Package: microxanox (via r-universe)

October 12, 2024

Version 0.9.3

Date 2023-11-13

Title Oxic-Anoxic Regime Shifts in Microbial Communities

Description Model to simulate a three functional group system with four chemical substrates using a set of ordinary differential equations. Simulations can be run individually or over a parameter range, to find stable states. The model features multiple species per functional group, where the number is only limited by computational constraints. The R package is constructed in such a way, that the results contain the input parameter used, so that a saved results can be loaded again and thesimulation be repeated.

Maintainer Owen L. Petchey <Owen.Petchey@ieu.uzh.ch>

URL https://github.com/UZH-PEG/microxanox/

BugReports https://github.com/UZH-PEG/microxanox/issues

License MIT + file LICENSE

Depends R (>= 4.1.0),

Imports magrittr, tibble, ggplot2, patchwork, grDevices, stats, mgcv, deSolve, dplyr, tidyr, stringr, multidplyr, ggpubr, DescTools

Suggests knitr, rmarkdown, rootSolve, tidyverse, testthat (>= 3.0.0), roxyglobals

RoxygenNote 7.3.1

Encoding UTF-8

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(roclets = c(``collate'', ``namespace'', ``rd'', ``roxyglobals::global_roclet''))

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Repository https://uzh-peg.r-universe.dev

RemoteUrl https://github.com/UZH-PEG/microxanox

RemoteRef HEAD

RemoteSha 00e345f078d11f7e8c5322a1e496ffc8b7d40a9c

Contents

microxanox-package . . . 3
add_strain_var . . . 3
bushplus_dynamic_model . . . 4
event_definition_1 . . . 5
event_definition_2 . . . 6
event_definition_symmetric . . . 7
get_final_states_a_N . . . 8
get_hysteresis_max . . . 8
get_hysteresis_min . . . 9
get_hysteresis_range . . . 9
get_hysteresis_total . . . 10
get_nonlinearity . . . 10
get_ssbyaN_parameter . . . 11
get_stability_measures . . . 11
get_symmetry_measurements . . . 12
growth1 . . . 13
growth2 . . . 13
inhibition . . . 14
new_CB_strain_parameter . . . 14
new_initial_state . . . 15
new_PB_strain_parameter . . . 16
new_replication_ssfind_parameter . . . 17
new_replication_ssfind_results . . . 17
new_runsim_parameter . . . 18
new_runsim_results . . . 19
new_SB_strain_parameter . . . 19
new_strain_parameter . . . 20
new_temporal_ssfind_results . . . 22
plot_dynamics . . . 22
plot_dynamics_symmetric . . . 23
plot_symmetry_measures . . . 24
plot_temporal_ss . . . 24
plot_trajectory_symmetry . . . 25
plot_trajectory_symmetry_compact . . . 25
run_replication_ssfind . . . 26
run_simulation . . . 27
run_simulation_symmetric . . . 27
run_temporal_ssfind . . . 28
run_temporal_ssfind_experiment . . . 28
run_temporal_ssfind_symmetric . . . 29
set_diffusivities . . . 30

set_temporal_ssfind_initial_state 30
symmetric_bushplus_dynamic_model 31

Index **33**

`microxanox-package` *R/microxanox: Microbial oxic and anoxic ecosystem simulations*

Description

Ecosystems containing microbes can be oxic (oxygen present) or anoxic (oxygen absent). Under some conditions these states can be both stable even with identical biotic and abiotic circumstances, i.e. they can be alternate stable states, implying history alone can determine the present state. The biological reason is that the microbes can alter the environmental conditions in a way that favour themselves, while making them less suitable for other organisms. This creates mutual inhibition. Which state occurs also depends on numerous abiotic and biotic factors. This package facilitates simulation studies of the influence of several of these factors, and is based on work described in (Bush et al) 2017 (DOI: 10.1038/s41467-017-00912-x)

Details

Vignettes include:

- A user guide. - A reproduction of some results of Bush et al 2017, on which the simulation is based. It shows the alternate stable states. It also extends the original study by adding temporal forcing and therefore temporal switching between states, and also effects of biotic composition on the response to environmental change.

