

# Package ‘gratia’

February 2, 2023

**Version** 0.8.1

**Date** 2023-02-02

**Title** Graceful 'ggplot'-Based Graphics and Other Functions for GAMs Fitted Using 'mgcv'

**Maintainer** Gavin L. Simpson <ucfagls@gmail.com>

**Depends** R (>= 4.1.0)

**Imports** mgcv (>= 1.8-41), ggplot2, tibble (>= 3.0.0), dplyr (>= 1.0.10), tidyr, rlang, patchwork, vctrs, grid, mvnfast, purrr, stats, tools, grDevices, stringr, tidyselect, lifecycle, pillar, cli, nlme

**Suggests** gamm4, lme4, testthat, vdiff, MASS, scam, datasets, withr, knitr, rmarkdown, forcats, GJRM

**Description** Graceful 'ggplot'-based graphics and utility functions for working with generalized additive models (GAMs) fitted using the 'mgcv' package. Provides a reimplementa-tion of the plot() method for GAMs that 'mgcv' provides, as well as 'tidyverse' compatible repre-sentations of estimated smooths.

**License** MIT + file LICENSE

**LazyData** true

**URL** <https://gavinsimpson.github.io/gratia/>

**BugReports** <https://github.com/gavinsimpson/gratia/issues>

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Gavin L. Simpson [aut, cre] (<<https://orcid.org/0000-0002-9084-8413>>),  
Henrik Singmann [ctb] (<<https://orcid.org/0000-0002-4842-3657>>)

**Repository** CRAN

**Date/Publication** 2023-02-02 16:50:10 UTC

**R topics documented:**

add_confint	4
add_constant	5
add_fitted	6
add_fitted.gam	6
add_partial_residuals	7
add_residuals	8
add_residuals.gam	9
appraise	10
basis	12
bird_move	14
check_user_select_smooths	15
coef.scam	15
compare_smooths	16
confint.fderiv	17
confint.gam	19
data_combos	21
data_sim	22
data_slice	23
derivatives	24
difference_smooths	26
draw	27
draw.basis	28
draw.compare_smooths	30
draw.derivatives	30
draw.difference_smooth	32
draw.evaluated_smooth	33
draw.gam	37
draw.gamlss	41
draw.mgcv_smooth	42
draw.parametric_effects	44
draw.penalty_df	45
draw.rootogram	46
draw.smooth_estimates	48
draw.smooth_samples	50
edf	52
evaluate_parametric_term	54
evaluate_smooth	54
eval_smooth	56
evenly	59
factor_combos	59
family.gam	60
family_name	60
family_type	61
fitted_samples	61
fitted_values	63
fixef	64

fixef.gam . . . . .	65
fix_offset . . . . .	66
get_by_smooth . . . . .	66
get_smooth . . . . .	67
get_smooths_by_id . . . . .	67
gss_vocab . . . . .	68
gw_f0 . . . . .	68
has_theta . . . . .	69
is_by_smooth . . . . .	69
is_factor_term . . . . .	70
is_mgcv_smooth . . . . .	71
is_offset . . . . .	71
link . . . . .	72
load_mgcv . . . . .	74
lp_matrix . . . . .	74
model_concurvity . . . . .	76
model_vars . . . . .	77
nb_theta . . . . .	78
n_smooths . . . . .	79
observed_fitted_plot . . . . .	79
overview . . . . .	80
parametric_effects . . . . .	81
parametric_terms . . . . .	82
partial_derivatives . . . . .	82
partial_residuals . . . . .	85
penalty . . . . .	87
posterior_samples . . . . .	88
predicted_samples . . . . .	90
qq_plot . . . . .	91
ref_level . . . . .	94
ref_sims . . . . .	94
rep_first_factor_value . . . . .	95
residuals_hist_plot . . . . .	95
residuals_linpred_plot . . . . .	96
rootogram . . . . .	97
seq_min_max_eps . . . . .	98
shift_values . . . . .	99
simulate.gam . . . . .	99
smallAges . . . . .	101
smooths . . . . .	102
smooth_coefs . . . . .	102
smooth_coef_indices . . . . .	103
smooth_data . . . . .	104
smooth_dim . . . . .	105
smooth_estimates . . . . .	106
smooth_samples . . . . .	107
smooth_type . . . . .	109
term_names . . . . .	112

term_variables . . . . .	112
theta . . . . .	113
tidy_basis . . . . .	114
too_far . . . . .	115
too_far_to_na . . . . .	115
to_na . . . . .	116
transform_fun . . . . .	116
typical_values . . . . .	117
variance_comp . . . . .	118
vars_from_label . . . . .	118
which_smooths . . . . .	119
worm_plot . . . . .	120
zooplankton . . . . .	122

