Package ‘CNORode’

May 29, 2024

Type Package
Title ODE add-on to CellNOptR
Version 1.46.0
Date 2022-03-22
Author David Henriques, Thomas Cokelaer, Attila Gabor, Federica Eduati, Enio Gjerga
Description Logic based ordinary differential equation (ODE) add-on to CellNOptR.
License GPL-2
LazyLoad yes
Depends CellNOptR, genalg
Enhances doParallel, foreach
Suggests knitr, rmarkdown
VignetteBuilder knitr
biocViews ImmunoOncology, CellBasedAssays, CellBiology, Proteomics, Bioinformatics, TimeCourse
RoxygenNote 7.1.2

Contents

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cnodata</td>
<td>2</td>
</tr>
<tr>
<td>cnolist</td>
<td>2</td>
</tr>
<tr>
<td>cnolistCNORodeExample</td>
<td>3</td>
</tr>
</tbody>
</table>
Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNORode</td>
<td>3</td>
</tr>
<tr>
<td>createLbodeContPars</td>
<td>4</td>
</tr>
<tr>
<td>crossvalidateODE</td>
<td>5</td>
</tr>
<tr>
<td>defaultParametersGA</td>
<td>6</td>
</tr>
<tr>
<td>defaultParametersSSm</td>
<td>7</td>
</tr>
<tr>
<td>getLbodeContObjFunction</td>
<td>8</td>
</tr>
<tr>
<td>getLbodeDataSim</td>
<td>10</td>
</tr>
<tr>
<td>getLbodeMINLPObjFunction</td>
<td>12</td>
</tr>
<tr>
<td>getLbodeModelSim</td>
<td>13</td>
</tr>
<tr>
<td>getLbodeSimFunction</td>
<td>15</td>
</tr>
<tr>
<td>getStates</td>
<td>16</td>
</tr>
<tr>
<td>incidence2Adjacency</td>
<td>17</td>
</tr>
<tr>
<td>indices</td>
<td>17</td>
</tr>
<tr>
<td>minlpLbodeSSm</td>
<td>18</td>
</tr>
<tr>
<td>model</td>
<td>20</td>
</tr>
<tr>
<td>parEstimationLbode</td>
<td>20</td>
</tr>
<tr>
<td>parEstimationLbodeGA</td>
<td>22</td>
</tr>
<tr>
<td>parEstimationLbodeSSm</td>
<td>24</td>
</tr>
<tr>
<td>pknmodel</td>
<td>26</td>
</tr>
<tr>
<td>plotLbodeFitness</td>
<td>27</td>
</tr>
<tr>
<td>plotLbodeModelSim</td>
<td>29</td>
</tr>
<tr>
<td>runCNORode</td>
<td>30</td>
</tr>
<tr>
<td>simdata2cnolist</td>
<td>32</td>
</tr>
<tr>
<td>simulate</td>
<td>33</td>
</tr>
</tbody>
</table>

A cnodata from CellNoptR

**Description**

A cnodata from CellNoptR to use with provided examples

---

A cnolist from CellNoptR

**Description**

A cnolist from CellNoptR to use with provided examples
A cnolist from CellNoptR to use with provided CNORode examples.

This package is used for the simulation and fitting of logic based ODE models based on the Odefy approach.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>CNORode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.2.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2012-03-14</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-3</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

David Henriques, Thomas Cokelaer
Maintainer: David Henriques <dhenriques@ebi.ac.uk>

References


createLBodeContPars


See Also


createLBodeContPars  Create a list with ODE parameter information needed to perform parameter estimation

Description

Creates a list with the continuous parameters to simulate the model, upper and lower bounds for the parameter estimation, parameters names, indices of the parameters and other information.

Usage

createLBodeContPars(model, LB_n = 1, LB_k = 0.1, LB_tau = 0.01, 
UB_n = 5, UB_k = 0.9, UB_tau = 10, default_n = 3, default_k = 0.5, 
default_tau = 1, LB_in = c(), UB_in = c(), opt_n = TRUE, opt_k = TRUE, 
opt_tau = TRUE, random = FALSE)

Arguments

model The logic model to be simulated.
LB_n A numeric value to be used as lower bound for all parameters of type n.
LB_k A numeric value to be used as lower bound for all parameters of type k.
LB_tau A numeric value to be used as lower bound for all parameters of type tau.
UB_n A numeric value to be used as upper bound for all parameters of type n.
UB_k A numeric value to be used as upper bound for all parameters of type k.
UB_tau A numeric value to be used as upper bound for all parameters of type tau.
default_n The default parameter to be used for every parameter of type n.
default_k The default parameter to be used for every parameter of type k.
default_tau The default parameter to be used for every parameter of type tau.
LB_in An array with the the same length as ode_parameters$parValues with lower bounds for each specific parameter.
UB_in An array with the the same length as ode_parameters$parValues with upper bounds for each specific parameter.
opt_n Add all parameter n to the index of parameters to be fitted.
opt_k Add all parameter k to the index of parameters to be fitted.
opt_tau Add all parameter tau to the index of parameters to be fitted.
random logical value that determines that a random solution is for the parameters to be optimized.

Value

parNames An array containing the names of the parameters.
parValues An array containing the values of the parameters, in the same order as the names.
index_opt_pars An array containing the indexes for the parameters to be fitted.
index_n An array containing the indexes of the parameters of type n.
index_k An array containing the indexes of the parameters of type k.
index_tau An array containing the indexes of the parameters of type tau.
LB An array containing the lower bound for each parameter.
UB An array containing the upper bound for each parameter.

Author(s)

David Henriques, Thomas Cokelaer

Examples

library(CNORode)
data("ToyCNOList",package="CNORode");
data("ToyModel",package="CNORode");
data("ToyIndices",package="CNORode");
ode_parameters=createLBodeContPars(model, opt_n=FALSE,default_n=2,
random=TRUE, LB_k=0.25, UB_k=0.8, LB_tau=0.01, UB_tau=10);

crossvalidateODE

Crossvalidate ODE model

Description

k-fold crossvalidation for logic ODE model

Usage

crossvalidateODE(
  CNOlist,
  model,
  nfolds = 10,
  foldid = NULL,
  type = "datapoint",
)
defaultParametersGA

```r
parallel = FALSE,
odet_parameters = NULL,
paramsSSm = NULL,
method = "essm"
)
```

**Arguments**

- **CNOlist**
  Cnolist which contains all the experiments

- **model**
  a model prepared for the training

- **nfolds**
  number of folds - default is 10. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets.

