

Package ‘MultiSpline’

October 11, 2026

Type Package

Title Spline-Based Nonlinear Modeling for Multilevel and Longitudinal Data

Version 0.3.0

Description Provides a unified framework for fitting, predicting, and interpreting nonlinear relationships in single-level, multilevel, and longitudinal regression models. Flexible functional forms are supported using natural cubic splines ('splines'), B-splines ('splines'), and GAM smooths ('mgcv'). Supports two-way and nested clustering via 'lme4', automatic knot selection by AIC or BIC, multilevel R-squared decomposition (marginal and conditional R-squared of Nakagawa and Schielzeth (2013) <[doi:10.1111/j.2041-210x.2012.00261.x](https://doi.org/10.1111/j.2041-210x.2012.00261.x)>, extended to random slopes as in Johnson (2014) <[doi:10.1111/2041-210X.12225](https://doi.org/10.1111/2041-210X.12225)>, with level-specific variance partitioning), a postestimation suite returning first and second derivatives with confidence bands, turning points and inflection regions, and a model comparison workflow contrasting linear, polynomial, and spline fits by AIC, BIC, and likelihood-ratio tests. Cluster heterogeneity in nonlinear effects is supported via random-slope spline terms. Any 'lme4' random-effects structure can be supplied, and population-averaged (marginal) curves are available for generalized linear mixed models.

Depends R (>= 4.2.0)

Imports stats, lme4, mgcv, dplyr, ggplot2, rlang, splines

Suggests lmerTest, knitr, rmarkdown, reformulas, numDeriv, patchwork, testthat (>= 3.0.0)

Config/testthat/edition 3

License GPL-3

Encoding UTF-8

Language en-US

URL <https://github.com/subirhait/MultiSpline>

BugReports <https://github.com/subirhait/MultiSpline/issues>

NeedsCompilation no

Author Subir Hait [aut, cre] (ORCID: <<https://orcid.org/0009-0004-9871-9677>>)
Maintainer Subir Hait <haitsubi@msu.edu>
Config/roxygen2/version 8.0.0
Repository CRAN
Date/Publication 2026-10-11 02:30:02 UTC

Contents

nl_compare	2
nl_derivatives	4
nl_fit	5
nl_het	7
nl_icc	9
nl_knots	10
nl_plot	11
nl_predict	13
nl_r2	15
nl_summary	16
nl_turning_points	17
nl_wald	18
print.nl_fit	19
print.summary_nl_fit	19
summary.nl_fit	20
Index	21

nl_compare	<i>Built-in model comparison workflow</i>
------------	---

Description

Compares a nonlinear spline model against simpler alternatives—a linear model and optional polynomial terms—to help researchers justify the spline approach over simpler specifications.

For each model, the function reports:

- AIC and BIC
- Log-likelihood (and likelihood-ratio test versus the linear model where available)
- Number of parameters (df)
- Residual variance / deviance

Mixed models are compared on their maximum-likelihood fits: the spline model is refitted by ML if it was estimated by REML, and the comparison models are fitted by ML. REML likelihoods cannot be compared across models with different fixed effects. The comparison models use the same random-effects structure as the spline model (random intercepts only if the spline model has random spline slopes).

Usage

```
nl_compare(
  object,
  polynomial_degrees = c(2L, 3L),
  digits = 3L,
  return_models = FALSE
)
```

Arguments

<code>object</code>	An <code>nl_fit</code> object (the spline model to compare against).
<code>polynomial_degrees</code>	Integer vector of polynomial degrees to include as additional comparators. Default <code>c(2L, 3L)</code> . Set to <code>integer(0)</code> to skip.
<code>digits</code>	Integer; decimal places for the output table. Default 3.
<code>return_models</code>	Logical; if TRUE, also returns the fitted comparison model objects. Default FALSE.

Value

A list (invisibly) of class "nl_compare" with:

`table` A data frame with the comparison statistics.

`models` Named list of fitted models (when `return_models = TRUE`).