`add_strain_var` *Create variability in strain parameters*

Description

Add variability as defined in the argument `variability` to the variables as follows:

- `strain_parameter$CB$g_max_CB <- variability(strain_parameter$CB$g_max_CB, CB_var_gmax)`
- `strain_parameter$CB$h_SR_CB <- variability(strain_parameter$CB$h_SR_CB, CB_var_h)`
- `strain_parameter$SB$g_max_SB <- variability(strain_parameter$SB$g_max_SB, SB_var_gmax)`
- `strain_parameter$SB$h_O_SB <- variability(strain_parameter$SB$h_O_SB, SB_var_h)`
- `strain_parameter$PB$g_max_PB <- variability(strain_parameter$PB$g_max_PB, PB_var_gmax)`
- `strain_parameter$PB$h_O_PB <- variability(strain_parameter$PB$h_O_PB, PB_var_h)`

Usage

```
add_strain_var(
  strain_parameter,
  CB_var_gmax = 0,
  CB_var_h = 0,
  SB_var_gmax = 0,
  SB_var_h = 0,
  PB_var_gmax = 0,
  PB_var_h = 0,
  variability = function(strain_parameter, var) {
     strain_parameter * 2^(seq(-var,
    var, length = length(strain_parameter)))
 }
)
```

Arguments

strain_parameter	object of class strain_parameter
CB_var_gmax	the argument var for the function variability for the variable strain_parameterCBg_max_CB
CB_var_h	the argument var for the function variability for the variable strain_parameterCBh_SR_CB
SB_var_gmax	the argument var for the function variability for the variable strain_parameterCBg_max_SB
SB_var_h	the argument var for the function variability for the variable strain_parameterCBh_O_SB
PB_var_gmax	the argument var for the function variability for the variable strain_parameterCBg_max_PB
PB_var_h	the argument var for the function variability for the variable strain_parameterCBh_O_PB
variability	function of which takes two arguments, i.e. strain_parameter and var and returns an object of the class strain parameter. The function will be applied to add variability, of amount var to the above mentioned variables.

Value

the value of strain_parameter with added strain variability

Examples

```
add_strain_var(new_strain_parameter(n_CB = 3, n_PB = 3, n_SB = 3), 2)
```

---

bushplus_dynamic_model

The rate equations, as published in the Bush et al 2017 R*hrefhttps://doi.org/10.1093/clinchem/39.5.766doi:10.1093/clinchem/39.5.766 paper, but with forcing of oxygen diffusivity* a_0 *potential added, and the possibility to simulate multiple strains per functional group*

Description

The rate equations, as published in the Bush et al 2017 doi:10.1093/clinchem/39.5.766 paper, but with forcing of oxygen diffusivity `a_0` potential added, and the possibility to simulate multiple strains per functional group

Usage

```
bushplus_dynamic_model(t, state, parameters, log10a_forcing_func, ...)
```

Arguments

`t`	The current time in the simulation
`state`	A vector containing the current (named) values of each state variable
`parameters`	An object of class `runsim_parameter` as returned by 'new_runsim_parameter()"
`log10a_forcing_func`	function to change oxygen diffusivity `a` depending on `t`
`...`	not used. Needed to catch additional parameters.

Value

a list containing two elements, namely the rate of change of the strains, and also the current values of oxygen diffusivity `a`.

`event_definition_1` *Event definition for the simulation.*

Description

This function contains events that can alter the state variables. In this event definition, densities are given a floor of 1 at every event.

Usage

```
event_definition_1(
  times,
  state,
  parms,
  log10a_forcing_func,
  noise_sigma,
  minimum_abundances
)
```

Arguments

times	the time points in the simulation when events happen
state	current state variable values
parms	object of class strain_parameter
log10a_forcing_func	function to change oxygen diffusivity a depending on t
noise_sigma	value of variation added to SO, SR, O, and P using the formula rnorm(1, 0, noise_sigma * X) * noise_sigma * X
minimum_abundances	minimum abundances for CB, PB and SB. if the values are lower, they will be set to these minimum_abundances

Value

the state vector, i.e. a vector containing the state variables.

---

event_definition_2 *Event definition for the simulation.*

Description

This function contains events that can alter the state variables. In this event definition, the abundances of all functional groups have 1 added to their density at each event.

Usage

```
event_definition_2(
  times,
  state,
  parms,
  log10a_forcing_func,
  noise_sigma,
  minimum_abundances
)
```

Arguments

times	the time points in the simulation when events happen
state	current state variable values
parms	object of class strain_parameter
log10a_forcing_func	function to change oxygen diffusivity a depending on t
noise_sigma	value of variation added to SO, SR, O, and P using the formula rnorm(1, 0, noise_sigma * X) * noise_sigma * X
minimum_abundances	minimum abundances for CB, PB and SB. if the values are lower, they will be set to these minimum_abundances

Value

the state vector, i.e. a vector containing the state variables.

`event_definition_symmetric` *Event definition for the symmetric simulation.*

Description

This function contains events that can alter the state variables. In this event definition, the abundances of all functional groups have 1 added to their density at each event.

