

# Package ‘MultiSpline’

May 7, 2026

**Type** Package

**Title** Spline-Based Nonlinear Modeling for Multilevel and Longitudinal Data

**Version** 0.2.0

**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 (Nakagawa-Schiegg marginal and conditional R-squared 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.

**Depends** R (>= 4.2.0)

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

**Suggests** lmerTest, knitr, rmarkdown, reformulas, numDeriv

**License** GPL-3

**Encoding** UTF-8

**Language** en-US

**URL** <https://github.com/causalfragility-lab/MultiSpline>

**BugReports** <https://github.com/causalfragility-lab/MultiSpline/issues>

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Subir Hait [aut, cre] (ORCID: <<https://orcid.org/0009-0004-9871-9677>>)

**Maintainer** Subir Hait <haitsubi@msu.edu>

**Repository** CRAN

**Date/Publication** 2026-04-16 07:02:39 UTC

## Contents

nl_compare . . . . .	2
nl_derivatives . . . . .	3
nl_fit . . . . .	4
nl_het . . . . .	6
nl_icc . . . . .	7
nl_knots . . . . .	8
nl_plot . . . . .	10
nl_predict . . . . .	11
nl_r2 . . . . .	12
nl_summary . . . . .	13
nl_turning_points . . . . .	14
print.nl_fit . . . . .	15
print.summary_nl_fit . . . . .	15
summary.nl_fit . . . . .	16
<b>Index</b>	<b>17</b>

---

 nl\_compare

*Built-in model comparison workflow*


---

## 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

## Usage

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

**Arguments**

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

```
## Not run:
fit <- nl_fit(data = mydata, y = "score", x = "age", df = 4)
nl_compare(fit)
nl_compare(fit, polynomial_degrees = c(2, 3, 4))

## End(Not run)
```

---

nl\_derivatives

*Compute first and second derivatives of the fitted spline curve*

---

**Description**

Uses numerical differentiation on the prediction grid to compute the first derivative (slope / marginal effect), the second derivative (curvature), and propagated confidence bands for both, using the delta method from the `se.fit` column of `nl_predict()`. Results are passed to [nl\\_turning\\_points](#) to identify local turning points and inflection regions.

**Usage**

```
nl_derivatives(pred_df, x, time = NULL, h = NULL, level = 0.95)
```

**Arguments**

pred_df	A data frame returned by <a href="#">nl_predict</a> .
x	Character; name of the focal predictor column.
time	Optional character; name of the time column. If present, derivatives are computed separately within each time level.
h	Numeric; step size for finite-difference approximation. Default NULL uses 0.1 percent of the x range.
level	Confidence level for derivative CIs. Default 0.95.

**Value**

A data frame with columns `x` (the focal predictor), `time` (if applicable), `fit` (predicted outcome), `d1` (first derivative), `d1_lwr` and `d1_upr` (lower and upper CI for the first derivative), `d2` (second derivative), and `d2_lwr` and `d2_upr` (CI for the second derivative).

**See Also**

[nl\\_turning\\_points](#), [nl\\_plot](#)

---

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").

Version 2 additions: two-way and nested clustering via the `nested` argument; random spline slopes via `random_slope`; B-spline basis via `method = "bs"`; automatic df selection via `df = "auto"`.

**Usage**

```
nl_fit(
  data,
  y,
  x,
  time = NULL,
  cluster = NULL,
  nested = FALSE,
  controls = 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(),
    ...
  )

```

## 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. NULL, "id", or c("id", "school") for two-way clustering.
<code>nested</code>	Logical; only used when <code>length(cluster) == 2</code> . If TRUE, uses a nested specification (1   g1/g2); if FALSE (default), uses cross-classified (1   g1) + (1   g2).
<code>controls</code>	Optional character vector of additional covariate names to include as linear fixed effects.
<code>method</code>	Spline basis to use: "ns" (natural cubic spline, default), "bs" (B-spline), or "gam" (GAM smooth via <code>mgcv::gam()</code> ). Multilevel fits require "ns" or "bs".
<code>df</code>	Degrees of freedom for the spline basis. Supply a single integer greater than or equal to 1, or the string "auto" 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: "AIC" (default) or "BIC".
<code>k</code>	Basis dimension for <code>mgcv::s()</code> when <code>method = "gam"</code> . Default 5.
<code>bs_degree</code>	Polynomial degree for <code>splines::bs()</code> when <code>method = "bs"</code> . Default 3 (cubic).
<code>random_slope</code>	Logical; if TRUE and <code>cluster</code> is supplied, fits random spline slopes in addition to random intercepts. Currently implemented only for single-level clustering. Default FALSE.
<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>...</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`).

## See Also

[nl\\_predict](#), [nl\\_derivatives](#), [nl\\_compare](#), [nl\\_r2](#), [nl\\_knots](#)

## Examples

```
## Not run:
# Single-level natural spline with automatic df selection
fit <- nl_fit(data = mydata, y = "score", x = "age", df = "auto")

# Two-way cross-classified clustering
fit2 <- nl_fit(
  data = mydata,
  y = "score",
  x = "age",
  cluster = c("student_id", "school_id"),
  nested = FALSE
)

# Nested clustering (students within schools)
fit3 <- nl_fit(
  data = mydata,
  y = "score",
  x = "age",
  cluster = c("student_id", "school_id"),
  nested = TRUE
)

# Random spline slopes
fit4 <- nl_fit(
  data = mydata,
  y = "score",
  x = "age",
  cluster = "id",
  random_slope = TRUE
)

## End(Not run)
```

---

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 a heterogeneity summary: variance of slopes across clusters 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.