- **foldid**
  an optional vector of values between ‘1’ and ‘nfold’ identifying what fold each observation is in. If supplied, ‘nfold’ can be missing.

- **type**
  define the way to do the crossvalidation. The default is ‘type=“datapoint”’, which assigns the data randomly into folds. The option ‘type=“experiment”’ uses whole experiments for crossvalidation (all data corresponding to a cue combination). The ‘type=observable’ uses the subset of nodes across all experiments for crossvalidation.

- **parallel**
  use for parallel execution, requires the doParallel package

- **ode_parameters**
  list of fitted logic ODE parameter

- **paramsSSm**
  parameters for the SSm optimizer for running the optimization in crossvalidation

- **method**
  Selection of optimization method: only "ga" or "essm" arguments are accepted

**Details**

Does a k-fold cross-validation for logic ODE CellNOpt models. In k-iterations a fraction of the data is eliminated from the CNOlist. The model is trained on the remaining data and then the model predicts the held-out data. Then the prediction accuracy is reported for each iteration.

**See Also**

parEstimationLBode

---

`defaultParametersGA` *Create default options to perform parameter estimation with a genetic algorithm.*

**Description**

This function returns a list with several arguments for performing parameter estimation with the genetic algorithm from the package genalg.

**Usage**

`defaultParametersGA()`
**Value**

- **mutationChance**: NA
- **popSize**: 200
- **iters**: 100
- **elitism**: NA
- **time**: 1
- **monitor**: TRUE
- **verbose**: 0
- **transfer_function**: 3
- **reltol**: 1e-04
- **atol**: 0.001
- **maxStepSize**: Inf
- **maxNumSteps**: 1e+05
- **maxErrTestsFails**: 50
- **nan_fac = 1**: 0

**Author(s)**

David Henriques, Thomas Cokelaer

**See Also**

- CellNOptR
- parEstimationLBode
- parEstimationLBodeGA

---

**defaultParametersSSm**

Create default options to perform parameter estimation with scatter search meta-heuristic.

**Description**

This function returns a list with several arguments for performing parameter estimation with scatter search meta-heuristic algorithm from the package essR.

**Usage**

defaultParametersSSm()
**Value**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxeval</td>
<td>Inf</td>
</tr>
<tr>
<td>maxtime</td>
<td>100</td>
</tr>
<tr>
<td>ndiverse</td>
<td>NULL</td>
</tr>
<tr>
<td>dim_refset</td>
<td>NULL</td>
</tr>
<tr>
<td>local_solver</td>
<td>NULL</td>
</tr>
<tr>
<td>verbose</td>
<td>0</td>
</tr>
<tr>
<td>transfer_function</td>
<td>3</td>
</tr>
<tr>
<td>reltol</td>
<td>1e-04</td>
</tr>
<tr>
<td>atol</td>
<td>0.001</td>
</tr>
<tr>
<td>maxStepSize</td>
<td>Inf</td>
</tr>
<tr>
<td>maxNumSteps</td>
<td>1e+05</td>
</tr>
<tr>
<td>maxErrTestsFails</td>
<td>50</td>
</tr>
<tr>
<td>nan_fac</td>
<td>1</td>
</tr>
<tr>
<td>lambda_tau</td>
<td>0</td>
</tr>
<tr>
<td>lambda_k</td>
<td>0</td>
</tr>
<tr>
<td>bootstrap</td>
<td>0</td>
</tr>
<tr>
<td>SSpenalty_fac</td>
<td>0</td>
</tr>
<tr>
<td>SScontrolPenalty_fac</td>
<td>0</td>
</tr>
<tr>
<td>boot_seed</td>
<td>sample(1:10000,1)</td>
</tr>
</tbody>
</table>

**Author(s)**

David Henriques, Thomas Cokelaer, Federica Eduati

**See Also**

CellNOptR parEstimationLBode parEstimationLBodeSSm

---

**getLBodeContObjFunction**

*Returns the objective function to perform parameter estimation.*

**Description**

This function configures returns the objective function that can be used to evaluate the fitness of a logic based ODE model using a particular set of parameters. This function can be particularly useful if you are planning to couple a nonlinear optimization solver. The returned value of the objective function corresponds to the mean squared value normalized by the number of data points.
**Usage**

```r
getLBodeContObjFunction(cnolist, model, ode_parameters, indices=NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, nan_fac = 1, lambda_tau=0, lambda_k=0, bootstrap=F, SSpenalty_fac=0, SScontrolPenalty_fac=0, boot_seed=sample(1:10000,1))
```

**Arguments**

- **cnolist**: A list containing the experimental design and data.
- **model**: The logic model to be simulated.
- **ode_parameters**: A list with the ODEs parameter information. Obtained with `createLBodeContPars`.
- **indices**: Indices to map data in the model. Obtained with `indexFinder` function from `CellNOptR`.
- **time**: An integer with the index of the time point to start the simulation. Default is 1.
- **verbose**: A logical value that triggers a set of comments.
- **transfer_function**: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
- **reltol**: Relative Tolerance for numerical integration.
- **atol**: Absolute tolerance for numerical integration.
- **maxStepSize**: The maximum step size allowed to ODE solver.
- **maxNumSteps**: The maximum number of internal steps between two points being sampled before the solver fails.
- **maxErrTestsFails**: Specifies the maximum number of error test failures permitted in attempting one step.
- **nan_fac**: A penalty for each data point the model is not able to simulate. We recommend higher than 0 and smaller that 1.
- **lambda_tau**: Tunable regularisation parameters to penalise L1-norm of parameters tau and induce sparsity. We recommend testing values between 0 and 100 (in log scale) to find best compromise between good fit and sparse model. Default =0, corresponding to no regularisation.
- **lambda_k**: Tunable regularisation parameters to penalise L1-norm of parameters k and induce sparsity. We recommend testing values between 0 and 100 (in log scale) to find best compromise between good fit and sparse model. Default =0, corresponding to no regularisation.
- **bootstrap**: If set to TRUE performs random sampling with replacement of the measurements used in the optimisation (to be run multiple times to get bootstrapped distribution of parameters). Default =FALSE, no bootstrapping.
- **SSpenalty_fac**: Penalty factor for penalising solutions which do not reach steady state. Default =0.
- **SScontrolPenalty_fac**: Penalty factor for penalising solutions for which the control (unperturbed) condition (assumed to be first row) does not reach steady state. Default =0.
- **boot_seed**: Seed used for random sampling if bootstrap=TRUE. Default chose random seed between 0 and 10000
getLBodeDataSim

**Details**

Check **CellNOptR** for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

**Value**

Returns a function to evaluate the model fitness. This function receives a vector containing both continuous parameters and integer values representing which reactions should be kept in the model.

