Package ‘BacArena’

May 20, 2020

Title  Modeling Framework for Cellular Communities in their Environments

Version  1.8.2

Author  Eugen Bauer [aut], Johannes Zimmermann [aut, cre]

Maintainer  Johannes Zimmermann <j.zimmermann@iem.uni-kiel.de>

Description  Can be used for simulation of organisms living in communities (Bauer and Zimmermann (2017) <doi:10.1371/journal.pcbi.1005544>). Each organism is represented individually and genome scale metabolic models determine the uptake and release of compounds. Biological processes such as movement, diffusion, chemotaxis and kinetics are available along with data analysis techniques.

URL  https://BacArena.github.io/

BugReports  https://github.com/euba/BacArena/issues

Depends  R (>= 3.5.0), sybil (>= 2.1.3), ReacTran (>= 1.4.2), deSolve (>= 1.12), Matrix (>= 1.2)

Imports  methods, utils, stats, graphics, ggplot2, reshape2, glpkAPI, plyr, Rcpp, igraph, stringr, R.matlab

Suggests  parallel, knitr, rmarkdown

LinkingTo  Rcpp, RcppArmadillo, RcppEigen

License  GPL-3 file LICENSE

VignetteBuilder  knitr

RoxygenNote  7.0.2

LazyData  true

NeedsCompilation  yes

Repository  CRAN

Date/Publication  2020-05-20 15:40:12 UTC
R topics documented:

addDefaultMed ........................................... 4
addEssentialMed ........................................... 4
addEval ....................................................... 5
addOrg ......................................................... 6
addSubs ......................................................... 7
Arena-class .................................................. 9
Arena-constructor .......................................... 10
Bac-class ...................................................... 11
Bac-Constructor ........................................... 11
BacArena ...................................................... 12
cellgrowth ..................................................... 12
changeDiff ..................................................... 13
changeFobj ..................................................... 14
changeOrg ..................................................... 15
changeSub ..................................................... 16
checkCorr ..................................................... 17
checkPhen ..................................................... 18
checkPhen_par .............................................. 19
chemotaxis .................................................... 19
colpal1 ......................................................... 20
colpal2 ......................................................... 21
colpal3 ......................................................... 21
colpal4 ......................................................... 22
colpal5 ......................................................... 22
colpal6 ......................................................... 23
constrain ..................................................... 23
consume ....................................................... 24
createGradient .............................................. 25
dat2mat ....................................................... 26
diffuse ........................................................ 27
diffusePDE ..................................................... 28
diffuseR ....................................................... 28
diffuse_par ..................................................... 29
emptyHood .................................................... 30
Eval-class .................................................... 31
Eval-constructor ........................................... 31
evalArena ..................................................... 32
extractMed .................................................... 33
findFeeding ................................................... 34
findFeeding2 .................................................. 35
findFeeding3 .................................................. 36
findFeeding3rep ............................................ 37
findInArena .................................................. 37
findrBiomass ................................................ 38
findRxnFlux ................................................... 39
flushSubs .................................................... 40
R topics documented:

- fluxVarSim
- getArena
- getCorrM
- getPhenoMat
- getPhenotype
- getSubHist
- getSublb
- getVarSubs
- growExp
- growLin
- growth
- growth_par
- Human-class
- Human-constructor
- lsd
- lysis
- minePheno
- move
- NemptyHood
- openArena
- optimizeLP
- Organism-class
- Organism-constructor
- plotAbundance
- plotCurves
- plotCurves2
- plotFluxVar
- plotFVA
- plotGrowthCurve
- plotInterNum
- plotPhenCurve
- plotPhenNum
- plotReaActivity
- plotShadowCost
- plotSpecActivity
- plotSubCurve
- plotSubDist
- plotSubDist2
- plotSubUsage
- plotSubVar
- plotTotFlux
- readMATmod
- redEval
- reset_screen
- rmSubs
- selPheno
- setKinetics
- sihumi_test
addDefaultMed

Add default medium of an organism to arena.

Description

The generic function addDefaultMed uses the lower bounds defined in an organism’s model file to compose minimal medium.

Usage

addDefaultMed(object, org, unit = "mM")

## S4 method for signature 'Arena'
addDefaultMed(object, org, unit = "mM")

Arguments

object An object of class Arena.
org An object of class Organism
unit A character used as chemical unit to set the amount of the substances to be added (valid values are: mmol/cell, mmol/cm2, mmol/arena, mM)

addEssentialMed

Add minimal medium of an organism to arena.

Description

The generic function addEssentialMed uses flux variability analysis to determine a essential growth medium components.
addEval

Usage

addEssentialMed(object, org, only_return = FALSE, limit = 10)

## S4 method for signature 'Arena'
addEssentialMed(object, org, only_return = FALSE, limit = 10)

Arguments

object An object of class Arena.
org An object of class Organism
only_return Set true if essential metabolites should only be returned but not added to arena
limit A metabolite is considered as essential if its remove whoeld decrease biomass growth below limit (between 0,100; default 10%)

addEval

Function for adding a simulation step

Description

The generic function addEval adds results of a simulation step to an Eval object.

Usage

addEval(object, arena, replace = F)

## S4 method for signature 'Eval'
addEval(object, arena, replace = F)

Arguments

object An object of class Eval.
arena An object of class Arena.
replace A boolean variable indicating if the last simulation step should be replaced by the new simulation step arena.

Details

The function addEval can be used in iterations to manipulate an Arena object and store the results in an Eval object.

See Also

Eval-class and Arena-class
Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
          minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
eval <- simEnv(arena, 5)
addEval(eval, arena)

addOrg

Add individuals to the environment

Description

The generic function `addOrg` adds individuals to the environment.

Usage

```
addOrg(
  object,
  specI,
  amount = 1,
  x = NULL,
  y = NULL,
  posmat = NULL,
  biomass = NA,
  n0 = NULL,
  n = NULL,
  m0 = NULL,
  m = NULL
)
```

## S4 method for signature 'Arena'
addOrg(
  object,
  specI,
  amount = 1,
  x = NULL,
  y = NULL,
  posmat = NULL,
  biomass = NA,
  n0 = NULL,
  n = NULL,
  m0 = NULL,
  m = NULL
)
Arguments

- **object**: An object of class Arena.
- **specI**: An object of class Organism.
- **amount**: A numeric number giving the number of individuals to add.
- **x**: A numeric vector giving the x positions of individuals on the grid.
- **y**: A numeric vector giving the y positions of individuals on the grid.
- **posmat**: A matrix with coordinates can be specified as an alternative to parameter x, y.
- **biomass**: A numeric vector giving the starting biomass of the individuals. (unit: fg)
- **n0**: Start column of matrix to take free positions from (default 1)
- **n**: End column of matrix to take free positions from (default arena@m)
- **m0**: Start row of matrix to take free positions from (default 1)
- **m**: End row of matrix to take free positions from (default arena@n)

Details

The arguments x and y should be in the same length as the number of organisms added (given by the argument amount).

See Also

- Arena-class and Bac-class

Examples

```r
data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms

# Alternative way: adding organisms by giving matrix with positions
arena <- Arena(n=20, m=20)
mat <- matrix(sample(c(0, 1), 400, replace = TRUE), nrow = 20, ncol = 20)
bac <- Bac(Ec_core)
arena <- addOrg(arena, specI=bac, posmat = mat)
```

Description

The generic function `addSubs` adds specific substances to the environment.
addSubs

Usage

addSubs(
  object,
  smax = 0,
  mediac = object@mediac,
  diffunc = "pde",
  pde = "Diff2d",
  difspeed = 0.02412,
  unit = "mmol/cell",
  add = TRUE,
  diffmat = NULL,
  template = FALSE,
  Dgrid = NULL,
  Vgrid = NULL,
  addAnyway = FALSE
)

## S4 method for signature 'Arena'
addSubs(
  object,
  smax = 0,
  mediac = object@mediac,
  diffunc = "pde",
  pde = "Diff2d",
  difspeed = 0.02412,
  unit = "mmol/cell",
  add = TRUE,
  diffmat = NULL,
  template = FALSE,
  Dgrid = NULL,
  Vgrid = NULL,
  addAnyway = FALSE
)

Arguments

object An object of class Arena.
smax A numeric vector indicating the maximum substance concentration per grid cell.
mediac A character vector giving the names of substances, which should be added to the environment (the default takes all possible substances).
diffunc A character vector ("pde","cpp" or "r") describing the function for diffusion.
pde Choose diffusion transport reaction to be used (default is diffusion only)
difspeed A number indicating the diffusion rate (given by cm^2/h). Default is set to glucose diffusion in a aqueous solution (6.7e-6 cm^2/s * 3600 s/h = 0.02412 cm^2/h )
unit A character used as chemical unit to set the amount of the substances to be added (valid values are: mmol/cell, mmol/cm2, mmol/arena, mM)
add       A boolean variable defining whether the amount of substance should be summed or replaced

diffmat   A matrix with spatial distributed initial concentrations (if not set, a homogenous matrix using smax is created)
template  True if diffmat matrix should be used as template only (will be multiplied with smax to obtain concentrations)
Dgrid     A matrix indicating the diffusion speed in x and y direction (given by cm^2/h).
Vgrid     A number indicating the advection speed in x direction (given by cm/h).
addAnyway If true substance will be added even if there is no connection (i.e. exchanges) with organisms

Details

If nothing but object is given, then all possible substrates are initialized with a concentration of 0. Afterwards, changeSub can be used to modify the concentrations of specific substances.

See Also

Arena-class and changeSub

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core,deathrate=0.05,
   minweight=0.05,growtype="exponential") # initialize a bacterium
arena <- Arena(n=20,m=20) # initialize the environment
arena <- addOrg(arena,bac,amount=10) # add 10 organisms
arena <- addSubs(arena,20,c("EX_glc(e)" ,"EX_o2(e)" ,"EX_pi(e)" )) # add glucose, o2, pi

Description

Structure of the S4 class "Arena"

Slots

orgdat    A data frame collecting information about the accumulated biomass, type, phenotype, x and y position for each individual in the environment.
specs     A list of organism types and their associated parameters.
media     A list of objects of class Substance-class for each compound in the environment.
phenotypes A list of unique phenotypes (metabolites consumed and produced), which occurred in the environment.
media \text{c} \quad \text{A character vector containing the names of all substances in the environment.}

tst{\text{e}}p \quad \text{A number giving the time (in h) per iteration.}

st{\text{i}}r \quad \text{A boolean variable indicating if environment should be stirred. If true, bacteria move to}
\quad \text{random positions within the environment and substances have a uniform concentration value.}

mflux \quad \text{A vector containing highly used metabolic reactions within the arena}

exchanges \quad \text{A data.frame containing last exchanges of each organism.}

shadow \quad \text{A vector containing shadow prices of metabolites present in the arena}

n \quad \text{A number giving the horizontal size of the environment.}

m \quad \text{A number giving the vertical size of the environment.}

Lx \quad \text{A number giving the horizontal grid size in cm.}

Ly \quad \text{A number giving the vertical grid size in cm.}

gridgeometry \quad \text{A list containing grid geometry parameter}

seed \quad \text{An integer refering to the random number seed used to be reproducible}

scale \quad \text{A numeric defining the scale factor used for intern unit conversion.}

models \quad \text{A list containing Objects of class sybil::modelorg which represent the genome scale metabolic}
\quad \text{models}

occupyM \quad \text{A matrix indicating grid cells that are obstacles}

sublb \quad \text{A data matrix containing positions with amounts of substance for all organism}

removeM \quad \text{A matrix indicating grid cells from which organisms are removed (i.e. killed) after each}
\quad \text{time step}

---

**Arena-constructor**

*Constructor of the S4 class Arena-class*

---

**Description**

Constructor of the S4 class **Arena-class**

**Usage**

```r
Arena(Lx = \text{NULL}, Ly = \text{NULL}, n = 100, m = 100, seed = \text{sample(1:10000, 1)}, \ldots)
```

**Arguments**

- **Lx** \quad \text{A number giving the horizontal grid size in cm.}
- **Ly** \quad \text{A number giving the vertical grid size in cm.}
- **n** \quad \text{A number giving the horizontal size of the environment.}
- **m** \quad \text{A number giving the vertical size of the environment.}
- **seed** \quad \text{An integer refering to the random number seed used to be reproducible}
- **\ldots** \quad \text{Arguments of Arena-class}
Bac-class  Structure of the S4 class "Bac"

Description

Structure of the S4 class Bac inheriting from class Organism-class representing bacterial cells.