Usage

```
event_definition_symmetric(
  times,
  state,
  parms,
  log10aO_forcing_func,
  log10aS_forcing_func,
  noise_sigma,
  minimum_abundances
)
```

Arguments

`times`	the time points in the simulation when events happen
`state`	current state variable values
`parms`	object of class `strain_parameter`
`log10aO_forcing_func`	function to change oxygen diffusivity `aO` depending on `t`
`log10aS_forcing_func`	function. to change sulfide diffusivity `aS` depending on `t`
`noise_sigma`	value of variation added to `SO`, `SR`, `O`, and `P` using the formula `rnorm(1, 0, noise_sigma * X) * noise_sigma * X`
`minimum_abundances`	minimum abundances for CB, PB and SB. if the values are lower, they will be set to these `minimum_abundances`

Value

the state vector, i.e. a vector containing the state variables.

get_final_states_a_N *Get the steady state solutions for a set of oxygen diffusivity and initial states.*

Description

Can be used directly. Also can be used indirectly, via the function `run_replication_ssfind_parameter()`

Usage

```
get_final_states_a_N(x, parameter)
```

Arguments

`x`	A vector of oxygen diffusivity and initial conditions.
`parameter`	object of class `replication_ssfind_parameter` as generated by `new_replication_ssfind_paramete`

Value

A vector of steady states

get_hysteresis_max *Get the maximum of environmental conditions for which alternate stable states exist*

Description

Get the maximum of environmental conditions for which alternate stable states exist

Usage

```
get_hysteresis_max(up, down, a, threshold_diff)
```

Arguments

`up`	State variable values as the environmental condition increases
`down`	State variable values as the environmental condition decreases
`a`	An environmental driver, here it is usually oxygen diffusivity
`threshold_diff`	Amount of different between up and down states to qualify as alternate stable states

Value

A numeric value, which is the extent of the range of conditions for which alternate stable states exist.

get_hysteresis_min *Get the minimum of environmental conditions for which alternate stable states exist*

Description

Get the minimum of environmental conditions for which alternate stable states exist

Usage

```
get_hysteresis_min(up, down, a, threshold_diff)
```

Arguments

up	State variable values as the environmental condition increases
down	State variable values as the environmental condition decreases
a	An environmental driver, here it is usually oxygen diffusivity
threshold_diff	Amount of different between up and down states to qualify as alternate stable states

Value

A numeric value, which is the extent of the range of conditions for which alternate stable states exist.

get_hysteresis_range *Get the range of environmental conditions for which alternate stable states exist*

Description

Get the range of environmental conditions for which alternate stable states exist

Usage

```
get_hysteresis_range(up, down, a, threshold_diff)
```

Arguments

up	State variable values as the environmental condition increases
down	State variable values as the environmental condition decreases
a	An environmental driver, here it is usually oxygen diffusivity
threshold_diff	Amount of different between up and down states to qualify as alternate stable states

Value

A numeric value, which is the extent of the range of conditions for which alternate stable states exist.

---

get_hysteresis_total *Get the total hysteresis of a system variable.*

Description

Get the total hysteresis of a system variable.

Usage

```
get_hysteresis_total(up, down)
```

Arguments

up	State variable values as the environmental condition increases
down	State variable values as the environmental condition decreases

Value

The total hysteresis.

---

get_nonlinearity *Get the amount of non-linearity in a state-environment relationship.*

Description

Method described in Emancipator & Knoll (1993) Clinical Chemistry 39, 766-772, doi:10.1093/clinchem/39.5.766

Usage

```
get_nonlinearity(x, y)
```

Arguments

x	Environmental driver state variable
y	Ecosystem state variable

Value

Measure of non-linearity

get_ssbyaN_parameter *Extract* runsim_parameter *from* replication_ssfind_parameter

Description

Extract the parameter to run an individual simulation from either a parameter set for a steady state determination (run_replication_ssfind_parameter()) or the result set returned.

Usage

```
get_ssbyaN_parameter(result, i)
```

Arguments

result	object of class replication_ssfind_parameter as generated by new_replication_ssfind_paramete
i	Index of the individual simulation to extract the parameter for

Value

an object of class replication_ssfind_parameter

get_stability_measures

Get various measures of the stability

Description

The measures include non-linearity and hysteresis measures, of an ecosystem response to environmental change. Takes steady state data as the input.

Usage

```
get_stability_measures(ss_object, threshold_diff_log10scale, ...)

## S3 method for class 'replication_ssfind_result'
get_stability_measures(ss_object, threshold_diff_log10scale = 3, ...)

get_stability_measures_replication_ssfind_result(ss_object)

## S3 method for class 'temporal_ssfind_result'
get_stability_measures(ss_object, threshold_diff_log10scale = 3, ...)

get_stability_measures_temporal_ssfind_result(ss_object)
```

Arguments

ss_object	object from which to calculate the stability measures. At the moment replication_ssfind_result or data.frame.
threshold_diff_log10scale	Amount of different between up and down states to qualify as alternate stable states, on log10 scale
...	additional arguments for methods

Value

A data frame of stability measures of each state variable

---

get_symmetry_measurements

Computes measures that make comparisons between collapse and recovery trajectory and between antagonistic environmental variables possible.

