Package 'sRACIPE'

February 20, 2026

Type Package

Title Systems biology tool to simulate gene regulatory circuits

Version 2.2.0

Description sRACIPE implements a randomization-based method for gene circuit modeling. It allows us to study the effect of both the gene expression noise and the parametric variation on any gene regulatory circuit (GRC) using only its topology, and simulates an ensemble of models with random kinetic parameters at multiple noise levels. Statistical analysis of the generated gene expressions reveals the basin of attraction and stability of various phenotypic states and their changes associated with intrinsic and extrinsic noises. sRACIPE provides a holistic picture to evaluate the effects of both the stochastic nature of cellular processes and the parametric variation.

License MIT + file LICENSE

Encoding UTF-8

Suggests knitr, BiocStyle, rmarkdown, tinytest

VignetteBuilder knitr

LinkingTo Rcpp

biocViews ResearchField, SystemsBiology, MathematicalBiology, GeneExpression, GeneRegulation, GeneTarget

Depends R (>= 3.6.0),SummarizedExperiment, methods, Rcpp

Imports ggplot2, reshape2, MASS, RColorBrewer, gridExtra,visNetwork, gplots, umap, htmlwidgets, S4Vectors, BiocGenerics, grDevices, stats, utils, graphics, doFuture, doRNG, future, foreach

RoxygenNote 7.3.2

URL https://github.com/lusystemsbio/sRACIPE, https://geneex.jax.org/, https://vivekkohar.github.io/sRACIPE/

BugReport https://github.com/lusystemsbio/sRACIPE/issues

LazyData true

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

git_branch RELEASE_3_22

git_last_commit 26d0307

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-02-19

Author Mingyang Lu [aut, cre] (ORCID: <https://orcid.org/0000-0001-8158-0593>),
Vivek Kohar [aut],
Aidan Tillman [aut],
Daniel Ramirez [aut]

Maintainer Mingyang Lu <m.lu@northeastern.edu>

Contents

.loadNetworkFile . . . 3
.ModelPvalue . . . 3
.NthMin . . . 4
.PermutedVar . . . 4
.SimulatedPValueAbs . . . 5
.SimulatedVarPValue . . . 5
allTypesDemoCircuit . . . 6
annotation,RacipeSE-method . . . 6
annotation<-,RacipeSE,ANY-method . . . 7
cellCycle . . . 7
configData . . . 8
CoupledToggleSwitchSA . . . 8
demoCircuit . . . 9
densityPlot . . . 9
EMT1 . . . 10
EMT2 . . . 10
RacipeSE . . . 11
RacipeSE-class . . . 12
repressilator . . . 12
sRACIPE . . . 13
sracipeCircuit . . . 14
sracipeCircuit<- . . . 14
sracipeCombineRacipeSE . . . 15
sracipeConfig . . . 16
sracipeConfig<- . . . 17
sracipeConverge . . . 17
sracipeConvergeDist . . . 18
sracipeGenParamNames . . . 19
sracipeGetTS . . . 19
sracipeHeatmapSimilarity . . . 20
sracipeIC . . . 21
sracipeIC<- . . . 22
sracipeKnockDown . . . 23
sracipeNormalize . . . 24
sracipeOverExp . . . 25
sracipeParams . . . 26
sracipeParams<- . . . 27
sracipePlotCircuit . . . 28
sracipePlotData . . . 29
sracipePlotParamBifur . . . 30
sracipeSimulate . . . 31
sracipeUniqueStates . . . 36

Index **38**

.loadNetworkFile *Loads the network/topology file.*

Description

The network file should contain three columns with headers, "Source" "Target" "Type" Here "Source" and "Target" are the names of the genes and "Type" refers to the regulation, "1" if source activates target and "2" if source inhibits target.

Usage

```
.loadNetworkFile(networkFile = "inputs/test.net")
```

Arguments

networkFile	Network file name

Value

Network as a dataframe

.ModelPvalue *Returns the variance array after permutations.*

Description

A utility function.

Usage

```
.ModelPvalue(
  dataSimulation,
  dataReference,
  clusterCut,
  permutations,
  refClusterVar,
  corMethod,
  simulatedClusterVar
)
```

Arguments

dataSimulation	Simulation data to be compared.
dataReference	Reference data with which to compare.
clusterCut	The original cluster assignments.
permutations	The number of permutations.
refClusterVar	SD of the clusters.
corMethod	Correlation method to be used.
simulatedClusterVar	Variance of simulated clusters

Value

An array of dimension n.models by nClusters by permutations.

`.NthMin` *Find nth minimum value from a vector*

Description

A utility function to find the nth minimum

Usage

```
.NthMin(x, index)
```

Arguments

`x`	the given unsorted vector
`index`	N.

Value

the nth minimum element of the vector

`.PermutedVar` *Find variance of permutations*

Description

A utility function to generate permutations

Usage

```
.PermutedVar(simulated.refCor, clusterCut, permutations, refClusterVar)
```

Arguments

`simulated.refCor`	Correlation matrix of simulated and reference data
`clusterCut`	The original cluster assignments
`permutations`	The number of permutations
`refClusterVar`	Reference Cluster Variance

Value

An array of dimension n.models by nClusters by permutations

.SimulatedPValueAbs *Finds the variance corresponding to a given value.*

Description

A utility function to calculate the absolute p values.

Usage

```
.SimulatedPValueAbs(permutedVar, simulatedClusterVar)
```

Arguments

permutedVar	An array containing the distance of clusters for each model for every permutation.
simulatedClusterVar	Variance of simulated clusters

Value

p-values for each model.

.SimulatedVarPValue *Finds the variance corresponding to a given value.*

Description

A utility function for calculating the p values.

Usage

```
.SimulatedVarPValue(permutedVar, pValue)
```

Arguments

permutedVar	An array containing the distance of clusters for each model for every permutation.
pValue	Cut off p vlaue.

Value

p-values for each model.

allTypesDemoCircuit *A circuit with every interaction type for simulations*

Description

This data contains an artificial 4-gene circuit with genes A, B, C, D which contains every possible type of gene interaction simulated by the sRACIPE package. The circuit is designed so that each pair of genes has only one type of interaction and each interaction appears only once, which makes it useful for testing.

Usage

```
allTypesDemoCircuit
```

Format

A data frame with 6 rows and 3 variables

annotation,RacipeSE-method *A method to get the annotation*

Description

A method to get the annotation

Usage

```
## S4 method for signature 'RacipeSE'
annotation(object, ...)
```

Arguments

object	RacipeSE object
...	Additional arguments, for use in specific methods.

