Package ‘rTPC’

August 17, 2023

Type Package
Title Fitting and Analysing Thermal Performance Curves
Version 1.0.4
Maintainer Daniel Padfield <d.padfield@exeter.ac.uk>
Description Helps to fit thermal performance curves (TPCs). ‘rTPC’ contains 26 model formulations previously used to fit TPCs and has helper functions to set sensible start parameters, upper and lower parameter limits and estimate parameters useful in downstream analyses, such as cardinal temperatures, maximum rate and optimum temperature. See Padfield et al. (2021) <doi:10.1111/2041-210X.13585>.
License GPL-3
URL https://github.com/padpadpadpad/rTPC
BugReports https://github.com/padpadpadpad/rTPC/issues
Depends R (>= 2.10)
Imports stats
Suggests boot, broom, car, forcats, ggplot2, ggrepel, knitr, minpack.lm, MuMIn, nls.multstart, nlstools, patchwork, progress, RColorBrewer, rmarkdown, stringr, testthat, tidyverse
VignetteBuilder knitr
Encoding UTF-8
LazyData true
RoxygenNote 7.2.3
NeedsCompilation no
Author Daniel Padfield [aut, cre], Hannah O'Sullivan [aut]
Repository CRAN
Date/Publication 2023-08-17 06:40:06 UTC
R topics documented:

- bacteria_tpc
- beta_2012
- boatman_2017
- briere2_1999
- calc_params
- chlorella_tpc
- delong_2017
- deutsch_2008
- flinn_1991
- gaussian_1987
- get_breadth
- get_ctmax
- get_ctmin
- get_e
- get_eh
- get_lower_lims
- get_model_names
- get_q10
- get_rmax
- get_skewness
- get_start_vals
- get_thermalsafety_margin
- get_thermal_tolerance
- get_topt
- get_upper_lims
- hinshelwood_1947
- joehnk_2008
- johnsonlewin_1946
- kamykowski_1985
- lactin2_1995
- lrf_1991
- modifiedgaussian_2006
- oneill_1972
- pawar_2018
- quadratic_2008
- ratkowsky_1983
- rezende_2019
- sharpeschoolfull_1981
- sharpeschoolhigh_1981
- sharpeschoollow_1981
- spain_1982
- thomas_2012
- thomas_2017
- weibull_1995
Example thermal performance curves of bacterial growth

Description

A dataset containing example data of growth rates of the bacteria Pseudomonas fluorescens in the presence and absence of its phage, phi2. Growth rates were measured across a range of assay temperatures to incorporate the entire thermal performance of the bacteria. The dataset is the cleaned version so some data points have been omitted. There are multiple independent measurements per temperature for each treatment.

Usage

data("bacteria_tpc")

Format

A data frame with 649 rows and 7 variables:

- **phage**: whether the bacteria was grown with or without phage
- **temp**: the assay temperature at which the growth rate was measured (degrees centigrade)
- **rate**: estimated growth rate per hour

Source

Daniel Padfield

References


Examples

data("bacteria_tpc")
library(ggplot2)
ggplot(bacteria_tpc) +
  geom_point(aes(temp, rate, col = phage))
Beta model for fitting thermal performance curves

**Description**

Beta model for fitting thermal performance curves

**Usage**

beta_2012(temp, a, b, c, d, e)

**Arguments**

- `temp` temperature in degrees centigrade
- `a` dimensionless parameter
- `b` dimensionless parameter
- `c` dimensionless parameter
- `d` dimensionless parameter
- `e` dimensionless parameter

**Details**

Equation:

\[
\text{rate} = \frac{a}{c} \left( \frac{\text{temp} - b + \frac{c(d-1)}{d+e-2}}{c} \right)^{d-1} \cdot \left( 1 - \frac{\text{temp} - b + \frac{c(d-1)}{d+e-2}}{c} \right)^{e-1}
\]

Start values in `get_start_vals` are derived from the data or sensible values from the literature.

Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

**Value**

a numeric vector of rate values based on the temperatures and parameter values provided to the function

**Note**

Generally we found this model difficult to fit.

**Author(s)**

Daniel Padfield
References


Examples

# load in ggplot
library(ggplot)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'beta_2012')
mod <- nls.multstart::nls_multstart(rate ~ beta_2012(temp = temp, a, b, c, d, e),
data = d,
iter = c(7, 7, 7, 7),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'beta_2012'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'beta_2012'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()
Usage

boatman_2017(temp, rmax, tmin, tmax, a, b)

Arguments

temp temperature in degrees centigrade
rmax the rate at optimum temperature
tmin low temperature (ºC) at which rates become negative
tmax high temperature (ºC) at which rates become negative
a shape parameter to adjust the skewness of the curve
b shape parameter to adjust the kurtosis of the curve

Details

Equation:

\[
rate = r_{max} \cdot \left( \sin \left( \pi \left( \frac{temp - t_{min}}{t_{max} - t_{min}} \right)^a \right) \right)^b
\]

Start values in get_start_vals are derived from the data or sensible values from the literature. Limits in get_lower_lims and get_upper_lims are derived from the data or based extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References


Examples

# load in ggrepplot
library(ggrepplot)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'boatman_2017')
# fit model
mod <- nls_multstart::nls_multstart(rate ~ boatman_2017(temp = temp, rmax, tmin, tmax, a, b), data = d, iter = c(4,4,4,4), start_lower = start_vals - 10, start_upper = start_vals + 10, lower = get_lower_lims(d$temp, d$rate, model_name = 'boatman_2017'), upper = get_upper_lims(d$temp, d$rate, model_name = 'boatman_2017'), supp_errors = 'Y', convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) + geom_point(aes(temp, rate), d) + geom_line(aes(temp, .fitted), col = 'blue') + theme_bw()

---

### briere2_1999

**Briere2 model for fitting thermal performance curves**

**Description**

Briere2 model for fitting thermal performance curves

**Usage**

briere2_1999(temp, tmin, tmax, a, b)

**Arguments**

- **temp**: temperature in degrees centigrade
- **tmin**: low temperature (°C) at which rates become negative
- **tmax**: high temperature (°C) at which rates become negative
- **a**: scale parameter to adjust maximum rate of the curve
- **b**: shape parameter to adjust the asymmetry of the curve
Details

Equation:
\[ rate = a \cdot temp \cdot (temp - t_{min}) \cdot (t_{max} - temp)^{\frac{1}{2}} \]

Start values in `get_start_vals` are derived from the data or sensible values from the literature. Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

Value

A numeric vector of rate values based on the temperatures and parameter values provided to the function.