Description

Computes measures that make comparisons between collapse and recovery trajectory and between antagonistic environmental variables possible.

Usage

```
get_symmetry_measurements(res)
```

Arguments

res	Results from run_temporal_ssfind() or from run_temporal_ssfind_symmetric()

Value

returns a frame containing symmetry measures for each variable, such as hysteresis area, shift magnitudes or distance between TP, as well as TP themselfes with their according state.

growth1 *Growth rate function on one resource X*

Description

Growth rate function on one resource X

Usage

```
growth1(X, g_max, k_X)
```

Arguments

X	Concentration of resource X
g_max	Maximum growth rate
k_X	Half saturation constant for resource X

Value

Growth rate

growth2 *Growth rate function on two resources X and Y*

Description

Growth rate function on two resources X and Y

Usage

```
growth2(X, Y, g_max, k_X, k_Y)
```

Arguments

X	Concentration of resource X
Y	Concentration of resource Y
g_max	Maximum growth rate
k_X	Half saturation constant for resource X
k_Y	Half saturation constant for resource Y

Value

Growth rate

inhibition *Growth inhibition function*

Description

Growth inhibition function

Usage

```
inhibition(X, h_X)
```

Arguments

X	Concentration of substance X
h_X	Concentration of substance X at which the inhibition factor is 0.5 (i.e. the concerned rate is halved)

Value

Inhibition factor

new_CB_strain_parameter *Create CB strain parameter values*

Description

Create CB strain parameter values

Usage

```
new_CB_strain_parameter(n = 1, values = "bush")
```

Arguments

n	number of strains
values	Allowed values are: • "bush": default values from Bush et al (2017) doi:10.1038/s41467017-00912x will be used • "NA": all parameter will be set to NA. Usable for e.g. setting own parameter

Value

object of class `CB_strain_parameter`. The object contains a `data.frame` with 8 columns and `n` rows. The columns are:

columns:

- strain_name: the name of the strain
- g_max_CB:
- k_CB_P:
- h_SR_CB:
- y_P_CB:
- Pr_CB:
- m_CB:
- i_CB:

`new_initial_state` *Create initial state of the system, selecting them from various preset options.*

Description

Create initial state of the system, selecting them from various preset options.

Usage

```
new_initial_state(n_CB = 1, n_PB = 1, n_SB = 1, values = "bush_anoxic_fig2ab")
```

Arguments

`n_CB`	number of CB strains
`n_PB`	number of PB strains
`n_SB`	number of SB strains
`values`	Allowed values are: • `"bush_anoxic_fig2ab"`: initial conditions for Bush et al 2017 figure 2 a & b • `"bush_oxic_fig2cd"`: initial conditions for Bush et al 2017 figure 2 c & d • `"bush_ssfig3"`: initial conditions for Bush et al 2017 figure e • `"NA"`: all initial values set to `NA`. Usable for e.g. setting own initial state

Value

initial state `vector` aof the system s part of the strain_starter

`new_PB_strain_parameter` *Create PB strain parameter*

Description

Create PB strain parameter

Usage

```
new_PB_strain_parameter(n = 1, values = "bush")
```

Arguments

`n`	number of strains
`values`	Allowed values are: • `"bush"`: default values from Bush et al (2017) doi:10.1038/s41467017-00912x will be used • `"NA"`: all parameter will be set to `NA`. Usable for e.g. setting own parameter

Value

object of class `PB_strain_parameter`. The object contains a `data.frame` with 8 columns and `n` rows. The columns are:

columns:

- strain_name: the name of the strain
- g_max_PB:
- k_PB_SR:
- k_PB_P:
- h_O_PB:
- y_SR_PB:
- y_P_PB:
- m_PB:
- i_CB:

`new_replication_ssfind_parameter`
Create parameter set to run a set of simulations to find stable states. Is passed to the function `run_replication_ssfind_parameter()` *to run such a set of simulations.*

Description

Create parameter set to run a set of simulations to find stable states. Is passed to the function `run_replication_ssfind_parameter()` to run such a set of simulations.

Usage

```
new_replication_ssfind_parameter(...)
```

Arguments

`...`	named parameter for the simulation to be set. An error will be raised, if they are not part of the parameter set.

Value

object of the class `replication_ssfind_parameter`. This object of class `replication_ssfind_parameter` is identical to an object of class `runsim_parameter` plus an additional field `ss_expt` which contains a `data.frame` with columns named

- `N_CB`
- `N_PB`
- `N_SB`
- `a_0` which contain the initial states. If more than one strain is present, the same initial state is assumed for each.