`best` Name of the model with the lowest AIC.

The table is pretty-printed automatically.

See Also

[nl_fit](#), [nl_knots](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
nl_compare(fit, polynomial_degrees = 2)
```

nl_derivatives	<i>First and second derivatives of the fitted curve</i>
----------------	---

Description

Computes the slope (first derivative) and curvature (second derivative) of the fitted curve with respect to x , with delta-method standard errors and pointwise confidence intervals.

Usage

```
nl_derivatives(
  object,
  x_seq = NULL,
  time_levels = NULL,
  controls_fixed = NULL,
  level = 0.95,
  type = c("conditional", "marginal"),
  scale = c("response", "link"),
  h = NULL,
  ...
)
```

Arguments

object	An <code>nl_fit</code> object. (Passing a data frame from <code>nl_predict()</code> , as in version 0.2.0, still returns derivative estimates but no intervals, with a warning.)
x_seq, time_levels, controls_fixed	Grid specification, as in nl_predict .
level	Confidence level. Default 0.95.
type, scale	As in nl_predict : derivatives of the conditional or marginal curve, on the response or link scale.
h	Step for the central differences. Default NULL uses 10^{-4} times the range of x . Because the basis is evaluated exactly, the result is insensitive to h .
...	Ignored; absorbs arguments of the 0.2.0 interface.

Details

Derivatives are taken of the model itself, not of a table of predictions. For each grid value the spline basis is evaluated at $x \pm h$ with the knots of the fitted model, giving the derivative as a linear combination $d^\top \beta$ (identity link) or, for other links, via the chain rule. Its variance is $g^\top V g$, where g is the gradient with respect to the fixed effects and V their covariance matrix. This accounts for the strong correlation between neighbouring points on the curve, which a calculation from a prediction table cannot do.

Intervals are pointwise, not simultaneous: they describe uncertainty at each x separately.

Value

A data frame with the grid columns, fit, d1, d1_se, d1_lwr, d1_upr, d2, d2_se, d2_lwr, and d2_upr.

See Also

[nl_turning_points](#), [nl_plot](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
deriv <- nl_derivatives(fit, x_seq = 0:9)
deriv[, c("Days", "d1", "d1_lwr", "d1_upr")]
```

nl_fit

Fit a nonlinear (spline or GAM) single-level or multilevel model

Description

Fits a nonlinear regression model for an outcome *y* with a focal predictor *x*, modeled using natural cubic splines ("ns"), B-splines ("bs"), or GAM smooths ("gam").

Mixed models are fitted with lme4. The level structure can be given simply through *cluster* (one grouping variable, or two that are nested or crossed) or in full through *random*, which accepts any lme4 random-effects specification.

Usage

```
nl_fit(
  data,
  y,
  x,
  time = NULL,
  cluster = NULL,
  nested = FALSE,
  controls = NULL,
  random = NULL,
  method = c("ns", "bs", "gam"),
  df = 4,
  df_range = 2:8,
  df_criterion = c("AIC", "BIC"),
  k = 5,
  bs_degree = 3,
  random_slope = FALSE,
  family = stats::gaussian(),
  REML = TRUE,
```

```
    ...
  )
```