**Author(s)**

David Henriques, Thomas Cokelaer, Federica Eduati

**See Also**

**CellNOptR** createLBodeContPars

**Examples**

```r
library(CNORode)
data("ToyCNOlist", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");

ode_parameters=createLBodeContPars(model, random=TRUE);
minlp_obj_function=getLBodeContObjFunction(cnolistCNORodeExample, model, ode_parameters, indices);

x=ode_parameters$parValues;
f=minlp_obj_function(x);
```

---

**getLBodeDataSim**  
*Simulate value signals a CNO list With Logic-Based ODEs.*

**Description**

This function receives a set of inputs, namely the cnolist and the model and returns a list with the same size of the cnolist$valueSignals.

**Usage**

```r
getLBodeDataSim(cnolist, model, ode_parameters = NULL, indices = NULL, 
    timeSignals=NULL, time = 1, verbose = 0, transfer_function = 3, 
    reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, 
    maxErrTestsFails = 50)
```
**getLBodeDataSim**

**Arguments**

- **cnolist**  
  A list containing the experimental design and data.

- **model**  
  A list with the ODEs parameter information. Obtained with `createLBodeContPars`.

- **ode_parameters**  
  A list with the ODEs parameter information. Obtained with `makeParameterList` function.

- **indices**  
  Indices to map data in the model. Obtained with `indexFinder` function from `CellNOptR`.

- **timeSignals**  
  An array containing a different timeSignals. If you use this argument, it will also modify the dimensions from `valueSignals`.

- **time**  
  An integer with the index of the time point to start the simulation. Default is 1.

- **verbose**  
  A logical value that triggers a set of comments.

- **transfer_function**  
  The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.

- **reltol**  
  Relative Tolerance for numerical integration.

- **atol**  
  Absolute tolerance for numerical integration.

- **maxStepSize**  
  The maximum step size allowed to ODE solver.

- **maxNumSteps**  
  The maximum number of internal steps between two points being sampled before the solver fails.

- **maxErrTestsFails**  
  Specifies the maximum number of error test failures permitted in attempting one step.

**Details**

Check `CellNOptR` for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

**Value**

Returns a list with simulated data that has the same structure as the cnolist$valueSignals. One matrix for each time-point.

**Author(s)**

David Henriques, Thomas Cokelaer

**See Also**

`CellNOptR` `parEstimationLBode` `parEstimationLBodeSSm`
**getLBodeMINLPObjFunction**

Get the objective function to evaluate the fitness of a given model structure and set of parameters.

**Description**

This function configures returns the objective function that can be used to evaluate the fitness of a logic based ODE model using a particular set of parameters and model structure. This function can be particular useful if you are planing to couple a mixed integer nonlinear programming optimization solver. The returned value of the objective function corresponds to the mean squared value.

**Usage**

```
getLBodeMINLPObjFunction(cnolist, model, ode_parameters, indices=NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, nan_fac = 1)
```

**Arguments**

- `cnolist`: A list containing the experimental design and data.
- `model`: The logic model to be simulated.
- `ode_parameters`: A list with the ODEs parameter information. Obtained with `createLBodeContPars`.
- `indices`: Indices to map data in the model. Obtained with indexFinder function from CellNOptR.
- `time`: An integer with the index of the time point to start the simulation. Default is 1.
- `verbose`: A logical value that triggers a set of comments.
- `transfer_function`: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
- `reltol`: Relative Tolerance for numerical integration.
- `atol`: Absolute tolerance for numerical integration.
- `maxStepSize`: The maximum step size allowed to ODE solver.
- `maxNumSteps`: The maximum number of internal steps between two points being sampled before the solver fails.
getLBodeModelSim

maxErrTestsFails
Specifies the maximum number of error test failures permitted in attempting one step.

nan_fac
A penalty for each data point the model is not able to simulate. We recommend higher than 0 and smaller than 1.

Details
Check CellNOptR for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

Value
Returns a function to evaluate the model fitness. This function receives a continuous parameter vector.

Author(s)
David Henriques, Thomas Cokelaer

See Also
CellNOptR createLBodeContPars

Examples
library(CNORode)
data("ToyCNOlist", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");

ode_parameters=createLBodeContPars(model, random=TRUE);
minlp_obj_function=getLBodeMINLPObjFunction(cnolistCNORodeExample, model, ode_parameters, indices);

n_int_vars=dim(model$interMat)[2];
x_int=round(runif(n_int_vars))
x_cont=ode_parameters$parValues;
x=c(x_cont,x_int);
f=minlp_obj_function(x);

getLBodeModelSim Simulate the logic-based ODE model

Description
This function simulates a logic-based ODE model and return a list with one matrix for each time point. The input species in the model are filled with NA values. If the simulation of a particular set of initial conditions fails the solver will fill the experience row with NA values.
**getLBodeModelSim**

**Usage**

```
getLBodeModelSim(cnolist, model, ode_parameters = NULL, indices = NULL, timeSignals=NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50)
```

**Arguments**

- **cnolist**
  A list containing the experimental design and data.

- **model**
  The logic model to be simulated.

- **ode_parameters**
  A list with the ODEs parameter information. Obtained with `createLBodeContPars`.

- **indices**
  Indices to map data in the model. Obtained with `indexFinder` function from `CellNOptR`.

- **timeSignals**
  An array containing a different timeSignals. If you use this argument, it will also modify the dimensions from valueSignals.

- **time**
  An integer with the index of the time point to start the simulation. Default is 1.

