

# Package ‘sRACIPE’

October 2, 2024

**Type** Package

**Title** Systems biology tool to simulate gene regulatory circuits

**Version** 1.21.0

**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, doFuture

**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

**RoxygenNote** 7.1.0

**URL** <https://vivekkohar.github.io/sRACIPE/>,  
<https://github.com/vivekkohar/sRACIPE>, <https://geneex.jax.org/>

**BugReport** <https://github.com/vivekkohar/sRACIPE/issues>

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

**git\_branch** devel

**git\_last\_commit** 7c85e5b

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-10-01

**Author** Vivek Kohar [aut, cre] (<<https://orcid.org/0000-0003-1813-1597>>),  
Mingyang Lu [aut]

**Maintainer** Vivek Kohar <[vivek.kohar@gmail.com](mailto:vivek.kohar@gmail.com)>

## Contents

.loadNetworkFile	2
.ModelPvalue	3
.NthMin	4
.PermutedVar	4
.SimulatedPValueAbs	5
.SimulatedVarPValue	5
annotation,RacipeSE-method	6
annotation<- ,RacipeSE,ANY-method	6
configData	7
CoupledToggleSwitchSA	7
demoCircuit	8
densityPlot	8
EMT1	9
EMT2	9
RacipeSE	10
RacipeSE-class	11
sRACIPE	11
sracipeCircuit	12
sracipeCircuit<-	13
sracipeConfig	14
sracipeConfig<-	14
sracipeGenParamNames	15
sracipeGetTS	15
sracipeHeatmapSimilarity	16
sracipeIC	17
sracipeIC<-	18
sracipeKnockDown	19
sracipeNormalize	20
sracipeOverExp	21
sracipeParams	22
sracipeParams<-	23
sracipePlotCircuit	23
sracipePlotData	24
sracipePlotParamBifur	26
sracipeSimulate	27

<b>Index</b>	<b>31</b>
--------------	-----------

---

.loadNetworkFile	<i>Loads the network/topology file.</i>
------------------	---

---

### 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	<i>Returns the variance array after permutations.</i>
--------------	---

---

**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

<code>x</code>	the given unsorted vector
<code>index</code>	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

<code>simulated.refCor</code>	Correlation matrix of simulated and reference data
<code>clusterCut</code>	The original cluster assignments
<code>permutations</code>	The number of permutations
<code>refClusterVar</code>	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.

---

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
```

---

 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), 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 degradation 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.

**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	<i>A toggle switch circuit for demonstrations</i>
-------------	---

---

**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. 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	<i>Density Plot</i>
-------------	---------------------

---

**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 involved 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 involved 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

**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, initial 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 activation, 2 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 degradation 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	<i>RacipeSE</i>
----------------	-----------------

---

**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.

---

sRACIPE	<i>sRACIPE: A package for stochastic random circuit perturbation.</i>
---------	---

---

**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 hierarchical clustering analysis, principal components, and uniform manifold approximation and projection. Can plot the stochastic as well as the knockout simulations.

[sracipeSimulate](#), [sracipeKnockDown](#), [sracipeOverExp](#), [sracipePlotData](#)

---

sracipeCircuit	<i>Method to get the circuit</i>
----------------	----------------------------------

---

**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" if source activates target and "2" if source inhibits target.

**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 - activation: 1, inhibition: 2

## 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)
```

---

sracipeConfig	<i>A method to access the simulation hyperparameters</i>
---------------	--

---

**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<-	<i>A method to access the simulation hyperparameters</i>
-----------------	--

---

**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)
```

---

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 distribution.
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. <a href="#">hclust</a> 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

**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)
```

---

<code>sracipeKnockDown</code>	<i>Perform in-silico knockdown analysis</i>
-------------------------------	---

---

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

### Arguments

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

**Value**

List containing fraction of models in different clusters in the original simulations and after knocking down different genes. Additionally, 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	<i>Normalize the simulated gene expression</i>
------------------	--

---

**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	<i>Perform in-silico over expression analysis</i>
----------------	---

---

**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 <code>sracipeSimulate</code> 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 knocking down different genes. Additionally, 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

**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.

**Usage**

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

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

**Arguments**

.object            RacipeSE object A list returned by [sracipeSimulate](#) function  
 plotToFile        (optional) logical. Default FALSE. Whether to save plots to a file.

