Package 'ecosim'

February 13, 2017

Type Package

Title Toolbox for Aquatic Ecosystem Modeling

Version 1.3

Date 2017-02-13

Author Peter Reichert

Maintainer Peter Reichert <peter.reichert@eawag.ch>

Description Classes and methods for implementing aquatic ecosystem models, for running these models, and for visualizing their results.

License GPL (>= 2)

Depends methods, deSolve, stoichcalc

NeedsCompilation no

Repository CRAN

Date/Publication 2017-02-13 23:35:47

R topics documented:

ecosim-package *Toolbox for Aquatic Ecosystem Modeling*

Description

Classes and methods for implementing aquatic ecosystem models, for running these models, and for visualizing their results.

Models are built by constructing objects of the classes

process-class,
reactor-class,
link-class,
system-class.

A transformation processes (process-class) is defined by a process rate (expression describing the dependence of the rate on substance or organism concentrations and external influence factors) and stoichiometric coefficients that describe how the rate affects different substances or organisms. It is recommended to calculate the stoichiometric coefficients with the function calc.stoich.coef of the package stoichcalc from substance and organism compositions. The output of this function can directly be used for the process definition. A reactor (reactor-class) describes a well-mixed compartment of the environment (or of a laboratory system). For each reactor, inflow, outflow, substance and organism input and transformation processes can be defined. A link (link-class) describes advective and/or diffusive exchange of substances and/or organisms between well-mixed reactors. Finally, a system (system-class) consists of a single reactor or a set of isolated or linked reactors and can be used to describe a community or meta-community model of an ecosystem and the biogeochemical cycles.

Once a model is described by an object of the class system (system-class), simulations can be performed using the member function

calcres,

This function integrates the system of ordinary differential equations numerically using the function ode of the package deSolve and produces time series of the volumes and substance and organisms concentrations as a R matrix. The results can be visualized with arbitrary R functions or a summary of all results can be produced with the function

plotres.

To propagate stochasticity and uncertainty to the results, stochastic parameter time series can be generated with the function

randou

and parameter samples can be sampled with the function

randnorm

to get a sample from the predictive distribution by Monte Carlo simulation.

Details

Package:	ecosim
Type:	Package
Version:	1.3
Date:	2017-02-13
License:	GPL (>= 2)
Depends:	deSolve, stoichcalc

Note

The following demos are available:

lakemodel_simple
lakemodel_intermediate
lakemodel_complex
rivermodel_simple
rivermodel_complex

Author(s)

Peter Reichert

Maintainer: Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci. Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

Soetaert, K., Petzoldt, T., and Woodrow Setzer, R. Solving differential equations in R: Package deSolve. Journal of Statistical Software, 33(9), 2010.

Soetaert, K., Cash, J., and Mazzia, F. Solving Differential Equations in R. Springer, Heidelberg, Germany. 2012.

See Also

deSolve, stoichcalc,

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3

# Definition of transformation processes:
# =======================================

# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))            # gDM/gDM

# Growth of zooplankton:
# ----------------------
```

```
gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),              # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))      # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))            # gDM/gDM

# Definition of reactor:
# ======================

# Epilimnion:
# -----------

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),     # gP/m3
                               C.ALG  = expression(C.ALG.ini),      # gDM/m3
                               C.ZOO  = expression(C.ZOO.ini)),     # gDM/m3
       inflow           = expression(Q.in*86400),                   # m3/d
       inflow.conc      = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0,
                               C.ZOO  = 0),
       outflow          = expression(Q.in*86400),
       processes        = list(gro.ALG,death.ALG,gro.ZOO,death.ZOO))

# Definition of system:
# =====================

# Lake system:
# ------------

system <- new(Class    = "system",
              name     = "Lake",
              reactors = list(epilimnion),
              param    = param,
              t.out    = seq(0,365,by=1))

