Package 'sybilccFBA'

March 25, 2019

Type Package

Title Cost Constrained Flux Balance Analysis (ccFBA): MetabOlic Modeling with ENzyme kineTics (MOMENT)

Version 3.0.0

Date 2019-03-23

Depends R (>= 3.2.0), sybil, Matrix, methods

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

Description Different techniques of cost constrained flux balance analysis.
1- MetabOlic Modeling with ENzyme kineTics 'MOMENT' which uses enzyme kinetic data and enzyme molecular
weights to constrain flux balance analysis(FBA) and it is described in Adadi, R., Volkmer, B., Milo, R.,
Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network
with Kinetic Parameters, 8(7). <doi:10.1371/journal.pcbi.1002575>.
2- an improvement of 'MOMENT' that considers multi-functional enzymes 'ccFBA'.
Described in Desouki, Abdelmoneim. Algorithms for improving the predictive power of flux balance analysis. PhD dissertation, 2016.
FBA is a mathematical 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

Author Abdelmoneim Amer Desouki [aut, cre]

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

Repository CRAN

Date/Publication 2019-03-25 10:00:24 UTC

R topics documented:

`sybilccFBA-package` *Cost Constrained Flux Balance Analysis(ccFBA)*

Description

The package `sybilccFBA` implements some methods to get cost constrained fluxes. It is required to supply the molecular weights. It can be calculated from genome data using function `calc_MW`. Also requires kinetic data along with the model.

Details

Package:	sybilccFBA
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 cfba_moment