<b>Index</b>	<b>123</b>
--------------	------------

---

add_confint	<i>Add a confidence interval to an existing object</i>
-------------	--

---

## Description

Add a confidence interval to an existing object

## Usage

```
add_confint(object, coverage = 0.95, ...)
```

```
## S3 method for class 'smooth_estimates'
add_confint(object, coverage = 0.95, ...)
```

```
## Default S3 method:
add_confint(object, coverage = 0.95, ...)
```

## Arguments

object	a R object.
coverage	numeric; the coverage for the interval. Must be in the range $0 < \text{coverage} < 1$ .
...	arguments passed to other methods.

---

add_constant	<i>Add a constant to estimated values</i>
--------------	---

---

**Description**

Add a constant to estimated values

**Usage**

```
add_constant(object, constant = NULL, ...)  
  
## S3 method for class 'evaluated_smooth'  
add_constant(object, constant = NULL, ...)  
  
## S3 method for class 'smooth_estimates'  
add_constant(object, constant = NULL, ...)  
  
## S3 method for class 'mgcv_smooth'  
add_constant(object, constant = NULL, ...)  
  
## S3 method for class 'parametric_effects'  
add_constant(object, constant = NULL, ...)  
  
## S3 method for class 'tbl_df'  
add_constant(object, constant = NULL, column = NULL, ...)  
  
## S3 method for class 'evaluated_parametric_term'  
add_constant(object, constant = NULL, ...)
```

**Arguments**

object	a object to add a constant to.
constant	the constant to add.
...	additional arguments passed to methods.
column	character; for the "tbl_df" method, which column to add the constant too.

**Value**

Returns object but with the estimate shifted by the addition of the supplied constant.

**Author(s)**

Gavin L. Simpson

---

add_fitted	<i>Add fitted values from a model to a data frame</i>
------------	---

---

**Description**

Add fitted values from a model to a data frame

**Usage**

```
add_fitted(data, model, value = ".value", ...)
```

**Arguments**

data	a data frame containing values for the variables used to fit the model. Passed to <code>stats::predict()</code> as <code>newdata</code> .
model	a fitted model for which a <code>stats::predict()</code> method is available. S3 method dispatch is performed on the <code>model</code> argument.
value	character; the name of the variable in which model predictions will be stored.
...	additional arguments passed to methods.

**Value**

A data frame (tibble) formed from `data` and fitted values from `model`.

---

add_fitted.gam	<i>Add fitted values from a GAM to a data frame</i>
----------------	---

---

**Description**

Add fitted values from a GAM to a data frame

**Usage**

```
## S3 method for class 'gam'
add_fitted(data, model, value = ".value", type = "response", prefix = ".", ...)
```

**Arguments**

data	a data frame containing values for the variables used to fit the model. Passed to <code>stats::predict()</code> as <code>newdata</code> .
model	a fitted model for which a <code>stats::predict()</code> method is available. S3 method dispatch is performed on the <code>model</code> argument.
value	character; the name of the variable in which model predictions will be stored.

type	character; the type of predictions to return. See <code>mgcv::predict.gam()</code> for options.
prefix	character; string to prepend to names of predicted values when type is "terms", "iterms", "lpmatrix". These prediction types result in a matrix of values being returned. prefix will be prepended to each of the names of columns returned by such prediction types.
...	additional arguments passed to <code>mgcv::predict.gam()</code> .

**Value**

A data frame (tibble) formed from data and predictions from model.

