Package 'flexFitR'

April 16, 2025

Type Package

Title Flexible Non-Linear Least Square Model Fitting

Version 1.2.0

Maintainer Johan Aparicio <aparicioarce@wisc.edu>

Description Provides tools for flexible non-linear least squares model fitting using general-purpose optimization techniques. The package supports a variety of optimization algorithms, including those provided by the 'optimx' package, making it suitable for handling complex non-linear models. Features include parallel processing support via the 'future' and 'foreach' packages, comprehensive model diagnostics, and visualization capabilities. Implements methods described in Nash and Varadhan (2011, <doi:10.18637/jss.v043.i09>).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

RoxygenNote 7.3.2

Author Johan Aparicio [cre, aut],
Jeffrey Endelman [aut],
University of Wisconsin Madison [cph]

Imports agriutilities, doFuture, dplyr, foreach, future, ggplot2, numDeriv, optimx, progressr, rlang, subplex, tibble, tidyr

URL https://apariciojohan.github.io/flexFitR/, https://github.com/AparicioJohan/flexFitR

BugReports https://github.com/AparicioJohan/flexFitR/issues

Suggests BB, dfoptim, ggpubr, kableExtra, knitr, lbfgsb3c, marqLevAlg, purrr, rmarkdown, ucminf

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2025-04-16 16:40:08 UTC

Contents

anova.modeler *Extra Sum-of-Squares F-Test for* modeler *objects*

Description

Perform an extra sum-of-squares F-test to compare two nested models of class modeler. This test assesses whether the additional parameters in the full model significantly improve the fit compared to the reduced model.

Usage

```
## S3 method for class 'modeler'
anova(object, full_model = NULL, ...)
```

Arguments

object	An object of class modeler representing the reduced model with fewer parameters.
full_model	An optional object of class modeler representing the full model with more parameters.
...	Additional parameters for future functionality.

Value

A tibble containing columns with the F-statistic and corresponding p-values, indicating whether the full model provides a significantly better fit than the reduced model.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
dt <- data.frame(X = 1:6, Y = c(12, 16, 44, 50, 95, 100))
mo_1 <- modeler(dt, X, Y, fn = "fn_lin", param = c(m = 10, b = -5))
plot(mo_1)
mo_2 <- modeler(dt, X, Y, fn = "fn_quad", param = c(a = 1, b = 10, c = 5))
plot(mo_2)
anova(mo_1, mo_2)
```

augment *Augment a* modeler *object with influence diagnostics*

Description

This function computes various influence diagnostics, including standardized residuals, studentized residuals, and Cook's distance, for an object of class modeler.

Usage

```
augment(x, id = NULL, metadata = TRUE, ...)
```

Arguments

x	An object of class modeler.
id	Optional unique identifier to filter by a specific group. Default is NULL.
metadata	Logical. If TRUE, metadata is included with the predictions. Default is FALSE
...	Additional parameters for future functionality.

Value

A tibble containing the following columns:

uid	Unique identifier for the group.
fn_name	Function name associated with the model.
x	Predictor variable values.
y	Observed response values.
.fitted	Fitted values from the model.
.resid	Raw residuals (observed - fitted).
.hat	Leverage values for each observation.
.cooksd	Cook's distance for each observation.
.std.resid	Standardized residuals.
.stud.resid	Studentized residuals.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_logistic",
    parameters = c(a = 0.199, t0 = 47.7, k = 100),
    subset = 2
  )
print(mod_1)
augment(mod_1)
```

---

c.modeler *Combine objects of class* modeler

Description

Combine objects of class `modeler`. Use with caution, some functions might not work as expected.

Usage

```
## S3 method for class 'modeler'
c(...)
```

Arguments

`...`	Objects of class `modeler`, typically the result of calling `modeler()`.

Value

A `modeler` object.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_logistic",
```

```
    parameters = c(a = 0.199, t0 = 47.7, k = 100),
    subset = 1:2
  )
mod_2 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 100),
    subset = 1:2
  )
mod <- c(mod_1, mod_2)
print(mod)
plot(mod, id = 1:2)
```

`coef.modeler` *Coefficients for an object of class* `modeler`

Description

Extract the estimated coefficients from an object of class `modeler`.

Usage

```
## S3 method for class 'modeler'
coef(object, id = NULL, metadata = FALSE, df = FALSE, ...)
```

Arguments

`object`	An object of class `modeler`, typically the result of calling the `modeler()` function.
`id`	An optional unique identifier to filter by a specific group. Default is `NULL`.
`metadata`	Logical. If `TRUE`, metadata is included along with the coefficients. Default is `FALSE`.
`df`	Logical. If `TRUE`, the degrees of freedom for the fitted model are returned alongside the coefficients. Default is `FALSE`.
`...`	Additional parameters for future functionality.

Value

A `data.frame` containing the model's estimated coefficients, standard errors, and optional metadata or degrees of freedom if specified.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 2, 45)
  )
print(mod_1)
coef(mod_1, id = 2)
```

---

compute_tangent *Compute tangent line(s) from a* modeler *object*

Description

Computes the slope and intercept of the tangent line(s) to a fitted curve at one or more specified x-values.

Usage

```
compute_tangent(object, x = NULL, id = NULL)
```

Arguments

object	A fitted object of class modeler, created by modeler().
x	A numeric vector of x-values at which to compute tangent lines.
id	Optional vector of uids indicating which groups to compute tangent lines for. If NULL, all groups are used.

Value

A tibble with one row per tangent line and the following columns:

- uid: unique identifier of the group.
- fn_name: Name of the fitted function.
- x: x-value where the tangent line is evaluated.
- y: Fitted y-value at x.
- slope: First derivative (slope of tangent) at x.
- intercept: y-intercept of the tangent line.

