Package ‘miLAG’

November 17, 2023

Type Package

Title Calculates Microbial Lag Duration (on the Population Level) from Provided Growth Curve Data

Version 1.0.2

Date 2023-11-14

Description Microbial growth is often measured by growth curves i.e. a table of population sizes and times of measurements.
This package allows to use such growth curve data to determine the duration of ``microbial lag phase'' i.e. the time needed for microbes to restart divisions.
It implements the most commonly used methods to calculate the lag duration, these methods are discussed and described in Opalek et.al. 2022.
Citation: ``How to determine microbial lag phase duration?'', M. Opalek, B. Smug, D. Wloch-Salamon (2022) <doi:10.1101/2022.11.16.516631>.

License GPL-3

RoxygenNote 7.2.3

Depends R (>= 4.3.0), dplyr (>= 1.0.8), ggplot2 (>= 3.4.1), testthat (>= 3.1.8), minpack.lm (>= 1.2.2), nlsMicrobio (>= 0.0.3)

Suggests knitr, rmarkdown

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Bogna Smug [aut, cre] (<`https://orcid.org/0000-0001-9364-163X`>)

Maintainer Bogna Smug <`bogna.smug@uj.edu.pl`>

Repository CRAN

Date/Publication 2023-11-17 08:50:03 UTC

R topics documented:

`calc_baranyi_fit_lag` *calc_baranyi_fit_lag*

Description

Calculates lag based on fitting baranyi model to data

Usage

```
calc_baranyi_fit_lag(
  data,
  n0,
  init_lag = NULL,
  init_gr_rate = NULL,
  algorithm = "auto",
  max_iter = 100
)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves

`n0`	a data frame describing initial biomass for each of the curves, i.e. it has two obligatory columns: "curve_id", "N0"
`init_lag`	initial value for the lag parameter, defaults to NULL in which case it will be approximated based on the data
`init_gr_rate`	initial value for the growth rate, defaults to NULL in which case it will be approximated based on the data
`algorithm`	eg. "auto", "Levenberg-Marquardt", "port", defaults to "auto"
`max_iter`	Maximum number of iterations, defaults to 100

Value

growth curve data with additional columns ('lag', and predicted biomass 'predicted')

`calc_lag` *calc_lag*

Description

The main function that calculates lags based on growth curve data, selected method and parameters and returns an extended growth rate data frame (extended by multiple columns with parameters related to lag calculation)

Usage

```
calc_lag(data, method, pars)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`method`	method of lag calculation, choose one of the follwoing: "exponential", "biomass increase", "max growth acceleration", "parameter fitting to a model"
`pars`	a list of parameters. Get.default.parameters function can be used to get the default ones. Otherwise create your onwn list with the following names: - model: if method = "parameter fitting to a model" , one of the following models needs to be chosen: "logistic", "baranyi" - n0_method: first.observation" if the first point is taken as the initial biomass or "minimal.observation" if the minimal biomass is taken is the initial point. In "healthy" growth curves these options should be equivalent but sometimes a drop in OD/biomass is observed at the beginning of a growth curve. In this case it is not obvious what to assume the initial biomass is. - tangent_method "local.regression" (if the tangent is fitted to a number of points around the maximal growth rate) or "to.point" (if the tangent is fitted only to the point where the growth rate is maximal); defaults to "to.point" - threshold: A value of the biomass increase that we can surely associate with the end

of the lag phase rather than random variation durinh the lag. Defaults to 10^2 - curve_points: if tangent.method = "local.regression" then curve_points is the number of points the line is fitted to; defaults to 3 i.e. the point with the maximal uptake rate one point before and one point after - init_gr_rate: if logistic model is fitted. Defaults to NULL in which case the initial value will be based on the data - init_lag: if a logistic model is fitted, Defaults to NULL in which case the initial value will be based on the data - algorithm: if method = "parameter fitting to a model", nls algorithm to run the model fit; defaults to "auto" which will choose the best between bounded and unbounded "Levenberg-Marquardt" and bounded "port" - max_iter = if method = "parameter fitting to a model", the maximum number of nls iterations, defaults to 100

Value

growth curve data (time, biomass, curve_id) with the following additional columns: log_biomass, lag, line_slope, line_intercept, lag_calc_method, predicted_data, diff, second_deriv_b, tangent_point, threshold