- **verbose**
  A logical value that triggers a set of comments.

- **transfer_function**
  The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.

- **reltol**
  Relative Tolerance for numerical integration.

- **atol**
  Absolute tolerance for numerical integration.

- **maxStepSize**
  The maximum number of internal steps between two points being sampled before the solver fails.

- **maxNumSteps**
  The maximum number of internal steps between two points being sampled before the solver fails.

- **maxErrTestsFails**
  Specifies the maximum number of error test failures permitted in attempting one step.

**Details**

Check `CellNOptR` for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

**Value**

Returns a list with simulated data with similar structure to cnolist$valueSignals. Contains one matrix for each time-point. Each matrix contains one row per experiment and one columns per model species.

**Author(s)**

David Henriques, Thomas Cokelaer

**See Also**

- `CellNOptR`
- `createLBodeContPars`
getLBodeSimFunction

Examples

library(CNORode)
data('ToyCNOlist', package='CNORode');
data('ToyModel', package='CNORode');
data('ToyIndices', package='CNORode');
modelSimulation = getLBodeModelSim(cnolistCNORodeExample, model, indices=indices);

getLBodeSimFunction Get a function to simulate a logic based ODE model.

Description

This function is internally used by CNORode to configure the simulation function with default arguments.

Usage

getLBodeSimFunction(cnolist1, model1, adjMatrix1, indices1, odeParameters1, time1 = 1, verbose1 = 0, transfer_function1 = 3, reltol1 = 1e-04, atol1 = 0.001, maxStepSize1 = Inf, maxNumSteps1 = 1e+05, maxErrTestsFails1 = 50)

Arguments

cnolist1 A list containing the experimental design and data.
model1 The logic model to be simulated.
adjMatrix1 An adjacency matrix from the model.
indices1 Indices to map data in the model. Obtained with indexFinder function from CellNOptR.
odeParameters1 A list with the ODEs parameter information. Obtained with createLBodeContPars.
time1 An integer with the index of the time point to start the simulation. Default is 1.
verbose1 A logical value that triggers a set of comments.
transfer_function1 The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
reltol1 Relative Tolerance for numerical integration.
atol1 Absolute tolerance for numerical integration.
maxStepSize1 The maximum step size allowed to ODE solver.
maxNumSteps1 The maximum number of internal steps between two points being sampled before the solver fails.
maxErrTestsFails1 Specifies the maximum number of error test failures permitted in attempting one step.
getStates

Value
A function that returns a simulated model.

Note
This function is for CNORode internal use.

Author(s)
David Henriques, Thomas Cokelaer

See Also
CellNOptR CNORode

getStates  

Find which species in the model are states.

Description
Receives an adjacency matrix (model$interMat from CellNOptR) and finds which species are states (i.e. not inputs).

Usage
getStates(adjacency)

Arguments
adjacency  
An adjacency matrix from the model.

Value
A numeric vector with 0’s for positions which are states and 1’s for positions which are.

Note
For internal use of CNORode.

Author(s)
David Henriques, Thomas Cokelaer

See Also
incidence2Adjacency
incidence2Adjacency

Convert an incidence matrix into an adjacency matrix.

Description

Convert the incidence matrix (model representation of CellNoptR) into an adjacency matrix. Denotes the inputs/output relationships.

Usage

incidence2Adjacency(model)

Arguments

model Model from CellNoptR.

Value

Directed Adjacency matrix of size n_species by n_species.

Note

For internal use of CNORode.

Author(s)

David Henriques, Thomas Cokelaer

See Also

CellNOptR

indices

Indices that relate cnolist to model

Description

A list with indices that relate the cnolist with the model from CellNOptR
minlpLBodeSSm

Search for the best combination of continuous parameters and logic gates.

Description

This function uses essR to search for the best set of continuous parameters and model structure. The objective function is the same as the one provided by `getLBodeMINLPObjFunction`.

Usage

```r
minlpLBodeSSm(cnolist, model, ode_parameters = NULL, int_x0=NULL, indices = NULL, maxeval = Inf, maxtime = 100, ndiverse = NULL, dim_refset = NULL, local_solver = NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, nan_fac = 1)
```

Arguments

- `cnolist`: A list containing the experimental design and data.
- `model`: The logic model to be simulated.
- `ode_parameters`: A list with the ODEs parameter information. Obtained with `createLBodeContPars`.
- `int_x0`: Vector with initial solution for integer parameters.
- `indices`: Indices to map data in the model. Obtained with `indexFinder` function from `CellNOptR`.
- `maxeval`: Maximum number of evaluation in the optimization procedure.
- `maxtime`: Maximum number of evaluation spent in optimization procedure.
- `ndiverse`: Duration of the optimization procedure.
- `dim_refset`: Number of diverse initial solutions.
- `local_solver`: Local solver to be used in SSm.
- `time`: An integer with the index of the time point to start the simulation. Default is 1.
- `verbose`: A logical value that triggers a set of comments.
- `transfer_function`: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and for normalized Hill function.
- `reltol`: Relative Tolerance for numerical integration.
- `atol`: Absolute tolerance for numerical integration.
- `maxStepSize`: The maximum step size allowed to ODE solver.
- `maxNumSteps`: The maximum number of internal steps between two points being sampled before the solver fails.
- `maxErrTestsFails`: Specifies the maximum number of error test failures permitted in attempting one step.
- `nan_fac`: A penalty for each data point the model is not able to simulate. We recommend higher than 0 and smaller that 1.
Details

