

Package ‘cvasi’

May 4, 2024

Type Package

Title Calibration, Validation, and Simulation of TKTD Models

Version 0.10.5

Description Eases the use of ecotoxicological effect models. Can simulate common toxicokinetic-toxicodynamic (TK/TD) models such as General Unified Threshold models of Survival (GUTS) and Lemna. It can derive effects and effect profiles (EPx) from scenarios. It supports the use of 'tidyr' workflows employing the pipe symbol. Time-consuming tasks can be parallelized.

URL <https://github.com/cvasi-tktd/cvasi>

BugReports <https://github.com/cvasi-tktd/cvasi/issues>

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports cli, glue, rlang, stringr, dplyr, tibble, purrr, furr, tidyr, magrittr, utils, stats, methods, grid, ggplot2, deSolve, lubridate, attempt, units

RoxygenNote 7.3.1

Config/testthat/edition 3

Collate 'cache_controls.R' 'class-ExposureSeries.R'
'class-EffectScenario.R' 'calibrate.R' 'class-Transferable.R'
'class-Algae.R' 'class-CalibrationSet.R' 'class-Deb.R'
'class-GutsRed.R' 'class-Lemna.R' 'class-Myriophyllum.R'
'class-ScenarioSequence.R' 'class-parameter_set.R' 'data.R'
'dose_response.R' 'effect.R' 'epx.R' 'get.R' 'globals.R'
'has.R' 'helper_functions.r' 'import.R' 'import_toxswa.R'
'is.R' 'log.R' 'morse.R' 'package.R' 'pl.R' 'plotting.r'
'pull.R' 'sequence.R' 'set.R' 'set_exposure.R' 'set_forcings.R'
'set_init.R' 'set_param.R' 'set_transfer.R' 'set_window.R'
'show.R' 'simulate.R' 'solver.R' 'survival.R' 'utils-pipe.R'
'utils-stats.R'

Suggests future, testthat, morse, knitr, rmarkdown, lemna,
roxyglobals, rjags

Depends R (>= 3.5.0)

VignetteBuilder knitr

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

NeedsCompilation yes

Author Nils Kehrein [aut, cre],
Dirk Nickisch [aut],
Johannes Witt [ctb],
Andre Gergs [ctb]

Maintainer Nils Kehrein <nils.kehrein@gmail.com>

Repository CRAN

Date/Publication 2024-05-03 23:20:04 UTC

R topics documented:

| | |
|--------------------------------|----|
| Algae-models | 4 |
| Algae_Simple | 5 |
| Algae_TKTD | 6 |
| Algae_Weber | 8 |
| americamysis | 9 |
| cache_controls | 10 |
| calibrate | 11 |
| CalibrationSet | 13 |
| CalibrationSet-class | 15 |
| DEB-models | 15 |
| DEB_abj | 16 |
| DEB_Daphnia | 18 |
| dmagna | 20 |
| dose_response | 21 |
| effect | 22 |
| epx | 23 |
| epx_mtw | 25 |
| ExposureSeries | 26 |
| focus1 | 27 |
| fx | 28 |
| get_model | 29 |
| get_tag | 29 |
| GUTS-RED-models | 30 |
| GUTS_RED_IT | 31 |
| GUTS_RED_SD | 33 |
| import_exposure_text | 34 |
| import_swash | 34 |
| import_toxswa | 35 |

| | |
|---------------------|----|
| is_DEB | 35 |
| is_GUTS | 36 |
| is_Lemna | 37 |
| is_LemnaThreshold | 37 |
| is_scenario | 38 |
| Lemna-models | 38 |
| Lemna_Schmitt | 39 |
| Lemna_SETAC | 43 |
| log_enable | 46 |
| log_envir | 47 |
| log_msg | 47 |
| log_scenarios | 48 |
| Macrophyte-models | 48 |
| mae | 49 |
| metsulfuron | 50 |
| minnow_it | 51 |
| minnow_sd | 52 |
| morse | 53 |
| Myrio | 54 |
| Myriophyllum-models | 56 |
| Myrio_log | 56 |
| no_exposure | 59 |
| parameter_set | 59 |
| pll_debug | 60 |
| plot_epx | 61 |
| plot_ppc | 62 |
| plot_ppc_combi | 63 |
| plot_sd | 63 |
| pull_metadata | 65 |
| Rsubcapitata | 65 |
| Scenarios | 66 |
| Schmitt2013 | 69 |
| sequence | 69 |
| set_endpoints | 70 |
| set_exposure | 71 |
| set_forcings | 73 |
| set_init | 74 |
| set_mode_of_action | 75 |
| set_noexposure | 76 |
| set_param | 76 |
| set_tag | 77 |
| set_times | 78 |
| set_transfer | 79 |
| set_window | 81 |
| simulate | 81 |
| simulate_batch | 84 |
| solver | 84 |
| survival | 87 |

Transferable 88

Index 89

Algae-models *Algae models*

Description

Overview of supported *Algae* models

Details

- [Algae_Weber\(\)](#) by Weber *et al.* (2012)
- [Algae_TKTD\(\)](#) based on Weber *et al.* (2012), but with scaled damage
- [Algae_Simple\(\)](#) Simple growth model without additional forcing variables

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using [set_transfer\(\)](#).

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

References

Weber D, Schaeffer D, Dorgerloh M, Bruns E, Goerlitz G, Hammel K, Preuss TG and Ratte HT, 2012. Combination of a higher-tier flow-through system and population modeling to assess the effects of time-variable exposure of isotroturon on the green algae *Desmodesmus subspicatus* and *Pseudokirchneriella subcapitata*. *Environmental Toxicology and Chemistry*, 31, 899-908. [doi:10.1002/etc.1765](https://doi.org/10.1002/etc.1765)

See Also

[Lemna-models](#), [Transferable](#)

Other algae models: [Algae_Simple\(\)](#), [Algae_TKTD\(\)](#), [Algae_Weber\(\)](#)

Other scenarios: [DEB-models](#), [GUTS-RED-models](#), [Lemna-models](#), [Macrophyte-models](#), [Myriophyllum-models](#), [Scenarios](#), [Transferable](#)

Algae_Simple

Algae model with exponential growth but without additional forcings

Description

The model is a mechanistic combined toxicokinetic-toxicodynamic (TK/TD) and growth model for algae. The model simulates the development of algal biomass. The growth of the algae population is simulated on the basis of growth rates, which are, in contrast to the Weber model, independent on environmental conditions which are usually optimal in laboratory effect studies. The toxicodynamic sub-model describes the effects of growth-inhibiting substances through a corresponding reduction in the photosynthesis rate on the basis of internal concentrations.

Usage

Algae_Simple()

Value

an S4 object of type [AlgaeSimpleScenario](#)

State variables

The model has two state variables:

- A, Biomass
- Dw, only used if scaled = 1

Model parameters

- Growth model
 - mu_max, Maximum growth rate (d-1)
 - const_growth, constant growth over time (0 = no, 1 = yes)
- Concentration response (Toxicodynamics)
 - EC_50, Effect concentration of 50% inhibition of growth rate (ug L-1)
 - b, slope of concentration effect curve at EC_50 (-)
 - dose_response, shape of the dose response curve (0 = logit, 1 = probit)
- External concentration (Toxicokinetics)
 - kD, dominant rate constant of toxicant in aquatic environments (d-1)
 - scaled, 0 = no internal scaled damage / 1 = yes (-)

Forcings

simplified model without additional forcings except the external concentration and a forcing variable f_growth if the growth rates are time-dependent. If not time dependent, f_growth should be set to 1.

Simulation output

Simulation results will contain state variables.

References

Weber D, Schaeffer D, Dorgerloh M, Bruns E, Goerlitz G, Hammel K, Preuss TG and Ratte HT, 2012. Combination of a higher-tier flow-through system and population modeling to assess the effects of time-variable exposure of isotroturon on the green algae *Desmodesmus subspicatus* and *Pseudokirchneriella subcapitata*. *Environmental Toxicology and Chemistry*, 31, 899-908. doi:10.1002/etc.1765

See Also

[Scenarios](#), [Transferable](#)

Other algae models: [Algae-models](#), [Algae_TKTD\(\)](#), [Algae_Weber\(\)](#)

| | |
|------------|---|
| Algae_TKTD | <i>Algae model with exponential growth, forcings (P, I) and scaled damage</i> |
|------------|---|

Description

Creates an *Algae* scenario. The model is a variant of the [Algae_Weber\(\)](#) model (Weber 2012) as cited in EFSA TKTD opinion (2018). This Algae model, [Algae_TKTD\(\)](#), a) provides an additional possibility to simulate the dose-response curve (probit) and b) considers an scaled internal damage instead of the external concentration.

Usage

`Algae_TKTD()`

Value

an S4 object of type [AlgaeTKTDScenario](#)

State variables

The model has four state variables:

- A, Biomass (ug fresh wt)
- Q, Mass of phosphorous internal (mg P/ug fresh wt)
- P, Mass of phosphorous external (mg P/L)
- Dw, Damage concentration (ug/L)

Model parameters

- Growth model
 - μ_{\max} , Maximum growth rate (d-1)
 - Q_{\min} , Minimum intracellular P (mg P/ug fresh wt)
 - Q_{\max} , Maximum intracellular P (mg P/ug fresh wt)
 - v_{\max} , Maximum P-uptake rate at non-limited growth (mg P/ug fresh wt/d)
 - k_s , Half-saturation constant for extracellular P (mg P/L)
 - m_{\max} , Natural mortality rate (1/d)
 - I_{opt} , Optimum light intensity for growth (uE/m²/s)
 - T_{opt} , Optimum temperature for growth (°C)
 - T_{\max} , Maximum temperature for growth (°C)
 - T_{\min} , Minimum temperature for growth (°C)
- Concentration response (Toxicodynamics)
 - EC_{50} , Effect concentration of 50% inhibition of growth rate (ug L-1)
 - b , slope of concentration effect curve at EC_{50} (-)
 - $dose_resp$, shape of the dose response curve (0 = logit, 1 = probit)
- External concentration (Toxicokinetics)
 - kD , dominant rate constant (used in `Algae_TKTD()` only) (d-1)

Forcings

Besides exposure events (C_w), the *Algae* model requires two environmental properties as time-series input: Irradiance (I , uE/m²/s) and temperature (T_{act} , deg C). Forcings time-series are represented by `data.frame` objects consisting of two columns. The first for time and the second for the environmental factor in question. The input format for all forcings is a list of the data frames.

Simulation output

Simulation results will contain state variables.

References

Weber D, Schaeffer D, Dorgerloh M, Bruns E, Goerlitz G, Hammel K, Preuss TG and Ratte HT, 2012. Combination of a higher-tier flow-through system and population modeling to assess the effects of time-variable exposure of isoproturon on the green algae *Desmodesmus subspicatus* and *Pseudokirchneriella subcapitata*. *Environmental Toxicology and Chemistry*, 31, 899-908. [doi:10.1002/etc.1765](https://doi.org/10.1002/etc.1765)

See Also

[Scenarios](#), [Transferable](#)

Other algae models: [Algae-models](#), [Algae_Simple\(\)](#), [Algae_Weber\(\)](#)

Algae_Weber

*Algae model with exponential growth and forcings (I, T)***Description**

The model is a mechanistic combined toxicokinetic-toxicodynamic (TK/TD) and growth model for algae. The model simulates the development of algal biomass under laboratory and environmental conditions and was developed by Weber et al. (2012) as cited in EFSA TKTD opinion (2018). The growth of the algae population is simulated on the basis of growth rates, which are dependent on environmental conditions (radiation, temperature and phosphorus). The toxicodynamic sub-model describes the effects of growth-inhibiting substances through a corresponding reduction in the photosynthesis rate on the basis of internal concentrations.

Usage

```
Algae_Weber()
```

Value

an S4 object of type [AlgaeWeberScenario](#)

State variables

The model has four state variables:

- A, Biomass (ug fresh wt)
- Q, Mass of phosphorous internal (mg P/ug fresh wt)
- P, Mass of phosphorous external (mg P/L)
- C, external substance concentration (ug/L)

Model parameters

- Growth model
 - mu_max, Maximum growth rate (d-1)
 - Q_min, Minimum intracellular P (mg P/ug fresh wt)
 - Q_max, Maximum intracellular P (mg P/ug fresh wt)
 - v_max, Maximum P-uptake rate at non-limited growth (mg P/ug fresh wt/d)
 - k_s, Half-saturation constant for extracellular P (mg P/L)
 - m_max, Natural mortality rate (1/d)
 - I_opt, Optimum light intensity for growth (uE/m²/s)
 - T_opt, Optimum temperature for growth (°C)
 - T_max, Maximum temperature for growth (°C)
 - T_min, Minimum temperature for growth (°C)
- Concentration response (Toxicodynamics)

- EC₅₀, Effect concentration of 50% inhibition of growth rate (ug/L)
- b, slope of concentration effect curve at EC₅₀ (-)
- External concentration (Toxicokinetics)
 - k, Degradation rate of toxicant in aquatic environments (d⁻¹)

Forcings

Besides exposure events (C_{in}), the *Algae* model requires three environmental properties as time-series input: Irradiance (I , $\mu E/m^2/s$) and temperature (T_{act} , deg C). Forcings time-series are represented by `data.frame` objects consisting of two columns. The first for time and the second for the environmental factor in question. The input format for all forcings is a list of the data frames.

Simulation output

Simulation results will contain state variables.

References

Weber D, Schaeffer D, Dorgerloh M, Bruns E, Goerlitz G, Hammel K, Preuss TG and Ratte HT, 2012. Combination of a higher-tier flow-through system and population modeling to assess the effects of time-variable exposure of isoproturon on the green algae *Desmodesmus subspicatus* and *Pseudokirchneriella subcapitata*. *Environmental Toxicology and Chemistry*, 31, 899-908. doi:10.1002/etc.1765

EFSA PPR Panel (EFSA Panel on Plant Protection Products and their Residues), Ockleford C, Adriaanse P, Berny P, Brock T, Duquesne S, Grilli S, Hernandez-Jerez AF, Bennekou SH, Klein M, Kuhl T, Laskowski R, Machera K, Pelkonen O, Pieper S, Smith RH, Stemmer M, Sundh I, Tiktak A, Topping CJ, Wolterink G, Cedergreen N, Charles S, Focks A, Reed M, Arena M, Ippolito A, Byers H and Teodorovic I, 2018. Scientific Opinion on the state of the art of Toxicokinetic/Toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms. *EFSA Journal*, 16(8), 5377. doi:10.2903/j.efsa.2018.5377

See Also

[Scenarios](#), [Transferable](#)

Other algae models: [Algae-models](#), [Algae_Simple\(\)](#), [Algae_TKTD\(\)](#)

Description

Species parameters were collected from the AddMyPet database entry on *Americamysis bahia* (Opossum shrimp). The exposure series consists of a constant exposure resulting in medium effects on length and reproduction.

Usage

```
americamysis
```

Format

An object of class DebAbj of length 1.

Source

https://www.bio.vu.nl/thb/deb/deblab/add_my_pet/entries_web/Americamysis_bahia/Americamysis_bahia_res.html

See Also

[DEB_abj\(\)](#)

cache_controls

Cache control simulations

Description

Cache control simulations

Usage

```
cache_controls(x, windows, skipZeroExposure = FALSE, ...)
```

Arguments

| | |
|------------------|---|
| x | vector of scenario objects |
| windows | list of window tuples |
| skipZeroExposure | logical, if TRUE, windows with zero exposure will not be included in calculations |
| ... | additional parameters passed on to effect() |

Value

Modified [scenario](#) objects

`calibrate`*Fit model parameters to experimental data*

Description

The function `calibrate()` performs the calibration (fitting) of model parameters to observed data. Two options are available:

1. Either a single `scenario` is fitted to a single dataset. The dataset must represent a time-series of an output variable of the model, e.g. observed biomass over time (effect data).
2. Or a scenario is fitted to one or multiple datasets at once. Each dataset is represented by a `CalibrationSet` which also contains a scenario which describes the conditions during observation, such as exposure and environmental properties. Each `CalibrationSet` can also be given a weight to be considered in the fitting. By default, the total sum of squared errors is used as the target function which is minimized.