Examples

```
## Not run:
data(Ec_core)
model=Ec_core
genedef=read.csv(paste0(path.package("sybilccFBA"), '/extdata/Ec_core_genedef.csv'),
     stringsAsFactors=FALSE)
mw=data.frame(gene=genedef[,'gene'],mw=genedef[,'mw'],stringsAsFactors=FALSE)
mw[mw[,1]=='s0001','mw']=0.001#spontenious

kl=read.csv(stringsAsFactors=FALSE,paste0(path.package("sybilccFBA"),
            '/extdata/','allKcats_upd34_dd_h.csv'))
    kl=kl[!is.na(kl[,'ijo_id']),]
   kcat=data.frame(rxn_id=kl[,'ijo_id'],val=kl[,'kcat_max'],dirxn=kl[,'dirxn'],src=kl[,'src'],
               stringsAsFactors=FALSE)
    kcat=kcat[kcat[,'rxn_id']%in% react_id(model),]
    kcat[(is.na(kcat[,'src'])),'src']='Max'

mod2=mod2irrev(model)
uppbnd(mod2)[react_id(mod2)=="EX_o2(e)_b"]=1000
lowbnd(mod2)[react_id(mod2)=="ATPM"]=0
uppbnd(mod2)[react_id(mod2)=="ATPM"]=0


nr=react_num(mod2)
medianVal=median(kcat[,'val'])
# sum(is.na(genedef[,'mw']))

C=0.13#as not all genes exist
sim_name=paste0('Ec_core_med',round(medianVal,2),'_C',100*C)

cpx_stoich =read.csv(paste0(path.package("sybilccFBA"),
                '/extdata/','cpx_stoich_me.csv'),stringsAsFactors=FALSE)

CSList=c("R_EX_glc_e__b","R_EX_glyc_e__b","R_EX_ac_e__b","R_EX_fru_e__b",
                "R_EX_pyr_e__b","R_EX_gal_e__b",
    "R_EX_lac_L_e__b","R_EX_malt_e__b","R_EX_mal_L_e__b","R_EX_fum_e__b",
                "R_EX_xyl_D_e__b","R_EX_man_e__b","R_EX_tre_e__b",
    "R_EX_mnl_e__b","R_EX_g6p_e__b","R_EX_succ_e__b","R_EX_gam_e__b",
                "R_EX_sbt_D_e__b","R_EX_glcn_e__b",
    "R_EX_rib_D_e__b","R_EX_gsn_e__b","R_EX_ala_L_e__b",
    "R_EX_akg_e__b","R_EX_acgam_e__b")

msrd=c(0.66,0.47,0.29,0.54,0.41,0.24,0.41,0.52,0.55,0.47,0.51,0.35,0.48,0.61,
                        0.78,0.50,0.40,0.48,0.68,0.41,0.37,0.24,0.24,0.61)
```

```
CA=c(6,3,2,6,3,6,3,12,4,4,5,6,12,6,6,4,6,6,6,5,10,3,5,8)

CSList=substring(gsub('_e_','(e)',CSList),3)
 react_name(mod2)[react_id(mod2) %in% CSList]
 msrd=msrd[CSList %in% react_id(mod2) ]
 CA=CA[CSList %in% react_id(mod2)]
 CSList=CSList[CSList %in% react_id(mod2)]
 uppbnd(mod2)[react_id(mod2) %in% CSList]=0
 mod2R=mod2
##---------------------
# xt=FALSE
st=TRUE
selected_rxns=react_id(model)[gpr(model)!=""]

if(st){
    mrMat = getccFBA_mat(model,mod2,kcat,MW=mw,verbose=2,RHS=C,cpx_stoich=cpx_stoich,
                               medval=3600*medianVal,selected_rxns=selected_rxns)
}else{#nost
    mrMat = getccFBA_mat(model,mod2,kcat,MW=mw,verbose=2,RHS=C,cpx_stoich=NULL,
                          medval=3600*medianVal,selected_rxns=selected_rxns)
 }
mrMat_st = mrMat

r1i=getGpr1iso(model,MW=mw,cpx_stoich=cpx_stoich)
gpr1iso=r1i$gpr1iso;
mod_cpx_mw=r1i$mod_cpx_mw; rgst=r1i$rgst;
  sum(is.na(gpr1iso[,'stoich']))

nOR = sapply(gprRules(model),function(x) nchar(x)-nchar(gsub('|','',x,fixed=TRUE)))
nAND = sapply(gprRules(model),function(x) nchar(x)-nchar(gsub('&','',x,fixed=TRUE)))

  rgm=mrMat$rxnGeneCol
write.csv(file='rgm_aa.csv',cbind(rgm,nOR[rgm[,'revRxn']],gpr(model)[rgm[,'revRxn']]))

rgm=rgm[!is.na(rgm[,'gCol']) & rgm[,'stoich']!=0,]# only used genes
cnames=mrMat$cnames
gpr1iso=mrMat$gpr1iso
bm_rxn=which(obj_coef(mod2)!=0)
all_flx=NULL
all_flx_mr=NULL
all_flx_org=NULL

Kcatorg=kcat
solver='glpkAPI'
solverParm=NA

for(cs in 1:length(CSList)){
print(CSList[cs])
mod2=mod2R
uppbnd(mod2)[react_id(mod2) %in% CSList]=0
uppbnd(mod2)[react_id(mod2)==CSList[cs]]=1000

prob=sysBiolAlg(mod2,LHS=mrMat$LHS,rlb=mrMat$rlb,rub=mrMat$rub,rtype=mrMat$rtype,
```

```
      lb=mrMat$clb,ub=mrMat$cub,obj=mrMat$obj_cf,lpdir='max',cnames=mrMat$cnames,solver=solver,
                algorithm="ccFBA",solverParm=solverParm,
pname=sprintf("ccFBA mfe %s: %s,cpxst,kcatupd25",solver,mod_name(model)),
writeProbToFileName=sprintf('EC_ccFBA_Mat_%s.lp',solver));
sol=optimizeProb(prob);
str(sol)
print(getMeanStatus(sol$stat,solver))
    sol_mr=cfba_moment_mr(model,mod2,kcat,MW=mw,verbose=1,RHS=C,solver="glpkAPI",
             medval=3600*medianVal)
sol_org=cfba_moment(model,mod2,kcat,MW=mw,verbose=1,RHS=C,solver="glpkAPI",
             medval=3600*medianVal)
    ### preparing output -------------------
all_flx=rbind(all_flx,data.frame(stringsAsFactors=FALSE,cs,csname=CSList[cs],
                                 rxn_id=react_id(mod2),flx=sol$fluxes[1:nr]))
    all_flx_mr=rbind(all_flx_mr,data.frame(stringsAsFactors=FALSE,cs,csname=CSList[cs],
                         rxn_id=react_id(mod2),flx=sol_mr$sol$fluxes[1:nr]))
    all_flx_org=rbind(all_flx_org,data.frame(stringsAsFactors=FALSE,cs,
            csname=CSList[cs],rxn_id=react_id(mod2),flx=sol_org$sol$fluxes[1:nr]))
}

upt=all_flx[all_flx[,'csname']==all_flx[,'rxn_id'],]

#bm=all_flx[all_flx[,'obj']!=0,]
bm=all_flx[react_id(mod2)[obj_coef(mod2)!=0]==all_flx[,'rxn_id'],]
bm_mr=all_flx_mr[react_id(mod2)[obj_coef(mod2)!=0]==all_flx_mr[,'rxn_id'],]
bm_org=all_flx_org[react_id(mod2)[obj_coef(mod2)!=0]==all_flx_org[,'rxn_id'],]

cor.test(bm[,'flx'],msrd,method='spearman')
cor.test(bm_mr[,'flx'],msrd,method='spearman')
cor.test(bm_org[,'flx'],msrd,method='spearman')

bm=cbind(bm,msrd)

#Wong
cor(as.numeric(bm[,"flx"]),(as.numeric(bm[,"flx"])/as.numeric(upt[,"flx"]))^0.5 )

\dontrun{

 plot(msrd,bm[,'flx'],ylab="pred bm",xlim=c(0,0.8),ylim=c(0,max(bm_org[,'flx'])),
   pch=19,col='red',
                   main="Predicting growth rate using Ec_core")
abline(a=0,b=1,col="grey",lwd=2,lty=2)
points(msrd,bm_org[,'flx'],ylab="pred bm",xlim=c(0,0.8),ylim=c(0,max(bm_org[,'flx'])),
   pch=19,col='green')
points(msrd,bm_mr[,'flx'],ylab="pred bm",xlim=c(0,0.8),ylim=c(0,max(bm_org[,'flx'])),
   pch=19,col='darkblue')

legend(legend=c('MOMENT','MOMENT_mr','ccFBA st'),col=c('green','darkblue','red'),
               x=0,y=1,cex=0.75,pch=19)

################################top 5 genes####################
# selected_genes =c('b2323','b2472','b3774','b2779','b0431') # excl 'b0241', too many rxns 248
selected_genes =c('b1773','b0118','b1779') # excl 'b0241', too many rxns 248
```

```
sel_rxns =react_id(mod2)[(rowSums(rxnGeneMat(mod2)[,allGenes(mod2)%in% selected_genes])>0)]
sel_rxns=c(sel_rxns,CSList[cs],react_id(mod2)[obj_coef(mod2)!=0])


 write.csv(file=sprintf('iAF_top4g_flx_le%s.csv',sim_name),
                    all_flx[all_flx[,'rxn_id']%in% sel_rxns,])
}

## End(Not run)
```

---

addGlcTrns *add Glucose Transport constraint*

Description

add glucose transport constraint to the problem. Put an upperbound on glucose consumption.

Usage

```
addGlcTrns(prob, mod2)
```

Arguments

prob	lp problem
mod2	An object of class modelorg with only irreversible reactions. It can be sent to save time of recalculating it with each call.

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.