Value

character

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 100)
ann <- annotation(rSet)
annotation(rSet) <- ann
```

annotation<-,RacipeSE,ANY-method *A method to set the circuit name or annotation*

Description

A method to set the circuit name or annotation

Usage

```
## S4 replacement method for signature 'RacipeSE,ANY'
annotation(object, ...) <- value
```

Arguments

object	RacipeSE object
...	Additional arguments, for use in specific methods.
value	annotation character

Value

A RacipeSE object

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 100)
ann <- annotation(rSet)
annotation(rSet) <- ann
```

cellCycle *A circuit for modeling the yeast cell cycle*

Description

This data contains a circuit with 15 nodes representing genes involved with the yeast cell cycle. For further details, see Katebi A, Kohar V, Lu M. Random Parametric Perturbations of Gene Regulatory Circuit Uncover State Transitions in Cell Cycle. iScience. 2020 Jun 26;23(6):101150. doi: 10.1016/j.isci.2020.101150. Epub 2020 May 11. PMCID: PMC7251928.

Usage

```
cellCycle
```

Format

A data frame with 38 rows and 3 variables.

configData *Configuration Data*

Description

It contains simulation parameters like integration method (stepper) and other lists or vectors like simParams, stochParams, hyperParams, options, thresholds etc. The list simParams contains values for parameters like the number of models (numModels), number of convergence iterations to run for simulations with convergence tests (numConvergenceIter), simulation time (simulationTime), step size for simulations (integrateStepSize), when to start recording the gene expressions (printStart), time interval between recordings (printInterval), number of initial conditions (nIC), output precision (outputPrecision), tolerance for adaptive runge kutta method (rkTolerance), parametric variation (paramRange). The list stochParams contains the parameters for stochastic simulations like the number of noise levels to be simulated (nNoise), the ratio of subsequent noise levels (noiseScalingFactor), maximum noise (initialNoise), whether to use same noise for all genes or to scale it as per the median expression of the genes (scaledNoise), ratio of shot noise to additive noise (shotNoise). The list hyperParams contains the parameters like the minimum and maximum production and degration of the genes, fold change, hill coefficient etc. The list options includes logical values like annealing (anneal), scaling of noise (scaledNoise), generation of new initial conditions (genIC), parameters (genParams), whether to integrate or not (integrate), and whether or not to check for limitcycles (limitcycles). The user modifiable simulation options can be specified as other arguments. This list should be used if one wants to modify many settings for multiple simulations.

Usage

```
configData
```

Format

list

CoupledToggleSwitchSA *Five coupled toggle switches*

Description

This data contains the topology of a circuit with ten genes A1...A5 and B1...B5. Genes with different alphabet inhibit each other whereas genes from different groups activate the other. For further details, see Kohar, V. and Lu, M., Role of noise and parametric variation in the dynamics of gene regulatory circuits, npj Systems Biology and Applications 4, Article number: 40 (2018)

Usage

```
CoupledToggleSwitchSA
```

Format

A data frame with 18 rows and 3 variables

demoCircuit *A toggle switch circuit for demonstrations*

Description

This data contains the topology of a circuit with two genes A and B both of which inhibit each other and have self activations. The interactions are transcription factor interactions. For further details, see Kohar, V. and Lu, M., Role of noise and parametric variation in the dynamics of gene regulatory circuits, npj Systems Biology and Applications 4, Article number: 40 (2018)

Usage

```
demoCircuit
```

Format

A data frame with 4 rows and 3 variables

densityPlot *Density Plot*

Description

Plot the density of points as an image alongwith histograms on the sides.

Usage

```
densityPlot(plotData, binCount = 40, plotColor = NULL, ...)
```

Arguments

plotData	Dataframe containing the data.
binCount	(optional) Integer. Default 40. The number of bins to be used for dividing the data along an axis.
plotColor	(optional) The color palette.
...	any additional arguments

Value

plot

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData, sracipeHeatmapSimilarity

EMT1 *A circuit for epithelial to mesenchymal transition*

Description

This data contains the topology of a circuit with sixteen genes invovled in EMT. For further details, see Kohar, V. and Lu, M., Role of noise and parametric variation in the dynamics of gene regulatory circuits, npj Systems Biology and Applications 4, Article number: 40 (2018)

Usage

```
EMT1
```

Format

A data frame with 59 rows and 3 variables

EMT2 *A circuit for epithelial to mesenchymal transition including microRNAs*

Description

This data contains the topology of a circuit with twenty two nodes including micro RNAs invovled in EMT. For further details, see Huang et al.,Interrogating the topological robustness of gene regulatory circuits by randomization, PLoS computational biology 13 (3), e1005456

Usage

```
EMT2
```

Format

A data frame with 82 rows and 3 variables

RacipeSE *RacipeSE constructor*

Description

Create an RacipeSE object. RacipeSE is an S4 class for Random Circuit Perturbation (RACIPE) simulations of networks in which a large number of models with randomized parameters are used for simulation of the circuit. Each model can be considered as a sample. It extends the SummarizedExperiment class to store and access the circuit, simulated gene expressions, parameters, intial conditions and other meta information. SummarizedExperiment slot assays is used for storing simulated gene expressions. The rows of these matrix-like elements correspond to various genes in the circuit and columns correspond to models. The first element is used for unperturbed deterministic simulations. The subsequent elements are used for stochastic simulations at different noise levels and/or knockout simulations. SummarizedExperiment slot rowData stores the circuit topology. It is a square matrix with dimension equal to the number of genes in the circuit. The values of the matrix represent the type of interaction in the gene pair given by row and column. 1 represents transcriptional activation, 2 transcriptional inhibition, 3 activation by inhibiting degradation, 4 inhibition by activating degradation, 5 signaling activation, 6 signaling inhibition, and 0 no interaction. This should not be set directly and `sracipeCircuit` accessor should be used instead. SummarizedExperiment slot colData contains the parameters and initial conditions for each model. Each gene in the circuit has two parameters, namely, its production rate and its degradation rate. Each interaction in the has three parameters, namely, threshold of activation, the hill coefficient, and the fold change. Each gene has one or more initial gene expression values as specified by nIC. This should not be modified directly and `sracipeParams` and `sracipeIC` accessors should be used instead. SummarizedExperiment slot metadata Contains metadata information especially the config list (containing the simulation settings), annotation, nInteraction (number of interactions in the circuit), normalized (whether the data is normalized or not), data analysis lists like pca, umap, cluster assignment of the models etc. The config list includes simulation parameters like integration method (stepper) and other lists or vectors like simParams, stochParams, hyperParams, options, thresholds etc. The list simParams contains values for parameters like the number of models (numModels), simulation time (simulationTime), step size for simulations (integrateStepSize), when to start recording the gene expressions (printStart), time interval between recordings (printInterval), number of initial conditions (nIC), output precision (outputPrecision), tolerance for adaptive runge kutta method (rkTolerance), parametric variation (paramRange). The list stochParams contains the parameters for stochastic simulations like the number of noise levels to be simulated (nNoise), the ratio of subsequent noise levels (noiseScalingFactor), maximum noise (initialNoise), whether to use same noise for all genes or to scale it as per the median expression of the genes (scaledNoise), ratio of shot noise to additive noise (shotNoise). The list hyperParams contains the parameters like the minimum and maximum production and degration of the genes, fold change, hill coefficient etc. The list options includes logical values like annealing (anneal), scaling of noise (scaledNoise), generation of new initial conditions (genIC), parameters (genParams) and whether to integrate or not (integrate). The user modifiable simulation options can be specified as arguments to `sracipeSimulate` function.

