have.

`minMaxDist` Named list containing different minimum and maximum constraints values calculated at the beginning of the simulation process.

`replicateParams` Named list containing the parameters a and b to be used in the replicates generation process, see the vignette for more info.

---

MOSimulatorRegion-class

*Virtual class containing general methods for simulators based on regions of the chromosomes, like DNase-seq, ChIP-seq or Methyl-seq*

---

### Description

Virtual class containing general methods for simulators based on regions of the chromosomes, like DNase-seq, ChIP-seq or Methyl-seq

Class to simulate RNA-seq data

Class to simulate transcription factor data

Class to simulate miRNA-seq

Class to simulate ChIP-seq data

Class to simulate DNase-seq data

Class to simulate Methyl-seq data.

### Slots

locs Vector containing the list of locations of the sites.

locsName Type of the site to simulate, only for debug.

splitChar Character symbol used to split identifiers in chr/start/end

nCpG numeric. Number of CpG sites to simulate.

pSuccessMethReg numeric. Probability of success in methylated region.

pSuccessDemethReg numeric. Probability of success in non methylated region

errorMethReg numeric. Error rate in methylated region

errorDemethReg numeric. Error rate in methylated region

nReadsMethReg numeric. Mean number of reads in methylated region.

nReadsDemethReg numeric. Mean number of reads in non methylated regions.

phaseDiff numeric. Phase difference in the differentially methylated regions between two samples

balanceHypoHyper numeric. Balance of hypo/hyper methylation

ratesHMMMatrix numeric. Matrix of values that describes the exponential decay functions that define the distances between CpG values.

distType character. Distribution used to generate replicates:

transitionSize numeric.

PhiMeth matrix. Transition matrix for CpG locations.

PhiDemeth matrix. <Not used>

typesLocation numeric. <Not used>

returnValue character. Selected column:

betaThreshold numeric. Beta threshold value used to calculate M values.

---

omicData	<i>Set customized data for an omic.</i>
----------	---

---

**Description**

Set customized data for an omic.

**Usage**

```
omicData(omic, data = NULL, associationList = NULL)
```

**Arguments**

omic	The name of the omic to provide data.
data	Data frame with the omic identifiers as row names and just one column named Counts containing numeric values used as initial sample for the simulation.
associationList	Only for regulatory omics, a data frame with 2 columns, the first called containing the regulator ID and the second called Gene with the gene identifier.

**Value**

Initialized simulation object with the given data.

**Examples**

```
# Take a subset of the included dataset for illustration
# purposes. We could also load it from a csv file or RData,
# as long as we transform it to have 1 column named "Counts"
# and the identifiers as row names.

data(sampleData)

custom_rnaseq <- head(sampleData$SimRNAseq$data, 100)

# In this case, 'custom_rnaseq' is a data frame with
# the structure:
head(custom_rnaseq)
##           Counts
## ENSMUSG0000000001 6572
## ENSMUSG0000000003     0
## ENSMUSG0000000028 4644
## ENSMUSG0000000031     8
## ENSMUSG0000000037     0
## ENSMUSG0000000049     0

# The helper 'omicData' returns an object with our custom data.
rnaseq_customdata <- omicData("RNA-seq", data = custom_rnaseq)
```

---

omicResults	<i>Retrieves the simulated data.</i>
-------------	--------------------------------------

---

### Description

Retrieves the simulated data.

### Usage

```
omicResults(simulation, omics = NULL, format = "data.frame")
```

### Arguments

simulation	A MOSimulation object.
omics	List of the omics to retrieve the simulated data.
format	Type of object to use for returning the results

### Value

A list containing an element for every omic specified, with the simulation data in the format indicated, or a numeric matrix with simulated data if the omic name is directly provided.