Check CellNOptR for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

Value

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB_n</td>
<td>A numeric value to be used as lower bound for all parameters of type n.</td>
</tr>
<tr>
<td>LB_k</td>
<td>A numeric value to be used as lower bound for all parameters of type k.</td>
</tr>
<tr>
<td>LB_tau</td>
<td>A numeric value to be used as lower bound for all parameters of type tau.</td>
</tr>
<tr>
<td>UB_n</td>
<td>A numeric value to be used as upper bound for all parameters of type n.</td>
</tr>
<tr>
<td>UB_k</td>
<td>A numeric value to be used as upper bound for all parameters of type k.</td>
</tr>
<tr>
<td>UB_tau</td>
<td>A numeric value to be used as upper bound for all parameters of type tau.</td>
</tr>
<tr>
<td>default_n</td>
<td>The default parameter to be used for every parameter of type n.</td>
</tr>
<tr>
<td>default_k</td>
<td>The default parameter to be used for every parameter of type k.</td>
</tr>
<tr>
<td>default_tau</td>
<td>The default parameter to be used for every parameter of type tau.</td>
</tr>
<tr>
<td>LB_in</td>
<td>An array with the the same length as ode_parameters$parValues with lower bounds for each specific parameter.</td>
</tr>
<tr>
<td>UB_in</td>
<td>An array with the the same length as ode_parameters$parValues with upper bounds for each specific parameter.</td>
</tr>
<tr>
<td>opt_n</td>
<td>Add all parameter n to the index of parameters to be fitted.</td>
</tr>
<tr>
<td>opt_k</td>
<td>Add all parameter k to the index of parameters to be fitted.</td>
</tr>
<tr>
<td>opt_tau</td>
<td>Add all parameter tau to the index of parameters to be fitted.</td>
</tr>
<tr>
<td>random</td>
<td>A logical value that determines that a random solution is for the parameters to be optimised.</td>
</tr>
<tr>
<td>model</td>
<td>The best fitting found model structure.</td>
</tr>
<tr>
<td>smm_results</td>
<td>A list containing the information provided by the nonlinear optimization solver.</td>
</tr>
</tbody>
</table>

Author(s)

David Henriques, Thomas Cokelaer

See Also

CellNOptR createLBodeContPars essR

Examples

```r
## Not run:
data("ToyCNOlist", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");

do_parameters=createLBodeContPars(model, random=TRUE);

#Visualize initial solution
```
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model,ode_parameters,indices=indices)
ode_parameters=minlplBodeSSm(cnolistCNORodeExample, model,ode_parameters);

model=ode_parameters$model;

#Visualize fitted solution
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model,indices=indices);

## End(Not run)

---

model

A model from CellNoptR

---

### Description
A model from CellNoptR to use with provided examples

---

### parEstimationLBode

**Perform parameter estimation using a genetic algorithm (package genalg) or ssm (if package essm available).**

---

### Description
This function is an alias to the parEstimationLBode variants (**parEstimationLBodeGA** and **parEstimationLBodeSSm**)

### Usage

```r
parEstimationLBode(cnolist, model, method="ga", ode_parameters = NULL, indices = NULL, paramsGA=NULL, paramsSSm=NULL)
```

### Arguments

- **cnolist** A list containing the experimental design and data.
- **model** The logic model to be simulated.
- **method** Only "ga" or "essm" arguments are accepted.
- **ode_parameters** A list with the ODEs parameter information. Obtained with `createLBodeContPars`.
- **indices** Indices to map data in the model. Obtained with `indexFinder` function from CellNOptR.
- **paramsGA** A list of GA parameters. default is the list returned by `defaultParametersGA`.
- **paramsSSm** A list of SSm parameters. default is the list returned by `defaultParametersSSm`.
**Value**

- **LB_n** A numeric value to be used as lower bound for all parameters of type n.
- **LB_k** A numeric value to be used as lower bound for all parameters of type k.
- **LB_tau** A numeric value to be used as lower bound for all parameters of type tau.
- **UB_n** A numeric value to be used as upper bound for all parameters of type n.
- **UB_k** A numeric value to be used as upper bound for all parameters of type k.
- **UB_tau** A numeric value to be used as upper bound for all parameters of type tau.
- **default_n** The default parameter to be used for every parameter of type n.
- **default_k** The default parameter to be used for every parameter of type k.
- **default_tau** The default parameter to be used for every parameter of type tau.
- **LB_in** An array with the the same length as ode_parameters$parValues with lower bounds for each specific parameter.
- **UB_in** An array with the the same length as ode_parameters$parValues with upper bounds for each specific parameter.
- **opt_n** Add all parameter n to the index of parameters to be fitted.
- **opt_k** Add all parameter k to the index of parameters to be fitted.
- **opt_tau** Add all parameter tau to the index of parameters to be fitted.
- **random** A logical value that determines that a random solution is for the parameters to be optimized.
- **res** A list containing the information provided by the solver.

**Author(s)**

David Henriques, Thomas Cokelaer

**See Also**

`CellNOptR` `createLBodeContPars` `rbga`

**Examples**

```r
data("ToyCNOlist", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");

ode_parameters=createLBodeContPars(model, random=TRUE);
#Visualize initial solution
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model, ode_parameters, indices=indices)
paramsGA = defaultParametersGA()
paramsGA$maxStepSize = 1
paramsGA$popSize = 10
paramsGA$iter = 10
paramsGA$transfer_function = 2

ode_parameters=parEstimationLBode(cnolistCNORodeExample, model, ode_parameters=ode_parameters,
```

The documentation appears to be part of a software library, possibly related to parameter estimation and optimization in a bioinformatics context. The function `parEstimationLBode` provides a lower bound estimation for parameters in a system of ordinary differential equations (ODEs), suggesting it is used for modeling and simulation in biological or chemical systems. The parameters and functions defined here are essential for configuring the optimization process, including specifying bounds for parameters, default values, and options for optimization methods.
paramsGA=paramsGA)
#Visualize fitted solution
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model, ode_parameters, indices=indices)

parEstimationLBodeGA  

Perform parameter estimation using a genetic algorithm (package genalg).

Description
This function uses a genetic algorithm (package genalg) to perform parameter estimation. The objective function is the same as the one provided by getLBodeContObjFunction.

Usage
parEstimationLBodeGA(cnolist, model, ode_parameters = NULL, indices = NULL, mutationChance = NA, popSize = 200, iters = 100, elitism = NA, time = 1, monitor = TRUE, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, nan_fac = 1)

Arguments

- **cnolist**: A list containing the experimental design and data.
- **model**: The logic model to be simulated.
- **ode_parameters**: A list with the ODEs parameter information. Obtained with createLBodeContPars.
- **indices**: Indices to map data in the model. Obtained with indexFinder function from CellNOptR.
- **mutationChance**: The chance that a gene in the chromosome mutates. By default 1/(size+1). It affects the convergence rate and the probing of search space: a low chance results in quicker convergence, while a high chance increases the span of the search space.
- **popSize**: The population size.
- **iters**: The number of iterations.
- **elitism**: The number of chromosomes that are kept into the next generation. By default is about 20% of the population size.
- **time**: An integer with the index of the time point to start the simulation. Default is 1.
- **monitor**: If TRUE a plot will be generated to monitor the objective function.
- **verbose**: A logical value that triggers a set of comments.
- **transfer_function**: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
- **reltol**: Relative Tolerance for numerical integration.
- **atol**: Absolute tolerance for numerical integration.
- **maxStepSize**: The maximum step size allowed to ODE solver.
The maximum number of internal steps between two points being sampled before the solver fails.