`calc_lagistic_fit_lag` *calc_lagistic_fit_lag*

Description

Calculates lag based on fitting logistic model to data

Usage

```
calc_lagistic_fit_lag(
  data,
  n0,
  init_gr_rate = NULL,
  init_K = NULL,
  init_lag = NULL,
  algorithm,
  max_iter,
  return_all_params = FALSE,
  min_b = 0.2,
  min_a = 0.8
)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`n0`	a data frame describing initial biomass for each of the curves, i.e. it has two obligatory columns: "curve_id", "N0"

`init_gr_rate`	initial value for the growth rate, defaults to NULL in which case it will be approximated based on the data
`init_K`	initial value for the saturation parameter K, defaults to NULL in which case it will be approximated based on the data
`init_lag`	initial value for the lag parameter, defaults to NULL in which case it will be approximated based on the data
`algorithm`	eg. "auto", "Levenberg-Marquardt", "port"
`max_iter`	Maximum number of iterations
`return_all_params`	defaults to FALSE, TRUE if you also want to get K and growth.rate apart from lag
`min_b`	defaults to 0.2; mina and minb define where to look for exponential phase: it will be where the biomass is between min + (max-min)*(lower.bound.for.gr TO upper.bound.for.gr)
`min_a`	defaults to 0.8

Value

growth curve data with additional columns ('lag', and predicted biomass 'predicted'), and the fitting object if return.all.params was set to TRUE

`calc_lag_fit_to_baranyi_with_lag`
calc_lag_fit_to_baranyi_with_lag

Description

Runs nlsLM/nls algorithms with three different parameter setups to fit the best Logistic model parameters to our data and chooses the best model

Usage

```
calc_lag_fit_to_baranyi_with_lag(
  gr_curve,
  LOG10N0 = NULL,
  init_lag = NULL,
  init_mumax = NULL,
  init_LOG10Nmax = NULL,
  algorithm = "auto",
  max_iter = 100,
  lower_bound = c(0, 0, 0, 0)
)
```

Arguments

`gr_curve`	data from one specific growth curve with these two columns: time and biomass
`LOG10N0`	the decimal logarithm of initial biomass
`init_lag`	initial value for the lag parameter
`init_mumax`	initial value for the mumax parameter
`init_LOG10Nmax`	initial value for the LOG10Nmax parameter
`algorithm`	defaults to "auto" which chooses between bounded and unbounded Levenberg-Marquardt method and the bounded port method
`max_iter`	max. number of itertaions; defaults to 100
`lower_bound`	lower.bound for the bounded nls optimisation; defaults to 0

Value

lag and the nls fitting object with parameters fitted to logistic model

---

`calc_lag_fit_to_logistic_with_lag` *calc_lag_fit_to_logistic_with_lag*

Description

Runs nlsLM/nls algorithm of the user's choice to fit the Logistic model parameters to our data

Usage

```
calc_lag_fit_to_logistic_with_lag(
  gr_curve,
  n0,
  init_gr_rate = init_gr_rate,
  init_K = init_K,
  init_lag = init_lag,
  algorithm = "auto",
  max_iter = 100,
  lower_bound = c(0, 0, 0)
)
```

Arguments

`gr_curve`	data from one specific growth curve with these two columns: time and biomass
`n0`	the initial biomass
`init_gr_rate`	initial value for the growth rate
`init_K`	initial value for the saturation parameter K
`init_lag`	initial value for the lag parameter

`algorithm`	defaults to "auto" which chooses between bounded and unbounded Levenberg-Marquardt method and the bounded port method
`max_iter`	max. number of iterations; defaults to 100
`lower_bound`	lower bound for the bounded nls optimization; defaults to 0

Value

lag and the nls fitting object with parameters fitted to logistic model

choose_lag_fit_algorithm_baranyi

choose_lag_fit_algorithm_baranyi

Description

Runs nlsLM/nls algorithms with three different parameter setups to fit the best Baranyi parameters to our data and chooses the best model

Usage

```
choose_lag_fit_algorithm_baranyi(
  gr_curve,
  LOG10N0,
  init_lag,
  init_mumax,
  init_LOG10Nmax,
  max_iter,
  lower_bound
)
```

Arguments

`gr_curve`	data from one specific growth curve with the following columns: LOG10N, t
`LOG10N0`	init value for the LOG10N0 parameter
`init_lag`	initial value for the lag
`init_mumax`	initial value for the mumax parameter
`init_LOG10Nmax`	initial value for the LOG10Nmax parameter
`max_iter`	max. number of iterations
`lower_bound`	lower bound for the bounded nls optimization;

Value

the best nls fitting object with parameters fitted to Baranyi model (lowest Res.Sum Sq provided that all coefficients are nonnegative)

choose_lag_fit_algorithm_logistic *choose_lag_fit_algorithm_logistic*

Description

Runs nlsLM/nls algorithms with three different parameter setups to fit the best Logistic model parameters to our data and chooses the best model

Usage