Note

Generally we found this model easy to fit.

References


Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'briere2_1999')
# fit model
mod <- nls.multstart::nls_multstart(rate~briere2_1999(temp = temp, tmin, tmax, a, b),
data = d, iter = c(4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'briere2_1999'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'briere2_1999'),
supp.errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)
```
# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()

---

**calc_params**

*Calculate extra parameters of a thermal performance curve*

**Description**

Calculate extra parameters of a thermal performance curve

**Usage**

calc_params(model)

**Arguments**

- **model** nls model object that contains a model of a thermal performance curve

**Details**

Currently estimates:

- maximum rate (rmax) using `get_rmax()`
- optimum temperature (topt) using `get_topt()`
- critical thermal maximum (ctmax) using `get_ctmax()`
- critical thermal minimum (ctmin) using `get_ctmin()`
- activation energy (e) using `get_e()`
- deactivation energy (eh) using `get_eh()`
- q10 value using `get_q10()`
- thermal safety margin using `get_thermalsafetymargin()`
- thermal tolerance using `get_thermaltolerance()`
- thermal performance breadth using `get_breadth()`
- skewness using `get_skewness()`

**Value**

a dataframe containing the estimates of key TPC traits for a given model object. If any parameters cannot be calculated for a thermal performance curve, they will return `NA`. 
Example metabolic thermal performance 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 performance of the populations. The dataset is the cleaned version so some datapoints have been omitted.

**Usage**

data("chlorella_tpc")

**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)
- **rate**: the metabolic rate measured (micro mol O2 micro gram C-1 hr-1)

**Source**

Daniel Padfield

**References**


**Examples**

data("chlorella_tpc")
library(ggplot2)
ggplot(chlorella_tpc) +
  geom_point(aes(temp, rate, col = process)) +
  facet_wrap(~ growth_temp + flux)
DeLong enzyme-assisted Arrhenius model for fitting thermal performance curves

Description

DeLong enzyme-assisted Arrhenius model for fitting thermal performance curves

Usage

delong_2017(temp, c, eb, ef, tm, ehc)

Arguments

temp  temperature in degrees centigrade
  c      potential reaction rate
  eb     baseline energy needed for the reaction to occur (eV)
  ef     temperature dependence of folding the enzymes used in the metabolic reaction, relative to the melting temperature (eV)
  tm     melting temperature in degrees centigrade
  ehc    temperature dependence of the heat capacity between the folded and unfolded state of the enzymes, relative to the melting temperature (eV)

Details

Equation:

\[
rate = c \cdot \exp\left( -eb - \left( ef \left( 1 - \frac{temp + 273.15}{tm} \right) \right) + ehc \cdot \left( (temp + 273.15) - tm - (temp + 273.15) \cdot \ln\left( \frac{temp + 273.15}{tm} \right) \right) \right)
\]

where \( k \) is Boltzmann’s constant with a value of 8.62e-5 and \( tm \) is actually \( tm - 273.15 \)

Start values in `get_start_vals` are derived from the data or sensible values from the literature.

Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.
References


Examples

# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'delong_2017')
mod <- nls.multstart::nls_multstart(rate~delong_2017(temp = temp, c, eb, ef, tm,ehc),
data = d,
iter = c(4,4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'delong_2017'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'delong_2017'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

---

**Description**

Modified deutsch model for fitting thermal performance curves

**Usage**

deuacht_2008(temp, rmax, topt, ctmax, a)
Arguments

temp temperature in degrees centigrade
rmax maximum rate at optimum temperature
topt optimum temperature (°C)
cctmax critical thermal maximum (°C)
a related to the full curve width

Details

Equation:

- if \( temp < t_{opt} \), \( rate = r_{max} \cdot \exp \left( -\frac{(temp - topt)^2}{2a} \right) \)
- if \( temp > t_{opt} \), \( rate = r_{max} \cdot \left( 1 - \frac{(temp - t_{opt})(t_{opt} - cct_{max})^2}{t_{opt} - cct_{max}} \right) \)

Start values in `get_start_vals` are derived from the data.
Limits in `get_lower_lims` and `get_upper_lims` are based on extreme values that are unlikely to occur in ecological settings.

Value

A numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References


Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'deutsch_2008')
# fit model
mod <- nls_multstart::nls_multstart(rate~deutsch_2008(temp = temp, rmax, topt, cctmax, a),
```
data = d,
iter = c(4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'deutsch_2008'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'deutsch_2008'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()
Value

A numeric vector of rate values based on the temperatures and parameter values provided to the function.

Note

Generally we found this model easy to fit.