**Examples**

```
load_mgcv()

df <- data_sim("eg1", seed = 1)
df <- df[, c("y", "x0", "x1", "x2", "x3")]
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

##
add_fitted(df, m)

## with type = "terms" or "iterms"
add_fitted(df, m, type = "terms")
```

---

add\_partial\_residuals *Add partial residuals*

---

**Description**

Add partial residuals

**Usage**

```
add_partial_residuals(data, model, ...)

## S3 method for class 'gam'
add_partial_residuals(data, model, select = NULL, partial_match = FALSE, ...)
```

**Arguments**

data	a data frame containing values for the variables used to fit the model. Passed to <code>stats::residuals()</code> as newdata.
model	a fitted model for which a <code>stats::residuals()</code> method is available. S3 method dispatch is performed on the model argument.

... arguments passed to other methods.

select character, logical, or numeric; which smooths to plot. If NULL, the default, then all model smooths are drawn. Numeric select indexes the smooths in the order they are specified in the formula and stored in object. Character select matches the labels for smooths as shown for example in the output from `summary(object)`. Logical select operates as per numeric select in the order that smooths are stored.

partial\_match logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.

## Examples

```
load_mgcv()

df <- data_sim("eg1", seed = 1)
df <- df[, c("y", "x0", "x1", "x2", "x3")]
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = 'REML')

## add partial residuals
add_partial_residuals(df, m)

## add partial residuals for selected smooths
add_partial_residuals(df, m, select = "s(x0)")
```

---

add_residuals	<i>Add residuals from a model to a data frame</i>
---------------	---

---

## Description

Add residuals from a model to a data frame

## Usage

```
add_residuals(data, model, value = ".residual", ...)
```

## Arguments

data a data frame containing values for the variables used to fit the model. Passed to `stats::residuals()` as `newdata`.

model a fitted model for which a `stats::residuals()` method is available. S3 method dispatch is performed on the `model` argument.

value character; the name of the variable in which model residuals will be stored.

... additional arguments passed to methods.

## Value

A data frame (tibble) formed from `data` and residuals from `model`.



---

add\_residuals.gam      *Add residuals from a GAM to a data frame*

---

## Description

Add residuals from a GAM to a data frame

## Usage

```
## S3 method for class 'gam'  
add_residuals(data, model, value = ".residual", type = "deviance", ...)
```

## Arguments

data	a data frame containing values for the variables used to fit the model. Passed to <code>stats::predict()</code> as newdata.
model	a fitted model for which a <code>stats::predict()</code> method is available. S3 method dispatch is performed on the model argument.
value	character; the name of the variable in which model predictions will be stored.
type	character; the type of residuals to return. See <code>mgcv::residuals.gam()</code> for options.
...	additional arguments passed to <code>mgcv::residuals.gam()</code> .

## Value

A data frame (tibble) formed from data and residuals from model.

## Examples

```
load_mgcv()  
  
df <- data_sim("eg1", seed = 1)  
df <- df[, c("y", "x0", "x1", "x2", "x3")]  
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = 'REML')  
  
##  
add_residuals(df, m)
```

appraise

*Model diagnostic plots***Description**

Model diagnostic plots

**Usage**

appraise(model, ...)

```
## S3 method for class 'gam'
appraise(
  model,
  method = c("uniform", "simulate", "normal", "direct"),
  n_uniform = 10,
  n_simulate = 50,
  type = c("deviance", "pearson", "response"),
  n_bins = c("sturges", "scott", "fd"),
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  level = 0.9,
  ci_col = "black",
  ci_alpha = 0.2,
  point_col = "black",
  point_alpha = 1,
  line_col = "red",
  ...
)

## S3 method for class 'lm'
appraise(model, ...)
```

**Arguments**

model	a fitted model. Currently only class "gam".
...	arguments passed to <code>patchwork::wrap_plots()</code> .
method	character; method used to generate theoretical quantiles. Note that <code>method = "direct"</code> is deprecated in favour of <code>method = "uniform"</code> .
n_uniform	numeric; number of times to randomize uniform quantiles in the direct computation method ( <code>method = "direct"</code> ) for QQ plots.
n_simulate	numeric; number of data sets to simulate from the estimated model when using the simulation method ( <code>method = "simulate"</code> ) for QQ plots.
type	character; type of residuals to use. Only "deviance", "response", and "pearson" residuals are allowed.

