

# Package ‘CNORode’

October 17, 2020

**Type** Package

**Title** ODE add-on to CellNOptR

**Version** 1.30.0

**Date** 2020-10-04

**Author** David Henriques, Thomas Cokelaer, Attila Gabor, Federica Eduati, Enio Gjerga

**Maintainer** Enio Gjerga <enio.gjerga@gmail.com>

**biocViews** ImmunoOncology, CellBasedAssays, CellBiology, Proteomics,  
Bioinformatics, TimeCourse

**Description** ODE add-on to CellNOptR

**License** GPL-2

**LazyLoad** yes

**Depends** CellNOptR (>= 1.5.14), genalg

**Enhances** MEIGOR

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

**git\_branch** RELEASE\_3\_11

**git\_last\_commit** 8c203be

**git\_last\_commit\_date** 2020-04-27

**Date/Publication** 2020-10-16

## R topics documented:

|                                    |    |
|------------------------------------|----|
| cnodata . . . . .                  | 2  |
| cnolist . . . . .                  | 2  |
| cnolistCNORodeExample . . . . .    | 2  |
| CNORode . . . . .                  | 2  |
| createLBodeContPars . . . . .      | 3  |
| crossvalidateODE . . . . .         | 5  |
| defaultParametersGA . . . . .      | 7  |
| defaultParametersSSm . . . . .     | 8  |
| getLBodeContObjFunction . . . . .  | 9  |
| getLBodeDataSim . . . . .          | 11 |
| getLBodeMINLPObjFunction . . . . . | 12 |
| getLBodeModelSim . . . . .         | 13 |
| getLBodeSimFunction . . . . .      | 15 |

|                                 |    |
|---------------------------------|----|
| getStates . . . . .             | 16 |
| incidence2Adjacency . . . . .   | 17 |
| indices . . . . .               | 17 |
| minlpLBodeSSm . . . . .         | 18 |
| model . . . . .                 | 20 |
| parEstimationLBode . . . . .    | 20 |
| parEstimationLBodeGA . . . . .  | 21 |
| parEstimationLBodeSSm . . . . . | 23 |
| pknmodel . . . . .              | 26 |
| plotLBodeFitness . . . . .      | 26 |
| plotLBodeModelSim . . . . .     | 28 |
| runCNORode . . . . .            | 29 |
| simdata2cnolist . . . . .       | 30 |
| simulate . . . . .              | 31 |

## Index 33

---

|         |                                 |
|---------|---------------------------------|
| cnodata | <i>A cnodata from CellNoptR</i> |
|---------|---------------------------------|

---

### Description

A cnodata from CellNoptR to use with provided examples

---

|         |                                 |
|---------|---------------------------------|
| cnolist | <i>A cnolist from CellNoptR</i> |
|---------|---------------------------------|

---

### Description

A cnolist from CellNoptR to use with provided examples

---

|                       |                                 |
|-----------------------|---------------------------------|
| cnolistCNORodeExample | <i>A cnolist from CellNoptR</i> |
|-----------------------|---------------------------------|

---

### Description

A cnolist from CellNoptR to use with provided CNORode examples.

---

|         |  |
|---------|--|
| CNORode | <i>Logic based ODE extension for CellNoptR</i> |
|---------|--|

---

### Description

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

### Details

Package: CNORode  
Type: Package  
Version: 1.2.0  
Date: 2012-03-14  
License: GPL-3  
LazyLoad: yes

### Author(s)

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

### References

Dominik Wittmann, Jan Krumsiek, Julio S. Rodriguez, Douglas Lauffenburger, Steffen Klamt, and Fabian Theis. Transforming boolean models to continuous models: methodology and application to t-cell receptor signaling. *BMC Systems Biology*, 3(1):98+, September 2009.

Egea, J.A., Maria, R., Banga, J.R. (2010) An evolutionary method for complex-process optimization. *Computers & Operations Research* 37(2): 315-324.

