Package ‘nls.multstart’

January 9, 2025

Type Package

Title Robust Non-Linear Regression using AIC Scores

Version 2.0.0

Maintainer Daniel Padfield `<d.padfield@exeter.ac.uk>`

Description Non-linear least squares regression with the Levenberg-Marquardt algorithm using multiple starting values for increasing the chance that the minimum found is the global minimum.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 3.2.1)

Imports minpack.lm, purrr, dplyr, tidyr, tibble, lhs, cli, rlang

Suggests ggplot2, broom, nlstools, testthat

NeedsCompilation no

Author Daniel Padfield [aut, cre],
Granville Matheson [aut],
Francis Windram [aut]

Repository CRAN

Date/Publication 2025-01-09 11:00:02 UTC

Contents

`Chlorella_TRC` *Example metabolic thermal response curves*

Description

A dataset containing example data of rates of photosynthesis and respiration of the phytoplankton Chlorella vulgaris. Instantaneous rates of metabolism were made across a range of assay temperatures to incorporate the entire thermal response of the populations. The dataset is the cleaned version so some datapoints have been omitted.

Usage

```
data("Chlorella_TRC")
```

Format

A data frame with 649 rows and 7 variables:

curve_id a unique value for each separate curve

growth.temp the growth temperature that the culture was maintained at before measurements were taken (degrees centigrade)

process whether the cultures had been kept for a long time at their growth temperature (adaptation/~100 generations) or a short time (a measure of acclimation/~10 generations)

flux whether the curve depicts respiration or gross photosynthesis

temp the assay temperature at which the metabolic rate was measured (degrees centigrade)

K the assay temperature in degrees Kelvin

ln.rate the metabolic rate measured (micro mol O2 micro gram C-1 hr-1)

Source

Daniel Padfield

References

Padfield, D., Yvon-durocher, G., Buckling, A., Jennings, S. & Yvon-durocher, G. (2015). Rapid evolution of metabolic traits explains thermal adaptation in phytoplankton, Ecology Letters, 19, 133-142.

Examples

```
data("Chlorella_TRC")
library(ggplot2)
ggplot(Chlorella_TRC) +
 geom_point(aes(temp, ln.rate, col = process)) +
 facet_wrap(~ growth.temp + flux)
```

`nls_multstart` *Finds the best fit of non-linear model based on AIC score*

Description

Finds the best estimated model using non-linear least squares regression using nlsLM(). The best fit is determined using AIC scores.

Arguments

`formula`	a non-linear model formula, with the response on the left of a ~ operator and an expression involving parameters on the right.
`data`	(optional) data.frame, list or environment in which to evaluate the variables in `formula` and `modelweights`.
`iter`	number of combinations of starting parameters which will be tried . If a single value is provided, then a shotgun/random-search/lhs approach will be used to sample starting parameters from a uniform distribution within the starting parameter bounds. If a vector of the same length as the number of parameters is provided, then a gridstart approach will be used to define each combination of that number of equally spaced intervals across each of the starting parameter bounds respectively. Thus, c(5,5,5) for three fitted parameters yields 125 model fits. Supplying a vector for `iter` will override `convergence_count`.
`start_lower`	lower boundaries for the start parameters. If missing, this will default to -1e+10.
`start_upper`	upper boundaries for the start parameters. If missing, this will default to 1e+10.
`supp_errors`	if `supp_errors = 'Y'`, then warning messages will be suppressed and no error messages from `nlsLM` will be shown, reducing the number of error messages printed while the model attempts to converge using poor starting parameters. We advise to only use `supp_errors = 'Y'` when confident in the bounds of your starting parameters.
`convergence_count`	The number of counts that the winning model should be undefeated for before it is declared the winner. This argument defaults to 100. If specified as `FALSE`, then all of the iterations will be fitted, and the best model selected. Note that `convergence_count` can only be used with a shotgun/random-search approach, and not with a gridstart approach. This argument will be ignored if a gridstart approach is specified by a vector input for `iter`.
`control`	specific control can be specified using `nls.lm.control`.
`modelweights`	Optional model weights for the nls. If `data` is specified, then this argument should be the name of the numeric weights vector within the `data` object.
`lhstype`	Method to use for Latin Hypercube Sampling using `lhs`. Choice of `"random"` (simple random lhs, fast), `"improved"` (lhs with optimised euclidean distance between points, medium speed), `"maximin"` (lhs with maximised minimum distance between points, medium speed), or `"genetic"` (lhs optimised to the S optimilaity criterion using a genetic algorithm, slow). If not set, a purely random (shotgun) approach is taken. Only used if `iter` is a single number.
`...`	Extra arguments to pass to `nlsLM` if necessary.

Value

returns a nls object of the best estimated model fit.

Note

Useful additional arguments for `nlsLM` include: `na.action = na.omit`, `lower/upper = c()` where these represent upper and lower boundaries for parameter estimates.

Author(s)

Daniel Padfield

Granville Matheson

Francis Windram

See Also

`nlsLM` for details on additional arguments to pass to the nlsLM function. `lhs` for details of Latin Hypercube Sampling

Examples

```
# load in data

data("Chlorella_TRC")
Chlorella_TRC_test <- Chlorella_TRC[Chlorella_TRC$curve_id == 1,]

# run nls_multstart()

# define the Sharpe-Schoolfield equation
schoolfield_high <- function(lnc, E, Eh, Th, temp, Tc) {
 Tc <- 273.15 + Tc
 k <- 8.62e-5
 boltzmann.term <- lnc + log(exp(E/k*(1/Tc - 1/temp)))
 inactivation.term <- log(1/(1 + exp(Eh/k*(1/Th - 1/temp))))
 return(boltzmann.term + inactivation.term)
}

fits <- nls_multstart(ln.rate ~ schoolfield_high(lnc, E, Eh, Th, temp = K, Tc = 20),
                data = Chlorella_TRC_test,
                iter = 500,
                start_lower = c(lnc=-10, E=0.1, Eh=0.5, Th=285),
                start_upper = c(lnc=10, E=2, Eh=5, Th=330),
                lower = c(lnc=-10, E=0, Eh=0, Th=0),
                supp_errors = 'Y')
```

Index