```
choose_lag_fit_algorithm_logistic(
  gr_curve,
  n0,
  init_gr_rate = init_gr_rate,
  init_K = init_K,
  init_lag = init_lag,
  max_iter = 100,
  lower_bound = c(0, 0, 0)
)
```

Arguments

`gr_curve`	data from one specific growth curve with the following columns: LOG10N, t
`n0`	the initial biomass
`init_gr_rate`	initial value for the growth rate
`init_K`	initial value for the saturation parameter K
`init_lag`	initial value for the lag parameter
`max_iter`	max. number of iterations; defaults to 100
`lower_bound`	lower bound for the bounded nls optimization; defaults to 0

Value

the best nls fitting object with parameters fitted to logistic model (lowest Res.Sum Sq provided that all coefficients are nonnegative)

`compare_algorithms` *compare_algorithms*

Description

Compares results of 3 objects obtained from running nls

Usage

```
compare_algorithms(nls_LM_no_bound, nls_PORT, nlsres_LM)
```

Arguments

`nls_LM_no_bound`	first object resulting from running nls
`nls_PORT`	second object resulting from running nls
`nlsres_LM`	third object resulting from running nls

Value

the best fitting object (lowest Res.Sum Sq provided that all coefficients are nonnegative)

`cut_the_data` *cut_the_data Subsets the data frame containing only the observations up to the specified maximum time*

Description

cut_the_data Subsets the data frame containing only the observations up to the specified maximum time

Usage

```
cut_the_data(data, max_time)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`max_time`	max. time at which we want to cut the growth curve data

Value

cut data

fit_exp_lag *fit_exp_lag*

Description

Fits the lag to multiple growth curves based on the basic tangent method

Usage

```
fit_exp_lag(data, tangent_method, n0, curve_points = 3)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`tangent_method`	"local.regression" (if the tangent is fitted to a number of points around the maximal growth rate) or "to.point" (if the tangent is fitted only to the point where the growth rate is maximal); defaults to "to.point"
`n0`	the initial biomass (a tangent line crossing N0 line will determine the lag)
`curve_points`	if tangent_method = "local.regression" then curve_points is the number of points the line is fitted to; defaults to 3 i.e. the point with the maximal uptake rate one point before and one point aftter

Value

growth curve data (as input) together with additional columns: lag, line.intercept and line.slope

fit_exp_lag_to_curve *fit_exp_lag_to_curve*

Description

Fits the lag to one growth curve based on the basic tangent method

Usage

```
fit_exp_lag_to_curve(data, n0, tangent_method = "to.point", curve_points = 3)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass", This is data from one growth curve only, one (mean) observation per time
`n0`	the initial biomass (a tangent line crossing N0 line will determine the lag)
`tangent_method`	"local.regression" (if the tangent is fitted to a number of points around the maximal growth rate) or "to.point" (if the tangent is fitted only to the point where the growth rate is maximal); defaults to "to.point"
`curve_points`	if tangent_method = "local.regression" then curve_points is the number of points the line is fitted to; defaults to 3 i.e. the point with the maximal uptake rate one point before and one point after

Value

line_slope: slope of the tangent line, line_intercept: intercept of the tangent line, lag: lag, tangent_points: i..e a data frame of all points selected for fitting the line

---

`fit_max_infl_lag` *fit_max_infl_lag*

Description

Fits the lag to multiple growth curves based on the max growth acceleration method It finds where the second derivative is the largest

Usage

```
fit_max_infl_lag(data)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves

Value

growth curve data (as input) together with additional columns: lag, log.biomass, time.diff, time.av, second.deriv.b, biomass.increase

`get_all_methods_lag` *get_all_methods_lag*

Description

Runs the main function that calculates lags based on growth curve data based on all possible methods.

Usage