Specifies the maximum number of error test failures permitted in attempting one step.

A penalty for each data point the model is not able to simulate. We recommend higher than 0 and smaller that 1.

Value

A numeric value to be used as lower bound for all parameters of type n.

A numeric value to be used as lower bound for all parameters of type k.

A numeric value to be used as lower bound for all parameters of type tau.

A numeric value to be used as upper bound for all parameters of type n.

A numeric value to be used as upper bound for all parameters of type k.

A numeric value to be used as upper bound for all parameters of type tau.

The default parameter to be used for every parameter of type n.

The default parameter to be used for every parameter of type k.

The default parameter to be used for every parameter of type tau.

An array with the the same length as ode_parameters$parValues with lower bounds for each specific parameter.

An array with the the same length as ode_parameters$parValues with upper bounds for each specific parameter.

Add all parameter n to the index of parameters to be fitted.

Add all parameter k to the index of parameters to be fitted.

Add all parameter tau to the index of parameters to be fitted.

A logical value that determines that a random solution is for the parameters to be optimized.

A list containing the information provided by the nonlinear optimization solver (genalg).

Author(s)

David Henriques, Thomas Cokelaer

See Also

CellNOptR createLBodeContPars rbga
parEstimationLBodeSSm

Perform parameter estimation using essR.

Description

This function uses essR to perform parameter estimation. The objective function is the same as the one provided by getLBodeContObjFunction.

Usage

parEstimationLBodeSSm(cnolist, model, ode_parameters = NULL, indices = NULL, maxeval = Inf, maxtime = 100, ndiverse = NULL, dim_refset = NULL, local_solver = NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, nan_fac = 1, lambda_tau = 0, lambda_k = 0, bootstrap = FALSE, SSpenalty_fac = 0, SScontrolPenalty_fac = 0, boot_seed = sample(1:10000,1))

Arguments

cnolist
A list containing the experimental design and data.

model
The logic model to be simulated.

ode_parameters
A list with the ODEs parameter information. Obtained with createLBodeContPars.

indices
Indices to map data in the model. Obtained with indexFinder function from CellNOptR.

maxeval
Maximum number of evaluation in the optimization procedure.

maxtime
Duration of the optimization procedure.

ndiverse
Number of diverse initial solutions.

dim_refset
Size of the reference set.

local_solver
Local solver to be used in SSm.

time
An integer with the index of the time point to start the simulation. Default is 1.
verbose  A logical value that triggers a set of comments.

transfer_function  The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.

reltol  Relative Tolerance for numerical integration.
atol  Absolute tolerance for numerical integration.

maxStepSize  The maximum step size allowed to ODE solver.

maxNumSteps  The maximum number of internal steps between two points being sampled before the solver fails.

maxErrTestsFails  Specifies the maximum number of error test failures permitted in attempting one step.

nan_fac  A penalty for each data point the model is not able to simulate. We recommend higher than 0 and smaller that 1.

lambda_tau  penalty parameter for node parameters (tau)

lambda_k  penalty parameter for edge parameters (k)

bootstrap  Boolean, default: FALSE. If the residuals should be bootstrapped.

SSpenalty_fac  Steady-state penalty: at the end of the simulation the model states should reach steady state. The steady state is measured by the sum of squares of the state derivatives.

SScontrolPenalty_fac  Steady-state penalty for a control experiment, the default is 0. The first condition should represent a control condition (no stimulus or inhibition). Then the model simulation is penalised if it deviates from the initial conditions. This is to make sure that the predicted dynamics is not due to the initial conditions, but because of the stimuli.

boot_seed  random seed used for the bootstrapping.

Details

Check CellNOptR for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

Value

LB_n  A numeric value to be used as lower bound for all parameters of type n.

LB_k  A numeric value to be used as lower bound for all parameters of type k.

LB_tau  A numeric value to be used as lower bound for all parameters of type tau.

UB_n  A numeric value to be used as upper bound for all parameters of type n.

UB_k  A numeric value to be used as upper bound for all parameters of type k.

UB_tau  A numeric value to be used as upper bound for all parameters of type tau.

default_n  The default parameter to be used for every parameter of type n.

default_k  The default parameter to be used for every parameter of type k.
default_tau  The default parameter to be used for every parameter of type tau.
LB_in        An array with the the same length as ode_parameters$parValues with lower bounds for each specific parameter.
UB_in        An array with the the same length as ode_parameters$parValues with upper bounds for each specific parameter.
opt_n        Add all parameter n to the index of parameters to be fitted.
opt_k        Add all parameter k to the index of parameters to be fitted.
opt_tau      Add all parameter tau to the index of parameters to be fitted.
random       A logical value that determines that a random solution is for the parameters to be optimized.
smm_results  A list containing the information provided by the nonlinear optimization solver.

Author(s)  David Henriques, Thomas Cokelaer

See Also  CellNOptR createLBodeContPars

Examples

## Not run:
data("ToyCN0list", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");

ode_parameters=createLBodeContPars(model, random=TRUE);

#Visualize intial simulation
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model, ode_parameters, indices=indices)

ode_parameters=parEstimationLBodeSSm(cnolistCNORodeExample, model, ode_parameters, indices=indices, maxtime=20, ndiverse=50, dim_refset=6);

#Visualize fitterd solution
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model, indices=indices, ode_parameters=ode_parameters);

## End(Not run)

pknmodel

A pknmodel from CellNOptR

Description

A pknmodel from CellNOptR to use with provided examples
Description

Plots the simulated values with the logic-based ODE against the data contained in the cnolist. The data values are represented with a black line and the simulated values with a blue line. Additionally, this function returns the simulated values.