Usage

```
calibrate(x, ...)  
  
## S4 method for signature 'EffectScenario'  
calibrate(  
  x,  
  par,  
  data,  
  endpoint,  
  hessian = TRUE,  
  metric_fun = sse,  
  as_tibble = FALSE,  
  catch_errors = TRUE,  
  verbose = FALSE,  
  ...  
)  
  
## S4 method for signature 'list'  
calibrate(  
  x,  
  par,  
  endpoint,  
  hessian = TRUE,  
  metric_fun = sse,  
  metric_total = sum,  
  as_tibble = FALSE,  
  catch_errors = TRUE,  
  verbose = FALSE,  
  ...  
)
```

Arguments

| | |
|---------------------------|---|
| <code>x</code> | either a single <code>EffectScenario</code> or a list of <code>CalibrationSet</code> objects to be fitted |
| <code>...</code> | additional parameters passed on to <code>stats::optim()</code> and <code>simulate()</code> |
| <code>par</code> | named numeric vector with parameters to fit and their start values |
| <code>data</code> | <code>data.frame</code> with two or more columns with experimental data, 1st column contains time points, 2nd and following columns contain values which the scenario is fitted to. |
| <code>endpoint</code> | character, name of model output column to optimize on |
| <code>hessian</code> | logical, if TRUE the Hessian is returned by the optimization routine |
| <code>metric_fun</code> | vectorized function to calculate an error term that is minimized during optimization, defaults to sum of squared errors |
| <code>as_tibble</code> | logical, if TRUE a tibble with scenario and optimization result is returned, else only the optimization result is returned |
| <code>catch_errors</code> | logical, if TRUE simulation errors due to e.g. out of bounds parameters are caught and optimization continues, else optimization fails |
| <code>verbose</code> | logical, if TRUE then debug outputs are displayed during optimization |
| <code>metric_total</code> | vectorized function to aggregate multiple error terms to a single value, defaults to <code>sum()</code> |

Details

Effect data must be supplied as a `data.frame` in long format with two columns: the first column contains timestamps and the second column the observed property. The effect data may contain replicates as extra rows.

Value

By default, a list of fitted parameters (as produced by `stats::optim()`) is returned. Alternatively, when setting the argument `as_tibble = TRUE`, a tibble is returned with 2 columns, the first column lists the scenario and the 2nd column the list of fitted parameters.

Methods (by class)

- `calibrate(EffectScenario)`: Fit a single scenario
- `calibrate(list)`: Fit using a [CalibrationSet](#)

Examples

```
## 1st option: Fit scenario parameters using a single dataset
metsulfuron

# Observed biomass from Schmitt et al. 2013 experiment
obs32 <- data.frame(
  time=c(0, 3, 5, 7, 7.01, 10, 12, 14),
  obs=c(12, 25, 30, 33, 33, 64, 127, 300)
)
```

```

# calibrate with an ExposureScenario and corresponding effect data
metsulfuron %>%
  calibrate(
    par=c(k_phot_max=1, k_resp=0.1 ),
    data=obs32,
    endpoint="BM",
  )

# plot trial data
obs32$trial <- "conc=1.0"
plot_sd(treatments=metsulfuron@exposure@series, rs_mean=obs32)

## 2nd option: Fit scenario parameters using CalibrationSets

# Create one CalibrationSet for each observed exposure level and time-series
# from Schmitt et al. (2013) dataset
library(dplyr)
Schmitt2013 %>%
  group_by(ID) %>%
  group_map(function(data, key) {
    exp <- data %>% select(t, conc)
    obs <- data %>% select(t, obs)
    sc <- metsulfuron %>% set_exposure(exp)
    CalibrationSet(sc, obs)
  }) -> cs
# Fit parameters
calibrate(cs, par=c(k_phot_max=1), endpoint="BM")

```

CalibrationSet

Create a CalibrationSet object

Description

`CalibrationSet()` creates an object of the `CalibrationSet` class, which combines an `EffectScenario` and the corresponding effect data into one object. The created `CalibrationSet` object can subsequently be used for calibration of a model.

Instead of an `EffectScenario`, a sequence can also be used, e.g., when during an experiment a change in conditions is created which should not be captured by the model (e.g., a pump failure, a sudden change in temperature, or the removal of some experimental organisms from the system to avoid overcrowding, see [sequence\(\)](#)).

Optionally, a weight can be added to the `CalibrationSet`, in case one `CalibrationSet` should have more weight in the subsequent model calibration compared to other `CalibrationSets`.

Usage

```
CalibrationSet(scenario, data, weight = 1)
```

Arguments

| | |
|----------|--|
| scenario | any kind of EffectScenario or <code>sequence()</code> . |
| data | a <code>data.frame</code> with effect data in long format containing two columns: the 1st column with numeric time points and 2nd column with numeric effect data to fit to. |
| weight | optional numeric weight to be applied to this datasets' error term during fitting, default values is 1. |

Value

`CalibrationSet()` returns a `CalibrationSet` object with three slots containing `scenario`, `data`, and `weight`. Warnings are returned when input data are not a `data.frame` with 2 columns.

Examples

```
# Lemna example with `EffectScenario` -----
# 1st get effect data
eff_df <- Schmitt2013 %>%
  dplyr::filter(ID == "T0") %>%
  dplyr::select(t, obs)
# 2nd, create `CalibrationSet`
Cal_set <- CalibrationSet(scenario = metsulfuron, data = eff_df)
# optionally, add a weight to the data
Cal_set2 <- CalibrationSet(scenario = metsulfuron, data = eff_df, weight = 4)
# look at what is returned
Cal_set@data
Cal_set@scenario
Cal_set@scenario@name
Cal_set@weight

# Lemna example with `sequence` -----
# Let's change the end of the exposure scenario in the `metsulfuron` `EffectScenario`
metsulfuron@exposure
altered_exposure_end <- data.frame(
  time = c(5:10),
  c = rep(1.5, 6)
)
metsulfuron %>%
  set_times(c(0:4)) -> metsulfuron_1sthalf
metsulfuron %>%
  set_exposure(altered_exposure_end) %>%
  set_times(c(5:10)) -> metsulfuron_2ndhalf
# then, create a scenario sequence
Sequence <- sequence(
  c(metsulfuron_1sthalf, metsulfuron_2ndhalf)
)
# finally, create a `CalibrationSet` with the sequence
obs32 <- Schmitt2013 %>%
  dplyr::filter(ID == "T0.32") %>%
  dplyr::select(t, obs)
Cal_set3 <- CalibrationSet(Sequence, obs32)
```

```
# look at what is returned
Cal_set3@scenario@scenarios[[1]]
```

CalibrationSet-class *CalibrationSet class*

Description

CalibrationSet class

Slots

scenario any kind of EffectScenario or [sequence\(\)](#)

data data.frame with two columns, 1st column with time points and 2nd column with numerical data to fit to

weight numeric weight to be applied to this datasets' error term during fitting

DEB-models *Dynamic Energy Budget (DEB) models*

Description

Supported models:

- [DEB_abj](#)
- [DEB_Daphnia](#)

See Also

Other DEB models: [DEB_Daphnia\(\)](#), [DEB_abj\(\)](#)

Other scenarios: [Algae-models](#), [GUTS-RED-models](#), [Lemna-models](#), [Macrophyte-models](#), [Myriophyllum-models](#), [Scenarios](#), [Transferable](#)

DEB_abj

*DEB_abj***Description**

Creates a *DEB abj* scenario. The *abj* model with type M acceleration is like model *std*, but acceleration occurs between birth and metamorphosis (V1-morph). Isomorphy is assumed before and after acceleration. Metamorphosis is before puberty and occurs at maturity E_{Hj} , which might or might not correspond with changes in morphology. The *abj* model is a one-parameter extension of model *std* ([DEB Wiki](#)).

Usage

DEB_abj()

Details**State variables:**

The following list describes the default names and standard units of the model's state variables:

- L, structural length (cm)
- E, energy reserve (J)
- H, energy invested in maturity (J)
- R, reproduction buffer (J)
- cV, internal concentration (C)
- Lmax, maximum structural length (cm)

All state variables are initialized with zero. See [set_init\(\)](#) on how to set the initial state.

Parameters:

The following model parameters are required:

- p_M, vol-spec somatic maintenance (J/d.cm³)
- v, energy conductance (cm/d)
- k_J, maturity maint rate coefficient (1/d)
- p_Am, surface-area specific maximum assimilation rate (J/d.cm²)
- kap, allocation fraction to soma (-)
- E_G, spec cost for structure (J/cm³)
- f, scaled functional response (-)
- E_Hj, maturity at metamorphosis (J)
- E_Hp, maturity at puberty (J)
- kap_R, reproduction efficiency (-)
- L_b, structural length at birth (cm)
- L_j, structural length at metamorphosis (cm)
- ke, elimination rate constant (d⁻¹)
- c0, no-effect concentration sub-lethal (C)

- cT, tolerance concentration (C)
- MoA, mode of action switch (-)

Mode of Actions:

Any combination of the following mode of actions (MoA) can be considered by the model:

- MoA = 1: effect on feeding
- MoA = 2: effect on maintenance costs
- MoA = 4: effect on overhead costs for making an egg
- MoA = 8: hazard during oogenesis
- MoA = 16: energy conductance

To activate more than one MoA, simply add up the corresponding codes. To disable all MoAs, set the parameter to zero. See also [set_mode_of_action\(\)](#).

Effects:

The state variables L (structural length) and R (reproduction buffer) are set as effect endpoints by default. All state variables are available as potential endpoints. The list of considered endpoints can be modified by using [set_endpoints\(\)](#).

To calculate effects, each *DEB* scenario is simulated twice: One simulation which considers exposure to a toxicant and one simulation without exposure, i.e. a control. See also [effect\(\)](#).

Value

an S4 object of type [DebAbj](#)

See Also

Other DEB models: [DEB-models](#), [DEB-Daphnia\(\)](#)

Examples

```
# Create an abj scenario from scratch and simulate it
DEB_abj() %>%
  set_init(c(L=0.02,E=0.1,H=0.01)) %>%
  set_param(c(p_M=3000,v=0.02,k_J=0.6,p_Am=300,kap=0.9,E_G=4000,f=1,
             E_Hj=0.05,E_Hp=0.3,kap_R=0.9,ke=1,c0=0,cT=1,L_b=0.02,
             L_j=0.04,MoA=0)) %>%
  set_exposure(no_exposure()) %>%
  set_times(0:10) %>%
  simulate()

# Print information about sample scenario 'americamysis'
americamysis

# Simulate 'americamysis' scenario
americamysis %>% simulate()
```

 DEB_Daphnia

 DEBtox Daphnia

Description

Creates a *DEBtox Daphnia* scemarop.

Usage

DEB_Daphnia()

Details

State variables:

The following list describes the default names and standard units of the model's state variables:

- D, scaled damage (C)
- L, body length (mm)
- R, cumulative reproduction (-)
- S, survival probability (-)

All state variables are initialized with zero. See [set_init\(\)](#) on how to set the initial state.

Parameters:

The following parameters are required:

- General
 - L_0 , body length at start (mm)
 - L_p , body length at puberty (mm)
 - L_m , maximum body length (mm)
 - r_B , von Bertalanffy growth rate constant (1/d)
 - R_m , maximum reproduction rate (#/d)
 - f , scaled functional response (-)
 - hb , background hazard rate (d⁻¹)
- Extra parameters
 - L_f , body length at half-saturation feeding (mm)
 - L_j , body length at which acceleration stops (mm)
 - T_{lag} , lag time for start development (d)
- TKTD parameters
 - k_d , dominant rate constant (d⁻¹)
 - z_b , effect threshold energy budget (C)
 - bb , effect strength energy-budget effects (1/C)
 - z_s , effect threshold survival (C)
 - bs , effect strength survival (1/(C d))
- Global parameters

- FBV, dry weight egg as fraction of structural body weight (-)
- KRV, part. coeff. repro buffer and structure (kg/kg) (for losses with reproduction)
- kap, approximation for kappa (for starvation response)
- yP, product of yVA and yAV (for starvation response)
- Lm_ref, reference max length for scaling rate constants (mm)
- len, switch to fit length 1) with shrinking, 2) without shrinking
- Tbp, brood-pouch delay (d)
- MoA, mode of action switch (-)
- FB, feedback on damage dynamics switch (-)

Mode of Action:

Any combination of the following mode of actions (MoA) can be considered by the model:

- MoA = 1: assimilation/feeding
- MoA = 2: costs for maintenance
- MoA = 4: costs for growth and reproduction
- MoA = 8: costs for reproduction
- MoA = 16: hazard for reproduction

To activate more than one MoA, simply add up the corresponding codes. To disable all MoAs, set the parameter to zero. See also [set_mode_of_action\(\)](#).

Feedbacks:

Any combination of the following damage feedbacks can be considered by the model:

- 1: surf:vol scaling uptake rate
- 2: surf:vol scaling elimination rate
- 4: growth dilution
- 8: losses with reproduction

To activate more than one feedback, simply add up the corresponding codes. To disable all feedbacks, set the parameter to zero.

Effects:

The state variables L (body length), R (cumulative reproduction), and S (survival probability) are set as effect endpoints by default. All state variables are available as potential endpoints. The list of considered endpoints can be modified by using [set_endpoints\(\)](#).

To calculate effects, each *DEB* scenario is simulated twice: One simulation which considers exposure to a toxicant and one simulation without exposure, i.e. a control. See also [effect\(\)](#).

Output variables:

The following intermediary model variables can be added to the model output on demand, see [simulate\(\)](#) for details:

- nout >= 3
 - f, actual f
 - fR, actual f considering starvation
 - kd, actual kd
- nout >= 6
 - s, stress level for metabolic effects

- sA, stress factor on assimilation/feeding
- h, hazard rate for effects on survival
- nout ≥ 10 , damage feedbacks
 - xu, factor for surf:vol scaling uptake rate
 - xe, factor for surf:vol scaling elimination rate
 - xG, factor for growth dilution
 - xR, factor for losses with repro

Value

an S4 object of type [DebDaphnia](#)

See Also

Other DEB models: [DEB-models](#), [DEB_abj\(\)](#)

dmagna

A DEBtox Daphnia magna scenario

Description

Species and substance parameters were collected from test runs of the original DEBtox Daphnia model.

Usage

dmagna

Format

An object of class `DebDaphnia` of length 1.

See Also

[DEB_Daphnia\(\)](#)

| | |
|---------------|--|
| dose_response | <i>Calculate a dose response curve</i> |
|---------------|--|

Description

Returns a `data.frame` with points on the dose response curve for the given effect scenario.

Usage

```
dose_response(  
  scenario,  
  range = c(1, 99),  
  n = 20,  
  strategy = c("exponential", "decadic", "vanilla"),  
  verbose = FALSE,  
  ...  
)
```

Arguments

| | |
|-----------------------|--|
| <code>scenario</code> | EffectScenario used for calculation |
| <code>range</code> | numeric vector specifying the required range of effect levels in percent (%), defaults to <code>c(1, 99)</code> |
| <code>n</code> | minimum number of points on the dose response curve |
| <code>strategy</code> | controls how multiplication factors are chosen, <code>vanilla</code> uses a fixed set of multiplication factors, <code>decadic</code> and <code>exponential</code> have varying step lengths if this is feasible depends on the environment setup as defined by the future package |
| <code>verbose</code> | logical, set to <code>TRUE</code> for additional status messages |
| <code>...</code> | additional arguments passed on to <code>effect()</code> |

Details

Derives a dose response curve from a `scenario`. The result will cover the requested range of effect levels. The tested multiplication factors can be chosen by different strategies, i.e. a `vanilla` approach using a fixed set of factors, or `decadic` and `exponential` approaches employing logarithmic and exponential factor scaling, respectively.

Value

`data.frame` with two columns, i.e. `factor` and `effect`

Examples

```
# basic dose response curve
minnow_sd %>% dose_response()

# modify the minimum number of points on the curve
minnow_sd %>% dose_response(n=10)

# select a subset of the effect range
minnow_sd %>% dose_response(range=c(10,20))

# use an alternative strategy for the selection of multiplication factors
minnow_sd %>% dose_response(strategy="decadic")

# provide additional output how multiplication factors were selected
minnow_sd %>% dose_response(verbose=TRUE)
```

| | |
|--------|---------------------|
| effect | <i>Effect level</i> |
|--------|---------------------|

Description

Derives the effect level due to toxicant exposure in the supplied scenarios. Either relative to a control scenario or derived directly from model endpoints, depending on model type. For scenarios with moving exposure windows, the maximum effect is returned.

Usage

```
effect(x, factor = 1, max_only = TRUE, ep_only = FALSE, marginal_effect, ...)
```

Arguments

| | |
|-----------------|--|
| x | vector of EffectScenario objects |
| factor | optional numeric value which scales the exposure time-series |
| max_only | logical, if TRUE only the maximum effect is returned, else results for all effect windows are reported |
| ep_only | logical, if TRUE only effect endpoints are returned as a vector |
| marginal_effect | numeric, if set, any effect smaller than this threshold will be reported as zero to exclude pseudo-effects originating from small numerical errors |
| ... | additional parameters passed on to simulate() |

Details

By default, only the maximum effect in all moving exposure windows will be returned. If argument `max_only=FALSE` is set, the returned table will be converted to long-format and will contain effect levels for each assessed exposure window.

Output formatting:

Start and end time of exposure windows can be disabled by setting `ep_only=TRUE`. Effect levels smaller than a certain threshold can be automatically set to zero (`0.0`) to avoid spurious effect levels introduced by numerical errors. Set `marginal_effect` to an adequate value less than 1%.

Computational efficiency:

Calculations can be sped up by providing a `data.frame` of pre-calculated control scenarios for each assessed time window. As control scenarios are by definition independent of any exposure multiplication factor, they can be reused for repeated calculations, e.g. to derive effect profiles or dose-response relationships.