Arguments

<code>data</code>	A data frame (often long format for longitudinal data).
<code>y</code>	Outcome variable name (string).
<code>x</code>	Focal nonlinear predictor name (string). Must be numeric.
<code>time</code>	Optional time variable name (string).
<code>cluster</code>	Optional character vector of grouping variable name(s) for random effects, e.g. <code>NULL</code> , <code>"id"</code> , or <code>c("id", "school")</code> for two-way clustering.
<code>nested</code>	Logical; only used when <code>length(cluster) == 2</code> . If <code>TRUE</code> , the first grouping variable is nested within the second: <code>cluster = c("student", "school")</code> gives <code>(1 school/student)</code> . (Version 0.2.0 built the reverse, <code>(1 student/school)</code> , which omitted the school effect.) If <code>FALSE</code> (default), uses cross-classified <code>(1 g1) + (1 g2)</code> . For anything more complex use <code>random</code> .
<code>controls</code>	Optional character vector of additional fixed-effect terms. Plain variable names and expressions are both accepted, e.g. <code>c("age", "factor(race)", "I(age^2)")</code> . Factor and character variables are expanded to indicator columns automatically.
<code>random</code>	Optional character string giving the full random-effects part of an <code>lme4</code> formula, e.g. <code>"(1 school/student)"</code> or <code>"(1 idcode) + (1 indcode)"</code> . When supplied it overrides <code>cluster</code> , <code>nested</code> , and <code>random_slope</code> , so any level structure that <code>lme4</code> accepts can be used. Grouping variables are read from the string.
<code>method</code>	Spline basis to use: <code>"ns"</code> (natural cubic spline, default), <code>"bs"</code> (B-spline), or <code>"gam"</code> (GAM smooth via <code>mgcv::gam()</code>). Multilevel fits require <code>"ns"</code> or <code>"bs"</code> .
<code>df</code>	Degrees of freedom for the spline basis. Supply a single integer greater than or equal to 1, or the string <code>"auto"</code> to trigger automatic selection by information criterion. Default 4.
<code>df_range</code>	Integer vector of candidate df values evaluated when <code>df = "auto"</code> . Default 2:8.
<code>df_criterion</code>	Information criterion used for automatic df selection: <code>"AIC"</code> (default) or <code>"BIC"</code> .
<code>k</code>	Basis dimension for <code>mgcv::s()</code> when <code>method = "gam"</code> . Default 5.
<code>bs_degree</code>	Polynomial degree for splines: <code>bs()</code> when <code>method = "bs"</code> . Default 3 (cubic).
<code>random_slope</code>	Logical; if <code>TRUE</code> and <code>cluster</code> is supplied, fits random spline slopes in addition to random intercepts. Currently implemented only for single-level clustering. Default <code>FALSE</code> .
<code>family</code>	A family object such as <code>stats::gaussian()</code> (default) or <code>stats::binomial()</code> . For multilevel fits, <code>gaussian()</code> uses <code>lme4::lmer()</code> and all other families use <code>lme4::glmer()</code> .
<code>REML</code>	Logical; for Gaussian mixed models, fit the final model by REML (default <code>TRUE</code>). Automatic df selection and all model comparisons always use maximum likelihood, because REML likelihoods are not comparable across different fixed effects.
<code>...</code>	Additional arguments passed to the underlying fitting function (<code>stats::lm()</code> , <code>mgcv::gam()</code> , <code>lme4::lmer()</code> , or <code>lme4::glmer()</code>).

Value

An object of class "nl_fit" (a named list). It contains the fitted model object, the method, variable names (y, x, time, cluster, controls), spline settings (df, df_selected, df_search, k, bs_degree), flags (nested, random_slope), the family, the model formula, the call, and meta-data used for prediction (x_info, levels_info, control_defaults).

Spline basis

method = "ns" uses `splines::ns()`: a natural cubic spline (cubic between knots, linear beyond the boundary knots). Interior knots are placed at equally spaced quantiles of x; the boundary knots are the range of x. df is the number of basis columns (interior knots = df - 1). With the same knots it spans the same functions as Stata's restricted cubic splines (`mkspline`, `cubic`, `makespline rcs`); the defaults differ only in knot placement (Stata puts the outer knots at percentiles, `ns()` at the range). The basis computed on the estimation data is stored in the model and reused for every prediction and derivative.

See Also

[nl_predict](#), [nl_derivatives](#), [nl_compare](#), [nl_r2](#), [nl_knots](#)

Examples

```
# Single-level natural spline
fit1 <- nl_fit(mtcars, y = "mpg", x = "hp", df = 3)
fit1

# Repeated measures: days nested in subjects
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)

# Any lme4 random-effects structure via `random`
fit2 <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
              random = "(1 | Subject)", df = 3)
```

nl_het

Cluster heterogeneity in nonlinear effects

Description

Quantifies and visualises how much the nonlinear relationship between x and y varies across clusters, using a model with random spline slopes.

