

Package ‘neuralGAM’

December 3, 2025

Type Package

Title Interpretable Neural Network Based on Generalized Additive Models

Version 2.0.1

Maintainer Ines Ortega-Fernandez <iortega@gradient.org>

Description

Neural Additive Model framework based on Generalized Additive Models from Hastie & Tibshirani (1990, ISBN:9780412343902), which trains a different neural network to estimate the contribution of each feature to the response variable. The networks are trained independently leveraging the local scoring and backfitting algorithms to ensure that the Generalized Additive Model converges and it is additive. The resultant Neural Network is a highly accurate and interpretable deep learning model, which can be used for high-risk AI practices where decision-making should be based on accountable and interpretable algorithms.

License MPL-2.0

BugReports <https://github.com/inesortega/neuralGAM/issues>

Encoding UTF-8

Imports tensorflow, keras, ggplot2, magrittr, reticulate,
formula.tools, matrixStats, patchwork, rlang

SystemRequirements python (>= 3.10), keras (== 2.15), tensorflow (== 2.15)

RoxygenNote 7.3.3

Suggests covr, testthat (>= 3.0.0), fs, withr

Config/testthat/edition 3

URL <https://inesortega.github.io/neuralGAM/>,
<https://github.com/inesortega/neuralGAM>

NeedsCompilation no

Author Ines Ortega-Fernandez [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0002-8041-6860>>),
Marta Sestelo [aut, cph] (ORCID:
<<https://orcid.org/0000-0003-4284-6509>>)

Repository CRAN

Date/Publication 2025-12-03 08:40:02 UTC

Contents

autoplot.neuralGAM	2
diagnose	4
install_neuralGAM	6
neuralGAM	6
plot.neuralGAM	9
plot_history	10
predict.neuralGAM	11
print.neuralGAM	14
sim_neuralGAM_data	15
summary.neuralGAM	16
Index	18

autoplot.neuralGAM	<i>Autoplot method for neuralGAM objects (epistemic-only)</i>
--------------------	---

Description

Produce effect/diagnostic plots from a fitted neuralGAM model. Supported panels:

- which = "response": fitted response vs. index, with optional epistemic *confidence intervals* (CI).
- which = "link": linear predictor (link scale) vs. index, with optional CI.
- which = "terms": single per-term contribution $g_j(x_j)$ on the link scale, with optional CI band for the smooth (epistemic).

Usage

```
## S3 method for class 'neuralGAM'
autoplot(
  object,
  newdata = NULL,
  which = c("response", "link", "terms"),
  interval = c("none", "confidence"),
  level = 0.95,
  forward_passes = 150,
  term = NULL,
  rug = TRUE,
  ...
)
```

Arguments

object	A fitted neuralGAM object.
newdata	Optional data.frame/list of covariates. If omitted, training data are used.
which	One of c("response", "link", "terms"). Default "response".
interval	One of c("none", "confidence"). Default "confidence".
level	Coverage level for confidence intervals (e.g., 0.95). Default 0.95.
forward_passes	Integer. Number of MC-dropout forward passes used when uncertainty_method %in% c("epistemic", "both").
term	Single term name to plot when which = "terms".
rug	Logical; if TRUE (default), add rugs to continuous term plots.
...	Additional arguments passed to predict.neuralGAM.

Details**Uncertainty semantics (epistemic only)**

- **CI:** Uncertainty about the fitted mean.
- For the response, SEs are mapped via the delta method;
- For terms, bands are obtained as $\hat{g}_j \pm z \cdot SE(\hat{g}_j)$ on the link scale.

