Package 'sRACIPE'

October 16, 2019

Type Package

Title Systems biology tool to simulate gene regulatory circuits

Version 1.0.0

Author Vivek Kohar

Maintainer Vivek Kohar <vivek.kohar@gmail.com>

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

VignetteBuilder knitr

LinkingTo Rcpp

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

Depends R (>= 3.6),SummarizedExperiment, methods, Rcpp

Imports ggplot2, reshape2, MASS, RColorBrewer, gridExtra,visNetwork, gplots, umap, htmlwidgets, S4Vectors, BiocGenerics, grDevices, stats, utils

RoxygenNote 6.1.1

URL https://github.com/vivekkohar/sRACIPE

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

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

git_branch RELEASE_3_9

git_last_commit 318a47d

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`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 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.

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. 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

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 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 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(...)
```

Arguments

...	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.

`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`

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" 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 *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)
```

`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)
```

---

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)
```

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 parameteres

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,
  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, 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.
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, 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, ...)
```

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. 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 ignoired.

`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 precison of the output.

`printStart` (optional) numeric (0-`simulationTime`). Default `simulationTime`. Its use should be avoided. 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.

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.
...	Other arguments

Value

List. A list containing the circuit, parameters, initial conditions, simulated gene expressions, and simulation configuration.

Related Functions

sracipeSimulate, sracipeKnockDown, sracipeOverExp, sracipePlotData

Examples

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

Index