### Examples

```
omic_list <- c("RNA-seq")
rnaseq_simulation <- mosim(omics = omic_list)
#' # This will be a data frame with RNA-seq counts
rnaseq_simulated <- omicResults(rnaseq_simulation, "RNA-seq")

#           Group1.Time0.Rep1 Group1.Time0.Rep2 Group1.Time0.Rep3 ...
# ENSMUSG00000073155          4539             5374          5808 ...
# ENSMUSG00000026251              0              0              0 ...
# ENSMUSG00000040472          2742             2714          2912 ...
# ENSMUSG00000021598          5256             4640          5130 ...
# ENSMUSG00000032348           421              348           492 ...
# ENSMUSG00000097226            16              14            9 ...
# ENSMUSG00000027857              0              0              0 ...
# ENSMUSG00000032081              1              0              0 ...
# ENSMUSG00000097164           794             822           965 ...
# ENSMUSG00000097871              0              0              0 ...
```



---

omicSettings	<i>Retrieves the settings used in a simulation</i>
--------------	--

---

### Description

Retrieves the settings used in a simulation

### Usage

```
omicSettings(  
  simulation,  
  omics = NULL,  
  association = FALSE,  
  reverse = FALSE,  
  only.linked = FALSE,  
  prefix = FALSE,  
  include.lagged = TRUE  
)
```

### Arguments

simulation	A MOSimulation object.
omics	List of omics to retrieve the settings.
association	A boolean indicating if the association must also be returned for the regulators.
reverse	A boolean, swap the column order in the association list in case we want to use the output directly and the program requires a different ordering.
only.linked	Return only the interactions that have an effect.
prefix	Logical indicating if the name of the omic should prefix the name of the regulator.
include.lagged	Logical indicating if interactions with transitory profile and different minimum/maximum time point between gene and regulator should be included or not.

### Value

A list containing a data frame with the settings used to simulate each of the indicated omics. If association is TRUE, it will be a list with 3 keys: 'associations', 'settings' and 'regulators', with the first two keys being a list containing the information for the selected omics and the last one a global data frame giving the merged information.

### Examples

```
omic_list <- c("RNA-seq", "miRNA-seq")  
multi_simulation <- mosim(omics = omic_list)  
  
# This will be a data frame with RNA-seq settings (DE flag, profiles)  
rnaseq_settings <- omicSettings(multi_simulation, "RNA-seq")
```



---

plotProfile	<i>Generate a plot of a feature's profile for one or two omics.</i>
-------------	---

---

### Description

Generate a plot of a feature's profile for one or two omics.

### Usage

```
plotProfile(simulation, omics, featureIDS, drawReps = FALSE, groups = NULL)
```

### Arguments

simulation	A MOSimulation object
omics	Character vector of the omics to simulate.
featureIDS	List containing the feature to show per omic. Must have the omics as the list names and the features as values.
drawReps	Logical to enable/disable the representation of the replicates inside the plot.
groups	Character vector indicating the groups to plot in the form "GroupX" (i.e. Group1)

### Value

A ggplot2 object.

### Examples

```
omic_list <- c("RNA-seq", "miRNA-seq")

rnaseq_options <- c(omicSim("miRNA-seq", totalFeatures = 2500))
rnaseq_simulation <- mosim(omics = omic_list,
                          omicsOptions = rnaseq_options)

#plotProfile(rnaseq_simulation,
#  omics = c("RNA-seq", "miRNA-seq"),
#  featureIDS = list("RNA-seq"="ENSMUSG00000007682", "miRNA-seq"="mmu-miR-320-3p")
#)
```

---

`random_unif_interval` *random\_unif\_interval* Function to call the C code This function is a copy of the 'random\_unif\_interval' function from the 'SPARSim' package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license.

---

### Description

`random_unif_interval` Function to call the C code This function is a copy of the 'random\_unif\_interval' function from the 'SPARSim' package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license.

### Usage

```
random_unif_interval(size, max_val)
```

### Arguments

<code>size</code>	from sparsim
<code>max_val</code>	from sparsim

---

<code>sampleData</code>	<i>Default data</i>
-------------------------	---------------------

---

### Description

Dataset with base counts and id-gene tables.

### Usage

```
data("sampleData")
```

### Format

An object of class `list` of length 6.