References

Flinn PW Temperature-dependent functional response of the parasitoid Cephalonomia waterstoni (Gahan) (Hymenoptera, Bethylidae) attacking rusty grain beetle larvae (Coleoptera, Cucujidae). Environmental Entomology, 20, 872–876, (1991)

Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'flinn_1991')
# fit model
mod <- nls.multstart::nls_multstart(rate~flinn_1991(temp = temp, a, b, c),
data = d,
iter = c(4,4,4),
start_lower = start_vals - 1,
start_upper = start_vals + 1,
lower = get_lower_lims(d$temp, d$rate, model_name = 'flinn_1991'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'flinn_1991'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
ggeom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()
```
Gaussian model for fitting thermal performance curves

Description

Gaussian model for fitting thermal performance curves

Usage

gaussian_1987(temp, rmax, topt, a)

Arguments

temp  temperature in degrees centigrade
rmax  maximum rate at optimum temperature
topt  optimum temperature (ºC)
a     related to the full curve width

Details

Equation:

\[ rate = r_{max} \cdot \exp\left(-0.5 \left(\frac{|temp - topt|}{a}\right)^2\right) \]

Start values in get_start_vals are derived from the data

Limits in get_lower_lims and get_upper_lims are based on extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References

Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'gaussian_1987')
# fit model
mod <- nls.multstart::nls_multstart(rate~gaussian_1987(temp = temp, rmax, topt, a),
data = d,
iter = c(4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'gaussian_1987'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'gaussian_1987'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()
```

---

table

<table>
<thead>
<tr>
<th>get_breadth</th>
<th>Estimate thermal performance breadth of a thermal performance curve</th>
</tr>
</thead>
</table>

Description

Estimate thermal performance breadth of a thermal performance curve

Usage

```r
get_breadth(model, level = 0.8)
```
get_ctmax

Arguments

- **model**: nls model object that contains a model of a thermal performance curve
- **level**: proportion of maximum rate over which thermal performance breadth is calculated

Details

Thermal performance breadth is calculated as the range of temperatures over which a curve’s rate is at least 0.8 of peak. This defaults to a proportion of 0.8 but can be changed using the `level` argument.

Value

Numeric estimate of thermal performance breadth (in °C)

---

**get_ctmax**

*Estimate the critical thermal maximum of a thermal performance curve*

Description

Estimate the critical thermal maximum of a thermal performance curve

Usage

`get_ctmax(model)`

Arguments

- **model**: nls model object that contains a model of a thermal performance curve

Details

Critical thermal maximum is calculated by predicting over a temperature range 50 °C beyond the maximum value in the dataset. The predicted rate value closest to 0 is then extracted. When this is impossible due to the curve formula (i.e. the Sharpe-Schoolfield model), the temperature where the rate is 5 percent of the maximum rate is estimated. Predictions are done every 0.001 °C so the estimate of the critical thermal maximum should be accurate up to 0.001 °C.

Value

Numeric estimate of critical thermal maximum (°C)
get_ctmin

Estimate the critical thermal minimum of a thermal performance curve

Description
Estimate the critical thermal minimum of a thermal performance curve

Usage
get_ctmin(model)

Arguments
model: nls model object that contains a model of a thermal performance curve

Details
Optimum temperature is calculated by predicting over a temperature range 50 degrees lower than the minimum value in the dataset. The predicted rate value closest to 0 is then extracted. When this is impossible due to the curve formula (i.e., the Sharpe-Schoolfield model), the temperature where the rate is 5 percent of the maximum rate is estimated. Predictions are done every 0.001 °C value so the estimate of the critical thermal minimum should be accurate up to 0.001 °C.

Value
Numeric estimate of critical thermal minimum (°C)

get_e

Estimate the activation energy of a thermal performance curve

Description
Estimate the activation energy of a thermal performance curve

Usage
get_e(model)

Arguments
model: nls model object that contains a model of a thermal performance curve

Details
Fits a modified-Boltzmann equation to all raw data below the optimum temperature (°C; as estimated by get_topt).
get_lower_lims

Value

Numeric estimate of activation energy (eV)

get_eh

Estimate the deactivation energy of a thermal performance curve

Description

Estimate the deactivation energy of a thermal performance curve

Usage

get_eh(model)

Arguments

model nls model object that contains a model of a thermal performance curve

Details

Fits a modified-Boltzmann equation to all raw data beyond the optimum temperature (ºC; as estimated by `get_topt`).

Value

Numeric estimate of activation energy (eV)

get_lower_lims

Set broad lower limits on parameter values

Description

Sets wide lower limits on parameter values for each TPC model

Usage

get_lower_lims(x, y, model_name)

Arguments

x vector of temperature values
y vector of rate values
model_name the name of the model being fitted
**get_model_names**

**Value**
Named list of lower limits given the data and model being fitted

**Author(s)**
Daniel Padfield

---

**get_model_names**  
*Lists the models available in rTPC*

**Description**
Lists the models available in rTPC

**Usage**

```r
get_model_names()
```

**Value**
character vector of thermal performance curves available in rTPC

**Examples**

```r
get_model_names()
```

---

**get_q10**  
*Estimate the q10 value of a thermal performance curve*

**Description**
Estimate the q10 value of a thermal performance curve

**Usage**

```r
get_q10(model)
```

**Arguments**

- `model`  
  nls model object that contains a model of a thermal performance curve

**Details**
Fits the q10 portion of `rezende_2019` to all raw data below the optimum temperature (ºC; as estimated by `get_topt`).

**Value**
Numeric estimate of q10 value
**get_rmax**

*Estimate maximum rate of a thermal performance curve*

**Description**

Estimate maximum rate of a thermal performance curve

**Usage**

```
get_rmax(model)
```

**Arguments**

- `model`: nls model object that contains a model of a thermal performance curve

**Details**

Maximum rate is calculated by predicting over the temperature range using the previously estimated parameters and picking the maximum rate value. Predictions are done every 0.001 °C.

**Value**

Numeric estimate of maximum rate

---

**get_skewness**

*Estimates skewness of a thermal performance curve*

**Description**

Estimates skewness of a thermal performance curve

**Usage**

```
get_skewness(model)
```

**Arguments**

- `model`: nls model object that contains a model of a thermal performance curve

**Details**

Skewness is calculated from the values of activation energy (e) and deactivation energy (eh) as: skewness = e - eh. A negative skewness indicates the TPC is left skewed, the drop after the optimum is steeper than the rise up to the optimum. A positive skewness means that the TPC is right skewed and a value of 0 would mean the curve is symmetrical around the optimum.


