Package ‘CNORdt’

March 22, 2017

Type  Package
Title  Add-on to CellNOptR: Discretized time treatments
Version  1.16.0
Date  2012-09-18
Author  A. MacNamara
Maintainer  A. MacNamara <aidan.macnamara@ebi.ac.uk>

Description  This add-on to the package CellNOptR handles time-course data, as opposed to steady state data in CellNOptR. It scales the simulation step to allow comparison and model fitting for time-course data. Future versions will optimize delays and strengths for each edge.

License  GPL-2
Depends  R (>= 1.8.0), CellNOptR (>= 0.99), abind
LazyLoad  yes

biocViews  CellBasedAssays, CellBiology, Proteomics, TimeCourse

NeedsCompilation  yes

R topics documented:

CNOlistPB ......................................................... 2
computeScoreDT ................................................. 2
convert2array .................................................. 3
cutAndPlotResultsDT ......................................... 4
gaBinaryDT ..................................................... 6
getFitDT .......................................................... 8
modelPB ........................................................... 9
simulatorDT ..................................................... 10

Index  12
**Description**

This data object contains the data associated with the toy model example from the package vignette, already loaded and formatted as a CNOlist object.

**Usage**

```r
CNOlistPB
```

**Format**

CNOlistPB is a list with the fields "namesCues" (character vector), "namesStimuli" (character vector), "namesInhibitors" (character vector), "namesSignals" (character vector), "timeSignals" (numerical vector), "valueCues" (numerical matrix), "valueInhibitors" (numerical matrix), "valueStimuli" (numerical matrix), "valueSignals" (numerical matrix).

**Source**

This data was generated with the CellNOptR add-on package CNORode. Full details of the data can be found in the reference below.

**References**


---

**Description**

Compute the score of a model/data (time-course) set using a bitString to cut the model.

**Usage**

```r
computeScoreDT(CNOlist, model, bString, simList=NULL, indexList=NULL, sizeFac=0.0001, NAFac=1, boolUpdates, lowerB=lowerB, upperB=upperB)
```
**convert2array**

Converting the output of simulatorDT to a 3D array for other functions.

**Description**

As above.

**Usage**

convert2array(x, nRow, nCol, nBool)

**Arguments**

- **CNOlist**: A CNOlist structure, as created by makeCNOlist.
- **model**: A model structure, as created by codereadSIF, normally pre-processed but that is not a requirement of this function.
- **bString**: A bit string of the same size as the number of reactions in the model above.
- **simList**: If provided, simList should be created by prep4sim, and has also already been cut to contain only the reactions to be evaluated.
- **indexList**: If provided, indexList should contain a list of indexes of the species stimulated/inhibited/measured in the model, as created by indexFinder.
- **sizeFac**: The scaling factor for the size term in the objective function, default to 0.0001.
- **NAFac**: The scaling factor for the NA term in the objective function, default to 1.
- **boolUpdates**: The number of synchronous updates performed by the boolean simulator.
- **lowerB**: The lower bound for the optimized value of the scaling factor.
- **upperB**: The upper bound for the optimized value of the scaling factor.

**Value**

- **score**: See gaBinaryT1 for details.

**Author(s)**

A. MacNamara

**Examples**

```r
library(CellNOptR)
library(CNORdt)
data(CNOlistPB, package="CNORdt")
data(modelPB, package="CNORdt")

# pre-process model
model = preprocessing(CNOlistPB, modelPB)

# compute score
score = computeScoreDT(CNOlistPB, model, bString=rep(1,16),
boolUpdates=10, lowerB=0.8, upperB=10)
```
cutAndPlotResultsDT

Plot the results of a time-course optimization

Description

This function takes the optimized bit string, cuts the model according to the string and then finds the optimized scaling factor to pass on to plotOptimResultsPan for visualization.

Usage

cutAndPlotResultsDT(model, bString, simList=NULL, CNOlist=NULL, indexList=NULL, plotPDF=FALSE, tag=NULL, plotParams=list(maxrow=10), boolUpdates=boolUpdates, lowerB=lowerB, upperB=upperB, sizeFac = 1e-04, NAFac = 1)
Arguments

model  The expanded model used as input for gaBinaryDT.
bString  A bit string as output by gabinaryDT (i.e. a vector of 1s and 0s).
simList  A simlist corresponding to the model, as output by prep4sim.
CNOlist  The CNOlist used in optimization.
indexList  An indexList, produced by indexFinder run on the model and the CNOlist above.
plotPDF  TRUE or FALSE; for pdf output.
tag  NULL or string; prefixes filenames with a tag (replaces the default behavior).
plotParams  A list of options related to the PDF and plotting outputs.  (1) maxrow is the maximum number of rows used to plot the results. See plotOptimResultsPan for other fields.
boolUpdates  The number of synchronous updates performed by the boolean simulator.
lowerB  The lower bound for the optimized value of the scaling factor.
upperB  The upper bound for the optimized value of the scaling factor.
sizeFac  The scaling factor for the size term in the objective function, default to 0.0001.
NAFac  The scaling factor for the NA term in the objective function, default to 1.