# Perform simulation:
# ===================

res <- calcres(system)
```

```
# Plot results:
# =============

plotres(res)              # plot to screen

plotres(res,file="ecosim_example_plot1.pdf")  # plot to pdf file

plotres(res, colnames=c("C.ALG", "C.ZOO"))  # plot selected variables

plotres(res, colnames=list("C.HPO4",c("C.ALG", "C.ZOO")))

plotres(res[1:100,], colnames=list("C.HPO4",c("C.ALG", "C.ZOO"))) # plot selected time steps

plotres(res      = res,    # plot to pdf file
        colnames = list("C.HPO4",c("C.ALG","C.ZOO")),
        file     = "ecosim_example_plot2.pdf",
        width    = 8,
        height   = 4)
```

`calcres` *Performs a Simulation of the Model Passed as the Argument*

Description

Calculates a dynamic solution of the model defined by the argument and returns a numeric matrix with the volumes and substance and organism concentrations of all reactors at all points in time.

This function integrates the system of ordinary differential equations numerically using the function `ode` of the package `deSolve`. The results can be visualized with arbitrary R functions or a summary of all results can be produced with the function

`plotres`.

Usage

```
calcres(system,method="lsoda",...)
```

Arguments

`system`	Object of type `system-class` that defines the model.
`method`	Integration algorithm to be used for numerically solving the system of ordinary differential equations. Available algorithms: "lsoda", "lsode", "lsodes", "lsodar", "vode", "daspk", "euler", "rk4", "ode23", "ode45", "radau", "bdf", "bdf_d", "adams", "impAdams", "impAdams_d". See function `ode` of the package `deSolve` for more details on the integration algorithms.
`...`	Further arguments are passed to the solver. See function `ode` of the package `deSolve` for more details on the integration algorithms and their control parameters.

Value

The function returns a numeric matrix with the volumes and concentrations of substances and organisms of all reactors (columns) for all points in time (rows)

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci. Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

Soetaert, K., Petzoldt, T., and Woodrow Setzer, R. Solving differential equations in R: Package deSolve. Journal of Statistical Software, 33(9), 2010.

Soetaert, K., Cash, J., and Mazzia, F. Solving Differential Equations in R. Springer, Heidelberg, Germany. 2012.

See Also

process-class, reactor-class, link-class, system-class, plotres.

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3
```

```
# Definition of transformation processes:
# =======================================

# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))            # gDM/gDM

# Growth of zooplankton:
# ----------------------

gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),              # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))      # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))            # gDM/gDM

# Definition of reactor:
# ======================

# Epilimnion:
# -----------

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),     # gP/m3
                               C.ALG  = expression(C.ALG.ini),      # gDM/m3
```

```
                              C.ZOO  = expression(C.ZOO.ini)),     # gDM/m3
        inflow          = expression(Q.in*86400),                  # m3/d
        inflow.conc     = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0,
                               C.ZOO  = 0),
        outflow         = expression(Q.in*86400),
        processes       = list(gro.ALG,death.ALG,gro.ZOO,death.ZOO))

# Definition of system:
# =====================

# Lake system:
# ------------

system <- new(Class    = "system",
              name     = "Lake",
              reactors = list(epilimnion),
              param    = param,
              t.out    = seq(0,365,by=1))

# Perform simulation:
# ===================

res <- calcres(system)

# Plot results:
# =============

plotres(res)              # plot to screen

plotres(res,file="ecosim_example_plot1.pdf")  # plot to pdf file

plotres(res, colnames=c("C.ALG", "C.ZOO"))  # plot selected variables

plotres(res, colnames=list("C.HPO4",c("C.ALG", "C.ZOO")))

plotres(res[1:100,], colnames=list("C.HPO4",c("C.ALG", "C.ZOO"))) # plot selected time steps

plotres(res      = res,    # plot to pdf file
        colnames = list("C.HPO4",c("C.ALG","C.ZOO")),
        file     = "ecosim_example_plot2.pdf",
        width    = 8,
        height   = 4)
```

---

calcres-methods calcres

Description

calcres calculates a dynamic solution of the model defined by the argument.