Usage

plotLBodeFitness(cnolist, model, ode_parameters = NULL, indices = NULL, adjMatrix = NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, plot_index_signals = NULL, plot_index_experiments = NULL, plot_index_cues = NULL, colormap="heat", plotParams=list(margin=0.1, width=15, height=12, cmap_scale=1, cex=1.6, ymin=NULL))

Arguments

cnolist: A list containing the experimental design and data.
model: The logic model to be simulated.
ode_parameters: A list with the ODEs parameter information. Obtained with createLBodeContPars.
indices: Indices to map data in the model. Obtained with indexFinder function from CellNOptR.
adjMatrix: Model representation in the form of an adjacency matrix. When not provided will be automatically computed based on the model.
time: An integer with the index of the time point to start the simulation. Default is 1.
verbose: A logical value that triggers a set of comments.
transfer_function: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
reltol: Relative Tolerance for numerical integration.
atol: Absolute tolerance for numerical integration.
maxStepSize: The maximum step size allowed to ODE solver.
maxNumSteps: The maximum number of internal steps between two points being sampled before the solver fails.
maxErrTestsFails: Specifies the maximum number of error test failures permitted in attempting one step.
plot_index_signals
In case you only want to plot some signals, provide an integer vector with the indexes.

plot_index_experiments
In case you only want to plot some experiments, provide an integer vector with the indexes.

plot_index_cues
In case you only want to plot some cues, provide an integer vector with the indexes.

colormap
Uses the same colormap as in CellNOptR by default. If set to "green", it uses the deprecated colormap.

plotParams
additional parameters to refine the ploggin. See plotOptimResultsPan function in CellNOptR for more details.

Details
Check CellNOptR for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.

Value
Returns a list with simulated data that has the same structure as the cnolist$valueSignals. One matrix for each time-point.

Author(s)
David Henriques, Thomas Cokelaer

See Also
CellNOptR createLBodeContPars

Examples
library(CNORode)
data("ToyCN0list", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");
ode_parameters=createLBodeContPars(model, random=TRUE);
dataSimulation=plotLBodeFitness(cnolistCNORodeExample, model, indices=indices);
plotLBodeModelSim

Simulate the model and plot the obtained with the different experimental conditions.

Description

Plots the simulated values of the logic based ODE model. Only dynamic states are plotted, i.e. those that are not inputs. a blue line. Additionally this functions returns the the simulated values.

Usage

plotLBodeModelSim(cnolist, model, ode_parameters = NULL, indices = NULL, adjMatrix = NULL, timeSignals=NULL, time = 1, verbose = 0, transfer_function = 3, reltol = 1e-04, atol = 0.001, maxStepSize = Inf, maxNumSteps = 1e+05, maxErrTestsFails = 50, large = FALSE, nsplit = 4, show = TRUE)

Arguments

cnolist: A list containing the experimental design and data.
model: The logic model to be simulated.
ode_parameters: A list with the ODEs parameter information. Obtained with createLBodeContPars.
indices: Indices to map data in the model. Obtained with indexFinder function from CellNOptR.
adjMatrix: Model representation in the form of an adjacency matrix. When not provided will be automatically computed based in the model.
timeSignals: An array containing a different timeSignals. If you use this argument, it will also modify the dimensions from valueSignals.
time: An integer with the index of the time point to start the simulation. Default is 1.
verbose: A logical value that triggers a set of comments.
transfer_function: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
reltol: Relative Tolerance for numerical integration.
atol: Absolute tolerance for numerical integration.
maxStepSize: The maximum step size allowed to ODE solver.
maxNumSteps: The maximum number of internal steps between two points being sampled before the solver fails.
maxErrTestsFails: Specifies the maximum number of error test failures permitted in attempting one step.
large: Boolean variable defining if the plot should split into several subplots.
split: In case the large plot options is selected define how many subplots will exist. Default is 4.
show: Boolean variable defining if we should plot the CNOlist object.


Value

Returns a list with simulated Model values. One matrix of size number of species by number of experimental conditions for each time-point.

Author(s)

David Henriques, Thomas Cokelaer

See Also

CellNOptR createLBodeContPars

Examples

```r
library(CNORode)
data("ToyCNOlist",package="CNORode");
data("ToyModel",package="CNORode");
data("ToyIndices",package="CNORode");
modelSimulation=plotLBodeModelSim(cnolistCNORodeExample, model,indices=indices);
```

Description

A one-line wrapper of the CNORode pipeline

Usage

```r
runCNORode(
  model,
  data,
  compression = TRUE,
  results_folder = "CNORode_results",
  cutNONC = TRUE,
  expansion = FALSE,
  LB_n = 1,
  LB_k = 0.1,
  LB_tau = 0.01,
  UB_n = 5,
  UB_k = 0.9,
  UB_tau = 10,
  default_n = 3,
  default_k = 0.5,
  default_tau = 1,
  opt_n = TRUE,
  opt_k = TRUE,
  opt_tau = TRUE,
)```
runCNORode

random = TRUE,
maxeval = 1e+05,
maxtime = 60,
transfer_function = 3,
nan_fac = 1,
lambda_tau = 0,
lambda_k = 0
)

Arguments

model A filename of prior knowledge network (PKN) in the SIF format
data A measurement filename in the MIDAS format
compression compress the prior knowledge network (TRUE), see preprocessing
results_folder results folder for the analysis.
cutNONC cut non-observable non-controllable node from PKN (TRUE), see preprocessing
expansion expand OR gates in the PKN (FALSE), see preprocessing
LB_n lower bound on parameter n, see createLBodeContPars
LB_k lower bound on parameter k, see createLBodeContPars
LB_taux lower bound on parameter tau, see createLBodeContPars
UB_n upper bound on parameter n, see createLBodeContPars
UB_k upper bound on parameter k, see createLBodeContPars
UB_taux upper bound on parameter tau, see createLBodeContPars
default_n default value of parameter n, see createLBodeContPars
default_k default value of parameter k, see createLBodeContPars
default_taux default value of parameter tau, see createLBodeContPars
opt_n should parameter n be optimised, see createLBodeContPars
opt_k should parameter k be optimised, see createLBodeContPars
opt_taux should parameter tau be optimised, see createLBodeContPars
random initial parameter vector generation (TRUE: random, FALSE: half of the LB-UB)
maxeval maximum number of function evaluations in the optimisation, see parEstimationLBodeSSm
maxtime maximum CPU time (in seconds) spent on optimisation before calling final refinement, see parEstimationLBodeSSm
transfer_function transfer function types represented by the edges, see parEstimationLBodeSSm
nan_fac penalty for NA simulations, see parEstimationLBodeSSm
lambda_taux regularisation penalty for tau parameters, see parEstimationLBodeSSm
lambda_k regularisation penalty for k parameters for optimisation, see parEstimationLBodeSSm
Examples