<code>n_bins</code>	character or numeric; either the number of bins or a string indicating how to calculate the number of bins.
<code>ncol, nrow</code>	numeric; the numbers of rows and columns over which to spread the plots.
<code>guides</code>	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>
<code>level</code>	numeric; the coverage level for QQ plot reference intervals. Must be strictly $0 < \text{level} < 1$ . Only used with <code>method = "simulate"</code> .
<code>ci_alpha, ci_col</code>	numeric; the level of alpha transparency for the QQ plot reference interval when <code>method = "simulate"</code> , or points drawn in plots.
<code>point_col, point_alpha</code>	colour and transparency used to draw points in the plots. See <code>graphics::par()</code> section <b>Color Specification</b> . This is passed to the individual plotting functions, and therefore affects the points of all plots.
<code>line_col</code>	colour specification for the 1:1 line in the QQ plot and the reference line in the residuals vs linear predictor plot.

**Note**

The wording used in `mgcv::qq.gam()` uses *direct* in reference to the simulated residuals method (`method = "simulated"`). To avoid confusion, `method = "direct"` is deprecated in favour of `method = "uniform"`.

**See Also**

The plots are produced by functions `qq_plot()`, `residuals_linpred_plot()`, `residuals_hist_plot()`, and `observed_fitted_plot()`.

**Examples**

```
load_mgcv()
## simulate some data...
dat <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat)
## run some basic model checks
appraise(mod, point_col = "steelblue", point_alpha = 0.4)

## To change the theme for all panels use the & operator, for example to
## change the ggplot theme for all panels
library("ggplot2")
appraise(mod, point_col = "steelblue", point_alpha = 0.4,
         line_col = "black") & theme_minimal()
```

---

basis	<i>Basis expansions for smooths</i>
-------	-------------------------------------

---

**Description**

Creates a basis expansion from a definition of a smoother using the syntax of *mgcv*'s smooths via `mgcv::s()`, `mgcv::te()`, `mgcv::ti()`, and `mgcv::t2()`, or from a fitted GAM(M).

**Usage**

```
basis(object, ...)

## S3 method for class 'gam'
basis(
  object,
  term = NULL,
  data = NULL,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  partial_match = FALSE,
  ...
)

## S3 method for class 'scam'
basis(
  object,
  term = NULL,
  data = NULL,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  partial_match = FALSE,
  ...
)

## S3 method for class 'gamm'
basis(
  object,
  term = NULL,
  data = NULL,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
```

```

    partial_match = FALSE,
    ...
)

## S3 method for class 'list'
basis(
  object,
  term = NULL,
  data = NULL,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  partial_match = FALSE,
  ...
)

## Default S3 method:
basis(object, data, knots = NULL, constraints = FALSE, at = NULL, ...)

```

### Arguments

object	a smooth specification, the result of a call to one of <code>mgcv::s()</code> , <code>mgcv::te()</code> , <code>mgcv::ti()</code> , or <code>mgcv::t2()</code> , or a fitted GAM(M) model.
...	other arguments passed to <code>mgcv::smoothCon()</code> .
term	character; select smooths in a fitted model
data	a data frame containing the variables used in smooth.
n	numeric; the number of points over the range of the covariate at which to evaluate the smooth.
n_2d	numeric; the number of new observations for each dimension of a bivariate smooth. Not currently used; n is used for both dimensions.
n_3d	numeric; the number of new observations to generate for the third dimension of a 3D smooth.
n_4d	numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a $k$ D smooth ( $k \geq 4$ ). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get <code>n_4d</code> new observations.
partial_match	logical; in the case of character select, should select match partially against smooths? If <code>partial_match = TRUE</code> , select must only be a single string, a character vector of length 1.
knots	a list or data frame with named components containing knots locations. Names must match the covariates for which the basis is required. See <code>mgcv::smoothCon()</code> .
constraints	logical; should identifiability constraints be applied to the smooth basis. See argument <code>absorb.cons</code> in <code>mgcv::smoothCon()</code> .
at	a data frame containing values of the smooth covariate(s) at which the basis should be evaluated.