Examples

```
library(flexFitR)
library(ggplot2)
data(dt_potato)
mod <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_logistic",
    parameters = c(a = 4, t0 = 40, k = 100),
    subset = 2
  )
plot(mod)
tl <- compute_tangent(mod, x = c(48.35, 65))
print(tl)
plot(mod) +
  geom_abline(
    data = tl,
    mapping = aes(slope = slope, intercept = intercept),
    linetype = 2,
    color = "blue"
  ) +
  geom_point(
    data = tl,
    mapping = aes(x = x, y = y),
    shape = 8,
    color = "blue",
    size = 2
  )
```

---

confint.modeler *Confidence intervals for a modeler object*

Description

Extract confidence intervals for the estimated parameters of an object of class `modeler`.

Usage

```
## S3 method for class 'modeler'
confint(object, parm = NULL, level = 0.95, id = NULL, ...)
```

Arguments

`object`	An object of class `modeler`, typically the result of calling the `modeler()` function.

parm	A character vector specifying which parameters should have confidence intervals calculated. If NULL, confidence intervals for all parameters are returned. Default is NULL.
level	A numeric value indicating the confidence level for the intervals. Default is 0.95, corresponding to a 95% confidence interval.
id	An optional unique identifier to filter by a specific group. Default is NULL.
...	Additional parameters for future functionality.

Value

A tibble containing the lower and upper confidence limits for each specified parameter.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 35, 45)
  )
print(mod_1)
confint(mod_1)
```

---

dt_potato *Drone-derived data from a potato breeding trial*

Description

Canopy and Green Leaf Index for a potato trial arranged in a p-rep design.

Usage

```
dt_potato
```

Format

A tibble with 1372 rows and 8 variables:

Trial chr trial name

Plot dbl denoting the unique plot id

Row dbl denoting the row coordinate

Range dbl denoting range coordinate

gid chr denoting the genotype id

DAP dbl denoting Days after planting

Canopy dbl Canopy UAV-Derived

GLI dbl Green Leaf Index UAV-Derived

Source

UW - Potato Breeding Program

`explorer` *Explore data*

Description

Explores data from a data frame in wide format.

Usage

```
explorer(data, x, y, id, metadata)
```

Arguments

`data`	A `data.frame` containing the input data for analysis.
`x`	The name of the column in `data` that contains x points.
`y`	The names of the columns in `data` that contain the variables to be analyzed.
`id`	The names of the columns in `data` that contains a grouping variable.
`metadata`	The names of the columns in `data` to keep across the analysis.

Details

This function helps to explore the dataset before being analyzed with `modeler()`.

Value

An object of class `explorer`, which is a list containing the following elements:

`summ_vars` A data.frame containing summary statistics for each trait at each x point, including minimum, mean, median, maximum, standard deviation, coefficient of variation, number of non-missing values, percentage of missing values, and percentage of negative values.

`summ_metadata` A data.frame summarizing the metadata.

`locals_min_max` A data.frame containing the local minima and maxima of the mean y values over x.

`dt_long` A data.frame in long format, with columns for uid, metadata, var, x, and y

`metadata` A character vector with the names of the variables to keep across.

Examples

```
library(flexFitR)
data(dt_potato)
results <- dt_potato |>
  explorer(
    x = DAP,
    y = c(Canopy, GLI),
    id = Plot,
    metadata = c(gid, Row, Range)
  )
names(results)
head(results$summ_vars)
plot(results, label_size = 4, signif = TRUE, n_row = 2)
# New data format
head(results$dt_long)
```

`fitted.modeler` *Extract fitted values from a* `modeler` *object*

Description

Extract fitted values from a `modeler` object

Usage

```
## S3 method for class 'modeler'
fitted(object, ...)
```

Arguments

`object`	An object of class ‘modeler‘
`...`	Additional parameters for future functionality.

Value

A numeric vector of fitted values.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 2, 45)
  )
fitted(mod_1)
```

---

fn_exp2_exp *Super-exponential exponential function*

Description

A piecewise function that models an initial exponential phase with quadratic time dependence, followed by a second exponential phase with a different growth rate.

Usage

```
fn_exp2_exp(t, t1, t2, alpha, beta)
```

Arguments

t	A numeric vector of input values (e.g., time).
t1	The onset time of the response. The function is 0 for all values less than t1.
t2	The transition time between the two exponential phases. Must be greater than t1.
alpha	The curvature-controlled exponential rate during the first phase (t1 to t2).
beta	The exponential growth rate after t2.

Details

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_exp2_exp",
  params = c(t1 = 35, t2 = 55, alpha = 1 / 600, beta = -1 / 30),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3,
  y_auc_label = 0.15
)
```

---

fn_exp2_lin *Super-exponential linear function*

Description

A piecewise function that models an initial exponential growth phase based on a squared time difference, followed by a linear phase.

Usage

```
fn_exp2_lin(t, t1, t2, alpha, beta)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The transition time between exponential and linear phases. Must be greater than `t1`.
`alpha`	The exponential growth rate controlling the curvature of the exponential phase.
`beta`	The slope of the linear phase after `t2`.

Details

The exponential section rises gradually from 0 at `t1` and accelerates as time increases. The linear section starts at `t2` with a value matching the end of the exponential phase, ensuring continuity but not necessarily matching the derivative.

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_exp2_lin",
  params = c(t1 = 35, t2 = 55, alpha = 1 / 600, beta = -1 / 80),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

---

fn_exp_exp *Double-exponential function*

Description

A piecewise function with two exponential phases. The first exponential phase occurs between `t1` and `t2`, and the second phase continues after `t2` with a potentially different growth rate. The function ensures continuity at the transition point but not necessarily smoothness (in derivative).