**get_start_vals**

Value

Numeric estimate of skewness

---

**get_start_vals** Estimate start values for TPC fitting

---

**Description**

Estimates sensible start values for fitting thermal performance curves

**Usage**

get_start_vals(x, y, model_name)

**Arguments**

- **x**: vector of temperature values
- **y**: vector of rate values
- **model_name**: the name of the model being fitted

**Value**

Named list of start parameters given the data and model being fitted

**Author(s)**

Daniel Padfield

---

**get_thermalsafetymargin**

Estimate thermal safety margin of a thermal performance curve

---

**Description**

Estimate thermal safety margin of a thermal performance curve

**Usage**

get_thermalsafetymargin(model)

**Arguments**

- **model**: nls model object that contains a model of a thermal performance curve
Details

Thermal safety margin is calculated as: CTmax - Topt. This is calculated using the functions get_ctmax and get_topt.

Value

Numeric estimate of thermal safety margin (in °C)

---

get_thermaltolerance | Estimate thermal tolerance of a thermal performance curve

---

Description

Estimate thermal tolerance of a thermal performance curve

Usage

get_thermaltolerance(model)

Arguments

model | nls model object that contains a model of a thermal performance curve

Details

Thermal tolerance is calculated as: CTmax - CTmin. This is calculated using the functions get_ctmax and get_ctmin.

Value

Thermal tolerance (in °C)

---

get_topt | Estimate optimum temperature of a thermal performance curve

---

Description

Estimate optimum temperature of a thermal performance curve

Usage

get_topt(model)

Arguments

model | nls model object that contains a model of a thermal performance curve
Details

Optimum temperature (°C) is calculated by predicting over the temperature range using the previously estimated parameters and keeping the temperature where the largest rate value occurs. Predictions are done every 0.001 °C so the estimate of optimum temperature should be accurate up to 0.001 °C.

Value

Numeric estimate of optimum temperature (in °C)

get_upper_lims

Set broad upper limits on parameter values

Description

Sets wide upper limits on parameter values for each TPC model

Usage

get_upper_lims(x, y, model_name)

Arguments

x vector of temperature values
y vector of rate values
model_name the name of the model being fitted

Value

Named list of upper limits given the data and model being fitted

Author(s)

Daniel Padfield
Description

Hinshelwood model for fitting thermal performance curves

Usage

hinshelwood_1947(temp, a, e, b, eh)

Arguments

temp  temperature in degrees centigrade
a     pre-exponential constant for the activation energy
e     activation energy (eV)
b     pre-exponential constant for the deactivation energy
eh    de-activation energy (eV)

Details

Equation:

\[ rate = a \cdot exp\left(\frac{-e}{(temp+273.15)}\right) - b \cdot exp\left(\frac{-eh}{(temp+273.15)}\right) \]

where \( k \) is Boltzmann’s constant with a value of 8.62e-05

Start values in get_start_vals are taken from the literature.

Limits in get_lower_lims and get_upper_lims are based on extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model difficult to fit.

References

**Examples**

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'hinshelwood_1947')
mod <- nls.multstart::nls_multstart(rate~hinshelwood_1947(temp = temp,a, e, b, eh),
data = d,
iter = c(5,5,5,5),
start_lower = start_vals - 1,
start_upper = start_vals + 1,
lower = get_lower_lims(d$temp, d$rate, model_name = 'hinshelwood_1947'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'hinshelwood_1947'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()
```

---

**joehnk_2008**

*Jöhnk model for fitting thermal performance curves*

**Description**

Jöhnk model for fitting thermal performance curves

**Usage**

`joehnk_2008(temp, rmax, topt, a, b, c)`

**Arguments**

- `temp` temperature in degrees centigrade
Equation:

\[ \text{rate} = r_{\text{max}} \left( 1 + a \left( \frac{b\text{temp} - t_{\text{opt}} - 1}{\ln(b)} - \frac{\ln(c)}{\ln(c)} (c\text{temp} - t_{\text{opt}} - 1) \right) \right) \]

Start values in `get_start_vals` are derived from the data or sensible values from the literature. Limits in `get_lower_lims` and `get_upper_lims` are based on extreme values that are unlikely to occur in ecological settings.

Value

A numeric vector of rate values based on the temperatures and parameter values provided to the function.

Note

Generally we found this model easy to fit.

References


Examples

```R
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'joehnk_2008')
# fit model
mod <- nls_multstart::nls_multstart(rate~joehnk_2008(temp = temp, rmax, topt, a, b, c),
data = d,
iter = c(3,3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'joehnk_2008'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'joehnk_2008'),
```
johnsonlewin_1946

johnsonlewin_1946

Johnson-Lewin model for fitting thermal performance curves

Description

Johnson-Lewin model for fitting thermal performance curves

Usage

johnsonlewin_1946(temp, r0, e, eh, topt)

Arguments

temp temperature in degrees centigrade

r0 scaling parameter

e activation energy (eV)

eh high temperature de-activation energy (eV)

topt optimum temperature (ºC)

Details

Equation:

\[ \text{rate} = \frac{r_0 \cdot \exp\left(-\frac{e}{k \cdot (temp + 273.15)}\right)}{1 + \exp}\]

where k is Boltzmann’s constant with a value of 8.62e-05.

Start values in get_start_vals are derived from the data.

Limits in get_lower_lims and get_upper_lims are derived from the data or based extreme values that are unlikely to occur in ecological settings.
Value

A numeric vector of rate values based on the temperatures and parameter values provided to the function.

Note

Generally we found this model difficult to fit.

References


Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'johnsonlewin_1946')
# fit model
mod <- suppressWarnings(nls.multstart::nls_multstart(rate~johnsonlewin_1946(temp, r0, e, eh, topt),
                                             data = d,
                                             iter = c(5,5,5,5),
                                             start_lower = start_vals - 1,
                                             start_upper = start_vals + 1,
                                             lower = get_lower_lims(d$temp, d$rate, model_name = 'johnsonlewin_1946'),
                                             upper = get_upper_lims(d$temp, d$rate, model_name = 'johnsonlewin_1946'),
                                             supp_errors = 'Y',
                                             convergence_count = FALSE))

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()
```
**Description**

Kamykowski model for fitting thermal performance curves

**Usage**

```
kamykowski_1985(temp, tmin, tmax, a, b, c)
```

**Arguments**

- `temp`: temperature in degrees centigrade
- `tmin`: low temperature (ºC) at which rates become negative
- `tmax`: high temperature (ºC) at which rates become negative
- `a`: parameter with no biological meaning
- `b`: parameter with no biological meaning
- `c`: parameter with no biological meaning

**Details**

Equation:

\[
rate = a \cdot (1 - e^{b(t_{\text{temp}} - t_{\text{min}})}) \cdot (1 - e^{c(t_{\text{max}} - \text{temp})})
\]

Start values in `get_start_vals` are derived from the data or sensible values from the literature. Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based extreme values that are unlikely to occur in ecological settings.