Egea, J.A., Balsa-Canto, E., Garcia, M.S.G., Banga, J.R. (2009) Dynamic optimization of nonlinear processes with an enhanced scatter search method. *Industrial & Engineering Chemistry Research* 49(9): 4388-4401.

Jan Krumsiek, Sebastian Polsterl, Dominik Wittmann, and Fabian Theis. Odefy - from discrete to continuous models. *BMC Bioinformatics*, 11(1):233+, 2010.

R. Serban and A. C. Hindmarsh, "CVODES: the Sensitivity-Enabled ODE Solver in SUNDIALS," Proceedings of IDETC/CIE 2005, Sept. 2005, Long Beach, CA. Also available as LLNL technical report UCRL-JP-200039.

C. Terfve, T. Cokelaer, A. MacNamara, D. Henriques, E. Goncalves, MK. Morris, M. van Iersel, DA Lauffenburger, J. Saez-Rodriguez. CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms. *BMC Systems Biology*, 2012, 6:133:

### See Also

[CellNOptR](#), [parEstimationLBode](#), [getLBodeModelSim](#), [parEstimationLBode](#) [plotLBodeFitness](#).

---

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      *k-fold crossvalidation for ODE models.*

---

**Description**

Cross-validation analysis for the logic-ode case.

**Usage**

```
crossvalidateODE = function(CNOList = cnolist, model = model, nfolds=10, foldid=NULL, type="datapoint",
ode_parameters = NULL, paramsSSm = NULL, method = "essm")
```

**Arguments**

|                |  |
|----------------|--|
| CNOList        | a CNOList on which the score is based (based on valueSignals[[2]], i.e. data at time 1).   |
| model          | a model structure, as created by readSIF, normally pre-processed but that is not a requirement of this function.   |
| 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       | verbose parameter, indicating wheter to parallelize the cross-validation procedure or not (default set to FALSE).  |
| ode_parameters | a list with the ODEs parameter information. Obtained with createLBodeContPars.   |
| paramsSSm      | a list of SSm parameters. default is the list returned bydefaultParametersSSm.   |

**Details**

Does a k-fold cross-validation for logic-ode 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.

**Value**

This function returns a list with elements:

|         |                         |
|---------|-------------------------|
| fit     | optimisation result     |
| cvScore | cross-validation scores |

**Author(s)**

A. Gabor, E. Gjerga

**References**

A. MacNamara, C. Terfve, D. Henriques, B.P. Bernabe, J. Saez-Rodriguez. State-time spectrum of signal transduction logic models, *Phys Biol.*, 9(4):045003, 2012. Toy example can be found here: <https://saezlab.github.io/CellNOptR/>

**Examples**

```
## Not run:
library(CellNOptR)
library(doParallel)

data("ToyModel", package="CellNOptR")
data("CN0listToy", package="CellNOptR")
pknmodel = ToyModel
cnolist = CN0list(CN0listToy)

# original and preprocessed network
plotModel(pknmodel,cnolist)
model = preprocessing(data = cnolist,model = pknmodel,compression = T,expansion = T)
plotModel(model,cnolist)
plotCN0list(CN0list = cnolist)

# set initial parameters
ode_parameters=createLBodeContPars(model, LB_n = 1, LB_k = 0,
                                   LB_tau = 0, UB_n = 4, UB_k = 1, UB_tau = 1, default_n = 3,
                                   default_k = 0.5, default_tau = 0.01, opt_n = FALSE, opt_k = TRUE,
                                   opt_tau = TRUE, random = TRUE)

## Parameter Optimization
# essm
paramsSSm=defaultParametersSSm()
paramsSSm$local_solver = "DHC"
paramsSSm$maxtime = 600;
paramsSSm$maxeval = Inf;
paramsSSm$atol=1e-6;
paramsSSm$reltol=1e-6;
paramsSSm$nan_fac=0;
paramsSSm$dim_refset=30;
paramsSSm$n_diverse=1000;
paramsSSm$maxStepSize=Inf;
paramsSSm$maxNumSteps=10000;
transferFun=4;
paramsSSm$transfer_function = transferFun;

paramsSSm$lambda_tau=0
```

```

paramsSSm$lambda_k=0
paramsSSm$bootstrap=F
paramsSSm$SSpenalty_fac=0
paramsSSm$SScontrolPenalty_fac=0

# run the optimisation algorithm
opt_pars=parEstimationLBode(cnolist,model, method="essm", ode_parameters=ode_parameters,
                           paramsSSm=paramsSSm)
plotLBodeFitness(cnolist = cnolist, model = model, ode_parameters = opt_pars,
                 transfer_function = 4)

# 10-fold crossvalidation using T1 data
# We use only T1 data for crossvalidation, because data in the T0 matrix is not independent.
# All rows of data in T0 describes the basal condition.

# Crossvalidation produce some text in the command window:
registerDoParallel(cores=3)
R=crossvalidateODE(CNolist = cnolist, model = model, type="datapoint", nfolds=3,
                  parallel = TRUE, ode_parameters = ode_parameters, paramsSSm = paramsSSm)

## End(Not run)