Usage

```
RacipeSE(
  .object = NULL,
  assays = SimpleList(),
  rowData = NULL,
  colData = DataFrame(),
  metadata = list(),
```

```
  ...
)
```

Arguments

`.object`	(optional) Another RacipeSE object.
`assays`	(optional) assay object for initialization
`rowData`	(optional) rowData for initialization
`colData`	(optional) colData for initialization
`metadata`	(optional) metadata for initialization
`...`	Arguments passed to SummarizedExperiment

Value

RacipeSE object

Examples

```
rSet <- RacipeSE()
```

`RacipeSE-class` *RacipeSE*

Description

An S4 class for Random Circuit Perturbation (RACIPE) simulations of networks. Extends the SummarizedExperiment class. RACIPE can simulate a gene regulatory circuit using the circuit and a large ensemble of parameters.

`repressilator` *A loop motif for demonstrating limit cycles in gene expression*

Description

This data contains the topology for a repressilator circuit of 3 genes cyclically inhibiting each other with transcription factor regulation. This is a simple circuit that can generate limit cycles in gene expression in roughly 5

Usage

```
repressilator
```

Format

A data frame with 3 rows and 3 variables.

sRACIPE *sRACIPE: A package for stochastic random circuit perturbation.*

Description

sRACIPE is a systems biology tool to study the role of noise and parameter variation in gene regulatory circuits. It implements a randomization-based method for gene circuit modeling. It allows us to study the effect of both the gene expression noise and the parametric variation on any gene regulatory circuit (GRC) using only its topology, and simulates an ensemble of models with random kinetic parameters at multiple noise levels. Statistical analysis of the generated gene expressions reveals the basin of attraction and stability of various phenotypic states and their changes associated with intrinsic and extrinsic noises. sRACIPE provides a holistic picture to evaluate the effects of both the stochastic nature of cellular processes and the parametric variation.

sRACIPE functions

`sracipeSimulate` Primary function to simulate a circuit. Contains options for plotting as well. `sracipeKnockDown` In-silico knockdown analysis of the circuit. Plots the relative changes in different cluster proportions.

`sracipeOverExp` In-silico over expression analysis of the circuit. Plots the relative changes in different cluster proportions.

`sracipePlotData` Plot the simulated data. Includes options to plot the hierarchichal clustering analysis, principal components, and uniform manifold approximation and projection. Can plot the stochastic as well as the knockout simulations.

`sracipeSimulate`, `sracipeKnockDown`, `sracipeOverExp`, `sracipePlotData`

Author(s)

Maintainer: Mingyang Lu <m.lu@northeastern.edu> (ORCID)

Authors:

- Vivek Kohar
- Aidan Tillman
- Daniel Ramirez

See Also

Useful links:

- https://github.com/lusystemsbio/sRACIPE
- https://geneex.jax.org/
- https://vivekkohar.github.io/sRACIPE/

sracipeCircuit *Method to get the circuit*

Description

The circuit file should contain three columns with headers, "Source" "Target" "Type" Here "Source" and "Target" are the names of the genes and "Type" refers to the regulation, "1" for TF activation and "2" for TF inhibition, "3" for degradation inhibition, "4" for degradation activation, "5" for signaling activation, and "6" for signaling inhibition.

Usage

```
sracipeCircuit(.object)

## S4 method for signature 'RacipeSE'
sracipeCircuit(.object)
```

Arguments

.object	RacipeSE object

Value

A dataframe

Examples

```
rs <- RacipeSE()
data("demoCircuit")
sracipeCircuit(rs) <- demoCircuit
circuitDataFrame <- sracipeCircuit(rs)
rm(rs, demoCircuit,circuitDataFrame)
```

sracipeCircuit<- *Initialize the circuit*

Description

Initialize the circuit from a topology file or a `data.frame` A typical topology file looks like

Source	Target	Type
geneA	geneB	2
geneB	geneC	1
geneB	geneA	2

Here the regulation type is specified by number - TF activation: `1`, TF inhibition: `2` degradation inhibition: `3` degradation activation: `4` signaling activation: `5` signaling inhibition: `6`

Usage

```
sracipeCircuit(.object) <- value

## S4 replacement method for signature 'RacipeSE'
sracipeCircuit(.object) <- value
```

Arguments

.object	RacipeSE object
value	data.frame containing the circuit information

Value

data.frame

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData

Examples

```
RacipeSet <- RacipeSE()
data("demoCircuit")
sracipeCircuit(RacipeSet) <- demoCircuit
sracipeCircuit(RacipeSet)
rm(RacipeSet, demoCircuit)
```

---

sracipeCombineRacipeSE

A method to combine RacipeSE objects with the same config

Description

It is often the case that to save time, multiple simulations of the same topology with the same config file are run in parallel. However, one still wants to condense the results of the different simulations into one object. This method does that, combining the params, ics, and expressions of a list of objects into one. If convergence testing is done, convergence data and unique state counts are combined as well. Limit cycle data is combined too. The method validates the RacipeSE objects as having essentially identical configurations compared to the first object provided

Usage

```
sracipeCombineRacipeSE(.object)

## S4 method for signature 'list'
sracipeCombineRacipeSE(.object)
```

Arguments

.object	RacipeSE object generated by sracipeSimulate function.

Value

RacipeSE object

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData

Examples

```
data("demoCircuit")
## Not run:
  data(democircuit)
  for(i in 1:5){
    rSet <- sracipeSimulate(demoCircuit, numModels = 100,
                                numConvergenceIter  = 20, nIC = 2)
    racipeList <- c(racipeList, results)
  }

  combinedObject <- sracipeCombineRacipeSE(racipeList)

## End(Not run)
```

---

sracipeConfig *A method to access the simulation hyperparameters*

Description

The hyperparameters like number of models, range from which parameters are to be sampled, simulation time etc.

Usage

```
sracipeConfig(.object)

## S4 method for signature 'RacipeSE'
sracipeConfig(.object)
```

Arguments

.object	RacipeSE object

Value

list

Examples

```
RacipeSet <- RacipeSE()
data("demoCircuit")
sracipeCircuit(RacipeSet) <- demoCircuit
sracipeConfig(RacipeSet)
rm(RacipeSet)
```

sracipeConfig<- *A method to access the simulation hyperparameters*

Description

The hyperparameters like number of models, range from which parameters are to be sampled, simulation time etc.