`new_replication_ssfind_results`
Create object of type `replication_ssfind_result` *which is returned by the function* `run_replication_ssfind()`.

Description

Create object of type `replication_ssfind_result` which is returned by the function `run_replication_ssfind()`.

Usage

```
new_replication_ssfind_results(parameter, result)
```

Arguments

parameter	object of class replication_ssfind_parameter which has been used to run the simulation
result	a data.frame containing the results of the simulation

Value

object of the class replication_ssfind_result. This object of class replication_ssfind_result is identical to an object of class replication_ssfind_parameter plus an additional field result which contains the result of the simulation.

---

new_runsim_parameter *Create parameter set with which to run a simulation.*

Description

Create parameter set with which to run a simulation.

Usage

```
new_runsim_parameter(...)
```

Arguments

...	named parameter for the simulation to be set. An error will be raised, if they are not part of the parameter set.

Value

parameter object of the class runsim_parameter. The object contains the following elements:

- dynamic_model : the dynamic model to be used. At the moment, only bushplus_dynamic_model is implemented. Fur further info, see the documen tation of bushplus_dynamic_model.
- event_definition : A function which alters the state variables. At the moment only event_definition_1() is included. User defined functions with the same signature can be used.
- strain_parameter : object of class strain_parameter as returned by new_strain_parameter()
- event_interval : interval, in timesteps, in which the event occurs"
- noise_sigma : value of variation added to S0, SR, 0, and P during the event
- minimum_abundances : Minimum abundances. Smaller abundances will be set to this value duting event_definition_1().
- sim_duration : duration of the simulation
- sim_sample_interval: interval, at which the simulation will be sampled
- log10a_series : A vector of values of log10 oxygen diffusivity parameter at which stable states will be found.

- asym_factor: : For symmetric simulations only: enables manipulating aS forcing in asymmetric manner to decrease (<1) or increase (>1) stress on cyanobacteria.
- solver_method : Used for the solver. Default is "radau". For other options, see the documentatioom of odeSolve::ode.

`new_runsim_results` *Create object of type* `runsim_result` *which is returned by the function* `run_sim()`.

Description

Create object of type `runsim_result` which is returned by the function `run_sim()`.

Usage

```
new_runsim_results(parameter, result)
```

Arguments

`parameter`	object of class `runsim_parameter` which has been used to run the simulation
`result`	a `data.frame` containing the results of the simulation

Value

object of the class `runsim_result`. This object of class `runsim_result` is identical to an object of class `runsim_parameter` plus an additional field `result` which contains the result of the simulation.

`new_SB_strain_parameter` *Create new* `SB_strain_parameter` *object*

Description

Create new `SB_strain_parameter` object

Usage

```
new_SB_strain_parameter(n = 1, values = "bush")
```

Arguments

`n`	number of strains
`values`	Allowed values are: • "bush": default values from Bush et al (2017) doi:10.1038/s41467017-00912x will be used • "NA": all parameter will be set to NA. Usable for e.g. setting own parameter

Value

object of class `SB_strain_parameter`. The object contains a `data.frame` with 8 columns and `n` rows. The columns are:

columns:

- strain_name: the name of the strain
- g_max_SB:
- k_PB_SO:
- k_SB_P:
- h_O_SB:
- y_SO_SB:
- y_PB_SB:
- m_SB:
- i_SB:

`new_strain_parameter` *Returns object of class* `strain_parameter`

Description

Creates a set of parameters and starting conditions for a simulation. This function assists with the initialization of a simulation, by providing various reference sets of parameter values and initial conditions. The default is to create the parameter set used in Bush et al. (2017) doi:10.1093/clinchem/39.5.766 and the anoxic favorable initial conditions used in the simulations for Figure 2a&b of that publication.

Usage

```
new_strain_parameter(
  n_CB = 1,
  values_CB = "bush",
  n_PB = 1,
  values_PB = "bush",
  n_SB = 1,
  values_SB = "bush",
  values_other = "bush",
  values_initial_state = "bush_anoxic_fig2ab"
)
```

Arguments

n_CB	number of CB strains, default 1.
values_CB	Allowed values are: • "bush": default values from Bush et al (2017) doi:10.1093/clinchem/39.5.766 will be used for the parameter for CB • "NA": all initial values set to NA. Usable for e.g. setting own parameter
n_PB	number of PB strains, default 1.
values_PB	Allowed values are: • "bush": default values from Bush et al (2017) doi:10.1093/clinchem/39.5.766 will be used for the parameter for PB • "NA": all parameter will be set to NA. Usable for e.g. setting own parameter
n_SB	number of SB strains, default 1.
values_SB	Allowed values are: • "bush": default values from Bush et al (2017) doi:10.1093/clinchem/39.5.766 will be used for the parameter for SB • "NA": all parameter will be set to NA. Usable for e.g. setting own parameter
values_other	Allowed values are: • "bush": default values from Bush et al (2017) doi:10.1093/clinchem/39.5.766will be used for additional parameter • "NA": all parameter will be set to NA. Usable for e.g. setting own parameter values to be used for other parameter or "bush", in which case the default from Bush et al (2017) will be used.
values_initial_state	values to be used for initial values or "bush_anoxic" or "bush_oxic", in which case the default from Bush et al (2017) doi:10.1093/clinchem/39.5.766will be used.

