Package 'CNORfuzzy'

April 12, 2022

Type Package

Title Addon to CellNOptR: Fuzzy Logic

Version 1.36.0

Date 2013-8-28

Author M. Morris, T. Cokelaer

Maintainer T. Cokelaer <cokelaer@ebi.ac.uk>

Description This package is an extension to CellNOptR. It contains additional functionality needed to simulate and train a prior knowledge network to experimental data using constrained fuzzy logic (cFL, rather than Boolean logic as is the case in CellNOptR). Additionally, this package will contain functions to use for the compilation of multiple optimization results (either Boolean or cFL).

License GPL-2

Depends R (>= 2.15.0), CellNOptR (>= 1.4.0), nloptr (>= 0.8.5)

Suggests xtable, Rgraphviz, RUnit, BiocGenerics

LazyLoad yes

biocViews Network

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

git_branch RELEASE_3_14

git_last_commit d6315a5

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`CNORfuzzy-package` *R version of CNOFuzzy, a Constrained Fuzzy Logic Network Optimisation*

Description

This package does optimisation of constrained Fuzzy logic networks of signalling pathways based on a previous knowledge network and a set of data collected upon perturbation of some of the nodes in the network.

Details

Package:	CNOR
Type:	Package
Version:	1.4.0
Date:	2013-08-28
License:	GPL-2
LazyLoad:	yes
Depends:	R (>= 2.15.0), CellNOptR (>= 1.3.29), nloptr (>= 0.8.5)

Author(s)

M.K. Morris
Maintainer: T. Cokelaer <cokelaer@ebi.ac.uk>

References

1. J. Saez-Rodriguez, L. G. Alexopoulos, J. Epperlein, R. Samaga, D. A. Lauffenburger, S. Klamt and P. K. Sorger. Discrete logic modeling as a means to link protein signaling networks with functional analysis of mammalian signal transduction, Molecular Systems Biology, 5:331, 2009.

2. Morris MK, Saez-Rodriguez J, Clarke DC, Sorger PK, Lauffenburger DA (2011). Training Signaling Pathway Maps to Biochemical Data with Constrained Fuzzy Logic: Quantitative Analysis of Liver Cell Responses to Inflammatory Stimuli. PLoS Comput Biol. 7(3): e1001099.

See Also

CellNOptR package.

Examples

```
    # Get data from CellNOptR package
    data(CNOlistToy,package="CellNOptR")
    data(ToyModel,package="CellNOptR")

    # Use the default parameters and set Data and Model
    paramsList=defaultParametersFuzzy()
    paramsList$data<-CNOlistToy
    paramsList$model<-ToyModel

## Not run:
    # Run the simulator
    Res = CNORwrapFuzzy(data=CNOlistToy, model=ToyModel, paramsList=paramsList)

## End(Not run)
```

CNORwrapFuzzy *CNORfuzzy analysis wrapper*

Description

This function is a wrapper around the whole CNOR Fuzzy analysis. It performs the following steps:

1. Plot the CNOlist
2. Checks data to model compatibility
3. Pre-processing steps
4. Prepare for simulation (see prep4simFuzzy)
5. Optimisation using Fuzzy transfer function (see gaDiscreteT1)
6. Refinement and reduction steps (see getRefinedModel and reduceFuzzy).

Usage

```
CNORwrapFuzzy(data, model, paramsList=NULL, verbose=TRUE)
```

Arguments

data	a CNOlist structure (as created by makeCNOlist) that contains the data that you will use (see readMIDAS and readSIF from CellNOptR).
model	the model that you want to optimise
paramsList	Use defaultParametersFuzzy function to create a template. Entries are 3-types: (i) GA algorithm parameters for the optimisation, (ii) Fuzzy parameters for the transfer functions and (iii) internal optimisation parameters within the refinement step. See defaultParametersFuzzy function for details on the parameters.
verbose	

Details

If you do not provide a parameter list argument, defaultParametersFuzzy is called internally to populate the paramsList argument.

Value

This function returns an object containing the results that can be used by other functions such as compileRes.