Usage

```
sracipeConfig(.object) <- value

## S4 replacement method for signature 'RacipeSE'
sracipeConfig(.object) <- value
```

Arguments

.object	RacipeSE object
value	list. Configuration as a list

Value

RacipeSE object

Examples

```
rSet <- RacipeSE()
tmpConfig <- sracipeConfig(rSet)
sracipeConfig(rSet) <- tmpConfig
rm(rSet, tmpConfig)
```

sracipeConverge *A method to get the convergence results for deterministic simulations*

Description

Gathers the convergence and speed of convergence for each model and condition for deterministic simulations

Usage

```
sracipeConverge(.object)

## S4 method for signature 'RacipeSE'
sracipeConverge(.object)
```

Arguments

.object	RacipeSE object

Value

DataFrame

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20)
cd <- sracipeIC(rSet)
rm(rSet, cd)
```

`sracipeConvergeDist` *A method to visualize convergence distributions*

Description

When convergence tests are done (deterministic systems with time series off), this method creates a plot of the proportion of converged initial conditions as the number of convergence tests increases, up to the total number done by the simulation. Note that when limit cycles are detected, they are automatically removed from consideration. This method also adds some statistics to the metadata of the input. Models with NaN values are also removed. Specifically, the final proportion of converged states is reported, and if it exists, the smallest number of convergence tests where at least 99 percent of models converged is also reported.

Usage

```
sracipeConvergeDist(.object, plotToFile = FALSE)

## S4 method for signature 'RacipeSE'
sracipeConvergeDist(.object, plotToFile = FALSE)
```

Arguments

`.object`	RacipeSE object generated by `sracipeSimulate` function.
`plotToFile`	(optional) logical. Default `FALSE`. Whether to save plots to a file.

Value

`RacipeSE` object

Related Functions

`sracipeSimulate`, `sracipeKnockDown`, `sracipeOverExp`, `sracipePlotData`

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrateStepSize = 0.1, numConvergenceIter = 30)
rSet <- sracipeConvergeDist(rSet)

## End(Not run)
```

sracipeGenParamNames *Generate parameter names for a circuit*

Description

Generate parameter names for a circuit

Usage

```
sracipeGenParamNames(circuit = "inputs/test.tpo")
```

Arguments

circuit	RacipeSE object or topology as data.frame or filename

Value

list

Examples

```
rSet <- RacipeSE()
data("demoCircuit")
sracipeCircuit(rSet) <- demoCircuit
paramNames <- sRACIPE::sracipeGenParamNames(rSet)
```

sracipeGetTS *A method to extract the time series*

Description

If timeSeries option is used in sracipeSimulate function, this method will return the simulated time series.

Usage

```
sracipeGetTS(.object)

## S4 method for signature 'RacipeSE'
sracipeGetTS(.object)
```

Arguments

.object	RacipeSE object

Value

List

Examples

```
data("demoCircuit")
RacipeSet <- RacipeSE()
sracipeCircuit(RacipeSet) <- demoCircuit
RacipeSet <- sracipeSimulate(demoCircuit, timeSeries = TRUE,
simulationTime = 2)
trajectories <- sracipeGetTS(RacipeSet)
rm(RacipeSet)
```

`sracipeHeatmapSimilarity` *Calculates the similarity between two gene expression data.*

Description

Comparison is done across columns, i.e., how similar are the columns in the two dataset. For gene expression data, format data so that gene names are in rows and samples in columns.

Usage

```
sracipeHeatmapSimilarity(
  dataReference,
  dataSimulation,
  clusterCut = NULL,
  nClusters = 3,
  pValue = 0.05,
  permutedVar,
  permutations = 1000,
  corMethod = "spearman",
  clusterMethod = "ward.D2",
  method = "pvalue",
  buffer = 0.001,
  permutMethod = "simulation",
  returnData = FALSE
)
```

Arguments

`dataReference`	Matrix. The reference data matrix, for example, the experimental gene expression values
`dataSimulation`	Matrix. The data matrix to be compared.
`clusterCut`	(optional) Integer vector. Clsuter numbers assigned to reference data. If clusterCut is missing, hierarchical clustering using /codeward.D2 and /codedistance = (1-cor(x, method = "spear"))/2 will be used to cluster the reference data.
`nClusters`	(optional) Integer. The number of clusters in which the reference data should be clustered for comparison. Not needed if clusterCut is provided.
`pValue`	(optional) Numeric. p-value to consider two gene expression sets as belonging to same cluster. Ward's method with spearman correlation is used to determine if a model belongs to a specific cluster.
`permutedVar`	(optional) Similarity scores computed after permutations.

permutations	(optional) Integer. Default 1000. Number of gene permutations to generate the null distibution.
corMethod	(optional) Correlation method. Default method is "spearman". For single cell data, use "kendall"
clusterMethod	(optional) Character - default ward.D2, other options include complete. Clustering method to be used to cluster the experimental data. hclust for other options.
method	(optional) character. Method to compare the gene expressions. Default pvalue. One can use variance as well which assigns clusters based on the cluster whose samples have minimum variance with the simulated sample.
buffer	(optional) Numeric. Default 0.001. The fraction of models to be assigned to clusters to which no samples could be assigned. For example, a minimum of 1 ghost sample in reference is assigned to NULL cluster.
permutMethod	"sample" or "reference"
returnData	(optional) Logical. Default FALSE. Whether to return the sorted and clustered data.

Value

A list containing the KL distance of new cluster distribution from reference data and the probability of each cluster in the reference and simulated data.

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData, sracipeHeatmapSimilarity

---

sracipeIC *A method to get the initial conditions used for simulations*

Description

The initial conditions of each of the models.

Usage

```
sracipeIC(.object)

## S4 method for signature 'RacipeSE'
sracipeIC(.object)
```

Arguments

.object	RacipeSE object

Value

DataFrame

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrate=FALSE)
ics <- sracipeIC(rSet)
rm(rSet,ics)
```

`sracipeIC<-` *A method to set the initial conditions*

Description

Set the initial conditions

Usage

```
sracipeIC(.object) <- value

## S4 replacement method for signature 'RacipeSE'
sracipeIC(.object) <- value
```

Arguments

`.object`	RacipeSE object
`value`	DataFrame containing the initial conditions, dimensions should be (# genes) rows by (numModels*nIC) columns

Value

A RacipeSE object

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 10,
integrate=FALSE)
ics <- sracipeIC(rSet)
sracipeIC(rSet) <- ics
rm(rSet, ics)
```

`sracipeKnockDown` *Perform in-silico knockdown analysis*

Description

Calculate the fraction of models in different clusters with full parameter range and on a subset of models with low production rate of a specific gene representing the knockdown of the specific gene.