The function:

1. Refits the model with random spline slopes (if not already fitted with `random_slope = TRUE`).
2. Extracts cluster-specific predicted curves (BLUPs).
3. Returns the standard deviation of the cluster-specific curves at each x value, and a plot of the cluster-specific trajectories.

4. Performs a likelihood-ratio test comparing the random-slope model against a random-intercept-only model to assess whether heterogeneity is statistically significant. Because the null hypothesis puts variances on the boundary of the parameter space, the reported p-value is conservative.

Usage

```
nl_het(
  object,
  n_clusters_plot = 30L,
  x_seq = NULL,
  level = 0.95,
  plot = TRUE,
  seed = 42L
)
```

Arguments

<code>object</code>	An <code>nl_fit</code> object fitted with a single cluster variable (and either <code>random_slope = TRUE</code> or <code>FALSE</code> ; in the latter case the model is refit internally).
<code>n_clusters_plot</code>	Maximum number of cluster curves to display in the trajectory plot. Default 30L. Use Inf for all clusters.
<code>x_seq</code>	Optional numeric vector of x values for prediction.
<code>level</code>	Confidence level for the population-mean ribbon. Default 0.95.
<code>plot</code>	Logical; print the trajectory plot. Default TRUE.
<code>seed</code>	Integer seed for reproducibility when sub-sampling clusters. Default 42L.

Value

A list (invisibly) with:

`trajectory_df` Long data frame of cluster-specific predicted values.

`between_cluster_sd` Data frame: the standard deviation of the cluster-specific predicted values (response scale) across the displayed clusters at each x value. Replaces the mislabelled `slope_variance` of version 0.2.0.

`lrt` LRT comparing random-slope vs random-intercept model (NULL if `random_slope = TRUE` was supplied).

`plot` A ggplot object.

See Also

[nl_fit](#), [nl_derivatives](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
het <- nl_het(fit, plot = FALSE)
het$lrt
```

nl_icc

*Intraclass correlation coefficients for a multilevel nl_fit model***Description**

Extracts variance components from a multilevel `nl_fit` object and computes intraclass correlation coefficients (ICCs) for each grouping level plus the residual.

The ICC for grouping factor g is defined as:

$$ICC_g = \frac{\sigma_g^2}{\sum_j \sigma_j^2 + \sigma_\epsilon^2}$$

For Gaussian models σ_ϵ^2 is the residual variance. For binary models it is the latent-scale variance ($\pi^2/3$ for logit, 1 for probit, $\pi^2/6$ for complementary log-log), which matches Stata's `estat icc`. With nested factors each value is the share of that level alone; the correlation between two units that share a lower-level group is the sum of the shares of that group and every group above it. With random slopes the ICC depends on x ; the function then averages each component over the data and warns.

Usage

```
nl_icc(object, include_residual = TRUE)
```

Arguments

object An `nl_fit` object returned by `nl_fit` that was fitted with one or more cluster variables (i.e., a multilevel model fitted via `lme4::lmer()` or `lme4::glmer()`).

include_residual Logical; if TRUE (default), includes the residual variance component `ICC_resid` in the output so that all values sum to 1.

Value

A named numeric vector of ICCs, one per grouping factor (named `ICC_<groupname>`) plus `Residual` for the residual variance (when `include_residual = TRUE`). All values sum to 1.

See Also

[nl_fit](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
nl_icc(fit)
```

nl_knots

Automatic knot / degrees-of-freedom selection for spline models

Description

Performs a grid search over candidate degrees of freedom (df) for a natural cubic spline ("ns") or B-spline ("bs") model and selects the best value by an information criterion (AIC or BIC). A diagnostic plot of the criterion against df is returned. Mixed models are fitted by maximum likelihood for this search, because REML criteria are not comparable across different fixed effects.

This function is called internally by `nl_fit` when `df = "auto"`, but it can also be called directly for exploration before fitting the final model.