Value

A single ggplot object.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo

Examples

```
## Not run:

library(neuralGAM)
dat <- sim_neuralGAM_data()
train <- dat$train
test  <- dat$test

ngam <- neuralGAM(
  y ~ s(x1) + x2 + s(x3),
  data = train, family = "gaussian", num_units = 128,
  uncertainty_method = "epistemic", forward_passes = 10
)
## --- Autoplot (epistemic-only) ---
# Per-term effect with CI band
autoplot(ngam, which = "terms", term = "x1", interval = "confidence") +
  ggplot2::xlab("x1") + ggplot2::ylab("Partial effect")

# Request a different number of forward passes or CI level:
```

```

autoplot(ngam, which = "terms", term = "x1", interval = "confidence",
forward_passes = 15, level = 0.7)
# Response panel
autoplot(ngam, which = "response")

# Link panel with custom title
autoplot(ngam, which = "link") +
  ggplot2::ggtitle("Main Title")

## End(Not run)

```

diagnose

*Diagnosis plots to evaluate a fitted neuralGAM model.***Description**

Produce a 2x2 diagnostic panel for a fitted neuralGAM model, mirroring the layout of **gratia**'s `appraise()` for **mgcv** GAMs: (top-left) a QQ plot of residuals with optional simulation envelope, (top-right) a histogram of residuals, (bottom-left) residuals vs linear predictor η , and (bottom-right) observed vs fitted values on the response scale.

Usage

```

diagnose(
  object,
  data = NULL,
  response = NULL,
  qq_method = c("uniform", "simulate", "normal"),
  n_uniform = 1000,
  n_simulate = 200,
  residual_type = c("deviance", "pearson", "quantile"),
  level = 0.95,
  point_col = "steelblue",
  point_alpha = 0.5,
  hist_bins = 30
)

```

Arguments

<code>object</code>	A fitted neuralGAM model.
<code>data</code>	Optional <code>data.frame</code> for out-of-sample evaluation. If supplied, response must name the response column.
<code>response</code>	Character scalar giving the response variable name in data (required when data is provided).
<code>qq_method</code>	Character; one of "uniform", "simulate", or "normal" for the QQ reference. See Details.

n_uniform	Integer; number of $U(0,1)$ replicates for qq_method = "uniform".
n_simulate	Integer; number of simulated datasets for qq_method = "simulate" (also controls the QQ bands).
residual_type	One of "deviance", "pearson", or "quantile". Quantile (Dunn-Smyth) residuals are recommended for discrete families (binomial/poisson) because they are continuous and approximately standard normal under the fitted model, improving QQ diagnostics.
level	Numeric in (0,1); coverage level for the QQ bands when qq_method = "simulate".
point_col	Character; colour for points in scatter/histogram panels.
point_alpha	Numeric in (0,1); point transparency.
hist_bins	Integer; number of bins in the histogram.

Details

The function uses `predict.neuralGAM()` to obtain the linear predictor (`type = "link"`) and the fitted mean on the response scale (`type = "response"`). Residuals are computed internally for supported families; by default we use *deviance residuals*:

- **Gaussian:** $r_i = y_i - \hat{\mu}_i$.
- **Binomial:** $r_i = \text{sign}(y_i - \hat{\mu}_i) \sqrt{2w_i\{y_i \log(y_i/\hat{\mu}_i) + (1 - y_i) \log[(1 - y_i)/(1 - \hat{\mu}_i)]\}}$, with optional per-observation weights w_i (e.g., trials for proportions).
- **Poisson:** $r_i = \text{sign}(y_i - \hat{\mu}_i) \sqrt{2w_i\{y_i \log(y_i/\hat{\mu}_i) - (y_i - \hat{\mu}_i)\}}$, adopting the convention $y_i \log(y_i/\hat{\mu}_i) = 0$ when $y_i = 0$.

For Gaussian models, these plots diagnose symmetry, tail behaviour, and mean/variance misfit similar to standard GLM/GAM diagnostics. For non-Gaussian families (Binomial, Poisson), interpret shapes on the *deviance* scale, which is approximately normal under a well-specified model. For discrete data, *randomized quantile (Dunn-Smyth)* residuals are also available and often yield smoother QQ behaviour.

QQ reference methods. qq_method controls how theoretical quantiles are generated (as in **gratia**):

- "uniform" (default): draw $U(0,1)$ and map through the inverse CDF of the fitted response distribution at each observation; convert to residuals and average the sorted curves over n_uniform draws. Fast and respects the mean-variance relationship.
- "simulate": simulate n_simulate datasets from the fitted model at the observed covariates, compute residuals, and average the sorted curves; also provides pointwise level bands on the QQ plot.
- "normal": use standard normal quantiles; a fallback when a suitable RNG or inverse CDF is unavailable.