### Details

List with 6 elements:

**SimRNAseq data** Dataframe with base counts with gene id as rownames.

**geneLength** Length of every gene.

**SimChIPseq data** Dataframe with base counts with regions as rownames.

**idToGene** Dataframe with region as "ID" column and gene name on "Gene" column.

- SimDNaseSeq data** Dataframe with base counts with regions as rownames.  
**idToGene** Dataframe with region as "ID" column and gene name on "Gene" column.
- SimMiRNASeq data** Dataframe with base counts with miRNA id as rownames.  
**idToGene** Dataframe with miRNA as "ID" column and gene name on "Gene" column.
- SimMethylSeq idToGene** Dataframe with region as "ID" column and gene name on "Gene" column.
- CpGisland** Dataframe of CpG to be used as initialization data, located on "Region" column

---

 scatac

*Data to test scMOSim*


---

**Description**

Data to test scMOSim

**Usage**

```
data("scatac")
```

**Format**

A seurat Object, subset from seuratData with ATAC

**assays** ATAC expression values**meta.data** annotations of celltypes

@source <https://github.com/satijalab/seurat-data>, we took 11 cells from each of 4 celltypes

---

 scrna

*Data to test scMOSim*


---

**Description**

Data to test scMOSim

**Usage**

```
data("scrna")
```

**Format**

A seurat Object, subset from seuratData with RNA

**assays** RNA expression values

**meta.data** annotations of celltypes

@source <https://github.com/satijalab/seurat-data>, we took 11 cells from each of 4 celltypes This is how: `dat <- pbmcMultiome.SeuratData::pbmc.rna dat <- subset(x = dat, subset = seurat_annotatons "cDC", "Memory B", "Treg")) unique_cell_types <- unique(dat@meta.data$seurat_annotatons) extracted_cells <- list() cellnames <- c() for (cell_type in unique_cell_types) type_cells <- subset(dat, subset = seurat_annotatons counts <- as.matrix(type_cells@assays[["RNA"]]@counts) extracted_cells[[cell_type]] <- counts[, 1:10] cellnames <- append(cellnames, replicate(11, cell_type)) scrna <- Reduce(cbind, extracted_cells)`

---

sc\_mosim

*sc\_mosim*

---

**Description**

Performs multiomic simulation of single cell datasets

**Usage**

```
sc_mosim(
  omics,
  cellTypes,
  numberReps = 1,
  numberGroups = 1,
  diffGenes = NULL,
  minFC = 0.25,
  maxFC = 4,
  numberCells = NULL,
  mean = NULL,
  sd = NULL,
  noiseRep = 0.1,
  noiseGroup = 0.5,
  regulatorEffect = NULL,
  associationList = NULL,
  feature_no = 8000,
  clusters = 3,
  cluster_size = NULL,
  TF = FALSE,
  TFdf = NULL
)
```