## The function is currently defined as
function (prob, mod2)
{
    Si = 0.2
    Hxt1 = 41 * Si/(Si + 107)
    Hxt2 = 16.1 * Si/(Si + 2.9)
    Hxt3 = 18.5 * Si/(Si + 29)
    Hxt4 = 12 * Si/(Si + 6.2)
```

```
  Hxt5 = 14 * Si/(Si + 10)
  Hxt6 = 11.4 * Si/(Si + 1.5)
  Hxt7 = 11.7 * Si/(Si + 1.3)
  Gal2 = 17.5 * Si/(Si + 1.5)
  colid = getNumCols(lp = problem(prob)) + 1
  trnsCol = NULL
  rowind = getNumRows(lp = problem(prob)) + 1
  glcRxn = which(react_id(mod2) == "R_GLCt1")
  addRowsToProb(lp = problem(prob), i = rowind, type = "U",
      lb = 0, ub = 0, cind = list(c(trnsCol[1, "Col"], trnsCol[2,
          "Col"], trnsCol[3, "Col"], trnsCol[4, "Col"], trnsCol[5,
          "Col"], trnsCol[6, "Col"], trnsCol[7, "Col"], glcRxn)),
      nzval = list(c(-Hxt1, -Hxt2, -Hxt3, -Hxt4, -Hxt5, -Hxt6,
          -Hxt7, 1)), rnames = "glcTrns")
  return(prob)
}
```

`calc_MW` *Calculate molecular weights*

Description

Calculate Molecular weights of different proteins using the genome .faa file.

Usage

```
calc_MW(aa_fname = "aa.txt", ptt_fname = "test2.ptt", faa_fname = "NC_000913.faa",
 nchrm = 1)
```

Arguments

`aa_fname`	file name of file containing lisT of amino acid names
`ptt_fname`	file name of file containing gene names with gene code
`faa_fname`	file name of file containing gene code and sequence of amino acids
`nchrm`	the number of chromosomes in the genome

Value

generate a file containing gene name , length, and molecular weight

Author(s)

Abdelmoneim Amer Desouki

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
## Not run:
aa_fname <- system.file("extdata", "aa.txt", package="sybilccFBA")
ptt_fname <- system.file("extdata", "test2.ptt", package="sybilccFBA")
faa_fname <- system.file("extdata", "NC_000913.faa", package="sybilccFBA")

geneCnt <- calc_MW(aa_fname,ptt_fname,faa_fname)
write.csv(file="geneCnt.csv",geneCnt)

## The function is currently defined as
"calc_MW"

## End(Not run)
```

`cfba_moment` *Function: cfba_moment: implement MOMENT method*

Description

This function uses GPR, kcat, and molecular weights to calculate fluxes according to MOMENT method.

Usage

```
cfba_moment(model,mod2=NULL, Kcat,MW=NULL,
selected_rxns=NULL,verboseMode=2,objVal=NULL,
RHS=NULL,solver=SYBIL_SETTINGS("SOLVER"),medval=NULL,
                  runFVA = FALSE, fvaRxn = NULL)
```

Arguments

`model`	An object of class `modelorg`.
`mod2`	An object of class `modelorg` with only irreversible reactions. It can be sent to save time of recalculating it with each call.
`Kcat`	kcat values in unit 1/S. Contains three slots: reaction id,direction(dirxn),value(val)
`MW`	list of molecular weights of all genes, using function calc_MW, in units g/mol
`selected_rxns`	optional parameter used to select a set of reactions not all, list of react_id
`verboseMode`	An integer value indicating the amount of output to stdout: 0: nothing, 1: status messages, 2: like 1 plus with more details, 3: generates files of the LP problem. Default: 2.
`RHS`	the budget C, for EColi 0.27
`objVal`	when not null the problem will be to find the minimum budget that give the specified objective value(biomass)

solver	Single character string giving the solver package to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
medval	median of Kcat values , used for missing values
runFVA	flag to choose to run flux variability default FALSE
fvaRxn	optional parameter to choose set of reaction ids to run FVA on them. Ids are from the irreversible model default all reactions. Ignored when runFVA is not set.

Details

Main steps 1- Add variables for all genes 2- for each selected reaction: parse gpr, 3- Add variables accordingly and constraints 4- Add solvant constraint

Value

returns a list containing slots:

sol	solution of the problem, instance of class optObj.
prob	object of class sysBiolAlg that contains the linear problem, this can be used for further processing like adding more constraints. To save it, function writeProb can be used.
geneCol	mapping of genes to variables in the problem.

Author(s)

Abdelmoneim Amer Desouki

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

Gelius-Dietrich, G., Desouki, A. A., Fritzemeier, C. J., & Lercher, M. J. (2013). sybil–Efficient constraint-based modelling in R. BMC systems biology, 7(1), 125.