Usage

```
sracipeKnockDown(
  .object,
  reduceProduction = 10,
  nClusters = 2,
  clusterOfInterest = 2,
  plotFilename = NULL,
  plotHeatmap = TRUE,
  plotBarPlot = TRUE,
  clusterCut = NULL,
  plotToFile = FALSE
)

## S4 method for signature 'RacipeSE'
sracipeKnockDown(
  .object,
  reduceProduction = 10,
  nClusters = 2,
  clusterOfInterest = 2,
  plotFilename = NULL,
  plotHeatmap = TRUE,
  plotBarPlot = TRUE,
  clusterCut = NULL,
  plotToFile = FALSE
)
```

Arguments

`.object`	RacipeSE object generated by `sracipeSimulate` function.
`reduceProduction`	(optional) Percentage to which production rate decreases on knockdown. Uses a default value of 10 percent.
`nClusters`	(optional) Number of clusters in the data. Uses a default value of 2.
`clusterOfInterest`	(optional) cluster number (integer) to be used for arranging the transcription factors
`plotFilename`	(optional) Name of the output file.
`plotHeatmap`	logical. Default TRUE. Whether to plot the heatmap or not.
`plotBarPlot`	logical. Default TRUE. Whether to plot the barplot.
`clusterCut`	integer or character. The cluster assignments.
`plotToFile`	logical. Default FALSE.

Value

List containing fraction of models in different clusters in the original simulations and after knowcking down different genes. Additionaly, it generates two pdf files in the results folder. First is barplot showing the percentage of different clusters in the original simulations and after knocking down each gene. The second pdf contains the heatmap of clusters after marking the models with cluster assignments.

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 100,
plots=FALSE, plotToFile = FALSE)
rSet <- sRACIPE::sracipeNormalize(rSet)
rSet <- sRACIPE::sracipeKnockDown(rSet, plotToFile = FALSE,
plotBarPlot = TRUE, plotHeatmap = FALSE, reduceProduction = 50)

## End(Not run)
```

---

sracipeNormalize *Normalize the simulated gene expression*

Description

Normalize the simulated gene expression including gene expressions for stochastic and knockout simulations

Usage

```
sracipeNormalize(.object)

## S4 method for signature 'RacipeSE'
sracipeNormalize(.object)
```

Arguments

.object	RacipeSE object

Value

A RacipeSE object

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData,

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrateStepSize = 0.1, simulationTime = 30)
rSet <- sracipeNormalize(rSet)
```

`sracipeOverExp` *Perform in-silico over expression analysis*

Description

Calculates the fraction of models in different clusters with full parameter range and on a subset of models with high production rate of a specific gene representing the over expression of the specific gene.

Usage

```
sracipeOverExp(
  .object,
  overProduction = 10,
  nClusters = 2,
  clusterOfInterest = 2,
  plotFilename = NULL,
  plotHeatmap = TRUE,
  plotBarPlot = TRUE,
  clusterCut = NULL,
  plotToFile = FALSE
)

## S4 method for signature 'RacipeSE'
sracipeOverExp(
  .object,
  overProduction = 10,
  nClusters = 2,
  clusterOfInterest = 2,
  plotFilename = NULL,
  plotHeatmap = TRUE,
  plotBarPlot = TRUE,
  clusterCut = NULL,
  plotToFile = FALSE
)
```

Arguments

`.object`	RacipeSE object generated by `sracipeSimulate` function.
`overProduction`	(optional) Percentage to which production rate decreases on knockdown. Uses a default value of 10 percent.
`nClusters`	(optional) Number of clusters in the data. Uses a default value of 2.
`clusterOfInterest`	(optional) cluster number (integer) to be used for arranging the transcription factors

plotFilename	(optional) Name of the output file.
plotHeatmap	logical. Default TRUE. Whether to plot the heatmap or not.
plotBarPlot	logical. Default TRUE. Whether to plot the barplot.
clusterCut	integer or character. The cluster assignments.
plotToFile	logical. Default FALSE.

Value

List containing fraction of models in different clusters in the original simulations and after knowcking down different genes. Additionaly, it generates two pdf files in the results folder. First is barplot showing the percentage of different clusters in the original simulations and after knocking down each gene. The second pdf contains the heatmap of clusters after marking the models with cluster assignments.

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData,

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 100,
plots=FALSE, plotToFile = FALSE)
rSet <- sRACIPE::sracipeNormalize(rSet)

## End(Not run)
```

---

sracipeParams *A method to access the simulation parameters*

Description

The parameters for each model.

Usage

```
sracipeParams(.object)

## S4 method for signature 'RacipeSE'
sracipeParams(.object)
```

Arguments

.object	RacipeSE object

Value

A data.frame

Examples

```
data("demoCircuit")
RacipeSet <- sracipeSimulate(demoCircuit, integrate = FALSE, numModels=20)
parameters <- sracipeParams(RacipeSet)
sracipeParams(RacipeSet) <- parameters
rm(parameters,RacipeSet)
```

---

sracipeParams<- *A method to set the simulation parameters*

Description

Set the parameters

Usage

```
sracipeParams(.object) <- value

## S4 replacement method for signature 'RacipeSE'
sracipeParams(.object) <- value
```

Arguments

.object	RacipeSE object
value	DataFrame containing the parameters. Dimensions should be (numModels) rows by (# of parameters) columns.

Value

A RacipeSE object

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrate = FALSE)
parameters <- sracipeParams(rSet)
sracipeParams(rSet) <- parameters
rm(parameters, rSet)
```

`sracipePlotCircuit` *Plot Gene Regulatory Circuit*

Description

Plot Gene Regulatory Circuit to a file or output device using visNetwork. Edge color coding: Transcription-"black", Protein Degradation-"red", Signaling Interaction-"green".