**Arguments**

omics	named list containing the omic to simulate as names, which can be "scRNA-seq" or "scATAC-seq".
cellTypes	list where the i-th element of the list contains the column indices for i-th experimental conditions. List must be a named list.
numberReps	OPTIONAL. Number of replicates per group
numberGroups	OPTIONAL. number of different groups
diffGenes	OPTIONAL. If number groups > 1, Percentage DE genes to simulate. List of vectors (one per group to compare to group 1) where the vector contains absolute number of genes for Up and Down ex: c(250, 500) or a percentage for up, down ex: c(0.2, 0.2). The rest will be NE
minFC	OPTIONAL. Threshold of FC below which are downregulated, by default 0.25
maxFC	OPTIONAL. Threshold of FC above which are upregulated, by default 4
numberCells	OPTIONAL. Vector of numbers. The numbers correspond to the number of cells the user wants to simulate per each cell type. The length of the vector must be the same as length of cellTypes.
mean	OPTIONAL. Vector of numbers of mean depth per each cell type. Must be specified just if numberCells is specified. The length of the vector must be the same as length of cellTypes.
sd	OPTIONAL. Vector of numbers of standard deviation per each cell type. Must be specified just if numberCells is specified. The length of the vector must be the same as length of cellTypes.
noiseRep	OPTIONAL. Number indicating the desired standard deviation between biological replicates.
noiseGroup	OPTIONAL. Number indicating the desired standard deviation between treatment groups
regulatorEffect	OPTIONAL. To simulate relationship scRNA-scATAC, list of vectors (one per group) where the vector contains absolute number of regulators for Activator and repressor ex: c(150, 200) or a percentage for Activator and repressor ex: c(0.2, 0.1). The rest will be NE. If not provided, no table of association between scRNA and scATAC is outputted.
associationList	REQUIRED A 2 columns dataframe reporting peak ids related to gene names. If user doesnt have one, load from package data("associationList")
feature_no	OPTIONAL. If only scRNA-seq to simulate or scRNA and scATAC but no regulatory constraints, total number of features to be distributed between the co-expression clusters.
clusters	OPTIONAL. Number of co-expression patterns the user wants to simulate
cluster_size	OPTIONAL. It may be inputted by the user. Recommended: by default, its the number of features divided by the number of patterns to generate.
TF	OPTIONAL default is FALSE, if true, extract TF dataframe
TFdf	OPTIONAL, default is NULL. If an association matrix of TF and Target_gene is given the TF expression values are extracted. If no data.frame is given, using the association of human TF from <a href="https://tflink.net/">https://tflink.net/</a>

**Value**

a list of Seurat object, one per each omic.

**Examples**

```
omic_list <- sc_omicData(list("scRNA-seq"))
cell_types <- list('Treg' = c(1:10), 'cDC' = c(11:20), 'CD4_TEM' = c(21:30),
'Memory_B' = c(31:40))
sim <- sc_mosim(omic_list, cell_types)
```

---

sc\_omicData

*sc\_omicData*


---

**Description**

Checks if the user defined data is in the correct format, or loads the default multiomics pbmc dataset, a subset from SeuratData package

**Usage**

```
sc_omicData(omics_types, data = NULL)
```

**Arguments**

omics_types	A list of strings which can be either "scRNA-seq" or "scATAC-seq"
data	A user input matrix with genes (peaks in case of scATAC-seq) as rows and cells as columns. By default, it loads the example data. If a user input matrix is included, cell columns must be sorted by cell type.

**Value**

a named list with omics type as name and the count matrix as value

**Examples**

```
# Simulate from PBMC
omicsList <- sc_omicData(list("scRNA-seq", "scATAC-seq"))
```



---

sc_omicResults	<i>sc_omicResults</i>
----------------	-----------------------

---

**Description**

sc\_omicResults

**Usage**

```
sc_omicResults(sim)
```

**Arguments**

sim                    a simulated object from sc\_mosim function

**Value**

list of seurat objects with simulated data

**Examples**

```
cell_types <- list('Treg' = c(1:10), 'cDC' = c(11:20), 'CD4_TEM' = c(21:30),  
  'Memory_B' = c(31:40))  
omicsList <- sc_omicData(list("scRNA-seq"))  
sim <- sc_mosim(omicsList, cell_types)  
res <- sc_omicResults(sim)
```

---

sc_omicSettings	<i>sc_omicSettings</i>
-----------------	------------------------

---

**Description**

sc\_omicSettings

**Usage**

```
sc_omicSettings(sim, TF = FALSE)
```

**Arguments**

sim                    a simulated object from sc\_mosim function  
TF                    OPTIONAL default is FALSE, if true, extract TF association matrix

**Value**

list of Association matrices explaining the effects of each regulator to each gene

**Examples**

```

cell_types <- list('Treg' = c(1:10), 'cDC' = c(11:20), 'CD4_TEM' = c(21:30),
'Memory_B' = c(31:40))
omicsList <- sc_omicData(list("scRNA-seq"))
sim <- sc_mosim(omicsList, cell_types)
res <- sc_omicSettings(sim)

```

---

```

sc_param_estimation  sc_param_estimation

```

---

**Description**

Evaluate the users parameters for single cell simulation and use SPARSim to simulate the main dataset. Internal function