Author(s)

M.K. Morris

References

CNORWrap function from CellNOptR package

See Also

defaultParametersFuzzy, compileMultiRes

Examples

```
# Load some data
data(CNOlistToy, package="CellNOptR")
data(ToyModel, package="CellNOptR")
# Get some default parameters to play with, limiting the duration of the GA
# algorithm and optimisation step
paramsList = defaultParametersFuzzy()
paramsList$maxTime = 20
paramsList$optimisation$maxtime = 10
results = CNORwrapFuzzy(CNOlistToy, ToyModel, paramsList)
```

compileMultiRes *Compiles results from multiple runs and produces graph for choosing Post Refinement Threshold*

Description

This function takes a list of objects returned by CNORwrapfuzzy (run using identical parameters, models, and data) and packages them together so they can be compared with plotMeanFuzzyFit and writeFuzzyNetwork. Because almost all training of cFL models are underdetermined problems, analyzing multiple runs together is essential.

Usage

```
compileMultiRes(allRes, tag=NULL, show=TRUE)
```

Arguments

allRes	list of objects returned by the CNORwrapFuzzy function.
tag	If provided, save the results in 3 files. Each file starts with the string "filename" that is provided. (<tag>_allRes.RData, <tag>_allFinalMSEs.RData and <tag>_allFinalNumParams.RData)
show	plot the MSE and mean number of parameters versus threshold. Can be switch off if show=FALSE

Author(s)

M.K. Morris, T. Cokelaer

Examples

```
data(ToyModel, package="CellNOptR")
data(CNOlistToy,package="CellNOptR")
paramsList = defaultParametersFuzzy(CNOlistToy, ToyModel)
N = 10
allRes = list()
## Not run:
    for (i in 1:N){
        Res = CNORwrapFuzzy(CNOlistToy, ToyModel, paramsList)
        allRes[[i]] = Res
    }

    summary = compileMultiRes(allRes)
    summary$allFinalMSEs
    summary$allFinalNumParams

    # You can save the resuls in files using the tag argument
    compileMultiRes(allRes, "output")
```

```
## End(Not run)
```

`computeScoreFuzzy` *Compute Score of a model compared to the data for a given intString.*

Description

compute and return score of a model (cut using a bitstring).

Usage

```
computeScoreFuzzy(CNOlist, model,simList=NULL, indexList=NULL, paramsList,
    intString=NULL, sizeFac=0.0001,NAFac=1)
```

Arguments

`CNOlist`	a CNOlist on which the score is based (based on valueSignals[[2]], i.e. data at t1)
`model`	a model list
`simList`	a list that contains additional fields for the simulator, as created by prep4sim applied to the model above
`indexList`	a list of indexes of species stimulated/inhibited/signals, as produced by indexfinder applied on the model and CNOlist above
`paramsList`	list of parameters. See defaultParametersFuzzy
`intString`	a bitstring of the same size as the number of reactions in the model 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

Value

`score`	See gaBinaryT1 for details

Author(s)

T. Cokelaer

defaultParametersFuzzy *Create a list of default parameters*

Description

Parameters are required at different levels in the Fuzzy optimisation. This function provides a list with all parameters that are necessary.

Usage

```
defaultParametersFuzzy(data=NA, model=NA, nTF=7)
```

Arguments

data	the CNOlist that contains the data that you will use
model	the model that you want to optimise
nTF	number of discrete values to be used for each transfer function parameter.

Details

The list contains 3 types of parameters:

- Fuzzy parameters (e.g, Type1Funs, Type2Funs, RedThresh, DoRefinement)
- GA parameters similar to those used in CellNOptR package (see gaDiscreteT1 or defaultParametersFuzzy
- optimisation parameters related to the refinement step.
  1. algorithm='NLOPT_LN_SBPLX'
  2. xtol_abs=0.001
  3. maxEval=1000
  4. maxTime=5*60
- In addition, you can set Model and Data (CNOlist).

Value

params	a list of default parameters.

Author(s)

T. Cokelaer

Examples

```
data(ToyModel,package="CellNOptR")
data(CNOlistToy,package="CellNOptR")
params = defaultParametersFuzzy(CNOlistToy, ToyModel)
```

`gaDiscreteT1` *Genetic algorithm used to optimise a cFL model*

Description

This function is the genetic algorithm to be used to optimise a cFL model by fitting to data containing one time point.