Value

Object of class strain_parameter. The object contains the following fields: The additional parameters are:

Strain parameter:

- CB: strain parameter from Cyano Bacteria
- PB: strain parameter for the Phototrophic bacteria
- SB: strain parameter for the Sulphur bacteria

substrate diffusivity:

- a_S: substrate diffusivity of sulphur <- 0.001
- a_O: substrate diffusivity of oxygen <- 8e-4
- a_P: substrate diffusivity of phosphorous <- 0.01

background substrate concentration:

- back_SR: background substrate concentration of XXX <- 300
- back_SO: background substrate concentration of XXX <- 300
- back_O: background substrate concentration of oxygen <- 300
- back_P: background substrate concentration of phosphorous <- 9.5

oxidisation rate of reduced sulphur:

- c: <- 4e-5

`new_temporal_ssfind_results` *Create object of type* `temporal_ssfind_result` *which is returned by the function* `run_temporal_ssfind_parameter()`.

Description

Create object of type `temporal_ssfind_result` which is returned by the function `run_temporal_ssfind_parameter()`.

Usage

```
new_temporal_ssfind_results(parameter = NULL, result)
```

Arguments

`parameter`	not used - for consistency.
`result`	a `data.frame` containing the results of the simulation

Value

the `result` object.

`plot_dynamics` *Plot the dynamics of a model run*

Description

This is a convenience function to plot the dynamics of a model run, with strains within functional groups displayed.

Usage

```
plot_dynamics(simulation_result, every_n = 1)
```

Arguments

simulation_result	Object returned by the run_simulation function
every_n	Plot data of every other n sample.

Value

returns the ggplot object of the plot. If it is assigned to a variable, the plot needs to be plotted, otherwise it is plotted.

---

plot_dynamics_symmetric

Plot the dynamics of a model run of a symmetric parameter set

---

Description

This is a convenience function to plot the dynamics of a model run particularly for a symmetric model with strains within functional groups displayed.

Usage

```
plot_dynamics_symmetric(simulation_result, every_n = 1, plot_a = FALSE)
```

Arguments

simulation_result	Object returned by the run_simulation function
every_n	Plot data of every other n sample.
plot_a	if TRUE, diffusivities will be plotted in lower pane, default is FALSE.

Value

returns the ggplot object of the plot. If it is assigned to a variable, the plot needs to be plotted, otherwise it is plotted.

plot_symmetry_measures

Visualizes the symmetry measures obtained by the function `get_symmetry_measures()`

Description

Visualizes the symmetry measures obtained by the function `get_symmetry_measures()`

Usage

```
plot_symmetry_measures(res, species)
```

Arguments

`res`	Results from `run_temporal_ssfind()` or from `run_temporal_ssfind_symmetric()`
`species`	Environmental varibale of which the symmetry measures shoudl be plotted

Value

ggplot object, with shifts visualized as arrows and hysteresis area ribbon

plot_temporal_ss

Plot the temporal dynamics of a model run of a symmetric parameter set

Description

This is a convenience function to plot the dynamics of a model run particularly for a symmetric model with strains within functional groups displayed.

Usage

```
plot_temporal_ss(temporal_results)
```

Arguments

`temporal_results`	Results from `run_temporal_ssfind()` or from `run_temporal_ssfind_symmetric()`

Value

returns the ggplot object of the plot. If it is assigned to a variable, the plot needs to be plotted, otherwise it is plotted.

plot_trajectory_symmetry *Visualizes the symmetry of antagonistic collapse or recovery trajectories, symmetry measures of shift are computed by the function* `get_symmetry_measurements()`

Description

Visualizes the symmetry of antagonistic collapse or recovery trajectories, symmetry measures of shift are computed by the function `get_symmetry_measurements()`

Usage

```
plot_trajectory_symmetry(
  res,
  trajectory = "recovery",
  typ = "substrate",
  plot_log10 = FALSE
)
```

Arguments

`res`	Results from `run_temporal_ssfind()` or from `run_temporal_ssfind_symmetric()`
`trajectory`	Trajectory to be plotted. Either 'collapse' or 'recovery'.'.
`typ`	Type of environmental variable. Either 'substrate' or 'bacteria'.
`plot_log10`	Logical. If `TRUE`, logarithmic plot of the data is done.