Value

a tibble, by default containing scenarios, effect levels, and the exposure window where the maximum effect level occurred. The number of columns depends on the enabled effect endpoints and function arguments.

By default, the first column, named `scenarios`, contains the original scenario objects that were the basis of the calculation. For each effect endpoint, it will be followed by one column with the maximum effect level and two columns containing start and end time of the associated exposure window. If exposure windows are disabled, the columns will just contain the start and end time of the simulation. The effect level column will have the name of the effect endpoint, start and end time will additionally have the suffixes `.dat.start` and `.dat.end`, respectively.

 ep_x
Effect profiles (EP_x values)

Description

Derives one or more EP_x/LP_x values for the supplied effect scenarios, i.e. it calculates the multiplication factors of an exposure profile that cause x% of effect. Scenarios are processed in parallel, if possible.

Usage

```
epx(
  scenarios,
  level = c(10, 50),
  effect_tolerance = 0.001,
  factor_cutoff = NA,
  min_factor = 1e-30,
  max_factor = 1e+30,
  verbose = FALSE,
  ep_only = FALSE,
  long_format = FALSE,
  ...
)
```

Arguments

| | |
|------------------|--|
| scenarios | table or vector of EffectScenario objects |
| level | effect levels in percent (%), defaults to <code>c(10, 50)</code> |
| effect_tolerance | numeric, minimum absolute accuracy of effect levels |
| factor_cutoff | optional numeric, the search for a multiplication factor will be cut short if tried factors exceed this value; the result will report the cutoff value as the final EPx value. |
| min_factor | numeric, if tried factors fall below this threshold, the algorithm will halt with an error |
| max_factor | numeric, if tried factors exceed this threshold, the algorithm will halt with an error |
| verbose | logic, if TRUE then infos about model evaluations are displayed |
| ep_only | logical, if TRUE then only EPx values are part of the output, any contextual information such as EffectScenario objects are left out |
| long_format | logical, if TRUE then EPx values are returned as a table in long format, any contextual information will be duplicated |
| ... | additional arguments passed on to <code>effect()</code> |

Details

To estimate EPx values, a *binary search* on multiplication factors is conducted. The algorithm can achieve arbitrary precision in terms of effects. The same approach is implemented in the `morse` package in the `MFx()` function. Convergence is often achieved in less than 10 iterations per effect level and endpoint.

Internally, a knowledge base of all tried factors and resulting effect levels is kept to speed up convergence if more than one endpoint or effect level was requested. The algorithm will automatically sweep the range of multiplication factors as needed but hard cutoff values are implemented to avoid infinite loops; the algorithm will halt with an error message if tried factors are smaller than $1e-30$ or greater than $1e30$.

Numerical precision:

The precision of reported EPx values is controlled by the argument `effect_tolerance` and is given as the upper absolute error threshold of effects that is deemed acceptable. The default value of `0.001` ensures that a derived EPx will result in an effect of $x\% \pm 0.1$. Decreasing the `effect_tolerance` will result in additional model iterations and longer runtime. Setting an extremely small tolerance value may lead to a breakdown of the algorithm due to the occurrence of extremely small, quasi-random numerical errors in simulation results.

Value

The original tibble with additional columns named after the request effect levels, e.g. `L.EP10`. If no tibble was used as argument, then a new one is created. The first column `scenario` will contain the supplied EffectScenario objects.

Examples

```
minnow_sd %>% epx()
minnow_sd %>% epx(level=c(10,23,42))

# displays infos about tested multiplication factors
minnow_sd %>% epx(verbose=TRUE)

# return results as a table in wide format
minnow_sd %>% epx(long_format=TRUE)
```

| | |
|---------|--|
| epx_mtw | <i>Calculate EPx values for a series of moving time window</i> |
|---------|--|

Description

Calls `epx()` to calculate the EPx value (i.e. the multiplication factors of an exposure profile that cause $x\%$ of effect) for moving windows with length `window_length` that move timesteps defined by `window_interval`.

Usage

```
epx_mtw(
  x,
  level = c(10, 50),
  factor_cutoff = 1000,
  window_length = 7,
  window_interval = 1,
  ...
)
```

Arguments

| | |
|------------------------------|--|
| <code>x</code> | a scenario |
| <code>level</code> | The target effect level of the effect, ie. the x of EPx. |
| <code>factor_cutoff</code> | above which cutoff is the EPx is not relevant |
| <code>window_length</code> | the length of the moving time window |
| <code>window_interval</code> | the interval that the moving time window moves |
| <code>...</code> | arguments passed to <code>epx</code> |

Value

a tibble with five columns

- `window.start`
- `window.end`

- endpoint
- level
- EPx

Examples

```
metsulfuron %>%
  set_window(length=7, interval=1) %>%
  epw_mtw()
```

| | |
|----------------|-----------------------------|
| ExposureSeries | <i>Exposure time-series</i> |
|----------------|-----------------------------|

Description

Creates an object that encapsulates an exposure time-series with its metadata, such as formatted datetime strings and file name where the series was loaded from. `no_exposure()` is shorthand to create a time-series of constant zero exposure.

Usage

```
ExposureSeries(series, dates, file, meta, context)
```

Arguments

| | |
|---------|--|
| series | data.frame with two columns containing a time-series |
| dates | vector, optional original list of time stamps |
| file | character, optional file name where data originates from |
| meta | list, optional metadata |
| context | list optional contextual metadata such as project ids |

Value

an S4 object of type [ExposureSeries](#)

Slots

| | |
|---------|--|
| dates | original time points of time-series, e.g. time stamps of the form 2000-01-01 12:00 |
| file | character, file name where data originates from, may be empty |
| meta | list, contains metadata |
| context | list, contains contextual metadata, such as project ids |
| series | data.frame containing the actual time-series |

See Also

[no_exposure\(\)](#)

focusd1

A Lemna_SETAC scenario with variable environment

Description

A mechanistic combined toxicokinetic-toxicodynamic (TK/TD) and growth model for the aquatic macrophytes *Lemna* spp. as published by Klein *et al.* (2021).

Usage

focusd1

Format

An object of class `LemnaSetacScenario` of length 1.

Details

The scenario will simulate a period of 365 days, a start population of 80 g/m² dry weight, variable environmental conditions, and a complex, time-varying exposure pattern.

The scenario setup was published by Hommen *et al.* (2015). Exposure pattern and substance specific parameters are of exemplary character and represent the herbicide *metsulfuron-methyl*. The parameters were derived by Schmitt *et al.* (2013) based on literature data.

References

Hommen U., Schmitt W., Heine S., Brock Theo CM., Duquesne S., Manson P., Meregalli G., Ochoa-Acuña H., van Vliet P., Arts G., 2015: How TK-TD and Population Models for Aquatic Macrophytes Could Support the Risk Assessment for Plant Protection Products. *Integr Environ Assess Manag* 12(1), pp. 82-95. doi:10.1002/ieam.1715

Klein J., Cedergreen N., Heine S., Reichenberger S., Rendal C., Schmitt W., Hommen U., 2021: Refined description of the *Lemna* TKTD growth model based on Schmitt *et al.* (2013) - equation system and default parameters. Report of the working group *Lemna* of the SETAC Europe Interest Group Effect Modeling. Version 1, uploaded on 22. Sept. 2021. <https://www.setac.org/group/effect-modeling.html>

Schmitt W., Bruns E., Dollinger M., Sowig P., 2013: Mechanistic TK/TD-model simulating the effect of growth inhibitors on *Lemna* populations. *Ecol Model* 255, pp. 1-10. doi:10.1016/j.ecolmodel.2013.01.017

See Also

[Lemna-models](#)

Examples

```
# Simulate the example scenario
focusd1 %>% simulate()
```

| | |
|-----------|-----------------------|
| get_model | <i>Get model name</i> |
|-----------|-----------------------|

Description

Returns the unique model name that is associated with a scenario, e.g. GUTS-RED-IT. The function supports vectorized arguments.

Usage

```
get_model(x)
```

Arguments

x (vector of) [scenarios](#) or [parameter_set](#) objects

Value

vector of character

Examples

```
# returns `GUTS-RED-IT`  
get_model(minnow_it)
```

| | |
|---------|-------------------------|
| get_tag | <i>Get scenario tag</i> |
|---------|-------------------------|

Description

Returns the user-defined, custom tag of a scenario, if available. Tags can be helpful to quickly distinguish scenarios of the same model type. The function supports vectorized inputs.

Usage

```
get_tag(x)
```

Arguments

x (vector of) [scenarios](#) or [parameter_set](#) objects

Value

vector of character

See Also

[set_tag\(\)](#)

Examples

```
# returns `fathead minnow`
get_tag(minnow_it)

# update or set a tag
myscenario <- minnow_it %>% set_tag("My Custom Tag")
# returns `My Custom Tag`
get_tag(myscenario)
```

GUTS-RED-models

GUTS-RED models

Description

Reduced *General Unified Threshold models of Survival* (GUTS) with stochastic death (*SD*) and individual tolerance (*IT*)

Details

The TKTD models *GUTS-RED-SD* and *GUTS-RED-IT* were described by EFSA (2018). GUTS-RED models assume a one-compartment model which directly links external concentration to the scaled damage. The scaled damage is given in units of concentration, equal to the units of measurement in the external medium, e.g. ug/L. The damage dynamics is connected to an individual hazard state variable, resulting in simulated mortality when an internal damage threshold is exceeded. The death mechanisms stochastic death (*SD*) and individual threshold (*IT*) are extreme cases of the *GUTS* theory.

For *SD* models, the threshold parameter for lethal effects is fixed and identical for all individuals of a group, meaning that the variance of the threshold values is zero. Hence, the killing rate relates the probability of a mortality event in proportion to the scaled damage. For *IT* models, the thresholds for effects are distributed among individuals of a group. Mortality of an individual follows immediately once the individual's tolerance is exceeded. Meaning in model terms that the killing rate is set to infinity (EFSA 2018).

State variables

The following list describes the default names and standard units of *GUTS-RED* state variables:

- D, scaled damage (conc)
- H, cumulative hazard (-)

The state variables are initialized with zero by default.

SD model parameters

- kd, dominant rate constant (time⁻¹)
- hb, background hazard rate (time⁻¹)
- z, threshold for effects (conc)
- kk, killing rate constant (time⁻¹)

IT model parameters

- kd, dominant rate constant (time⁻¹)
- hb, background hazard rate (time⁻¹)
- alpha, median of thresholds (conc)
- beta, shape parameter (-)

Effects

The effect endpoint L (lethality) is available for *GUTS-RED* models. A value of zero (0.0) denotes *no effect* on organism survival. A value of one (1.0) denotes a lethality rate of 100%, i.e. no survivors.

In addition, the survival rate over time can be derived for all output time points using [survival\(\)](#).

References

EFSA PPR Panel (EFSA Panel on Plant Protection Products and their Residues), Ockleford C, Adriaanse P, Berny P, et al., 2018: *Scientific Opinion on the state of the art of Toxicokinetic/Toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms*. EFSA Journal 2018; 16(8):5377, 188 pp. doi:10.2903/j.efsa.2018.5377

See Also

Other GUTS-RED models: [GUTS_RED_IT\(\)](#), [GUTS_RED_SD\(\)](#)

Other scenarios: [Algae-models](#), [DEB-models](#), [Lemna-models](#), [Macrophyte-models](#), [Myriophyllum-models](#), [Scenarios](#), [Transferable](#)

GUTS_RED_IT

GUTS-RED-IT scenario

Description

Reduced *General Unified Threshold models of Survival* (GUTS) with individual tolerance (*IT*).

Usage

GUTS_RED_IT(param, init)

Arguments

| | |
|-------|---|
| param | optional named list or vector with model parameters |
| init | optional named numeric vector to use as initial state |

Value

an S4 object of type [GutsRedIt](#)

State variables

The following list describes the default names and standard units of *GUTS-RED* state variables:

- D, scaled damage (conc)
- H, cumulative hazard (-)

The state variables are initialized with zero by default.

IT model parameters

- kd, dominant rate constant (time⁻¹)
- hb, background hazard rate (time⁻¹)
- alpha, median of thresholds (conc)
- beta, shape parameter (-)

Effects

The effect endpoint L (lethality) is available for *GUTS-RED* models. A value of zero (0.0) denotes *no effect* on organism survival. A value of one (1.0) denotes a lethality rate of 100%, i.e. no survivors.

In addition, the survival rate over time can be derived for all output time points using [survival\(\)](#).

References

EFSA PPR Panel (EFSA Panel on Plant Protection Products and their Residues), Ockleford C, Adriaanse P, Berny P, et al., 2018: *Scientific Opinion on the state of the art of Toxicokinetic/Toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms*. EFSA Journal 2018; 16(8):5377, 188 pp. doi:10.2903/j.efsa.2018.5377

See Also

Other GUTS-RED models: [GUTS-RED-models](#), [GUTS_RED_SD\(\)](#)

GUTS_RED_SD

GUTS-RED-SD scenario

Description

Reduced *General Unified Threshold models of Survival* (GUTS) with stochastic death (SD).

Usage

GUTS_RED_SD(param, init)

Arguments

| | |
|-------|---|
| param | optional named list or vector with model parameters |
| init | optional named numeric vector to use as initial state |

Value

an S4 object of type [GutsRedSd](#)

State variables

The following list describes the default names and standard units of *GUTS-RED* state variables:

- D, scaled damage (conc)
- H, cumulative hazard (-)

The state variables are initialized with zero by default.

SD model parameters

- kd, dominant rate constant (time⁻¹)
- hb, background hazard rate (time⁻¹)
- z, threshold for effects (conc)
- kk, killing rate constant (time⁻¹)

Effects

The effect endpoint L (lethality) is available for *GUTS-RED* models. A value of zero (0.0) denotes *no effect* on organism survival. A value of one (1.0) denotes a lethality rate of 100%, i.e. no survivors.

In addition, the survival rate over time can be derived for all output time points using [survival\(\)](#).

References

EFSA PPR Panel (EFSA Panel on Plant Protection Products and their Residues), Ockleford C, Adriaanse P, Berny P, et al., 2018: *Scientific Opinion on the state of the art of Toxicokinetic/Toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms*. EFSA Journal 2018; 16(8):5377, 188 pp. doi:10.2903/j.efsa.2018.5377

See Also

Other GUTS-RED models: [GUTS-RED-models](#), [GUTS_RED_IT\(\)](#)

import_exposure_text *Import exposure profiles from text files*

Description

Read several exposure profiles at their corresponding paths at 'pathfiles'. The exposure profiles are read for each file that are character separated text files in a table format. The expected table format includes the columns time and conc for the concentration at each timestep.

Usage

```
import_exposure_text(pathfiles, profileNames = NA, sep = "\t")
```

Arguments

| | |
|--------------|-------------------------------|
| pathfiles | path to files |
| profileNames | the names of the profiles |
| sep | the field separator character |

Value

list of exposure profiles

import_swash *SWASH project exposure profile import*

Description

Read all TOXSWA files within a SWASH project directory.

Usage

```
import_swash(swash_dir)
```

Arguments

swash_dir path to the SWASH project directory

Value

a list of exposure profiles each with two columns (time and concentration)

| | |
|---------------|---|
| import_toxswa | <i>Read TOXSWA exposure profiles from out-files</i> |
|---------------|---|

Description

Read several TOXSWA exposure profiles from out-files at their corresponding paths given by 'pathtofiles'.

Usage

```
import_toxswa(pathtofiles, profileNames = NA)
```

Arguments

pathtofiles paths to toxswa files that should be read
 profileNames a vector of names to be used for the loaded toxswa files. if "NA" the filename without extension will be used

Value

list of exposure profiles each with two columns (time and concentration)

| | |
|--------|--|
| is_DEB | <i>Test if argument is a DEB model</i> |
|--------|--|

Description

Test if argument is a DEB model

Usage

```
is_DEB(x)
```

Arguments

x vector of EffectScenario objects

Value

vector of logical values

| | |
|---------|---|
| is_GUTS | <i>Test if argument is a GUTS model</i> |
|---------|---|

Description

Test if argument is a GUTS model

Usage

```
is_GUTS(x)
```

```
is_GUTS_IT(x)
```

```
is_GUTS_SD(x)
```

Arguments

x vector of EffectScenario objects

Value

vector of logical values

Functions

- `is_GUTS_IT()`: Test if argument is a GUTS-IT model
- `is_GUTS_SD()`: Test if argument is a GUTS-IT model

Examples

```
# returns `TRUE`  
is_GUTS(minnow_it)  
is_GUTS(GUTS_RED_IT())  
  
# returns `c(TRUE,TRUE,TRUE)`  
is_GUTS(c(minnow_it, minnow_it, minnow_it))  
  
# returns `FALSE`  
is_GUTS_SD(minnow_it)
```

| | |
|----------|--|
| is_Lemna | <i>Test if argument is a Lemna model</i> |
|----------|--|

Description

Also returns TRUE for LemnaThreshold models

Usage

```
is_Lemna(x)
```

Arguments

x vector of [scenarios](#) objects

Value

vector of logical values

See Also

[is_LemnaThreshold\(\)](#)

| | |
|-------------------|---|
| is_LemnaThreshold | <i>Test if argument is a LemnaThreshold model</i> |
|-------------------|---|

Description

Test if argument is a LemnaThreshold model

Usage

```
is_LemnaThreshold(x)
```

Arguments

x vector of [scenarios](#) objects

Value

vector of logical values

See Also

[is_Lemna\(\)](#)

| | |
|-------------|---|
| is_scenario | <i>Test if argument is an effect scenario</i> |
|-------------|---|

Description

Supports vectorized arguments.

