Package 'sybilcycleFreeFlux'

June 1, 2016

Type Package

Title Cycle-Free Flux Balance Analysis

Version 2.0.0

Date 2016-05-30

Maintainer Abdelmoneim Amer Desouki <abdelmoneim.amer@uni-duesseldorf.de>

Author Abdelmoneim Amer Desouki [aut, cre]

Depends R (>= 3.0.3), sybil, Matrix, MASS

Imports methods

Suggests cplexAPI (>= 1.2.6), glpkAPI (>= 1.2.1)

Description Implement cycle-Free flux balance analysis, flux variability, and Random Sampling of solution space. Flux balance analysis is a technique to find fluxes in metabolic models at steady state. It is described in Orth, J.D., Thiele, I. and Palsson, B.O. What is flux balance analysis? Nat. Biotech. 28, 245-248 (2010).

LazyLoad yes

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-01 14:11:13

R topics documented:

sybilcycleFreeFlux-package *Find flux distribution free of cycles*

Description

The package sybilcycleFreeFlux implements some ideas to get rid of cycles. Also implement sampling and loopless sampling.

Details

Package:	sybilcycleFreeFlux
Type:	Package
Version:	0.0.1
Date:	2013-06-03
License:	GPL Version 3
LazyLoad:	yes
Depends:	sybil, methods

Author(s)

Abdelmoneim Amer Desouki

See Also

sybil cfFBA

ACHR *Random Sampling of Solution Space*

Description

implements sampling algorithm

Usage

```
ACHR(model, W = 2000, nPoints = 5000, stepsPerPoint = 10,
solver = SYBIL_SETTINGS("SOLVER"), method = SYBIL_SETTINGS("METHOD"))
```

Arguments

model	An object of class modelorg.
W	Number of warmup points. It should be more than double the number of reactions of the model.
nPoints	Number of points to be generated
stepsPerPoint	number of steps per point, default is 10 steps.
solver	Single character value. The solver to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
method	Single character value. The optimization algorithm to use. Possible values depend on the setting in solver. See SYBIL_SETTINGS for possible values. Default: LP_METHOD(SYBIL_SETTINGS).

Details

Starts by calculating warm up points

Author(s)

Abdelmoneim Amer Desouki

See Also

modelorg cfFBA

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
library(sybilcycleFreeFlux)
data(Ec_core);
model=Ec_core;
solver="cplexAPI"
W=500
nPnts=1000
s1=ACHR(model,W,nPoints=nPnts,stepsPerPoint=10)


sFVA=fluxVar(model,solver=solver)
fva_min=sFVA@lp_obj[(c(1:length(react_id(model))))];
fva_max=sFVA@lp_obj[(c((length(react_id(model))+1):(2*length(react_id(model)))) )];
table(lp_stat(sFVA))

pnts=s1$Points
fvamin=apply(pnts,1,min)
fvamax=apply(pnts,1,max)
```

```
write.csv(file="fva.csv",cbind(fva_min,fvamin,fva_max,fvamax,lb=lowbnd(model),
ub=uppbnd(model)))
#####Plot samples
bmrxn=which(obj_coef(model)==1)
bmrow=S(model)[bmrxn,]

objvals=NULL
solver="glpkAPI"
nRxns=react_num(model);
llpnts=  matrix(rep(0,nRxns*nPnts),ncol=nPnts);
for(i in 1:nPnts){
objvals=rbind(objvals,obj= pnts[bmrxn,i])
lrf=lrFBA(model,wtflux=pnts[,i],solver=solver,objVal= pnts[bmrxn,i])
llpnts[,i]=lrf$fluxes;
#Sys.time()
print(sprintf("point %d:%f",i,objvals[i]))
}
llfvamin=apply(llpnts,1,min)
llfvamax=apply(llpnts,1,max)

 write.csv(file="objv.csv",objvals)
write.csv(file="llfva.csv",cbind(fva_min,llmin=llfvamin,fva_max,llmax=llfvamax,fvamin,
fvamax,lb=lowbnd(model),ub=uppbnd(model)))
nloopflux=NULL
loopflxll=NULL
loopflxlp=NULL

for(i in (1:length(react_id(model))))
for(j in (1:nPnts)){
#print(c(i,j))
if(abs(pnts[i,j]-llpnts[i,j])<1e-7){
nloopflux=c(nloopflux,pnts[i,j])
}else{
loopflxll=c(loopflxll,llpnts[i,j])
loopflxlp=c(loopflxlp,pnts[i,j])
}
}
layout(matrix(c(1,2,3,1,2,3), 2, 3, byrow = TRUE))
hist(log10(abs(loopflxlp)),col="lightblue",main="a-loop fluxes",xlim=c(-3,3),
xlab="Log10(flux)")
hist(log10(abs(loopflxll)),col="orange",main="b-using cycleFreeFlux",
xlim=c(-3,3),xlab="Log10(flux)")
hist(log10(abs(nloopflux)),col="lightgreen",main="c-non-loop fluxes",
xlim=c(-3,3),xlab="Log10(flux)")


