Package ‘CNORode’

January 14, 2017

Type Package
Title ODE add-on to CellNOptR
Version 1.16.0
Date 2014-01-10
Author David Henriques, Thomas Cokelaer
Maintainer David Henriques <davidpcnet@hotmail.com>
biocViews CellBasedAssays, CellBiology, Proteomics, Bioinformatics, TimeCourse
Description ODE add-on to CellNOptR
License GPL-2
LazyLoad yes
Depends CellNOptR (>= 1.5.14), genalg
Enhances MEIGOR
NeedsCompilation yes

R topics documented:

   cnolistCNORodeExample                          2
   CNORode                                         2
   createLBodeContPars                             3
   defaultParametersGA                            4
   defaultParametersSSm                           5
   getLBodeContObjFunction                        6
   getLBodeDataSim                                 7
   getLBodeMINLPObjFunction                       9
   getLBodeModelSim                               10
   getLBodeSimFunction                            12
   getStates                                      13
   incidence2Adjacency                            13
   indices                                         14
   minlplBodeSSm                                  14
   model                                          16
   parEstimationLBode                              16
   parEstimationLBodeGA                           18
   parEstimationLBodeSSm                          20
   plotLBodeFitness                               22
index

**cnolistCNORodeExample**  
A cnolist from CellNoptR

**Description**

A cnolist from CellNoptR to use with provided CNORode examples.

**CNORode**  
*Logic based ODE extension for CellNoptR*

**Description**

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**

```r
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);
```

---

**Description**

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

**Usage**

```r
defaultParametersGA()
```

**Value**

- **mutationChance**: NA
- **popSize**: 200
- **iters**: 100
- **elitism**: NA
- **time**: 1
- **monitor**: TRUE
- **verbose**: 0
- **transfer_function**: 3

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

reitol  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
maxeval  Inf
maxtime  100
ndiverse  NULL
dim_refset  NULL
local_solver  NULL
verbose  0
transfer_function  3
reitol  1e-04
atol  0.001
maxStepSize  Inf
maxNumSteps  1e+05
maxErrTestsFails  50
nan_fac  1
Author(s)

David Henriques, Thomas Cokelaer

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

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, useVariances = F, initial_state=0.1)

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.
useVariances if True, use the variance in the fitness
initial_state initial state of the dynamic nodes (non-measured) (Defaults to 0.1)
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

**See Also**

**CellNOptR** createLBodeContPars

**Examples**

```r
library(CNORode)
data("ToyCNList",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, initial_state=0.1)
```
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.
initial_state: initial state of the dynamic nodes (non-measured) (Defaults to 0.1)

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 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

```r
library(CNORode)
data("ToyCNOList",package="CNORode");
data("ToyModel",package="CNORode");
data("ToyIndices",package="CNORode");
dataSimulation=getLBodeDataSim(cnolistCNORodeExample, model,indices=indices);
```
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.
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.
getLBodeModelSim

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.

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.
getLBodeModelSim

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

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, 
initial_state1=0.1)

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.
- initial_state1: initial state of the dynamic nodes (non-measured) (Defaults to 0.1)

Value
A function that returns a simulated model.

Note
This function is for CNORode internal use.

Author(s)
David Henriques, Thomas Cokelaer
**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**

```r
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**

```r
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

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.
duration of the optimisation 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

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 optimised.

model  The best fitting found model structure.

smm_results  A list containing the information provided by the nonlinear optimization solver.
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");

ode_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

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 same length as `ode_parameters$parValues` with lower bounds for each specific parameter.
- **UB_in**: An array with the same length as `ode_parameters$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, paramsGA=paramsGA)
# Visualize fitted solution
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model, ode_parameters, indices=indices)
```


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

```r
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, initial_state=0.1)
```

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 $1/(\text{size}+1)$. 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.
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.
initial_state initial state of the dynamic nodes (non-measured) (Defaults to 0.1)

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_tn 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_tn 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_tn 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_tn 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 nonlinear optimization solver (genalg).

Author(s)

David Henriques, Thomas Cokelaer
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