Usage

```
is_scenario(x)
```

Arguments

x Some value or object

Value

vector of logical values

Examples

```
# returns `TRUE`  
is_scenario(minnow_it)  
  
# returns `FALSE`  
is_scenario(list())
```

| | |
|--------------|---------------------|
| Lemna-models | <i>Lemna models</i> |
|--------------|---------------------|

Description

Overview of supported *Lemna* models

Details

- [Lemna_Schmitt\(\)](#) by Schmitt *et al.* (2013)
- [Lemna_SETAC\(\)](#) by Klein *et al.* (2021)

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using `set_transfer()`.

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

See Also

[Macrophyte-models](#)

Other Lemna models: [Lemna_SETAC\(\)](#), [Lemna_Schmitt\(\)](#)

Other scenarios: [Algae-models](#), [DEB-models](#), [GUTS-RED-models](#), [Macrophyte-models](#), [Myriophyllum-models](#), [Scenarios](#), [Transferable](#)

Lemna_Schmitt

Lemna model (Schmitt et al. 2013)

Description

The model is a mechanistic combined toxicokinetic-toxicodynamic (TK/TD) and growth model for the aquatic macrophytes *Lemna spp.* The model simulates the development of *Lemna* biomass under laboratory and environmental conditions and was developed by Schmitt *et al.* (2013). Growth of the *Lemna* population is simulated on basis of photosynthesis and respiration rates which are functions of environmental conditions. The toxicodynamic sub-model describes the effects of growth-inhibiting substances by a respective reduction in the photosynthesis rate based on internal concentrations.

Usage

```
Lemna_Schmitt(param, init)
```

```
Lemna_SchmittThold(param, init)
```

Arguments

| | |
|-------|---|
| param | optional named list or vector of model parameters |
| init | optional named numeric vector of initial state values |

Details

Constructors to ease creation of scenarios based on the *Lemna* model by Schmitt *et al.* (2013). A variant of this *Lemna* model, `Lemna_SchmittThold()`, provides an additional cumulative exposure threshold parameter. The Lemna biomass stops growing if the integral of exposure over time exceeds the threshold. The integral of exposure is internally accounted for by an additional state variable AUC (Area Under Curve).

Value

an S4 object of type `LemnaSchmittScenario`

Functions

- `Lemna_SchmittThold()`: model variant with cumulative exposure threshold

State variables

The following list describes the default names and standard units of the model's state variables:

- BM, g_dw/m², dry weight biomass per square meter
- E, -, effect [0,1]
- M_int, ug, internal toxicant mass
- AUC, ug/L, cumulative exposure (**only** for LemnaThreshold model)

Biomass (BM) and internal toxicant mass (M_int) are initialized to zero by default. See `set_init()` on how to set the initial states.

Model parameters

The following model parameters are required:

- Fate and biomass
 - `k_phot_fix`, logical, TRUE then `k_phot_max` is not changed by environmental factors, else FALSE
 - `k_phot_max`, 1/d, maximum photosynthesis rate
 - `k_resp`, 1/d, respiration rate
 - `k_loss`, 1/d, rate of loss (e.g. flow rate)
 - `mass_per_fron`, g_dw/frond, dry weight per frond
 - `BMw2BMd`, g_fw/g_dw, Fresh weight/dry weight
- Effect
 - `E_max`, -, maximum effect [0,1]
 - `EC50`, ug/L, midpoint of effect curve
 - `b`, -, slope of effect curve
- Toxicokinetics
 - `P_up`, cm/d, Permeability for uptake
 - `AperBM`, cm²/g_dw, $A_{leaf} / d_{leaf} = 1/d_{leaf}$ (for circular disc, d=0.05 cm)

- K_{bm}, -, Biomass(fw) : water partition coefficient
- P_Temp, logical, TRUE to enable temperature dependence of cuticle permeability, else FALSE
- MolWeight, g/mol, Molmass of molecule (determines Q₁₀_permeability)
- Temperature dependence
 - T_{min}, deg C, minimum temperature for growth
 - T_{max}, deg C, maximum temperature for growth
 - T_{opt}, deg C, optimal temperature for growth
 - t_{ref}, deg C, reference temperature for respiration rate
 - Q₁₀, -, temperature dependence factor for respiration rate
- Light dependence
 - k₀, 1/d, light dependence: intercept of linear part
 - a_k, (1/d)/(kJ/m².d), light dependence: slope of linear part
- Phosphorus dependence (Hill like dep.)
 - C_P, mg/L, phosphorus concentration in water
 - CP₅₀, mg/L, phosphorus conc. where growth rate is halved
 - a_p, -, Hill coefficient
 - Ki_P, mg/L, p-inhibition constant for very high p-conc.
- Nitrogen dependence (Hill like dep.)
 - C_N, mg/L, nitrogen concentration in water
 - CN₅₀, mg/L, n-conc. where growth rate is halved
 - a_N, -, Hill coefficient
 - Ki_N, mg/L, n-inhibition constant for very high p-conc.
- Density dependence
 - BM₅₀, g_{dw}/m², cut off BM

The Lemna_SchmittThold model requires the following additional parameter:

- threshold, ug/L, cumulative exposure threshold

Forcings

Besides exposure, the Lemna model requires two environmental properties as time-series input: global radiation (rad, kJ/m².d) and temperature (temp, deg C). Forcings time-series are represented by `data.frame` objects consisting of two columns. The first for time and the second for the environmental factor in question.

Entries of the `data.frame` need to be ordered chronologically. A time-series can consist of only a single row; in this case it will represent constant environmental conditions. See [scenarios](#) for more details.

Effects

Supported effect endpoints include *BM* (biomass) and *r* (average growth rate during simulation). The effect on biomass is calculated from the last state of a simulation. Be aware that endpoint *r* is incompatible with frond transfers.

Simulation output

Simulation results will contain two additional columns besides state variables:

- C_int, ug/L, internal concentration of toxicant
- FrondNo, -, number of fronds

It is possible to amend the output of `simulate()` with additional model quantities that are not state variables, for e.g. debugging purposes or to analyze model behavior. To enable or disable additional outputs, use the optional argument `nout` of `simulate()`, see examples below. `nout=1` enables reporting of internal concentration (C_int), `nout=14` enables all additional outputs, and `nout=0` will disable additional outputs.

The available output levels are as follows:

- `nout=1`
 - C_int, internal concentration (ug/L)
- `nout=2`
 - FrondNo, number of fronds (-)
- `nout=3`
 - C_int_u, unbound internal concentration (ug/l)
- `nout=8`, growth and TK/TD
 - BM_fresh, fresh weight biomass (g_fw/m2)
 - k_photo_eff, current photosynthesis rate (1/d)
 - k_resp_eff, current respiration rate (1/d)
 - f_Eff, toxic effect factor (-)
 - P_up_eff, current permeability for uptake (cm/d)
- `nout=11`, environmental factors
 - actConc, current toxicant concentration in surrounding medium (ug/L)
 - actTemp, current environmental temperature (deg C)
 - actRad, current environmental radiation (kJ/m2.d)
- `nout=14`, derivatives
 - d BM/dt, current change in state variable BM
 - d E/dt, current change in effect
 - d M_int/dt, current change in internal toxicant mass

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using `set_transfer()`.

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

References

Schmitt W., Bruns E., Dollinger M., and Sowig P., 2013: *Mechanistic TK/TD-model simulating the effect of growth inhibitors on Lemna populations*. Ecol Model 255, pp. 1-10. doi:10.1016/j.ecolmodel.2013.01.017

See Also

[Lemna-models](#), [Macrophyte-models](#), [Transferable](#), [Scenarios](#)

Other Lemna models: [Lemna-models](#), [Lemna_SETAC\(\)](#)

Other macrophyte models: [Lemna_SETAC\(\)](#), [Macrophyte-models](#), [Myrio\(\)](#), [Myrio_log\(\)](#)

Lemna_SETAC

Lemna model (Klein et al. 2021)

Description

The model was described and published by the SETAC Europe Interest Group Effect Modeling (Klein et al. 2022). It is based on the *Lemna* model by Schmitt (2013). The model is a mechanistic combined toxicokinetic-toxicodynamic (TK/TD) and growth model for the aquatic macrophytes *Lemna spp.*. The model simulates the development of Lemna biomass under laboratory and environmental conditions. Growth of the Lemna population is simulated on basis of photosynthesis and respiration rates which are functions of environmental conditions. The toxicodynamic sub-model describes the effects of growth-inhibiting substances by a respective reduction in the photosynthesis rate based on internal concentrations.

Usage

Lemna_SETAC()

Value

an S4 object of type [LemnaSetacScenario](#)

State variables

The model has two state variables:

- BM, Biomass (g dw m⁻²)
- M_int, Mass of toxicant in plant population (mass per m², e.g. ug m⁻²)

Model parameters

- Growth model
 - k_{photo_fixed} , Model switch for unlimited growth conditions (TRUE/FALSE)
 - k_{photo_max} , Maximum photosynthesis rate (d-1)
 - k_{loss} , Reference loss rate (d-1)
 - $BM_threshold$, Lower biomass abundance threshold, (g dw m-2)
 - BM_min , Reservoir for biomass recovery, (g dw m-2)
- Temperature response of photosynthesis
 - T_{opt} , Optimum growth temperature (°C)
 - T_{min} , Minimum growth temperature (°C)
 - T_{max} , Maximum growth temperature (°C)
- Temperature response of biomass loss rate
 - Q_{10} , Temperature coefficient (-)
 - T_{ref} , Reference temperature for response=1 (°C)
- Irradiance response of photosynthesis
 - α , Slope of irradiance response (m² d kJ⁻¹)
 - β , Intercept of irradiance response (-)
- Nutrient response of photosynthesis
 - N_{50} , Half-saturation constant of Nitrogen (mg N L⁻¹)
 - P_{50} , Half-saturation constant of Phosphorus (mg P L⁻¹)
- Density dependence of photosynthesis
 - BM_L , Carrying capacity (g dw m⁻²)
- Concentration response (Toxicodynamics)
 - $EC50_{int}$, Internal concentration resulting in 50% effect (ug L⁻¹)
 - E_{max} , Maximum inhibition (-)
 - b , Slope parameter (-)
- Internal concentration (Toxicokinetics)
 - P , Permeability (cm d⁻¹)
 - r_{A_DW} , Area per dry-weight ratio (cm² g⁻¹)
 - r_{FW_DW} , Fresh weight per dry weight ratio (-)
 - r_{FW_V} , Fresh weight density (g cm⁻³)
 - r_{DW_FN} , Dry weight per frond ratio (g dw)
 - K_{pw} , Partitioning coefficient plant:water (-)
 - k_{met} , Metabolisation rate (d-1)

Forcings

Besides exposure, the model requires four environmental properties as time-series input:

- tmp , temperature (°C)
- irr , irradiance (kJ m⁻² d⁻¹)

- P, Phosphorus concentration (mg P L⁻¹)
- N, Nitrogen concentration (mg N L⁻¹)

Forcings time-series are represented by `data.frame` objects consisting of two columns. The first for time and the second for the environmental factor in question.

Entries of the `data.frame` need to be ordered chronologically. A time-series can consist of only a single row; in this case it will represent constant environmental conditions. See [scenarios](#) for more details.

Effects

Supported effect endpoints include *BM* (biomass) and *r* (average growth rate during simulation). The effect on biomass is calculated from the last state of a simulation. Be aware that endpoint *r* is incompatible with biomass transfers.

Simulation output

For reasons of convenience, the return value contains by default two additional variables derived from simulation results: the internal concentration `C_int` as well as the number of fronds `FronNo`. These can be disabled by setting the argument `nout = 0`.

The available output levels are as follows:

- `nout >= 1`
 - `C_int`, internal concentration (mass per volume)
- `nout >= 2`
 - `FronNo`, frond number (-)
- `nout >= 4`
 - `f_loss`, respiration dependency function (-)
 - `f_photo`, photosynthesis dependency function (-)
- `nout >= 10`
 - `fT_photo`, temperature response of photosynthesis (-)
 - `fI_photo`, irradiance response of photosynthesis (-)
 - `fP_photo`, phosphorus response of photosynthesis (-)
 - `fN_photo`, nitrogen response of photosynthesis (-)
 - `fBM_photo`, density response of photosynthesis (-)
 - `fCint_photo`, concentration response of photosynthesis (-)
- `nout >= 16`
 - `C_int_unb`, unbound internal concentration (mass per volume)
 - `C_ext`, external concentration (mass per volume)
 - `Tmp`, temperature (deg C)
 - `Irr`, irradiance (kJ m⁻² d⁻¹)
 - `Phs`, Phosphorus concentration (mg P L⁻¹)
 - `Ntr`, Nitrogen concentration (mg N L⁻¹)
- `nout >= 18`
 - `dBM`, biomass derivative (g dw m⁻² d⁻¹)
 - `dM_int`, mass of toxicant in plants derivative (mass per m² d⁻¹)

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using `set_transfer()`.

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

References

Klein J., Cedergreen N., Heine S., Reichenberger S., Rendal C., Schmitt W., Hommen U., 2021: *Refined description of the Lemna TKTD growth model based on Schmitt et al. (2013) - equation system and default parameters*. Report of the working group *Lemna* of the SETAC Europe Interest Group Effect Modeling. Version 1, uploaded on 22. Sept. 2021. <https://www.setac.org/group/effect-modeling.html>

Schmitt W., Bruns E., Dollinger M., and Sowig P., 2013: *Mechanistic TK/TD-model simulating the effect of growth inhibitors on Lemna populations*. Ecol Model 255, pp. 1-10. [doi:10.1016/j.ecolmodel.2013.01.017](https://doi.org/10.1016/j.ecolmodel.2013.01.017)

See Also

[Lemna-models](#), [Macrophyte-models](#), [Transferable](#), [Scenarios](#)

Other Lemna models: [Lemna-models](#), [Lemna_Schmitt\(\)](#)

Other macrophyte models: [Lemna_Schmitt\(\)](#), [Macrophyte-models](#), [Myrio\(\)](#), [Myrio_log\(\)](#)

log_enable

Start and stop logging

Description

Start and stop logging

Usage

```
log_enable(file = NULL, append = TRUE, envir = parent.frame())
```

```
log_disable()
```

Arguments

| | |
|--------|---|
| file | character, file name or path to a log file |
| append | logical, if TRUE output will be appended to an existing log file, otherwise the log file will be replaced |
| envir | log will be automatically disabled if environment is exited, set to NULL to disable |

Value

no return value

| | |
|-----------|-------------------------------------|
| log_envir | <i>Log R environment properties</i> |
|-----------|-------------------------------------|

Description

Log R environment properties

Usage

```
log_envir()
```

Value

no return value

| | |
|---------|--------------------------|
| log_msg | <i>Add a log message</i> |
|---------|--------------------------|

Description

Message will only appear in the console or in log file if logging was enabled using log_enable().

Usage

```
log_msg(...)
```

Arguments

... elements will be concatenated using paste0()

Value

no return value

Examples

```
log_msg("this message will not appear")

log_enable()
log_msg("this message will appear")
log_msg("a number of ", "elements to ", 42, " concatenate")
```

| | |
|---------------|--------------------------------|
| log_scenarios | <i>Log scenario properties</i> |
|---------------|--------------------------------|

Description

Log scenario properties

Usage

```
log_scenarios(x, header = TRUE)
```

Arguments

| | |
|--------|--|
| x | vector of EffectScenario objects |
| header | logical, if TRUE a header line will be printed |

Value

unmodified argument x

| | |
|-------------------|--------------------------|
| Macrophyte-models | <i>Macrophyte models</i> |
|-------------------|--------------------------|

Description

Population models of standard test macrophytes, such as *Lemna spp.*

Details

Available macrophyte models:

- [Lemna](#)
- [Myriophyllum](#)

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using `set_transfer()`.

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

See Also

[Scenarios](#)

Other macrophyte models: [Lemna_SETAC\(\)](#), [Lemna_Schmitt\(\)](#), [Myrio\(\)](#), [Myrio_log\(\)](#)

Other scenarios: [Algae-models](#), [DEB-models](#), [GUTS-RED-models](#), [Lemna-models](#), [Myriophyllum-models](#), [Scenarios](#), [Transferable](#)

mae

Error functions

Description

Functions measuring the accuracy between original and predicted data. Can be used as target functions for e.g. optimization routines which minimize the error term.