```

---

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**

```

maxeval      Inf
maxtime      100
ndiverse     NULL
dim_refset   NULL
local_solver NULL
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    0
SSpenalty_fac 0
SScontrolPenalty_fac
              0
boot_seed    sample(1:10000,1)

```



**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 planing 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, 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 <a href="#">createLBodeContPars</a> .                    |
| indices           | Indices to map data in the model. Obtained with <code>indexFinder</code> function from <code>CellNOptR</code> .    |
| 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 than 1.    |

|                                   |  |
|-----------------------------------|--|
| <code>lambda_tau</code>           | 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. |
| <code>lambda_k</code>             | 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.   |
| <code>bootstrap</code>            | 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.   |
| <code>SSpenalty_fac</code>        | Penalty factor for penalising solutions which do not reach steady state. Default =0.   |
| <code>SScontrolPenalty_fac</code> | Penalty factor for penalising solutions for which the control (unperturbed) condition (assumed to be first row) does not reach steady state. Default =0.   |
| <code>boot_seed</code>            | Seed used for random sampling if bootstrap=TRUE. Default chose random seed between 0 and 10000   |

**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**

```
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

```
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)
```

### Arguments

|                                |  |
|--------------------------------|--|
| <code>cnolist</code>           | A list containing the experimental design and data.  |
| <code>model</code>             | A list with the ODEs parameter information. Obtained with <a href="#">createLBodeContPars</a> .  |
| <code>ode_parameters</code>    | A list with the ODEs parameter information. Obtained with <code>makeParameterList</code> function.   |
| <code>indices</code>           | Indices to map data in the model. Obtained with <code>indexFinder</code> function from <code>CellNOptR</code> .  |
| <code>timeSignals</code>       | An array containing a different <code>timeSignals</code> . If you use this argument, it will also modify the dimensions from <code>valueSignals</code> . |
| <code>time</code>              | An integer with the index of the time point to start the simulation. Default is 1.   |
| <code>verbose</code>           | A logical value that triggers a set of comments.   |
| <code>transfer_function</code> | The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function.                                       |
| <code>reltol</code>            | Relative Tolerance for numerical integration.  |
| <code>atol</code>              | Absolute tolerance for numerical integration.  |
| <code>maxStepSize</code>       | The maximum step size allowed to ODE solver.   |
| <code>maxNumSteps</code>       | The maximum number of internal steps between two points being sampled before the solver fails.   |
| <code>maxErrTestsFails</code>  | 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=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 <a href="#">createLBodeContPars</a> .                       |
| indices           | Indices to map data in the model. Obtained with <a href="#">indexFinder</a> function from <a href="#">CellNOptR</a> . |
| 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 than 1. |

### Details

Check [CellNOptR](#) for details about the codelist 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(codelistCNORodeExample, 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 <a href="#">createLBodeContPars</a> .                              |
| indices           | Indices to map data in the model. Obtained with <code>indexFinder</code> function from <code>CellNOptR</code> .              |
| 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('ToyCN0list',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 <code>indexFinder</code> function from <code>CellNOptR</code> .    |
| odeParameters1     | A list with the ODEs parameter information. Obtained with <a href="#">createLBodeContPars</a> .                    |
| 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.                              |