**Value**

a numeric vector of rate values based on the temperatures and parameter values provided to the function

**Note**

Generally we found this model easy to fit.

**References**

Examples

# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'kamykowski_1985')

# fit model
mod <- nls_multstart::nls_multstart(rate = kamykowski_1985(temp = temp, tmin, tmax, a, b, c),
data = d,
iter = c(3,3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'kamykowski_1985'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'kamykowski_1985'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

---

**lactin2_1995**

Lactin2 model for fitting thermal performance curves

Description

Lactin2 model for fitting thermal performance curves

Usage

lactin2_1995(temp, a, b, tmax, delta_t)

Arguments

temp temperature in degrees centigrade
Equation:  
\[
rate = \exp^{a \cdot temp} - \exp^{a \cdot t_{\text{max}} - \frac{(t_{\text{max}} - \text{temp})}{\delta t}} + b
\]

Details  
Start values in `get_start_vals` are derived from the data or sensible values from the literature. Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

Value  
a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note  
Generally we found this model easy to fit.

References  

Examples  
```r  
# load in ggplot  
library(ggplot2)  

# subset for the first TPC curve  
data('chlorella_tpc')  
d <- subset(chlorella_tpc, curve_id == 1)  

# get start values and fit model  
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'lactin2_1995')  
mod <- nls_multstart::nls_multstart(rate~lactin2_1995(temp = temp, a, b, tmax, delta_t), 
data = d,  
iters = c(3,3,3),  
start_lower = start_vals - 10,  
start_upper = start_vals + 10,  
lower = get_lower_lims(d$temp, d$rate, model_name = 'lactin2_1995'),  
upper = get_upper_lims(d$temp, d$rate, model_name = 'lactin2_1995'),  
supp_errors = 'Y',  
```
convergence_count = FALSE)
# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

---

lrf_1991  Lobry-Rosso-Flandros (LRF) model for fitting thermal performance curves

Description

Lobry-Rosso-Flandros (LRF) model for fitting thermal performance curves

Usage

lrf_1991(temp, rmax, topt, tmin, tmax)

Arguments

temp  temperature in degrees centigrade
rmax  maximum rate at optimum temperature
topt  optimum temperature (ºC)
tmin  low temperature (ºC) at which rates become negative
tmax  high temperature (ºC) at which rates become negative

Details

Equation:

\[
rate = rmax \cdot \frac{(temp - t_{max}) \cdot (temp - t_{min})^2}{(t_{opt} - t_{min}) \cdot (t_{opt} - t_{min}) \cdot (temp - t_{opt}) - (t_{opt} - t_{max}) \cdot (t_{opt} + t_{min} - 2 \cdot temp)}
\]

Start values in get_start_vals are derived from the data.

Limits in get_lower_lims and get_upper_lims are derived from the data or based extreme values that are unlikely to occur in ecological settings.
**Value**

a numeric vector of rate values based on the temperatures and parameter values provided to the function

**Note**

Generally we found this model easy to fit.

**Author(s)**

Daniel Padfield

**References**


**Examples**

```r
# load in ggplot
library(ggplot2)
library(nls.multstart)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'sharpeschoolhigh_1981')
mod <- nls_multstart(rate~lrf_1991(temp = temp, rmax, topt, tmin, tmax),
data = d,
iter = c(3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'lrf_1991'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'lrf_1991'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
```
Modified gaussian model for fitting thermal performance curves

Usage

modifiedgaussian_2006(temp, rmax, topt, a, b)

Arguments

temp temperature in degrees centigrade
rmax maximum rate at optimum temperature
topt optimum temperature
a related to full curve width
b allows for asymmetry in the curve fit

Details

Equation:

\[ rate = r_{max} \cdot \exp\left(-0.5 \left(\frac{|temp - topt|}{a}\right)^b\right) \]

Start values in get_start_vals are derived from the data and gaussian_1987
Limits in get_lower_lims and get_upper_lims are based on extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model difficult to fit.

References

Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'modifiedgaussian_2006')
# fit model
mod <- nls.multstart::nls_multstart(rate~modifiedgaussian_2006(temp = temp, rmax, topt, a, b),
data = d,
iter = c(3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'modifiedgaussian_2006'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'modifiedgaussian_2006'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()
```

oneill_1972

_"O’Neill model for fitting thermal performance curves"

Description

O’Neill model for fitting thermal performance curves

Usage