Methods

```
signature(system = "ANY")
signature(system = "system")
```

`calcsens` *Performs a Sensitivity Analysis of the Model Passed as the Argument*

Description

Calculates dynamic solutions of the model defined by the argument with modified parameter values. Each of the specified parameters is multiplied in sequence by the provided scaling factors to produce complete model output matrices for all individual parameter modifications.

Usage

```
calcsens(system, param.sens, scaling.factors=c(1,0.5,2))
```

Arguments

`system`	Object of type `system-class` that defines the model.
`param.sens`	Vector of parameter names for which sensitivity analysis should be performed. The names must be a subset of the names used in the parameter entry of the argument `system`.
`scaling.factors`	Numerical vector of scaling factors with which the parameters should be multiplied. It is recommended that the scaling factors include unity as the first entry to get the basic simulation results. The default is to use the scaling factors 1, 0.5 and 2.

Value

The function returns a list of lists of result matrices of the method `calcres-methods`. The outer list is over the parameters, the inner list over the different values of each parameter. Note that the function `plotres` can be used to plot the results.

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci.

Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

Soetaert, K., Petzoldt, T., and Woodrow Setzer, R. Solving differential equations in R: Package deSolve. Journal of Statistical Software, 33(9), 2010.

Soetaert, K., Cash, J., and Mazzia, F. Solving Differential Equations in R. Springer, Heidelberg, Germany. 2012.

See Also

process-class, reactor-class, link-class, system-class, plotres. calcres-methods.

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3

# Definition of transformation processes:
# =======================================

# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------
```

```
death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))           # gDM/gDM

# Growth of zooplankton:
# ----------------------

gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),             # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))     # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))           # gDM/gDM

# Definition of reactor:
# ======================

# Epilimnion:
# -----------

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),    # gP/m3
                               C.ALG  = expression(C.ALG.ini),     # gDM/m3
                               C.ZOO  = expression(C.ZOO.ini)),    # gDM/m3
       inflow           = expression(Q.in*86400),                  # m3/d
       inflow.conc      = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0,
                               C.ZOO  = 0),
       outflow          = expression(Q.in*86400),
       processes        = list(gro.ALG,death.ALG,gro.ZOO,death.ZOO))

# Definition of system:
# =====================

# Lake system:
# ------------

system <- new(Class    = "system",
              name     = "Lake",
              reactors = list(epilimnion),
```

```
              param    = param,
              t.out    = seq(0,365,by=1))

# Perform simulation:
# ===================

res <- calcres(system)

# Plot results:
# =============

plotres(res)              # plot to screen

plotres(res,file="ecosim_example_plot1.pdf")  # plot to pdf file

plotres(res, colnames=c("C.ALG", "C.ZOO"))  # plot selected variables

plotres(res, colnames=list("C.HPO4",c("C.ALG", "C.ZOO")))

plotres(res[1:100,], colnames=list("C.HPO4",c("C.ALG", "C.ZOO"))) # plot selected time steps

plotres(res      = res,    # plot to pdf file
        colnames = list("C.HPO4",c("C.ALG","C.ZOO")),
        file     = "ecosim_example_plot2.pdf",
        width    = 8,
        height   = 4)
```

`calcsens-methods` `calcsens`

Description

`calcsens` Performs a sensitivity analysis of the model passed as the argument.

Methods

`signature(system = "ANY")`

`signature(system = "system")`

`link-class` *Class* `"link"`

Description

This class represents a link between mixed reactors. A link is characterized by the two reactors it connects to, the flow through the link, advective transfer coefficients, and diffusive exchange coefficients.

Objects from the Class

Objects can be created by calls of the form `new("link", ...)`.

Slots

`name`: Character string specifying the name of the process.

`from`: Character string specifying the name of the reactor where the link starts.

`to`: Character string specifying the name of the reactor where the link ends.

`flow`: Expression specifying the flow through the link.

`qadv.gen`: Expression specifying a general advective transfer coefficient

`qadv.spec`: List of expressions specifying substance/organism-specific advective transfer coefficients.

`qdiff.gen`: Expression specifying a general diffusive exchange coefficient.

`qdiff.spec`: List of expressions specifying substance/organism-specific diffusion coefficients.

Methods

calc.rates.statevar.link Calculates rates of change; internal use only.

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci. Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

See Also

`process-class`, `reactor-class`, `system-class`, `calcres`, `plotres`.

plotres *Plot Simulation Results*

Description

Produces a simple standardized plot of the results generated by calcres or, more generally, plots of all or selected columns of a matrix as a function of the row names interpreted as numbers.

Usage