Usage

```
fn_exp_exp(t, t1, t2, alpha, beta)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The transition time between the two exponential phases. Must be greater than `t1`.
`alpha`	The exponential growth rate during the first phase (`t1` to `t2`).
`beta`	The exponential growth rate after `t2`.

Details

The function rises from 0 starting at `t1` with exponential growth rate `alpha`, and transitions to a second exponential phase with rate `beta` at `t2`. The value at the transition point is preserved, ensuring continuity.

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_exp_exp",
  params = c(t1 = 35, t2 = 55, alpha = 1 / 20, beta = -1 / 30),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3,
  y_auc_label = 0.2
)
```

---

fn_exp_lin *Exponential-linear function*

Description

A piecewise function that models a response with an initial exponential growth phase followed by a linear phase. Commonly used to describe processes with rapid early increases that slow into a linear trend, while maintaining continuity.

Usage

```
fn_exp_lin(t, t1, t2, alpha, beta)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The transition time between exponential and linear phases. Must be greater than `t1`.
`alpha`	The exponential growth rate during the exponential phase.
`beta`	The slope of the linear phase after `t2`.

Details

The exponential segment starts from 0 at `t1`, and the linear segment continues smoothly from the end of the exponential part. This ensures value continuity at `t2`, but not necessarily smoothness in slope.

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_exp_lin",
  params = c(t1 = 35, t2 = 55, alpha = 1 / 20, beta = -1 / 40),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

---

fn_lin *Linear function*

Description

A basic linear function of the form `f(t) = m * t + b`, where `m` is the slope and `b` is the intercept.

Usage

```
fn_lin(t, m, b)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`m`	The slope of the line.
`b`	The intercept (function value when `t = 0`).

Details

Value

A numeric vector of the same length as `t`, giving the linear function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_lin",
  params = c(m = 2, b = 10),
  interval = c(0, 108),
  n_points = 2000
)
```

`fn_lin_logis` *Linear-logistic function*

Description

A piecewise function that models an initial linear increase followed by a logistic saturation.

Usage

```
fn_lin_logis(t, t1, t2, k)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The transition time between the linear and logistic phases. Must be greater than `t1`.
`k`	The plateau height. The function transitions toward this value in the logistic phase.

Details

The linear segment rises from 0 starting at `t1`, and the logistic segment begins at `t2`, smoothly approaching the plateau value `k`.

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_lin_logis",
  params = c(t1 = 35, t2 = 50, k = 100),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

`fn_lin_plat` *Linear plateau function*

Description

A simple piecewise function that models a linear increase from zero to a plateau. The function rises linearly between two time points and then levels off at a constant value.

Usage

```
fn_lin_plat(t, t1 = 45, t2 = 80, k = 0.9)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The time at which the plateau begins. Must be greater than `t1`.
`k`	The height of the plateau. The function linearly increases from 0 to `k` between `t1` and `t2`, then remains constant.

Details

This function is continuous but not differentiable at `t1` and `t2` due to the piecewise transitions. It is often used in agronomy and ecology to describe growth until a resource limit or developmental plateau is reached.

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_lin_plat",
  params = c(t1 = 34.9, t2 = 61.8, k = 100),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

`fn_lin_pl_lin` *Linear plateau linear function*

Description

A piecewise function that models an initial linear increase up to a plateau, maintains that plateau for a duration, and then decreases linearly.

Usage

```
fn_lin_pl_lin(t, t1, t2, t3, k, beta)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The time when the linear growth phase ends and the plateau begins. Must be greater than `t1`.
`t3`	The time when the plateau ends and the linear decline begins. Must be greater than `t2`.
`k`	The height of the plateau. The first linear phase increases to this value, which remains constant until `t3`.
`beta`	The slope of the final linear phase (typically negative), controlling the rate of decline after `t3`.

Details

The function transitions continuously between all three phases but is not differentiable at the transition points `t1`, `t2`, and `t3`.

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_lin_pl_lin",
  params = c(t1 = 38.7, t2 = 62, t3 = 90, k = 0.32, beta = -0.01),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

`fn_lin_pl_lin2` *Linear plateau linear with constrains*

Description

A piecewise function that models an initial linear increase to a plateau, followed by a specified duration of stability, and then a linear decline. This version parameterizes the plateau using its duration rather than an explicit end time, making it convenient for box type of constraints optimizations.

Usage

```
fn_lin_pl_lin2(t, t1, t2, dt, k, beta)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`t1`	The onset time of the response. The function is 0 for all values less than `t1`.
`t2`	The time when the linear growth phase ends and the plateau begins. Must be greater than `t1`.
`dt`	The duration of the plateau phase. The plateau ends at `t2 + dt`.
`k`	The height of the plateau. The linear phase increases to this value, which remains constant for `dt` units of time.
`beta`	The slope of the decline phase that begins after the plateau. Typically negative.

Details

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_lin_pl_lin2",
  params = c(t1 = 38.7, t2 = 62, dt = 28, k = 0.32, beta = -0.01),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

`fn_logistic` *Logistic function*

Description

A standard logistic function commonly used to model sigmoidal growth. The curve rises from near zero to a maximum value `k`, with inflection point at `t0` and growth rate `a`.

Usage

```
fn_logistic(t, a, t0, k)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`a`	The growth rate (steepness of the curve). Higher values lead to a steeper rise.
`t0`	The time of the inflection point (midpoint of the transition).
`k`	The upper asymptote or plateau (maximum value as `t -> Inf`).

Details

This is a classic sigmoid (S-shaped) curve that is symmetric around the inflection point `t0`.