Slots

deathrate A numeric value giving the factor by which the biomass should be reduced in every iteration if no growth is possible (default (E.coli): 0.21 pg)

minweight A numeric value giving the growth limit at which the organism dies. (default (E.coli): 0.083 pg)

cellarea A numeric value indicating the surface that one organism occupies (default (E.coli): 4.42 \mu m^2)

maxweight A numeric value giving the maximal dry weight of single organism (default (E.coli): 1.172 pg)

chem A character vector indicating name of substance which is the chemotaxis attractant. Empty character vector if no chemotaxis.

Bac Constructor Constructor of the S4 class Bac-class

Description

Constructor of the S4 class Bac-class

Usage

Bac(model, chem = "", ...)

Arguments

model model

chem A character vector indicating name of substance which is the chemotaxis attractant. Empty character vector if no chemotaxis.

... Arguments of Organism

Value

Object of class Bac-class
**BacArena:** An Agent-Based Modeling Framework for Cellular Communities

---

**Description**

The BacArena package provides six classes: Arena (subclass Eval), Organism (subclasses Bac, Human) and Substance. Accordingly there are three categories of important functions: Arena, Organism and Substance.

**Arena functions**

The Arena functions ...

**Organism functions**

The Organism functions ...

**Substance functions**

The Substance functions ...

---

**cellgrowth**  
*Function implementing a growth model of a human cell*

---

**Description**

The generic function `cellgrowth` implements different growth models for an object of class Human.

**Usage**

```r
cellgrowth(object, population, j, occupyM, fbasol, tstep)
```

## S4 method for signature 'Human'

```

cellgrowth(object, population, j, occupyM, fbasol, tstep)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object of class Human.</td>
</tr>
<tr>
<td>population</td>
<td>An object of class Arena.</td>
</tr>
<tr>
<td>j</td>
<td>The number of the iteration of interest.</td>
</tr>
<tr>
<td>occupyM</td>
<td>A matrix indicating grid cells that are obstacles</td>
</tr>
<tr>
<td>fbasol</td>
<td>Problem object according to the constraints and then solved with <code>optimizeProb</code>.</td>
</tr>
<tr>
<td>tstep</td>
<td>A number giving the time intervals for each simulation step.</td>
</tr>
</tbody>
</table>
Details

Linear growth of organisms is implemented by adding the calculated growth rate by optimizeLP to the already present growth value. Exponential growth of organisms is implemented by adding the calculated growth rate multiplied with the current growth calculated by optimizeLP plus to the already present growth value.

Value

Boolean variable of the jth individual indicating if individual died.

See Also

Human-class, growLin and growExp

changeDiff

Change substance concentration patterns in the environment

Description

The generic function changeDiff changes specific substance concentration patterns in the environment.

Usage

changeDiff(object, newdiffmat, mediach)

## S4 method for signature 'Arena'
changeDiff(object, newdiffmat, mediach)

Arguments

object
An object of class Arena.

newdiffmat
A matrix giving the new gradient matrix of the specific substances in the environment.

mediach
A character vector giving the names of substances, which should be added to the environment (the default takes all possible substances).

Details

This function can be used to add gradients of specific substances in the environment. The default conditions in changeSubs assumes an equal concentration in every grid cell of the environment.

See Also

Arena-class and changeSub
Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
           minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20,m=20) # initialize the environment
arena <- addOrg(arena,bac,amount=10) # add 10 organisms
arena <- addSubs(arena,30) # add all substances with no concentrations.
gradient <- matrix(1:200,20,20)
arena <- changeDiff(arena,gradient,c("EX_glc(e)","EX_o2(e)","EX_pi(e)"))
# add substances glucose, oxygen and phosphate

changeFobj

Function for changing the objective function of the model

Description

The generic function changeFobj changes the objective function, which is used for the linear programming in optimizeLP.

Usage

changeFobj(object, new_fobj, model, alg = "fba")

## S4 method for signature 'Human'
changeFobj(object, new_fobj, model, alg = "fba")

Arguments

object
A character vector giving the reaction name of the new objective function.

model
The original model structure which is converted into a problem object used for the next optimization.

alg
A character vector giving the algorithm which should be used for the optimization (default is flux balance analysis).

Details

To avoid the bias to just one particular objective function, the objective can be changed dynamically in this function.

See Also

Human-class and optimizeLP
changeOrg

Examples

```r
data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
human <- Human(Ec_core, deathrate=0.05,
    minweight=0.05, growtype="exponential") # initialize a bacterium
changeFobj(human, 'EX_glc(e)', Ec_core)
```

---

changeOrg

### Change organisms in the environment

**Description**

The generic function `changeOrg` changes organisms in the environment.

**Usage**

```r
changeOrg(object, neworgdat)
```

```
## S4 method for signature 'Arena'
changeOrg(object, neworgdat)
```

**Arguments**

- `object`: An object of class Arena.
- `neworgdat`: A data frame with new information about the accumulated biomass, type, phenotype, x and y position for each individual in the environment.

**Details**

The argument `neworgdat` contains the same information as the `orgdat` slot of `Arena-class`. The `orgdat` slot of an Arena object can be used to create `neworgdat`.

**See Also**

`Arena-class` and `addOrg`

**Examples**

```r
data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
    minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
neworgdat <- arena@orgdat # get the current orgdat
neworgdat <- neworgdat[-1,] # remove the first individual
arena <- changeOrg(arena, neworgdat)
```
Description

The generic function changeSub changes specific substances in the environment.

Usage

changeSub(object, smax, mediac, unit = "mmol/cell")

## S4 method for signature 'Arena'
changeSub(object, smax, mediac, unit = "mmol/cell")

Arguments

- **object**: An object of class Arena.
- **smax**: A number or vector of numbers indicating the maximum substance concentration per grid cell.
- **mediac**: A character vector giving the names of substances, which should be added to the environment (the default takes all possible substances).
- **unit**: A character used as chemical unit to set the amount of the substances to be added (valid values are: mmol/cell, mmol/cm², mmol/arena, mM)

Details

If nothing but object is given, then all possible substrates are initialized with a concentration of 0. Afterwards, changeSub can be used to modify the concentrations of specific substances.

See Also

Arena-class and addSubs

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena) # add all substances with no concentrations.
arena <- changeSub(arena, 20, c("EX_glc(e)", "EX_o2(e)", "EX_pi(e)")) # add substances glucose, oxygen and phosphate
checkCorr  

Function to show correlations of a simulated organism or substrate

Description

The generic function checkCorr returns the correlation matrix of several objects.

Usage

checkCorr(object, corr = NULL, tocheck = list())

## S4 method for signature 'Eval'
checkCorr(object, corr = NULL, tocheck = list())

Arguments

object An object of class Eval.
corr A correlation matrix (getCorrM)
tocheck A list with substrate, reactions or organism names whose correlations should be shown

Details

Returns correlation matrix which can be used for statistical analysis

See Also

Eval-class and getCorrM

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, 
         minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
eval <- simEnv(arena, 5)
checkCorr(eval, tocheck="o2")
Description

The generic function `checkPhen` checks and adds the phenotypes of organisms in the environment.

Usage

```r
checkPhen(object, org, cutoff = 1e-06, fbasol)
```

### S4 method for signature 'Arena'

```r
checkPhen(object, org, cutoff = 1e-06, fbasol)
```

Arguments

- **object**: An object of class Arena.
- **org**: An object of class Organism.
- **cutoff**: A number giving the cutoff for values of the objective function and fluxes of exchange reactions.
- **fbasol**: Problem object according to the constraints and then solved with `optimizeProb`.

Details

The phenotypes are defined by flux through exchange reactions, which indicate potential differential substrate usages. Uptake of substances are indicated by a negative and production of substances by a positive number.

Value

Returns a number indicating the number of the phenotype in the phenotype list.

See Also

`Arena-class` and `getPhenotype`
**checkPhen_par**  
*Function for checking phenotypes in the environment*

**Description**

The generic function `checkPhen_par` checks and adds the phenotypes of organisms in the environment.

**Usage**

```r
checkPhen_par(object, org, cutoff = 1e-06, fbasol)
```

## S4 method for signature 'Arena'

```r
checkPhen_par(object, org, cutoff = 1e-06, fbasol)
```

**Arguments**

- `object`: An object of class `Arena`.
- `org`: An object of class `Organism`.
- `cutoff`: A number giving the cutoff for values of the objective function and fluxes of exchange reactions.
- `fbasol`: Problem object according to the constraints and then solved with `optimizeProb`.

---

**chemotaxis**  
*Function for chemotaxis of bacteria to their preferred substrate*

**Description**

The generic function `chemotaxis` implements a bacterial movement in the Moore neighbourhood to the highest substrate concentration.

**Usage**

```r
chemotaxis(object, population, j, chemo, occupyM)
```

## S4 method for signature 'Bac'

```r
chemotaxis(object, population, j, chemo, occupyM)
```

**Arguments**

- `object`: An object of class `Bac`.
- `population`: An object of class `Arena`.
- `j`: The number of the iteration of interest.
- `chemo`: The vector that contains the preferred substrate.
- `occupyM`: A matrix indicating grid cells that are obstacles.
Details

Bacteria move to a position in the Moore neighbourhood which has the highest concentration of the preferred substrate, which is not occupied by other individuals. The preferred substance is given by slot chem in the Bac object. If there is no free space the individuals stays in the same position. If the concentration in the Moore neighbourhood has the same concentration in every position, then random movement is implemented.

See Also

Bac-class and emptyHood

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, chem = "EX_o2(e)",
            minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
chemotaxis(bac, arena, 1, "EX_glc(e)", arena@occupyM)

---

colpal1

Color palette

Description

color dictionary of 269 maximally distinct colors from all previous colors

Usage

colpal1

Format

A vector with 269 color hex-codes

Source

**colpal2**  

*Color palette*

**Description**

20 optimally distinct colors

**Usage**

colpal2

**Format**

A vector with 20 color hex-codes

**Source**

http://graphicdesign.stackexchange.com/questions/3682/where-can-i-find-a-large-palette-set-of-contrasting-colors

---

**colpal3**  

*Color palette*

**Description**

K. Kelly (1965): Twenty-two colors of maximum contrast. // Color Eng., 3(6), 1965

**Usage**

colpal3

**Format**

A vector with 22 color hex-codes

**Source**

colpal4  

**Color palette**

**Description**


**Usage**

colpal4

**Format**

A vector with 26 color hex-codes

**Source**


---

colpal5  

**Color palette**

**Description**

64 distinct colors

**Usage**

colpal5

**Format**

A vector with 64 color hex-codes

**Source**

http://graphicdesign.stackexchange.com/questions/3682/where-can-i-find-a-large-palette-set-of-contrasting-colors


**colpal6**  

*Color palette*

**Description**

64 distinct colors

**Usage**

colpal6

**Format**

A vector with 64 color hex-codes

**Source**

http://graphicdesign.stackexchange.com/questions/3682/where-can-i-find-a-large-palette-set-of-contrasting-colors-for-coloring-many-d

**constrain**  

*Function for constraining the models based on metabolite concentrations*

**Description**

The generic function `constrain` changes the constraints of the model representation of an organism.

**Usage**

`constrain(object, reacts, lb, dryweight, tstep, scale, j, cutoff = 1e-06)`