**Usage**

```

sc_param_estimation(
  omics,
  cellTypes,
  diffGenes = list(c(0.2, 0.2)),
  minFC = 0.25,
  maxFC = 4,
  numberCells = NULL,
  mean = NULL,
  sd = NULL,
  noiseGroup = 0.5,
  group = 1,
  genereggroup
)

```

**Arguments**

omics	named list containing the omics to simulate as names, which can be "scRNA-seq" or "scATAC-seq".
cellTypes	list where the i-th element of the list contains the column indices for i-th cell type. List must be a named list.
diffGenes	If number groups > 1, Percentage DE genes to simulate. List of vectors (one per group to compare to group 1) where the vector contains absolute number of genes for Up and Down ex: c(250, 500) or a percentage for up, down ex: c(0.2, 0.2). The rest will be NE
minFC	Threshold of FC below which are downregulated, by default 0.25
maxFC	Threshold of FC above which are upregulated, by default 4
numberCells	vector of numbers. The numbers correspond to the number of cells the user wants to simulate per each cell type. The length of the vector must be the same as length of cellTypes.

mean	vector of numbers of mean depth per each cell type. Must be specified just if numberCells is specified.
sd	vector of numbers of standard deviation per each cell type. Must be specified just if numberCells is specified.
noiseGroup	OPTIONAL. Number indicating the desired standard deviation between treatment groups
group	Group for which to estimate parameters
genereggroup	List with information of genes, clusters and regulators that must be related to each other

### Value

a list of Seurat object, one per each omic.

a named list with simulation parameters for each omics as values.

### Examples

```
omicsList <- sc_omicData(list("scRNA-seq"))
cell_types <- list('Treg' = c(1:10), 'cDC' = c(11:20), 'CD4_TEM' = c(21:30),
'Memory_B' = c(31:40))
#estimated_params <- sc_param_estimation(omicsList, cell_types)
```

---

shuffle\_group\_matrix *shuffle\_group\_matrix, Reorder cell type-specific expression matrix during co-expression simulation. Copied from ACORDE (<https://github.com/ConesaLab/acorde>) to facilitate stability and running within our scripts This function is a slightly modified copy of the 'shuffle\_group\_matrix' function from the 'Acorde' package (v1.0.0), originally developed by Arzalluz-Luque A, Salguero P, Tarazona S, Conesa A. (2022). *acorde unravels functionally interpretable networks of isoform co-usage from single cell data. Nature communications* 1828. DOI: 10.1038/s41467-022-29497-w. The original package is licensed under the GPL-3 license.*

---

### Description

This function is used internally by acorde to perform the shuffling of simulated features for an individual cell type, as part of the co-expression simulation process. The function is called recursively by `simulate_coexpression()` to perform the simulation on a full scRNA-seq matrix.

### Usage

```
shuffle_group_matrix(sim_data, feature_ids, group_pattern, ngroups)
```

**Arguments**

sim_data	A count matrix with features as rows and cells as columns. Feature IDs must be included in an additional column named feature.
feature_ids	A two-column tibble containing top and bottom columns, each including the feature IDs of features to be used as highly or lowly expressed when shuffling by the indicated expression pattern.
group_pattern	A logical vector, containing TRUE to indicate that high expression in that cell type is desired and FALSE if the opposite. The vector must be ordered as the cell types in sim_data.
ngroups	An integer indicating the number of groups that top and bottom features should be divided into. It is computed by dividing the number of features selected as highly/lowly expressed by the size of the clusters that are to be generated.

**Value**

An expression matrix, with the same characteristics as sim\_data, and a number of features defined as the total amount of top/bottom features selected divided by the number of clusters for which co-expression patterns were supplied.