```
plotres(res,colnames=list(),
        file=NA,width=7,height=7,
        ncol=NA,nrow=NA,
        lwd=2,
      col=c("black","red","blue","pink","orange","violet","cyan","brown","purple"),
      lty=c("solid",paste(2*4:1,2,sep=""),paste(paste(2*4:2,2,sep=""),22,sep="")),
        main=NA,xlab=NA,ylab=NA)
```

Arguments

res	Numerical matrix, list of numerical matrices or list of lists of numerical matrices of data to be plotted. If res is a single numerical matrix, all columns or the columns selected under colnames are plotted versus the row names that are converted to numerics. If res is a list of numerical matrices, the same plots are produced as for a single numerical matrix, but in all plots multiple lines are produced from the different list entries. In the legend, the column names are then appended by the names of the lists. If res is a list of lists of numerical matrices, a set of plots according to the description above are produced. Note that the format of res is compatible with the output of the member functions calcres and calcsens of the class system-class.
colnames	Selection of column names to be plotted. Each list element defines a plot and may contain a vector of column names in which case multiple lines are plotted into the same plot. An empty list indicates that all columns should be plotted.
file	Optional file name to which the plot should be redirected. The file will be written in pdf format.
width	Optional width of the graphics region of the pdf in inches.
height	Optional height of the graphics region of the pdf in inches.
ncol	Optional number of columns of plots to be plotted on the same page.
nrow	Optional number of rows of plots to be plotted on the same page.
lwd	Optional line width(s); values are recycled if more lines are plotted then line widths provided.
lty	Optional line type(s); values are recycled if more lines are plotted then line types provided.
col	Optional line color(s); values are recycled if more lines are plotted then line colors provided.

main	Optional title(s) of plots; values are recycled if more plots are produced than titles provided.
xlab	Optional label(s) of the x axes; values are recycled if more plots are produced than labels provided.
ylab	Optional label(s) of the y axes; values are recycled if more plots are produced than labels provided.

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

See Also

process-class, reactor-class link-class system-class calcres.

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3

# Definition of transformation processes:
# =======================================

# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------
```

```
death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))           # gDM/gDM

# Growth of zooplankton:
# ----------------------

gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),             # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))     # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))           # gDM/gDM

# Definition of reactor:
# ======================

# Epilimnion:
# -----------

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),     # gP/m3
                               C.ALG  = expression(C.ALG.ini),      # gDM/m3
                               C.ZOO  = expression(C.ZOO.ini)),     # gDM/m3
       inflow           = expression(Q.in*86400),                   # m3/d
       inflow.conc      = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0,
                               C.ZOO  = 0),
       outflow          = expression(Q.in*86400),
       processes        = list(gro.ALG,death.ALG,gro.ZOO,death.ZOO))

# Definition of system:
# =====================

# Lake system:
# ------------

system <- new(Class    = "system",
              name     = "Lake",
```

```
                reactors = list(epilimnion),
                param    = param,
                t.out    = seq(0,365,by=1))

# Perform simulation:
# ===================

res <- calcres(system)

# Plot results:
# =============

plotres(res)              # plot to screen

plotres(res,file="ecosim_example_plot1.pdf")  # plot to pdf file

plotres(res, colnames=c("C.ALG", "C.ZOO"))  # plot selected variables

plotres(res, colnames=list("C.HPO4",c("C.ALG", "C.ZOO")))

plotres(res[1:100,], colnames=list("C.HPO4",c("C.ALG", "C.ZOO"))) # plot selected time steps

plotres(res      = res,    # plot to pdf file
        colnames = list("C.HPO4",c("C.ALG","C.ZOO")),
        file     = "ecosim_example_plot2.pdf",
        width    = 8,
        height   = 4)
```

---

process-class *Class* "process"

Description

This class represents a transformation process of substances/organisms in the modelled system. Such a process is characterized by a transformation rate and a list of stoichiometric coefficients for the affected substances and organisms. It is recommended to calculate the stoichiometric coefficients with the function `calc.stoich.coef` of the package `stoichcalc` from substance and organism compositions. The output of this function can directly be used for the process definition.