Usage

`mae(orig, pred)`

`nmae(orig, pred)`

`rmse(orig, pred)`

`nrmse(orig, pred)`

`smape(orig, pred)`

`sse(orig, pred)`

Arguments

| | |
|-------------------|---|
| <code>orig</code> | numeric vector of original data points |
| <code>pred</code> | numeric vector of predicted data points |

Details

The following error functions are available:

- Mean Absolute Error (MAE)
- Normalized Mean Absolute Error (NMAE)
- Root Mean Square Error (RMSE)
- Normalized Root Mean Square Error (NRMSE)
- Symmetric Mean Absolute Percentage Error (SMAPE)
- Sum of Squared Errors (SSE)

Value

numeric error term

Examples

```
# Mean Absolute Error
mae(1:5, 1:5+0.1)
# Sum of Squared Errors
sse(1:5, 1:5+0.1)
```

metsulfuron

Lemna data published by Schmitt (2013)

Description

Data set for the parametrisation of a mechanistic combined toxicokinetic-toxicodynamic (TK/TD) and growth model for the aquatic macrophytes *Lemna* spp. as published by Schmitt *et al.* (2013). The growth model was parameterised by Schmitt *et al.* based on these data while toxicokinetic and toxicodynamic parameters were determined by calibrating the model using substance specific effect data of metsulfuron-methyl.

Usage

metsulfuron

Format

An object of class `LemnaSchmittScenario` of length 1.

References

Schmitt W., Bruns E., Dollinger M., and Sowig P., 2013: *Mechanistic TK/TD-model simulating the effect of growth inhibitors on Lemna populations*. *Ecol Model* 255, pp. 1-10. doi:10.1016/j.ecolmodel.2013.01.017

See Also

[Lemna-models](#)

`minnow_it`*A GUTS-RED-IT scenario of the fathead minnow*

Description

The scenario consists of a parameterized GUTS-RED-IT model and a constant exposure series. Model parameters were derived from an acute fish toxicity study of the fathead minnow and chlorpyrifos by Geiger *et al.* (1988). The dataset is also referred to as the *GUTS Ring-test dataset C* and was downloaded from the [MOSAIC](https://mosaic.univ-lyon1.fr/guts) website. Fitted parameters were estimated using the [morse](#) package.

Usage

`minnow_it`

Format

An object of class `GutsRedIt` of length 1.

Details

The background mortality rate (`hb`) was set to zero.

Source

<https://mosaic.univ-lyon1.fr/guts>

References

Geiger D.L., Call D.J., and Brooke L.T., 1988: *Acute toxicities of organic chemicals to fathead minnows (Pimephales promelas): Volume IV*, pp. 195-197. University of Wisconsin-Superior, Center for Lake Superior Environmental Studies. ISBN 9780961496838.

See Also

[GUTS-RED-models](#)

`minnow_sd`*A GUTS-RED-SD scenario of the fathead minnow*

Description

The scenario consists of a parameterized GUTS-RED-SD model and a constant exposure series. Model parameters were derived from an acute fish toxicity study of the fathead minnow and chlorpyrifos by Geiger *et al.* (1988). The dataset is also referred to as the *GUTS Ring-test dataset C* and was downloaded from the [MOSAIC](https://mosaic.univ-lyon1.fr/guts) website. Fitted parameters were estimated using the [morse](#) package.

Usage

`minnow_sd`

Format

An object of class `GutsRedSd` of length 1.

Details

The background mortality rate (`hb`) was set to zero.

Source

<https://mosaic.univ-lyon1.fr/guts>

References

Geiger D.L., Call D.J., and Brooke L.T., 1988: *Acute toxicities of organic chemicals to fathead minnows (Pimephales promelas): Volume IV*, pp. 195-197. University of Wisconsin-Superior, Center for Lake Superior Environmental Studies. ISBN 9780961496838.

See Also

[GUTS-RED-models](#)

| | |
|-------|--------------------------------------|
| morse | <i>Import morse model parameters</i> |
|-------|--------------------------------------|

Description

Loads GUTS model parameters which were fitted by the morse package.

Usage

```
morse(  
  file,  
  find.SD = TRUE,  
  find.IT = TRUE,  
  reset.hb = TRUE,  
  params = c("estim", "all"),  
  mcmc.size  
)
```

Arguments

| | |
|-----------|--|
| file | Path to .RData file |
| find.SD | a logical value. If TRUE, it will try to find fitted parameters of the GUTS_SD model |
| find.IT | a logical value. If TRUE, it will try to find fitted parameters of the GUTS_IT model |
| reset.hb | a logical value. If TRUE, the background hazard rate hb is set to zero |
| params | character, if "estim" is selected then only the fitted parameter set for each model is returned, else for "auto" all parameter sets in the MCM chains are returned |
| mcmc.size | optional integer, sets the maximum number of imported parameter sets per MCMC, all parameters are imported by default |

Value

vector of parameter_set objects

Examples

```
# import all parameter fits  
try(morse("path/to/morse_fit.RData"))  
  
# import parameters for a specific model  
try(morse("path/to/morse_fit.RData", find.IT=TRUE, find.SD=FALSE))  
  
# modify model objects  
try(models %>% set_param(morse("path/to/morse_fit.RData")))
```

Myrio

Myriophyllum model with exponential growth**Description**

The *Myriophyllum* model is derived from the *Lemma* TKTD model by Klein *et al.* (2021). The *Myriophyllum* model is mathematically equivalent to the Tier 2C version of the *Lemma* model by Klein *et al.* (2021), cf. [Lemma_SETAC\(\)](#). Recommended settings for Tier 2C are `k_photo_fixed=TRUE` and `k_resp=0` (Klein *et al.* 2021). In particular, the growth model is a simple exponential growth model, which is considered to be the typical situation for a laboratory macrophyte study. Instead of frond numbers as for *Lemma*, the biomass is also returned as total shoot length (TSL) in simulation results. Consequently, the model has the additional parameter `r_DW_TSL` (dry weight per total shoot length ratio) instead of `r_DW_FN` (dry weight per frond number ratio).

Usage

```
Myrio()
```

Value

an S4 object of type [MyrioExpScenario](#)

State variables

The model has two state variables:

- BM, Biomass (g dw m⁻²)
- M_int, Mass of toxicant in plant population (mass per m², e.g. ug m⁻²)

Model parameters

- Growth model
 - `k_photo_max`, Maximum photosynthesis rate (d⁻¹), default: 0.47
- Concentration response (Toxicodynamics)
 - `EC50_int`, Internal concentration resulting in 50% effect (ug L⁻¹)
 - `E_max`, Maximum inhibition (-), default: 1
 - `b`, Slope parameter (-)
- Internal concentration (Toxicokinetics)
 - `P`, Permeability (cm d⁻¹)
 - `r_A_DW`, Area per dry-weight ratio (cm² g⁻¹), default: 1000
 - `r_FW_DW`, Fresh weight per dry weight ratio (-), default: 16.7
 - `r_FW_V`, Fresh weight density (g cm⁻³), default: 1
 - `r_DW_TSL`, Dry weight per total shoot length ratio (?)
 - `K_pw`, Partitioning coefficient plant:water (-), default: 1
 - `k_met`, Metabolisation rate (d⁻¹), default: 0

Environmental factors

None.

Simulation output

Simulation results will contain two additional columns besides state variables:

- C_{int} , internal concentration of toxicant (mass per volume)
- TSL, total shoot length (?)

The available output levels are as follows:

- $nout \geq 1$
 - C_{int} , internal concentration (mass per volume)
- $nout \geq 2$
 - TSL, total shoot length (?)
- $nout \geq 3$
 - f_{photo} , photosynthesis dependency function (-)
- $nout \geq 5$, growth and TK/TD
 - C_{int_unb} , unbound internal concentration (mass per volume)
 - C_{ext} , external concentration (mass per volume)
- $nout \geq 7$, environmental factors
 - dBM , biomass derivative ($g\ dw\ m^{-2}\ d^{-1}$)
 - dM_{int} , mass of toxicant in plants derivative (mass per $m^2\ d^{-1}$)

Effects

Supported effect endpoints include BM (biomass) and r (average growth rate during simulation). The effect on biomass is calculated from the last state of a simulation. Be aware that endpoint r is incompatible with biomass transfers.

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using [set_transfer\(\)](#).

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

References

Klein J., Cedergreen N., Heine S., Reichenberger S., Rendal C., Schmitt W., Hommen U., 2021: *Refined description of the Lemna TKTD growth model based on Schmitt et al. (2013) - equation system and default parameters*. Report of the working group *Lemna* of the SETAC Europe Interest Group Effect Modeling. Version 1, uploaded on 22. Sept. 2021. <https://www.setac.org/group/effect-modeling.html>

See Also

[Macrophyte-models](#), [Transferable](#), [Scenarios](#)

Other Myriophyllum models: [Myrio_log\(\)](#), [Myriophyllum-models](#)

Other macrophyte models: [Lemna_SETAC\(\)](#), [Lemna_Schmitt\(\)](#), [Macrophyte-models](#), [Myrio_log\(\)](#)

Myriophyllum-models *Myriophyllum models*

Description

Supported models:

- [Myrio\(\)](#), with exponential growth
- [Myrio_log\(\)](#), with logistic growth

See Also

[Lemna-models](#), [Transferable](#)

Other Myriophyllum models: [Myrio\(\)](#), [Myrio_log\(\)](#)

Other scenarios: [Algae-models](#), [DEB-models](#), [GUTS-RED-models](#), [Lemna-models](#), [Macrophyte-models](#), [Scenarios](#), [Transferable](#)

Myrio_log *Myriophyllum model with logistic growth*

Description

The *Myriophyllum* model is derived from the *Lemna* TKTD model by Klein *et al.* (2021). [Myrio_log\(\)](#) modifies the [Myrio\(\)](#) model to feature logistic growth, i.e. control growth is described by the differential equation $d\text{ BM}/dt = k_{\text{photo_max}} \cdot \text{BM} \cdot (1 - \text{BM}/\text{BM_L})$ where BM_L is the carrying capacity.

Usage

[Myrio_log\(\)](#)

Value

an S4 object of type [MyrioLogScenario](#)

Model parameters

- Growth model
 - `k_photo_max`, Maximum photosynthesis rate (d-1), default: 0.47
 - `BM_L`, Carrying capacity (g dw m-2)
- Concentration response (Toxicodynamics)
 - `EC50_int`, Internal concentration resulting in 50% effect (ug L-1)
 - `E_max`, Maximum inhibition (-), default: 1
 - `b`, Slope parameter (-)
- Internal concentration (Toxicokinetics)
 - `P`, Permeability (cm d-1)
 - `r_A_DW`, Area per dry-weight ratio (cm² g-1), default: 1000
 - `r_FW_DW`, Fresh weight per dry weight ratio (-), default: 16.7
 - `r_FW_V`, Fresh weight density (g cm-3), default: 1
 - `r_DW_TSL`, Dry weight per total shoot length ratio (?)
 - `K_pw`, Partitioning coefficient plant:water (-), default: 1
 - `k_met`, Metabolisation rate (d-1), default: 0

State variables

The model has two state variables:

- `BM`, Biomass (g dw m-2)
- `M_int`, Mass of toxicant in plant population (mass per m², e.g. ug m-2)

Environmental factors

None.

Simulation output

Simulation results will contain two additional columns besides state variables:

- `C_int`, internal concentration of toxicant (mass per volume)
- `TSL`, total shoot length (?)

The available output levels are as follows:

- `nout >= 1`
 - `C_int`, internal concentration (mass per volume)
- `nout >= 2`
 - `TSL`, total shoot length (?)

- nout >= 3
 - f_photo, photosynthesis dependency function (-)
- nout >= 5, growth and TK/TD
 - C_int_unb, unbound internal concentration (mass per volume)
 - C_ext, external concentration (mass per volume)
- nout >= 7, environmental factors
 - dBM, biomass derivative (g dw m⁻² d⁻¹)
 - dM_int, mass of toxicant in plants derivative (mass per m² d⁻¹)

Effects

Supported effect endpoints include *BM* (biomass) and *r* (average growth rate during simulation). The effect on biomass is calculated from the last state of a simulation. Be aware that endpoint *r* is incompatible with biomass transfers.

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using `set_transfer()`.

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

References

Klein J., Cedergreen N., Heine S., Reichenberger S., Rendal C., Schmitt W., Hommen U., 2021: *Refined description of the Lemna TKTD growth model based on Schmitt et al. (2013) - equation system and default parameters*. Report of the working group *Lemna* of the SETAC Europe Interest Group Effect Modeling. Version 1, uploaded on 22. Sept. 2021. <https://www.setac.org/group/effect-modeling.html>

See Also

[Transferable, Scenarios](#)

Other Myriophyllum models: [Myrio\(\)](#), [Myriophyllum-models](#)

Other macrophyte models: [Lemna_SETAC\(\)](#), [Lemna_Schmitt\(\)](#), [Macrophyte-models](#), [Myrio\(\)](#)

| | |
|-------------|----------------------|
| no_exposure | <i>Zero exposure</i> |
|-------------|----------------------|

Description

Creates an [ExposureSeries](#) with zero concentration. When setting the zero exposure, pay attention not to accidentally reset the output times of your scenario as the zero exposure series contains only a single time point. See the examples.

Usage

```
no_exposure()
```

Value

an S4 object of type [ExposureSeries](#)

See Also

[set_noexposure\(\)](#)

Examples

```
# this will reset the output times of the sample scenario,
# simulate() will quit with an error
try(
  minnow_it %>%
    set_exposure(no_exposure()) %>%
    simulate()
)

# set zero exposure, but keep original output times
minnow_it %>%
  set_exposure(no_exposure(), reset_times=FALSE) %>%
  simulate()
```

| | |
|---------------|--------------------------------|
| parameter_set | <i>Set of model parameters</i> |
|---------------|--------------------------------|

Description

Set of model parameters

Usage

```
parameter_set(model, param = list(), tag = NA_character_)
```

Arguments

| | |
|-------|---|
| model | character, a string containing a model name, e.g. "GUTS-RED-IT" |
| param | named list of model parameters |
| tag | character, an optional identifier |

Value

an S4 object of type `parameter_set`

Slots

| | |
|-------|---|
| model | character, a string containing a model name, e.g. "GUTS-RED-IT" |
| tag | character, an optional identifier |
| param | named list of model parameters |

Examples

```
# create a parameter set and assign it
ps <- parameter_set("GUTS-RED-IT", list(kd=0.12, hb=0.3))
GUTS_RED_IT() %>% set_param(ps)

# multiple scenarios can be modified at once
c(GUTS_RED_IT(), GUTS_RED_IT()) %>%
  set_param(ps)

# model names must match, otherwise an error will be raised
try(GUTS_RED_SD() %>% set_param(ps))
```

pll_debug

Disable parallelization for debugging

Description

In certain cases it might be beneficial to disable parallel execution of e.g. effect profile calculations. By disabling, all processes run sequentially and instantly pass messages to the console which would be delayed during parallel processing. This makes it easier to pinpoint problems within the data or algorithm.

Usage

```
pll_debug(state = TRUE)
```

Arguments

| | |
|-------|---|
| state | logical, if TRUE then parallelization is disabled |
|-------|---|

Value

no return value

| | |
|----------|------------------------|
| plot_epx | <i>Plot EPx values</i> |
|----------|------------------------|

Description

Plot EPx values

Usage

```
plot_epx(
  EPx_ts,
  exposure_ts,
  draw = TRUE,
  time_col = "time",
  conc_col = "conc"
)
```

Arguments

| | |
|-------------|--|
| EPx_ts | the result of <code>epx_mtw</code> , ie. a tibble with <code>window.start</code> , <code>window.end</code> , <code>endpoint</code> , <code>level</code> and <code>EPx</code> |
| exposure_ts | an exposure time series with columns for time 't' and concentration 'conc' |
| draw | Should the whole plot be drawn? If FALSE the exposure plot and the EPx plot are returned as a list for later modification |
| time_col | the name of the time column in the exposure dataset |
| conc_col | the name of the concentration column in the exposure dataset |

Value

a grid of ggplots

Examples

```
ti <- 0:21
expo <- abs(0.01*ti + rnorm(length(ti), 0, 0.05))
exposure <- data.frame(time = ti, conc = expo)
metsulfuron_epx_mtw <- metsulfuron %>%
  set_exposure(exposure) %>%
  ep_mtw(level = 10, factor_cutoff = 1000)
metsulfuron_epx_mtw
plot_epx(EPx_ts = metsulfuron_epx_mtw,
  exposure_ts = exposure)
```

| | |
|----------|--|
| plot_ppc | <i>Creates a PPC plot for a single dataset</i> |
|----------|--|

Description

A sample of parameters representing the uncertainty within the dataset is passed to the function. All parameter combinations and exposure patterns are simulated and the range of predicted frond numbers is derived for a single study. The uncertainty is displayed by a Posterior Predictive Plot (PPC).