Usage

```
nl_knots(
  data,
  y,
  x,
  time = NULL,
  cluster = NULL,
  nested = FALSE,
  controls = NULL,
  random = NULL,
  method = c("ns", "bs"),
  df_range = 2:10,
  criterion = c("AIC", "BIC"),
  family = stats::gaussian(),
  plot = TRUE,
  ...
)
```

Arguments

<code>data</code>	A data frame.
<code>y</code>	Outcome variable name (string).
<code>x</code>	Focal predictor name (string).
<code>time</code>	Optional time variable name.
<code>cluster</code>	Optional character vector of cluster variable names.
<code>nested</code>	Logical; nested clustering. Default FALSE.

controls	Optional character vector of control variable names.
random	Optional full random-effects string, as in nl_fit .
method	Either "ns" (default) or "bs".
df_range	Integer vector of candidate df values. Default 2:10.
criterion	Either "AIC" (default) or "BIC".
family	A family object. Default stats::gaussian().
plot	Logical; if TRUE, prints a diagnostic plot of criterion vs df. Default TRUE.
...	Additional arguments passed to the underlying fitter.

Value

A list with:

best_df The df value with the lowest criterion value.
 search_table Data frame with columns df and criterion.
 criterion The criterion used ("AIC" or "BIC").
 plot A ggplot object (if plot = TRUE).

See Also

[nl_fit](#), [nl_compare](#)

Examples

```
nl_knots(mtcars, y = "mpg", x = "hp", df_range = 2:5, plot = FALSE)
```

nl_plot

Plot predictions and derivatives from nl_fit models

Description

Visualises results from [nl_predict](#) or [nl_derivatives](#). The type argument selects the plot:

- "trajectory": predicted outcome vs x with optional confidence ribbon (v1 behaviour, now default).
- "slope": first derivative (marginal effect) vs x.
- "curvature": second derivative vs x.
- "combo": trajectory + slope panels side by side.

Usage

```
nl_plot(
  pred_df = NULL,
  deriv_df = NULL,
  x = NULL,
  time = NULL,
  type = c("trajectory", "slope", "curvature", "combo"),
  show_ci = TRUE,
  ci_level = 0.95,
  show_turning_points = FALSE,
  turning_points = NULL,
  zero_line = TRUE,
  y_lab = NULL,
  x_lab = NULL,
  title = NULL,
  legend_title = NULL
)
```

Arguments

<code>pred_df</code>	A data frame from <code>nl_predict</code> (for type = "trajectory" or "combo").
<code>deriv_df</code>	A data frame from <code>nl_derivatives</code> (for type = "slope", "curvature", or "combo").
<code>x</code>	Character; name of the focal predictor column. Taken from the <code>nl_predict()</code> / <code>nl_derivatives()</code> output by default.
<code>time</code>	Optional character; name of the time column. Auto-detected if NULL.
<code>type</code>	Plot type: "trajectory" (default), "slope", "curvature", or "combo".
<code>show_ci</code>	Logical; add confidence ribbons. Default TRUE.
<code>ci_level</code>	Numeric; confidence level when deriving CI from <code>se_fit</code> . Default 0.95.
<code>show_turning_points</code>	Logical; overlay turning-point markers on the trajectory plot. Default FALSE.
<code>turning_points</code>	Data frame from <code>nl_turning_points</code> (only used when <code>show_turning_points</code> = TRUE).
<code>zero_line</code>	Logical; for slope / curvature plots, draw a dashed horizontal line at zero. Default TRUE.
<code>y_lab</code>	Y-axis label.
<code>x_lab</code>	X-axis label. Defaults to <code>x</code> .
<code>title</code>	Optional plot title.
<code>legend_title</code>	Optional legend title.

Value

A ggplot object. For "combo", a single combined figure if **patchwork** is installed, otherwise a list of two plots.