**Value**

A tibble.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()

df <- data_sim("eg4", n = 400, seed = 42)

bf <- basis(s(x0), data = df)
bf <- basis(s(x2, by = fac, bs = "bs"), data = df, constraints = TRUE)
```

---

bird\_move

*Simulated bird migration data*

---

**Description**

Data generated from a hypothetical study of bird movement along a migration corridor, sampled throughout the year. This dataset consists of simulated sample records of numbers of observed locations of 100 tagged individuals each from six species of bird, at ten locations along a latitudinal gradient, with one observation taken every four weeks. Counts were simulated randomly for each species in each location and week by creating a species-specific migration curve that gave the probability of finding an individual of a given species in a given location, then simulated the distribution of individuals across sites using a multinomial distribution, and subsampling that using a binomial distribution to simulation observation error (i.e. not every bird present at a location would be detected). The data set (bird\_move) consists of the variables count, latitude, week and species.

**Format**

A data frame

**Source**

Pedersen EJ, Miller DL, Simpson GL, Ross N. 2018. Hierarchical generalized additive models: an introduction with mgcv. *PeerJ Preprints* 6:e27320v1 doi:10.7287/peerj.preprints.27320v1.

---

`check_user_select_smooths`*Select smooths based on user's choices*

---

**Description**

Given a vector indexing the smooths of a GAM, returns a logical vector selecting the requested smooths.

**Usage**

```
check_user_select_smooths(  
  smooths,  
  select = NULL,  
  partial_match = FALSE,  
  model_name = NULL  
)
```

**Arguments**

<code>smooths</code>	character; a vector of smooth labels.
<code>select</code>	numeric, logical, or character vector of selected smooths.
<code>partial_match</code>	logical; in the case of character <code>select</code> , should <code>select</code> match partially against smooths? If <code>partial_match = TRUE</code> , <code>select</code> must only be a single string, a character vector of length 1.
<code>model_name</code>	character; a model name that will be used in error messages.

**Value**

A logical vector the same length as `length(smooths)` indicating which smooths have been selected.

**Author(s)**

Gavin L. Simpson

---

`coef.scam`*Extract coefficients from a fitted scam model.*

---

**Description**

Extract coefficients from a fitted scam model.

**Usage**

```
## S3 method for class 'scam'  
coef(object, parametrized = TRUE, ...)
```

**Arguments**

object	a model object fitted by scam()
parametrized	logical; extract parametrized coefficients, which respect the linear inequality constraints of the model.
...	other arguments.

---

compare_smooths	<i>Compare smooths across models</i>
-----------------	--------------------------------------

---

**Description**

Compare smooths across models

**Usage**

```
compare_smooths(
  model,
  ...,
  smooths = NULL,
  n = 100,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE
)
```

**Arguments**

model	Primary model for comparison.
...	Additional models to compare smooths against those of model.
smooths	character; vector of smooths to compare. If not specified comparisons will be performed for smooths common to all models supplied.
n	numeric; the number of points over the range of the covariate at which to evaluate the smooth.
data	a data frame of covariate values at which to evaluate the smooth.
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.
overall_uncertainty	logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?



**Examples**

```

load_mgcv()
dat <- data_sim("eg1", seed = 2)

## models to compare smooths across - artificially create differences
m1 <- gam(y ~ s(x0, k = 5) + s(x1, k = 5) + s(x2, k = 5) + s(x3, k = 5),
          data = dat, method = "REML")
m2 <- gam(y ~ s(x0, bs = 'ts') + s(x1, bs = 'ts') + s(x2, bs = 'ts') +
          s(x3, bs = 'ts'), data = dat, method = "REML")

## build comparisons
comp <- compare_smooths(m1, m2)
comp
## notice that the result is a nested tibble

draw(comp)

```

---

confint.fderiv	<i>Point-wise and simultaneous confidence intervals for derivatives of smooths</i>
----------------	--

---

**Description**

Calculates point-wise confidence or simultaneous intervals for the first derivatives of smooth terms in a fitted GAM.

**Usage**

```

## S3 method for class 'fderiv'
confint(
  object,
  parm,
  level = 0.95,
  type = c("confidence", "simultaneous"),
  nsim = 10000,
  ncores = 1L,
  ...
)

```

**Arguments**

object	an object of class "fderiv" containing the estimated derivatives.
parm	which parameters (smooth terms) are to be given intervals as a vector of terms. If missing, all parameters are considered.
level	numeric, $0 < \text{level} < 1$ ; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.

type	character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.
nsim	integer; the number of simulations used in computing the simultaneous intervals.
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
...	additional arguments for methods

### Value

a data frame with components:

1. term; factor indicating to which term each row relates,
2. lower; lower limit of the confidence or simultaneous interval,
3. est; estimated derivative
4. upper; upper limit of the confidence or simultaneous interval.

### Author(s)

Gavin L. Simpson

### Examples