Usage

```
plot_ppc(
  rs_mean,
  rs_range,
  obs_mean = NULL,
  obs_full = NULL,
  xy_lim = NULL,
  study = NULL,
  outfile = NULL,
  ...
)
```

Arguments

| | |
|----------|--|
| rs_mean | data.frame, model results best fit params |
| rs_range | data.frame, predictions (min, max from param.sample run) |
| obs_mean | data.frame, observations with means per treatment level |
| obs_full | data.frame, full data set including results for replicates |
| xy_lim | optional numeric, limits of x and y axis for plotting |
| study | optional string, name of study which can be used as key |
| outfile | optional string, file path is given = save graph |
| ... | any additional parameters |

Value

data.frame with fitted and observed frond numbers

| | |
|----------------|---|
| plot_ppc_combi | <i>Create PPC plot for one or more datasets</i> |
|----------------|---|

Description

The function expects a data.frame with four mandatory and one optional column. The mandatory columns are as follows:

- pred: mean of predictions e.g. frond number for lemna
- max: maximum of predictions
- min: minimum of predictions
- obs: observations The optional column is to be named study and contains a study identifier. If more than one study identifier is present in the table, individual studies will be plotted in different colors and a legend will be displayed.

Usage

```
plot_ppc_combi(table, xy_lim = NULL)
```

Arguments

| | |
|--------|--|
| table | data.frame containing return values of calls to plot_ppc() |
| xy_lim | optional numeric, limits of x and y axis for plotting |

| | |
|---------|---|
| plot_sd | <i>Creates plot of model results (uncertainties optional)</i> |
|---------|---|

Description

All parameter combinations and exposure patterns are simulated and the mean of predicted frond numbers is derived for a single study. The uncertainty is passed to the function due to computation time. Results are displayed by plotting the time series including the uncertainty interval. Observation data can be optionally displayed. Data should be provided in long format.

Usage

```
plot_sd(
  model_base,
  treatments,
  rs_mean,
  rs_range = NULL,
  obs_mean = NULL,
  obs_full = NULL,
  x_breaks = NULL,
```

```

    y_lim = NULL,
    grid_labels = NULL,
    grid_ncol = 2,
    plot_col = 2,
    y_title = NULL,
    ...
  )

```

Arguments

| | |
|-------------|---|
| model_base | effect scenario object with mean parameters |
| treatments | treatments exposure levels as data frame |
| rs_mean | data.frame, model results best fit params |
| rs_range | data.frame, uncertainties as data frame |
| obs_mean | data.frame, observation data with means per treatment level |
| obs_full | data.frame, full set including results for replicates |
| x_breaks | optional vector of breaks of x-axis |
| y_lim | optional vector containing limits of y-axis |
| grid_labels | optional labels of grid headers |
| grid_ncol | optional number of grid columns |
| plot_col | output column which should be plotted |
| y_title | optional title of y-axis |
| ... | any additional parameters |

Value

a ggplot2 plot object

Examples

```

set.seed(124)
exposure <- data.frame(
  time = 0:21,
  conc = rnorm(n=22, mean=0.1, sd=0.06),
  trial = "T1")
forcings <- list(temp=12, rad=15000)
param <- list(EC50=0.3, b=4.16, P_up=0.0054)
inits <- list(BM=0.0012, E=1, M_int=0)

scenario <- Lemna_Schmitt() %>%
  set_forcings(forcings) %>%
  set_param(param) %>%
  set_init(inits)

sim_result <- simulate_batch(model_base = scenario,
                             treatments = exposure,
                             param_sample = NULL)

```



```
plot_sd(model_base = scenario,
        treatments = exposure,
        rs_mean = sim_result)
```

| | |
|---------------|-------------------------------------|
| pull_metadata | <i>Pull metadata from scenarios</i> |
|---------------|-------------------------------------|

Description

The method pulls available metadata from scenario objects and returns a table with additional columns. If the argument already was a `data.frame` object, the columns are appended. May overwrite existing columns of the same name.

Usage

```
pull_metadata(x, model = TRUE, exposure = TRUE)
```

Arguments

| | |
|-----------------------|--|
| <code>x</code> | vector of <code>scenarios</code> or a <code>data.frame</code> containing a column <code>scenario</code> with <code>EffectScenario</code> objects |
| <code>model</code> | logical, if <code>TRUE</code> then model metadata is pulled |
| <code>exposure</code> | logical, if <code>TRUE</code> then exposure series metadata is pulled |

Value

a `data.frame`

Examples

```
metsulfuron %>%
  pull_metadata()
```

| | |
|--------------|--------------------------|
| Rsubcapitata | <i>An algae scenario</i> |
|--------------|--------------------------|

Description

Data are from Weber 2012 publication.

Usage

```
Rsubcapitata
```

Format

An object of class `AlgaeTKTDScenario` of length 1.

References

Weber D, Schaeffer D, Dorgerloh M, Bruns E, Goerlitz G, Hammel K, Preuss TG and Ratte HT, 2012. Combination of a higher-tier flow-through system and population modeling to assess the effects of time-variable exposure of isotroturon on the green algae *Desmodesmus subspicatus* and *Pseudokirchneriella subcapitata*. *Environmental Toxicology and Chemistry*, 31, 899-908. doi:10.1002/etc.1765

See Also

[Algae_TKTD](#)

Scenarios

Effect scenario classes

Description

The `EffectScenario` class is the base for all of the basic scenario types and models. It contains slots for data and settings that are required by most models such as a vector of model parameters and a vector of initial states. For each particular model, the class's slots are filled with certain default or fixed values. Some models derive from this class and add slots to store additional data.

Details

Certain behaviors that are required to model complex processes cannot be represented by a single `EffectScenario`. As an example, the parameters of a scenario are generally fixed during the simulated time period. In order to represent a change in parameter values, the original scenario would need to split into two scenarios *A* and *B* which differ by parameter values and simulated time period. By combining these scenarios to a *scenario sequence*, the sequence would be treated as a single, complex scenario. See [sequence\(\)](#) for more information.

Parameters:

Most parameters are represented by numerical types but other types are possible depending on model. Please refer to the model description which parameters are required and in which unit. Some or all parameters may be required to start a simulation. If required parameters are missing, simulation will fail with an error message.

Initial state:

The *initial state* represents the starting values of state variables when starting a simulation. A scenario's default initial state may be insufficient to get sensible results. It is advisable to set an initial state explicitly when creating a new scenario, see [set_init\(\)](#).

In theory, a scenario's state variables can be renamed by modifying the names of the initial state vector. However, this is strongly discouraged as this will affect other routines such as [effect\(\)](#) and [epx\(\)](#) and may render results useless.

Exposure:

Exposure refers to the concentration of toxicant an organism is exposed to. In case of aquatic organisms, this would commonly be the concentration of a toxicant in water. Other interpretations are possible depending on model assumptions.

Exposure time-series are generally represented by a `data.frame` containing two columns. The first column representing time, the second representing the exposure level. The ordering of columns is mandatory. The column names are essentially irrelevant but sensible names may help documenting the scenario and its data. The rows must be ordered chronologically. A time-series can consist of only a single row; in this case it will represent constant exposure. Exposure time-series are set to a scenario using `set_exposure()`.

Handling time-series is a costly task for the ODE solver due to consistency checks and interpolation between time steps. How the solver interpolates the time-series can be controlled by certain arguments to functions such as `simulate()` and `effect()`. Please refer to `simulate()` for a brief overview and `deSolve::forcings` for a detailed description.

Exposure time-series should be kept as short as possible and as complex as needed for optimal computational efficiency.

Environmental forcings:

Forcings generally refer to model parameters that change over time as part of an external function such as environmental temperature and exposure levels. Due to the importance of exposure in regulatory assessments, this R package explicitly distinguishes between environmental forcings and exposure. However, the same restrictions and features apply to both of them.

Forcing time-series are handled the same way as exposure time-series, i.e. they are represented by a `data.frame` containing two columns. The first column representing time, the second representing the parameter that is a function of time. The ordering of columns is mandatory. The rows must be ordered chronologically. Forcings time-series are set using `set_forcings()`. Please refer to the *Exposure* section for more information on how time-series are handled.

Output times:

A scenario's simulated time period is defined by its minimum and maximum output time. Simulation results will only be returned for the defined output times even though the ODE solver may use smaller time steps between output times. Output times can be explicitly set using `set_times()`. The number and distance of output times may have influence on the precision of simulation results and numerical stability, cf. `simulate()`.

Be aware that `set_exposure()` will overwrite previously defined output times if not requested otherwise.

Effects:

Generally, all state variables can be used as effect endpoints but models may provide additional endpoints. Use `set_endpoints()` to enable or disable endpoints for a scenario.

Some scenarios or models require control runs to calculate effects under exposure. Generally, control simulations will run automatically where needed. However, when conducting a large number of repeated simulations, e.g. when calculating effect profiles (EPx values) or simulating moving exposure windows, it may be computationally efficient to run control simulations only once and cache their results within the scenario. Please refer to `cache_controls()` for details.

Moving exposure windows:

The time frame relevant for effects may be much shorter than the assessed exposure time-series for certain organisms. This fact can be represented by moving exposure windows which divide a long time period in a number of consecutive windows of the same length. Each window is simulated individually and effects are calculated. By default, methods such as `effect()` will only return the maximum effect of all considered windows but detailed results can be presented on demand.

To use moving exposure windows, the exposure time-series must be regular, i.e. must have an equidistant step length in time. The length of the window is defined as the number of time steps of the exposure time-series. As an example, assume the time-series has daily granularity and a moving window of seven days length is required. In this case, the moving window must have a length of seven (7) time steps. If the exposure time-series had hourly granularity, the same window would need to have a length of 168 ($=7*24$) time steps. Please refer to `set_window()` for details.

Slots

`name` character, unique model name

`tag` character, an optional identifier

`param` list of parameter key-value pairs

`param.req` character vector of required parameters

`forcings` list of data.frames representing forcing time-series

`forcings.req` character vector or required model forcings data, e.g. temperature

`init` list of initial model states

`times` numeric vector of output times, beginning and end also define the simulated period

`endpoints` character vector of endpoints to calculate results for

`exposure` data.frame with two columns representing an exposure time-series

`control` list of named numerical vectors, contains the control values for all relevant moving windows

`control.req` logical, if TRUE then control values are required to calculate effects

`window.length` numeric, maximum length of the simulated period, if `window.length` is shorter than the exposure pattern, then all possible exposure sub-patterns are evaluated for effect calculation. This is also referred to as a moving window approach.

`window.interval` numeric, interval determining distance between moving windows during effect calculation. First window starts at first time point in exposure pattern.

See Also

Other scenarios: [Algae-models](#), [DEB-models](#), [GUTS-RED-models](#), [Lemna-models](#), [Macrophyte-models](#), [Myriophyllum-models](#), [Transferable](#)

Schmitt2013

A Lemna data set with multiple treatment levels

Description

Data are from Schmitt 2013 publication.

Usage

Schmitt2013

Format

An object of class `data.frame` with 56 rows and 4 columns.

References

Schmitt W., Bruns E., Dollinger M., and Sowig P., 2013: *Mechanistic TK/TD-model simulating the effect of growth inhibitors on Lemna populations*. *Ecol Model* 255, pp. 1-10. doi:10.1016/j.ecolmodel.2013.01.017

See Also

[Lemna-models](#)

sequence

Sequence of scenarios

Description

A sequence of scenarios is treated as a single scenario and each scenario is simulated one after the other. If scenario n in a sequence was simulated, scenario $n+1$ will start off in the model state where n had ended. Scenario sequences can be used to e.g. implement changes in model parameters over time.

Details

Requirements:

All scenarios in a sequence must fulfill the following requirements:

- All scenarios must have identical state variables
- The *output times* of all scenarios must represent a continuous time series without gaps or overlaps

Limitations:

Only simulation of sequences are supported, at the moment. Effects and effect profiles (EPx values) cannot be derived, yet.

Value

an S4 object of type [ScenarioSequence](#)

Examples

```
# create two scenarios that need to be simulated one after the other
scen1 <- minnow_it %>% set_times(0:3)
scen2 <- minnow_it %>% set_times(3:6) %>% set_param(c(kd=0))

# create a sequence and assign scenarios
sq <- sequence(list(scen1, scen2))

# simulate the sequence

simulate(sq)
```

| | |
|---------------|-----------------------------|
| set_endpoints | <i>Set effect endpoints</i> |
|---------------|-----------------------------|

Description

Effect endpoints calculated by functions such as [effect\(\)](#) and [epx\(\)](#) can be enabled and disabled. If an endpoint is not required for an assessment, it should be disabled for reasons of computational efficiency. Please refer to the model description for a list of available endpoints.

Usage

```
set_endpoints(x, endpoints)
```

Arguments

| | |
|-----------|------------------------------------|
| x | vector of EffectScenario objects |
| endpoints | character vector of endpoint names |

Value

Modified EffectScenario objects

Examples

```
# Only enable reproduction (R) endpoint for americamysis scenario
americamysis %>%
  set_endpoints("R") %>%
  effect()

# Enable endpoints length (L) and reproduction (R)
americamysis %>%
  set_endpoints(c("L", "R")) %>%
  effect()
```

| | |
|--------------|---------------------------------|
| set_exposure | <i>Set exposure time-series</i> |
|--------------|---------------------------------|

Description

Exposure refers to the toxicant concentration an organism is exposed to. In case of aquatic organisms, this would commonly be the concentration of a toxicant in water. Other interpretations are possible depending on model assumptions.

Usage

```
set_exposure(scenarios, series, ...)

## S4 method for signature 'ANY,ANY'
set_exposure(scenarios, series)

## S4 method for signature 'EffectScenario,data.frame'
set_exposure(scenarios, series, ...)

## S4 method for signature 'EffectScenario,ExposureSeries'
set_exposure(scenarios, series, reset_times = TRUE)

## S4 method for signature 'EffectScenario,list'
set_exposure(scenarios, series, ...)

## S4 method for signature 'list,list'
set_exposure(scenarios, series, ...)

## S4 method for signature 'list,ANY'
set_exposure(scenarios, series, ...)
```

Arguments

| | |
|-------------|---|
| scenarios | vector of scenarios |
| series | vector of ExposureSeries objects or a single data.frame |
| ... | additional arguments |
| reset_times | logical, if TRUE, the exposure time-series' time points will be set as output times. Defaults to TRUE |

Details

Exposure time-series are generally represented by a data.frame containing two columns. The first column for time, the second representing the exposure level. The ordering of columns is mandatory. The column names are non-relevant but sensible names may help documenting the scenario and its data. The data.frame's rows must be ordered chronologically. A time-series can consist of only a single row; in this case it will represent constant exposure.

For convenience, a time-series with zero exposure can be set using [set_noexposure\(\)](#).

Computational efficiency:

Handling time-series is a costly task for the ODE solver due to consistency checks and interpolation between time steps. How the solver interpolates the time-series can be controlled by optional arguments to functions such as `simulate()` and `effect()`. Please refer to `simulate()` for a brief overview and `deSolve::forcings` for a detailed description.

Exposure time-series should be kept as short as possible and as complex as needed for optimal computational efficiency.

Output times:

By default, the exposure time-series' time points will also be used as output times of the scenario. Any output times previously set by `set_times()` will be lost. If this behavior is undesired, set the function argument `reset_times=FALSE`.

Multiple exposure series and scenarios:

The functions supports modifying multiple scenarios at once: by calling it with lists of `scenario` and `ExposureSeries` objects. The cartesian product of all scenarios and exposure series will be returned, iff the parameter `expand = TRUE` is set.

As an example for the *expand* mode, two scenarios A and B and one exposure series g will result in two scenarios Ag and Bg, both using exposure series g. Two scenarios A and B as well as two exposure series g and h will result in four scenarios Ag,Ah,Bg, and Bh.

Value

list of EffectScenario objects

Examples

```
# set a data.frame as exposure series
Lemna_Schmitt() %>% set_exposure(data.frame(time=c(0, 1, 2, 3), conc=c(1, 1, 0, 0)))

# set one ExposureSeries
es1 <- ExposureSeries(data.frame(time=0, conc=0))
Lemna_Schmitt() %>% set_exposure(es1)

# set two ExposureSeries to create two scenarios
es2 <- ExposureSeries(data.frame(time=5:10, conc=1))
Lemna_Schmitt() %>% set_exposure(c(es1, es2))

# set one ExposureSeries without resetting existing output times
Lemna_Schmitt() %>%
  set_times(0:5) %>%
  set_exposure(es1, reset_times=FALSE)
```

| | |
|--------------|--------------------------------------|
| set_forcings | <i>Set time-dependent parameters</i> |
|--------------|--------------------------------------|

Description

Parameters which change their value over time are referred to as *forcings*. If and what parameters can vary over time depends on the model in question. In many cases, *forcings* represent time-series of environmental properties.