Objects from the Class

Objects can be created by calls of the form `new("process", ...)`.

Slots

`name`: Character string specifying the name of the process.

`rate`: Expression characterizing the dependence of the transformation rate on substance/organism concentrations and external influence factors

`stoich`: List of expressions or numbers defining the stoichiometric coefficient of the substance/organism given by the label of the list component.

`pervol`: Logical variable defining the process rate as per volume of the reactor (pervol=TRUE) or per surface area (pervol=FALSE).

Methods

calc.trans.rates Calculates transformation rates; internal use only.

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci. Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

See Also

reactor-class, link-class, system-class, calcres, plotres.

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3

# Definition of transformation processes:
# =======================================
```

```
# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))            # gDM/gDM

# Growth of zooplankton:
# ----------------------

gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),              # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))      # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))            # gDM/gDM
```

---

randnorm *Sample from a Univariate Normal or Lognormal Distribution*

Description

Samples from a univariate Normal or Lognormal distribution with parameters valid in the original units. Just invokes a parameter transformation and calls `rnorm` as users often prefer to specify uncertainty in their original units rather than on the log scale as it is done in `rlnorm`.

Usage

```
randnorm(mean=0,sd=1,log=FALSE,n=1)
```

Arguments

mean	Mean of the random variable.
sd	Standard deviation of the random variable.
log	Indicator whether the log of the variable should be normally distributed (log=TRUE) rather than the variable itself. (Note: mean and sd are interpreted in original units also for log=TRUE.)
n	Sample size.

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

See Also

rnorm, rlnorm, randou.

Examples

```
n <- 10000

samp <- randnorm(mean=0,sd=1,n=n)
plot(1:length(samp),samp,xlab="index",ylab="y",cex=0.2)
mean(samp)
sd(samp)

samp <- randnorm(mean=2,sd=2,log=TRUE,n=n)
plot(1:length(samp),samp,ylim=c(0,25),xlab="index",ylab="y",cex=0.2)
mean(samp)
sd(samp)
```

---

randou *Sample from an Ornstein-Uhlenbeck Process*

Description

Samples from an Ornstein-Uhlenbeck process and optionally exponetiates the results. In contrast to most parameterizations, sd represents the asymptotic standard deviation rather than the coefficient in the drift term of the corresponding stochastic differential equation. As in randnorm, mean and sd are interpreted in original, not in log-transformed units to facilitate the characterization of uncertainty in original units.

Usage

```
randou(mean=0,sd=1,tau=0.1,y0=NA,t=0:1000/1000,log=FALSE)
```

Arguments

mean	Asymptotic mean of the process.
sd	Asymptotic standard deviation of the process.
tau	Correlation time of the process.
y0	Starting value of the process. If no value is given, the starting value will be drawn randomly from the asymptotic distribution.
t	Time points at which the process should be sampled. (Note: the value at t[1] will be the starting value y0.)
log	Indicator whether the log of the variable should be an Ornstein-Uhlenbeck process (log=TRUE) rather than the variable itself. (Note: mean and sd are interpreted in original units also for log=TRUE.)

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

See Also

randnorm.

Examples

