Package 'MOSim'

June 1, 2023

Title Multi-Omics Simulation (MOSim)

Version 1.15.0

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, methods, matrixStats, dplyr, stringi, lazyeval, rlang, stats, utils, purrr, scales, stringr, tibble, tidyr, ggplot2, Biobase, IRanges, S4Vectors

Suggests testthat, knitr, rmarkdown, BiocStyle

Collate 'AllClass.R' 'AllGeneric.R' 'Simulator.R' 'SimulatorRegion.R' 'ChIP-seq.R' 'DNase-seq.R' 'functions.R' 'Simulation.R' 'MOSim.R' 'RNA-seq.R' 'simulate_WGBS_functions.R' 'methyl-seq.R' 'miRNA-seq.R' 'zzz.R'

RoxygenNote 7.2.3

VignetteBuilder knitr

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

git_branch devel

git_last_commit 16d2ffc

git_last_commit_date 2023-04-25

Date/Publication 2023-06-01

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

Maintainer Sonia Tarazona <sotacam@gmail.com>

R topics documented:

`MOSim-package` *MOSim*

Description

Multiomics simulation package.

`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.

`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`	List containing the options to simulate each omic. We recommend to apply the helper method omicSim to create this list in a friendly way, and the function omicData to provide custom data (see the related sections for more information). Each omic may have different configuration parameters, but the common ones are:

simuData/idToGene Seed sample and association tables for regulatory omics. The helper function omicData should be used to provide this information (see the following section).

regulatorEffect 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.

totalFeatures Number of features to simulate. By default, the total number of features in the seed dataset.

depth Sequencing depth in millions of reads. If not provided, it takes the global parameter passed to mosim function.

replicateParams List with parameters *a* and *b* for adjusting the variability in the generation of replicates using the negative binomial. See vignette for more information.

`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 experimetal 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 % or 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. Overwrited 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 *Set customized data for an omic.*

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
## ENSMUSG00000000001   6572
## ENSMUSG00000000003      0
## ENSMUSG00000000028   4644
## ENSMUSG00000000031      8
## ENSMUSG00000000037      0
## ENSMUSG00000000049      0


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

omicResults *Retrieves the simulated data.*

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 specifiec, 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` *Retrieves the settings used in a simulation*

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")
```

```
# This will be a list containing all the simulated omics (RNA-seq
# and DNase-seq in this case)
all_settings <- omicSettings(multi_simulation)
```

`omicSim` *Set the simulation settings for an omic.*

Description

Set the simulation settings for an omic.

Usage

```
omicSim(omic, depth = NULL, totalFeatures = NULL, regulatorEffect = NULL)
```

Arguments

`omic`	Name of the omic to set the settings.
`depth`	Sequencing depth in millions of counts. If not provided will take the global parameter passed to mosim function.
`totalFeatures`	Limit the number of features to simulate. By default include all present in the dataset.
`regulatorEffect`	only for regulatory omics. Associative list containing the percentage of effects over the total number of regulator, including repressor, association and no effect (NE).

Value

A list with the appropiate structure to be given as options in mosim function.

Examples

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

rnaseq_options <- c(omicSim("miRNA-seq", totalFeatures = 2500))

# The return value is an associative list compatible with
# 'omicsOptions'
rnaseq_simulation <- mosim(omics = omic_list,
                           omicsOptions = rnaseq_options)
```

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

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")
)
```

`sampleData` *Default data*

Description

Dataset with base counts and id-gene tables.

Usage

```
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

Index