See Also

[nl_predict](#), [nl_derivatives](#), [nl_turning_points](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
pred <- nl_predict(fit)
deriv <- nl_derivatives(fit)
nl_plot(pred)
nl_plot(deriv_df = deriv, type = "slope")
```

nl_predict

Generate predictions from an nl_fit model

Description

Evaluates the fitted curve over a grid of the focal predictor x (and over time levels, if the model has a time interaction), holding control variables at fixed values.

Usage

```
nl_predict(
  object,
  x_seq = NULL,
  time_levels = NULL,
  controls_fixed = NULL,
  se = TRUE,
  level = 0.95,
  type = c("conditional", "marginal"),
  scale = c("response", "link"),
  re_form = NA,
  glmer_ci = c("delta", "boot"),
  n_boot = 500L,
  ...
)
```

Arguments

object	An nl_fit object.
x_seq	Optional numeric vector of x values.
time_levels	Optional vector of time levels.
controls_fixed	Optional named list of fixed control values.
se	Logical; include standard errors and intervals. Default TRUE.
level	Confidence level. Default 0.95.

type	"conditional" (default) or "marginal"; see <i>What is predicted</i> .
scale	"response" (default) or "link".
re_form	Retained for compatibility. Only NA (random effects set to zero) is supported; use nl_het for cluster-specific curves.
glmer_ci	For non-Gaussian mixed models: "delta" (default) or "boot" (parametric bootstrap via <code>lme4::bootMer()</code>).
n_boot	Number of bootstrap replicates when <code>glmer_ci = "boot"</code> .
...	Reserved for future use.

Value

A data frame (tibble) with the grid columns, `fit`, `se.fit`, `lwr`, and `upr`. Attributes `type` and `scale` record what was predicted.

What is predicted

- The grid has 200 evenly spaced values of `x` between its 1st and 99th sample percentiles unless `x_seq` is supplied. Each row is an exact evaluation of the fitted model; nothing is interpolated. Plots join these points with straight line segments.
- Controls are held at their sample mean (numeric) or first level (factor / character) unless set in `controls_fixed`.
- `type = "conditional"` (default) sets all random effects to zero: the curve for a cluster whose random effects are at their mean. For Gaussian models this equals the population-averaged curve.
- `type = "marginal"` averages the response-scale mean over the estimated random-effect distribution (Gauss-Hermite quadrature). For non-identity links (e.g. logistic models) this population-averaged curve is flatter than the conditional curve. Available for random intercepts and for the random spline slopes created by `nl_fit(random_slope = TRUE)`.
- `scale = "response"` (default) returns means on the outcome scale (probabilities for binary outcomes); `scale = "link"` returns the linear predictor.
- Standard errors use the delta method with the full covariance matrix of the fixed effects, treating variance components as known. Intervals are formed on the link scale and transformed, so they stay inside the range of the outcome.

See Also

[nl_fit](#), [nl_plot](#), [nl_derivatives](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
pred <- nl_predict(fit)
head(pred)
attr(pred, "type")
```

nl_r2

*Multilevel R-squared decomposition for nl_fit models***Description**

Computes a suite of R-squared statistics for models fitted by `nl_fit`. For single-level models the standard R-squared and adjusted R-squared are returned. For multilevel models (`lmerMod` / `glmerMod`) two quantities are reported: the Nakagawa-Schielezeth marginal R-squared (variance explained by fixed effects only) and the conditional R-squared (fixed plus all random effects), together with a level-specific variance partition table analogous to the `r2_mlm` / Raudenbush-Bryk approach.

Usage

```
nl_r2(object, digits = 4L)
```

Arguments

`object` An `nl_fit` object returned by `nl_fit`.
`digits` Integer; decimal places for display. Default 4.

Details

Marginal and conditional R-squared for LMMs follow the Nakagawa and Schielzeth (2013) formulae extended to multiple random effects by Nakagawa, Johnson and Schielzeth (2017). The fixed-effects variance σ_f^2 is computed as the variance of the linear predictor from fixed effects only ($\hat{\mu} = X\hat{\beta}$).

