Package 'MultiSpline'

March 16, 2026

Type Package

Title Spline-Based Nonlinear Modeling for Multilevel and Longitudinal Data

Version 0.1.1

Description Provides tools for fitting, predicting, and visualizing nonlinear relationships in single-level, multilevel, and longitudinal regression models. Nonlinear functional forms are represented using natural cubic splines from 'splines' and smooth terms from 'mgcv'. The package offers a unified interface for specifying nonlinear effects, interactions with time variables, random-intercept clustering structures, and additional linear covariates. Utilities are included to generate prediction grids and produce effect plots, facilitating interpretation and visualization of nonlinear relationships in applied regression workflows. The implementation builds on established methods for spline-based regression and mixed-effects modeling (Hastie and Tibshirani, 1990 <doi:10.1201/9780203738535>; Bates et al., 2015 <doi:10.18637/jss.v067.i01>; Wood, 2017 <doi:10.1201/9781315370279>). Applications include hierarchical and longitudinal data structures common in education, health, and social science research.

Depends R (>= 4.2.0)

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

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

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Subir Hait [aut, cre]

Maintainer Subir Hait <haitsubi@msu.edu>

Repository CRAN

Date/Publication 2026-03-16 16:40:33 UTC

Contents

nl_fit 2
nl_icc 5
nl_plot 6
nl_predict 8
nl_summary 10
print.nl_fit 11
print.summary_nl_fit 12
summary.nl_fit 13

Index **14**

`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 either by a natural cubic spline (`splines::ns()`) or a GAM smooth (`mgcv::s()`).

Optionally includes a time variable `time`. For spline fits (`method = "ns"`), multilevel random-intercept structure can be added via one or more grouping variables in `cluster`.

Models fitted:

- Single-level spline: `stats::lm()`
- Single-level GAM: `mgcv::gam()`
- Multilevel spline: `lme4::lmer()` (Gaussian) or `lme4::glmer()` (non-Gaussian)

Usage

```
nl_fit(
  data,
  y,
  x,
  time = NULL,
  cluster = NULL,
  controls = NULL,
  method = c("ns", "gam"),
  df = 4,
  k = 5,
  family = stats::gaussian(),
  ...
)
```

Arguments

data	A data frame (often long format for longitudinal data).
y	Outcome variable name (string).
x	Focal nonlinear predictor name (string). Must be numeric.
time	Optional time variable name (string). If provided, a spline-by-time interaction is included for method = "ns"; for method = "gam", a factor time uses s(x, by = time) (group-specific smooths), while a numeric time uses s(x, k = k) + time + ti(x, time, k = k) (tensor interaction).
cluster	Optional character vector of grouping variable name(s) for random intercepts, e.g., NULL, "id", or c("id", "schid").
controls	Optional character vector of additional covariate names to include linearly.
method	Either "ns" (natural spline) or "gam" (GAM smooth). Multilevel fits are currently supported only for "ns".
df	Degrees of freedom for splines::ns() when method = "ns". Must be a single numeric value >= 1. Default is 4.
k	Basis dimension for mgcv::s() and mgcv::ti() when method = "gam". Must be a single numeric value >= 3. Default is 5.
family	A model family object, e.g., stats::gaussian() or stats::binomial(). For multilevel fits, gaussian() uses lme4::lmer() and non-Gaussian families use lme4::glmer().
...	Additional arguments passed to the underlying fitting function (stats::lm(), mgcv::gam(), lme4::lmer(), or lme4::glmer()).

Value

An object of class "nl_fit" (a named list) containing:

model The fitted model object.

method The fitting method used ("ns" or "gam").

y Name of the outcome variable.

x Name of the focal predictor.

time Name of the time variable, or NULL.

cluster Character vector of clustering variables, or NULL.

controls Character vector of control variable names, or NULL.

df Degrees of freedom used for splines::ns().

k Basis dimension used for mgcv::s() / mgcv::ti().

family The family object used.

formula The model formula passed to the fitter.

call The matched call.

x_info A list with quantiles and range of x for building prediction grids: q01, q99, min, max, n.

levels_info A named list of factor levels for time and any factor control variables.

control_defaults A named list of typical values for control variables: the mean for numeric variables and the first level for factors.