## End(Not run)
```

---

cfFBA *cycle free flux*

Description

finds a cycle free flux distribution given a flux distribution and a network model.

Usage

```
cfFBA(model, wtflux, objVal = NA #min objval
,fixExchRxn=TRUE
,excReactPos=NULL
 ,lpdir = SYBIL_SETTINGS("OPT_DIRECTION")
 ,solver = SYBIL_SETTINGS("SOLVER")
         ,method = SYBIL_SETTINGS("METHOD")
 ,tol=SYBIL_SETTINGS("TOLERANCE")
         ,solverParm=NA
,verboseMode = 2
,safeBounds=FALSE #
######### ADDED BY GABRIEL ###############
,retOptSol = TRUE)
```

Arguments

`model`	An object of class `modelorg`.
`wtflux`	initial flux distribution that may contain loops.
`objVal`	value of the objective function.
`fixExchRxn`	a logical value default is true, which indicates that the exchange reactions should be fixed or not.
`excReactPos`	list of the exchange reactions that can be sent to avoid recalculation
`lpdir`	Character value, direction of optimisation. Can be set to `"min"` or `"max"`. Default: `SYBIL_SETTINGS("OPT_DIRECTION")`.
`solver`	Single character value. The solver to use. See `SYBIL_SETTINGS` for possible values. Default: `SYBIL_SETTINGS("SOLVER")`.
`method`	Single character value. The optimization algorithm to use. Possible values depend on the setting in `solver`. See `SYBIL_SETTINGS` for possible values. Default: `LP_METHOD(SYBIL_SETTINGS)`.
`solverParm`	additional parameters to the solver
`tol`	tolerance of resulting solution
`safeBounds`	can be used to deal with false infeasibility state returned by solvers
`verboseMode`	level of displaying messages.
`retOptSol`	If set to `TRUE`, the function returns an object of class `optsol_optimizeProb`, otherwise a list.

Value

return a list that contains the status of the solution , the objective value and the fluxes that are free of cycles.

Author(s)

Abdelmoneim Amer Desouki

See Also

modelorg

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
data(iAF1260)
model=iAF1260
fba=optimizeProb(model)
cfopt=cfFBA(model,wtflux=getFluxDist(fba),objVal=lp_obj(fba), retOptSol=FALSE)
llflx=cfopt$fluxes
flx=getFluxDist(fba)
## Not run:
layout(matrix(c(1,2,3,1,2,3), 2, 3, byrow = TRUE))
hist(log10(abs(flx[abs(llflx-flx)>1e-3])),main="loop flux",col="lightblue")
hist(log10(abs(llflx[abs(llflx-flx)>1e-3])),main="after removing loops",col="orange")
hist(log10(abs(flx[abs(llflx-flx)<1e-3])),main="fluxes not in loops",col="lightgreen")

## End(Not run)
## The function is currently defined as
"cfFBA"
```

---

cfFVA *cycle free flux variability*

Description

finds flux variability without loops. Maximize individual reactions and test if loop exists using cfFBA. Then breaks the loop by setting the fluxes that goes to zero in loopless flux to zero.

Usage

```
cfFVA(model, rxnList, solver = SYBIL_SETTINGS("SOLVER"),pct_objective=100
  ,solverParm=NA
  ,verboseMode = 2,includeRxnEqn=TRUE
  ,boundFlg=FALSE
)
```

Arguments

model	An object of class modelorg.
rxnList	a character vector of reaction ID's to find their flux variability
solver	Single character value. The solver to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
pct_objective	percentage of maximum biomass to be achieved.
solverParm	extra parameters to solver, like tolerance.
includeRxnEqn	when true (default) the reaction equation is returned.
verboseMode	level of displaying messages.
boundFlg	should be set to FALSE to enumrateCycles.

Value

return two lists res and maxFlx. The first list contains reactions with the computed values. The second list contains the details of calculations. If a reaction is involved in a loop it will appear at least twice in the second list.