See Also

modelorg, optimizeProb

Examples

```
## Not run:
library(sybilccFBA)
data(iAF1260)
model= iAF1260
  data(mw)
  data(kcat)
   mod2=mod2irrev(model)
```

```
uppbnd(mod2)[react_id(mod2)=="R_EX_glc_e__b"]=1000
uppbnd(mod2)[react_id(mod2)=="R_EX_glyc_e__b"]=0
uppbnd(mod2)[react_id(mod2)=="R_EX_ac_e__b"]=0
uppbnd(mod2)[react_id(mod2)=="R_EX_o2_e__b"]=1000
lowbnd(mod2)[react_id(mod2)=="R_ATPM"]=0

  sol=cfba_moment(model,mod2,kcat,MW=mw,verbose=3,RHS=0.27,solver="glpkAPI",medval=3600*22.6)
  bm_rxn = which(obj_coef(mod2)!=0)
  print(sprintf('biomass=%f',sol$sol$fluxes[bm_rxn]))
 # Enzyme concentrations:


## End(Not run)
data(Ec_core)
model=Ec_core
genedef=read.csv(paste0(path.package("sybilccFBA"), '/extdata/Ec_core_genedef.csv'),
     stringsAsFactors=FALSE)
mw=data.frame(gene=genedef[,'gene'],mw=genedef[,'mw'],stringsAsFactors=FALSE)
mw[mw[,1]=='s0001','mw']=0.001#spontenious
##########
##Kcats
kl=read.csv(stringsAsFactors=FALSE,paste0(path.package("sybilccFBA"),
'/extdata/','allKcats_upd34_dd_h.csv'))
    kl=kl[!is.na(kl[,'ijo_id']),]
    kcat=data.frame(rxn_id=kl[,'ijo_id'],val=kl[,'kcat_max'],dirxn=kl[,'dirxn'],
                src=kl[,'src'],stringsAsFactors=FALSE)
    kcat=kcat[kcat[,'rxn_id']%in% react_id(model),]
    kcat[(is.na(kcat[,'src'])),'src']='Max'
 ########## ----------------

mod2=mod2irrev(model)
uppbnd(mod2)[react_id(mod2)=="EX_o2(e)_b"]=1000
lowbnd(mod2)[react_id(mod2)=="ATPM"]=0
uppbnd(mod2)[react_id(mod2)=="ATPM"]=0

nr=react_num(mod2)
medianVal=median(kcat[,'val'])
# sum(is.na(genedef[,'mw']))

C_mr=0.13#as not all genes exist
sim_name=paste0('Ec_org_med',round(medianVal,2),'_C',100*C_mr)

cpx_stoich =read.csv(paste0(path.package("sybilccFBA"),
'/extdata/','cpx_stoich_me.csv'),stringsAsFactors=FALSE)


##-----------------
CSList=c("R_EX_glc_e__b","R_EX_glyc_e__b","R_EX_ac_e__b","R_EX_fru_e__b",
"R_EX_pyr_e__b","R_EX_gal_e__b",
"R_EX_lac_L_e__b","R_EX_malt_e__b","R_EX_mal_L_e__b","R_EX_fum_e__b","R_EX_xyl_D_e__b",
"R_EX_man_e__b","R_EX_tre_e__b",
"R_EX_mnl_e__b","R_EX_g6p_e__b","R_EX_succ_e__b","R_EX_gam_e__b","R_EX_sbt_D_e__b",
```

```
"R_EX_glcn_e__b",
"R_EX_rib_D_e__b","R_EX_gsn_e__b","R_EX_ala_L_e__b","R_EX_akg_e__b","R_EX_acgam_e__b")

msrd=c(0.66,0.47,0.29,0.54,0.41,0.24,0.41,0.52,0.55,0.47,0.51,0.35,0.48,0.61,
0.78,0.50,0.40,0.48,0.68,0.41,0.37,0.24,0.24,0.61)
CA=c(6,3,2,6,3,6,3,12,4,4,5,6,12,6,6,4,6,6,6,5,10,3,5,8)

CSList=substring(gsub('_e_','(e)',CSList),3)
 react_name(mod2)[react_id(mod2) %in% CSList]
 msrd=msrd[CSList %in% react_id(mod2) ]
 CA=CA[CSList %in% react_id(mod2)]
 CSList=CSList[CSList %in% react_id(mod2)]
 uppbnd(mod2)[react_id(mod2) %in% CSList]=0
 mod2R=mod2
##---------------------
bm_rxn=which(obj_coef(mod2)!=0)
all_flx=NULL
all_flx_MC=NULL
all_rg_MC=NULL
Kcatorg=kcat
solver='glpkAPI'
solverParm=NA

for(cs in 1:length(CSList)){
print(CSList[cs])
mod2=mod2R
uppbnd(mod2)[react_id(mod2) %in% CSList]=0
uppbnd(mod2)[react_id(mod2)==CSList[cs]]=1000
sol_org=cfba_moment(model,mod2,kcat,MW=mw,verbose=3,RHS=0.27,solver="glpkAPI",
    medval=3600*medianVal)

### preparing output --------------------
all_flx=rbind(all_flx,data.frame(stringsAsFactors=FALSE,cs,csname=CSList[cs],
            rxn_id=react_id(mod2),
            flx=sol_org$sol$fluxes[1:nr], ub=uppbnd(mod2),ubR=uppbnd(mod2R)))
        # print(paste0("nrow all_rg_MC=",nrow(all_rg_MC)))
}

upt=all_flx[all_flx[,'csname']==all_flx[,'rxn_id'],]
bm=all_flx[react_id(mod2)[obj_coef(mod2)!=0]==all_flx[,'rxn_id'],]

cor.test(bm[,'flx'],msrd,method='spearman')
```

---

cfba_moment_mr *Function: cfba_moment_mr: implement MOMENT method taking into account multifunctional enzymes*

Description

This function uses GPR, kcat, and molecular weights to calculate fluxes according to MOMENT method taking into account multifunctional enzymes. Whenever a protein i was involved in more than one reaction, we introduced auxiliary concentration variables xi,j for each of these reactions. These xi,j replaced the global concentration variable gi for the protein in the corresponding equation that limits the flux through this reaction based on the enzyme concentration. The sum of the xi,j is then equal to the total concentration of protein gi included in the global enzyme solvent capacity constraint.

Usage

```
cfba_moment_mr(model,mod2=NULL, Kcat,MW=NULL,
selected_rxns=NULL,verboseMode=2,objVal=NULL,
RHS=NULL,solver=SYBIL_SETTINGS("SOLVER"),C_mu_coef = 0,medval=NULL,
                 runFVA = FALSE, fvaRxn = NULL)
```

Arguments

model	An object of class modelorg.
mod2	An object of class modelorg with only irreversible reactions. It can be sent to save time of recalculating it with each call.
Kcat	kcat values in unit 1/S. Contains three slots: reaction id,direction(dirxn),value(val)
MW	list of molecular weights of all genes, using function calc_MW, in units g/mol
selected_rxns	optional parameter used to select a set of reactions not all, list of react_id
verboseMode	An integer value indicating the amount of output to stdout: 0: nothing, 1: status messages, 2: like 1 plus with more details, 3: generates files of the LP problem. Default: 2.
RHS	the budget C, for EColi 0.27
objVal	when not null the problem will be to find the minimum budget that give the specified objective value(biomass)
solver	Single character string giving the solver package to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
C_mu_coef	used to have C as a linear function of mu (biomass) : C = RHS + C_mu_coef*Biomass
medval	median of Kcat values , used for missing values
runFVA	flag to choose to run flux variability default FALSE
fvaRxn	optional parameter to choose set of reaction ids to run FVA on them. Ids are from the irreversible model default all reactions. Ignored when runFVA is not set.