Usage

```
gaDiscreteT1(CNOlist, model, paramsList, initBstring=NULL, sizeFac=0.0001,
    NAFac=1, popSize=50, pMutation=0.5, maxTime=60, maxGens=500,
    stallGenMax=100, selPress=1.2, elitism=5, relTol=0.1,
    verbose=TRUE,maxSizeHashTable = 1000)
```

Arguments

`CNOlist`	a CNOlist on which the score is based (based on valueSignals[[2]], i.e. data at t1)
`model`	a Model list
`paramsList`	CellNOptR software parameters (this functions uses transfer functions to choose from)
`initBstring`	an initial bitsring to be tested, should be of the same size as the number of reactions in the model 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.
`popSize`	the population size for the genetic algorithm, default 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.
`selPress`	the selective pressure in the genetic algorithm, default set to 1.2.
`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 can solutions be to be reported as well, default set to 0.1.
`verbose`	logical (default to TRUE) do you want the statistics of each generation to be printed on the screen?
`maxSizeHashTable`	a hash table is use to store bitstring and related score. This allows the GA to be very efficient is the case of small models. The size of the hash table is 5000 by default, which may be too large for large models

Details

The GA procedure is implemented based on the gaBinaryT1 in CellNOptR (see those man pages for a basic description). Necessary extensions to optimize a string of numbers rather than zero and one have been made. Additionally, since the scoring function is defined inside the function, it has also been altered for this function.

The parameters are similar to those used in CellNOptR and the returned list contains similar results as well.

Value

This function returns a list with elements:

`bString`	The best bitstring
`results`	A matrix with columns "Generation", "Best_score", "Best_bitString", "Stall_Generation", "Avg_Score_Gen", "Best_score_Gen" ,"Best_bit_Gen", "Iter_time".
`stringsTol`	The bitstrings whose scores are within the tolerance
`stringsTolScores`	The scores of the above-mentioned strings

Author(s)

M. Morris based on gaBinaryT1 function by C. Terfve (CellNOptR package)

See Also

prep4simFuzzy, simFuzzyT1

Examples

```
data(CNOlistToy,package="CellNOptR")
data(ToyModel,package="CellNOptR")

#pre-process model
model <- preprocessing(CNOlistToy, ToyModel, verbose=FALSE)

#set parameters
paramsList <- defaultParametersFuzzy(CNOlistToy, ToyModel)


# the GA algorithm
ToyT1opt<-gaDiscreteT1(
 CNOlist=CNOlistToy,
 model=model,
 paramsList=paramsList,
 maxTime=3,
 verbose=FALSE)
```

`getRefinedModel` *Refinement of Parameters of cFL model*

Description

Performs refinement of cFL model parameters

Usage

```
getRefinedModel(res,CNOlist,cutModel, cutSimList, refParams)
```

Arguments

`res`	Optimum returned by gaDiscreteT1
`CNOlist`	a CNOlist on which the score is based (based on valueSignals[[2]], i.e. data at T1)
`cutModel`	Model (with unnesssary edges cut in reduceFuzzy or interpretDiscreteGA
`cutSimList`	Fields for simulation based on cutModel (again, cut with reduceFuzzy or interpretDiscreteGA)
`refParams`	parameter list object as returned by defaultParametersFuzzy.

Details

After the discreteGA chooses transfer functions from a discrete set of transfer functions and removing interactions inconsistent with the data, this function 'refines' the parameters by using optim to go to the local minimum of error to data

Value

`refModel`	A refined model
`finalSet`	Set of final fuzzy parameter
`MSE`	The MSE value

Author(s)

M.K. Morris

See Also

gaDiscreteT1, reduceFuzzy, interpretDiscreteGA

`interpretDiscreteGA` *Interpreter of output of discrete genetic algorithm*

Description

This function takes the integer string output by the discrete genetic algorithm for training a cFL model and generates the corresponding model based on the Fuzzy parameters.

Usage

```
interpretDiscreteGA(model,  paramsList, intString, bitString=NULL)
```

Arguments

`model`	PKN trained (same model input as to gaDiscrete).
`paramsList`	List containing parameters (see defaultParametersFuzzy). Only the fuzzy parameters are used.
`intString`	Integer string resulting from gaDiscrete (in bString field of gaDiscreteT1 output).
`bitString`	(optional) if you want to cut additional interactions from the model. Used in reduceFuzzy function.

Details

After the discreteGA chooses transfer functions from a discrete set of transfer functions and removing interactions inconsistent with the data, this function interprets the output and returns an actual model using these transfer functions as well as a model from which logical redundancy was cut.

Value

`model`	The selected initial model based on the provided bitstring.
`simList`	The corresponding data related to Model field
`bitString`	The bitstring corresponding to the Model field
`cutModel`	Same as Model but redundant reactions are also removed.
`cutSimList`	The corresponding data related to cutModel field
`cutBitString`	The corresponding bitstring related to cutModel field.