Value

A numeric vector of the same length as `t`, representing the logistic function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_logistic",
  params = c(a = 0.199, t0 = 47.7, k = 100),
  interval = c(0, 108),
  n_points = 2000
)
```

fn_quad *Quadratic function*

Description

A standard quadratic function of the form `f(t) = a * t^2 + b * t + c`, where `a` controls curvature, `b` is the linear coefficient, and `c` is the intercept.

Usage

```
fn_quad(t, a, b, c)
```

Arguments

`t`	A numeric vector of input values (e.g., time).
`a`	The quadratic coefficient (curvature).
`b`	The linear coefficient (slope at the origin).
`c`	The intercept (function value when `t = 0`).

Details

This function represents a second-degree polynomial. The sign of `a` determines whether the parabola opens upward (`a > 0`) or downward (`a < 0`).

Value

A numeric vector of the same length as `t`, representing the quadratic function values.

Examples

```
library(flexFitR)
plot_fn(fn = "fn_quad", params = c(a = 1, b = 10, c = 5))
```

fn_quad_plat *Quadratic-plateau function*

Description

Computes a value based on a quadratic-plateau growth curve.

Usage

```
fn_quad_plat(t, t1 = 45, t2 = 80, b = 1, k = 100)
```

Arguments

t	A numeric vector of input values (e.g., time).
t1	The onset time of the response. The function is 0 for all values less than t1.
t2	The time at which the plateau begins. Must be greater than t1.
b	The initial slope of the curve at t1.
k	The plateau height. The function transitions to this constant value at t2.

Details

This function allows the user to specify the initial slope b. The curvature term is automatically calculated so that the function reaches the plateau value k exactly at t2. The transition to the plateau is continuous in value but not necessarily smooth in derivative.

Value

A numeric vector of the same length as t, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_quad_plat",
  params = c(t1 = 35, t2 = 80, b = 4, k = 100),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

---

fn_quad_pl_sm *Smooth Quadratic-plateau function*

Description

A piecewise function that models a quadratic increase from zero to a plateau value. The function is continuous and differentiable, modeling growth processes with a smooth transition to a maximum response.

Usage

```
fn_quad_pl_sm(t, t1, t2, k)
```

Arguments

t	A numeric vector of input values (e.g., time).
t1	The onset time of the response. The function is 0 for all values less than t1.
t2	The time at which the plateau begins. Must be greater than t1.
k	The plateau height. The function transitions to this constant value at t2.

Details

The coefficients of the quadratic section are chosen such that the curve passes through `(t1, 0)` and `(t2, k)` with a continuous first derivative (i.e., smooth transition).

Value

A numeric vector of the same length as `t`, representing the function values.

Examples

```
library(flexFitR)
plot_fn(
  fn = "fn_quad_pl_sm",
  params = c(t1 = 35, t2 = 80, k = 100),
  interval = c(0, 108),
  n_points = 2000,
  auc_label_size = 3
)
```

---

goodness_of_fit *Akaike's An Information Criterion for an object of class* `modeler`

Description

Generic function calculating Akaike's 'An Information Criterion' for fitted model object of class `modeler`.

Usage

```
## S3 method for class 'modeler'
AIC(object, ..., k = 2)

## S3 method for class 'modeler'
BIC(object, ...)
```

Arguments

`object`	An object inheriting from class `modeler` resulting of executing the function `modeler()`
`...`	Further parameters. For future improvements.
`k`	Numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC.

Value

A `tibble` with columns giving the corresponding AIC and BIC.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
dt <- data.frame(X = 1:6, Y = c(12, 16, 44, 50, 95, 100))
mo_1 <- modeler(dt, X, Y, fn = "fn_lin", param = c(m = 10, b = -5))
mo_2 <- modeler(dt, X, Y, fn = "fn_quad", param = c(a = 1, b = 10, c = 5))
AIC(mo_1)
AIC(mo_2)
BIC(mo_1)
BIC(mo_2)
```

---

inverse_predict.modeler

Inverse prediction from a `modeler` *object*

Description

Computes the x-value at which a fitted model reaches a user-specified response value (y-value).

Usage

```
## S3 method for class 'modeler'
inverse_predict(
  object,
  y,
  id = NULL,
  interval = NULL,
  tol = 1e-06,
  resolution = 1000,
  ...
)
```

Arguments

`object`	A fitted object of class `modeler`.
`y`	A numeric scalar giving the target y-value for which to compute the corresponding x.
`id`	Optional vector of `uids` for which to perform inverse prediction. If `NULL`, all groups are used.
`interval`	Optional numeric vector of length 2 specifying the interval in which to search for the root. If `NULL`, the interval is inferred from the range of the observed x-values.
`tol`	Numerical tolerance passed to `uniroot` for root-finding accuracy.
`resolution`	Integer. Number of grid points used to scan the interval.
`...`	Additional parameters for future functionality.

Details

The function uses numeric root-finding to solve `f(t, ...params) = y`. If no root is found in the interval, `NA` is returned.

Value

A `tibble` with one row per group, containing:

- `uid` – unique identifier of the group,
- `fn_name` – the name of the fitted function,
- `lower` and `upper` – the search interval used,
- `y` – the predicted y-value (from the function at the root),
- `x` – the x-value at which the function reaches `y`.

See Also

`predict.modeler`, `uniroot`

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 2, 45)
  )
print(mod_1)
inverse_predict(mod_1, y = 50)
inverse_predict(mod_1, y = 75, interval = c(20, 80))
```

---

list_funs *Print available functions in flexFitR*

Description

Print available functions in flexFitR

Usage

```
list_funs()
```

Value

A vector with available functions

Examples

```
library(flexFitR)
list_funs()
```

`list_methods` *Print available methods in flexFitR*

Description

Print available methods in flexFitR

Usage

```
list_methods(bounds = FALSE, check_package = FALSE)
```

Arguments

bounds	If TRUE, returns methods for box (or bounds) constraints. FALSE by default.
check_package	If TRUE, ensures solvers are installed. FALSE by default.

Value

A vector with available methods

Examples

```
library(flexFitR)
list_methods()
```

`logLik.modeler` *Extract Log-Likelihood for an object of class* `modeler`

Description

logLik for an object of class `modeler`

Usage

```
## S3 method for class 'modeler'
logLik(object, ...)
```

Arguments

object	An object inheriting from class modeler resulting of executing the function modeler()
...	Further parameters. For future improvements.

Value

A tibble with the Log-Likelihood for the fitted models.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
dt <- data.frame(X = 1:6, Y = c(12, 16, 44, 50, 95, 100))
mo_1 <- modeler(dt, X, Y, fn = "fn_lin", param = c(m = 10, b = -5))
plot(mo_1)
logLik(mo_1)
```

---

metrics *Metrics for an object of class* modeler

Description

Computes various performance metrics for a modeler object. The function calculates Sum of Squared Errors (SSE), Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R-squared).