```
n <- 10000
tau <- 0.1

proc1 <- randou(mean=0,sd=1,tau=tau,y0= 0,t=0:n/n,log=FALSE)
proc2 <- randou(mean=0,sd=1,tau=tau,y0= 1,t=0:n/n,log=FALSE)
proc3 <- randou(mean=0,sd=1,tau=tau,y0=-1,t=0:n/n,log=FALSE)
plot(proc1,xlim=c(0,1),ylim=c(-2.5,2.5),xlab="t",ylab="y",type="l")
lines(proc2,col="red")
lines(proc3,col="blue")
abline(h=0)
mean(proc1$y)
mean(proc2$y)
mean(proc3$y)
sd(proc1$y)
sd(proc2$y)
sd(proc3$y)

procl1 <- randou(mean=2,sd=2,tau=tau,y0=1,t=0:n/n,log=TRUE)
procl2 <- randou(mean=2,sd=2,tau=tau,y0=2,t=0:n/n,log=TRUE)
procl3 <- randou(mean=2,sd=2,tau=tau,y0=3,t=0:n/n,log=TRUE)
plot(procl1,xlim=c(0,1),ylim=c(0,6),xlab="t",ylab="y",type="l")
lines(procl2,col="red")
lines(procl3,col="blue")
mean(procl1$y)
mean(procl2$y)
mean(procl3$y)
sd(procl1$y)
```

```
sd(procl2$y)
sd(procl3$y)
```

`reactor-class` *Class* `"reactor"`

Description

This class represents a well-mixed part of the system as a "reactor". A reactor is characterized by its initial volume, a surface area available for sessile organisms or attached substances, initial concentrations of substances and organisms in the water column and on the surface area, input into the reactor not associated with inflow, inflow into the reactor substance and organism concentrations in the inflow, outflow out of the reactor, environmental conditions to which the reactor is exposed, and processes active in the reactor.

Objects from the Class

Objects can be created by calls of the form `new("reactor", ...)`.

Slots

`name`**:** Character string specifying the name of the reactor.

`volume.ini`**:** Expression specifying the initial volume of the reactor

`area`**:** Expression specifying the surface area available for sessile organisms or attached substances.

`conc.pervol.ini`**:** List of expressions specifying the initial concentrations of substances/organisms in the reactor.

`conc.perarea.ini`**:** List of expressions specifying the initial surface density of sessile organisms or attached substances.

`input`**:** List of expressions specifying the input of substances/organisms into the reactor.

`inflow`**:** Expression specifying the volumetric inflow rate into the reactor.

`inflow.conc`**:** List of expressions specifying the substance/organism concentrations in the inflow.

`outflow`**:** Expression specifying the volumetric outflow rate of the reactor.

`cond`**:** List of expressions specifying the environmental conditions to which the reactor is exposed.

`processes`**:** List of processes that are active in the reactor.

`a`**:** Evaluated area; for internal use only.

Methods

calc.rates.statevar.reactor Calculates rates of change; internal use only.

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci. Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

See Also

process-class, link-class, system-class, calcres, plotres.

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3

# Definition of transformation processes:
# =======================================

# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------
```

```
death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))           # gDM/gDM

# Growth of zooplankton:
# ----------------------

gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),             # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))     # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))           # gDM/gDM

# Definition of reactor:
# ======================

# Epilimnion:
# -----------

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),    # gP/m3
                               C.ALG  = expression(C.ALG.ini),     # gDM/m3
                               C.ZOO  = expression(C.ZOO.ini)),    # gDM/m3
       inflow           = expression(Q.in*86400),                  # m3/d
       inflow.conc      = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0,
                               C.ZOO  = 0),
       outflow          = expression(Q.in*86400),
       processes        = list(gro.ALG,death.ALG,gro.ZOO,death.ZOO))
```

---

system-class *Class* "system"

Description

This class represents a system consisting of linked reactors, substances/organisms in the reactors and transformation processes.

Once a model is described by an object of the class system (system-class), simulations can be performed using the member function

calcres,

This function integrates the system of ordinary differential equations numerically using the function ode of the package deSolve and produces time series of the volumes and substance and organisms concentrations as a R matrix. The results can be visualized with arbitrary R functions or a summary of all results can be produced with the function

plotres.

Objects from the Class

Objects can be created by calls of the form `new("system", ...)`.

Slots

`name`: Character string specifying the name of the system.