Value

ggplot object, with antagonistic shifts of related trajectory visualized as arrows.

plot_trajectory_symmetry_compact *Visualizes the symmetry of upards and downwards trajectories in the same figure, symmetry measures of shift are computed by the function* `get_symmetry_measurements()`

Description

Visualizes the symmetry of upards and downwards trajectories in the same figure, symmetry measures of shift are computed by the function `get_symmetry_measurements()`

Usage

```
plot_trajectory_symmetry_compact(res, typ = "substrate", plot_log10 = FALSE)
```

Arguments

res	Results from run_temporal_ssfind() or from run_temporal_ssfind_symmetric()
typ	Type of environmental variable. Either ‘substrate‘ or ‘bacteria‘.
plot_log10	Logical. If TRUE, logarithmic plot of the data is done.

Value

ggplot object, with antagonistic shifts of related trajectory visualized as arrows.

---

run_replication_ssfind

Run simulations to determine the steady states by the replication method.

Description

Function to get the steady states for combinations of a (oxygen diffusivity) and initial states. This function is multithreaded, and the value of mc.cores determines the umber of parallel threads.

Usage

```
run_replication_ssfind(parameter, mc.cores = getOption("mc.cores", 0))
```

Arguments

parameter	object of class replication_ssfind_parameter as generated by new_replication_ssfind_paramete
mc.cores	the number of cores to be used. If 0, the old sequential version is used. The default is read from the option mc.cores, i.e. using getOption("mc.cores", 0).

Value

Processed data about steady states

run_simulation *Run the simulation*

Description

This function takes the `parameter` object and runs a simulation based on these. It returns an object of class `runsim_result` which contains an additional entry, i.e. `result` which contains the results of the simulation. The simulation can be re-run using the returned object as input `parameter`.

Usage

```
run_simulation(parameter)
```

Arguments

`parameter`	an object of class `runsim_parameter` as returned by `new_runsim_parameter()`.

Value

an object of class `runsim_result`, obtained from running the simulation as defined in `parameter`.‘

run_simulation_symmetric *Run the symmetric simulation*

Description

This function takes the `parameter` object and runs a simulation based on these. It returns an object of class `runsim_result` which contains an additional entry, i.e. `result` which contains the results of the simulation. The simulation can be re-run using the returned object as input `parameter`.

Usage

```
run_simulation_symmetric(parameter)
```

Arguments

`parameter`	an object of class `runsim_parameter` as returned by `new_runsim_parameter()`. Needs to hold key sym_axis.

Value

an object of class `runsim_result`, obtained from running the simulation as defined in `parameter`.‘

`run_temporal_ssfind` *Used to find the stable states for a parameter set by increasing and decreasing the oxygen diffusivity in a stepwise fashion. If increasing* `parameter$sim_duration` *while keeping the length of* `parameter$log10_series` *doesn't change response dynanimcs, stable states have been found.*

Description

Used to find the stable states for a parameter set by increasing and decreasing the oxygen diffusivity in a stepwise fashion. If increasing `parameter$sim_duration` while keeping the length of `parameter$log10_series` doesn't change response dynanimcs, stable states have been found.

Usage

```
run_temporal_ssfind(parameter)
```

Arguments

`parameter`	an object of class `runsim_parameter` as returned by `new_runsim_parameter()`.

Value

A data frame of final states and oxygen diffusivity values

`run_temporal_ssfind_experiment` *Run a stable state finding experiment via the temporal method (e.g. get the stable states for different levels of oxygen diffusivity when oxygen diffusivity is varied in a stepwise fashion).*

Description

Calls the function `run_temporal_ssfind` for each parameter set.

Usage

```
run_temporal_ssfind_experiment(
  parameter,
  var_expt,
  total_initial_abundances,
  cores = 1
)
```

Arguments

parameter	an object of class runsim_parameter as returned by new_runsim_parameter().
var_expt	An object that describes different levels of diversity that should be examined. This object is **not** created in the 'microxanox' package.
total_initial_abundances	An object containing the total abundance in each functional group.
cores	Number of cores to use. If more than one, then multidplyr is used

Value

an tibble object containing the stable state result, as well as the simulation parameters.

---

run_temporal_ssfind_symmetric

Used to find the stable states for a parameter set by increasing and decreasing the oxygen diffusivity in a stepwise fashion. If increasing parameter$sim_duration *while keeping the length of* parameter$log10_series *doesn't change response dynanimcs, stable states have been found.*

Description

Used to find the stable states for a parameter set by increasing and decreasing the oxygen diffusivity in a stepwise fashion. If increasing parameter$sim_duration while keeping the length of parameter$log10_series doesn't change response dynanimcs, stable states have been found.

Usage

```
run_temporal_ssfind_symmetric(parameter)
```

Arguments

parameter	an object of class runsim_parameter as returned by new_runsim_parameter().