```r
oneill_1972(temp, rmax, ctmax, topt, q10)
```

Arguments

temp temperature in degrees centigrade

maximum rate at optimum temperature
high temperature (°C) at which rates become negative
optimum temperature (°C)
defines the fold change in performance as a result of increasing the temperature by 10 °C

Equation:
\[
rate = r_{max} \cdot \left( \frac{ct_{max} - temp}{ct_{max} - t_{opt}} \right)^x \cdot \exp^{x \cdot \frac{temp - t_{opt}}{ct_{max} - t_{opt}}}
\]

where: \( x = \frac{w^2}{400} \cdot \left( 1 + \sqrt{1 + \frac{40}{w}} \right)^2 \)

and: \( w = (q_{10} - 1) \cdot (ct_{max} - t_{opt}) \)

Start values in get_start_vals are derived from the data and previous values in the literature
Limits in get_lower_lims and get_upper_lims are based on extreme values that are unlikely to occur in ecological settings.

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Generally we found this model easy to fit.


# load in ggplot
library(ggplot2)
# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)
# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'oneill_1972')
# fit model
mod <- nls.multstart::nls_multstart(rate~oneill_1972(temp = temp, rmax, ctmax, topt, q10),
data = d,
iter = c(4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'oneill_1972'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'oneill_1972'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()

---

pawar_2018

*Pawar model for fitting thermal performance curves*

**Description**

Pawar model for fitting thermal performance curves

**Usage**

pawar_2018(temp, r_tref, e, eh, topt, tref)

**Arguments**

- **temp**: temperature in degrees centigrade
- **r_tref**: rate at the standardised temperature, tref
- **e**: activation energy (eV)
- **eh**: high temperature de-activation energy (eV)
- **topt**: optimum temperature (°C)
- **tref**: standardisation temperature in degrees centigrade. Temperature at which rates are not inactivated by high temperatures
Details
This model is a modified version of sharpeSchoolHigh_1981 that explicitly models the optimum temperature. Equation:

\[
rate = \frac{r_{\text{tref}} \cdot e^{\frac{-1}{r_{\text{tref}} + 273.15} \cdot \frac{1}{\text{temp} + 273.15}}}{1 + \left(\frac{e_{\text{eh}}}{e_{\text{eh}}}\right) \cdot e^{\frac{-1}{r_{\text{tref}} + 273.15} \cdot \frac{1}{\text{topt} + 273.15}}}
\]

where \(k\) is Boltzmann's constant with a value of 8.62e-05.

Start values in `get_start_vals` are derived from the data.
Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

Value
a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note
Generally we found this model easy to fit.

Author(s)
Daniel Padfield

References

Examples
# load in ggplot
library(ggplot2)
library(nls.multstart)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'pawar_2018')
# fit model
mod <- nls_multstart(rate~pawar_2018(temp = temp, r_tref, e, eh, topt, tref = 20),
data = d,
iter = c(3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'pawar_2018'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'pawar_2018'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

---

**quadratic_2008**

**Quadratic model for fitting thermal performance curves**

**Description**

Quadratic model for fitting thermal performance curves

**Usage**

`quadratic_2008(temp, a, b, c)`

**Arguments**

- `temp`: temperature in degrees centigrade
- `a`: parameter that defines the rate at 0 °C
- `b`: parameter with no biological meaning
- `c`: parameter with no biological meaning

**Details**

Equation:

\[ rate = a + b \cdot temp + c \cdot temp^2 \]

Start values in `get_start_vals` are derived from the data using previous methods in the literature.

Limits in `get_lower_lims` and `get_upper_lims` are based on extreme values that are unlikely to occur in ecological settings.
Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References


Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'quadratic_2008')
mod <- nls.multstart::nls_multstart(rate~quadratic_2008(temp = temp, a, b, c),
data = d,
iter = c(4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'quadratic_2008'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'quadratic_2008'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()
```
Ratkowsky model for fitting thermal performance curves

Description

Ratkowsky model for fitting thermal performance curves

Usage

ratkowsky_1983(temp, tmin, tmax, a, b)

Arguments

temp temperature in degrees centigrade
tmin low temperature (ºC) at which rates become negative
tmax high temperature (ºC) at which rates become negative
a parameter defined as sqrt(rate)/(temp - tmin)
b empirical parameter needed to fit the data for temperatures beyond the optimum temperature

Details

Equation:

\[
rate = \left( a \cdot (temp - t_{min}) \right)^2 \cdot \left( 1 - \exp(b \cdot (temp - t_{max})) \right)^2
\]

Start values in get_start_vals are derived from the data and previous values in the literature.
Limits in get_lower_lims and get_upper_lims are based on extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References

Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'ratkowsky_1983')
# fit model
mod <- nls.multstart::nls_multstart(rate~ratkowsky_1983(temp = temp, tmin, tmax, a, b),
data = d,
iter = c(4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'ratkowsky_1983'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'ratkowsky_1983'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()
```

rezende_2019

Rezende model for fitting thermal performance curves

Description

Rezende model for fitting thermal performance curves

Usage

```r
rezende_2019(temp, q10, a, b, c)
```

Arguments

- `temp` temperature in degrees centigrade
q10 defines the fold change in performance as a result of increasing the temperature by 10 °C

a parameter describing shifts in rate

b parameter threshold temperature (°C) beyond which the downward curve starts

c parameter controlling the rate of decline beyond the threshold temperature, b

Details

Equation:

\[
\text{if } \text{temp} < b: \text{rate} = a \cdot 10^{\frac{\log_{10}(q10)}{\text{temp}}} \\
\text{if } \text{temp} > b: \text{rate} = a \cdot 10^{\frac{\log_{10}(q10)}{\text{temp}}} \cdot \left(1 - c \cdot (b - \text{temp})^2\right)
\]

Start values in get_start_vals are derived from the data and previous values in the literature.

Limits in get_lower_lims and get_upper_lims are based on extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References


Examples

# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'rezende_2019')
# fit model
mod <- nls.multstart::nls_multstart(rate~rezende_2019(temp = temp, q10, a, b, c),
data = d,
iter = c(4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'rezende_2019'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'rezende_2019'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

---

sharpeschoolfull_1981  *Full Sharpe-Schoolfield model for fitting thermal performance curves*

**Description**

Full Sharpe-Schoolfield model for fitting thermal performance curves

**Usage**