**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 | <i>Search for the best combination of continuous parameters and logic gates.</i> |
|---------------|--|

---

### 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

|                                |  |
|--------------------------------|--|
| <code>cnolist</code>           | A list containing the experimental design and data.  |
| <code>model</code>             | The logic model to be simulated.   |
| <code>ode_parameters</code>    | A list with the ODEs parameter information. Obtained with <a href="#">createLBodeContPars</a> .                  |
| <code>int_x0</code>            | Vector with initial solution for integer parameters.   |
| <code>indices</code>           | Indices to map data in the model. Obtained with <code>indexFinder</code> function from <code>CellNOptR</code> .  |
| <code>maxeval</code>           | Maximum number of evaluation in the optimization procedure.  |
| <code>maxtime</code>           | Maximum number of evaluation spent in optimization procedure.  |
| <code>ndiverse</code>          | Duration of the optimisation procedure.  |
| <code>dim_refset</code>        | Number of diverse initial solutions.   |
| <code>local_solver</code>      | Local solver to be used in SSm.  |
| <code>time</code>              | An integer with the index of the time point to start the simulation. Default is 1.                               |
| <code>verbose</code>           | A logical value that triggers a set of comments.   |
| <code>transfer_function</code> | The type of used transfer. Use 1 for no transfer function, 2 for Hill function and for normalized Hill function. |
| <code>reltol</code>            | Relative Tolerance for numerical integration.  |
| <code>atol</code>              | Absolute tolerance for numerical integration.  |
| <code>maxStepSize</code>       | The maximum step size allowed to ODE solver.   |
| <code>maxNumSteps</code>       | The maximum number of internal steps between two points being sampled before the solver fails.                   |
| <code>maxErrTestsFails</code>  | Specifies the maximum number of error test failures permitted in attempting one step.                            |
| <code>nan_fac</code>           | 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**

|             |   |
|-------------|---|
| 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**

```
## 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 | <i>A model from CellNoptR</i> |
|-------|-------------------------------|

---

### Description

A model from CellNoptR to use with provided examples

---

|                    |  |
|--------------------|--|
| parEstimationLBode | <i>Perform parameter estimation using a genetic algorithm (package genalg) or ssm (if package essm available).</i> |
|--------------------|--|

---

### 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 <a href="#">createLBodeContPars</a> .      |
| indices        | Indices to map data in the model. Obtained with <a href="#">indexFinder</a> function from CellNoptR. |
| paramsGA       | A list of GA parameters. default is the list returned by <a href="#">defaultParametersGA</a> .       |
| paramsSSm      | A list of SSm parameters. default is the list returned by <a href="#">defaultParametersSSm</a> .     |

### 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**

```
data("ToyCN0list", 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)
```

---

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 = NA, 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 <a href="#">createLBodeContPars</a> .   |
| indices           | Indices to map data in the model. Obtained with <a href="#">indexFinder</a> function from <a href="#">CellNOptR</a> .   |
| 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.  |
| 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 than 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_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 nonlinear optimization solver (genalg).                     |

**Author(s)**

David Henriques, Thomas Cokelaer

**See Also**

[CellNOptR createLBodeContPars rgba](#)

**Examples**

```
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=parEstimationLBodeGA(cnolistCNORodeExample, model, ode_parameters=ode_parameters,
indices=indices, maxStepSize=1, atol=1e-3, reltol=1e-5, transfer_function=2, popSize=10, iter=40);

#Visual solution after optimization
simulatedData=plotLBodeFitness(cnolistCNORodeExample, model, indices=indices, ode_parameters=ode_parameters);
```