Usage

```
metrics(x, by_grp = TRUE)
```

Arguments

x	An object of class ‘modeler‘ containing the necessary data to compute the metrics.
by_grp	Return the metrics by id? TRUE by default.

Details

Value

A data frame containing the calculated metrics grouped by uid, metadata, and variables.

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(1:2)
  )
plot(mod_1, id = c(1:2))
print(mod_1)
metrics(mod_1)
```

`modeler` *Modeler: Non-linear regression for curve fitting*

Description

A versatile function for performing non-linear least squares optimization on grouped data. It supports customizable optimization methods, flexible initial/fixed parameters, and parallel processing.

Usage

```
modeler(
  data,
  x,
  y,
  grp,
  keep,
  fn = "fn_lin_plat",
  parameters = NULL,
  lower = -Inf,
  upper = Inf,
  fixed_params = NULL,
  method = c("subplex", "pracmanm", "anms"),
  subset = NULL,
  options = modeler.options(),
  control = list()
)
```

Arguments

`data`	A `data.frame` containing the input data for analysis.
`x`	The name of the column in `data` representing the independent variable (x points).

`y` The name of the column in `data` containing the dependent variable to analyze (response variable).

`grp` Column(s) in `data` used as grouping variable(s). Defaults to `NULL`. (Optional)

`keep` Names of columns to retain in the output. Defaults to `NULL`. (Optional)

`fn` A string. The name of the function used for curve fitting. Example: `"fn_lin"`. Defaults to `"fn_lin_plat"`.

`parameters` A numeric vector, named list, or `data.frame` providing initial values for parameters:

Numeric vector Named vector specifying initial values (e.g., `c(k = 0.5, t1 = 30)`).

Data frame Requires a `uid` column with group IDs and parameter values for each group.

List Named list where parameter values can be numeric or expressions (e.g., `list(k = "max(y)", t1 = 40)`).

Defaults to `NULL`.

`lower` A numeric vector specifying lower bounds for parameters. Defaults to `-Inf` for all parameters.

`upper` A numeric vector specifying upper bounds for parameters. Defaults to `Inf` for all parameters.

`fixed_params` A list or `data.frame` for fixing specific parameters:

List Named list where parameter values can be numeric or expressions (e.g., `list(k = "max(y)", t1 = 40)`).

Data frame Requires a `uid` column for group IDs and fixed parameter values.

Defaults to `NULL`.

`method` A character vector specifying optimization methods. Check available methods using `list_methods()` and their dependencies using `optimx::checkallsolvers()`. Defaults to `c("subplex", "pracmanm", "anms")`.

`subset` A vector (optional) containing levels of `grp` to filter the data for analysis. Defaults to `NULL` (all groups are included).

`options` A list of additional options. See `modeler.options()`

`progress` Logical. If `TRUE` a progress bar is displayed. Default is `FALSE`. Try this before running the function: `progressr::handlers("progress", "beepr")`.

`parallel` Logical. If `TRUE` the model fit is performed in parallel. Default is `FALSE`.

`workers` The number of parallel processes to use. `parallel::detectCores()`

`trace` If `TRUE` , convergence monitoring of the current fit is reported in the console. `FALSE` by default.

`return_method` Logical. If `TRUE`, includes the optimization method used in the result. Default is `FALSE`.

`control` A list of control parameters to be passed to the optimization function. For example: `list(maxit = 500)`.

Value

An object of class `modeler`, which is a list containing the following elements:

`param` Data frame containing optimized parameters and related information.

`dt` Data frame with input data, fitted values, and residuals.

`metrics` Metrics and summary of the models.

`execution` Total execution time for the analysis.

`response` Name of the response variable analyzed.

`keep` Metadata retained based on the `keep` argument.

`fun` Name of the curve-fitting function used.

`parallel` List containing parallel execution details (if applicable).

`fit` List of fitted models for each group.

Examples

```
library(flexFitR)
data(dt_potato)
explorer <- explorer(dt_potato, x = DAP, y = c(Canopy, GLI), id = Plot)
# Example 1
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = GLI,
    grp = Plot,
    fn = "fn_lin_pl_lin",
    parameters = c(t1 = 38.7, t2 = 62, t3 = 90, k = 0.32, beta = -0.01),
    subset = 195
  )
plot(mod_1, id = 195)
print(mod_1)
# Example 2
mod_2 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = 195
  )
plot(mod_2, id = 195)
print(mod_2)
```

`performance` *Compare performance of different models*

Description

Computes indices of model performance for different models at once and hence allows comparison of indices across models.

Usage

```
performance(..., metrics = "all", metadata = FALSE, digits = 2)
```

Arguments

`...`	Multiple model objects (only of class 'modeler').
`metrics`	Can be "all" or a character vector of metrics to be computed (one or more of "logLik", "AIC", "AICc", "BIC", "Sigma", "SSE", "MAE", "MSE", "RMSE", "R2"). "all" by default.
`metadata`	Logical. If `TRUE`, metadata is included with the performance metrics. Default is `FALSE`.
`digits`	An integer. The number of decimal places to round the output. Default is 2.

Value

A data.frame with performance metrics for models in (...).

Examples

```
library(flexFitR)
data(dt_potato)
# Model 1
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 90),
    subset = 40
  )
print(mod_1)
# Model 2
mod_2 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_logistic",
```