Usage

```
sracipePlotCircuit(
  .object,
  plotToFile = FALSE,
  physics = TRUE,
  namedNodes = TRUE
)

## S4 method for signature 'RacipeSE'
sracipePlotCircuit(
  .object,
  plotToFile = TRUE,
  physics = TRUE,
  namedNodes = TRUE
)
```

Arguments

`.object`	RacipeSE object A list returned by `sracipeSimulate` function
`plotToFile`	(optional) logical. Default `FALSE`. Whether to save plots to a file.
`physics`	(optional) logical. Default `TRUE`. Whether or not to enable physics in the nodes of the visNetwork graph.
`namedNodes`	(optional) logical. Default `TRUE`. Whether or not to display gene names in the circuit visualization

Value

circuit plot

Related Functions

`sracipeSimulate`, `sracipeKnockDown`, `sracipeOverExp`, `sracipePlotData`

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrateStepSize = 0.1, simulationTime = 30)
sracipePlotCircuit(rSet, plotToFile = FALSE, physics = TRUE)
rm(rSet)

## End(Not run)
```

`sracipePlotData` *Plot sRACIPE data*

Description

Plots heatmap, pca, umap of the data simulated using sRACIPE

Usage

```
sracipePlotData(
  .object,
  plotToFile = FALSE,
  nClusters = 2,
  heatmapPlot = TRUE,
  pcaPlot = TRUE,
  umapPlot = TRUE,
  networkPlot = TRUE,
  clustMethod = "ward.D2",
  col = col,
  distType = "euclidean",
  assignedClusters = NULL,
  corMethod = "spearman",
  ...
)

## S4 method for signature 'RacipeSE'
sracipePlotData(
  .object,
  plotToFile = TRUE,
  nClusters = 2,
  heatmapPlot = TRUE,
  pcaPlot = TRUE,
  umapPlot = TRUE,
  networkPlot = TRUE,
  clustMethod = "ward.D2",
  col = col,
  distType = "euclidean",
  assignedClusters = NULL,
  corMethod = "spearman",
  ...
)
```

Arguments

`.object`	List A list returned by `sracipeSimulate` function
`plotToFile`	(optional) logical. Default `FALSE`. Whether to save plots to a file.
`nClusters`	(optional) Integer. Default 2. Expected number of clusters in the simulated data. Hierarchical clustering will be used to cluster the data and the the models will be colored in UMAP and PCA plots according to these clustering results. The clusters can be also supplied using `assignedClusters`.

heatmapPlot	(optional) logical. Default TRUE. Whether to plot hierarchichal clustering.
pcaPlot	(optional) logical. Default TRUE. Whether to plot PCA embedding.
umapPlot	(optional) logical. Default TRUE. Whether to plot UMAP embedding
networkPlot	(optional) logical. Default TRUE. Whether to plot the network.
clustMethod	(optional) character. Default "ward.D2". Clustering method for heatmap. See heatmap.2
col	(optional) Color palette
distType	(optional) Distance type. Used only if specified explicitly. Otherwise, 1-cor is used. See dist, hclust
assignedClusters	vector integer or character. Default NULL. Cluster assignment of models.
corMethod	(optional) character. Default "spearman". Correlation method for distance function.
...	Other arguments

Value

RacipeSE object

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData,

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrateStepSize = 0.1, simulationTime = 30)
rSet <- sracipePlotData(rSet)

## End(Not run)
```

---

sracipePlotParamBifur *Parameter bifurcation plots*

Description

Plot the expression of the genes against parameter values to understand the effect of parameters on the gene expressions.

Usage

```
sracipePlotParamBifur(
  .object,
  paramName,
  data = NULL,
  paramValue = NULL,
  assignedClusters = NULL,
  plotToFile = FALSE
```

```
)

## S4 method for signature 'RacipeSE'
sracipePlotParamBifur(
  .object,
  paramName,
  data = NULL,
  paramValue = NULL,
  assignedClusters = NULL,
  plotToFile = FALSE
)
```

Arguments

.object	RacipeSE object generated by sracipeSimulate function.
paramName	character. The name of the parameter to be plotted.
data	(optional) dataframe. Default rSet geneExpression. The data to be plotted. For example, use rSet$stochasticSimulations$[noise] to plot the stochastic data.
paramValue	(optional) Dataframe. The parameter values if rSet$params is not to be used.
assignedClusters	(optional) Dataframe. The cluster assignment of data.
plotToFile	(optional) logical. Default FALSE. Whether to save plots to a file.

Value

none

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 100,
plots=FALSE, plotToFile = FALSE)
rSet <- sRACIPE::sracipeNormalize(rSet)
sracipePlotParamBifur(rSet, "G_A")

## End(Not run)
```

---

sracipeSimulate *Simulate a gene regulatory circuit*

Description

Simulate a gene regulatory circuit using its topology as the only input. It will generate an ensemble of random models.

Usage

```
sracipeSimulate(
  circuit = "inputs/test.tpo",
  config = config,
  anneal = FALSE,
  knockOut = NA_character_,
  numModels = 2000,
  paramRange = 100,
  prodRateMin = 1,
  prodRateMax = 100,
  degRateMin = 0.1,
  degRateMax = 1,
  foldChangeMin = 1,
  foldChangeMax = 100,
  hillCoeffMin = 1L,
  hillCoeffMax = 6L,
  integrateStepSize = 0.02,
  simulationTime = 50,
  nIC = 1L,
  nNoise = 0L,
  simDet = TRUE,
  initialNoise = 50,
  noiseScalingFactor = 0.5,
  shotNoise = 0,
  ouNoise_t = 1,
  scaledNoise = FALSE,
  outputPrecision = 12L,
  printStart = 50,
  printInterval = 10,
  stepper = "RK4",
  thresholdModels = 5000,
  plots = FALSE,
  plotToFile = FALSE,
  genIC = TRUE,
  genParams = TRUE,
  integrate = TRUE,
  rkTolerance = 0.01,
  timeSeries = FALSE,
  signalRate = 10,
  uniqueDigits = 4,
  convergThresh = 1e-12,
  numStepsConverge = 500,
  numConvergenceIter = 25,
  limitcycles = FALSE,
  LCSimTime = 10,
  LCSimStepSize = 0.01,
  maxLCs = 10,
  LCIter = 20,
  maxPeriods = 100,
  numSampledPeriods = 3,
  allowedPeriodError = 3,
  samePointProximity = 0.1,
```