---

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 <a href="#">createLBodeContPars</a> .  |
| indices              | Indices to map data in the model. Obtained with <a href="#">indexFinder</a> 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 than 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("ToyCNOlist", 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

---

plotLBodeFitness              *Plot data against simulated values.*

---

### Description

Plots the simulated values with the logic-based ODE against the the data contained contained 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 functions returns the 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 <a href="#">createLBodeContPars</a> .                               |
| indices           | Indices to map data in the model. Obtained with <a href="#">indexFinder</a> function from CellNoptR.                          |
| adjMatrix         | Model representation in the form of an adjacency matrix. When not provided will be automatically computed based in 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("ToyCNolist", package="CNORode");
data("ToyModel", package="CNORode");
data("ToyIndices", package="CNORode");
ode_parameters=createLBodeContPars(model, random=TRUE);
dataSimulation=plotLBodeFitness(cnolistCNORodeExample, model, indices=indices);
```

---

|                   |   |
|-------------------|---|
| plotLBodeModelSim | <i>Simulate the model and plot the obtained with the different experimental conditions.</i> |
|-------------------|---|

---

### 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 <a href="#">createLBodeContPars</a> .                               |
| indices           | Indices to map data in the model. Obtained with <a href="#">indexFinder</a> function from <a href="#">CellNOptR</a> .         |
| 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              | 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**

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

---

runCNORode

*runCNORode*

---

**Description**

A one-line wrapper of the CNORode pipeline

**Usage**

```
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, 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 <a href="#">preprocessing</a>              |
| results_folder | results folder for the analysis.  |
| cutNONC        | cut non-observable non-controllable node from PKN (TRUE), see <a href="#">preprocessing</a> |
| expansion      | expand OR gates in the PKN (FALSE), see <a href="#">preprocessing</a>                       |
| LB_n           | lower bound on parameter n, see <a href="#">createLBodeContPars</a>                         |
| LB_k           | lower bound on parameter k, see <a href="#">createLBodeContPars</a>                         |
| LB_tau         | lower bound on parameter tau, see <a href="#">createLBodeContPars</a>                       |
| UB_n           | upper bound on parameter n, see <a href="#">createLBodeContPars</a>                         |

|                   |  |
|-------------------|--|
| UB_k              | upper bound on parameter k, see <a href="#">createLBodeContPars</a>  |
| UB_tau            | upper bound on parameter tau, see <a href="#">createLBodeContPars</a>  |
| default_n         | default value of parameter n, see <a href="#">createLBodeContPars</a>  |
| default_k         | default value of parameter k, see <a href="#">createLBodeContPars</a>  |
| default_tau       | default value of parameter tau, see <a href="#">createLBodeContPars</a>  |
| opt_n             | should parameter n be optimised, see <a href="#">createLBodeContPars</a>   |
| opt_k             | should parameter k be optimised, see <a href="#">createLBodeContPars</a>   |
| opt_tau           | should parameter tau be optimised, see <a href="#">createLBodeContPars</a>   |
| random            | initial parameter vector generation (TRUE: random, FALSE: half of the LB-UB)   |
| maxeval           | maximum number of function evaluations in the optimisation, see <a href="#">parEstimationLBodeSSm</a>                          |
| maxtime           | maximum CPU time (in seconds) spent on optimisation before calling final refinement, see <a href="#">parEstimationLBodeSSm</a> |
| transfer_function | transfer function types represented by the edges, see <a href="#">parEstimationLBodeSSm</a>                                    |
| nan_fac           | penalty for NA simulations, see <a href="#">parEstimationLBodeSSm</a>  |
| lambda_tau        | regularisation penalty for tau parameters, see <a href="#">parEstimationLBodeSSm</a>   |
| lambda_k          | regularisation penalty for k parameters for optimisation, see <a href="#">parEstimationLBodeSSm</a>                            |