Value

A data frame of final states, oxygen and sulfide diffusivity values

set_diffusivities *Used to update the asymmetry factor with logaO_series & log10aS_series accordingly*

Description

The `set_diffusivities` enables manipulating `aS` forcing in asymmetric manner to decrease (<1) or increase (>1) stress on cyanobacteria. If the vector in `parameter$log10a_series` has multiple local minima, maxima respectively, it is directly mirrored. ATTENTION: Mean value of the series has to be the axis at which the vector is mirrored!

Usage

```
set_diffusivities(param, asym_factor = 1)
```

Arguments

`param`	:parameter set of type `runsim_parameter`
`asym_factor`	:asym_factor to update `runsim_parameter`

Value

updated parameterset

set_temporal_ssfind_initial_state *Convenience function for setting initial states during the process of finding stable states using the function* `run_temporal_ssfind_experiment` *Should only be used internally.*

Description

Convenience function for setting initial states during the process of finding stable states using the function `run_temporal_ssfind_experiment` Should only be used internally.

Usage

```
set_temporal_ssfind_initial_state(
  p,
  initial_total_CB,
  initial_total_PB,
  initial_total_SB
)
```

Arguments

p	parameters for the simulation
initial_total_CB	Total initial abundance of CB
initial_total_PB	Total initial abundance of PB
initial_total_SB	Total initial abundance of SB

Value

The passed parameter object but with initial states set (overwritten by the new ones)

`symmetric_bushplus_dynamic_model`

The rate equations, modified from publication of Bush et al 2017 Rhrefhttps://doi.org/10.1093/clinchem/39.5.766doi:10.1093/clinchem/39.5.766. Equations of antagonistic environmental variables are identical, while the oxygen and sulfide diffusivities are forced in antisymmetric manner. Possibility to simulate multiple strain per functional group as published in Rhrefhttps://doi.org/10.1111/ele.14217doi:10.1111/ele.14217 by Limberger et al 2023 remains.

Description

The rate equations, modified from publication of Bush et al 2017 doi:10.1093/clinchem/39.5.766. Equations of antagonistic environmental variables are identical, while the oxygen and sulfide diffusivities are forced in antisymmetric manner. Possibility to simulate multiple strain per functional group as published in doi:10.1111/ele.14217 by Limberger et al 2023 remains.

Usage

```
symmetric_bushplus_dynamic_model(
  t,
  state,
  parameters,
  log10aO_forcing_func,
  log10aS_forcing_func,
  ...
)
```

Arguments

t	The current time in the simulation
state	A vector containing the current (named) values of each state variable
parameters	An object of class runsim_parameter as returned by 'new_runsim_parameter()"
log10aO_forcing_func	function to change oxygen diffusivity aO depending on t
log10aS_forcing_func	function to change sulfide diffusivity aS depending on t
...	not used. Needed to catch additional parameters.

Value

a list containing two elements, namely the rate of change of the strains, and also the current values of oxygen diffusivity aO, as well as sulfide diffusivity aS.

Index

-package *(*microxanox-package*)*, 3

add_strain_var, 3

bushplus_dynamic_model, 4

event_definition_1, 5
event_definition_2, 6
event_definition_symmetric, 7

get_final_states_a_N, 8
get_hysteresis_max, 8
get_hysteresis_min, 9
get_hysteresis_range, 9
get_hysteresis_total, 10
get_nonlinearity, 10
get_ssbyaN_parameter, 11
get_stability_measures, 11
get_stability_measures_replication_ssfind_result
*(*get_stability_measures*)*, 11
get_stability_measures_temporal_ssfind_result
*(*get_stability_measures*)*, 11
get_symmetry_measurements, 12
growth1, 13
growth2, 13

inhibition, 14

microxanox *(*microxanox-package*)*, 3
microxanox-package, 3

new_CB_strain_parameter, 14
new_initial_state, 15
new_PB_strain_parameter, 16
new_replication_ssfind_parameter, 17
new_replication_ssfind_results, 17
new_runsim_parameter, 18
new_runsim_results, 19
new_SB_strain_parameter, 19
new_strain_parameter, 20
new_temporal_ssfind_results, 22

plot_dynamics, 22
plot_dynamics_symmetric, 23
plot_symmetry_measures, 24
plot_temporal_ss, 24
plot_trajectory_symmetry, 25
plot_trajectory_symmetry_compact, 25

run_replication_ssfind, 26
run_simulation, 27
run_simulation_symmetric, 27
run_temporal_ssfind, 28
run_temporal_ssfind_experiment, 28
run_temporal_ssfind_symmetric, 29

set_diffusivities, 30
set_temporal_ssfind_initial_state, 30
symmetric_bushplus_dynamic_model, 31