```r
## S4 method for signature 'Organism'
constrain(object, reacts, lb, dryweight, tstep, scale, j, cutoff = 1e-06)
```

**Arguments**

- `object`  
  An object of class Organisms.
- `reacts`  
  A character vector giving the names of reactions which should be constrained.
- `lb`  
  A numeric vector giving the constraint values of lower bounds (e.g. available metabolite concentrations)
- `dryweight`  
  A number giving the current dryweight of the organism.
- `tstep`  
  A number giving the time intervals for each simulation step.
- `scale`  
  A numeric defining the scaling (units for linear programming has to be in certain range)
- `j`  
  Debugging index to track cell
- `cutoff`  
  Value used to define numeric accuracy while interpreting optimization results
consume

Details

The constraints are calculated according to the flux definition as mmol/(gDW*hr) with the parameters dryweight and tstep.

Value

Returns the lower bounds, which carry the constraints and names of relevant reactions.

See Also

Organism-class

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
org <- Organism(Ec_core, deathrate=0.05, 
  minweight=0.05, growtype="exponential") #initialize an organism
lobnds <- constrain(org, org@medium, org@lbnd[[org@medium]], 1, 1, 1, 1, 1)

consume Function to account for the consumption and production of substances

Description

The generic function consume implements the consumption and production of substances based on the flux of exchange reactions of organisms

Usage

consume(object, sublb, cutoff = 1e-06, bacnum, fbasol)

## S4 method for signature 'Organism'
consume(object, sublb, cutoff = 1e-06, bacnum, fbasol)

Arguments

- **object**: An object of class Organisms.
- **sublb**: A vector containing the substance concentrations in the current position of the individual of interest.
- **cutoff**: A number giving the cutoff value by which value of objective function is considered greater than 0.
- **bacnum**: Integer indicating the number of bacteria individuals per gridcell
- **fbasol**: Problem object according to the constraints and then solved with optimizeProb.

Details

The consumption is implemented by adding the flux of the exchange reactions to the current substance concentrations.
createGradient

Value

Returns the updated vector containing the substance concentrations in the current position of the individual of interest.

See Also

Organism-class

Examples

NULL

createGradient  Change substance concentration patterns in the environment according to a gradient

Description

The generic function createGradient changes specific substance concentration patterns in the environment.

Usage

createGradient(
  object,
  mediac,
  position,
  smax,
  steep,
  add = FALSE,
  unit = "mmol/cell"
)

## S4 method for signature 'Arena'
createGradient(
  object,
  mediac,
  position,
  smax,
  steep,
  add = FALSE,
  unit = "mmol/cell"
)
Arguments

object

An object of class Arena.

mediac

A character vector giving the names of substances, which should be added to the environment (the default takes all possible substances).

caption

A character vector giving the position (top, bottom, right and left) of the gradient.

smax

A number giving the maximum concentration of the substance.

steep

A number between 0 and 1 giving the steepness of the gradient (concentration relative to the arena size).

add

A boolean variable defining whether the amount of substance should be summed or replaced.

unit

A character used as chemical unit to set the amount of the substances to be added (valid values are: mmol/cell, mmol/cm2, mmol/arena, mM)

Details

This function can be used to add gradients of specific substances in the environment.

See Also

Arena-class and changeSub

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
    minweight=0.05,growtype="exponential") # initialize a bacterium
arena <- Arena(n=20,m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena,30) # add all substances with no concentrations.
arena <- createGradient(arena, smax=50, mediac=c("EX_glc(e)","EX_o2(e)","EX_pi(e)")
    position='top',steep=0.5, add=FALSE)

---

dat2mat

Function for transforming the organism data frame to a presence/absence matrix of organisms

Description

The generic function dat2mat simulates the event of mixing all substrates and organisms in the environment.

Usage

dat2mat(object)

## S4 method for signature 'Arena'
dat2mat(object)
diffuse

Arguments

  object  An object of class Arena.

Value

Returns the presence/absence matrix of organisms on the grid based on the orgdat slot of the Arena class.

See Also

Arena-class and getSublb

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
          minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
occmat <- dat2mat(arena)
image(occmat)

---

diffuse  

Function for diffusion

Description

The generic function diffuse computes the media distribution via diffusion

Usage

diffuse(object, lrw, sublb, verbose = TRUE)

## S4 method for signature 'Arena'
diffuse(object, lrw, sublb, verbose = TRUE)

Arguments

  object  An object of class Arena.
  lrw     A numeric value needed by solver to estimate array size (by default lrw is estimated in the simEnv() by the function estimate_lrw())
  sublb   A matrix with the substrate concentration for every individual in the environment based on their x and y position.
  verbose Set to false if no status messages should be printed.
diffusePDE  
Function for diffusion of the Substance matrix

Description

The generic function diffusePDE implements the diffusion by solving the diffusion equation.

Usage

```r
diffusePDE(object, init_mat, gridgeometry, lrw = NULL, tstep)
```

## S4 method for signature 'Substance'

diffusePDE(object, init_mat, gridgeometry, lrw = NULL, tstep)

Arguments

- `object`: An object of class Substance.
- `init_mat`: A matrix with values to be used by the diffusion.
- `gridgeometry`: A list specifying the geometry of the Arena.
- `lrw`: A numeric value needed by solver to estimate array size (by default lrw is estimated in simEnv() by the function estimate_lrw()).
- `tstep`: A numeric value giving the time step of integration.

Details

Partial differential equation is solved to model 2D diffusion process in the arena.

See Also

Substance-class and diffuseR

diffuseR  
Function for naive diffusion (neighbourhood) of the Substance matrix

Description

The generic function diffuseR implements the diffusion in the Moore neighbourhood in R.

Usage

```r
diffuseR(object)
```

## S4 method for signature 'Substance'

diffuseR(object)
diffuse_par

Arguments

object An object of class Substance.

Details

The diffusion is implemented by iterating through each cell in the grid and taking the cell with the lowest concentration in the Moore neighbourhood to update the concentration of both by their mean.

See Also

Substance-class and diffusePDE

diffuse_par Function for parallelized diffusion

Description

The generic function diffuse_par computes the media distribution via diffusion in parallel

Usage

diffuse_par(object, lrw, cluster_size, sublb)

## S4 method for signature 'Arena'
diffuse_par(object, lrw, cluster_size, sublb)

Arguments

object An object of class Arena.

lrw A numeric value needed by solver to estimate array size (by default lrw is estimated in the simEnv() by the function estimate_lrw())

cluster_size Amount of cores to be used

sublb A matrix with the substrate concentration for every individual in the environment based on their x and y position.
emptyHood

Function to check if there is a free place in the Moore neighbourhood

Description

The generic function emptyHood gives a free space which is present in the Moore neighbourhood of an individual of interest.

Usage

emptyHood(object, pos, n, m, x, y, occupyM, inverse = FALSE)

## S4 method for signature 'Organism'
emptyHood(object, pos, n, m, x, y, occupyM, inverse = FALSE)

Arguments

object An object of class Organisms.
pos A dataframe with all occupied x and y positions
n A number giving the horizontal size of the environment.
m A number giving the vertical size of the environment.
x A number giving the x position of the individual of interest in its environment.
y A number giving the y position of the individual of interest in its environment.
occupyM A matrix indicating grid cells that are obstacles
inverse Return occupied positions instead

Value

Returns the free position in the Moore neighbourhood, which is not occupied by other individuals. If there is no free space NULL is returned.

See Also

Organism-class

Examples

NULL
Structure of the S4 class "Eval"

Description

Structure of the S4 class Eval inheriting from class Arena-class for the analysis of simulations.

Slots

- medlist  A list of compressed medium concentrations (only changes of concentrations are stored) per time step.
- simlist  A list of the organism features per time step.
- mfluxlist  A list of containing highly used metabolic reactions per time step.
- shadowlist  A list of containing shadow prices per time step.
- subchange  A vector of all substrates with numbers indicating the degree of change in the overall simulation.
- exchangeslist  A list of containing exchanges per time step.

Constructor of the S4 class Eval-class

Description

Constructor of the S4 class Eval-class

Usage

Eval(arena)

Arguments

arena  An object of class Arena.
evalArena

Function for plotting spatial and temporal change of populations and/or concentrations

Description

The generic function evalArena plots heatmaps from the simulation steps in an Eval object.

Usage

evalArena(
  object,
  plot_items = "Population",
  phencol = F,
  retdata = F,
  time = (seq_along(object@simlist) - 1),
  show_legend = TRUE,
  legend_pos = "left"
)

## S4 method for signature 'Eval'

evalArena(
  object,
  plot_items = "Population",
  phencol = F,
  retdata = F,
  time = (seq_along(object@simlist) - 1),
  show_legend = TRUE,
  legend_pos = "left"
)

Arguments

object
  An object of class Eval.

plot_items
  A character vector giving the name of the items which should be plotted such as the population structure and several metabolites.

phencol
  A boolean variable indicating if the phenotypes of the organisms in the environment should be integrated as different colors in the population plot.

retdata
  A boolean variable indicating if the data used to generate the plots should be returned.

time
  A numeric vector giving the simulation steps which should be plotted.

show_legend
  A boolean variable indicating if a legend should be shown.

legend_pos
  Position of the legend.
Details

If phencol is TRUE then different phenotypes of the same organism are visualized by varying colors, otherwise different organism types are represented by varying colors. The parameter retdata can be used to access the data used for the returned plots to create own custom plots.

Value

Returns several plots of the chosen plot items. Optional the data to generate the original plots can be returned.

See Also

Eval-class and Arena-class

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
  minweight=0.05, growtype="exponential") #initialize a bacterium
arena <- Arena(n=20, m=20) #initialize the environment
arena <- addOrg(arena, bac, amount=10) #add 10 organisms
arena <- addSubs(arena, 40) #add all possible substances
eval <- simEnv(arena, 5)
evalArena(eval)
## Not run:
## if animation package is installed a movie of the simulation can be stored:
library(animation)
saveVideo(evalArena(eval), video.name="Ecoli_sim.mp4")
## End(Not run)

extractMed

Function for re-constructing a medium concentrations from simulations

Description

The generic function extractMed re-constructs a list of vectors of medium concentrations from a simulation step in an Eval object.

Usage

extractMed(object, time = length(object@medlist), mediac = object@mediac)

## S4 method for signature 'Eval'
extractMed(object, time = length(object@medlist), mediac = object@mediac)
Arguments

- object: An object of class Eval.
- time: A number giving the simulation step of interest.
- mediac: A character vector giving the names of substances, which should be added to the environment (the default takes all possible substances).

Details

Medium concentrations in slot medlist of an object of class Eval store only the changes of concentrations in the simulation process. The function `extractMed` reconstructs the original and uncompressed version of medium concentrations.

Value

Returns a list containing concentration vectors of all medium substances.

See Also

Eval-class and Arena-class

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") #initialize a bacterium
arena <- Arena(n=20, m=20) #initialize the environment
arena <- addOrg(arena, bac, amount=10) #add 10 organisms
arena <- addSubs(arena, 40) #add all possible substances
eval <- simEnv(arena, 5)
med5 <- extractMed(eval, 5)

---

findFeeding  
*Function for investigation of feeding between phenotypes*

Description

The generic function `findFeeding`

Usage

```r
findFeeding(
  object,
  dict = NULL,
  tcut = 5,
  scut = NULL,
  org_dict = NULL,
  legendpos = "topleft",
)```


findFeeding2

lwd = 1
)

## S4 method for signature 'Eval'
findFeeding(
  object,
  dict = NULL,
  tcut = 5,
  scut = NULL,
  org_dict = NULL,
  legendpos = "topleft",
  lwd = 1
)

Arguments

object An object of class Eval.
dict List defining new substance names. List entries are interpreted as old names and the list names as the new ones.
tcut Integer giving the minimal mutual occurrence to be considered (dismiss very seldom feedings)
scut substance names which should be ignored
org_dict A named list/vector with names that should replace (eg. unreadable) IDs
legendpos A character variable declaring the position of the legend
lwd Line thickness scale in graph

Value

Graph (igraph)

findFeeding2 Function for investigation of feeding between phenotypes

Description

The generic function findFeeding2

Usage

findFeeding2(object, time, mets, rm_own = T, ind_threshold = 0, collapse = F)

## S4 method for signature 'Eval'
findFeeding2(object, time, mets, rm_own = T, ind_threshold = 0, collapse = F)
Arguments

object  An object of class Eval.
time  A numeric vector giving the simulation steps which should be plotted.mets  Character vector of substance names which should be consideredrm_own  A boolean flag indicating if interactions within same species should be plottedind_threshold  A number indicating the threshold of individuals to be considered as producers/consumerscollapse  A boolean flag indicating if all phenotypes for every species should be collapsed to either producers or consumers

Value

Graph (igraph)

---

findFeeding3  Function for investigation of feeding between phenotypes

Description

The generic function findFeeding3

Usage

findFeeding3(object, time, mets, plot = TRUE, cutoff = 1e-06)

## S4 method for signature 'Eval'
findFeeding3(object, time, mets, plot = TRUE, cutoff = 1e-06)

Arguments

object  An object of class Eval.
time  A numeric vector giving the simulation steps which should be plotted.mets  Character vector of substance names which should be consideredplot  Should the graph also be plotted?cutoff  Accuracy of crossfeeding interaction (minimal flux to be considered)

Value

Graph (igraph)
**findFeeding3rep**  
*Function for investigation of cross feeding patterns of replicate simulations*

**Description**

The generic function `findFeeding3rep` investigates the cross feeding patterns of replicate simulations.