Details

Main steps 1- Add variables for all genes 2- for each selected reaction: parse gpr, 3- Add variables accordingly and constraints 4- Add solvant constraint

Value

returns a list containing slots:

sol	solution of the problem.
prob	object of class sysBiolAlg that contains the linear problem, this can be used for further processing like adding more constraints. To save it, function writeProb can be used.
geneCol	mapping of genes to variables in the problem.
geneConc	the concentration of each gene, when the gene is catalyzing more than one reaction there will be a row with 'rxn' column set to NA containing the total.
rxnMC	for each reaction (GPR) the molecular crowding of it (total sum to budget)
rxnGeneMC	the contribution of each gene to all of its reactions.

Author(s)

Abdelmoneim Amer Desouki

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

Gelius-Dietrich, G., Desouki, A. A., Fritzemeier, C. J., & Lercher, M. J. (2013). sybil–Efficient constraint-based modelling in R. BMC systems biology, 7(1), 125.

See Also

modelorg, optimizeProb

Examples

```
## Not run:
library(sybilccFBA)
data(iAF1260)
model= iAF1260
  data(mw)
  data(kcat)
   mod2=mod2irrev(model)

uppbnd(mod2)[react_id(mod2)=="R_EX_glc_e__b"]=1000
uppbnd(mod2)[react_id(mod2)=="R_EX_glyc_e__b"]=0
uppbnd(mod2)[react_id(mod2)=="R_EX_ac_e__b"]=0
uppbnd(mod2)[react_id(mod2)=="R_EX_o2_e__b"]=1000
lowbnd(mod2)[react_id(mod2)=="R_ATPM"]=0

  sol_mr=cfba_moment_mr(model,mod2,kcat,MW=mw,verbose=3,RHS=0.27,solver="glpkAPI",medval=3600*22.6)
   bm_rxn = which(obj_coef(mod2)!=0)
   print(sprintf('biomass=%f',sol_mr$sol$fluxes[bm_rxn]))
  # Enzyme concentrations:
```

```
gconc=sol_mr$geneConc

## End(Not run)
```

---

cfba_moment_pw *Function: cfba_moment_pw: implement MOMENT method*

Description

This function uses GPR, kcat, and molecular weights to calculate fluxes according to MOMENT method. MOMENT pairwise OR like MATLAB implementation

Usage

```
cfba_moment_pw(model,mod2=NULL, Kcat,MW=NULL,
selected_rxns=NULL,verboseMode=2,objVal=NULL,
RHS=NULL,solver=SYBIL_SETTINGS("SOLVER"),medval=NULL)
```

Arguments

model	An object of class modelorg.
mod2	An object of class modelorg with only irreversible reactions. It can be sent to save time of recalculating it with each call.
Kcat	kcat values in unit 1/S. Contains three slots: reaction id,direction(dirxn),value(val)
MW	list of molecular weights of all genes, using function calc_MW, in units g/mol
selected_rxns	optional parameter used to select a set of reactions not all, list of react_id
verboseMode	An integer value indicating the amount of output to stdout: 0: nothing, 1: status messages, 2: like 1 plus with more details, 3: generates files of the LP problem. Default: 2.
RHS	the budget C, for EColi 0.27
objVal	when not null the problem will be to find the minimum budget that give the specified obective value(biomass)
solver	Single character string giving the solver package to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
medval	median of Kcat values , used for missing values

Details

Main steps 1- Add variables for all genes 2- for each selected reaction: parse gpr, 3- Add variables accordingly and constraints 4-Add solvant constraint

Value

returns a list containing slots: prob:problem object that contains data and model sol: solution of the problem. geneCol: mapping of genes to variables in the problem.

Author(s)

Abdelmoneim Amer Desouki

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

See Also

modelorg, optimizeProb

Examples

```
## Not run:
library(sybilccFBA)
data(iAF1260)
model= iAF1260
  data(mw)
  data(kcat)
   mod2=mod2irrev(model)

uppbnd(mod2)[react_id(mod2)=="R_EX_glc_e__b"]=1000
uppbnd(mod2)[react_id(mod2)=="R_EX_glyc_e__b"]=0
uppbnd(mod2)[react_id(mod2)=="R_EX_ac_e__b"]=0
uppbnd(mod2)[react_id(mod2)=="R_EX_o2_e__b"]=1000
lowbnd(mod2)[react_id(mod2)=="R_ATPM"]=0

  sol=cfba_moment(model,mod2,kcat,MW=mw,verbose=3,RHS=0.27,solver="glpkAPI",medval=3600*22.6)


## End(Not run)
```

---

getccFBA_mat *getccFBA_mat: get ccFBA model as a constraint matrix*

Description

given Kcat vector (in 1/Sec), and molecular weights and optionally protein complex stoichiometry formulate a ccFBA problem(Maximize biomass). Improvements over MOMENT: 1- use multi-functioning enzyme constraint. 2-use one enzyme (the cheapest) for each reaction catalayzed my multiple isoenzymes and thus eliminate under estimation of rules of form ((A AND B) OR (A AND C)) 3-optionally include protein complex stoichiometry

Usage

```
getccFBA_mat(model, mod2 = NULL, Kcat, MW = NULL, selected_rxns = NULL, verboseMode = 2,
objVal = NULL, RHS = NULL, solver = SYBIL_SETTINGS("SOLVER"), medval = NULL,
cpx_stoich = NULL, C_mu_coef = 0)
```

Arguments

model	An object of class modelorg.
mod2	An object of class modelorg with only irreversible reactions. It can be sent to save time of recalculating it with each call.
Kcat	kcat values in unit 1/S. Contains three slots: reaction id,direction(dirxn),value(val)
MW	list of molecular weights of all genes, using function calc_MW, in units g/mol
selected_rxns	optional parameter used to select a set of reactions not all, list of react_id
verboseMode	An integer value indicating the amount of output to stdout: 0: nothing, 1: status messages, 2: like 1 plus with more details, 3: generates files of the LP problem. Default: 2.
RHS	the budget C, for EColi 0.27
objVal	when not null the problem will be to find the minimum budget that give the specified objective value(biomass)
solver	Single character string giving the solver package to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").
medval	median of Kcat values , used for missing values
cpx_stoich	giving the stoichiometry of complexes: data frame containing at least two columns 'genes','stoich' 'gene' : delimited string of genes(ordered by geneid(e.g bnumber)), 'stoich': number of subunits of each gene in the same order in 'genes' Default: NULL (e.g. all stoichiometry coefficients equal one)
C_mu_coef	used to have C as a linear function of mu (biomass) : C = RHS + C_mu_coef*Biomass default 0 (e.g. ignored).