```
get_all_methods_lag(data, biomass_incr_threshold, pars = NULL)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`biomass_incr_threshold`	A value of the biomass increase that we can surely associate with the end of the lag phase rather than random variation during the lag. Needs to be set specifically to avoid unconscious use of the value set by default. If set to NULL, the value from pars will be taken
`pars`	a list of parameters. defaults to the ones set by get_def_pars function. Otherwise create your own list with the following names: - model: if method = "parameter fitting to a model" , one of the following models needs to be chosen: "logistic", "baranyi" - n0_method: first.observation" if the first point is taken as the initial biomass or "minimal.observation" if the minimal biomass is taken is the initial point. In "healthy" growth curves these options should be equivalent but sometimes a drop in OD/biomass is observed at the beginning of a growth curve. In this case it is not obvious what to assume the initial biomass is. - tangent.method "local.regression" (if the tangent is fitted to a number of points around the maximal growth rate) or "to.point" (if the tangent is fitted only to the point where the growth rate is maximal); defaults to "to.point" - threshold: A value of the biomass increase that we can surely associate with the end of the lag phase rather than random variation during the lag. Defaults to 10^2 - curve_points: if tangent_method = "local.regression" then curve_points is the number of points the line is fitted to; defaults to 3 i.e. the point with the maximal uptake rate one point before and one point after - init_growth.rate: if logistic model is fitted. Defaults to NULL in which case the initial value will be based on the data - init_lag: if a logistic model is fitted, Defaults to NULL in which case the initial value will be based on the data - algorithm: if method = "parameter fitting to a model", nls algorithm to run the model fit; defaults to "auto" which will choose the best between bounded and unbounded "Levenberg-Marquardt" and bounded "port" - max_iter = if method = "parameter fitting to a model", the maximum number of nls iterations, defaults to 100

Value

growth curve data (time, biomass, curve_id) with the column: lag_calculation_method, and with the following additional columns: log_biomass, lag, line_slope, line_intercept, lag_calculation_method, predicted_data, diff, second_deriv_b, tangent_point, threshold Note that each growth curve will appear

---

`get_def_pars` *get_def_pars Set defaults parameters used by calc_lag function*

Description

get_def_pars Set defaults parameters used by calc_lag function

Usage

```
get_def_pars()
```

Value

list of parameters

`get_init_pars_baranyi` *get_init_pars_baranyi*

Description

Finds reasonable approximation for baranyi growth curve parameters (init_mumax, lag) based on the growth curve and some initial values These approximations will be used as the initial values for the proper optimization algorithm run later.

Usage

```
get_init_pars_baranyi(
  data_this_curve,
  this_n0,
  init_lag,
  init_gr_rate,
  min_b = 0.2,
  min_a = 0.8
)
```

Arguments

`data_this_curve`	data from one specific growth curve with these two columns: time and biomass
`this_n0`	the initial biomass
`init_lag`	initial value for the lag parameter
`init_gr_rate`	initial value for the growth rate
`min_b`	defaults to 0.2; mina and minb define where to look for exponential phase: it will be where the biomass is between min + (max-min)*(mina TO minb)
`min_a`	defaults to 0.8

Value

list of parameters: init_mumax, init_lag

`get_init_pars_logistic` *get_init_pars_logistic*

Description

Finds reasonable approximation for logistic growth curve parameters (K, lag. growth rate) based on the growth curve and some initial values These approximations will be used as the initial values for the proper optimization algorithm run later.

Usage

```
get_init_pars_logistic(
  data_this_curve,
  this_n0,
  init_K,
  init_lag,
  init_gr_rate,
  min_b = 0.2,
  min_a = 0.8
)
```

Arguments

`data_this_curve`	data from one specific growth curve with these two columns: time and biomass
`this_n0`	the initial biomass
`init_K`	initial value for the saturation parameter K
`init_lag`	initial value for the lag parameter
`init_gr_rate`	initial value for the growth rate
`min_b`	defaults to 0.2; mina and minb define where to look for exponential phase: it will be where the biomass is between min + (max-min)*(min_a TO min_b)
`min_a`	defaults to 0.8

Value

list of parameters: init_K, init_lag, init_gr_rate,

`get_lag` *get_lag*

Description

The most basic function that calculates lags based on growth curve data, selected method and parameters. It uses calc_lag function and strips the results to only get lag parameter for each growth curve id.