Value

This function doesn’t return anything, it only plots the graph in your graphic window.

Author(s)

A. MacNamara

See Also

plotOptimResultsPan

Examples

data(CNOlistPB, package="CNORdt")
data(modelPB, package="CNORdt")

# pre-process model
model <- preprocessing(CNOlistPB, modelPB)
indices = indexFinder(CNOlistPB, model)
fields4Sim <- prep4sim(model=model)
initBstring <- rep(1, length(model$reacID))

# optimize
opt1 <- gaBinaryDT(CNOlist=CNOlistPB, model=model, initBstring=initBstring,
   verbose=TRUE, boolUpdates=10, maxTime=30, lowerB=0.8, upperB=10)

cutAndPlotResultsDT(
   model=model,
   CNOlist=CNOlistPB,
   bString=opt1$bString,
   plotPDF=FALSE,
   boolUpdates=10,
   lowerB=0.8,
**Description**

The genetic algorithm used to optimize a model by fitting to data consisting of multiple time points. The data can be fitted by applying a single scaling factor to the boolean simulation.

**Usage**

```r
gaBinaryDT(CNOlist, model, initBstring = NULL, sizeFac = 1e-04, NAFac = 1, popSize = 50, pMutation = 0.5, maxTime = 60, maxGens = 500, stallGenMax = 100, selPress = 1.2, elitism = 5, relTol = 0.1, verbose = TRUE, priorBitString = NULL, maxSizeHashTable = 5000, boolUpdates, lowerB = lowerB, upperB = upperB)
```

**Arguments**

- `CNOlist`: A CNOlist on which the score is based.
- `model`: A model structure, as created by `readSIF`, normally pre-processed but that is not a requirement of this function.
- `initBstring`: An initial bitstring to be tested, should be of the same size as the number of reactions in the model above (model$reacID). The default is all ones.
- `sizeFac`: The scaling factor for the size term in the objective function, it defaults to 0.0001.
- `NAFac`: The scaling factor for the NA term in the objective function, it defaults to 1.
- `popSize`: The population size for the genetic algorithm, it is set to 50.
- `pMutation`: The mutation probability for the genetic algorithm, default set to 0.5.
- `maxTime`: The maximum optimisation time in seconds, default set to 60.
- `maxGens`: The maximum number of generations in the genetic algorithm, default set to 500.
- `stallGenMax`: The maximum number of stall generations in the genetic algorithm, default to 100.
- `elitism`: The number of best individuals that are propagated to the next generation in the genetic algorithm, default set to 5.
- `relTol`: The relative tolerance for the best bitstring reported by the genetic algorithm, i.e., how different from the best solution, default set to 0.1.
- `verbose`: Logical (default to TRUE): do you want the statistics of each generation to be printed on the screen?
- `priorBitString`: At each generation, the GA algorithm creates a population of bitstrings that will be used to perform the optimization. If the user knows the values of some bits, they can be used to overwrite bit values proposed by the GA algorithm. If provided, the priorBitString must have the same length as the initial bitstring and be made of 0, 1 or NA (by default, this bitstring is set to NULL, which is equivalent to setting all bits to NA). Bits that are set to 0 or 1 are used to replace the bits created by the GA itself (see example).
A hash table is used to store bitstrings and related scores. This allows the GA to be very efficient in the case of small models. The size of the hash table is 5000 by default, which may be too large for large models.

The number of synchronous updates performed by the boolean simulator.

The lower bound for the optimized value of the scaling factor.

The upper bound for the optimized value of the scaling factor.

This is the modified version of gaBinaryT1 from the CellNOptR package, which is able to use all data from CNOlist$valueSignals.

This function returns a list with elements:

- **bString**: The best bitstring.
- **StringsTol**: The bitstrings whose scores are within the tolerance.
- **StringsTolScores**: The scores of the above-mentioned strings.