**Usage**

```r
findFeeding3rep(simlist, time, mets, plot = TRUE, mfunction = "mean")
```

**Arguments**

- `simlist`: A list with objects of class Eval.
- `time`: A numeric vector giving the simulation steps which should be plotted.
- `mets`: Character vector of substance names which should be considered.
- `plot`: Should the graph also be plotted?
- `mfunction`: Function by which the replicate simulations should be combined e.g. "mean" or "median".

**Value**

Graph (igraph)

---

**findInArena**  
*Function for searching a keyword in arena organisms and media*

**Description**

The generic function `findInArena` tries to find information (e.g. full names) about a specific keyword.

**Usage**

```r
findInArena(object, pattern, search_rea = TRUE, search_sub = TRUE)
```

```r
## S4 method for signature 'Arena'
findInArena(object, pattern, search_rea = TRUE, search_sub = TRUE)
```
findrBiomass

Arguments

object An object of class Arena.
pattern A pattern for searching
search_rea Only search for reactions
search_sub Only search for substances

Examples

data(Ec_core)
bac <- Bac(Ec_core)
arena <- Arena(n=20,m=20)
arena <- addOrg(arena,bac,amount=10)
findInArena(arena, "acetate")

findrBiomass Find biomass reaction in model

Description

Helper function to search for biomass reaction in available reactions of a model

Usage

findrBiomass(model, keys = c("biom", "cpd11416"))

Arguments

model Object of class sybil::modelorg containg the genome sclae metabolic model
keys Vector with strings which are used to find biomass reaction in model

Value

Vector with reaction ids for biomass reaction(s)
findRxnFlux

Function to get all reactions fluxes that are associated with the metabolite of a given exchange reactions

Description

The generic function findRxnFlux returns a matrix with the flux for each organism and the reaction that is using the metabolite of the given exchange reaction.

Usage

findRxnFlux(object, ex, time, print_reactions = FALSE, drop_unused = TRUE)

## S4 method for signature 'Eval'
findRxnFlux(object, ex, time, print_reactions = FALSE, drop_unused = TRUE)

Arguments

- **object**: An object of class Eval.
- **ex**: An exchange reaction of which the metabolite should be shared for in all reactions.
- **time**: The time point of the simulation which should be considered.
- **print_reactions**: If true the detailed definition of each reaction is printed.
- **drop_unused**: If true then inactive reactions will be excluded.

Details

Returns a list with the minimum and maximum substance usage for each time point.

See Also

Eval-class and simEnv

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
eval <- simEnv(arena, 5)
fluxlist <- findRxnFlux(evalu, "EX_h(e)", 5)
flushSubs **Remove all substances in the environment**

**Description**

The generic function `flushSubs` removes specific substances in the environment.

**Usage**

```r
flushSubs(object)

## S4 method for signature 'Arena'
flushSubs(object)
```

**Arguments**

- `object` An object of class Arena.

**See Also**

`Arena-class` and `addSubs`

**Examples**

```r
data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
           minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, smax=40) # add all substances with no concentrations.
arena <- changeSub(arena, 20, c("EX_glc(e)" , "EX_o2(e)" , "EX_pi(e)"))
# add substances glucose, oxygen and phosphate
arena <- flushSubs(arena) # remove all created substance concentrations
```

---

fluxVarSim **Function to compute flux variability analysis on an simulation object to get min/max of substance usage**

**Description**

The generic function `fluxVarSim` returns a list with the minimum and maximum substance usage of all individuals for each simulation step.
Usage

fluxVarSim(object, rnd)

## S4 method for signature 'Eval'
fluxVarSim(object, rnd)

Arguments

object An object of class Eval.
rnd An integer giving the decimal place to which min/max flux should be rounded.

Details

Returns a list with the minimum and maximum substance usage for each time point.

See Also

eval-class and simEnv

Description

The generic function getArena re-constructs an Arena object from a simulation step within an Eval object.

Usage

getArena(object, time = (length(object@medlist) - 1))

## S4 method for signature 'Eval'
getArena(object, time = (length(object@medlist) - 1))

Arguments

object An object of class Eval.
time A number giving the simulation step of interest.

Details

The function addEval can be used to manipulate an Arena object from a simulation step to modify the subsequent simulation steps.

Value

Returns an object of class Arena containing the organisms and substance conditions in simulation step time.
getCorrM

Function to compute and return correlation matrix

Description

The generic function `getCorrM` returns the correlation matrix of several objects.

Usage

```
getCorrM(object, reactions = TRUE, bacs = TRUE, substrates = TRUE)
```

Arguments

- `object`: An object of class `Eval`.
- `reactions`: A boolean indicating whether reactions should be included in correlation matrix.
- `bacs`: A boolean indicating whether bacteria should be included in correlation matrix.
- `substrates`: A boolean indicating whether substrates should be included in correlation matrix.

Details

Returns correlation matrix which can be used for statistical analysis.

Value

correlation matrix

See Also

`Eval-class`
getPhenoMat

Function for getting a matrix of phenotypes from the dataset

Description
The generic function getPhenoMat reconstructs a matrix with the usage of exchange reactions of the different organisms in the environment.

Usage
getPhenoMat(object, time = "total", sparse = F)

## S4 method for signature 'Eval'
getPhenoMat(object, time = "total", sparse = F)

Arguments
- object: An object of class Eval.
- time: An integer indicating the time step to be used (default value is character "total")
- sparse: A boolean indicating whether zero entries should be removed from return matrix

Details
The phenotypes are defined by flux through exchange reactions, which indicate potential differential substrate usages.

Value
Returns a matrix with different phenotypes of the organism as rows and all possible exchange reactions as columns. A value of 1 means secretion, 2 means uptake and 0 means no usage of the substance of interest.

See Also
Eval-class and getPhenotype
Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
          minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
eval <- simEnv(arena, 5)
phenmat <- getPhenoMat(eval)

getPhenotype

Function to extract the phenotype of an organism object

Description

The generic function getPhenotype implements an identification of organism phenotypes.

Usage

getPhenotype(object, cutoff = 1e-06, fbasol, par = FALSE)

## S4 method for signature 'Organism'
getPhenotype(object, cutoff = 1e-06, fbasol, par = FALSE)

Arguments

- object: An object of class Organisms.
- cutoff: A number giving the cutoff value by which value of objective function is considered greater than 0.
- fbasol: Problem object according to the constraints and then solved with optimizeProb.
- par: A boolean indicating if running in parallel mode.

Details

The phenotypes are defined by flux through exchange reactions, which indicate potential differential substrate usages. Uptake of substances is indicated by a negative and production of substances by a positive number.

Value

Returns the phenotype of the organisms where the uptake of substances is indicated by a negative and production of substances by a positive number.

See Also

Organism-class, checkPhen and minePheno
getSubHist

Function to get timeline of a substance

Description

The generic function getSubHist returns list with amount of substance for each timestep.

Usage

getSubHist(object, sub, unit = "fmol/cell")

## S4 method for signature 'Eval'
getSubHist(object, sub, unit = "fmol/cell")

Arguments

- object: An object of class Eval.
- sub: Vector with substances.
- unit: Unit to be used.

getSublb

Function for calculated the substrate concentration for every organism

Description

The generic function getSublb calculates the substrate concentration for every individual in the environment based on their x and y position.

Usage

getSublb(object)

## S4 method for signature 'Arena'
getSublb(object)

Arguments

- object: An object of class Arena.

Value

Returns the substrate concentration for every individual in the environment with substrates as well as x and y positions as columns and rows for each organism.
### getVarSubs

Function to get varying substances

---

**Description**

The generic function `getVarSubs` returns ordered list of substances that showed variance during simulation.

**Usage**

```r
getVarSubs(
  object,
  show_products = TRUE,
  show_substrates = TRUE,
  cutoff = 1e-06,
  size = NULL
)
```

```
## S4 method for signature 'Eval'
getVarSubs(
  object,
  show_products = FALSE,
  show_substrates = FALSE,
  cutoff = 1e-06,
  size = NULL
)
```

**Arguments**

- **object**: An object of class Eval.
- **show_products**: A boolean indicating if only products should be shown.
- **show_substrates**: A boolean indicating if only substrates should be shown.
- **cutoff**: Value used to define numeric accuracy while interpreting optimization results.
- **size**: Maximal number of returned substances (default: show all).

---

**See Also**

- **Arena-class**

**Examples**

```r
data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
  minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
sublb <- getSublb(arena)
```

```
getVarSubs
```
**growExp**

*Function for letting organisms grow exponentially*

**Description**

The generic function `growExp` implements a growth model of organisms in their environment.

**Usage**

```r
growExp(object, biomass, fbasol, tstep)
```

```r
## S4 method for signature 'Organism'
growExp(object, biomass, fbasol, tstep)
```

**Arguments**

- `object`: An object of class Organisms.
- `biomass`: A number indicating the current biomass, which has to be updated.
- `fbasol`: Problem object according to the constraints and then solved with `optimizeProb`.
- `tstep`: A number giving the time intervals for each simulation step.

**Details**

Exponential growth of organisms is implemented by adding the calculated growthrate multiplied with the current growth calculated by `optimizeLP` plus to the already present growth value.

**Value**

Returns the updated biomass of the organisms of interest.

**See Also**

- `Organism-class` and `optimizeLP`

---

**growLin**

*Function for letting organisms grow linearly*

**Description**

The generic function `growLin` implements a growth model of organisms in their environment.

**Usage**

```r
growLin(object, biomass, fbasol, tstep)
```

```r
## S4 method for signature 'Organism'
growLin(object, biomass, fbasol, tstep)
```
growth

Arguments

object An object of class Organisms.
biomass A number indicating the current biomass, which has to be updated.
fbasol Problem object according to the constraints and then solved with optimizeProb.
tstep A number giving the time intervals for each simulation step.

Details

Linear growth of organisms is implemented by adding the calculated growthrate by optimizeLP to the already present growth value.

Value

Returns the updated biomass of the organisms of interest.

See Also

Organism-class and optimizeLP

growth Function implementing a growth model of a bacterium

Description

The generic function growth implements different growth models for an object of class Bac.

Usage

growth(object, population, j, occupyM, fbasol, tstep)

## S4 method for signature 'Bac'
growth(object, population, j, occupyM, fbasol, tstep)

Arguments

object An object of class Bac.
population An object of class Arena.
j The index of the organism of interest in orgdat.
occupyM A matrix indicating grid cells that are obstacles
fbasol Problem object according to the constraints and then solved with optimizeProb.
tstep A number giving the time intervals for each simulation step.
Details

Linear growth of organisms is implemented by adding the calculated growth rate by \texttt{optimizeLP} to the already present growth value. Exponential growth of organisms is implemented by adding the calculated growth rate multiplied with the current growth calculated by \texttt{optimizeLP} plus to the already present growth value.

Value

Boolean variable of the \( j \)th individual indicating if individual died.