Usage

```
set_forcings(x, ...)  
  
## S4 method for signature 'EffectScenario'  
set_forcings(x, ...)  
  
## S4 method for signature 'list'  
set_forcings(x, ...)
```

Arguments

| | |
|-----|--|
| x | (vector of) scenario objects |
| ... | named argument list to set as forcings |

Details

Forcing time-series are always represented by a `data.frame` containing two columns. The first column representing time, the second representing the parameter that is a function of time. The ordering of columns is mandatory. The column names are essentially irrelevant but may help documenting the scenario and its data. The rows must be ordered chronologically. A time-series can consist of only a single row; in this case it will represent constant conditions.

Handling forcing time-series is a costly task for the ODE solver due to consistency checks and interpolation between timesteps. How the solver interpolates the forcing time-series can be controlled by certain arguments to functions such as [simulate\(\)](#) and [effect\(\)](#). Please refer to [simulate\(\)](#) for a brief overview and [deSolve::forcings](#) for a detailed description.

Forcing time-series should be kept as short as possible and as complex as needed for optimal computational efficiency.

Value

Modified [scenarios](#)

Examples

```
# constant values will be automatically converted to a data.frame
Lemna_Schmitt() %>% set_forcings(temp=20) -> lemna
lemna@forcings

# setting multiple forcings at once
df <- data.frame(t=0:14, temp=rnorm(15, mean=20)) # random temperature series
Lemna_Schmitt() %>% set_forcings(temp=df, rad=15000) -> lemna
lemna@forcings

# forcings can also be supplied as a named list
Lemna_Schmitt() %>% set_forcings(list(temp=20, rad=15000)) -> lemna
lemna@forcings
```

set_init

Set initial state

Description

The *initial state* represents the starting values of a scenario's state variables when starting a simulation. A scenario's default initial state may be insufficient to get sensible results.

Usage

```
set_init(x, init)

## S4 method for signature 'vector'
set_init(x, init)

## S4 method for signature 'EffectScenario'
set_init(x, init)
```

Arguments

```
x          vector of EffectScenario objects
init       named numeric vector
```

Details

In theory, a scenarios's state variables can be renamed by modifying the names of the initial state vector. However, this is strongly discouraged as this will affect other routines such as [effect\(\)](#) and [epx\(\)](#) and may render results useless.

Value

modified EffectScenario objects

Examples

```
# Set initial biomass to 1.0
metsulfuron %>% set_init(c(BM=1.0)) %>% simulate()
```

| | |
|--------------------|---------------------------|
| set_mode_of_action | <i>Set mode of action</i> |
|--------------------|---------------------------|

Description

Updates the model parameter MoA to a certain value

Usage

```
set_mode_of_action(x, code)

set_moa(x, code)
```

Arguments

| | |
|------|----------------------------------|
| x | vector of EffectScenario objects |
| code | a code for a mode of action |

Value

modified EffectScenario objects

Functions

- set_moa(): Shorthand version

Examples

```
# Set MoA=8, i.e. hazard during oogenesis
americamysis %>%
  set_mode_of_action(8) %>%
  effect(method="ode45")

# alternative approach using the parameter directly
americamysis %>%
  set_param(c(MoA=8)) %>%
  effect(method="ode45")
```

| | |
|----------------|--------------------------|
| set_noexposure | <i>Set zero exposure</i> |
|----------------|--------------------------|

Description

The scenarios current exposure is replaced by a constant exposure time-series of value zero(0.0). Output times are unaffected.

Usage

```
set_noexposure(x)
```

Arguments

x vector of [scenarios](#)

Value

vector of [scenarios](#)

Examples

```
# Derive effect size in sample scenario without toxicant exposure
minnow_it %>%
  set_noexposure() %>%
  effect()
```

| | |
|-----------|-----------------------------|
| set_param | <i>Set model parameters</i> |
|-----------|-----------------------------|

Description

Modifies the parameters of one or more EffectScenario objects.

Usage

```
set_param(x, param)
```

```
## S4 method for signature 'EffectScenario,vector'
set_param(x, param)
```

```
## S4 method for signature 'EffectScenario,parameter_set'
set_param(x, param)
```

```
## S4 method for signature 'list,parameter_set'
set_param(x, param)
```

```
## S4 method for signature 'list,vector'
set_param(x, param)

## S4 method for signature 'ScenarioSequence,vector'
set_param(x, param)

## S4 method for signature 'ScenarioSequence,parameter_set'
set_param(x, param)
```

Arguments

| | |
|-------|--|
| x | object(s) to modify |
| param | named numeric vector with parameter names and value OR a list of parameter_set objects |

Details

Most parameters are represented by numerical types but other types are possible depending on model. Please refer to the model description which parameters are required and in which unit. Some or all parameters may be required to start a simulation. If required parameters are missing, simulation will fail with an error message.

Value

Vector of modified objects

Examples

```
Lemna_Schmitt() %>% set_param(c(Emax=1,EC50=0.12))
```

| | |
|---------|------------------|
| set_tag | <i>Set a tag</i> |
|---------|------------------|

Description

Sets the user-defined, custom tag of a scenario. Tags can be helpful to quickly distinguish scenarios of the same model type.

Usage

```
set_tag(x, tag)
```

Arguments

| | |
|-----|------------------------------------|
| x | (vector of) EffectScenario objects |
| tag | vector of character |

Value

(vector of) modified EffectScenario objects

See Also

[get_tag\(\)](#)

Examples

```
# set a custom tag
myscenario <- GUTS_RED_SD() %>% set_tag("My Custom Tag")

# returns `My Custom Tag`
get_tag(myscenario)

# the tag also appears in the scenario overview
myscenario
```

set_times

Set output times

Description

Minimum and maximum output times define the simulated period for a scenario. Simulation results will be returned for each output time, see [simulate\(\)](#).

Usage

```
set_times(x, times)
```

Arguments

| | |
|-------|-------------------------------------|
| x | vector of scenarios |
| times | numerical vector |

Details

Be aware that output times may be modified by [set_exposure\(\)](#). Precision of simulation results may be influenced by chosen output times, see [simulate\(\)](#) for more information.

Value

Vector of modified ExposureScenario objects

See Also

[simulate\(\)](#)

Examples

```
# Set simulated period to [2,4] with output intervals of length 1
minnow_it %>% set_times(c(2,3,4))

# Decrease output interval length to 0.1
minnow_it %>% set_times(seq(2, 4, 0.1))
```

| | |
|--------------|----------------------------|
| set_transfer | <i>Set transfer events</i> |
|--------------|----------------------------|

Description

A *transfer* refers to an event where a certain amount of biomass (*BM*) is moved to a new medium after a period of time. This feature replicates a procedure occurring e.g. in Lemna effect studies and may be necessary to recreate study results.

Usage

```
set_transfer(x, interval, times, biomass, scaled_comp)

## S4 method for signature 'ANY'
set_transfer(x, interval, times, biomass, scaled_comp)

## S4 method for signature 'Transferable'
set_transfer(x, interval, times, biomass, scaled_comp)
```

Arguments

| | |
|--------------------------|--|
| <code>x</code> | vector of EffectScenario objects |
| <code>interval</code> | optional numeric, interval in time units of the scenario, set to -1 to disable transfers. |
| <code>times</code> | optional numeric vector of time points where transfers occur |
| <code>biomass</code> | optional numeric vector, amount of biomass that is being transferred at each transfer |
| <code>scaled_comp</code> | optional character vector of affected compartments that are scaled according to new biomass levels |

Details

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

Transferred biomass:

At each transfer, a defined amount of biomass is transferred to a new medium. This is modeled by interrupting the simulation at a transfer time point, modifying the biomass level *BM*, and scaling

affected compartments according to new biomass levels. Scaling of compartments depending on biomass, such as *internal toxicant mass*, is necessary to correctly reflect mass balances and concentrations over time.

Transferred biomass is set using the biomass parameter. It is either a single numerical value in which case the same biomass level is set at each transfer. Or it is a vector of numerical values with the same length as the times parameter in which case a custom biomass level can be set for each transfer. Multiple biomass levels can only be set in conjunction with custom transfer time points. Some scenario types define default values for transferred biomass based on common study set ups.

Regular and custom transfer time points:

Transfers can occur either in regular intervals of time or at selected, custom time points. For regular intervals, the parameter interval is set to a single numeric value which has the same unit as the scenario's time dimension. As an example: if a scenario uses the unit of *days* for time, the transfer interval is also specified in *days*:

Transfers occurring at custom time points are set by passing a numerical vector to the parameter times. The time points' units must match with the unit of time in the scenario. A custom transfer time point **must not occur at the starting time point of a simulation**.

Affected compartments:

Some compartments depend on biomass to correctly reflect mass balances and concentrations over time, such as *internal toxicant mass*. These compartments need to be scaled linearly to reflect the change in biomass levels. The parameter scaled_comp accepts a character vector of compartment names which are scaled at each transfer. This parameter should only be used with custom, user-defined models. If no compartment needs to be scaled, set or use the default value of character(0).

Value

Modified EffectScenario objects

See Also

[Lemna_Schmitt\(\)](#)

Examples

```
# Simulate biomass transfer of 50 *g dw/m²* at a regular interval of 7 *days*
metsulfuron %>%
  set_transfer(interval=7, biomass=50) %>%
  simulate()

# Simulate irregular biomass transfers occurring at days 5, 10, and 12
metsulfuron %>%
  set_transfer(times=c(5,10,12), biomass=50) %>%
  simulate()

# Simulate irregular transfers with changing amounts of transferred biomass
metsulfuron %>%
  set_transfer(times=c(5,10,12), biomass=c(50,20,10)) %>%
  simulate()
```

| | |
|------------|--------------------------|
| set_window | <i>Set window length</i> |
|------------|--------------------------|

Description

Exposure windows are defined as a period of time at the scale of the exposure series. As an example: if an exposure series has an hourly time step, a window length of 24 will consider the exposure within 24 hours intervals for effect calculation. The same applies for the window interval, i.e. the period between considered exposure windows. Set length=-1 to disable moving windows.

Usage

```
set_window(x, length, interval)
```

Arguments

| | |
|----------|--|
| x | vector of EffectScenario objects |
| length | numeric, length of exposure window to consider for effect calculation, set length=-1 to disable moving windows |
| interval | numeric, interval between considered exposure windows |

Value

modified EffectScenario objects

Examples

```
# calculate the maximum effect for all windows of 10 days length
metsulfuron %>%
  set_window(length=10, interval=1) %>%
  effect()
```

| | |
|----------|------------------------------------|
| simulate | <i>Simulate an effect scenario</i> |
|----------|------------------------------------|

Description

The supplied EffectScenario is passed on to the ODE solver for numerical integration. Internally, simulate() is split up into several functions dedicated to particular models, e.g. one for GUTS and one for Lemna type models. The package will take care of using the correct function for each model when simulate() is called.

Usage

```
simulate(x, ...)  
  
## S4 method for signature 'EffectScenario'  
simulate(x, ...)  
  
## S4 method for signature 'Transferable'  
simulate(x, ...)  
  
## S4 method for signature 'ScenarioSequence'  
simulate(x, ...)
```

Arguments

| | |
|-----|---|
| x | scenario to simulate |
| ... | additional parameters passed on to ODE solver |

Details

Simulation results are returned as a time-series for each state variable. Some models provide additional properties describing the model state, e.g. the internal concentration of a toxicant within the organism. Refer to the respective [scenario](#) for more information.

Additional arguments to `simulate()` will be passed on to `deSolve::ode()` which enables control of the numerical integration parameters.

Output times and windows:

The minimum and maximum of given time points generally define the simulated period. Function argument `times` overrides settings of the scenario, i.e. time points set in `scenario@times`. However, the simulation can be limited to a subset of time points by enabling a moving exposure window, see [set_window\(\)](#).

Results will be returned for each time point. Precision of the numeric solver may be affected by chosen output times in certain cases. Hence, small deviations in results should be expected if different output times are set. This effect can be mitigated by either defining are sufficiently small time step for the solver using argument `hmax` or by decreasing the error tolerances `atol` and `rtol`. These arguments are passed to the solver, see e.g. [deSolve::lsoda\(\)](#) for details.

Numerical precision and stability:

Each model was assigned a default ODE solver which handles most of the occurring inputs well. In most cases, this will be an explicit numerical scheme of the Runge-Kutta family with variable step width. For certain extreme parameters settings, such as very high uptake/permeability of the contaminant or exposure series which represent step functions, the numerical approximation might deteriorate and yield invalid results. In this case try to decrease the allowed max step width by setting the argument `hmax` with various values. Start with `hmax=1` and decrease the value by orders of 10. It is not possible or desirable to reduce `hmax` to extremely small values, as the ODE solver will require more CPU time and simulation will become inefficient.

Often times, it will be computational more efficient to adapt the solver's error tolerances `atol` and `rtol` than reducing the step width `hmax` to achieve stable numerics. Start by decreasing

deSolve's default values by orders of ten until the simulation yields acceptable results, see e.g. `deSolve::lsoda()` for more information on error tolerances.

As an alternative to adapting solver parameters, it might be worthwhile to try other numerical schemes which can handle stiff ODEs, such as Radau, LSODA, or LSODES. To change solvers, set the `method` argument. To select e.g. the Radau scheme, set `method="radau"`. For LSODA, set `method="lsoda"`. Radau performs better than LSODA in some cases, as the latter method can return biologically nonsensical results without raising an error. See `deSolve::ode()` for details on available ODE solvers.

Value

A `data.frame` with the time-series of simulation results

Examples

```
minnow_sd %>% simulate() # tidy syntax
simulate(minnow_sd) # base R syntax

# Set new output times
minnow_sd %>% simulate(times=c(0,4))

# Modify simulated time frame
minnow_sd %>% simulate(times=c(0,10))

# Use an alternative exposure profile than defined in the scenario
minnow_sd %>% set_exposure(data.frame(t=0,c=10), reset_times=FALSE) %>% simulate()

##
## Precision of results

# A large number of output times forces smaller solver time steps
minnow_it %>% simulate(times=seq(0,1,0.001)) %>% tail()

# Defining only two output times allows for larger solver time steps
minnow_it %>% simulate(times=c(0,1))

# The difference between results can be reduced by limiting the solver's
# maximum step length
minnow_it %>% simulate(times=c(0,1), hmax=0.001)

# The same numerical precision can be achieved by switching to
# the Radau scheme
minnow_it %>% simulate(times=c(0,1), method="radau")

# A very small step length may reduce the difference even further but may
# also exceed the allowed number of steps per output interval. The following
# simulation will be aborted with a solver error:
try(
  minnow_it %>% simulate(times=c(0,1), hmax=0.0001)
)

# However, the solver's max number of allowed steps can be increased:
```

```
minnow_it %>% simulate(times=c(0,1), hmax=0.0001, maxsteps=10^5)
```

| | |
|----------------|---|
| simulate_batch | <i>Returns data frame with simulation results</i> |
|----------------|---|

Description

Returns data frame with simulation results

Usage

```
simulate_batch(model_base, treatments, param_sample = NULL)
```

Arguments

| | |
|--------------|---|
| model_base | effect scenario object with mean parameters |
| treatments | treatments exposure levels as data frame (t, conc, trial) |
| param_sample | data.frame with parameter sample |

Value

a data.frame

| | |
|--------|--|
| solver | <i>Calls ODE solver for a particular model</i> |
|--------|--|

Description

Please refer to the *Modeling Howto* vignette on how to implement custom models by overloading the solver function.