With random slopes, each random-effects term contributes the variance of its contribution to the linear predictor averaged over the estimation sample, $\overline{z_i^\top \Sigma z_i}$ (Johnson 2014). For GLMMs the distribution-specific variance is $\pi^2/3$ (logit), 1 (probit), $\pi^2/6$ (complementary log-log), or the log-normal approximation for Poisson models (Nakagawa et al. 2017).

The level-specific variance partition (`r2_mlm`-style) decomposes the total modelled variance ($\sigma_f^2 + \sum \sigma_j^2 + \sigma_e^2$) to show how much each source contributes, printed as a breakdown table.

Value

A list of class "nl_r2" returned invisibly and pretty-printed automatically. It contains `type` (one of "OLS", "GAM", "LMM", or "GLMM"), `r2` (a named numeric vector: `R2` and `R2_adj` for OLS; `R2_dev` for GAM; `R2m` and `R2c` for LMM/GLMM), and `variance_partition` (a data frame with columns `component`, `variance`, and `proportion` for multilevel models, or `NULL` for single-level models).

References

Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R-squared from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, 4(2), 133–142.

Nakagawa, S., Johnson, P. C. D., & Schielzeth, H. (2017). The coefficient of determination R-squared and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. *Journal of the Royal Society Interface*, 14(134), 20170213.

Johnson, P. C. D. (2014). Extension of Nakagawa & Schielzeth's R2GLMM to random slopes models. *Methods in Ecology and Evolution*, 5(9), 944–946.

Rights, J. D., & Sterba, S. K. (2019). Quantifying explained variance in multilevel models: An integrative framework for defining R-squared measures. *Psychological Methods*, 24(3), 309–338.

See Also

[nl_fit](#), [nl_icc](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
nl_r2(fit)
```

nl_summary	<i>Tidy coefficient table for an nl_fit model</i>
------------	---

Description

Produces a tidy coefficient table for a model fitted by [nl_fit](#). For linear mixed models (`lmerMod`), optional p-values and denominator degrees of freedom are obtained via **lmerTest** (Satterthwaite method). For GLMMs (`glmerMod`), z-tests are reported from `summary()`.

Usage

```
nl_summary(
  object,
  digits = 3,
  pvals = TRUE,
  df_method = c("satterthwaite", "none")
)
```

Arguments

<code>object</code>	An <code>nl_fit</code> object returned by nl_fit .
<code>digits</code>	Integer; number of decimal places for rounding. If <code>NULL</code> , no rounding is applied. Default 3.
<code>pvals</code>	Logical; if <code>TRUE</code> , attempts to include p-values. Default <code>TRUE</code> .
<code>df_method</code>	For <code>lmerMod</code> with <code>pvals = TRUE</code> : "satterthwaite" (requires lmerTest) or "none". Default "satterthwaite".

Value

A data frame (or tibble) with columns: Term, Estimate, Std.Error, df (if available), statistic, and p.value (if available).

See Also

[nl_fit](#), [summary.nl_fit](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
nl_summary(fit)
```

nl_turning_points	<i>Identify turning points and inflection regions</i>
-------------------	---

Description

From the derivative table returned by [nl_derivatives](#), finds turning points (values of x where the first derivative crosses zero), inflection regions (intervals where the second derivative changes sign), and slope regions (contiguous stretches where the curve is increasing, decreasing, or flat).

Usage

```
nl_turning_points(
  deriv_df,
  x = attr(deriv_df, "x"),
  time = attr(deriv_df, "time"),
  tol = 1e-06
)
```

Arguments

deriv_df	A data frame returned by nl_derivatives .
x	Character; name of the focal predictor column. Taken from the nl_derivatives() output by default.
time	Optional character; name of the time column. Turning points are identified separately within each time level when supplied.
tol	Numeric tolerance for near-zero detection of d1. Default 1e-6.