Details

variable names: g1_x_1: indicates that gene x is in one-to-one relation gr_x_r:indicates that gene x catalyzes more than one reaction and this variable is the portion that catalyzes reaction r. gmr_x_cnt: variable for a gene catalyzing more than one reaction (i.e cnt reactions) and this variable is the sum of individual parts. cpx_r: used to represent a complex catalyzing reaction r.

Value

return a LIST,

LHS	constraint matrix, consists of S matrix and ccFBA constraints
rlb,rub	bounds for constraints (i.e rows)
clb,cub	bounds for variables (i.e columns)
rxnkcatRow	mapping of Kcat values to constraints

rgst	complex stoichiometry as returned by getGpr1iso
rxnGeneCol	rxn_id,gene,Col : used to get MCrowding
geneCol	low level structure, used for debugging
mod_objc_ind	row index of constraint on FBA model objective
cc_ind	capacity constraint index

Author(s)

Abdelmoneim Amer Desouki

References

Desouki, Abdelmoneim. "Algorithms for improving the predictive power of flux balance analysis." PhD diss., 2016.

See Also

getGpr1iso, cfba_moment_mr,simulate_EColi

---

getGpr1iso *Choose the the smallest isoenzyme*

Description

preprocessing of model to get the minimal cost iso enzyme and eliminate the OR from GPR's.

Usage

```
getGpr1iso(model, cpx_stoich, MW)
```

Arguments

model	object of class modelorg representing the metabolic network.
cpx_stoich	giving the stoichiometry of complexes: data frame containing at least two columns 'genes','stoich' 1/10/2015 'gene' : delimited string of genes(ordered by geneid(e.g bnumber)), 'stoich': number of subunits of each gene in the same order in 'genes'
MW	MW measurement for gene using readfaa.r, calc_MW or gene annotation [units g/mmol]

Value

return a LIST:

gpr1iso	data frame containing the choosen term (complex or isoenzyme)
mod_cpx_mw	molecular weight of isoenzyme
rgst	matrix of reaction gene with stoichiometry

Author(s)

Abdelmoneim Amer Desouki

See Also

getccFBA_mat

---

getRevFlux *get fluxes of reversible model*

Description

get fluxes of reversible model from fluxes of irreversible model

Usage

```
getRevFlux(model, irrxns, fdirrev)
```

Arguments

model	metabolic network of type modelorg
irrxns	list of ids of reactions in the irreversible model
fdirrev	flux distribution of irreversible reaction

Value

reaction id in the model and the forward and backward flux, the net flux equals fwd-bwd.

Author(s)

Abdelmoneim Amer Desouki

---

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

---

iJO1366 *Escherichia coli Metabolic Model iJO1366*

Description

The dataset is a genome scale metabolic network of the *E. coli*. It consists of 2253 internal reactions, 330 exchange reactions and a biomass objective function.

Usage

```
data(iJO1366)
```

Format

An object of class `modelorg`

References

Orth, J. D., Conrad, T. M., Na, J., Lerman, J. A., Nam, H., Feist, A. M., & Palsson, B. O. (2011). A comprehensive genome-scale reconstruction of Escherichia coli metabolism—2011. *Molecular systems biology*, 7(1).

---

iML1515 *Escherichia coli Metabolic Model iML1515*

Description

The dataset is a genome scale metabolic network of the *E. coli*. It consists of 2381 internal reactions, 331 exchange reactions and a biomass objective function. It contains 1515 genes.

Usage

```
data(iML1515)
```

Format

An object of class `modelorg`

References

Monk, Jonathan M., et al. "iML1515, a knowledgebase that computes Escherichia coli traits." Nature biotechnology 35.10 (2017): 904.

`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.

`initialize-methods` *Initialize Problem Object*

Description

Initialize ccFBA Problem.

Usage

```
## S4 method for signature 'sysBiolAlg_ccFBA'
initialize(.Object, model, LHS, rlb, rub, rtype, lb, ub, obj,
                lpdir, cnames = NULL, rnames = NULL, pname = NULL,
                scaling = NULL, writeProbToFileName = NULL, ...)
```

Arguments

.Object	Problem object
model	An object of class modelorg.
lpdir	Single character string containing the direction of optimization. Can be set to "min" or "max". Default: "max".
LHS	constraint matrix, consists of S matrix and ccFBA constraints
rlb,rub	bounds for constraints (i.e rows)
lb,ub	bounds for variables (i.e columns)
rtype	row constraint type
obj	index of objective
cnames	A character vector giving the variable names. If set to NULL, the reaction id's of model are used. Default: NULL.
rnames	A character vector giving the constraint names. If set to NULL, the metabolite id's of model are used. Default: NULL.
pname	A single character string containing a name for the problem object. Default: NULL.
scaling	Scaling options used to scale the constraint matrix. If set to NULL, no scaling will be performed (see scaleProb). Default: NULL.
writeProbToFileName	A single character string containing a file name to which the problem object will be written in LP file format. Default: NULL.
...	Further arguments passed to the initialize method of sysBiolAlg. They are solver, method and solverParm.

Author(s)

Abdelmoneim Amer Desouki

See Also

Constructor function sysBiolAlg and superclass sysBiolAlg.

kcat *Escherichia coli KCAT values used in MOMENT method*

Description

The dataset is a list of kcat values used in method MOMENT. Values are in unit 1/S.