```r
## Not run:
model = system.file("extdata", "ToyModelMBB_FeedbackAnd.sif",package="CNORode")
data = system.file("extdata", "ToyModelMBB_FeedbackAnd.csv", package="CNORode")
res = runCNORode(model,data,results_folder = ".\results")

## End(Not run)
```

---

**simdata2cnolist**

converts output of `getLBodeModelSim` to `cnolist`

---

**Description**

This function converts the simulated data returned by `getLBodeModelSim` into a valid CNOlist data structure.

**Usage**

```r
simdata2cnolist(sim_data, cnolist, model)
```

**Arguments**

- `sim_data`: structure returned by `getLBodeModelSim`
- `cnolist`: A list containing the experimental design and data.
- `model`: The logic model to be simulated.

**Value**

a CNOlist

**Author(s)**

Thomas Cokelaer

**See Also**

`CellNOptR` `createLBodeContPars`

**Examples**

```r
data('ToyCNOlist',package='CNORode');
data('ToyModel',package='CNORode');
data('ToyIndices',package='CNORode');
simdata = getLBodeModelSim(cnolistCNORodeExample, model,indices=indices)
cnolist = simdata2cnolist(simdata, cnolistCNORodeExample, model)

cnolist = simdata2cnolist(simdata, cnolistCNORodeExample, model)
```
**simulate**  

Simulate value signals a CNO list With Logic-Based ODEs.

**Description**

This function receives a set of inputs, namely the cnolist and the model and returns a list with the same size of the cnolist$valueSignals.

**Usage**

```
simulate(cnolist, model, ode_parameters=NULL, indices=NULL,  
adjMatrix=NULL, time=1, verbose=0, transfer_function=3,  
reltol=1e-04, atol=0.001, maxStepSize=Inf, maxNumSteps=1e+05,  
maxErrTestsFails=50)
```

**Arguments**

- `cnolist`: A list containing the experimental design and data.
- `model`: A list with the ODEs parameter information. Obtained with `createLBodeContPars`.
- `ode_parameters`: A list with the ODEs parameter information. Obtained with `makeParameterList` function.
- `indices`: Indices to map data in the model. Obtained with `indexFinder` function from `CellNOptR`.
- `adjMatrix`: The adjacency matrix. Recomputed if not provided.
- `time`: An integer with the index of the time point to start the simulation. Default is 1.
- `verbose`: A logical value that triggers a set of comments.
- `transfer_function`: The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.
- `reltol`: Relative Tolerance for numerical integration.
- `atol`: Absolute tolerance for numerical integration.
- `maxStepSize`: The maximum step size allowed to ODE solver.
- `maxNumSteps`: The maximum number of internal steps between two points being sampled before the solver fails.
- `maxErrTestsFails`: Specifies the maximum number of error test failures permitted in attempting one step.

**Details**

Check `CellNOptR` for details about the cnolist and the model format. For more details in the configuration of the ODE solver check the CVODES manual.
Value

Returns a list with simulated data that has the same structure as the cnolist$valueSignals. One matrix for each time-point.

Author(s)

David Henriques, Thomas Cokelaer

See Also

CellNOptR parEstimationLBode parEstimationLBodeSSm

Examples

library(CNORode)
data("ToyCNOlist",package="CNORode");
data("ToyModel",package="CNORode");
data("ToyIndices",package="CNORode");
dataSimulation = simulate(cnolistCNORodeExample, model,indices=indices);
Index

* CNORode
  CNORode, 3
* CVODES
  getLBodeSimFunction, 15
* CellNOptR
  parEstimationLBode, 20
  parEstimationLBodeGA, 22
* SSm
  defaultParametersSSm, 7
* adjacency
  getStates, 16
  incidence2Adjacency, 17
* algorithm
  defaultParametersGA, 6
  parEstimationLBode, 20
  parEstimationLBodeGA, 22
* default
  defaultParametersGA, 6
  defaultParametersSSm, 7
* essR
  defaultParametersSSm, 7
* genetic
  defaultParametersGA, 6
  parEstimationLBode, 20
  parEstimationLBodeGA, 22
* incidence
  incidence2Adjacency, 17
* logic
  parEstimationLBode, 20
  parEstimationLBodeGA, 22
* matrix
  incidence2Adjacency, 17
* model
  parEstimationLBode, 20
  parEstimationLBodeGA, 22
* parameters
  defaultParametersGA, 6
* states
  getStates, 16

CellNOptR, 4, 7, 8, 10, 11, 13, 14, 16, 17, 19, 21, 23, 25, 26, 28, 30, 32–34
cnodata, 2
cnolist, 2
cnolistCNORodeExample, 3
CNORode, 3, 16
createLBodeContPars, 4, 9–15, 18–24, 26–33
crossvalidateODE, 5
defaultParametersGA, 6
defaultParametersSSm, 7
getLBodeContObjFunction, 8, 22, 24
getLBodeDataSim, 10
getLBodeMINLPObjFunction, 12, 18
getLBodeModelSim, 4, 13
getLBodeSimFunction, 15
getStates, 16
incidence2Adjacency, 16, 17
indices, 17
min1pLBodeSSm, 18
model, 20
parEstimationLBode, 4, 6–8, 11, 20, 34
parEstimationLBodeGA, 7, 20, 22
parEstimationLBodeSSm, 8, 11, 20, 24, 31, 34
pnkmodel, 26
plotLBodeFitness, 4, 27
plotLBodeModelSim, 29
preprocessing, 31
rbga, 21, 23
runCNORode, 30
simdata2cnolist, 32
simulate, 33