```
sharpeschoolfull_1981(temp, r_tref, e, el, tl, eh, th, tref)
```

**Arguments**

- `temp` temperature in degrees centigrade
- `r_tref` rate at the standardised temperature, tref
- `e` activation energy (eV)
- `el` low temperature de-activation energy (eV)
- `tl` temperature (°C) at which enzyme is 1/2 active and 1/2 suppressed due to low temperatures
- `eh` high temperature de-activation energy (eV)
- `th` temperature (°C) at which enzyme is 1/2 active and 1/2 suppressed due to high temperatures
- `tref` standardisation temperature in degrees centigrade. Temperature at which rates are not inactivated by either high or low temperatures
Details

Equation:

\[
rate = \frac{r_{tref} \cdot \exp\left(\frac{k}{T} (1 \text{temp} + 273.15 - 1 \text{tref} + 273.15)\right)}{1 + \exp\left(\frac{k}{T} (1 \text{tl} - 1 \text{temp} + 273.15)\right) + \exp\left(\frac{k}{T} (1 \text{th} - 1 \text{temp} + 273.15)\right)}
\]

where \(k\) is Boltzmann’s constant with a value of 8.62e-05.

Start values in get_start_vals are derived from the data.

Limits in get_lower_lims and get_upper_lims are derived from the data or based extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

Author(s)

Daniel Padfield

References


Examples

# load in ggplot
library(ggplot2)
library(nls.multstart)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'sharpeSchoolfull_1981')
# fit model
mod <- nls_multstart(rate~sharpeSchoolfull_1981(temp = temp, r_tref, e, el, tl, eh, th, tref = 20),
data = d,
iter = c(3,3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'sharpeSchoolfull_1981'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'sharpeSchoolfull_1981'),
supp_errors = 'Y',

sharpeSchoolfull_1981
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

sharpeSchoolhigh_1981  Sharpe-Schoolfield model (high temperature inactivation only) for fitting thermal performance curves

Description
Sharpe-Schoolfield model (high temperature inactivation only) for fitting thermal performance curves

Usage
sharpeSchoolhigh_1981(temp, r_tref, e, eh, th, tref)

Arguments

- **temp**: temperature in degrees centigrade
- **r_tref**: rate at the standardised temperature, tref
- **e**: activation energy (eV)
- **eh**: high temperature de-activation energy (eV)
- **th**: temperature (ºC) at which enzyme is 1/2 active and 1/2 suppressed due to high temperatures
- **tref**: standardisation temperature in degrees centigrade. Temperature at which rates are not inactivated by high temperatures

Details
Equation:

\[
rate = \frac{r_{tref} \times \exp^{\frac{e}{k(b + \frac{1}{temp} + 273.15) - \frac{1}{r_{tref} + 273.15)}}}{1 + \exp^{\frac{eh}{k(b + \frac{1}{th} + 273.15)}}}
\]

where \(k\) is Boltzmann’s constant with a value of 8.62e-05.

Start values in get_start_vals are derived from the data.

Limits in get_lower_lims and get_upper_lims are derived from the data or based extreme values that are unlikely to occur in ecological settings.
Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

Author(s)

Daniel Padfield

References


Examples

# load in ggplot
library(ggplot2)
library(nls.multstart)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'sharpeschoolhigh_1981')
# fit model
mod <- nls_multstart(rate~sharpeschoolhigh_1981(temp = temp, r_tref, e, eh, th, tref = 20),
data = d,
iter = c(3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'sharpeschoolhigh_1981'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'sharpeschoolhigh_1981'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()
Sharpe-Schoolfield model (low temperature inactivation only) for fitting thermal performance curves

**Description**
Sharpe-Schoolfield model (low temperature inactivation only) for fitting thermal performance curves

**Usage**
```
sharpeSchoollow_1981(temp, r_tref, e, el, tl, tref)
```

**Arguments**
- **temp**: temperature in degrees centigrade
- **r_tref**: rate at the standardised temperature, tref
- **e**: activation energy (eV)
- **el**: low temperature de-activation energy (eV)
- **tl**: temperature (ºC) at which enzyme is 1/2 active and 1/2 suppressed due to low temperatures
- **tref**: standardisation temperature in degrees centigrade. Temperature at which rates are not inactivated by high temperatures

**Details**

\[
rate = \frac{r_{tref} \cdot \exp \left( \frac{-e}{k (temp + 273.15)} \right)}{1 + \exp \left( \frac{el}{k (tl - temp + 273.15)} \right)}
\]

where k is Boltzmann’s constant with a value of 8.62e-05.

Start values in `get_start_vals` are derived from the data.

Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

**Value**
a numeric vector of rate values based on the temperatures and parameter values provided to the function

**Note**
Generally we found this model easy to fit.
Author(s)
Daniel Padfield

References

Examples
# load in ggplot
library(ggplot2)
library(nls.multstart)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'sharpeschoollow_1981')
# fit model
mod <- nls_multstart(rate~sharpeschoollow_1981(temp = temp, r_tref, e, el, tl, tref = 20),
data = d,
iter = c(3,3,3,3),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
lower = get_lower_lims(d$temp, d$rate, model_name = 'sharpeschoollow_1981'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'sharpeschoollow_1981'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()
Description

Spain model for fitting thermal performance curves

Usage

spain_1982(temp, a, b, c, r0)

Arguments

temp  temperature in degrees centigrade
a     constant that determines the steepness of the rising portion of the curve
b     constant that determines the position of topt
b     constant that determines the steepness of the decreasing part of the curve
r0    the apparent rate at 0 ºC

Details

Equation:

\[ \text{rate} = r_0 \cdot e^{a \cdot \text{temp}} \cdot (1 - b \cdot e^{c \cdot \text{temp}}) \]

Start values in get_start vals are derived from the data or plucked from thin air.
Limits in get_lower_lims and get_upper_lims are derived from the data or plucked from thin air.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References

BASIC Microcomputer Models in Biology. Addison-Wesley, Reading, MA. 1982

Examples

# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'spain_1982')
# fit model
mod <- nls.multstart::nls_multstart(rate~spain_1982(temp = temp, a, b, c, r0),
data = d,
iter = c(3,3,3,3),
start_lower = start_vals - 1,
start_upper = start_vals + 1,
lower = get_lower_lims(d$temp, d$rate, model_name = 'spain_1982'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'spain_1982'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()

thomas_2012

Thomas model (2012) for fitting thermal performance curves

Description

Thomas model (2012) for fitting thermal performance curves

Usage

thomas_2012(temp, a, b, c, topt)

Arguments

temp temperature in degrees centigrade

a arbitrary constant

b arbitrary constant

c the range of temperatures over which growth rate is positive, or the thermal
niche width (°C)

topt determines the location of the maximum of the quadratic portion of this function.
When b = 0, tref would equal topt
Details

Equation:

\[ \text{rate} = a \cdot \exp^{b \cdot \text{temp}} \left( 1 - \left( \frac{\text{temp} - t_{\text{opt}}}{c} \right)^2 \right) \]

Start values in `get_start_vals` are derived from the data.
Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

Value

a numeric vector of rate values based on the temperatures and parameter values provided to the function

Note

Generally we found this model easy to fit.

References


Examples

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'thomas_2012')
# fit model
mod <- nls.multstart::nls_multstart(rate~thomas_2012(temp = temp, a, b, c, topt),
data = d, iter = c(4,4,4,4),
start_lower = start_vals - 1,
start_upper = start_vals + 2,
lower = get_lower_lims(d$temp, d$rate, model_name = 'thomas_2012'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'thomas_2012'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)
```
# plot
ggplot(preds) +
geom_point(aes(temp, rate), d) +
geom_line(aes(temp, .fitted), col = 'blue') +
theme_bw()