Author(s)

M.K. Morris

See Also

gaDiscreteT1

Examples

```
data(ToyModel, package="CellNOptR")
data(CNOlistToy,package="CellNOptR")
paramsList = defaultParametersFuzzy()
## Not run:

    # preprocessing (see CNORwrapFuzzy or gaDiscreteT1)
    T1opt = gaDiscreteT1(....)  # see CNORwrapFuzzy or gaDiscreteT1 for details
    interpretDiscreteGA(ToyModel, paramsList, T1opt$bString)

## End(Not run)
```

`plotMeanFuzzyFit` *Simulates models returned from multiple cFL runs and plots mean fit to data*

Description

Uses post refinement threshold (selection threshold) to choose reduced refined model resulting from each run. Simulates model and plots result and fit to data

Usage

```
plotMeanFuzzyFit(postRefThresh, allFinalMSEs, allRes, plotPDF=FALSE, tag=NULL,
show=TRUE, plotParams=list(cex=0.8, cmap_scale=1))
```

Arguments

`postRefThresh`	Post refinement threshold (selection threshold) chosen from plot produced by compileMultiRes.
`allFinalMSEs`	matrix containing MSEs produced by compileMultiRes
`allRes`	list containing results of several CNORwrapFuzzy runs
`plotPDF`	TRUE or FALSE depending on if a PDF file should be saved
`tag`	String to include in filename of PDF plot
`show`	If the plot should be displayed
`plotParams`	a list of option related to the PDF and plotting outputs. (1) cex is the font size of the header. (2) cmap_scale below 1 allows to put more emphasizes on low errors (default 1 means all colors have the same weight). See `plotOptimResultsPan` from CellNOptR for other fields.

Value

This function does not have any output, it just plots and saves results if applicable.

Author(s)

M.K. Morris

Examples

```
data(ToyModel, package="CellNOptR")
data(CNOlistToy,package="CellNOptR")
paramsList = defaultParametersFuzzy(CNOlistToy, ToyModel)
N = 10
allRes = list()

## Not run:
for (i in 1:N){
    Res = CNORwrapFuzzy(CNOlistToy, ToyModel, paramsList)
    allRes[[i]] = Res
}

summary = compileMultiRes(allRes)
plotMeanFuzzyFit(0.1, summary$allFinalMSEs, allRes)


## End(Not run)
```

`prep4simFuzzy` *Prepare a model for simulation*

Description

Adds to the model some fields that are used by the simulation engine and calls prep4sim function from CellNOptR package

Usage

```
prep4simFuzzy(model, paramsList, verbose=TRUE)
```

Arguments

`model`	A model, as output by readSIF, normally pre-processed but that is not a requirement of this function.
`paramsList`	A parameters list that must contain at least the CNOlist parameter in the *Data* field (param = list(data=cnolist)) and possibly optional arguments related to the fuzzy logic (see defaultParametersFuzzy)
`verbose`	a verbose option set to TRUE by default.

Details

This adds fields that are necessary for the simulation engine for both Boolean and constrained Fuzzy logic simulation.

Value

In addition to the fields returned by the prep4sim function of CellNOptR, this function appends the following fields:

`finalCube`	stores, for each reac(row) the location of its inputs (col)
`ixNeg`	stores, for each reac(row) and each input (col) whether it is a negative input
`ignoreCube`	logical matrix of the same size as the 2 above, that tells whether the particular cell is filled or not
`maxIx`	row vector that stores, for each reac, the location of its output
`modelname`	stores the name of the model from which these fields were derived

Author(s)

C. Terfve, modified by M.K. Morris and T. Cokelaer

See Also

simFuzzyT1

Examples

```
data(ToyModel, package="CellNOptR")
data(CNOlistToy, package="CellNOptR")
params <- defaultParametersFuzzy()
params$data = CNOlistToy
fields4sim <- prep4simFuzzy(ToyModel, params)
```

---

reduceFuzzy *Remove unnecessary interactions from cFL model*

Description

Determine if interactions in cFL model are necessary to fit the data

Usage

```
reduceFuzzy(firstCutOff, CNOlist, model,  res, params )
```

Arguments

`firstCutOff`	Threshold for removing or replacing and edge. If the score doesn't get any worse than this, its removed or replaced
`CNOlist`	a CNOlist on which the score is based (based on `valueSignals[[2]]`, i.e. data at t1)
`model`	Model input into gaDiscreteT1 (uncut)
`res`	Optimum returned by gaDiscrete
`params`	parameters (as generated from defaultParametersFuzzy (only Fuzzy parameters required)

Details

After gaDiscrete removes interactions that are inconsistent with the data, this function asks if they are necessary to fit the data. For all interactions, it asks if they can be removed without making the score worse by a value of the firstCutOff. For AND interactions, it asks if they can be replaced with an OR gate.