`reactors`: List of reactors that build the system.

`links`: List of links that connect the reactors.

`cond`: List of expressions that specify global environmental conditions to which all reactrs are exposed.

`param`: List of model parameters in the form of numerical values or lists of vectors for x and y values describing a realization of a time-dependent parameter.

`t.out`: Numeric vector of points in time at which output should be calculated.

Methods

calcres Calculates simulation results, see calcres

Author(s)

Peter Reichert <peter.reichert@eawag.ch>

References

Omlin, M., Reichert, P. and Forster, R., Biogeochemical model of lake Zurich: Model equations and results, Ecological Modelling 141(1-3), 77-103, 2001.

Reichert, P., Borchardt, D., Henze, M., Rauch, W., Shanahan, P., Somlyody, L. and Vanrolleghem, P., River Water Quality Model no. 1 (RWQM1): II. Biochemical process equations, Water Sci. Tech. 43(5), 11-30, 2001.

Reichert, P. and Schuwirth, N., A generic framework for deriving process stoichiometry in environmental models, Environmental Modelling & Software, 25, 1241-1251, 2010.

Soetaert, K., Petzoldt, T., and Woodrow Setzer, R. Solving differential equations in R: Package deSolve. Journal of Statistical Software, 33(9), 2010.

Soetaert, K., Cash, J., and Mazzia, F. Solving Differential Equations in R. Springer, Heidelberg, Germany. 2012.

See Also

process-class, reactor-class, link-class, calcres, plotres.

Examples

```
# Definition of parameters:
# =========================

param    <- list(k.gro.ALG   = 1,        # 1/d
                 k.gro.ZOO   = 0.8,      # m3/gDM/d
                 k.death.ALG = 0.4,      # 1/d
                 k.death.ZOO = 0.08,     # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 Y.ZOO       = 0.2,      # gDM/gDM
                 alpha.P.ALG = 0.002,    # gP/gDM
                 A           = 8.5e+006, # m2
                 h.epi       = 4,        # m
                 Q.in        = 4,        # m3/s
                 C.ALG.ini   = 0.05,     # gDM/m3
                 C.ZOO.ini   = 0.1,      # gDM/m3
                 C.HPO4.ini  = 0.02,     # gP/m3
                 C.HPO4.in   = 0.04)     # gP/m3

# Definition of transformation processes:
# =======================================

# Growth of algae:
# ----------------

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# Death of algae:
# ---------------

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
```

```
                rate   = expression(k.death.ALG*C.ALG),
                stoich = list(C.ALG  = expression(-1)))           # gDM/gDM

# Growth of zooplankton:
# ----------------------

gro.ZOO   <- new(Class  = "process",
                 name   = "Growth of zooplankton",
                 rate   = expression(k.gro.ZOO
                                     *C.ALG
                                     *C.ZOO),
                 stoich = list(C.ZOO  = expression(1),             # gDM/gDM
                               C.ALG  = expression(-1/Y.ZOO)))     # gP/gDM

# Death of zooplankton:
# ---------------------

death.ZOO <- new(Class  = "process",
                 name   = "Death of zooplankton",
                 rate   = expression(k.death.ZOO*C.ZOO),
                 stoich = list(C.ZOO  = expression(-1)))           # gDM/gDM

# Definition of reactor:
# ======================

# Epilimnion:
# -----------

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),    # gP/m3
                               C.ALG  = expression(C.ALG.ini),     # gDM/m3
                               C.ZOO  = expression(C.ZOO.ini)),    # gDM/m3
       inflow           = expression(Q.in*86400),                  # m3/d
       inflow.conc      = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0,
                               C.ZOO  = 0),
       outflow          = expression(Q.in*86400),
       processes        = list(gro.ALG,death.ALG,gro.ZOO,death.ZOO))

# Definition of system:
# =====================

# Lake system:
# ------------

system <- new(Class    = "system",
              name     = "Lake",
              reactors = list(epilimnion),
              param    = param,
              t.out    = seq(0,365,by=1))
```

Index