```
    parameters = c(a = 0.199, t0 = 47.7, k = 100),
    subset = 40
  )
print(mod_2)
# Model 3
mod_3 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin",
    parameters = c(m = 20, b = 2),
    subset = 40
  )
print(mod_3)
performance(mod_1, mod_2, mod_3, metrics = c("AIC", "AICc", "BIC", "Sigma"))
```

`plot.explorer` *Plot an object of class* `explorer`

Description

Creates various plots for an object of class `explorer`. Depending on the specified type, the function can generate plots that show correlations between variables over x, correlations between x values for each variable, or the evolution of variables over x.

Usage

```
## S3 method for class 'explorer'
plot(
  x,
  type = "var_by_x",
  label_size = 4,
  signif = FALSE,
  method = "pearson",
  filter_var = NULL,
  id = NULL,
  n_row = NULL,
  n_col = NULL,
  base_size = 13,
  return_gg = FALSE,
  add_avg = FALSE,
  ...
)
```

Arguments

`x`	An object inheriting from class `explorer`, resulting from executing the function `explorer()`.

type	Character string or number specifying the type of plot to generate. Available options are: "var_by_x" or 1 Plots correlations between variables over x (default). "x_by_var" or 2 Plots correlations between x points for each variable (y). "evolution" or 3 Plot the evolution of the variables (y) over x. "xy" or 4 Scatterplot (x, y)
label_size	Numeric. Size of the labels in the plot. Default is 4. Only works with type 1 and 2.
signif	Logical. If TRUE, adds p-values to the correlation plot labels. Default is FALSE. Only works with type 1 and 2.
method	Character string specifying the method for correlation calculation. Available options are "pearson" (default), "spearman", and "kendall". Only works with type 1 and 2.
filter_var	Character vector specifying the variables to exclude from the plot.
id	Optional unique identifier to filter the evolution type of plot. Default is NULL. Only works with type 3.
n_row	Integer specifying the number of rows to use in facet_wrap(). Default is NULL. Only works with type 1 and 2.
n_col	Integer specifying the number of columns to use in facet_wrap(). Default is NULL. Only works with type 1 and 2.
base_size	Numeric. Base font size for the plot. Default is 13.
return_gg	Logical. If TRUE, returns the ggplot object instead of printing it. Default is FALSE.
add_avg	Logical. If TRUE, returns evolution plot with the average trend across groups. Default is FALSE.
...	Further graphical parameters for future improvements.

Value

A ggplot object and an invisible data.frame containing the correlation table when type is "var_by_x" or "x_by_var".

Examples

```
library(flexFitR)
data(dt_potato)
results <- explorer(dt_potato, x = DAP, y = c(Canopy, GLI), id = Plot)
table <- plot(results, label_size = 4, signif = TRUE, n_row = 2)
table
plot(results, type = "x_by_var", label_size = 4, signif = TRUE)
```

plot.modeler *Plot an object of class* modeler

Description

Creates several plots for an object of class modeler.

Usage

```
## S3 method for class 'modeler'
plot(
  x,
  id = NULL,
  type = 1,
  label_size = 4,
  base_size = 14,
  linewidth = 0.5,
  color = "red",
  color_points = "black",
  parm = NULL,
  n_points = 1000,
  title = NULL,
  add_points = FALSE,
  add_ci = TRUE,
  color_ci = "blue",
  color_pi = "red",
  add_ribbon_ci = FALSE,
  add_ribbon_pi = FALSE,
  color_ribbon_ci = "blue",
  color_ribbon_pi = "red",
  ...
)
```

Arguments

x	An object of class modeler, typically the result of calling modeler().
id	An optional group ID to filter the data for plotting, useful for avoiding overcrowded plots.
type	Numeric value (1-6) to specify the type of plot to generate. Default is 1.
	type = 1 Plot of raw data with fitted curves.
	type = 2 Plot of coefficients with confidence intervals.
	type = 3 Plot of fitted curves, colored by group.
	type = 4 Plot of fitted curves with confidence intervals.
	type = 5 Plot of first derivative with confidence intervals.
	type = 6 Plot of second derivative with confidence intervals.

label_size	Numeric value for the size of labels. Default is 4.
base_size	Numeric value for the base font size in pts. Default is 14.
linewidth	Numeric value specifying size of line geoms. Default is 0.5.
color	Character string specifying the color for the fitted line when type = 1. Default is "red".
color_points	Character string specifying the color for the raw data points when type = 1. Default is "black".
parm	Character vector specifying the parameters to plot for type = 2. If NULL, all parameters are included.
n_points	Numeric value specifying the number of points for interpolation along the x-axis. Default is 2000.
title	Optional character string to add a title to the plot.
add_points	Logical value indicating whether to add raw observations to the plot for type = 3 and 4. Default is FALSE.
add_ci	Logical value indicating whether to add confidence intervals for type = 4, 5, 6. Default is TRUE.
color_ci	Character string specifying the color of the confidence interval when type = 4, 5, 6. Default is "blue".
color_pi	Character string specifying the color of the prediction interval when type = 4. Default is "red".
add_ribbon_ci	Logical value indicating whether to add a ribbon for confidence intervals in type = 4, 5, 6. Default is FALSE.
add_ribbon_pi	Logical value indicating whether to add a ribbon for prediction intervals in type = 4. Default is FALSE.
color_ribbon_ci	Character string specifying the color of the ribbon (ci). Default is "blue".
color_ribbon_pi	Character string specifying the color of the ribbon (pi). Default is "red".
...	Additional graphical parameters for future extensions.

Value

A ggplot object representing the specified plot.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
# Example 1
mod_1 <- dt_potato |>
  modeler(
```

```
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(1:3)
  )
print(mod_1)
plot(mod_1, id = 1:2)
plot(mod_1, id = 1:3, type = 2, label_size = 10)
```

`plot.performance` *Plot an object of class* `performance`

Description

Creates plots for an object of class `performance`

Usage

```
## S3 method for class 'performance'
plot(
  x,
  id = NULL,
  type = 1,
  rescale = FALSE,
  linewidth = 1,
  base_size = 12,
  return_table = FALSE,
  ...
)
```