**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)
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 = FALSE,
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 <a href="#">sracipeSimulate</a> 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 <a href="#">heatmap.2</a>
col	(optional) Color palette
distType	(optional) Distance type. Used only if specified explicitly. Otherwise, 1-cor is used. See <a href="#">dist</a> , <a href="#">hclust</a>
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 'RecipeSE'
sracipePlotParamBifur(
  .object,
  paramName,
  data = NULL,
  paramValue = NULL,
  assignedClusters = NULL,
  plotToFile = FALSE
)

```

## Arguments

.object	RecipeSE object generated by <a href="#">sracipeSimulate</a> 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,
  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,
nCores = 1L,
...
)

```

## Arguments

<code>circuit</code>	data.frame or character. The file containing the circuit or
<code>config</code>	(optional) List. It contains simulation parameters like integration method (step-per) and other lists or vectors like <code>simParams</code> , <code>stochParams</code> , <code>hyperParams</code> , <code>options</code> , <code>thresholds</code> etc. The list <code>simParams</code> contains values for parameters like the number of models ( <code>numModels</code> ), simulation time ( <code>simulationTime</code> ), step size for simulations ( <code>integrateStepSize</code> ), when to start recording the gene expressions ( <code>printStart</code> ), time interval between recordings ( <code>printInterval</code> ), number of initial conditions ( <code>nIC</code> ), output precision ( <code>outputPrecision</code> ), tolerance for adaptive runge kutta method ( <code>rkTolerance</code> ), parametric variation ( <code>paramRange</code> ). The list <code>stochParams</code> contains the parameters for stochastic simulations like the number of noise levels to be simulated ( <code>nNoise</code> ), the ratio of subsequent noise levels ( <code>noiseScalingFactor</code> ), maximum noise ( <code>initialNoise</code> ), whether to use same noise for all genes or to scale it as per the median expression of the genes ( <code>scaledNoise</code> ), ratio of shot noise to additive noise ( <code>shotNoise</code> ). The list <code>hyperParams</code> contains the parameters like the minimum and maximum production and degradation of the genes, fold change, hill coefficient etc. The list <code>options</code> includes logical values like <code>annealing</code> ( <code>anneal</code> ), scaling of noise ( <code>scaledNoise</code> ), generation of new initial conditions ( <code>genIC</code> ), parameters ( <code>genParams</code> ) and whether to integrate or not ( <code>integrate</code> ). 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.
<code>anneal</code>	(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.
<code>knockOut</code>	(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 <code>knockOut = "all"</code> . If it is a vector, then all the genes in the vector will be knocked out simultaneously.
<code>numModels</code>	(optional) Integer. Default 2000. Number of random models to be simulated.
<code>paramRange</code>	(optional) numeric (0-100). Default 100. The relative range of parameters (production rate, degradation rate, fold change).
<code>prodRateMin</code>	(optional) numeric. Default 1. Minimum production rate.
<code>prodRateMax</code>	(optional) numeric. Default 100. Maximum production rate.
<code>degRateMin</code>	(optional) numeric. Default 0.1. Minimum degradation rate.
<code>degRateMax</code>	(optional) numeric. Default 1. Maximum degradation rate.
<code>foldChangeMin</code>	(optional) numeric. Default 1. Minimum fold change for interactions.
<code>foldChangeMax</code>	(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. Total simulation time.
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 ignored.
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.
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 precision 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 simulated 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.