A. MacNamara


getFitDT, simulatorDT

```r
library(CellNOptR)
library(CNORdt)
data(CNOlistPB, package="CNORdt")
data(modelPB, package="CNORdt")

# pre-process model
model = preprocessing(CNOlistPB, modelPB)

# optimise
initBstring <- rep(1, length(model$reacID))
opt1 <- gaBinaryDT(CNOlist=CNOlistPB, model=model, initBstring=initBstring, verbose=TRUE, boolUpdates=10, maxTime=30, lowerB=0.8, upperB=10)
```
getFitDT

The optimization function that finds the scaling factor for the boolean simulation

description
This function is called from gaBinaryDT. Using the model passed as input, it finds a scaling factor that minimizes the mean squared error between the data from the boolean simulation and the experimental data. A spline is fitted to the experimental data to allow this.

Usage
getFitDT(simResults, CNOlist, model, indexList, sizeFac = 1e-04, NAFac = 1, nInTot, boolUpdates, lowerB, upperB)

Arguments
- simResults: The simulation results as output from simulatorDT
- CNOlist: A CNOlist on which the score is based (based on all valueSignals).
- model: A model list.
- indexList: A list of indexes of species stimulated/inhibited/signals, as produced by indexfinder applied on the model and CNOlist above.
- sizeFac: The scaling factor for the size term in the objective function, default to 0.0001.
- NAFac: The scaling factor for the NA term in the objective function, default to 1.
- nInTot: The number of inputs in the model prior to cutting, used to normalise the size penalty.
- boolUpdates: The number of synchronous updates performed by the boolean simulator.
- lowerB: The lower bound for the optimized value of the scaling factor.
- upperB: The upper bound for the optimized value of the scaling factor.

details
The function optim() is used to find the optimal scaling factor.

value
This function returns a list with elements:
- score: The mean squared error between simulation and experiment with NA and model size penalties.
- estimate: The scaling factor used to compare boolean and real data.
- xCoords: The x-axis coordinates after multiplication with the scaling factor.
- yInter: The interpolated values of the experimental data.
- yBool: The boolean simulation results at each time point.

author(s)
A. MacNamara
## Description

This data object contains the toy model from the package vignette, already loaded and formatted as a Model object.

## Usage

Model

## Format

modelPB is a list with fields "reacID" (character vector), "namesSpecies" (character vector), "interMat" (numerical matrix), and "notMat" (numerical matrix).
**Source**

This data and model is from the Physical Biology tutorial, "State-time spectrum of signal transduction logic models". It is used to demonstrate the assumptions and limitations of different logic model formalisms.

**References**


---

**simulatorDT**

*Discrete time simulation of a boolean model*

**Description**

Simulates multiple time points within C (for speed).

**Usage**

```r
simulatorDT(CNOlist, model, simList, indices, boolUpdates, prevSim=NULL)
```

**Arguments**

- **CNOlist**
  A CNOlist.
- **model**
  A model that only contains the reactions to be evaluated.
- **simList**
  A simList as created by prep4sim, that has also already been cut to contain only the reactions to be evaluated.
- **indices**
  An indexList as created by indexFinder.
- **boolUpdates**
  The number of update rounds the simulator should run for.
- **prevSim**
  The results from simulatorT0 can be used here as initial conditions.

**Value**

A 3-dimensional array that gives the value of all species under each condition at each update (conditions, species, update).

**Author(s)**

A. MacNamara

**See Also**

gaBinaryDT, getFitDT
**Examples**

```r
# this computes the output of the full model,
# which is normally not done on a stand alone basis,
# but if you have a model and would like to visualise
# its output compared to your data, then this is what you should do.

library(CellNOptR)
library(CNORdt)
data(CNOlistPB, package="CNORdt")
data(modelPB, package="CNORdt")

indexOrig <- indexFinder(CNOlistPB, modelPB, verbose=TRUE)
fields4Sim <- prep4sim(modelPB)
boolUpdates=10

simResults <- simulatorDT(
  CNOlist=CNOlistPB,
  model=modelPB,
  simList=fields4Sim,
  indices=indexOrig,
  boolUpdates=boolUpdates
)

simResults = convert2array(simResults, dim(CNOlistPB-valueSignals[[1]][1]),
  length(modelPB$namesSpecies), boolUpdates)
```
Index

«Topic **datasets**
   CN0listPB, 2
   modelPB, 9

CN0listPB, 2
computeScoreDT, 2
convert2array, 3
cutAndPlotResultsDT, 4
gaBinaryDT, 6
getFitDT, 8
modelPB, 9
plotOptimResultsPan, 5
simulatorDT, 10