See Also

\texttt{Bac-class}, \texttt{growLin} and \texttt{growExp}

---

\textit{growth_par} \hspace{1cm} \textit{Function implementing a growth model of a bacterium}

Description

The generic function \texttt{growth_par} implements different growth models for an object of class \texttt{Bac}.

Usage

\begin{verbatim}
growth_par(object, population, j, fbasol, tstep)

## S4 method for signature 'Bac'
growth_par(object, population, j, fbasol, tstep)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{object} \hspace{1cm} An object of class \texttt{Bac}.
  \item \texttt{population} \hspace{1cm} An object of class \texttt{Arena}.
  \item \texttt{j} \hspace{1cm} The index of the organism of interest in \texttt{orgdat}.
  \item \texttt{fbasol} \hspace{1cm} Problem object according to the constraints and then solved with \texttt{optimizeProb}.
  \item \texttt{tstep} \hspace{1cm} A number giving the time intervals for each simulation step.
\end{itemize}

Value

A list
Description

Structure of the S4 class "Human" inheriting from class Organism-class representing human cells.

Slots

objective A character vector representing the current reaction which should be used as an objective function for the flux balance analysis.

Usage

Human(
  model,
  objective = model@react_id[which(model@obj_coef != 0)],
  speed = 0,
  ...
)

Arguments

model model
objective A character vector representing the current reaction which should be used as an objective function for the flux balance analysis.
speed A integer vector representing the speed by which bacterium is moving (given by cell per iteration).
... Arguments of Organism

Value

Object of class Human-class
**lsd**  
*Computer standard deviation lower bound*

**Description**

Helper function to get lower error bounds in plotting

**Usage**

```r
lsd(y)
```

**Arguments**

- `y` Vector with numbers

---

**lysis**  
*Lysis function of organismal cells by adding biomass compounds to the medium*

**Description**

The generic function `lysis` implements cell lysis by the stochiometric concentration of the biomass compounds of organisms to the concentration of substances in the environment

**Usage**

```r
lysis(object, sublb, factor = object@minweight)
```

```r
# S4 method for signature 'Organism'
lysis(object, sublb, factor = object@minweight)
```

**Arguments**

- `object` An object of class Organisms.
- `sublb` A vector containing the substance concentrations in the current position of the individual of interest.
- `factor` A number given the factor with which the biomass compound concentrations are multiplied to achieve the final concentration which is added to the environment

**Details**

Lysis is implemented by taking the intersect between biomass compounds and the substances in the environment and adding the normalized stochiometric concentrations of the biomass compounds to the medium.
minePheno

Value

Returns the updated vector containing the substance concentrations in the current position of the dead individual of interest.

See Also

Organism-class and optimizeLP

Examples

NULL

Arguments

object An object of class Eval.
plot_type A character vector giving the plot which should be returned (either "pca" for a principle coordinate analysis or "hclust" for hierarchical clustering).
legend Boolean variable indicating if legend should be plotted
time An integer indicating the time step to be used (default value is character "total")

Details

The phenotypes are defined by flux through exchange reactions, which indicate potential differential substrate usages.

Value

Returns a plot for each simulation step representing the similarity of phenotypes of organisms within the environment.
move

Function for random movement of organisms

Description
The generic function move implements a random movement in the Moore neighbourhood of an individual.

Usage
move(object, pos, n, m, j, occupyM)

## S4 method for signature 'Organism'
m...ove(object, pos, n, m, j, occupyM)

Arguments
- **object**: An object of class Organism.
- **pos**: A dataframe with all occupied x and y positions
- **n**: A number giving the horizontal size of the environment.
- **m**: A number giving the vertical size of the environment.
- **j**: The number of the iteration of interest.
- **occupyM**: A matrix indicating grid cells that are obstacles

Details
Organisms move in a random position the Moore neighbourhood, which is not occupied by other individuals. If there is no free space the individuals stays in the same position.

See Also
- Organism-class, emptyHood
Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
  minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
move(bac, n=20, m=20, j=1, pos=arena@orgdat[,c("Var_x","Var_y")], occupyM=arena@occupyM)

NemptyHood

Function to check if there is a free place in the Moore neighbourhood

Description

The generic function NemptyHood gives a free space which is present in the Moore neighbourhood of an individual of interest.

Usage

NemptyHood(object, pos, n, m, x, y, occupyM, inverse = FALSE)

## S4 method for signature 'Organism'
NemptyHood(object, pos, n, m, x, y, occupyM, inverse = FALSE)

Arguments

object An object of class Organisms.
pos A dataframe with all occupied x and y positions
n A number giving the horizontal size of the environment.
m A number giving the vertical size of the environment.
x A number giving the x position of the individual of interest in its environment.
y A number giving the y position of the individual of interest in its environment.
occupyM A matrix indicating grid cells that are obstacles
inverse Return occupied positions instead

Value

Returns the free position in the Moore neighbourhood, which is not occupied by other individuals. If there is no free space NULL is returned.

See Also

Organism-class

Examples

NULL
openArena  

**Description**

The function `openArena` can be used to start a default simulation.

**Usage**

```r
openArena()
```

**Value**

Returns an object of class `Eval` which can be used for subsequent analysis steps.

**Examples**

```r
sim <- openArena()
evalArena(sim, time=5, phencol = TRUE,
plot_items=c("Population", "EX_o2(e)", "EX_for(e)", "EX_glc(e)"))
```

---

**optimizeLP**  

*Function for computing the linear programming according to the model structure*

**Description**

The generic function `optimizeLP` implements a linear programming based on the problem structure and refined constraints.

**Usage**

```r
optimizeLP(
  object,
  lpob = object@lpobj,
  lb = object@lbnd,
  ub = object@ubnd,
  cutoff = 1e-06,
  j,
  sec_obj = "none",
  with_shadow = FALSE
)
```

```r
## S4 method for signature 'Organism'
```
optimizeLP(
    object,
    lpob = object@lpobj,
    lb = object@lbnd,
    ub = object@ubnd,
    cutoff = 1e-06,
    j,
    sec_obj = "none",
    with_shadow = FALSE
)

Arguments

object An object of class Organisms.
lpob A linear programing object encoding the problem to solve.
lb A numeric vector giving the constraint values of lower bounds.
ub A numeric vector giving the constraint values of upper bounds.
cutoff value used to define numeric accuracy while interpreting optimization results
j debugging index to track cell
sec_obj character giving the secondary objective for a bi-level LP if wanted. Use "mtf" for minimizing total flux, "opt_rxn" for optimizing a random reaction, "opt_ex" for optimizing a random exchange reaction, and "sumex" for optimizing the sum of all exchange fluxes.
with_shadow True if shadow costs should be stored (default false).

Details

The parameter for sec_obj can be used to optimize a bi-level LP with a secondary objective if wanted. This can be helpful to subselect the solution space and create less alternative optimal solution. The secondary objective can be set to "mtf" to minimize the total flux, to simulate minimal enzyme usage of an organisms. If set to "opt_rxn" or "opt_ex", the secondary objective is picked as a random reaction or exchange reaction respectively everytime a fba is performed. This means that every individual of a population will select a different secondary reaction to optimize. The "sumex" option maximizes the secretion of products.

Value

Modified problem object according to the constraints and then solved with optimizeProb.

See Also

Organism-class, optimizeProb and sysBiolAlg

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
org <- Organism(Ec_core,deathrate=0.05,
               minweight=0.05,growtype="exponential") #initialize a organism
org@Fbasol <- optimizeLP(org)
Description

Structure of the S4 class `Organism` representing the organisms present in the environment.

Slots

- `lbnd` A numeric vector containing the lower bounds of the model structure.
- `ubnd` A numeric vector containing the upper bounds of the model structure.
- `type` A character vector containing the description of the organism.
- `medium` A character vector containing all exchange reactions of the organism.
- `lpobj` A sybil optimization object containing the linear programing problem.
- `fbasol` A list with the solutions of the flux balance analysis.
- `lyse` A boolean variable indicating if the organism should lyse after death.
- `feat` A list containing conditional features for the object (contains at the moment only biomass components for lysis).
- `deathrate` A numeric value giving the factor by which the biomass should be reduced in every iteration if no growth is possible (default (E.coli): 0.21 pg)
- `minweight` A numeric value giving the growth limit at which the organism dies. (default (E.coli): 0.083 pg)
- `growtype` A character vector giving the functional type for growth (linear or exponential).
- `kinetics` A list containing Km and v_max values for each reactions.
- `speed` A integer vector representing the speed by which bacterium is moving (given by cell per iteration).
- `cellarea` A numeric value indicating the surface that one organism occupies (default (E.coli): 4.42 \mu m^2)
- `maxweight` A numeric value giving the maximal dry weight of single organism (default (E.coli): 1.172 pg)
- `cellweight_mean` A numeric giving the mean of starting biomass (default (E.coli): 0.489 pg)
- `cellweight_sd` A numeric giving the standard derivation of starting biomass (default (E.coli): 0.132 pg)
- `model` Object of class sybil::modelorg containing the genome sclaee metabolic model
- `algo` Algorithm to be used during optimization (default fba)
- `rbiomass` Name of biomass reactions which is used for growth model (set automatically but needs input if objective is not biomass optimization)
- `limit_growth` If true then a upper bound on growth will be set, see maxweight (default: True).
- `coupling_constraints` List with coupling parameters.
- `predator` Name of organism which can kill this one.
Organism-constructor

Constructor of the S4 class Organism-class

Description
Constructor of the S4 class Organism-class

Usage
Organism(
  model,
  algo = "fba",
  ex = "EX_",
  ex_comp = NA,
  csuffix = "\\[c\\]",
  esuffix = "\\[e\\]",
  lyse = FALSE,
  feat = list(),
  typename = NA,
  setExInf = TRUE,
  setAllExInf = FALSE,
  coupling_constraints = list(),
  predator = "",
  ...
)

Arguments
model Object of class sybil::modelorg containing the genome scale metabolic model
algo A single character string giving the name of the algorithm to use. See SYBIL_SETTINGS
ex Identifier for exchange reactions
ex_comp Defining exchange reactions whose compounds should be added to the medium of the arena (default: all)
csuffix suffix for intern metabolites used by lysis function.
esuffix suffix for external metabolites used by lysis function.
lyse A boolean variable indicating if the organism should lyse after death.
feat A list containing conditional features for the object (contains at the moment only biomass components for lysis).
typename A string defining the name (set to model name in default case)
setExInf Enable if all lower bounds of exchange reaction which are set to zero (i.e. no uptake possible!) should be set to -infinity (default: true)
setAllExInf Enable if all lower bounds of exchange reaction should be set to -infinity (default: false)
plotAbundance

- coupling_constraints: List with coupling parameters.
- predator: Name of organism which can kill this one.
- ... Arguments of Organism-class

Value

Object of class Organism

plotAbundance  
Plot abundances of species

Description

The function plotAbundance takes a list of simulations and return a boxplot with species abundances.

Usage

plotAbundance(
  simlist,  
  time = c(NULL, NULL),  
  col = colpal3,  
  return_dat = F,  
  use_biomass = F
)

Arguments

- simlist: A list of simulations (eval objects).
- time: A vector with start and end time to be considered (default: total time)
- col: Vector with color that should be used
- return_dat: Should plain text mean abundances be returned? (default false)
- use_biomass: If enabled then biomass is used instead of cell number
plotCurves  
*Function for plotting the overall change as curves*

**Description**

The generic function `plotCurves` plots the growth curves and concentration changes of substances from simulation steps in an `Eval` object.

**Usage**

```r
plotCurves(
  object,
  medplot = object@mediac,
  retdata = F,
  remove = F,
  legend = F,
  graph = T
)
```

```r
## S4 method for signature 'Eval'
plotCurves(
  object,
  medplot = object@mediac,
  retdata = F,
  remove = F,
  legend = F,
  graph = T
)
```

**Arguments**

- `object` An object of class `Eval`.
- `medplot` A character vector giving the name of substances which should be plotted.
- `retdata` A boolean variable indicating if the data used to generate the plots should be returned.
- `remove` A boolean variable indicating if substances, which don’t change in their concentration should be removed from the plot.
- `legend` Boolean variable indicating if legend should be plotted
- `graph` True if graphic should be plotted.