**Usage**

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

**Arguments**

object	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).
n_clusters_plot	Maximum number of cluster curves to display in the trajectory plot. Default 30L. Use <code>Inf</code> for all clusters.
x_seq	Optional numeric vector of x values for prediction.
level	Confidence level for the population-mean ribbon. Default 0.95.
plot	Logical; print the trajectory plot. Default <code>TRUE</code> .
seed	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.

`slope_variance` Named numeric: SD of first-derivative estimates across clusters at each x value.

`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](#)

---

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 `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**

```
## Not run:
fit <- nl_fit(
  data = mydata,
  y = "math_score",
  x = "SES",
  cluster = c("id", "schid"),
  df = 4
)
nl_icc(fit)

## End(Not run)
```

---

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.

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,
  method = c("ns", "bs"),
  df_range = 2:10,
  criterion = c("AIC", "BIC"),
  family = stats::gaussian(),
  plot = TRUE,
  ...
)
```

**Arguments**

data	A data frame.
y	Outcome variable name (string).
x	Focal predictor name (string).
time	Optional time variable name.
cluster	Optional character vector of cluster variable names.
nested	Logical; nested clustering. Default FALSE.
controls	Optional character vector of control variable names.
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](#)

**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,
  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 <code>type = "trajectory" or "combo"</code> ).
<code>deriv_df</code>	A data frame from <code>nl_derivatives</code> (for <code>type = "slope", "curvature", or "combo"</code> ).
<code>x</code>	Character; name of the focal predictor column in <code>pred_df / deriv_df</code> .
<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.

turning_points	Data frame from <a href="#">nl_turning_points</a> (only used when <code>show_turning_points = TRUE</code> ).
zero_line	Logical; for slope / curvature plots, draw a dashed horizontal line at zero. Default TRUE.
y_lab	Y-axis label.
x_lab	X-axis label. Defaults to <code>x</code> .
title	Optional plot title.
legend_title	Optional legend title.

**Value**

A ggplot object (or a combined plot list for "combo").

**See Also**

[nl\\_predict](#), [nl\\_derivatives](#), [nl\\_turning\\_points](#)

---

nl_predict	<i>Generate predictions from an nl_fit model</i>
------------	--

---

**Description**

Creates a prediction data frame over a grid of the focal predictor `x` (and optionally over time), holding control variables at typical values. For mixed models, predictions default to population-level curves (random effects set to zero).

v2 improvements:

- **CI for glmerMod:** approximate confidence intervals are computed via the parametric bootstrap or the delta method. Set `glmer_ci = "delta"` (default, fast) or `"boot"` (more accurate, slower).
- **Cluster-specific predictions:** set `re_form = NULL` to include random effects in the predictions.

**Usage**

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

**Arguments**

object	An <code>nl_fit</code> object.
x_seq	Optional numeric vector of x values. If NULL, 200 evenly-spaced points between the 1st and 99th percentiles.
time_levels	Optional vector of time levels.
controls_fixed	Optional named list of fixed control values.
se	Logical; include SEs and CIs. Default TRUE.
level	Confidence level. Default 0.95.
re_form	For mixed models: NA (population-level, default) or NULL (include random effects).
glmer_ci	Method for glmerMod CIs: "delta" (default) or "boot" (parametric bootstrap).
n_boot	Number of bootstrap replicates when <code>glmer_ci = "boot"</code> . Default 500.
...	Reserved for future use.

**Value**

A data frame with columns for the focal predictor, time (if any), controls at fixed values, `fit`, `se.fit`, `lwr`, and `upr`.

**See Also**

[nl\\_fit](#), [nl\\_plot](#), [nl\\_derivatives](#)

---

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-Schiezeth 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 <code>nl_fit</code> object returned by <code>nl_fit</code> .
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  $\sigma_f^2$  is computed as the variance of the linear predictor from fixed effects only ( $\hat{\mu} = X\hat{\beta}$ ).

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.

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](#)

---

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 <code>nl_fit</code> object returned by <code>nl_fit</code> .
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 <code>lmerMod</code> with <code>pvals = TRUE</code> : "satterthwaite" (requires <b>lmerTest</b> ) 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](#)

---

`nl_turning_points`      *Identify turning points and inflection regions*

---

**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, time = NULL, tol = 1e-06)
```

**Arguments**

deriv_df	A data frame returned by <code>nl_derivatives</code> .
x	Character; name of the focal predictor column.
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](#)

---

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 <code>nl_fit</code> .
...	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 <code>summary_nl_fit</code> object.
...	Further arguments passed to <code>print</code> .

**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

<code>object</code>	An <code>nl_fit</code> object returned by <a href="#">nl_fit</a> .
<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 <b>lmerTest</b> ) or "none". Default "satterthwaite".
<code>...</code>	Further arguments passed to <a href="#">nl_summary</a> .

### Value

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

# Index

`nl_compare`, 2, 5, 9  
`nl_derivatives`, 3, 5, 7, 10–12, 14, 15  
`nl_fit`, 3, 4, 7–9, 12–16  
`nl_het`, 6  
`nl_icc`, 7, 13  
`nl_knots`, 3, 5, 8  
`nl_plot`, 4, 10, 12, 15  
`nl_predict`, 4, 5, 10, 11, 11  
`nl_r2`, 5, 12  
`nl_summary`, 13, 16  
`nl_turning_points`, 3, 4, 11, 14  
  
`print.nl_fit`, 15  
`print.summary_nl_fit`, 15  
  
`summary.nl_fit`, 14, 15, 16