Arguments

`x`	An object of class `performance`, typically the result of calling `performance()`.
`id`	An optional group ID to filter the data for plotting, useful for avoiding overcrowded plots. This argument is not used when type = 2.
`type`	Numeric value (1-3) to specify the type of plot to generate. Default is 1. `type = 1` Radar plot by uid `type = 2` Radar plot averaging `type = 3` Line plot by model-metric `type = 4` Ranking plot by model
`rescale`	Logical. If `TRUE`, metrics in type 3 plot are (0, 1) rescaled to improve interpretation. Higher values are better models. `FALSE` by default.
`linewidth`	Numeric value specifying size of line geoms.
`base_size`	Numeric value for the base font size in pts. Default is 12
`return_table`	Logical. If `TRUE`, table to generate the plot is returned. `FALSE` by default.
`...`	Additional graphical parameters for future extensions.

Value

A ggplot object representing the specified plot.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
# Model 1
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 90),
    subset = 40
  )
# Model 2
mod_2 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_logistic",
    parameters = c(a = 0.199, t0 = 47.7, k = 100),
    subset = 40
  )
# Model 3
mod_3 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin",
    parameters = c(m = 20, b = 2),
    subset = 40
  )
plot(performance(mod_1, mod_2, mod_3), type = 1)
plot(performance(mod_1, mod_2, mod_3, metrics = c("AICc", "BIC")), type = 3)
```

---

plot_fn *Plot user-defined function*

Description

This function plots a function over a specified interval and annotates the plot with the calculated Area Under the Curve (AUC) and parameter values. The aim of 'plot_fn' is to allow users to play with different starting values in their functions before fitting any models.

Usage

```
plot_fn(
  fn = "fn_lin_plat",
  params = c(t1 = 34.9, t2 = 61.8, k = 100),
  interval = c(0, 100),
  n_points = 1000,
  auc = FALSE,
  x_auc_label = NULL,
  y_auc_label = NULL,
  auc_label_size = 4,
  param_label_size = 4,
  base_size = 12,
  color = "red",
  label_color = "grey30"
)
```

Arguments

fn	A character string representing the name of the function to be plotted. Default is "fn_lin_plat".
params	A named numeric vector of parameters to be passed to the function. Default is c(t1 = 34.9, t2 = 61.8, k = 100).
interval	A numeric vector of length 2 specifying the interval over which the function is to be plotted. Default is c(0, 100).
n_points	An integer specifying the number of points to be used for plotting. Default is 1000.
auc	Print AUC in the plot? Default is FALSE.
x_auc_label	A numeric value specifying the x-coordinate for the AUC label. Default is NULL.
y_auc_label	A numeric value specifying the y-coordinate for the AUC label. Default is NULL.
auc_label_size	A numeric value specifying the size of the AUC label text. Default is 3.
param_label_size	A numeric value specifying the size of the parameter label text. Default is 3.
base_size	A numeric value specifying the base size for the plot's theme. Default is 12.
color	A character string specifying the color for the plot lines and area fill. Default is "red".
label_color	A character string specifying the color for the labels. Default is "grey30".

Value

A ggplot object representing the plot.

Examples

```
# Example usage
plot_fn(
  fn = "fn_lin_plat",
  params = c(t1 = 34.9, t2 = 61.8, k = 100),
  interval = c(0, 100),
  n_points = 1000
)
plot_fn(
  fn = "fn_lin_pl_lin",
  params <- c(t1 = 38.7, t2 = 62, t3 = 90, k = 0.32, beta = -0.01),
  interval = c(0, 100),
  n_points = 1000,
  base_size = 12
)
```

---

predict.modeler *Predict an object of class* modeler

Description

Generate model predictions from an object of class `modeler`. This function allows for flexible prediction types, including point predictions, area under the curve (AUC), first or second order derivatives, and functions of the parameters.

Usage

```
## S3 method for class 'modeler'
predict(
  object,
  x = NULL,
  id = NULL,
  type = c("point", "auc", "fd", "sd"),
  se_interval = c("confidence", "prediction"),
  n_points = 1000,
  formula = NULL,
  metadata = FALSE,
  parallel = FALSE,
  workers = NULL,
  ...
)
```

Arguments

`object`	An object of class `modeler`, typically the result of calling the `modeler()` function.

x	A numeric value or vector specifying the points at which predictions are made. For type = "auc", x must be a vector of length 2 that specifies the interval over which to calculate the AUC.
id	Optional unique identifier to filter predictions by a specific group. Default is NULL.
type	A character string specifying the type of prediction. Default is "point".
	"point" Predicts the value of y for the given x.
	"auc" Calculates the area under the curve (AUC) for the fitted model over the interval specified by x.
	"fd" Returns the first derivative (rate of change) of the model at the given x value(s).
	"sd" Returns the second derivative of the model at the given x value(s).
se_interval	A character string specifying the type of interval for standard error calculation. Options are "confidence" (default) or "prediction". Only works with "point" estimation.
n_points	An integer specifying the number of points used to approximate the area under the curve (AUC) when type = "auc". Default is 1000.
formula	A formula specifying a function of the parameters to be estimated (e.g., ~ b * 500). Default is NULL.
metadata	Logical. If TRUE, metadata is included with the predictions. Default is FALSE.
parallel	Logical. If TRUE the prediction is performed in parallel. Default is FALSE. Use only when a large number of groups are being analyzed and x is a grid of values.
workers	The number of parallel processes to use. parallel::detectCores()
...	Additional parameters for future functionality.

Value

A data.frame containing the predicted values, their associated standard errors, and optionally the metadata.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 2, 45)
```

```
  )
print(mod_1)
# Point Prediction
predict(mod_1, x = 45, type = "point", id = 2)
# AUC Prediction
predict(mod_1, x = c(0, 108), type = "auc", id = 2)
# First Derivative
predict(mod_1, x = 45, type = "fd", id = 2)
# Second Derivative
predict(mod_1, x = 45, type = "sd", id = 2)
# Function of the parameters
predict(mod_1, formula = ~ t2 - t1, id = 2)
```

`print.modeler` *Print an object of class* `modeler`

Description

Prints information about `modeler` function.