For Poisson models, include offsets for exposure in the linear predictor (e.g., $\log(E)$). The QQ methods use $\hat{\mu}_i$ with `qpois/rpois` for "uniform"/"simulate", respectively.

Value

A **patchwork** object combining four **ggplot2** plots. You can print it, add titles/themes, or extract individual panels if needed.

Dependencies

Requires **ggplot2** and **patchwork**.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo

References

Augustin, N.H., Sauleau, E.A., Wood, S.N. (2012). On quantile-quantile plots for generalized linear models. *Computational Statistics & Data Analysis*, **56**, 2404-2409. <https://doi.org/10.1016/j.csda.2012.01.026>

Dunn, P.K., Smyth, G.K. (1996). Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, **5**(3), 236-244.

install_neuralGAM	<i>Install neuralGAM python requirements</i>
-------------------	--

Description

Creates a conda environment (installing miniconda if required) and set ups the Python requirements to run neuralGAM (Tensorflow and Keras).

Miniconda and related environments are generated in the user's cache directory given by:

```
tools::R_user_dir('neuralGAM', 'cache')
```

Usage

```
install_neuralGAM()
```

neuralGAM	<i>Fit a neuralGAM model</i>
-----------	------------------------------

Description

Fits a Generalized Additive Model where smooth terms are modeled by keras neural networks. In addition to point predictions, the model can optionally estimate **uncertainty bands** via Monte Carlo Dropout across forward passes.

Usage

```

neuralGAM(
  formula,
  data,
  family = "gaussian",
  num_units = 64,
  learning_rate = 0.001,
  activation = "relu",
  kernel_initializer = "glorot_normal",
  kernel_regularizer = NULL,
  bias_regularizer = NULL,
  bias_initializer = "zeros",
  activity_regularizer = NULL,
  loss = "mse",
  uncertainty_method = c("none", "epistemic"),
  alpha = 0.05,
  forward_passes = 100,
  dropout_rate = 0.1,
  validation_split = NULL,
  w_train = NULL,
  bf_threshold = 0.001,
  ls_threshold = 0.1,
  max_iter_backfitting = 10,
  max_iter_ls = 10,
  seed = NULL,
  verbose = 1,
  ...
)

```

Arguments

formula	Model formula. Smooth terms must be wrapped in <code>s(...)</code> . You can specify per-term NN settings, e.g.: <code>y ~ s(x1, num_units = 1024) + s(x3, num_units = c(1024, 512))</code> .
data	Data frame containing the variables.
family	Response distribution: "gaussian", "binomial", "poisson".
num_units	Default hidden layer sizes for smooth terms (integer or vector). Mandatory unless every <code>s(...)</code> specifies its own <code>num_units</code> .
learning_rate	Learning rate for Adam optimizer.
activation	Activation function for hidden layers. Either a string understood by <code>tf\$keras\$activations\$get()</code> or a function.
kernel_initializer, bias_initializer	Initializers for weights and biases.
kernel_regularizer, bias_regularizer, activity_regularizer	Optional Keras regularizers.
loss	Loss function to use. Can be any Keras built-in (e.g., "mse", "mae", "huber", "logcosh") or a custom function, passed directly to <code>keras::compile()</code> .

uncertainty_method	Character string indicating the type of uncertainty to estimate. One of: <ul style="list-style-type: none"> • "none" (default): no uncertainty estimation. • "epistemic": MC Dropout for mean uncertainty (CIs)
alpha	Significance level for confidence intervals, e.g. 0.05 for 95% coverage.
forward_passes	Integer. Number of MC-dropout forward passes used when uncertainty_method %in% c("epistemic", "both").
dropout_rate	Dropout probability in smooth-term NNs (0,1). <ul style="list-style-type: none"> • During training: acts as a regularizer. • During prediction (if uncertainty_method is "epistemic"): enables MC Dropout sampling.
validation_split	Optional fraction of training data used for validation.
w_train	Optional training weights.
bf_threshold	Convergence criterion of the backfitting algorithm. Defaults to 0.001
ls_threshold	Convergence criterion of the local scoring algorithm. Defaults to 0.1
max_iter_backfitting	An integer with the maximum number of iterations of the backfitting algorithm. Defaults to 10.
max_iter_ls	An integer with the maximum number of iterations of the local scoring Algorithm. Defaults to 10.
seed	Random seed.
verbose	Verbosity: 0 silent, 1 progress messages.
...	Additional arguments passed to keras::optimizer_adam().