```
load_mgcv()

dat <- data_sim("eg1", n = 1000, dist = "normal", scale = 2, seed = 2)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

# new data to evaluate the derivatives at, say over the middle 50% of range
# of each covariate
middle <- function(x, n = 25, coverage = 0.5) {
  v <- (1 - coverage) / 2
  q <- quantile(x, prob = c(0 + v, 1 - v), type = 8)
  seq(q[1], q[2], length = n)
}
new_data <- sapply(dat[c("x0", "x1", "x2", "x3")], middle)
new_data <- data.frame(new_data)
## first derivatives of all smooths...
fd <- fderiv(mod, newdata = new_data)

## point-wise interval
ci <- confint(fd, type = "confidence")
ci

## simultaneous interval for smooth term of x2
x2_sint <- confint(fd, parm = "x2", type = "simultaneous",
  nsim = 10000, ncores = 2)

x2_sint
```

---

 confint.gam

*Point-wise and simultaneous confidence intervals for smooths*


---

### Description

Calculates point-wise confidence or simultaneous intervals for the smooth terms of a fitted GAM.

### Usage

```
## S3 method for class 'gam'
confint(
  object,
  parm,
  level = 0.95,
  data = newdata,
  n = 100,
  type = c("confidence", "simultaneous"),
  nsim = 10000,
  shift = FALSE,
  transform = FALSE,
  unconditional = FALSE,
  ncores = 1,
  partial_match = FALSE,
  ...,
  newdata = NULL
)

## S3 method for class 'gamm'
confint(object, ...)

## S3 method for class 'list'
confint(object, ...)
```

### Arguments

object	an object of class "gam" or "gamm".
parm	which parameters (smooth terms) are to be given intervals as a vector of terms. If missing, all parameters are considered, although this is not currently implemented.
level	numeric, $0 < \text{level} < 1$ ; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.
data	data frame; new values of the covariates used in the model fit. The selected smooth(s) will be evaluated at the supplied values.

n	numeric; the number of points to evaluate smooths at.
type	character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.
nsim	integer; the number of simulations used in computing the simultaneous intervals.
shift	logical; should the constant term be add to the smooth?
transform	logical; should the smooth be evaluated on a transformed scale? For generalised models, this involves applying the inverse of the link function used to fit the model. Alternatively, the name of, or an actual, function can be supplied to transform the smooth and it's confidence interval.
unconditional	logical; if TRUE (and freq == FALSE) then the Bayesian smoothing parameter uncertainty corrected covariance matrix is returned, if available.
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
partial_match	logical; should matching parm use a partial match or an exact match? Can only be used if <code>length(parm)</code> is 1.
...	additional arguments for methods
newdata	DEPRECATED! data frame; containing new values of the covariates used in the model fit. The selected smooth(s) will be evaluated at the supplied values.

### Value

a tibble with components:

1. smooth; character indicating to which term each row relates,
2. type; the type of smooth,
3. by the name of the by variable if a by smooth, NA otherwise,
4. one or more vectors of values at which the smooth was evaluated, named as per the variables in the smooth,
5. zero or more variables containing values of the by variable,
6. est; estimated value of the smooth,
7. se; standard error of the estimated value of the smooth,
8. crit; critical value for the  $100 * level\%$  confidence interval.
9. lower; lower limit of the confidence or simultaneous interval,
10. upper; upper limit of the confidence or simultaneous interval,

### Author(s)

Gavin L. Simpson

**Examples**

```

load_mgcv()

dat <- data_sim("eg1", n = 1000, dist = "normal", scale = 2, seed = 2)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

# new data to evaluate the smooths at, say over the middle 50% of range
# of each covariate
middle <- function(x, n = 50, coverage = 0.5) {
  v <- (1 - coverage) / 2
  q <- quantile(x, prob = c(0 + v, 1 - v), type = 8)
  seq(q[1], q[2], length = n)
}
new_data <- sapply(dat[c("x0", "x1", "x2", "x3")], middle)
new_data <- data.frame(new_data)

## point-wise interval for smooth of x2
ci <- confint(mod, parm = "s(x2)", type = "confidence", data = new_data)
ci

## simultaneous interval for smooth of x2

si <- confint(mod, parm = "s(x2)", data = new_data,
              type = "simultaneous", nsim = 3000, ncores = 2)
si

```

---

data_combos	<i>All combinations of factor levels plus typical values of continuous variables</i>
-------------	--

---

**Description**

All combinations of factor levels plus typical values of continuous variables

**Usage**

```

data_combos(object, ...)