<code>integrate</code>	(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 first 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.
<code>rkTolerance</code>	(optional) numeric. Default 0.01. Error tolerance for adaptive integration method.
<code>timeSeries</code>	(optional) logical. Default FALSE. Whether to generate time series for a single model instead of performing RACIPE simulations.
<code>nCores</code>	(optional) integer. Default 1. Number of cores to be used for computation. Utilizes <code>multiprocess</code> from <code>doFuture</code> package. Will not work in Rstudio.
<code>...</code>	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 corresponding getters.

### Related Functions

[sracipeSimulate](#), [sracipeKnockDown](#), [sracipeOverExp](#), [sracipePlotData](#)

### Examples

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

# Index

- \* **datasets**
  - configData, 7
  - CoupledToggleSwitchSA, 7
  - demoCircuit, 8
  - EMT1, 9
  - EMT2, 9
  - .ModelPvalue, 3
  - .NthMin, 4
  - .PermutedVar, 4
  - .RacipeSE (RacipeSE-class), 11
  - .SimulatedPValueAbs, 5
  - .SimulatedVarPValue, 5
  - .loadNetworkFile, 2
- annotation, RacipeSE-method, 6
- annotation<-, RacipeSE, ANY-method, 6
- configData, 7
- CoupledToggleSwitchSA, 7
- demoCircuit, 8
- densityPlot, 8
- dist, 25
- EMT1, 9
- EMT2, 9
- hclust, 17, 25
- heatmap.2, 25
- RacipeSE, 10
- RacipeSE-class, 11
- sRACIPE, 11
- sracipeCircuit, 10, 12
- sracipeCircuit, RacipeSE-method (sracipeCircuit), 12
- sracipeCircuit-set (sracipeCircuit<-), 13
- sracipeCircuit<-, 13
- sracipeCircuit<-, RacipeSE-method (sracipeCircuit<-), 13
- sracipeConfig, 14
- sracipeConfig, RacipeSE-method (sracipeConfig), 14
- sracipeConfig-set (sracipeConfig<-), 14
- sracipeConfig<-, 14
- sracipeConfig<-, RacipeSE-method (sracipeConfig<-), 14
- sracipeGenParamNames, 15
- sracipeGetTS, 15
- sracipeGetTS, RacipeSE-method (sracipeGetTS), 15
- sracipeHeatmapSimilarity, 8, 16, 17
- sracipeIC, 10, 17
- sracipeIC, RacipeSE-method (sracipeIC), 17
- sracipeIC-set (sracipeIC<-), 18
- sracipeIC<-, 18
- sracipeIC<-, RacipeSE-method (sracipeIC<-), 18
- sracipeKnockDown, 8, 12, 13, 17, 19, 20, 22, 24, 25, 30
- sracipeKnockDown, RacipeSE-method (sracipeKnockDown), 19
- sracipeNormalize, 20
- sracipeNormalize, RacipeSE-method (sracipeNormalize), 20
- sracipeOverExp, 8, 12, 13, 17, 20, 21, 22, 24, 25, 30
- sracipeOverExp, RacipeSE-method (sracipeOverExp), 21
- sracipeParams, 10, 22
- sracipeParams, RacipeSE-method (sracipeParams), 22
- sracipeParams-set (sracipeParams<-), 23
- sracipeParams<-, 23
- sracipeParams<-, RacipeSE-method (sracipeParams<-), 23
- sracipePlotCircuit, 23
- sracipePlotCircuit, RacipeSE-method (sracipePlotCircuit), 23
- sracipePlotData, 8, 12, 13, 17, 20, 22, 24, 24, 25, 30
- sracipePlotData, RacipeSE-method (sracipePlotData), 24
- sracipePlotParamBifur, 26
- sracipePlotParamBifur, RacipeSE-method

(`sracipePlotParamBifur`), [26](#)  
`sracipeSimulate`, [8](#), [10](#), [12](#), [13](#), [17](#), [19–22](#),  
[24–26](#), [27](#), [30](#)  
`SummarizedExperiment`, [10](#), [11](#)