See Also

ns, gam, lmer, glmer

Examples

```
# --- Toy example (automatically tested by CRAN) ---
# Single-level natural spline on small simulated data
set.seed(1)
mydata <- data.frame(
  outcome        = rnorm(120),
  age            = runif(120, 18, 65),
  id             = rep(1:30, each = 4),
  wave           = factor(rep(1:4, times = 30)),
  sex            = factor(sample(c("F", "M"), 120, replace = TRUE)),
  baseline_score = rnorm(120)
)

fit_sl <- nl_fit(data = mydata, y = "outcome", x = "age", df = 4)


# Single-level GAM
fit_gam <- nl_fit(
  data   = mydata,
  y      = "outcome",
  x      = "age",
  method = "gam",
  k      = 5
)

# Multilevel spline with random intercepts
fit_ml <- nl_fit(
  data    = mydata,
  y       = "outcome",
  x       = "age",
  cluster = "id",
  df      = 4
)

# Spline with time interaction and controls
fit_t <- nl_fit(
  data     = mydata,
  y        = "outcome",
  x        = "age",
  time     = "wave",
  controls = c("sex", "baseline_score"),
  df       = 4
)
```

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:

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

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 `Residual` in the output so that all values sum to 1 (when a residual variance is available).

Value

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

See Also

`nl_fit`

Examples

```
# --- Toy example (automatically tested by CRAN) ---
# Small multilevel data: 10 clusters, 5 obs each (50 rows total)
set.seed(1)
toy <- data.frame(
  outcome = rnorm(50),
  age     = runif(50, 18, 65),
  id      = rep(1:10, each = 5)
)
fit <- nl_fit(
  data    = toy,
  y       = "outcome",
  x       = "age",
```

```
  cluster = "id",
  df      = 2
)
nl_icc(fit)


# Two-level clustering: students nested in schools
set.seed(1)
mydata <- data.frame(
  math_score = rnorm(240),
  SES        = rnorm(240),
  id         = rep(1:60, each = 4),
  schid      = rep(1:12, each = 20)
)
fit_ml <- nl_fit(
  data    = mydata,
  y       = "math_score",
  x       = "SES",
  cluster = c("id", "schid"),
  df      = 4
)
nl_icc(fit_ml)

# Without residual variance in output
nl_icc(fit_ml, include_residual = FALSE)
```

`nl_plot` *Plot predictions from nl_predict()*

Description

Visualizes predicted nonlinear effects returned by `nl_predict`. If a time variable is present (or detected), draws separate colored curves for each time level with optional confidence ribbons.

Usage

```
nl_plot(
  pred_df,
  x,
  time = NULL,
  show_ci = TRUE,
  ci_level = 0.95,
  y_lab = "Predicted outcome",
  x_lab = NULL,
  title = NULL,
  legend_title = NULL
)
```

Arguments

pred_df	A data frame returned by nl_predict. Must contain at minimum columns fit and the focal predictor x. For confidence bands, provide either lwr/upr or se.fit.
x	Character string naming the focal predictor column in pred_df.
time	Optional character string naming the time variable column in pred_df. If NULL, the function will try to auto-detect a sensible time column (e.g., "TimePoint", "time", "wave") if present.
show_ci	Logical; if TRUE, adds a confidence ribbon. Default TRUE.
ci_level	Numeric in (0, 1); confidence level used when computing CI from se.fit. Default 0.95.
y_lab	Character label for the y-axis. Default "Predicted outcome".
x_lab	Character label for the x-axis. Defaults to the value of x.
title	Optional plot title string.
legend_title	Optional legend title string. Defaults to the value of time.

Value

A ggplot object.

See Also

nl_predict, nl_fit

Examples

```
# --- Toy example (automatically tested by CRAN) ---
# Single-level spline: fit, predict, then plot
set.seed(1)
mydata <- data.frame(
  outcome = rnorm(120),
  age     = runif(120, 18, 65),
  id      = rep(1:30, each = 4)
)
fit  <- nl_fit(data = mydata, y = "outcome", x = "age", df = 4)
pred <- nl_predict(fit)
nl_plot(pred, x = "age")


# Custom axis labels and title
nl_plot(
  pred,
  x     = "age",
  y_lab = "Predicted Math Score",
  x_lab = "Age (years)",
  title = "Nonlinear Effect of Age"
)
```

```
# Without confidence ribbon
nl_plot(pred, x = "age", show_ci = FALSE)

# With time variable: separate curves per wave
set.seed(1)
mydata2 <- data.frame(
  outcome = rnorm(120),
  age     = runif(120, 18, 65),
  id      = rep(1:30, each = 4),
  wave    = factor(rep(1:4, times = 30))
)
fit_t  <- nl_fit(
  data = mydata2,
  y    = "outcome",
  x    = "age",
  time = "wave",
  df   = 4
)
pred_t <- nl_predict(fit_t)
nl_plot(pred_t, x = "age", time = "wave", legend_title = "Wave")
```

`nl_predict` *Generate predictions from an nl_fit model*

Description

Creates a prediction data frame over a grid of the focal predictor `x` (and optionally over `time`), holding control variables at typical values (means for numeric; reference levels for factors). For mixed models, predictions default to population-level curves (random effects excluded).