---

simulate\_coexpression *simulate coexpression*

---

**Description**

Adapted from ACORDE (<https://github.com/ConesaLab/acorde>) to adapt to our data input type. Simulates coexpression of genes along celltypes

**Usage**

```
simulate_coexpression(
  sim_matrix,
  feature_no,
  cellTypes,
  patterns,
  cluster_size = NULL
)
```

**Arguments**

sim_matrix	Matrix with rows as features and columns as cells
feature_no	Total number of features to be distributed between the coexpression clusters
cellTypes	list where the i-th element of the list contains the column indices for i-th experimental conditions. List must be a named list.
patterns	Tibble with TRUE FALSE depicting the cluster patterns to simulate. Generated by the user or by make_cluster_patterns.
cluster_size	OPTIONAL. It may be inputted by the user. By default, its the number of features divided by the number of patterns to generate.

**Details**

This function is a slightly modified copy of the ‘simulate\_coexpression’ function from the ‘Acorde’ package (v1.0.0), originally developed by Arzalluz-Luque A, Salguero P, Tarazona S, Conesa A. (2022). acorde unravels functionally interpretable networks of isoform co-usage from single cell data. Nature communications 1828. DOI: 10.1038/s41467-022-29497-w. The original package is licensed under the GPL-3 license.

**Value**

the simulated coexpression

---

simulate_hyper	<i>Simulate technical variability</i>
----------------	---------------------------------------

---

**Description**

Function to simulate the technical variability (i.e. a multivariate hypergeometric on a gamma expression value array)

**Usage**

```
simulate_hyper(avgAbund, seqdepth = NULL, digits, max_val)
```

**Arguments**

avgAbund	array containing the intensity values for each feature. It describes the intensity of a single sample
seqdepth	sequencing depth (i.e. sample size of the MH)
digits	number of digits for random number generation
max_val	max value for random number generation

**Details**

This function is a copy of the ‘simulate\_hyper’ function from the ‘SPARSim’ package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license.

**Value**

An array of length(avgAbund) elements representing the count values for the current sample

---

```
sparsim_create_simulation_parameter
```

*Create SPARSim simulation parameter*

---

### Description

Function to create a SPARSim simulation parameter. This function is a copy of the 'SPARSIM\_create\_simulation\_parameter' function from the 'SPARSim' package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license. To simulate N feature (e.g. genes), user must specify N values of gene expression level and gene expression variability in the function input parameters `intensity` and `variability`, respectively. To simulate M samples (i.e. cells), user must specify M values of sample library size in the function input parameter `library_size`.

### Usage

```
sparsim_create_simulation_parameter(
  intensity,
  variability,
  library_size,
  feature_names = NA,
  sample_names = NA,
  condition_name = NA,
  intensity_2 = NULL,
  variability_2 = NULL,
  p_bimod = NULL
)
```

### Arguments

<code>intensity</code>	Array of gene expression intensity values
<code>variability</code>	Array of gene expression variability values
<code>library_size</code>	Array of library size values
<code>feature_names</code>	Array of feature names. It must be of the same length of <code>intensity</code> array. If NA (default), feature will be automatically named "gene_1", "gene_2", ... "gene_<N>", where N = length(intensity)
<code>sample_names</code>	Array of sample names. It must be of the same length of <code>library_size</code> array. If NA (default), sample will be automatically named "<condition_name>_cell1", "<condition_name>_cell2", ..., "<condition_name>_cell<M>", where M = length(library_size)
<code>condition_name</code>	Name associated to the current experimental condition. If NA (default), it will be set to "cond<l1><l2>", where l1 and l2 are two random letters.
<code>intensity_2</code>	Array of gene expression intensity values for the second expression mode, if simulating genes with bimodal gene expression. Entries containing NAs will be ignored. If NULL (default), no bimodal gene expression is simulated.

variability_2	Array of gene expression variability values for the second expression mode, if simulating genes with bimodal gene expression. If NULL (default), no bimodal gene expression is simulated.
p_bimod	Array of bimodal gene expression probabilities; the $i$ -th value indicates the probability $p$ of the $i$ -th gene to be expressed in the first mode (i.e. the one specified in the $i$ -th entries of parameters <code>intensity</code> and <code>variability</code> ); with probability $1-p$ the $i$ -th gene will be expressed in the second mode (i.e. the one specified in the $i$ -th entries of parameters <code>intensity_2</code> and <code>variability_2</code> )

### Details

User can optionally specify the names to assign at the single feature and sample to simulate (function input parameters `feature_names` and `sample_names`, respectively, as well as the name of the experimental condition (function input parameter `condition_name`). If the user does not specify such information, the function will set some default values.