```r
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, useVariances=F, initial_state=0.1)
```

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`: An integer indicating the transfer function.
- `reltol`: Relative tolerance.
- `atol`: Absolute tolerance.
- `maxStepSize`: Maximum step size.
- `maxNumSteps`: Maximum number of steps.
- `maxErrTestsFails`: Maximum number of failed error tests.
- `nan_fac`: A factor for handling NaN values.
- `useVariances`: A logical value indicating whether to use variances.
- `initial_state`: Initial state for the simulation.
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.
useVariances Uses variance if any
initial_state initial state of the dynamic nodes (non-measured) (Defaults to 0.1)

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

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.

Author(s)

David Henriques, Thomas Cokelaer
plotLBodeFitness

See Also

CellNOptR createLBodeContPars

Examples

```r
## Not run:
data("ToyCNOlist", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");
ode_parameters=createLBodeContPars(model, random=TRUE);

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

## End(Not run)
```

plotLBodeFitness  

*Plot data against simulated values.*

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

```r
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), initial_state=0.1)
```

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.
- **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.
plotLBodeFitness

verbose
transfer_function
reltol
atol
maxStepSize
maxNumSteps
maxErrTestsFails
plot_index_signals
plot_index_experiments
plot_index_cues
colormap
plotParams
initial_state

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.

initial_state
initial state of the dynamic nodes (non-measured) (Defaults to 0.1)

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 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("ToyCNolist",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=T)

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.
nsplit In case the large plot options is selected define how many subplots will exist. Default is 4.
show show the error (defaults to TRUE)
**simdata2cnolist**

**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);
```

---

**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**

```r
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**

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

**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**

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

*Topic **CNORode**  
CNORode, 2

*Topic **CVODES**  
getLBodeSimFunction, 12

*Topic **CellNOptR**  
parEstimationLBode, 16  
parEstimationLBodeGA, 18

*Topic **SSm**  
defaultParametersSSm, 5

*Topic **adjacency**  
getStates, 13  
incidence2Adjacency, 13

*Topic **algorithm**  
defaultParametersGA, 4  
parEstimationLBode, 16  
parEstimationLBodeGA, 18

*Topic **default**  
defaultParametersGA, 4  
defaultParametersSSm, 5

*Topic **essR**  
defaultParametersSSm, 5

*Topic **genetic**  
defaultParametersGA, 4  
parEstimationLBode, 16  
parEstimationLBodeGA, 18

*Topic **incidence**  
incidence2Adjacency, 13

*Topic **logic**  
parEstimationLBode, 16  
parEstimationLBodeGA, 18

*Topic **matrix**  
incidence2Adjacency, 13

*Topic **model**  
parEstimationLBode, 16  
parEstimationLBodeGA, 18

*Topic **parameters**  
defaultParametersGA, 4

*Topic **states**  
getStates, 13

cnolistCNORodeExample, 2  
CNORode, 2, 13

createLBodeContPars, 3, 6–12, 14, 16–18, 20, 22–26

defaultParametersGA, 4  
defaultParametersSSm, 5

gLBodeContObjFunction, 6, 18, 20  
gLBodeDataSim, 7  
gLBodeMINLPObjFunction, 9, 14  
gLBodeModelSim, 3, 10  
gLBodeSimFunction, 12  
gGetStates, 13

incidence2Adjacency, 13, 13  
indices, 14

minLPBodeSSm, 14  
model, 16

parEstimationLBode, 3, 5, 6, 8, 16, 27  
parEstimationLBodeGA, 5, 16, 18  
parEstimationLBodeSSm, 6, 8, 16, 20, 27  
plotLBodeFitness, 3, 22  
plotLBodeModelSim, 24

rbga, 17, 20

simdata2cnolist, 25  
simulate, 26