Value

An object of class "neuralGAM", a list with elements including:

- muhat** Numeric vector of fitted mean predictions (training data).
- partial** Data frame of partial contributions $g_j(x_j)$ per smooth term.
- y** Observed response values.
- eta** Linear predictor $\eta = \eta_0 + \sum_j g_j(x_j)$.
- lwr, upr** Lower/upper confidence interval bounds (response scale)
- x** Training covariates (inputs).
- model** List of fitted Keras models, one per smooth term (+ "linear" if present).
- eta0** Intercept estimate η_0 .
- family** Model family.
- stats** Data frame of training/validation losses per backfitting iteration.
- mse** Training mean squared error.
- formula** Parsed model formula (via get_formula_elements()).

history List of Keras training histories per term.

globals Global hyperparameter defaults.

alpha PI significance level (if trained with uncertainty).

build_pi Logical; whether the model was trained with uuncertainty estimation enabled

uncertainty_method Type of predictive uncertainty used ("none", "epistemic").

var_epistemic Matrix of per-term epistemic variances (if computed).

Author(s)

Ines Ortega-Fernandez, Marta Sestelo.

Examples

```
## Not run:

library(neuralGAM)
dat <- sim_neuralGAM_data()
train <- dat$train
test  <- dat$test

# Per-term architecture and confidence intervals
ngam <- neuralGAM(
  y ~ s(x1, num_units = c(128,64), activation = "tanh") +
    s(x2, num_units = 256),
  data = train,
  uncertainty_method = "epistemic",
  forward_passes = 10,
  alpha = 0.05
)
ngam

## End(Not run)
```

plot.neuralGAM

Visualization of neuralGAM object with base graphics

Description

Visualization of a fitted neuralGAM. Plots learned partial effects, either as scatter/line plots for continuous covariates or s for factor covariates. Confidence intervals can be added if available.

Usage

```
## S3 method for class 'neuralGAM'
plot(
  x,
  select = NULL,
```

```

    xlab = NULL,
    ylab = NULL,
    interval = c("none", "confidence", "prediction", "both"),
    level = 0.95,
    ...
  )

```

Arguments

<code>x</code>	A fitted neuralGAM object as produced by <code>neuralGAM()</code> .
<code>select</code>	Character vector of terms to plot. If <code>NULL</code> (default), all terms are plotted.
<code>xlab</code>	Optional custom x-axis label(s).
<code>ylab</code>	Optional custom y-axis label(s).
<code>interval</code>	One of <code>c("none", "confidence", "prediction", "both")</code> . Default <code>"none"</code> . Controls whether intervals are plotted.
<code>level</code>	Coverage level for intervals (e.g. <code>0.95</code>). Default <code>0.95</code> .
<code>...</code>	Additional graphical arguments passed to <code>plot()</code> .

Value

Produces plots on the current graphics device.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo.

<code>plot_history</code>	<i>Plot training loss history for a neuralGAM model</i>
---------------------------	---

Description

This function visualizes the training and/or validation loss at the end of each backfitting iteration for each term-specific model in a fitted neuralGAM object. It is designed to work with the history component of a trained neuralGAM model.

Usage

```
plot_history(model, select = NULL, metric = c("loss", "val_loss"))
```

Arguments

<code>model</code>	A fitted neuralGAM model.
<code>select</code>	Optional character vector of term names (e.g. <code>"x1"</code> or <code>c("x1", "x3")</code>) to subset the history. If <code>NULL</code> (default), all terms are included.
<code>metric</code>	Character vector indicating which loss metric(s) to plot. Options are <code>"loss"</code> , <code>"val_loss"</code> , or both. Defaults to both.