## S3 method for class 'gam'
data_combos(object, vars = everything(), complete = TRUE, ...)

```

**Arguments**

object	a fitted model object.
...	arguments passed to methods.
vars	terms to include or exclude from the returned object. Uses tidyselect principles.
complete	logical; should all combinations of factor levels be returned? If FALSE, only those combinations of levels observed in the model are retained.

---

data_sim	<i>Simulate example data for fitting GAMs</i>
----------	---

---

## Description

A tidy reimplementation of the functions implemented in `mgcv::gamSim()` that can be used to fit GAMs. An new feature is that the sampling distribution can be applied to all the example types.

## Usage

```
data_sim(  
  model = "eg1",  
  n = 400,  
  scale = 2,  
  theta = 3,  
  dist = c("normal", "poisson", "binary", "negbin", "tweedie"),  
  seed = NULL  
)
```

## Arguments

model	character; either "egX" where X is an integer 1:7, or the name of a model. See Details for possible options.
n	numeric; the number of observations to simulate.
scale	numeric; the level of noise to use.
theta	numeric; the dispersion parameter $\theta$ to use. The default is entirely arbitrary, chosen only to provide simulated data that exhibits extra dispersion beyond that assumed by under a Poisson.
dist	character; a sampling distribution for the response variable.
seed	numeric; the seed for the random number generator. Passed to <code>base::set.seed()</code> .

## Examples

```
data_sim("eg1")
```

---

data_slice	<i>Prepare a data slice through model covariates</i>
------------	--

---

### Description

Prepare a data slice through model covariates

### Usage

```
data_slice(object, ...)  
  
## Default S3 method:  
data_slice(object, ...)  
  
## S3 method for class 'gam'  
data_slice(object, ..., data = NULL)  
  
## S3 method for class 'gamm'  
data_slice(object, ...)  
  
## S3 method for class 'list'  
data_slice(object, ...)
```

### Arguments

object	an R model object.
...	<dynamic-dots> User supplied variables defining the data slice. Arguments passed via ... need to <i>named</i>
data	an alternative data frame of values containing all the variables needed to fit the model. If NULL, the default, the data used to fit the model will be recovered using <code>model.frame</code> . User-supplied expressions passed in ... will be evaluated in <code>data</code> .

### Examples

```
load_mgcv()  
  
# simulate some Gaussian data  
df <- data_sim("eg1", n = 50, seed = 2)  
  
# fit a GAM with 1 smooth and 1 linear term  
m1 <- gam(y ~ s(x2, k = 7) + x1, data = df, method = "REML")  
  
# Want to predict over f(x2) while holding `x1` at some value.  
# Default will use the observation closest to the median for unspecified  
# variables.  
ds <- data_slice(m1, x2 = evenly(x2, n = 50))
```

```
# for full control, specify the values you want
ds <- data_slice(m1, x2 = evenly(x2, n = 50), x1 = 0.3)

# or provide an expression (function call) which will be evaluated in the
# data frame passed to `data` or `model.frame(object)`
ds <- data_slice(m1, x2 = evenly(x2, n = 50), x1 = mean(x1))
```

---

derivatives

*Derivatives of estimated smooths via finite differences*


---

### Description

Derivatives of estimated smooths via finite differences

### Usage

```
derivatives(object, ...)