Usage

```
nl_predict(
  object,
  x_seq = NULL,
  time_levels = NULL,
  controls_fixed = NULL,
  se = TRUE,
  level = 0.95,
  re_form = NA,
  ...
)
```

Arguments

`object`	An `nl_fit` object returned by `nl_fit`.

x_seq	Optional numeric vector of x values to predict over. If NULL, uses 100 evenly-spaced points between the stored 1st and 99th percentiles (with fallback to the stored min/max range).
time_levels	Optional vector of time levels to predict for. If NULL and a time variable is present, uses stored factor levels.
controls_fixed	Optional named list giving specific values for control variables. If NULL, stored defaults are used (mean for numeric; first level for factors).
se	Logical; if TRUE, includes standard errors and confidence intervals where available. Default TRUE.
level	Confidence level for the interval. Default 0.95.
re_form	For mixed models (lmerMod / glmerMod), passed to predict(). Default NA gives population-level predictions (random effects set to zero).
...	Reserved for future use.

Value

A data frame (or tibble) with columns: the focal predictor x, time (if applicable), any control variables (at fixed values), fit, se.fit, lwr, and upr.

See Also

nl_fit, nl_plot

Examples

```
# --- Toy example (automatically tested by CRAN) ---
# Single-level natural spline: fit then predict
set.seed(1)
mydata <- data.frame(
  outcome = rnorm(120),
  age     = runif(120, 18, 65),
  id      = rep(1:30, each = 4)
)
fit  <- nl_fit(data = mydata, y = "outcome", x = "age", df = 4)
pred <- nl_predict(fit)
head(pred)


# Custom x grid
pred_custom <- nl_predict(fit, x_seq = seq(20, 60, by = 5))

# Without standard errors
pred_nose <- nl_predict(fit, se = FALSE)

# With time variable (spline x time interaction)
set.seed(1)
mydata2 <- data.frame(
  outcome = rnorm(120),
  age     = runif(120, 18, 65),
```

```
  id      = rep(1:30, each = 4),
  wave    = factor(rep(1:4, times = 30))
)
fit_t <- nl_fit(
  data = mydata2,
  y    = "outcome",
  x    = "age",
  time = "wave",
  df   = 4
)
pred_t <- nl_predict(fit_t)

# Multilevel model: population-level predictions
fit_ml <- nl_fit(
  data    = mydata,
  y       = "outcome",
  x       = "age",
  cluster = "id",
  df      = 4
)
pred_ml <- nl_predict(fit_ml)
```

`nl_summary` *Tidy coefficient table for an nl_fit model*

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

`object`	An `nl_fit` object returned by `nl_fit`.
`digits`	Integer; number of decimal places for rounding. If `NULL`, no rounding is applied. Default `3`.
`pvals`	Logical; if `TRUE`, attempts to include p-values. Default `TRUE`.
`df_method`	For `lmerMod` with `pvals = TRUE`: `"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

```
# --- Toy example (automatically tested by CRAN) ---
# Single-level natural spline on small simulated data
set.seed(1)
mydata <- data.frame(
  outcome = rnorm(120),
  age     = runif(120, 18, 65),
  id      = rep(1:30, each = 4)
)
fit <- nl_fit(data = mydata, y = "outcome", x = "age", df = 4)
nl_summary(fit)


# With p-values suppressed
nl_summary(fit, pvals = FALSE)

# No rounding
nl_summary(fit, digits = NULL)

# Multilevel model (lmerMod) with Satterthwaite p-values via lmerTest
set.seed(1)
mydata2 <- data.frame(
  outcome = rnorm(240),
  age     = runif(240, 18, 65),
  id      = rep(1:60, each = 4)
)
fit_ml <- nl_fit(
  data    = mydata2,
  y       = "outcome",
  x       = "age",
  cluster = "id",
  df      = 4
)
nl_summary(fit_ml)
```

`print.nl_fit` *Print method for nl_fit objects*

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` *Print method for summary_nl_fit objects*

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 *Summary method for nl_fit objects*

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

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

Value

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

Index

gam, *4*
glmer, *4*

lmer, *4*

nl_fit, 2, *5*, *7–13*
nl_icc, 5
nl_plot, 6, *9*
nl_predict, *6, 7*, 8
nl_summary, 10, *13*
ns, *4*

print.nl_fit, 11
print.summary_nl_fit, 12

summary.nl_fit, *11, 12*, 13