Usage

```
data(kcat)
```

Format

A data frame with three columns

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

mw *Escherichia coli molecular weight values used in MOMENT method*

Description

The dataset is a list of molecular weights values used in method MOMENT. Values are in unit g/mol.

Usage

```
data(mw)
```

Format

A data frame with two columns

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

mw_iML1515 *SEColi metabolic genes molecular weight values used in MOMENT/ccFBA method*

Description

The dataset is a list of molecular weights values used in method in application of MOMENT method to EColi. Values are in unit g/mol. Calculated from the EColi genome sequence available at NCBI using `calc_MW`.

Usage

```
data(mw_iML1515)
```

Format

A data frame with two columns

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

readmodel *read MOMENT model*

Description

create lp from lists generated from MATLAB MOMENT model.

Usage

```
readmodel(mat, mets, rxns, rbnds, cbnds, solver = "glpkAPI")
```

Arguments

`mat`	contain the constraints matrix
`mets`	list of metabolites
`rxns`	list of reactions and their bounds
`rbnds`	bounds of rows of constraint matrix
`cbnds`	bounds of columns of constraint matrix
`solver`	solver used to solve the lp, can be glpkAPI or cplexAPI

Value

return fluxes obtained using the lp.

Author(s)

Abdelmoneim Amer Desouki

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

See Also

cfba_moment

`simulate_EColi` *Simulate Ecoli*

Description

calls different methods to simulate metabolism in EColi model.

Usage

```
simulate_EColi(model, mod2, kcat, mw, budget_C, CSList, atpmz = TRUE, trns_rxns = NULL,
                cpx_stoich = NULL, solver = "glpkAPI")
```

Arguments

`model`	An object of class modelorg.
`mod2`	An object of class modelorg with only irreversible reactions. It can be sent to save time of recalculating it with each call.
`kcat`	kcat values in unit 1/S. Contains three slots: reaction id,direction(dirxn),value(val)
`mw`	list of molecular weights of all genes, using function calc_MW, in units g/mol
`budget_C`	the budget C, for EColi 0.27
`CSList`	the list of carbon sources to be simulated, given as reaction id's
`atpmz`	set ATPM reaction to zero, default TRUE
`trns_rxns`	trnsport reactions to be excluded from budget (may contain other reactions if necessary)
`cpx_stoich`	giving the stoichiometry of complexes: data frame containing at least two columns 'genes','stoich' 'gene' : delimited string of genes(ordered by geneid(e.g bnumber)), 'stoich': number of subunits of each gene in the same order in 'genes' Default: NULL (e.g. all stoichiometry coefficients equal one)

solver	Single character string giving the solver package to use. See SYBIL_SETTINGS for possible values. Default: SYBIL_SETTINGS("SOLVER").

Details

This function is to show examples of simulation

Value

return a LIST,

gr_data	the objective value (growth rate) of all carbon sources
uptake	the uptake of the different conditions
all_flx	data frame containing carbon source, reaction id, and flux in column flx
rg_MC	reaction gene molecular crowding(MC), data frame containing carbon source, reaction id, and flux in column flx, geneConc, rxnMC: MC for reaction, rgMC: MC for gene/complex, the result is sorted in decreasing order of MC
all_flx_MC	reaction molecular crowding(MC), data frame containing carbon source, reaction id, and flux in column flx, geneConc, rxnMC: MC for reaction, rgMC: MC for gene/complex, result not sorted.

Author(s)

Abdelmoneim Amer Desouki

References

Desouki, Abdelmoneim. "Algorithms for improving the predictive power of flux balance analysis." PhD diss., 2016.

See Also

getccFBA_mat,cfba_moment_mr

Examples

```
## Not run:
   data(iML1515)
   #1-get metabolic model
   model=iML1515
   #2-get Molecular weights
   print(load(paste0(path.package("sybilccFBA"), '/extdata/mw_iML1515.RData')))

   mw=mw_iML1515
   mw=rbind(mw,data.frame(Synonym="s0001", mw=0.001))
   colnames(mw)[1]='gene'
   #3-get kcat list
   kl=read.csv(stringsAsFactors=FALSE,paste0(path.package("sybilccFBA"),
       '/extdata/','allKcats_upd34_dd_h.csv'))
```

```
    kl=kl[!is.na(kl[,'ijo_id']),]
   kcat=data.frame(rxn_id=kl[,'ijo_id'],val=kl[,'kcat_max'],dirxn=kl[,'dirxn'],src=kl[,'src'],
    stringsAsFactors=FALSE)
    kcat=kcat[kcat[,'rxn_id']%in% react_id(model),]
    kcat[(is.na(kcat[,'src'])),'src']='Max'
  #4-get complex stoichiometry if used
  cpx_stoich =read.csv(paste0(path.package("sybilccFBA"),
        '/extdata/','cpx_stoich_me.csv'),stringsAsFactors=FALSE)
  #5-identify Carbon sources to be tested
  csl=c("EX_glc__D(e)_b","EX_glyc(e)_b", "EX_ac(e)_b" ,  "EX_fru(e)_b",
  "EX_pyr(e)_b",  "EX_gal(e)_b" ,
   "EX_lac__L(e)_b",  "EX_malt(e)_b" ,"EX_mal__L(e)_b", "EX_fum(e)_b" ,
   "EX_xyl__D(e)_b","EX_man(e)_b", "EX_tre(e)_b",
   "EX_mnl(e)_b",   "EX_g6p(e)_b",   "EX_succ(e)_b", "EX_gam(e)_b",
   "EX_sbt__D(e)_b", "EX_glcn(e)_b",
   "EX_rib__D(e)_b","EX_gsn(e)_b" ,"EX_ala__L(e)_b",  "EX_akg(e)_b" ,
   "EX_acgam(e)_b")

  msrd=c(0.66,0.47,0.29,0.54,0.41,0.24,0.41,0.52,0.55,0.47,0.51,0.35,0.48,0.61,
      0.78,0.50,0.40,0.48,0.68,0.41,0.37,0.24,0.24,0.61)
  CA=c(6,3,2,6,3,6,3,12,4,4,5,6,12,6,6,4,6,6,6,5,10,3,5,8)

  #6-get irreversible model
  sum(react_id(model) %in% gsub('_b$','',csl))
  model1=model
  react_rev(model1)[react_id(model) %in% gsub('_b$','',csl)]=TRUE
  mod2=mod2irrev(model1)

   react_name(mod2)[react_id(mod2) %in% csl]

   uppbnd(mod2)[react_id(mod2) %in% csl]=0
   uppbnd(mod2)[react_id(mod2) %in% gsub('_b$','_f',csl)]=0

  uppbnd(mod2)[react_id(mod2)=="EX_o2(e)_b"]=1000
  trns_rxns=grepl("tex$",react_id(model))

  ##Call function
 tmp_res=simulate_EColi(model,mod2,mw=mw,budget_C=0.27,kcat=kcat,cpx_stoich=cpx_stoich,
  atpmz=FALSE,trns_rxns=trns_rxns,CSList=csl)
  bm=tmp_res[[1]]
  cor.test(bm[,'flx'],msrd,method='spearman')
   plot(msrd,bm[,'flx'],ylab="Predicted Growth Rate",xlim=c(0,0.8),
               ylim=c(0,max(bm[,'flx'])),xlab='Measured Growth Rate',
     main=sprintf("Effect of Keff, Corr=%.2f,nkcat=%d",cor(bm[,'flx'],msrd),nrow(kcat)))
  abline(a=0,b=1,col="red",lwd=2,lty=2)


## End(Not run)
```