Usage

```
get_lag(data, method, pars)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`method`	method of lag calculation, choose one of the follwoing: "exponential", "biomass increase", "max growth acceleration", "parameter fitting to a model"
`pars`	a list of parameters. Get.default.parameters function can be used to get the default ones. Otherwise create your onwn list with the following names: - model: if method = "parameter fitting to a model" , one of the following models needs to be chosen: "logistic", "baranyi" - n0_method: first.observation" if the first point is taken as the initial biomass or "minimal.observation" if the minimal biomass is taken is the initial point. In "healthy" growth curves these options should be equivalent but sometimes a drop in OD/biomass is observed at the beginning of a growth curve. In this case it is not obvious what to assume the initial biomass is. - tangent_method "local.regression" (if the tangent is fitted to a number of points around the maximal growth rate) or "to.point" (if the tangent is fitted only to the point where the growth rate is maximal); defaults to "to.point" - threshold: A value of the biomass increase that we can surely associate with the end of the lag phase rather than random variation durinh the lag. Defaults to 10^2 - curve_points: if tangent.method = "local.regression" then curve_points is the number of points the line is fitted to; defaults to 3 i.e. the point with the maximal uptake rate one point before and one point after - init_gr_rate: if logistic model is fitted. Defaults to NULL in which case the initial value will be based on the data - init_lag: if a logistic model is fitted, Defaults to NULL in which case the initial value will be based on the data - algorithm: if method = "parameter fitting to a model", nls algorithm to run the model fit; defaults to "auto" which will choose the best between bounded and unbounded "Levenberg-Marquardt" and bounded "port" - max_iter = if method = "parameter fitting to a model", the maximum number of nls iterations, defaults to 100

Value

lag per each curve_id

`get_n0` *get_n0*

Description

Gets the initial biomass to relate to

Usage

```
get_n0(biomass, n0_method)
```

Arguments

`biomass`	vector of biomass (chronologically ordered as in growth curve)
`n0_method`	"first.observation" if the first point is taken as the initial biomass or "minimal.observation" if the minimal biomass is taken is the initial point. In "healthy" growth curves these options should be equivalent but sometimes a drop in OD/biomass is observed at the beginning of a growth curve. In this case it is not obvious what to assume the initial biomass is.

Value

a value of the initial biomass (either the first observation or the minimum value depending on the parameter N0.method)

`get_theme` *get_theme*

Description

This function sets a ggplot theme without grid. The theme removes the major and minor grid lines, sets a white background with a gray border and adjusts the text size.

Usage

```
get_theme(text_size = 12)
```

Arguments

`text_size`	defaults to 12

Value

a ggplot theme

`lag_biomass_incr` *lag_biomass_incr*

Description

Fits the lag to multiple growth curves based on the biomass increase method

Usage

```
lag_biomass_incr(data, threshold, n0)
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`threshold`	A value of the biomass increase that we can surely associate with the end of the lag phase rather than random variation during the lag
`n0`	the initial biomass (lag will be defined as the time point where the difference between biomass and N0 reaches a predefined threshold)

Value

growth curve data (as input) together with additional columns: N0, increase.from.N0, lag

`make_grwoth_curve_df` *make_grwoth_curve_df*

Description

Create a growth curve data frame that can be later passed to the lag clculation functions

Usage

```
make_grwoth_curve_df(time, biomass, curve_id = NULL)
```

Arguments

`time`	numeric vector of times when biomass was measured (chronologically ordered as in growth curve)
`biomass`	numeric vector of measured biomass values (chronologically ordered as in growth curve)
`curve_id`	character vector of growth curve identifiers (i.e. if there are multiple measurements done at the same time point, they should have different curve_id)

Value

a data frame representing growth curve data

`plot_data` *plot_data*

Description

Plots the provided growth curve (one single growth curve) on logarithmic scale

Usage

```
plot_data(data_new, log10_transform = TRUE)
```

Arguments

`data_new`	a data frame with two required columns names: "time" and "biomass"
`log10_transform`	if to plot y axis (biomass) on log10 scale

Value

ggplot object with a growth curve

`plot_lag_fit` *plot_lag_fit*

Description

Plots the provided growth curve (one single growth curve) together with the calculated lag and and the rationale for lag calculation

Usage

```
plot_lag_fit(data_new, print_lag_info = TRUE, log10_transform = TRUE)
```

Arguments

`data_new`	a data frame output by Calculate.Lag function: it needs to have the following columns: "time", "biomass", "tangent.point", "predicted.data", "threshold", "N0", "second.deriv.b", "line.intercept", "line.slope"
`print_lag_info`	if set to "TRUE" prints the lag length on the graph
`log10_transform`	if to plot y axis (biomass) on log10 scale

Value

ggplot object with a growth curve

smooth_data *smooth_data Smoothens growth curves data*

Description

smooth_data Smoothens growth curves data

Usage

```
smooth_data(data, smooth_kind = "3RS3R")
```

Arguments

`data`	a data frame with two required columns names: "time" and "biomass",and one optional column: "curve_id" This is data from may come from multiple growth curves
`smooth_kind`	kind used for the smooth functions, defaults to "3RS3R"

Value

smoothened data

Index