**Details**

The parameter `retdata` can be used to access the data used for the returned plots to create own custom plots.
plotCurves2

Value

Returns two graphs in one plot: the growth curves and the curves of concentration changes. Optional
the data to generate the original plots can be returned.

See Also

Eval-class and Arena-class

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core,deathrate=0.05,
    minweight=0.05,growtype="exponential") # initialize a bacterium
arena <- Arena(n=20,m=20) # initialize the environment
arena <- addOrg(arena,bac,amount=10) # add 10 organisms
arena <- addSubs(arena,40) # add all possible substances
eval <- simEnv(arena,5)
plotCurves(eval)

plotCurves2

Function for plotting the overall change as curves with maximally dis-
tinct colors

Description

The generic function plotCurves2 plots the growth curves and concentration changes of the most
changing substances from simulation steps in an Eval object using maximally distinct colors.

Usage

plotCurves2(
    object,
    legendpos = "topleft",
    ignore = c("EX_h(e)", "EX_pi(e)", "EX_h2o(e)",
    num = 10,
    phencol = FALSE,
    biomcol = FALSE,
    dict = NULL,
    subs = list(),
    growthCurve = TRUE,
    subCurve = TRUE
)

## S4 method for signature 'Eval'
plotCurves2(
    object,
    legendpos = "topright",
    ignore = c("EX_h(e)", "EX_pi(e)", "EX_h2o(e)",

Arguments

- **object**: An object of class Eval.
- **legendpos**: A character variable declaring the position of the legend
- **ignore**: A list of character variables with substance names that should be omitted in the plot
- **num**: An integer defining the number of substrates to be plotted
- **phencol**: Boolean variable indicating whether phenotypes should be highlighted
- **biomcol**: A boolean indicating if biomass should be included in the growth curve
- **dict**: List defining new substance names. List entries are interpreted as old names and the list names as the new ones.
- **subs**: List of substance names. If empty, substances with highest variance will be used.
- **growthCurve**: True if growth curve should be shown (default TRUE)
- **subCurve**: True if substance curve should be shown (default TRUE)

Details

The parameter `retdata` can be used to access the data used for the returned plots to create own custom plots.

Value

Returns two graphs in one plot: the growth curves and the curves of concentration changes.

See Also

- **Eval-class** and **Arena-class**

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
eval <- simEnv(arena, 5)
plotCurves2(eval)
**plotFluxVar**

*Plot population flux variations*

**Description**

The function `plotFluxVar` takes a list of simulations and metabolites, returning a plot with metabolite fluxes for each species.

**Usage**

`plotFluxVar(simlist, metsel)`

**Arguments**

- **simlist** A list of simulations (eval objects).
- **metsel** A vector with the name of exchange reactions of interest

**plotFVA**

*Function to plot population level minimum and maximum flux from alternative optimal solutions obtained by FVA*

**Description**

The generic function `plotFVA` plots population level flux results obtained from function `fluxVarSim`.

**Usage**

`plotFVA(fvares, mediac)`

**Arguments**

- **fvares** results of FVA results to plot, obtained from function `fluxVarSim`
- **mediac** List with substances.

**Details**

Returns ggplot objects
plotGrowthCurve  
*Plot growth curve for several simulations*

**Description**

The function `plotGrowthCurve` takes a list of simulations and plots the time course of species with standard deviation.

**Usage**

```r
plotGrowthCurve(
  simlist,
  time = c(NULL, NULL),
  ret_data = FALSE,
  use_biomass = F,
  specs = NULL
)
```

**Arguments**

- `simlist`: A list of simulations (eval objects).
- `time`: Vector with two entries defining start and end time.
- `ret_data`: Set true if data should be returned.
- `use_biomass`: If enabled then biomass is used instead of cell number.
- `specs`: List of species for which a growth curve should be shown (default: all).

plotInterNum  
*Plot number of variation in number of interactions for several simulations*

**Description**

The function `plotInterNum` takes a list of simulations and plots the time course of the number of metabolic interactions with standard deviation.

**Usage**

```r
plotInterNum(simlist, title = "Variation in number of interactions", size = 1)
```

**Arguments**

- `simlist`: A list of simulations (eval objects).
- `title`: Title of the plot.
- `size`: A scaling factor for plot text and line size.
plotPhenCurve  

Plot growth curve for several simulations

Description

The function `plotPhenCurve` takes a list of simulations and plots the time course of species with standard deviation.

Usage

```r
plotPhenCurve(
  simlist,
  subs,
  phens = NULL,
  time = c(NULL, NULL),
  cluster = TRUE,
  inAll = TRUE,
  col = colpal3,
  with_gc = FALSE,
  return_dat = FALSE
)
```

Arguments

- **simlist**: A list of simulations (eval objects).
- **subs**: A vector of substance names that are used for phenotype clustering.
- **phens**: If `phencurve` is given then `phens` specifies the phenotypes which should be plotted again.
- **time**: Vector with two entries defining start and end time
- **cluster**: True if phenotypes should be clustered/condensed.
- **inAll**: True if only phenotypes which occur in all replicates should be considered
- **col**: Vector with color that should be used
- **with_gc**: True if growth curve of organisms should be included
- **return_dat**: Should data be returned? (default false)
### plotPhenNum

**Plot number of phenotypes curve for several simulations**

**Description**

The function `plotPhenNum` takes a list of simulations and plots the time course of the number of phenotypes with standard deviation.

**Usage**

```r
plotPhenNum(simlist, title = "Phenotype number variation", size = 1)
```

**Arguments**

- `simlist`: A list of simulations (eval objects).
- `title`: Title of the plot
- `size`: A scaling factor for plot text and line size

### plotReaActivity

**Function to plot reaction activity for every species**

**Description**

The generic function `plotReaActivity` displays the usage of reactions for all species.

**Usage**

```r
plotReaActivity(
    simlist, 
    reactions = list(),
    spec_list = NULL,
    ret_data = FALSE
)
```

**Arguments**

- `simlist`: An object of class Eval or a list with objects of class Eval.
- `reactions`: List of reaction names
- `spec_list`: List of species names to be considered (default all)
- `ret_data`: Set true if data should be returned

**Details**

Returns ggplot objects
**plotShadowCost**  

*Function to plot substance shadow costs for a specie*

---

**Description**

The generic function `plotShadowCost` plots substances have the highest impact on further growth (shadow cost < 0)

**Usage**

```r
plotShadowCost(
  object,
  spec_nr = 1,
  sub_nr = 10,
  cutoff = -1,
  noplot = FALSE,
  useNames = FALSE
)
```

```r
## S4 method for signature 'Eval'
plotShadowCost(
  object,
  spec_nr = 1,
  sub_nr = 10,
  cutoff = -1,
  noplot = FALSE,
  useNames = FALSE
)
```

**Arguments**

- `object`  
  An object of class Eval.
- `spec_nr`  
  Number of the specie
- `sub_nr`  
  Maximal number of substances to be show
- `cutoff`  
  Shadow costs should be smaller than cutoff
- `noplot`  
  Do not plot
- `useNames`  
  Use substance names instead of ids

**Details**

Returns ggplot objects
plotSpecActivity

Function to plot substance usage for every species

Description

The generic function plotSpecActivity displays the input/output substances with the highest variance (could also be defined manually) for all species.

Usage

```
plotSpecActivity(
  simlist,
  subs = list(),
  var_nr = 10,
  spec_list = NULL,
  ret_data = FALSE,
  useNames = FALSE,
  rm_unused = TRUE,
  cutoff = 1e-06
)
```

Arguments

- `simlist`: An object of class Eval or a list with objects of class Eval.
- `subs`: List of substance names
- `var_nr`: Number of most varying substances to be used (if subs is not specified)
- `spec_list`: List of species names to be considered (default all)
- `ret_data`: Set true if data should be returned
- `useNames`: Use substance names instead of ids
- `rm_unused`: Remove substances which do not change from plot
- `cutoff`: Minimum value for fluxes to be considered

Details

Returns ggplot objects
plotSubCurve  

Plot substance curve for several simulations

Description

The function `plotSubCurve` takes a list of simulations and plots the time course of substances with standard deviation.

Usage

```r
plotSubCurve(
  simlist,
  mediac = NULL,
  time = c(NULL, NULL),
  scol = NULL,
  unit = "mmol",
  ret_data = FALSE,
  num_var = 10,
  useNames = FALSE
)
```

Arguments

- `simlist` A list of simulations (eval objects).
- `mediac` A vector of substances (if not specified most varying substances will be taken.)
- `time` Vector with two entries defining start and end time.
- `scol` Vector with colors that should be used.
- `unit` Unit for the substances which should be used for plotting (default: mmol)
- `ret_data` Set true if data should be returned
- `num_var` Number of varying substances to be shown (if mediac is not specified)
- `useNames` Use substance names instead of ids

Value

list of three ggplot object for further formatting
**plotSubDist**

*Function to overview the spatial distribution of a substance over time.*

**Description**

The generic function `plotSubDist` returns a plot for every time step which shows the substance concentration in the environment.

**Usage**

```r
plotSubDist(object, sub, times = NULL)
```

```r
## S4 method for signature 'Eval'
plotSubDist(object, sub, times = NULL)
```

**Arguments**

- `object`: An object of class Eval.
- `sub`: Name of a substance.
- `times`: Time points to be considered.

**Details**

Returns a plot with

**plotSubDist2**

*Function to overview the spatial distribution of a substance over time.*

**Description**

The generic function `plotSubDist2` returns a plot for every time step which shows the substance concentration in the environment.

**Usage**

```r
plotSubDist2(object, sub, times = NULL)
```

```r
## S4 method for signature 'Eval'
plotSubDist2(object, sub, times = NULL)
```

**Arguments**

- `object`: An object of class Eval.
- `sub`: Name of a substance.
- `times`: Time points to be considered.
plotSubUsage

Details

Returns a plot with

plotSubUsage

Function to plot usage of substances species wise

Description

The generic function plotSubUsage displays for given substances the quantities of absorption and production for each species.

Usage

plotSubUsage(simlist, subs = vector(), cutoff = 0.01, ret_data = FALSE)

Arguments

- simlist: An object of class Eval or a list with objects of class Eval.
- subs: List of substance names.
- cutoff: Total values below cutoff will be dismissed.
- ret_data: Set true if data should be returned.

Details

Returns ggplot objects.

plotSubVar

Plot substance variations

Description

The function plotSubVar takes a list of simulations and returns a barplot with most varying substances.

Usage

plotSubVar(simlist, metsel)

Arguments

- simlist: A list of simulations (eval objects).
- metsel: A vector with the name of exchange reactions of interest.
**plotTotFlux**  
*Function for plotting the overall change in reaction activity*

**Description**

The generic function `plotTotFlux` plots the time course of reactions with high variation in activity for an `Eval` object.

**Usage**

```r
plotTotFlux(object, legendpos = "topright", num = 20)
```

```r
## S4 method for signature 'Eval'
plotTotFlux(object, legendpos = "topright", num = 20)
```

**Arguments**

- `object`  
  An object of class `Eval`.  
- `legendpos`  
  A character variable declaring the position of the legend.  
- `num`  
  An integer defining the number of substrates to be plot.

**Examples**

```r
data(Ec_core, envir = environment())  # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate = 0.05,  
           minweight = 0.05, growtype = "exponential")  # initialize a bacterium
arena <- Arena(n = 20, m = 20)  # initialize the environment
arena <- addOrg(arena, bac, amount = 10)  # add 10 organisms
arena <- addSubs(arena, 40)  # add all possible substances
eval <- simEnv(arena, 5)
plotTotFlux(eval)
```

---

**readMATmod**  
*Read matlab model*

**Description**

The generic function `readMATmod` imports matlab cobra models into sybil model files.

**Usage**

```r
readMATmod(file)
```

**Arguments**

- `file`  
  Full path to matlab model file.
redEval

Details
Returns sybil model object (time needed: bacterial model ~ 10s, recon2 ~ 60s)

redEval Function for reducing the size of an Eval object by collapsing the medium concentrations

Description
The generic function redEval reduces the object size of an Eval object.