Author(s)

Abdelmoneim Amer Desouki

See Also

modelorg cfFBA

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
library(sybilcycleFreeFlux)
data(Ec_core)
model=Ec_core

fva=cfFVA(model,react_id(model))
write.csv(file="cfFVA1_cnd_res.csv",fva[[1]]);
write.csv(file="cfFVA1_cnd_det.csv",fva[[2]]);
#plot fluxes

## The function is currently defined as
"cfFVA"
```

enumerateCycles *function to enumerate cycles*

Description

Uses cycleFreeFlux to enumerate cycles in a given metabolic network.

Usage

```
enumerateCycles(model, rxnList, solver = SYBIL_SETTINGS("SOLVER"))
```

Arguments

model	An object of class modelorg.
rxnList	a character vector of reaction ID's to find their flux variability
solver	Single character value. The solver to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").

Value

return a list of unique loops found in the model.

Author(s)

Abdelmoneim Amer Desouki

See Also

modelorg cfFBA

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
## Not run:
data(Ec_core)
model=Ec_core

cycles=enumerateCycles(model,react_id(model))

## End(Not run)

## The function is currently defined as
"enumerateCycles"
```

getModel_WW *based on the algorithm given in Wright & Wagner 2008*

Description

get a model that contains only reactions in loops.

Usage

```
getModel_WW(model, solver = SYBIL_SETTINGS("SOLVER"))
```

Arguments

model	An object of class modelorg.
solver	Single character value. The solver to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").

Value

return an object of class modelorg containing subset of reactions of the original model that participate in at least one loop.

Author(s)

Abdelmoneim Amer Desouki

References

Wright, J. and Wagner, A. (2008). Exhaustive identification of steady state cycles in large stoichiometric networks. BMC systems biology, 2, 61.

See Also

modelorg cfFBA

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
## Not run:
 data(iAF1260)
 loopiAF=getModel_WW(iAF1260)

## End(Not run)
```

iAF1260 *Escherichia coli Metabolic Model iAF1260*

Description

The dataset is a genome scale metabolic network of the *E. coli*. It consists of 2077 internal reactions, 304 exchange reactions and a biomass objective function.

Usage

```
data(iAF1260)
```

Format

An object of class `modelorg`

References

Feist AM, Henry CS, Reed JL, Krummenacker M, Joyce AR, Karp PD, Broadbelt LJ, Hatzimanikatis V, Palsson BØ (2007) A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. Mol Syst Biol *3*: 121

iMM904 *Saccharomyces cerevisiae Metabolic Model*

Description

The dataset is a genome scale metabolic network of the *Saccharomyces cerevisiae*. It consists of 1412 internal reactions, 164 exchange reactions and a biomass objective function.

Usage

```
data(iMM904)
```

Format

An object of class `modelorg`

References

Mo ML, Palsson BO, Herrgard MJ: Connecting extracellular metabolomic measurements to intracellular flux states in yeast. BMC Syst Biol 2009,3:37.

llFBA *looplessFBA*

Description

implements loopless FBA as described in Schellenberger et al 2011 algorithm.

Usage

```
llFBA(model, lpdir = SYBIL_SETTINGS("OPT_DIRECTION"), solver = SYBIL_SETTINGS("SOLVER"),
  method = SYBIL_SETTINGS("METHOD"), solverParm = data.frame(CPX_PARAM_EPRHS = 1e-07),
  verboseMode = 2)
```

Arguments

model	An object of class modelorg.
lpdir	Character value, direction of optimisation. Can be set to "min" or "max". Default: SYBIL_SETTINGS("OPT_DIRECTION").
solver	Single character value. The solver to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
method	Single character value. The optimization algorithm to use. Possible values depend on the setting in solver. See SYBIL_SETTINGS for possible values. Default: LP_METHOD(SYBIL_SETTINGS).
solverParm	additional parameters to the solver
verboseMode	level of output messages

Value

status returned from the solver, objective value, and the fluxes returned from solver.

Author(s)

Abdelmoneim Amer Desouki

References

Schellenberger J, Lewis NE, Palsson BO (2011) Elimination of thermodynamically infeasible loops in steadystate metabolic models. Biophysical journal 100: 544 53

See Also

modelorg cfFBA

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
library(sybilcycleFreeFlux)
data(Ec_core)
model=Ec_core
llFBA(model,solver="glpkAPI",verbose=3)

## The function is currently defined as
"llFBA"
```

Index