Usage

```
solver(scenario, times, ...)

## S4 method for signature 'ANY'
solver(scenario, times, ...)

## S4 method for signature 'GutsRedSd'
solver(scenario, times, ...)

## S4 method for signature 'GutsRedIt'
solver(scenario, times, ...)

## S4 method for signature 'LemnaSchmittScenario'
```

```
solver(scenario, times, ...)  
  
## S4 method for signature 'LemnaSetacScenario'  
solver(scenario, times, ...)  
  
## S4 method for signature 'MyrioExpScenario'  
solver(scenario, times, ...)  
  
## S4 method for signature 'MyrioLogScenario'  
solver(scenario, times, ...)  
  
## S4 method for signature 'DebAbj'  
solver(scenario, times, ...)  
  
## S4 method for signature 'DebDaphnia'  
solver(  
  scenario,  
  times,  
  approx = c("linear", "constant"),  
  f = 1,  
  method = "ode45",  
  ...  
)  
  
## S4 method for signature 'AlgaeWeberScenario'  
solver(  
  scenario,  
  times,  
  approx = c("linear", "constant"),  
  rule = 2,  
  f = 1,  
  method = "lsoda",  
  hmax = 0.1,  
  ...  
)  
  
## S4 method for signature 'AlgaeTKTDSscenario'  
solver(  
  scenario,  
  times,  
  approx = c("linear", "constant"),  
  rule = 2,  
  f = 1,  
  method = "lsoda",  
  hmax = 0.1,  
  ...  
)
```

```
## S4 method for signature 'AlgaeSimpleScenario'
solver(
  scenario,
  times,
  approx = c("linear", "constant"),
  rule = 2,
  f = 1,
  method = "ode45",
  hmax = 0.01,
  ...
)
```

Arguments

| | |
|----------|---|
| scenario | scenario object |
| times | numeric vector of output times, overrides any scenario setting |
| ... | additional parameters passed on to <code>deSolve::ode()</code> |
| approx | string, interpolation method of exposure series, see <code>stats::approxfun()</code> |
| f | if approx="constant", a number between 0 and 1 inclusive, see <code>stats::approxfun()</code> |
| method | string, numerical solver used by <code>deSolve::ode()</code> |
| rule | how to handle data points outside of time-series, see <code>deSolve::forcings</code> |
| hmax | numeric, maximum step length in time, see <code>deSolve::ode()</code> |

Value

data.frame with simulation results

Methods (by class)

- `solver(ANY)`: Default solver, raises an error
- `solver(GutsRedSd)`: Numerically integrates GUTS-RED-SD models
- `solver(GutsRedIt)`: Numerically integrates GUTS-RED-IT models
- `solver(LemnaSchmittScenario)`: Numerically integrates Lemna_Schmitt models
- `solver(LemnaSetacScenario)`: Numerically integrates Lemna_SETAC models
- `solver(MyrioExpScenario)`: Numerically integrates Myrio models
- `solver(MyrioLogScenario)`: Numerically integrates Myrio_log models
- `solver(DebAbj)`: Numerically integrates DEB_abj models
- `solver(DebDaphnia)`: Numerically integrates DEB_Daphnia models
- `solver(AlgaeWeberScenario)`: numerically integrates Algae_Weber models
- `solver(AlgaeTKTDScenario)`: numerically integrates Algae_TKTD models
- `solver(AlgaeSimpleScenario)`: numerically integrates Algae_Simple models

| | |
|----------|----------------------|
| survival | <i>Survival rate</i> |
|----------|----------------------|

Description

Derives the survival rate of individuals for *Reduced GUTS* models

Usage

```
survival(scenario, ...)
```

Arguments

| | |
|----------|---|
| scenario | an EffectScenario to simulate |
| ... | additional parameters passed on to simulate() |

Details

The survival rate describes the survival probability at each time point. The function simulates the *GUTS* scenario and appends a column `survival` to the simulation result. A value of one (1.0) denotes that all individuals survive. A value of zero (0.0) denotes that no individuals survived.

Only available for *Reduced GUTS* models, see [GUTS-RED-models](#). The equations were described by EFSA (2018).

Value

a `data.frame` containing simulation results

References

EFSA PPR Panel (EFSA Panel on Plant Protection Products and their Residues), Ockleford C, Adriaanse P, Berny P, et al., 2018: *Scientific Opinion on the state of the art of Toxicokinetic/Toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms*. EFSA Journal 2018; 16(8):5377, 188 pp. [doi:10.2903/j.efsa.2018.5377](https://doi.org/10.2903/j.efsa.2018.5377)

See Also

[GUTS-RED-models](#)

Examples

```
# calculate survival rate
minnow_it %>% survival()

# plot survival over time based on a random exposure profile
minnow_sd %>%
  set_exposure(data.frame(t=1:100, c=runif(100)*10)) %>%
  survival() -> df
plot(df$time, df$survival, "l")
```

 Transferable

Biomass transfer class

Description

By inheriting from class `Transferable`, a scenario's behavior can be extended to support transfer and reset of biomass at dedicated points during simulation.

Slots

`transfer.times` numeric, vector of time points at which transfers occur, e.g. `c(7, 10, 14)`
`transfer.interval` numeric, interval length until frond transfer to new medium
`transfer.biomass` numeric, amount of biomass transferred to new medium
`transfer.comp.biomass` character state variable which describes biomass
`transfer.comp.scaled` character vector of state variable which will be scaled 1:1 when biomass is modified, e.g. internal toxicant mass

Biomass transfer

Models supporting biomass transfer can be instructed to move a fixed amount of biomass to a new medium after a period of time. This feature replicates a procedure occurring in e.g. Lemna effect studies and may be necessary to recreate study results.

The biomass transfer feature assumes that always a fixed amount of biomass is transferred. Transfers can occur at any fixed point in time or in regular intervals. During a transfer, the biomass is reset to the transferred amount and additional compartments can be scaled 1:1 accordingly, to e.g. reflect the change in internal toxicant mass when biomass is modified. Transfer settings can be modified using `set_transfer()`.

Any transfer time point must also be an output time point. If a transfer occurs, simulation results of that time point will report the model state **before** the transfer. Be aware that in order to use transfers at regular intervals, the simulation must start at time point zero.

See Also

[set_transfer\(\)](#)

Other scenarios: [Algae-models](#), [DEB-models](#), [GUTS-RED-models](#), [Lemna-models](#), [Macrophyte-models](#), [Myriophyllum-models](#), [Scenarios](#)

Index

- * **DEB models**
 - DEB-models, 15
 - DEB_abj, 16
 - DEB_Daphnia, 18
- * **GUTS-RED models**
 - GUTS-RED-models, 30
 - GUTS_RED_IT, 31
 - GUTS_RED_SD, 33
- * **Lemna models**
 - Lemna-models, 38
 - Lemna_Schmitt, 39
 - Lemna_SETAC, 43
- * **Myriophyllum models**
 - Myrio, 54
 - Myrio_log, 56
 - Myriophyllum-models, 56
- * **algae models**
 - Algae-models, 4
 - Algae_Simple, 5
 - Algae_TKTD, 6
 - Algae_Weber, 8
- * **datasets**
 - americamysis, 9
 - dmagna, 20
 - focusd1, 27
 - metsulfuron, 50
 - minnow_it, 51
 - minnow_sd, 52
 - Rsubcapitata, 65
 - Schmitt2013, 69
- * **macrophyte models**
 - Lemna_Schmitt, 39
 - Lemna_SETAC, 43
 - Macrophyte-models, 48
 - Myrio, 54
 - Myrio_log, 56
- * **scenarios**
 - Algae-models, 4
 - DEB-models, 15
 - GUTS-RED-models, 30
 - Lemna-models, 38
 - Macrophyte-models, 48
 - Myriophyllum-models, 56
 - Scenarios, 66
 - Transferable, 88
- Algae-class (Algae-models), 4
- Algae-models, 4, 28
- Algae_Simple, 4, 5, 7, 9
- Algae_Simple(), 4
- Algae_TKTD, 4, 6, 6, 9, 66
- Algae_TKTD(), 4, 6
- Algae_Weber, 4, 6, 7, 8
- Algae_Weber(), 4, 6
- AlgaeSimpleScenario, 5
- AlgaeSimpleScenario-class (Algae_Simple), 5
- AlgaeTKTDScenario, 6
- AlgaeTKTDScenario-class (Algae_TKTD), 6
- AlgaeWeberScenario, 8
- AlgaeWeberScenario-class (Algae_Weber), 8
- americamysis, 9
- Biomass-transfer (Transferable), 88
- C, 16–18
- cache_controls, 10
- cache_controls(), 67
- calibrate, 11
- calibrate, EffectScenario-method (calibrate), 11
- calibrate, list-method (calibrate), 11
- CalibrationSet, 11, 12, 13
- CalibrationSet-class, 15
- DEB-models, 15
- DEB_abj, 15, 16, 20
- DEB_abj(), 10

- DEB_Daphnia, [15, 17, 18](#)
- DEB_Daphnia(), [20](#)
- DebAbj, [17](#)
- DebAbj-class (DEB_abj), [16](#)
- DebDaphnia, [20](#)
- DebDaphnia-class (DEB_Daphnia), [18](#)
- DebScenario-class (DEB-models), [15](#)
- deSolve::forcings, [67, 72, 73, 86](#)
- deSolve::lsoda(), [82, 83](#)
- deSolve::ode(), [82, 83, 86](#)
- dmagna, [20](#)
- dose_response, [21](#)
- effect, [22](#)
- effect(), [10, 17, 19, 21, 24, 66–68, 70, 72–74](#)
- EffectScenario-class (Scenarios), [66](#)
- ePx, [23](#)
- ePx(), [66, 70, 74](#)
- ePx_mtw, [25](#)
- ExposureSeries, [26, 26, 59, 71, 72](#)
- ExposureSeries-class (ExposureSeries), [26](#)
- focusd1, [27](#)
- fx, [28](#)
- fx, Algae-method (fx), [28](#)
- fx, ANY-method (fx), [28](#)
- fx, GutsRedIt-method (fx), [28](#)
- fx, GutsRedSd-method (fx), [28](#)
- fx, Lemna-method (fx), [28](#)
- fx, Myriophyllum-method (fx), [28](#)
- get_model, [29](#)
- get_model, ANY-method (get_model), [29](#)
- get_model, EffectScenario-method (get_model), [29](#)
- get_model, list-method (get_model), [29](#)
- get_model, parameter_set-method (get_model), [29](#)
- get_model_name (get_model), [29](#)
- get_tag, [29](#)
- get_tag(), [78](#)
- get_tag, ANY-method (get_tag), [29](#)
- get_tag, EffectScenario-method (get_tag), [29](#)
- get_tag, list-method (get_tag), [29](#)
- get_tag, parameter_set-method (get_tag), [29](#)
- GUTS-RED-IT, [30](#)
- GUTS-RED-models, [28, 30, 51, 52, 87](#)
- GUTS-RED-SD, [30](#)
- GUTS_RED_IT, [31, 31, 34](#)
- GUTS_RED_SD, [31, 32, 33](#)
- GutsRedIt, [32](#)
- GutsRedIt-class (GUTS_RED_IT), [31](#)
- GutsRedSd, [33](#)
- GutsRedSd-class (GUTS_RED_SD), [33](#)
- import_exposure_text, [34](#)
- import_swash, [34](#)
- import_toxswa, [35](#)
- is_DEB, [35](#)
- is_GUTS, [36](#)
- is_GUTS_IT (is_GUTS), [36](#)
- is_GUTS_SD (is_GUTS), [36](#)
- is_Lemna, [37](#)
- is_Lemna(), [37](#)
- is_LemnaThreshold, [37](#)
- is_LemnaThreshold(), [37](#)
- is_scenario, [38](#)
- Lemna, [48](#)
- Lemna-class (Lemna-models), [38](#)
- Lemna-models, [4, 27, 28, 38, 43, 46, 50, 56, 69](#)
- Lemna_Schmitt, [39, 39, 46, 49, 56, 58](#)
- Lemna_Schmitt(), [38, 80](#)
- Lemna_SchmittThold (Lemna_Schmitt), [39](#)
- Lemna_SETAC, [39, 43, 43, 49, 56, 58](#)
- Lemna_SETAC(), [38, 54](#)
- LemnaSchmittScenario, [40](#)
- LemnaSchmittScenario-class (Lemna_Schmitt), [39](#)
- LemnaSetacScenario, [43](#)
- LemnaSetacScenario-class (Lemna_SETAC), [43](#)
- log_disable (log_enable), [46](#)
- log_enable, [46](#)
- log_envir, [47](#)
- log_msg, [47](#)
- log_scenarios, [48](#)
- Macrophyte-models, [28, 39, 43, 46, 48, 56](#)
- mae, [49](#)
- metsulfuron, [50](#)
- minnow_it, [51](#)
- minnow_sd, [52](#)
- morse, [51, 52, 53](#)
- Myrio, [43, 46, 49, 54, 56, 58](#)

- Myrio(), [56](#)
- Myrio_log, [43](#), [46](#), [49](#), [56](#), [56](#)
- Myrio_log(), [56](#)
- MyrioExpScenario, [54](#)
- MyrioExpScenario-class (Myrio), [54](#)
- MyrioLogScenario, [57](#)
- MyrioLogScenario-class (Myrio_log), [56](#)
- Myriophyllum, [48](#)
- Myriophyllum-class
 - (Myriophyllum-models), [56](#)
- Myriophyllum-models, [56](#)
- nmae (mae), [49](#)
- no_exposure, [59](#)
- no_exposure(), [26](#)
- nrmse (mae), [49](#)
- parameter_set, [29](#), [59](#), [60](#), [77](#)
- parameter_set-class (parameter_set), [59](#)
- pll_debug, [60](#)
- plot_epx, [61](#)
- plot_ppc, [62](#)
- plot_ppc_combi, [63](#)
- plot_sd, [63](#)
- pull_metadata, [65](#)
- rmse (mae), [49](#)
- Rsubcapitata, [65](#)
- scenario, [10](#), [11](#), [21](#), [25](#), [28](#), [72](#), [73](#), [82](#), [86](#)
- scenario (Scenarios), [66](#)
- Scenarios, [4](#), [6](#), [7](#), [9](#), [15](#), [31](#), [39](#), [43](#), [46](#), [49](#), [56](#), [58](#), [66](#), [88](#)
- scenarios, [29](#), [37](#), [41](#), [45](#), [65](#), [71](#), [73](#), [76](#), [78](#)
- scenarios (Scenarios), [66](#)
- ScenarioSequence, [70](#)
- ScenarioSequence-class (sequence), [69](#)
- Schmitt2013, [69](#)
- sequence, [69](#)
- sequence(), [13–15](#), [66](#)
- set_endpoints, [70](#)
- set_endpoints(), [17](#), [19](#), [67](#)
- set_exposure, [71](#)
- set_exposure(), [67](#), [78](#)
- set_exposure, ANY, ANY-method
 - (set_exposure), [71](#)
- set_exposure, EffectScenario, data.frame-method
 - (set_exposure), [71](#)
- set_exposure, EffectScenario, ExposureSeries-method
 - (set_exposure), [71](#)
- set_exposure, EffectScenario, list-method
 - (set_exposure), [71](#)
- set_exposure, list, ANY-method
 - (set_exposure), [71](#)
- set_exposure, list, list-method
 - (set_exposure), [71](#)
- set_forcings, [73](#)
- set_forcings(), [67](#)
- set_forcings, EffectScenario-method
 - (set_forcings), [73](#)
- set_forcings, list-method
 - (set_forcings), [73](#)
- set_init, [74](#)
- set_init(), [16](#), [18](#), [40](#), [66](#)
- set_init, EffectScenario-method
 - (set_init), [74](#)
- set_init, vector-method (set_init), [74](#)
- set_moa (set_mode_of_action), [75](#)
- set_mode_of_action, [75](#)
- set_mode_of_action(), [17](#), [19](#)
- set_noexposure, [76](#)
- set_noexposure(), [59](#), [71](#)
- set_param, [76](#)
- set_param, EffectScenario, parameter_set-method
 - (set_param), [76](#)
- set_param, EffectScenario, vector-method
 - (set_param), [76](#)
- set_param, list, parameter_set-method
 - (set_param), [76](#)
- set_param, list, vector-method
 - (set_param), [76](#)
- set_param, ScenarioSequence, parameter_set-method
 - (set_param), [76](#)
- set_param, ScenarioSequence, vector-method
 - (set_param), [76](#)
- set_tag, [77](#)
- set_tag(), [30](#)
- set_times, [78](#)
- set_times(), [67](#), [72](#)
- set_transfer, [79](#)
- set_transfer(), [4](#), [39](#), [42](#), [46](#), [49](#), [55](#), [58](#), [88](#)
- set_transfer, ANY-method (set_transfer), [79](#)
- set_transfer, Transferable-method
 - (set_transfer), [79](#)
- set_window, [81](#)
- set_window(), [68](#), [82](#)
- simulate, [81](#)

`simulate()`, [19](#), [22](#), [42](#), [67](#), [72](#), [73](#), [78](#), [87](#)
`simulate`, `EffectScenario`-method
 (`simulate`), [81](#)
`simulate`, `ScenarioSequence`-method
 (`simulate`), [81](#)
`simulate`, `Transferable`-method
 (`simulate`), [81](#)
`simulate_batch`, [84](#)
`smape (mae)`, [49](#)
`solver`, [84](#)
`solver`, `AlgaeSimpleScenario`-method
 (`solver`), [84](#)
`solver`, `AlgaeTKDScenario`-method
 (`solver`), [84](#)
`solver`, `AlgaeWeberScenario`-method
 (`solver`), [84](#)
`solver`, `ANY`-method (`solver`), [84](#)
`solver`, `DebAbj`-method (`solver`), [84](#)
`solver`, `DebDaphnia`-method (`solver`), [84](#)
`solver`, `GutsRedIt`-method (`solver`), [84](#)
`solver`, `GutsRedSd`-method (`solver`), [84](#)
`solver`, `LemnaSchmittScenario`-method
 (`solver`), [84](#)
`solver`, `LemnaSetacScenario`-method
 (`solver`), [84](#)
`solver`, `MyrioExpScenario`-method
 (`solver`), [84](#)
`solver`, `MyrioLogScenario`-method
 (`solver`), [84](#)
`sse (mae)`, [49](#)
`stats::approxfun()`, [86](#)
`stats::optim()`, [12](#)
`survival`, [87](#)
`survival()`, [31–33](#)

`Transferable`, [4](#), [6](#), [7](#), [9](#), [15](#), [31](#), [39](#), [43](#), [46](#),
 [49](#), [56](#), [58](#), [68](#), [88](#)
`Transferable`-class (`Transferable`), [88](#)