Value

`redModel`

`redSimList`

`bitString`

`intString`

`MSE`

Author(s)

M.K. Morris

See Also

gaDiscreteT1

`simFuzzyT1` *Simulation of a cFL model*

Description

This is the simulator, inspired from CNOfuzzySimEngv23 in the Matlab CellNOpt, to be used on one time point simulations

Usage

```
simFuzzyT1(CNOlist, model, simList)
```

Arguments

`CNOlist`	a CNOlist
`model`	a Model that only contains the reactions to be evaluated
`simList`	a SimList as created by prep4simFuzzy, that has also already been cut to contain only the reactions to be evaluated

Details

I'm not sure if this applies to the one I wrote, which was based on I think an older version by Camille maybe? Differences from the BoolSimEngMKM simulator include: the valueInhibitors has not been previously flipped; the function outputs the values across all conditions for all species in the model, instead of only for the signal species. This is because then the output of this function can be used as initial values for the version of the simulator that works on time point 2 (not implemented in this version).

If you would like to compute the output of a model that contains some of the gates in the model but not all, we suggest that you use the function `SimulateT1` and specify in the `bStringT1` argument which gates you want to be included. Indeed, SimulateT1 is a wrapper around simulatorT1 that takes care of cutting the model for you before simulating it.

Value

This function outputs a single matrix of format similar to valueSignals in the CNOlist but that contains an output for each species in the model. This matrix is the simulated equivalent of valueSignals at time 1, if you consider only the columns given by indexSignals.

Author(s)

M.K. Morris based on function by C. Terfve from CellNOptR package.

See Also

`SimulateT1` from CellNOptR

Examples

```
#This computes the output of the full model, which is normally not done on a stand alone basis, but if you have a mode
    data(CNOlistToy,package="CellNOptR")
    data(ToyModel,package="CellNOptR")

    paramsList <- defaultParametersFuzzy(data=CNOlistToy, model=ToyModel)

    indicesToy<-indexFinder(CNOlistToy,ToyModel,verbose=TRUE)
    ToyFields4Sim<-prep4simFuzzy(ToyModel,paramsList)

    Sim<-simFuzzyT1(
     CNOlist=CNOlistToy,
     model=ToyModel,
     simList=ToyFields4Sim)
```

---

writeFuzzyNetwork *Despict the network results of training a cFL model to data in multiple runs.*

Description

Fuzzy network results output.

Usage

```
writeFuzzyNetwork(postRefThresh, allFinalMSEs, allRes, tag=NULL,verbose=FALSE)
```

Arguments

postRefThresh	Post refinement threshold (selection threshold) chosen from plot produced by compileMultiRes
allFinalMSEs	matrix containing MSEs produced by compileMultiRes
allRes	list containing all results produced by compileMultiRes
tag	String to include in filename of pdf plot
verbose	If extra warnings should be displayed

Details

The weights of the edges are computed as the mean across models using post refinement threshold (selection threshold) to choose reduced refined model resulting from each run.

As with writeNetwork, this function maps back the edges weights from the optimised (expanded and compressed) model to the original model. The mapping back only works if the path has length 2 at most (i.e. you have node1-comp1-comp2-node2, where comp refer to nodes that have been compressed).

Value

This function does not have any output, it just writes a SIF file, an edge attribute file, and a node attribute file

Note

The mapback of this function is still an open question, even in the Matlab version. Future developments will include more robust versions of the mapping back algorithm, probably as a separate mapback function.

Author(s)

M.K. Morris based on code by C. Terfve

See Also

writeNetwork

Examples

```
        data(ToyModel, package="CellNOptR")
        data(CNOlistToy,package="CellNOptR")
        paramsList = defaultParametersFuzzy(CNOlistToy, ToyModel)
        N = 10
        allRes = list()
 ## Not run:
```

```
        for (i in 1:N){
            Res = CNORwrapFuzzy(CNOlistToy, ToyModel, paramsList)
            allRes[[i]] = Res
        }

        summary = compileMultiRes(allRes)
        summary$allFinalMSEs
        summary$allFinalNumParams

        writeFuzzyNetwork(postRefThresh, summary$allFinalMSEs, allRes)

## End(Not run)
```

Index