Usage

```
## S3 method for class 'modeler'
print(x, ...)
```

Arguments

`x`	An object fitted with the function `modeler()`.
`...`	Options used by the tibble package to format the output. See 'tibble::print()' for more details.

Value

an object inheriting from class `modeler`.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
```

```
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(1:5)
  )
plot(mod_1, id = c(1:4))
print(mod_1)
```

`residuals.modeler` *Extract residuals from a* `modeler` *object*

Description

Extract residuals from a `modeler` object

Usage

```
## S3 method for class 'modeler'
residuals(object, ...)
```

Arguments

`object`	An object of class ‘modeler‘
`...`	Additional parameters for future functionality.

Value

A numeric vector of residuals

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 2, 45)
  )
residuals(mod_1)
```

series_mutate *Transform variables in a data frame*

Description

This function performs transformations on specified columns of a data frame, including truncating maximum values, handling negative values, and adding a zero to the series. It allows for grouping and supports retaining metadata in the output.

Usage

```
series_mutate(
  data,
  x,
  y,
  grp,
  metadata,
  max_as_last = FALSE,
  check_negative = FALSE,
  add_zero = FALSE,
  interval = NULL
)
```

Arguments

data	A data.frame containing the input data for analysis.
x	The name of the column in data representing the independent variable (x points).
y	The name of the column(s) in data containing variables to transform.
grp	Column(s) in data used as grouping variable(s). Defaults to NULL (optional).
metadata	Names of columns to retain in the output. Defaults to NULL (optional).
max_as_last	Logical. If TRUE, appends the maximum value after reaching the maximum. Default is FALSE.
check_negative	Logical. If TRUE, converts negative values in the data to zero. Default is FALSE.
add_zero	Logical. If TRUE, adds a zero value to the series at the start. Default is FALSE.
interval	A numeric vector of length 2 (start and end) specifying the range to filter the data. Defaults to NULL.

Value

A transformed data.frame with the specified modifications applied.

Examples

```
data(dt_potato)
new_data <- series_mutate(
  data = dt_potato,
  x = DAP,
  y = GLI,
  grp = gid,
  max_as_last = TRUE,
  check_negative = TRUE
)
```

`subset.modeler` *Subset an object of class* `modeler`

Description

Subset an object of class `modeler`

Usage

```
## S3 method for class 'modeler'
subset(x, id = NULL, ...)
```

Arguments

`x`	An object of class `modeler`, typically the result of calling `modeler()`.
`id`	Unique identifier to filter a `modeler` object by a specific group. Default is `NULL`.
`...`	Additional parameters for future functionality.

Value

A `modeler` object.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_logistic",
    parameters = c(a = 0.199, t0 = 47.7, k = 100),
```

```
    subset = 1:2
  )
print(mod)
mod_new <- subset(mod, id = 2)
print(mod_new)
```

`update.modeler` *Update a* `modeler` *object*

Description

It creates a new fitted object using the parameter values from the current model as initial values. It can also be used to perform a few additional iterations of a model that has not converged.

Usage

```
## S3 method for class 'modeler'
update(object, method = NULL, track = TRUE, eps = 1e-06, ...)
```

Arguments

`object`	An object of class `modeler`.
`method`	A character vector specifying optimization methods. Check available methods using `list_methods()`. Defaults to the ones in `object`.
`track`	Logical. If `TRUE`, the function compares the SSE before and after the update and reports how many groups improved. Useful for evaluating whether the refit led to better convergence.
`eps`	Numeric. The minimum change in SSE required to consider a fit improved. Defaults to `1e-6`. Smaller values may include numerical noise as improvements.
`...`	Additional parameters for future functionality.

Value

An object of class `modeler`, which is a list containing the following elements:

`param` Data frame containing optimized parameters and related information.

`dt` Data frame with input data, fitted values, and residuals.

`metrics` Metrics and summary of the models.

`execution` Total execution time for the analysis.

`response` Name of the response variable analyzed.

`keep` Metadata retained based on the `keep` argument.

`fun` Name of the curve-fitting function used.

`parallel` List containing parallel execution details (if applicable).

`fit` List of fitted models for each group.

Examples

```
library(flexFitR)
data(dt_potato)
mo_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = GLI,
    grp = Plot,
    fn = "fn_lin_pl_lin",
    parameters = c(t1 = 10, t2 = 62, t3 = 90, k = 0.32, beta = -0.01),
    subset = 195
  )
plot(mo_1)
mo_2 <- update(mo_1)
plot(mo_2)
```

---

vcov.modeler *Variance-Covariance matrix for an object of class* modeler

---

Description

Extract the variance-covariance matrix for the parameter estimates from an object of class `modeler`.

Usage

```
## S3 method for class 'modeler'
vcov(object, id = NULL, ...)
```

Arguments

`object`	An object of class `modeler`, typically the result of calling the `modeler()` function.
`id`	An optional unique identifier to filter by a specific group. Default is `NULL`.
`...`	Additional parameters for future functionality.

Value

A list of matrices, where each matrix represents the variance-covariance matrix of the estimated parameters for each group or fit.

Author(s)

Johan Aparicio [aut]

Examples

```
library(flexFitR)
data(dt_potato)
mod_1 <- dt_potato |>
  modeler(
    x = DAP,
    y = Canopy,
    grp = Plot,
    fn = "fn_lin_plat",
    parameters = c(t1 = 45, t2 = 80, k = 0.9),
    subset = c(15, 2, 45)
  )
print(mod_1)
vcov(mod_1)
```

Index