To simulate  $T$  different experimental conditions in a single count table, then  $T$  different simulation parameters must be created.

### Value

SPARSim simulation parameter describing one experimental condition

---

`sparsim_estimate_intensity`

*Estimate SPARSim "intensity" parameter*

---

### Description

Function to estimate the intensity values from the genes in data. The intensity is computed as mean of normalized counts for each gene.

### Usage

```
sparsim_estimate_intensity(data)
```

### Arguments

<code>data</code>	normalized count data matrix (gene on rows, samples on columns). <code>rownames(data)</code> must contain gene names.
-------------------	---

### Details

This function is a copy of the ‘`SPARSIM_estimate_intensity`’ function from the ‘`SPARSim`’ package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license. This function is used in `sparsim_estimate_parameter_from_data` to compute SPARSim "intensity" parameter, given a real count table as input. If the count table contains more than one experimental condition, then the function is applied to each experimental conditions.

**Value**

An array of intensity values having  $N_{\text{genes}}$  elements ( $N_{\text{genes}} = \text{nrow}(\text{data})$ ). Array entries are named with gene names.

---

sparsim\_estimate\_library\_size

*Estimate SPARSim "library size" parameter*

---

**Description**

Function to estimate the library sizes from the samples in data.

**Usage**

```
sparsim_estimate_library_size(data)
```

**Arguments**

data                      raw count data matrix (gene on rows, samples on columns)

**Details**

This function is a copy of the ‘SPARSIM\_estimate\_library\_size’ function from the ‘SPARSim’ package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license. This function is used in sparsim\_estimate\_parameter\_f to compute SPARSim "library size" parameter, given a real count table as input. If the count table contains more than one experimental condition, then the function is applied to each experimental conditions.

**Value**

An array of library size values having  $N_{\text{samples}}$  elements ( $N_{\text{samples}} = \text{ncol}(\text{data})$ )

---

sparsim\_estimate\_parameter\_from\_data

*Estimate SPARSim simulation parameter from a given count table*

---

**Description**

Function to estimate SPARSim simulation parameters (intensity, variability and library sizes) from a real count table. If the real count table contains more than one experimental condition, it is possible to estimate the parameters for each experimental condition.

**Usage**

```
sparsim_estimate_parameter_from_data(raw_data, norm_data, conditions)
```



**Arguments**

raw_data	count matrix (gene on rows, samples on columns) containing raw count data
norm_data	count matrix (gene on rows, samples on columns) containing normalized count data
conditions	list where the i-th element of the list contains the column indices for i-th experimental conditions. List must be a named list.

**Details**

This function is a copy of the ‘SPARSIM\_estimate\_parameter\_from\_data’ function from the ‘SPARSim’ package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license.

**Value**

A SPARSim simulation parameters

---

sparsim\_estimate\_variability

*Estimate SPARSim "variability" parameter*

---

**Description**

Function to estimate the variability values from the genes in data.

**Usage**

```
sparsim_estimate_variability(data)
```

**Arguments**

data	raw count data matrix (gene on rows, samples on columns)
------	--

**Details**

This function is a copy of the ‘SPARSIM\_estimate\_variability’ function from the ‘SPARSim’ package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license. This function is used in sparsim\_estimate\_parameter\_from\_data to compute SPARSim "variability" parameter, given a real count table as input. If the count table contains more than one experimental condition, then the function is applied to each experimental conditions.

**Value**

An array of variability values having N\_genes elements (N\_genes = nrow(data))

---

sparsim\_simulation      *Function to simulate a raw count table*

---

### Description

This function is a copy of the ‘SPARSIM\_simulation‘ function from the ‘SPARSim‘ package (v0.9.5), originally developed by Giacomo Baruzzo, Ilaria Patuzzi, Barbara Di Camillo (2020). The original package is licensed under the GPL-3 license.