Value

A named list with three elements. `turning_points` is a data frame with columns `x`, `time` (if applicable), `fit`, and `type` (one of "maximum", "minimum", or "saddle"). `inflection_regions` is a data frame with columns `x_start`, `x_end`, `time`, and `direction` ("concave_to_convex" or "convex_to_concave"). `slope_regions` is a data frame with columns `x_start`, `x_end`, `direction`, and `time`.

See Also

[nl_derivatives](#), [nl_plot](#)

Examples

```
fit <- nl_fit(mtcars, y = "mpg", x = "hp", df = 3)
tp <- nl_turning_points(nl_derivatives(fit))
tp$slope_regions
```

nl_wald

Joint Wald test of the focal predictor

Description

Tests whether x is associated with the outcome at all, i.e. whether every fixed-effect coefficient of a term involving x (all spline basis columns, and their interactions with time) is zero. The individual spline coefficients reported by [nl_summary](#) have no separate interpretation; this joint test is the meaningful one. To test nonlinearity (spline versus a straight line) use [nl_compare](#).

Usage

```
nl_wald(object)
```

Arguments

`object` An `nl_fit` object fitted with `method = "ns"` or `"bs"`.

Value

A one-row data frame with the Wald chi-square statistic, its degrees of freedom, and the p-value. Variance components are treated as known, so for small numbers of clusters the test is approximate.

See Also

[nl_compare](#), [nl_summary](#)

Examples

```
fit <- nl_fit(lme4::sleepstudy, y = "Reaction", x = "Days",
             cluster = "Subject", df = 3)
nl_wald(fit)
```

print.nl_fit	<i>Print method for nl_fit objects</i>
--------------	--

Description

Compact console display for objects returned by [nl_fit](#). Shows key metadata (method, outcome, predictor, time, clustering, family, and controls) and the fitted model formula.

Usage

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

Arguments

x	An object of class nl_fit.
...	Further arguments (currently ignored).

Value

x invisibly.

print.summary_nl_fit	<i>Print method for summary_nl_fit objects</i>
----------------------	--

Description

Prints a summary_nl_fit object returned by [summary.nl_fit](#).

Usage

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

Arguments

x	A summary_nl_fit object.
...	Further arguments passed to print.

Value

x invisibly.

summary.nl_fit	<i>Summary method for nl_fit objects</i>
----------------	--

Description

Produces a tidy coefficient table via [nl_summary](#) and wraps it in a `summary_nl_fit` object for pretty printing.

Usage

```
## S3 method for class 'nl_fit'
summary(
  object,
  digits = 3,
  pvals = TRUE,
  df_method = c("satterthwaite", "none"),
  ...
)
```

Arguments

<code>object</code>	An <code>nl_fit</code> object returned by nl_fit .
<code>digits</code>	Number of decimal places for rounding. Default 3.
<code>pvals</code>	Logical; if TRUE, attempts to include p-values. Default TRUE.
<code>df_method</code>	For <code>lmerMod</code> : "satterthwaite" (requires lmerTest) or "none". Default "satterthwaite".
<code>...</code>	Further arguments passed to nl_summary .

Value

An object of class `summary_nl_fit` containing call, formula, method, and table.

Index

`nl_compare`, [2](#), [7](#), [11](#), [18](#)
`nl_derivatives`, [4](#), [7](#), [8](#), [11–14](#), [17](#), [18](#)
`nl_fit`, [3](#), [5](#), [8–11](#), [14–17](#), [19](#), [20](#)
`nl_het`, [7](#), [14](#)
`nl_icc`, [9](#), [16](#)
`nl_knots`, [3](#), [7](#), [10](#)
`nl_plot`, [5](#), [11](#), [14](#), [18](#)
`nl_predict`, [4](#), [7](#), [11–13](#), [13](#)
`nl_r2`, [7](#), [15](#)
`nl_summary`, [16](#), [18](#), [20](#)
`nl_turning_points`, [5](#), [12](#), [13](#), [17](#)
`nl_wald`, [18](#)

`print.nl_fit`, [19](#)
`print.summary_nl_fit`, [19](#)

`summary.nl_fit`, [17](#), [19](#), [20](#)