Usage
redEval(object, time = "all")

## S4 method for signature 'Eval'
redEval(object, time = 1:length(object@medlist))

Arguments
object An object of class Eval.
time A number giving the simulation step of interest.

Details
The function redEval can be used to reduce the size of an Eval object from a simulation step.

Value
Returns an object of class Arena containing the organisms and substance conditions in simulation step time.

See Also
Eval-class and Arena-class

Examples
data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
bac <- Bac(Ec_core,deathrate=0.05,
        minweight=0.05,growtype="exponential") #initialize a bacterium
arena <- Arena(n=20,m=20) #initialize the environment
arena <- addOrg(arena,bac,amount=10) #add 10 organisms
arena <- addSubs(arena,40) #add all possible substances
eval <- simEnv(arena,5)
eval_reduce <- redEval(eval,5)
reset_screen

Description

The function `reset_screen` set plotting window to default.

Usage

```r
reset_screen()
```

rmSubs

Description

The generic function `rmSubs` removes all amounts of substances available in the arena for given compounds.

Usage

```r
rmSubs(object, mediac)
```

Arguments

- **object**: An object of class Arena.
- **mediac**: A character vector giving the names of substances, which should be added to the environment (the default takes all possible substances).

selPheno

Function for selecting phenotypes which occurred on the arena from specific iterations and species.

Description

The generic function `selPheno` selects phenotypes from specific simulation step in an `Eval` object.
Usage

selPheno(object, time, type, reduce = F)

Arguments

object An object of class Eval.
time A numeric vector giving the simulation steps which should be plotted.
type A names indicating the species of interest in the arena.
reduce A boolean variable indicating if the resulting matrix should be reduced.

Details

The phenotypes are defined by flux through exchange reactions, which indicate potential differential substrate usages.

Value

Returns a matrix with the substrate usage and the number of individuals using the phenotype.

See Also

Eval-class and getPhenoMat

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") #initialize a bacterium
arena <- Arena(n=20, m=20) #initialize the environment
arena <- addOrg(arena, bac, amount=10) #add 10 organisms
arena <- addSubs(arena, 40) #add all possible substances
eval <- simEnv(arena, 5)
setPheno(eval, time=5, type='ecoli_core_model', reduce=TRUE)

setKinetics

Function to set Michaelis-Menten kinetics for uptake of a substance

Description

The generic function setKinetics provides kinetics for exchange reactions.
Usage

setKinetics(object, exchangeR, Km, vmax)

## S4 method for signature 'Organism'
setKinetics(object, exchangeR, Km, vmax)

Arguments

object  An object of class Organisms.
exchangeR  Name of an exchange reaction
Km  Parameter Michaelis-Menten-Kinetics (in mM)
vmax  Parameter Michaelis-Menten-Kinetics (in mmol/(g*h))

sihumi_test  Multi-species test data set

Description

Data contains results of a simulation with 8 gut bacteria for 10 time steps

Usage

data(sihumi_test)

Format

An object of class eval

simBac  Function for one simulation iteration for objects of Bac class

Description

The generic function simBac implements all necessary functions for the individuals to update the complete environment.
Usage

```r
simBac(
  object,
  arena,
  j,
  sublb,
  bacnum,
  sec_obj = "none",
  cutoff = 1e-06,
  pcut = 1e-06,
  with_shadow = FALSE
)
```

## S4 method for signature 'Bac'

```r
simBac(
  object,
  arena,
  j,
  sublb,
  bacnum,
  sec_obj = "none",
  cutoff = 1e-06,
  pcut = 1e-06,
  with_shadow = FALSE
)
```

Arguments

- **object** An object of class Bac.
- **arena** An object of class Arena defining the environment.
- **j** The index of the organism of interest in orgdat.
- **sublb** A vector containing the substance concentrations in the current position of the individual of interest.
- **bacnum** integer indicating the number of bacteria individuals per gridcell
- **sec_obj** character giving the secondary objective for a bi-level LP if wanted.
- **cutoff** value used to define numeric accuracy.
- **pcut** A number giving the cutoff value by which value of objective function is considered greater than 0.
- **with_shadow** True if shadow cost should be stores (default off).

Details

Bacterial individuals undergo step by step the following procedures: First the individuals are constrained with constrain to the substrate environment, then flux balance analysis is computed with optimizeLP, after this the substrate concentrations are updated with consume, then the bacterial
growth is implemented with growth, the potential new phenotypes are added with checkPhen, finally the additional and conditional functions lysis, move or chemotaxis are performed. In case of many compounds in the vector of chemotaxis, the change of the position takes place by the order of the compounds in the vector of chemotaxis. Can be used as a wrapper for all important bacterial functions in a function similar to simEnv.

Value

Returns the updated environment of the population parameter with all new positions of individuals on the grid and all new substrate concentrations.

See Also

Bac-class, Arena-class, simEnv, constrain, optimizeLP, consume, growth, checkPhen, lysis, move and chemotaxis

Examples

NULL

simBac_par Function for one simulation iteration for objects of Bac class

Description

The generic function simBac_par implements all necessary functions for the individuals to update the complete environment.

Usage

simBac_par(
  object,
  arena,
  j,
  sublb,
  bacnum,
  lpobject,
  sec_obj = "none",
  cutoff = 1e-06,
  with_shadow = FALSE
)

## S4 method for signature 'Bac'
simBac_par(
  object,
  arena,
  j,
  sublb,
bacnum,
lpobject,
sec_obj = "none",
cutoff = 1e-06,
with_shadow = FALSE
)

Arguments

object  An object of class Bac.
arena   An object of class Arena defining the environment.
j       The index of the organism of interest in orgdat.
sublb   A vector containing the substance concentrations in the current position of the
        individual of interest.
bacnum  integer indicating the number of bacteria individuals per gridcell
lpobject linear programming object (copy of organism@lpobj) that have to be a deep copy
        in parallel due to pointer use in sybil.
sec_obj character giving the secondary objective for a bi-level LP if wanted.
cutoff  value used to define numeric accuracy
with_shadow True if shadow cost should be stores (default off).

Value

Returns the updated environment of the population parameter with all new positions of individu-
als on the grid and all new substrate concentrations.

---

**simEnv**

*Main function for simulating all processes in the environment*

Description

The generic function **simEnv** for a simple simulation of the environment.

Usage

```r
simEnv(
  object,
  time,
  lrw = NULL,
  continue = FALSE,
  reduce = FALSE,
  diffusion = TRUE,
  diff_par = FALSE,
  cl_size = 2,
  sec_obj = "none",
)```

```r
simEnv = 1e-06,
cutoff = 1e-06,
with_shadow = TRUE,
verbose = TRUE
)

## S4 method for signature 'Arena'
simEnv(
  object,
  time,
  lrw = NULL,
  continue = FALSE,
  reduce = FALSE,
  diffusion = TRUE,
  diff_par = FALSE,
  cl_size = 2,
  sec_obj = "none",
  cutoff = 1e-06,
  pcut = 1e-06,
  with_shadow = TRUE,
  verbose = TRUE
)

Arguments

- **object**: An object of class Arena or Eval.
- **time**: A number giving the number of iterations to perform for the simulation.
- **lrw**: A numeric value needed by solver to estimate array size (by default lrw is estimated in the simEnv() by the function estimate_lrw()).
- **continue**: A boolean indicating whether the simulation should be continued or restarted.
- **reduce**: A boolean indicating if the resulting Eval object should be reduced.
- **diffusion**: True if diffusion should be done (default on).
- **diff_par**: True if diffusion should be run in parallel (default off).
- **cl_size**: If diff_par is true then cl_size defines the number of cores to be used in parallelized diffusion.
- **sec_obj**: character giving the secondary objective for a bi-level LP if wanted. Use "mtf" for minimizing total flux, "opt_rxn" for optimizing a random reaction, "opt_ex" for optimizing a random exchange reaction, and "sumex" for optimizing the sum of all exchange fluxes.
- **cutoff**: value used to define numeric accuracy.
- **pcut**: A number giving the cutoff value by which value of objective function is considered greater than 0.
- **with_shadow**: True if shadow cost should be stored.
- **verbose**: Set to false if no status messages should be printed.
```
Details

The returned object itself can be used for a subsequent simulation, due to the inheritance between Eval and Arena. The parameter for sec_obj can be used to optimize a bi-level LP with a secondary objective if wanted. This can be helpful to subselect the solution space and create less alternative optimal solution. The secondary objective can be set to "mtf" to minimize the total flux, to simulate minimal enzyme usage of an organism. If set to "opt_rxn" or "opt_ex", the secondary objective is picked as a random reaction or exchange reaction respectively every time a fba is performed. This means that every individual of a population will select a different secondary reaction to optimize. The "sumex" option maximizes the secretion of products.

Value

Returns an object of class Eval which can be used for subsequent analysis steps.

See Also

Arena-class and Eval-class

Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05, minweight=0.05, growtype="exponential") # initialize a bacterium
arena <- Arena(n=20, m=20) # initialize the environment
arena <- addOrg(arena, bac, amount=10) # add 10 organisms
arena <- addSubs(arena, 40) # add all possible substances
eval <- simEnv(arena, 5)
## S4 method for signature 'Arena'

```r
code
```

### Arguments

- **object**: An object of class Arena or Eval.
- **time**: A number giving the number of iterations to perform for the simulation.
- **lrw**: A numeric value needed by solver to estimate array size (by default lrw is estimated in the simEnv() by the function estimate_lrw()).
- **continue**: A boolean indicating whether the simulation should be continued or restarted.
- **reduce**: A boolean indicating if the resulting Eval object should be reduced.
- **cluster_size**: Number of cpu cores to be used.
- **diffusion**: True if diffusion should be done (default on).
- **sec_obj**: character giving the secondary objective for a bi-level LP if wanted.
- **cutoff**: value used to define numeric accuracy.
- **with_shadow**: True if shadow cost should be stores (default off).
- **verbose**: Set to false if no status messages should be printed.

### Details

The returned object itself can be used for a subsequent simulation, due to the inheritance between Eval and Arena.

### Value

Returns an object of class Eval which can be used for subsequent analysis steps.

### See Also

- Arena-class
- Eval-class
Examples

data(Ec_core, envir = environment()) # get Escherichia coli core metabolic model
bac <- Bac(Ec_core, deathrate=0.05,
           minweight=0.05,growtype="exponential") # initialize a bacterium
arena <- Arena(n=20,m=20) # initialize the environment
arena <- addOrg(arena,bac,amount=10) # add 10 organisms
arena <- addSubs(arena,40) # add all possible substances
eval <- simEnv(arena,5)

---

**simHum**

*Function for one simulation iteration for objects of Human class*

Description

The generic function `simHum` implements all necessary functions for the individuals to update the complete environment.

Usage

```r
simHum(
  object,
  arena,
  j,
  sublb,
  bacnum,
  sec_obj = "none",
  cutoff = 1e-06,
  pcut = 1e-06,
  with_shadow = FALSE
)
```

## S4 method for signature 'Human'

```r
simHum(
  object,
  arena,
  j,
  sublb,
  bacnum,
  sec_obj = "none",
  cutoff = 1e-06,
  pcut = 1e-06,
  with_shadow = FALSE
)
```
Arguments

- **object**: An object of class Human.
- **arena**: An object of class Arena defining the environment.
- **j**: The number of the iteration of interest.
- **sublb**: A vector containing the substance concentrations in the current position of the individual of interest.
- **bacnum**: integer indicating the number of bacteria individuals per gridcell.
- **sec_obj**: character giving the secondary objective for a bi-level LP if wanted.
- **cutoff**: value used to define numeric accuracy.
- **pcut**: A number giving the cutoff value by which value of objective function is considered greater than 0.
- **with_shadow**: True if shadow cost should be stores (default off).

Details

Human cell individuals undergo the step by step the following procedures: First the individuals are constrained with `constrain` to the substrate environment, then flux balance analysis is computed with `optimizeLP`, after this the substrate concentrations are updated with `consume`, then the cell growth is implemented with `cellgrowth`, the potential new phenotypes are added with `checkPhen`, finally the conditional function `lysis` is performed. Can be used as a wrapper for all important cell functions in a function similar to `simEnv`.

Value

Returns the updated environment of the arena parameter with all new positions of individuals on the grid and all new substrate concentrations.

See Also

- Human-class, Arena-class, simEnv, constrain, optimizeLP, consume, cellgrowth, checkPhen and lysis

Examples