### Description

Thomas model (2017) for fitting thermal performance curves

### Usage

thomas_2017(temp, a, b, c, d, e)

### Arguments

- **temp**: temperature in degrees centigrade
- **a**: birth rate at 0 °C
- **b**: describes the exponential increase in birth rate with increasing temperature
- **c**: temperature-independent mortality term
- **d**: along with e controls the exponential increase in mortality rates with temperature
- **e**: along with d controls the exponential increase in mortality rates with temperature

### Details

Equation:

\[
\text{rate} = a \cdot \exp^{b \cdot \text{temp}} - (c + d \cdot \exp^{e \cdot \text{temp}})
\]

Start values in get_start_vals are derived from the data.

Limits in get_lower_lims and get_upper_lims are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

### Value

A numeric vector of rate values based on the temperatures and parameter values provided to the function

### Note

Generally we found this model easy to fit.
weibull_1995

Weibull model for fitting thermal performance curves

Description

Weibull model for fitting thermal performance curves

Usage

weibull_1995(temp, a, topt, b, c)
**Arguments**

- **temp**: temperature in degrees centigrade
- **a**: scale the height of the curve
- **topt**: optimum temperature
- **b**: defines the breadth of the curve
- **c**: defines the curve shape

**Details**

Equation:

\[
rate = a \cdot \left( \frac{c - 1}{c} \right)^{\frac{b}{c}} \left( \frac{\text{temp} - \text{topt}}{b} + \left( \frac{c - 1}{c} \right) \right)^{\frac{c - 1}{c}} \exp \left( -\frac{\text{temp} - \text{topt}}{b} + \left( \frac{c - 1}{c} \right) \right)^c + \frac{c - 1}{c}
\]

Start values in `get_start_vals` are derived from the data.

Limits in `get_lower_lims` and `get_upper_lims` are derived from the data or based on extreme values that are unlikely to occur in ecological settings.

**Value**

A numeric vector of rate values based on the temperatures and parameter values provided to the function.

**Note**

Generally we found this model easy to fit.

**References**


**Examples**

```r
# load in ggplot
library(ggplot2)

# subset for the first TPC curve
data('chlorella_tpc')
d <- subset(chlorella_tpc, curve_id == 1)

# get start values and fit model
start_vals <- get_start_vals(d$temp, d$rate, model_name = 'weibull_1995')
# fit model
mod <- nls_multstart::nls_multstart(rate~weibull_1995(temp = temp, a, topt, b, c),
data = d,
iter = c(4,4,4,4),
start_lower = start_vals - 10,
start_upper = start_vals + 10,
```
lower = get_lower_lims(d$temp, d$rate, model_name = 'weibull_1995'),
upper = get_upper_lims(d$temp, d$rate, model_name = 'weibull_1995'),
supp_errors = 'Y',
convergence_count = FALSE)

# look at model fit
summary(mod)

# get predictions
preds <- data.frame(temp = seq(min(d$temp), max(d$temp), length.out = 100))
preds <- broom::augment(mod, newdata = preds)

# plot
ggplot(preds) +
  geom_point(aes(temp, rate), d) +
  geom_line(aes(temp, .fitted), col = 'blue') +
  theme_bw()
Index

* dataset
  - bacteria_tpc, 3
  - chlorella_tpc, 10

bacteria_tpc, 3
beta_2012, 4
boatman_2017, 5
briere2_1999, 7

calc_params, 9
chlorella_tpc, 10
delong_2017, 11
deutsch_2008, 12
flinn_1991, 14
gaussian_1987, 16
get_breadth, 17
get_breadth(), 9
get_ctmax, 18
get_ctmax(), 9
get_ctmin, 19
get_ctmin(), 9
get_e, 19
get_e(), 9
get_eh, 20
get_eh(), 9
get_lower_lims, 20
get_model_names, 21
get_q10, 21
get_q10(), 9
get_rmax, 22
get_rmax(), 9
get_skewness, 22
get_skewness(), 9
get_start_vals, 23
get_thermalsafetymargin, 23
get_thermalsafetymargin(), 9
get_thermaltolerance, 24
get_thermaltolerance(), 9
get_topt, 24
get_topt(), 9
get_upper_lims, 25

hinshelwood_1947, 26
joehnk_2008, 27
johnsonlewin_1946, 29
kamykowski_1985, 31
lactin2_1995, 32
lrf_1991, 34
modifiedgaussian_2006, 36
oneill_1972, 37
pawar_2018, 39
quadratic_2008, 41
ratkowsky_1983, 43
rezende_2019, 44

sharpeschoolfull_1981, 46
sharpschoolhigh_1981, 48
sharpschoollow_1981, 50
spain_1982, 51
thomas_2012, 53
thomas_2017, 55
weibull_1995, 56