```
  LCStepper = "RK4",
  paramSignalVals = data.frame(),
  geneClamping = data.frame(),
  nCores = 1L,
  ...
)
```

Arguments

circuit	data.frame or character. The file containing the circuit or
config	(optional) List. It contains simulation parameters like integration method (stepper) and other lists or vectors like simParams, stochParams, hyperParams, options, thresholds etc. The list simParams contains values for parameters like the number of models (numModels), simulation time (simulationTime), step size for simulations (integrateStepSize), when to start recording the gene expressions (printStart), time interval between recordings (printInterval), number of initial conditions (nIC), output precision (outputPrecision), tolerance for adaptive runge kutta method (rkTolerance), parametric variation (paramRange). The list stochParams contains the parameters for stochastic simulations like the number of noise levels to be simulated (nNoise), the ratio of subsequent noise levels (noiseScalingFactor), maximum noise (initialNoise), whether to use same noise for all genes or to scale it as per the median expression of the genes (scaledNoise), ratio of shot noise to additive noise (shotNoise). The list hyperParams contains the parameters like the minimum and maximum production and degration of the genes, fold change, hill coefficient etc. The list options includes logical values like annealing (anneal), scaling of noise (scaledNoise), generation of new initial conditions (genIC), parameters (genParams) and whether to integrate or not (integrate). The user modifiable simulation options can be specified as other arguments. This list should be used if one wants to modify many settings for multiple simulations.
anneal	(optional) Logical. Default FALSE. Whether to use annealing for stochastic simulations. If TRUE, the gene expressions at higher noise are used as initial conditions for simulations at lower noise.
knockOut	(optional) List of character or vector of characters. Simulation after knocking out one or more genes. To knock out all the genes in the circuit, use `knockOut = "all"`. If it is a vector, then all the genes in the vector will be knocked out simultaneously.
numModels	(optional) Integer. Default 2000. Number of random models to be simulated.
paramRange	(optional) numeric (0-100). Default 100. The relative range of parameters (production rate, degradation rate, fold change).
prodRateMin	(optional) numeric. Default 1. Minimum production rate.
prodRateMax	(optional) numeric. Default 100. Maximum production rate.
degRateMin	(optional) numeric. Default 0.1. Minimum degradation rate.
degRateMax	(optional) numeric. Default 1. Maximum degradation rate.
foldChangeMin	(optional) numeric. Default 1. Minimum fold change for interactions.
foldChangeMax	(optional) numeric. Default 100. Maximum fold change for interactions.
hillCoeffMin	(optional) integer. Default 1. Minimum hill coefficient.
hillCoeffMax	(optional) integer. Default 6. Maximum hill coefficient.

`integrateStepSize`
: (optional) numeric. Default 0.02. step size for integration using "EM" and "RK4" steppers.

`simulationTime`
: (optional) numeric. Default 50. Total simulation time. Only used for stochastic and time series simulations. For adjusting deterministic simulation run times, see numConvergenceIter.

`nIC`
: (optional) integer. Default 1. Number of initial conditions to be simulated for each model.

`nNoise`
: (optional) integer. Default 0. Number of noise levels at which simulations are to be done. Use nNoise = 1 if simulations are to be carried out at a specific noise. If nNoise > 0, simulations will be carried out at nNoise levels as well as for zero noise. "EM" stepper will be used for simulations and any argument for stepper will be ignoired.

`simDet`
: (optional) logical. Default TRUE. Whether to simulate at zero noise as well also when using nNoise > 0.

`initialNoise`
: (optional) numeric. Default 50/sqrt(number of genes in the circuit). The initial value of noise for simulations. The noise value will decrease by a factor `noiseScalingFactor` at subsequent noise levels.

`noiseScalingFactor`
: (optional) numeric (0-1) Default 0.5. The factor by which noise will be decreased when nNoise > 1.

`shotNoise`
: (optional) numeric. Default 0. The ratio of shot noise to additive noise.

`ouNoise_t`
: (optional) numeric. Default 1. Correlation time parameter for OU noise. Only used when "EM_OU" stepper is selected

`scaledNoise`
: (optional) logical. Default FALSE. Whether to scale the noise in each gene by its expected median expression across all models. If TRUE the noise in each gene will be proportional to its expression levels.

`outputPrecision`
: (optional) integer. Default 12. The decimal point precison of the output.

`printStart`
: (optional) numeric (0-`simulationTime`). Default `simulationTime`. To be used only when `timeSeries` is `TRUE`. The time from which the output should be recorded. Useful for time series analysis and studying the dynamics of a model for a particular initial condition.

`printInterval`
: (optional) numeric (`integrateStepSize`- `simulationTime` - `printStart`). Default 10. The separation between two recorded time points for a given trajectory. To be used only when `timeSeries` is `TRUE`.

`stepper`
: (optional) Character. Stepper to be used for integrating the differential equations. The options include `"EM"` for Euler-Maruyama O(1), `"RK4"` for fourth order Runge-Kutta O(4) and `"DP"` for adaptive stepper based Dormand-Prince algorithm. The default method is `"RK4"` for deterministic simulations and the method defaults to `"EM"` for stochastic simulations.

`thresholdModels`
: (optional) integer. Default 5000. The number of models to be used for calculating the thresholds for genes.

`plots`
: (optional) logical Default `FALSE`. Whether to plot the simuated data.

`plotToFile`
: (optional) Default `FALSE`. Whether to save the plots to a file.

`genIC`
: (optional) logical. Default `TRUE`. Whether to generate the initial conditions. If `FALSE`, the initial conditions must be supplied as a dataframe to `circuit$ic`.

`genParams` (optional) logical. Default `TRUE`. Whether to generate the parameters. If `FALSE`, the parameters must be supplied as a dataframe to `circuit$params`.

`integrate` (optional) logical. Default `TRUE`. Whether to integrate the differential equations or not. If `FALSE`, the function will only generate the parameters and initial conditions. This can be used iteratively as one can fist generate the parameters and initial conditions and then modify these before using these modified values for integration. For example, this can be used to knockOut genes by changing the production rate and initial condition to zero.

`rkTolerance` (optional) numeric. Default `0.01`. Error tolerance for adaptive integration method.

`timeSeries` (optional) logical. Default `FALSE`. Whether to generate time series for a single model instead of performing RACIPE simulations.

`signalRate` (optional) numeric. Default `10`. The factor by which the differential equations for fast genes are multiplied by. Fast genes only have inward signaling interactions of types 5 or 6

`uniqueDigits` (optional) integer. Default `4`. Deterministic simulations with nIC > 1 count the number of unique steady states per model by truncating the expression by this many digits and counting the number of unique expressions

`convergThresh` (optional) numeric. Default `1e-12`. The threshold for convergence to a steady state for deterministic simulations.

`numStepsConverge` (optional) integer Default `500`. The number of integration steps between convergence tests for deterministic simulations.

`numConvergenceIter` (optional) integer. Default `25`. The total number of convergence test iterations to run per model initial condition in deterministic simulations.

`limitcycles` (optional) logical. Default `FALSE`. Whether to check for limit cycles in deterministic simulations.

`LCSimTime` (optional) numeric. Default `10`. The length of each iteration for the secondary limit cycle simulation

`LCSimStepSize` (optional) numeric. Default `0.01`. The integration step size for the secondary limit cycle simulation.

`maxLCs` (optional) integer. Default `10`. The maximum allowable number of limit cycles that can be detected for each model.

`LCIter` (optional) integer. Default `20`. The number of iterations to run in the secondary limit cycle simulation

`maxPeriods` (optional) integer. Default `100`. The number of periods to count in the distance function constructed by the limit cycle detection algorithm

`numSampledPeriods` (optional) integer. Default `3`. The number of times the limit cycle detection algorithm tries to calculate the period of the simulated trajectory using the local minima in a constructed distance function.

`allowedPeriodError` (optional) integer. Default `3`. The allowed difference in the sampled periods for a trajectory from the limit cycle detection algorithm. Decrease this value to make for more stringent limit cycle detection

`samePointProximity` (optional) numeric. Default `0.1`. The max allowed square euclidean difference between two minima of the distance function constructed by the limit cycle detection algorithm before they are considered different points in phase space.

LCStepper	(optional) Character. Default "RK4" The integration method used for the limit cycle simulation. The options include "E" for first order Euler and "RK4" for fourth order Runge Kutta. Any other input will cause the simulation to use the Euler method
paramSignalVals	(optional) Data Frame. Default data.frame(). The first column must be a vector of time values with the first element as 0 and the last element as simulationTime. The other column names must be valid production or degradation parameter names for the circuit. Use sracipeGenParamNames() to generate valid parameter names. The first column must be a vector of time values with the first element as 0 and the last element as simulationTime.
geneClamping	(optional) Data Frame. Default data.frame(). The column names must be genes in the circuit. The number of columns must either be one or equal to numModels. If the number of columns is one, the selected genes are clamped to those values for every model. Otherwise, the gene is clamped to the value of the corresponding row for a particular model. If genIC is set to FALSE while geneClamping is provided, the provided clamps will override the original initial conditions.
nCores	(optional) integer. Default 1 Number of cores to be used for computation. Utilizes multisession from doFuture pacakge, as well and doRNG package.
...	Other arguments

Value

RacipeSE object. RacipeSE class inherits SummarizedExperiment and contains the circuit, parameters, initial conditions, simulated gene expressions, and simulation configuration. These can be accessed using correponding getters.

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData

Examples