```r
NULL
```

---

**statPheno**

*Function for investigating a specific phenotype of an organism*

Description

The generic function `statPheno` provides statistical and visual information about a certain phenotype.
stirEnv

Usage

statPheno(object, type_nr = 1, phenotype_nr, dict = NULL)

## S4 method for signature 'Eval'
statPheno(object, type_nr = 1, phenotype_nr, dict = NULL)

Arguments

object An object of class Eval.
type_nr A number indicating the Organism type of the phenotype to be investigated (from orgdat)
phenotype_nr A number indicating the phenotype to be investigated (from orgdat)
dict A character vector of all substance IDs with names that should be used instead of possibly cryptic IDs

Details

The phenotypes are defined by flux through exchange reactions, which indicate potential differential substrate usages.

See Also

Eval-class

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
bac <- Bac(Ec_core,deathrate=0.05, minweight=0.05,growtype="exponential") #initialize a bacterium
arena <- Arena(n=20,m=20) #initialize the environment
arena <- addOrg(arena,bac,amount=10) #add 10 organisms
arena <- addSubs(arena,40) #add all possible substances
eval <- simEnv(arena,5)
statPheno(eval, type_nr=1, phenotype_nr=2)

stirEnv

Function for stirring/mixing the complete environment

Description

The generic function stirEnv simulates the event of mixing all substrates and organisms in the environment.

Usage

stirEnv(object, sublb)

## S4 method for signature 'Arena'
stirEnv(object, sublb)
Substance-class

Arguments

object An object of class Arena.
sublb A matrix with the substrate concentration for every individual in the environment based on their x and y position.

Details

The stirring is implemented as a random permutation of organism positions and the equalization of all substrate concentrations.

Value

Returns the substrate concentration for every individual in the environment with substrates as well as x and y positions as columns and rows for each organism.

See Also

Arena-class and getSublb

Examples

data(Ec_core, envir = environment()) #get Escherichia coli core metabolic model
bac <- Bac(Ec_core,deathrate=0.05,
          minweight=0.05,growthtype="exponential") #initialize a bacterium
arena <- Arena(n=20,m=20) #initialize the environment
arena <- addOrg(arena,bac,amount=10) #add 10 organisms
arena <- addSubs(arena,40) #add all possible substances
sublb <- getSublb(arena)
stirEnv(arena,sublb)

---

Substance-class Structure of the S4 class "Substance"

Description

Structure of the S4 class Substance representing substances in the environment which can be produced or consumed.

Slots

smax A number representing the start concentration of the substance for each grid cell in the environment.
diffmat A sparse matrix containing all concentrations of the substance in the environment.
name A character vector representing the name of the substance.
id A character vector representing the identifier of the substance.
difunc A character vector ("pde", "cpp" or "r") describing the function for diffusion.
**difspeed** A number indicating the diffusion rate (given by cm^2/h). Default is set to glucose diffusion in a aqueous solution (6.7e-6 cm^2/s * 3600 s/h = 0.02412 cm^2/h).

**advspeed** A number indicating the advection rate in x direction (given by cm/h).

**diffgeometry** Diffusion coefficient defined on all grid cells (initially set by constructor).

**pde** Choose diffusion transport reaction to be used (default is diffusion only)

**boundS** A number defining the attached amount of substance at the boundary (Warning: boundary-function must be set in pde!)

---

**Substance-constructor**  
*Constructor of the S4 class Substance*

---

**Description**

The constructor to get a new object of class Substance

**Usage**

```r
Substance(
  n,
  m,
  smax,
  gridgeometry,
  difspeed = 0.02412,
  advspeed = 0,
  occupyM,
  Dgrid = NULL,
  Vgrid = NULL,
  diffmat = NULL,
  template = FALSE,
  ...
)
```

**Arguments**

- **n**  
  A number giving the horizontal size of the environment.

- **m**  
  A number giving the vertical size of the environment.

- **smax**  
  A number representing the start concentration of the substance for each grid cell in the environment.

- **gridgeometry**  
  A list containing grid geometry parameter

- **difspeed**  
  A number indicating the diffusion speed in x and y direction (given by cm^2/h). For more complex setup define Dgrid.

- **advspeed**  
  A number indicating the advection speed in x direction (given by cm/h). For more complex setup define Vgrid.

- **occupyM**  
  A matrix indicating grid cells that are obstacles
Dgrid  A matrix indicating the diffusion speed in x and y direction (given by cm^2/h).
Vgrid  A number indicating the advection speed in x direction (given by cm/h).
diffmat A matrix with spatial distributed initial concentrations (unit in fmol) (if not set, a homogenous matrix using smax is created)
template True if diffmat matrix should be used as tempalte only (will be multiplied with smax to obtain cocentrations)
... Arguments of Substance-class

Value

Object of class Substance

---

unit_conversion Function for unit conversion

Description

The generic function unit_conversion converts units for e.g. substance concentrations

Usage

unit_conversion(object, unit)

## S4 method for signature 'Arena'
unit_conversion(object, unit)

Arguments

object An object of class Arena or Eval.
unit Unit to be converted to fmol/cell

Value

Conversion factor
**Description**

Helper function to get upper error bounds in plotting

**Usage**

```r
usd(y)
```

**Arguments**

- `y` Vector with numbers

---

**Computer standard deviation upper bound**
Index

*Topic datasets
  colpal1, 20
  colpal2, 21
  colpal3, 21
  colpal4, 22
  colpal5, 22
  colpal6, 23
  sihumi_test, 76

addDefaultMed, 4
addDefaultMed, Arena-method (addDefaultMed), 4
addEssentialMed, 4
addEssentialMed, Arena-method (addEssentialMed), 4
addEval, 5
addEval, Eval-method (addEval), 5
addOrg, 6, 15
addOrg, Arena-method (addOrg), 6
addSubs, 7, 16, 40
addSubs, Arena-method (addSubs), 7
Arena (Arena-constructor), 10
Arena-class, 9, 10
Arena-constructor, 10

Bac (Bac-Constructor), 11
Bac-class, 11, 11
Bac-Constructor, 11
BacArena, 12

cellgrowth, 12
cellgrowth, Human-method (cellgrowth), 12
changeDiff, 13
changeDiff, Arena-method (changeDiff), 13
changeFobj, 14
changeFobj, Human-method (changeFobj), 14
changeOrg, 15
changeOrg, Arena-method (changeOrg), 15
changeSub, 9, 13, 16, 16, 26
changeSub, Arena-method (changeSub), 16

checkCorr, 17
checkCorr, Eval-method (checkCorr), 17
checkPhen, 18, 44
checkPhen, Arena-method (checkPhen), 18
checkPhen_par, 19
checkPhen_par, Arena-method (checkPhen_par), 19
chemotaxis, 19
chemotaxis, Bac-method (chemotaxis), 19
colpal1, 20
colpal2, 21
colpal3, 21
colpal4, 22
colpal5, 22
colpal6, 23
constrain, 23
constrain, Organism-method (constrain), 23
consume, 24
consume, Organism-method (consume), 24
createGradient, 25
createGradient, Arena-method (createGradient), 25
dat2mat, 26
dat2mat, Arena-method (dat2mat), 26
diffuse, 27
diffuse, Arena-method (diffuse), 27
diffuse_par, 29
diffuse_par, Arena-method (diffuse_par), 29
diffusePDE, 28, 29
diffusePDE, Substance-method (diffusePDE), 28
diffuseR, 28, 28
diffuseR, Substance-method (diffuseR), 28
emptyHood, 20, 30, 53
emptyHood, Organism-method (emptyHood), 30
INDEX

Eval (Eval-constructor), 31
Eval-class, 31
Eval-constructor, 31
evalArena, 32
evalArena, Eval-method (evalArena), 32
extractMed, 33
extractMed, Eval-method (extractMed), 33
findFeeding, 34
findFeeding, Eval-method (findFeeding), 34
findFeeding2, 35
findFeeding2, Eval-method (findFeeding2), 35
findFeeding3, 36
findFeeding3, Eval-method (findFeeding3), 36
findFeeding3rep, 37
findInArena, 37
findInArena, Arena-method (findInArena), 37
findrBiomass, 38
findRxnFlux, 39
findRxnFlux, Eval-method (findRxnFlux), 39
flushSubs, 40
flushSubs, Arena-method (flushSubs), 40
fluxVarSim, 40
fluxVarSim, Eval-method (fluxVarSim), 40
getArena, 41
getArena, Eval-method (getArena), 41
getCorrM, 17, 42
getCorrM, Eval-method (getCorrM), 42
getPhenoMat, 43, 53, 75
getPhenoMat, Eval-method (getPhenoMat), 43
getPhenotype, 18, 43, 44
getPhenotype, Organism-method (getPhenotype), 44
getSubHist, 45
getSubHist, Eval-method (getSubHist), 45
getSublb, 27, 45, 86
getSublb, Arena-method (getSublb), 45
getVarSubs, 46
getVarSubs, Eval-method (getVarSubs), 46
growExp, 13, 47, 49
growExp, Organism-method (growExp), 47
growLin, 13, 47, 49
growLin, Organism-method (growLin), 47
growth, 48
growth, Bac-method (growth), 48
growth_par, 49
growth_par, Bac-method (growth_par), 49
Human (Human-constructor), 50
Human-class, 50
Human-constructor, 50
lsd, 51
lysis, 51
lysis, Organism-method (lysis), 51
minePheno, 44, 52
minePheno, Eval-method (minePheno), 52
move, 53
move, Organism-method (move), 53
NemptyHood, 54
NemptyHood, Organism-method (NemptyHood), 54
openArena, 55
optimizeLP, 14, 47, 48, 52, 55
optimizeLP, Organism-method (optimizeLP), 55
optimizeProb, 56
Organism, 11, 50
Organism (Organism-constructor), 58
Organism-class, 57, 58
Organism-constructor, 58
plotAbundance, 59
plotCurves, 60
plotCurves, Eval-method (plotCurves), 60
plotCurves2, 61
plotCurves2, Eval-method (plotCurves2), 61
plotFluxVar, 63
plotFVA, 63
plotGrowthCurve, 64
plotInterNum, 64
plotPhenCurve, 65
plotPhenNum, 66
plotReaActivity, 66
plotShadowCost, 67
plotShadowCost, Eval-method (plotShadowCost), 67
plotSpecActivity, 68
plotSubCurve, 69
plotSubDist, 70
plotSubDist,Eval-method (plotSubDist), 70
plotSubDist2, 70
plotSubDist2,Eval-method (plotSubDist2), 70
plotSubUsage, 71
plotSubVar, 71
plotTotFlux, 72
plotTotFlux,Eval-method (plotTotFlux), 72
readMATmod, 72
redEval, 73
redEval,Eval-method (redEval), 73
reset_screen, 74
rmSubs, 74
rmSubs,Arena-method (rmSubs), 74
selPheno, 74
selPheno,Eval-method (selPheno), 74
setKinetics, 75
setKinetics,Organism-method (setKinetics), 75
sihumi_test, 76
simBac, 76
simBac,Bac-method (simBac), 76
simBac_par, 78
simBac_par,Bac-method (simBac_par), 78
simEnv, 39, 41, 78, 79, 84
simEnv,Arena-method (simEnv), 79
simEnv_par, 81
simEnv_par,Arena-method (simEnv_par), 81
simHum, 83
simHum,Human-method (simHum), 83
statPheno, 84
statPheno,Eval-method (statPheno), 84
stirEnv, 85
stirEnv,Arena-method (stirEnv), 85
Substance (Substance-constructor), 87
Substance-class, 86
Substance-constructor, 87
SYBIL_SETTINGS, 58
sysBiolAlg, 56

unit_conversion, 88
unit_conversion,Arena-method (unit_conversion), 88
usd, 89