Value

A ggplot object showing the loss curves by backfitting iteration, with facets per term.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo

Examples

```
## Not run:
set.seed(123)
n <- 200
x1 <- runif(n, -2, 2)
x2 <- runif(n, -2, 2)
y <- 2 + x1^2 + sin(x2) + rnorm(n, 0, 0.1)
df <- data.frame(x1 = x1, x2 = x2, y = y)

model <- neuralGAM::neuralGAM(
  y ~ s(x1) + s(x2),
  data = df,
  num_units = 8,
  family = "gaussian",
  max_iter_backfitting = 2,
  max_iter_ls = 1,
  learning_rate = 0.01,
  seed = 42,
  validation_split = 0.2,
  verbose = 0
)

plot_history(model)                # Plot all terms
plot_history(model, select = "x1") # Plot just x1
plot_history(model, metric = "val_loss") # Plot only validation loss

## End(Not run)
```

predict.neuralGAM	<i>Produces predictions from a fitted neuralGAM object</i>
-------------------	--

Description

Generate predictions from a fitted neuralGAM model. Supported types:

- type = "link" (default): linear predictor on the link scale.
- type = "response": predictions on the response scale.
- type = "terms": per-term contributions to the linear predictor (no intercept).

Uncertainty estimation via MC Dropout (epistemic only)

- If `se.fit = TRUE`, standard errors (SE) of the *fitted mean* are returned (mgcv-style via Monte Carlo Dropout).
- For `type = "response"`, SEs are mapped to the response scale by the delta method: $se_{\mu} = |d\mu/d\eta| \cdot se_{\eta}$.
- `interval = "confidence"` returns CI bands derived from SEs; prediction intervals are not supported.
- For `type = "terms"`, `interval="confidence"` returns per-term CI matrices (and `se.fit` when requested).

Details

- Epistemic SEs (CIs) are obtained via Monte Carlo Dropout. When `type != "terms"` and SEs/CIs are requested in the presence of smooth terms, uncertainty is aggregated *jointly* to capture cross-term covariance in a single MC pass set. Otherwise, per-term variances are used (parametric variances are obtained from `stats::predict(..., se.fit=TRUE)`).
- For `type="terms"`, epistemic SEs and CI matrices are returned when requested.
- PIs are not defined on the link scale and are not supported.

Usage

```
## S3 method for class 'neuralGAM'
predict(
  object,
  newdata = NULL,
  type = c("link", "response", "terms"),
  terms = NULL,
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  forward_passes = 150,
  verbose = 1,
  ...
)
```

Arguments

<code>object</code>	A fitted <code>neuralGAM</code> object.
<code>newdata</code>	Optional <code>data.frame</code> /list of covariates at which to predict. If omitted, the training data cached in the object are used.
<code>type</code>	One of <code>c("link", "response", "terms")</code> . Default <code>"link"</code> .
<code>terms</code>	If <code>type = "terms"</code> , character vector of term names to include. If <code>NULL</code> , all terms are returned. Intercept is not included (as in mgcv).
<code>se.fit</code>	Logical; if <code>TRUE</code> , return SEs of the fitted mean (epistemic). Default <code>FALSE</code> . For <code>type="terms"</code> , returns a matrix of per-term SEs when available.
<code>interval</code>	One of <code>c("none", "confidence")</code> (default <code>"none"</code>). For <code>type="terms"</code> , setting <code>interval="confidence"</code> returns per-term CI matrices.

level	Coverage level for confidence intervals (e.g., 0.95). Default 0.95.
forward_passes	Integer; number of MC-dropout forward passes when computing epistemic uncertainty.
verbose	Integer (0/1). Default 1.
...	Other options (passed on to internal predictors).

Value

- type="terms":
 - interval="none": matrix of per-term contributions; if se.fit=TRUE, a list with \$fit, \$se.fit.
 - interval="confidence": a list with matrices \$fit, \$se.fit, \$lwr, \$upr.
- type="link" or type="response":
 - interval="none": vector (or list with \$fit, \$se.fit if se.fit=TRUE).
 - interval="confidence": data.frame with fit, lwr, upr.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo

Examples

```
## Not run:

library(neuralGAM)
dat <- sim_neuralGAM_data()
train <- dat$train
test  <- dat$test

ngam0 <- neuralGAM(
  y ~ s(x1) + x2 + s(x3),
  data = train, family = "gaussian",
  num_units = 128, uncertainty_method = "epistemic"
)
link_ci <- predict(ngam0, type = "link", interval = "confidence",
  level = 0.95, forward_passes = 10)
resp_ci <- predict(ngam0, type = "response", interval = "confidence",
  level = 0.95, forward_passes = 10)
trm_se  <- predict(ngam0, type = "terms",
  se.fit = TRUE, forward_passes = 10)

## End(Not run)
```

print.neuralGAM	<i>Short neuralGAM summary</i>
-----------------	--------------------------------

Description

Default print method for a neuralGAM object.