`sysBiolAlg_ccFBA-class`
Class `"sysBiolAlg_ccFBA"`

Description

The class `sysBiolAlg_ccFBA` holds an object of class `optObj` which is generated to meet the requirements of the ccFBA algorithm.

Details

The `initialize` method has the following arguments:

model An object of class `modelorg`.

lpdir Single character string containing the direction of optimization. Can be set to `"min"` or `"max"`.
Default: `"max"`.

LHS constraint matrix, consists of S matrix and ccFBA constraints

rlb,rub bounds for constraints (i.e rows)

lb,ub bounds for variables (i.e columns)

rtype row constraint type

obj index of objective

cnames A character vector giving the variable names. If set to `NULL`, the reaction id's of `model` are used.
Default: `NULL`.

rnames A character vector giving the constraint names. If set to `NULL`, the metabolite id's of `model` are used.
Default: `NULL`.

pname A single character string containing a name for the problem object.
Default: `NULL`.

scaling Scaling options used to scale the constraint matrix. If set to `NULL`, no scaling will be performed (see `scaleProb`).
Default: `NULL`.

writeProbToFileName A single character string containing a file name to which the problem object will be written in LP file format.
Default: `NULL`.

... Further arguments passed to the initialize method of `sysBiolAlg`. They are `solver`, `method` and `solverParm`.

The problem object is built to be capable to perform cost constraint flux balance analysis (FBA) with a given model, which is basically the solution of a linear programming problem

$$\begin{aligned} \max \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{v} \\ \text{s.t.} \quad & \boldsymbol{S}\boldsymbol{v} = 0 \\ & \alpha_i \leq v_i \leq \beta_i \qquad \forall i \in \{1, \ldots, n\} \end{aligned}$$

with $\boldsymbol{S}$ being the stoichiometric matrix, α_i and β_i being the lower and upper bounds for flux (variable) i respectively. The total number of variables of the optimization problem is denoted by n. The solution of the optimization is a flux distribution maximizing the objective function $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{v}$ under the a given environment and the assumption of steady state. The optimization can be executed by using `optimizeProb`.

Objects from the Class

Objects can be created by calls of the form

`sysBiolAlg(model, algorithm = "ccFBA", ...)`.

Arguments to `...` which are passed to method `initialize` of class `sysBiolAlg_ccFBA` are described in the Details section.

Slots

`problem`: Object of class `"optObj"` containing the problem object.

`algorithm`: Object of class `"character"` containing the name of the algorithm.

`nr`: Object of class `"integer"` containing the number of rows of the problem object.

`nc`: Object of class `"integer"` containing the number of columns of the problem object

`fldind`: Object of class `"integer"` pointers to columns (variables) representing a flux (reaction) in the original network. The variable `fldind[i]` in the problem object represents reaction `i` in the original network.

`alg_par`: Object of class `"list"` containing a named list containing algorithm specific parameters.

Extends

Class `"sysBiolAlg"`, directly.

Methods

No methods defined with class "sysBiolAlg_ccFBA" in the signature.

Author(s)

Abdelmoneim Amer Desouki

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

Gelius-Dietrich, G., Desouki, A. A., Fritzemeier, C. J., & Lercher, M. J. (2013). sybil–Efficient constraint-based modelling in R. BMC systems biology, 7(1), 125.

See Also

Constructor function `sysBiolAlg` and superclass `sysBiolAlg`.

yst_kcat *Saccharomyces cerevisiae KCAT values used in MOMENT method*

Description

The dataset is a list of yst_kcat values used in method MOMENT. Values are in unit 1/S.

Usage

```
data(yst_kcat)
```

Format

A data frame with three columns

References

Richter C. Kosten und Effizienz von Enzymen als zusaetzliche Bedingung fuer die Flussverteilung in metabolischen Netzwerken [Diploma Thesis]. Berlin: Humboldt University; 2011.

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

yst_mw *Saccharomyces cerevisiae molecular weight values used in MOMENT method*

Description

The dataset is a list of molecular weights values used in method in application of MOMENT method to Saccharomyces cerevisiae. Values are in unit g/mol. Calculated from the yeast genome sequence available at NCBI (Saccharomyces cerevisiae S288c) using `calc_MW`.

Usage

```
data(yst_mw)
```

Format

A data frame with two columns

References

Adadi, R., Volkmer, B., Milo, R., Heinemann, M., & Shlomi, T. (2012). Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters, 8(7). doi:10.1371/journal.pcbi.1002575

Index