```
data("demoCircuit")
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit)
```

---

sracipeUniqueStates *A method for grabbing unique states*

Description

This method selects the unique expression states for every model in an sRACIPE object. The method of evaluating uniqueness is the same as in the main sracipeSimulate function.Non-converged expressions are filtered out using the simulation convergence data. This method should only be used for deterministic simulations with nIC > 1.

Usage

```
sracipeUniqueStates(.object)

## S4 method for signature 'RacipeSE'
sracipeUniqueStates(.object)
```

Arguments

.object	RacipeSE object generated by sracipeSimulate function.

Value

list object. Element i of the list is a data frame containing the unique expressions of model i in the input RacipeSE object

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeConvergeDist, sracipePlotData

Examples

```
data("demoCircuit")
## Not run:
rSet <- sRACIPE::sracipeSimulate(circuit = demoCircuit, numModels = 20,
integrateStepSize = 0.1, numConvergenceIter = 30)
stateList <- sracipeUniqueStates(rSet)
comginedStates <- do.call(cbind, stateList)

## End(Not run)
```

Index

∗ **datasets**
allTypesDemoCircuit, 6
cellCycle, 7
configData, 8
CoupledToggleSwitchSA, 8
demoCircuit, 9
EMT1, 10
EMT2, 10
repressilator, 12
.ModelPvalue, 3
.NthMin, 4
.PermutedVar, 4
.RacipeSE *(*RacipeSE-class*)*, 12
.SimulatedPValueAbs, 5
.SimulatedVarPValue, 5
.loadNetworkFile, 3

allTypesDemoCircuit, 6
annotation,RacipeSE-method, 6
annotation<-,RacipeSE,ANY-method, 7

cellCycle, 7
configData, 8
CoupledToggleSwitchSA, 8

demoCircuit, 9
densityPlot, 9
dist, *30*

EMT1, 10
EMT2, 10

hclust, *21*, *30*
heatmap.2, *30*

RacipeSE, 11
RacipeSE-class, 12
repressilator, 12

sRACIPE, 13
sRACIPE-package *(*sRACIPE*)*, 13
sracipeCircuit, *11*, 14
sracipeCircuit,RacipeSE-method *(*sracipeCircuit*)*, 14
sracipeCircuit-set *(*sracipeCircuit<-*)*, 14
sracipeCircuit<-, 14
sracipeCircuit<-,RacipeSE-method *(*sracipeCircuit<-*)*, 14
sracipeCombineRacipeSE, 15
sracipeCombineRacipeSE,list-method *(*sracipeCombineRacipeSE*)*, 15
sracipeConfig, 16
sracipeConfig,RacipeSE-method *(*sracipeConfig*)*, 16
sracipeConfig-set *(*sracipeConfig<-*)*, 17
sracipeConfig<-, 17
sracipeConfig<-,RacipeSE-method *(*sracipeConfig<-*)*, 17
sracipeConverge, 17
sracipeConverge,RacipeSE-method *(*sracipeConverge*)*, 17
sracipeConvergeDist, 18, *37*
sracipeConvergeDist,RacipeSE-method *(*sracipeConvergeDist*)*, 18
sracipeGenParamNames, 19
sracipeGetTS, 19
sracipeGetTS,RacipeSE-method *(*sracipeGetTS*)*, 19
sracipeHeatmapSimilarity, *9*, 20, *21*
sracipeIC, *11*, 21
sracipeIC,RacipeSE-method *(*sracipeIC*)*, 21
sracipeIC-set *(*sracipeIC<-*)*, 22
sracipeIC<-, 22
sracipeIC<-,RacipeSE-method *(*sracipeIC<-*)*, 22
sracipeKnockDown, *9*, *13*, *15*, *16*, *18*, *21*, 23, *24*, *26*, *28*, *30*, *36*, *37*
sracipeKnockDown,RacipeSE-method *(*sracipeKnockDown*)*, 23
sracipeNormalize, 24
sracipeNormalize,RacipeSE-method *(*sracipeNormalize*)*, 24
sracipeOverExp, *9*, *13*, *15*, *16*, *18*, *21*, *24*, 25, *26*, *28*, *30*, *36*
sracipeOverExp,RacipeSE-method

(sracipeOverExp), 25
sracipeParams, *11*, 26
sracipeParams,RacipeSE-method
(sracipeParams), 26
sracipeParams-set (sracipeParams<-), 27
sracipeParams<-, 27
sracipeParams<-,RacipeSE-method
(sracipeParams<-), 27
sracipePlotCircuit, 28
sracipePlotCircuit,RacipeSE-method
(sracipePlotCircuit), 28
sracipePlotData, *9*, *13*, *15, 16*, *18*, *21*, *24*, *26*, *28*, 29, *30*, *36, 37*
sracipePlotData,RacipeSE-method
(sracipePlotData), 29
sracipePlotParamBifur, 30
sracipePlotParamBifur,RacipeSE-method
(sracipePlotParamBifur), 30
sracipeSimulate, *9*, *11*, *13*, *15, 16*, *18*, *21*, *23–26*, *28–31*, 31, *36, 37*
sracipeUniqueStates, 36
sracipeUniqueStates,RacipeSE-method
(sracipeUniqueStates), 36
SummarizedExperiment, *11, 12*