### Usage

```
sparsim_simulation(
  dataset_parameter,
  output_sim_param_matrices = FALSE,
  output_batch_matrix = FALSE,
  count_data_simulation_seed = NULL
)
```

### Arguments

**dataset\_parameter**  
list containing, the intensity, variability and lib sizes of each experimental condition. It is the return value of "estimate\_parameter\_from\_data" or could be created by the users

**output\_sim\_param\_matrices**  
boolean flag. If TRUE, the function will output two additional matrices, called `abundance_matrix` and `variability_matrix`, containing the gene intensities and gene variabilities used as simulation input. (Default: FALSE)

**output\_batch\_matrix**  
boolean flag. If TRUE, the function will output an additional matrix, called `batch_factors_matrix`, containing the multiplicative factors used in batch effect simulation. (Default: FALSE)

**count\_data\_simulation\_seed**  
inherited from sparsim

### Value

A list of 5 elements:

- `count_matrix`: the simulated count matrix (genes on rows, samples on columns)
- `gene_matrix`: the simulated gene expression levels (genes on rows, samples on columns)
- `abundance_matrix`: the input gene intensity values provided as input (genes on rows, samples on columns), if `output_sim_param_matrices = TRUE`. NULL otherwise.
- `variability_matrix`: the input gene variability values provided as input (genes on rows, samples on columns), if `output_sim_param_matrices = TRUE`. NULL otherwise.
- `batch_factors_matrix`: the multiplicative factor used in batch generation (genes on rows, samples on columns), if `output_batch_matrix = TRUE`. NULL otherwise.

---

TF\_human

*Data to extract human TF*

---

**Description**

Data to extract human TF

**Usage**

```
data("TF_human")
```

**Format**

vector of gene names

**data.frame** gene names corresponding to TF and to Target genes

@source <https://tfink.net/>

# Index

- \* **datasets**
  - sampleData, 20
- \* **internal**
  - MOSim-package, 3
  - MOSimulation-class, 12
  - MOSimulator-class, 13
  - MOSimulatorRegion-class, 14
- associationList, 3
- calculate\_mean\_per\_list\_df, 4
- check\_patterns, 4
- discretize, 5
- experimentalDesign, 6
- is.declared, 6
- make\_association\_dataframe, 7
- make\_cluster\_patterns, 8
- match\_gene\_regulator, 8
- match\_gene\_regulator\_cluster, 9
- MOSim (MOSim-package), 3
- mosim, 10, 11
- MOSim-package, 3
- MOSimulation-class, 12
- MOSimulator-class, 13
- MOSimulatorRegion-class, 14
- omicData, 11, 15
- omicResults, 16
- omicSettings, 17
- omicSim, 11, 18
- plotProfile, 19
- random\_unif\_interval, 20
- sampleData, 20
- sc\_mosim, 22
- sc\_omicData, 24
- sc\_omicResults, 25
- sc\_omicSettings, 25
- sc\_param\_estimation, 26
- scatac, 21
- scrna, 21
- shuffle\_group\_matrix, 27
- SimChIPseq-class
  - (MOSimulatorRegion-class), 14
- SimDNaseseq-class
  - (MOSimulatorRegion-class), 14
- SimMethylseq-class
  - (MOSimulatorRegion-class), 14
- SimmiRNAseq-class
  - (MOSimulatorRegion-class), 14
- SimRNAseq-class
  - (MOSimulatorRegion-class), 14
- SimTF-class (MOSimulatorRegion-class), 14
- simulate\_coexpression, 28
- simulate\_coexpression(), 27
- simulate\_hyper, 29
- sparsim\_create\_simulation\_parameter, 30
- sparsim\_estimate\_intensity, 31
- sparsim\_estimate\_library\_size, 32
- sparsim\_estimate\_parameter\_from\_data, 32
- sparsim\_estimate\_variability, 33
- sparsim\_simulation, 34
- TF\_human, 35