Usage

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

Arguments

x	A neuralGAM object.
...	Additional arguments (currently unused).

Value

Prints a brief summary of the fitted model including:

Distribution family The distribution family used ("gaussian", "binomial", or "poisson").

Formula The model formula.

Intercept value The fitted intercept (η_0).

Mean Squared Error (MSE) The training MSE of the model.

Training sample size The number of observations used to train the model.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo.

Examples

```
## Not run:

library(neuralGAM)
dat <- sim_neuralGAM_data()
train <- dat$train
test <- dat$test

ngam <- neuralGAM(
  y ~ s(x1) + x2 + s(x3),
  data = train,
  num_units = 128,
  family = "gaussian",
  activation = "relu",
  learning_rate = 0.001,
```

```

    bf_threshold = 0.001,
    max_iter_backfitting = 10,
    max_iter_ls = 10,
    seed = 1234
)
print(ngam)

## End(Not run)

```

sim_neuralGAM_data	<i>Simulate Example Data for NeuralGAM</i>
--------------------	--

Description

Generate a synthetic dataset for demonstrating and testing [neuralGAM](#). The response is constructed from three covariates: a quadratic effect, a linear effect, and a sinusoidal effect, plus Gaussian noise.

Usage

```
sim_neuralGAM_data(n = 2000, seed = 42, test_prop = 0.3)
```

Arguments

n	Integer. Number of observations to generate. Default 2000.
seed	Integer. Random seed for reproducibility. Default 42.
test_prop	Numeric in $[0, 1]$. Proportion of data to reserve for the test set. Default 0.3.

Details

The data generating process is:

$$y = 2 + x_1^2 + 2x_2 + \sin(x_3) + \varepsilon,$$

where $\varepsilon \sim N(0, 0.25^2)$.

Covariates x_1, x_2, x_3 are drawn independently from $U(-2.5, 2.5)$.

Value

A list with two elements:

- train: data.frame with training data.
- test: data.frame with test data.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo.

Examples

```
## Not run:
set.seed(123)
dat <- sim_neuralGAM_data(n = 500, test_prop = 0.2)

train <- dat$train
test  <- dat$test

## End(Not run)
```

summary.neuralGAM

*Summary of a neuralGAM model***Description**

Summarizes a fitted neuralGAM object: family, formula, sample size, intercept, training MSE, per-term neural net settings, per-term NN layer configuration, and training history. If a linear component is present, its coefficients are also reported.

Usage

```
## S3 method for class 'neuralGAM'
summary(object, ...)
```

Arguments

```
object      A neuralGAM object.
...         Additional arguments (currently unused).
```

Value

Invisibly returns object. Prints a human-readable summary.

Author(s)

Ines Ortega-Fernandez, Marta Sestelo

Examples

```
## Not run:

library(neuralGAM)
dat <- sim_neuralGAM_data()
train <- dat$train
test  <- dat$test

ngam <- neuralGAM(
```



```
y ~ s(x1) + x2 + s(x3),
data = train,
num_units = 128,
family = "gaussian",
activation = "relu",
learning_rate = 0.001,
bf_threshold = 0.001,
max_iter_backfitting = 10,
max_iter_ls = 10,
seed = 1234
)
summary(ngam)

## End(Not run)
```

Index

`autoplot.neuralGAM`, [2](#)
`diagnose`, [4](#)
`install_neuralGAM`, [6](#)
`neuralGAM`, [6](#), [15](#)
`plot.neuralGAM`, [9](#)
`plot_history`, [10](#)
`predict.neuralGAM`, [11](#)
`print.neuralGAM`, [14](#)
`sim_neuralGAM_data`, [15](#)
`summary.neuralGAM`, [16](#)