### Examples

```
## Not run:
model = system.file("extdata", "ToyModelMMB_FeedbackAnd.sif", package="CNORode")
data = system.file("extdata", "ToyModelMMB_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

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

### Arguments

|          |   |
|----------|---|
| sim_data | structure returned by <code>getLBodeModelSim</code> |
| 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**

```

data('ToyCN0list',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**

|                                |  |
|--------------------------------|--|
| <code>cnolist</code>           | A list containing the experimental design and data.  |
| <code>model</code>             | A list with the ODEs parameter information. Obtained with <a href="#">createLBodeContPars</a> .                    |
| <code>ode_parameters</code>    | A list with the ODEs parameter information. Obtained with <code>makeParameterList</code> function.                 |
| <code>indices</code>           | Indices to map data in the model. Obtained with <code>indexFinder</code> function from <code>CellNOptR</code> .    |
| <code>adjMatrix</code>         | The adjacency matrix. Recomputed if not provided   |
| <code>time</code>              | An integer with the index of the time point to start the simulation. Default is 1.                                 |
| <code>verbose</code>           | A logical value that triggers a set of comments.   |
| <code>transfer_function</code> | The type of used transfer. Use 1 for no transfer function, 2 for Hill function and 3 for normalized Hill function. |
| <code>reltol</code>            | Relative Tolerance for numerical integration.  |
| <code>atol</code>              | Absolute tolerance for numerical integration.  |
| <code>maxStepSize</code>       | 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, 2
  - \* **CVODES**
    - getLBodeSimFunction, 15
  - \* **CellNOptR**
    - parEstimationLBode, 20
    - parEstimationLBodeGA, 21
  - \* **SSm**
    - defaultParametersSSm, 8
  - \* **adjacency**
    - getStates, 16
    - incidence2Adjacency, 17
  - \* **algorithm**
    - defaultParametersGA, 7
    - parEstimationLBode, 20
    - parEstimationLBodeGA, 21
  - \* **default**
    - defaultParametersGA, 7
    - defaultParametersSSm, 8
  - \* **essR**
    - defaultParametersSSm, 8
  - \* **genetic**
    - defaultParametersGA, 7
    - parEstimationLBode, 20
    - parEstimationLBodeGA, 21
  - \* **incidence**
    - incidence2Adjacency, 17
  - \* **logic**
    - parEstimationLBode, 20
    - parEstimationLBodeGA, 21
  - \* **matrix**
    - incidence2Adjacency, 17
  - \* **model**
    - parEstimationLBode, 20
    - parEstimationLBodeGA, 21
  - \* **parameters**
    - defaultParametersGA, 7
  - \* **states**
    - getStates, 16
- CellNOptR, 3, 8–14, 16, 17, 19, 21, 23, 25, 27, 29, 31, 32
- cnodata, 2
- cnolist, 2
- cnolistCNORodeExample, 2
- CNORode, 2, 16
- createLBodeContPars, 3, 9–15, 18–31
- crossvalidateODE, 5
- defaultParametersGA, 7
- defaultParametersSSm, 8
- getLBodeContObjFunction, 9, 21, 23
- getLBodeDataSim, 11
- getLBodeMINLPObjFunction, 12, 18
- getLBodeModelSim, 3, 13
- getLBodeSimFunction, 15
- getStates, 16
- incidence2Adjacency, 16, 17
- indices, 17
- minlpLBodeSSm, 18
- model, 20
- parEstimationLBode, 3, 8, 9, 12, 20, 32
- parEstimationLBodeGA, 8, 20, 21
- parEstimationLBodeSSm, 9, 12, 20, 23, 30, 32
- pknmodel, 26
- plotLBodeFitness, 3, 26
- plotLBodeModelSim, 28
- preprocessing, 29
- rbga, 21, 23
- runCNORode, 29
- simdata2cnolist, 30
- simulate, 31