## Default S3 method:
derivatives(object, ...)

## S3 method for class 'gamm'
derivatives(object, ...)

## S3 method for class 'gam'
derivatives(
  object,
  term,
  data = newdata,
  order = 1L,
  type = c("forward", "backward", "central"),
  n = 200,
  eps = 1e-07,
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  level = 0.95,
  unconditional = FALSE,
  frequentist = FALSE,
  offset = NULL,
  ncores = 1,
  partial_match = FALSE,
  ...,
  newdata = NULL
)
```



**Arguments**

object	an R object to compute derivatives for.
...	arguments passed to other methods.
term	character; vector of one or more smooth terms for which derivatives are required. If missing, derivatives for all smooth terms will be returned. Can be a partial match to a smooth term; see argument <code>partial_match</code> below.
data	a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths.
order	numeric; the order of derivative.
type	character; the type of finite difference used. One of "forward", "backward", or "central".
n	numeric; the number of points to evaluate the derivative at.
eps	numeric; the finite difference.
interval	character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.
n_sim	integer; the number of simulations used in computing the simultaneous intervals.
level	numeric; $0 < \text{level} < 1$ ; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.
unconditional	logical; use smoothness selection-corrected Bayesian covariance matrix?
frequentist	logical; use the frequentist covariance matrix?
offset	numeric; a value to use for any offset term
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
partial_match	logical; should smooths be selected by partial matches with term? If TRUE, term can only be a single string to match against.
newdata	Deprecated: use data instead.

**Value**

A tibble, currently with the following variables:

- `smooth`: the smooth each row refers to,
- `var`: the name of the variable involved in the smooth,
- `data`: values of `var` at which the derivative was evaluated,
- `derivative`: the estimated derivative,
- `se`: the standard error of the estimated derivative,
- `crit`: the critical value such that  $\text{derivative} \pm (\text{crit} * \text{se})$  gives the upper and lower bounds of the requested confidence or simultaneous interval (given `level`),
- `lower`: the lower bound of the confidence or simultaneous interval,
- `upper`: the upper bound of the confidence or simultaneous interval.

**Note**

derivatives() will ignore any random effect smooths it encounters in object.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()

dat <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 42)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

## first derivatives of all smooths using central finite differences
derivatives(mod, type = "central")

## derivatives for a selected smooth
derivatives(mod, type = "central", term = "s(x1)")
## or via a partial match
derivatives(mod, type = "central", term = "x1", partial_match = TRUE)
```

---

difference\_smooths      *Differences of factor smooth interactions*

---

**Description**

Estimates pairwise differences (comparisons) between factor smooth interactions (smooths with a factor by argument) for pairs of groups defined by the factor. The group means can be optionally included in the difference.

**Usage**

```
difference_smooths(model, ...)

## S3 method for class 'gam'
difference_smooths(
  model,
  smooth,
  n = 100,
  ci_level = 0.95,
  data = NULL,
  group_means = FALSE,
  partial_match = TRUE,
  unconditional = FALSE,
  frequentist = FALSE,
```

```
    ...
  )
```

### Arguments

model	A fitted model.
...	arguments passed to other methods. Not currently used.
smooth	character; which smooth to compute differences for.
n	numeric; the number of points at which to evaluate the difference between pairs of smooths.
ci_level	numeric between 0 and 1; the coverage of credible interval.
data	data frame of locations at which to evaluate the difference between smooths.
group_means	logical; should the group means be included in the difference?
partial_match	logical; should smooth match partially against smooths? If partial_match = TRUE, smooth must only be a single string, a character vector of length 1. Unlike similar functions, the default here is TRUE because the intention is that users will be matching against factor-by smooth labels.
unconditional	logical; account for smoothness selection in the model?
frequentist	logical; use the frequentist covariance matrix?

### Examples

```
load_mgcv()

df <- data_sim("eg4", seed = 42)
m <- gam(y ~ fac + s(x2, by = fac) + s(x0), data = df, method = "REML")

sm_dif <- difference_smooths(m, smooth = "s(x2)")
sm_dif

draw(sm_dif)

# include the groups means for `fac` in the difference
sm_dif2 <- difference_smooths(m, smooth = "s(x2)", group_means = TRUE)
draw(sm_dif2)
```

---

 draw

*Generic plotting via ggplot2*


---

### Description

Generic plotting via ggplot2

**Usage**

```
draw(object, ...)
```

**Arguments**

object            and R object to plot.  
...                arguments passed to other methods.

**Details**

Generic function for plotting of R objects that uses the ggplot2 package.

**Value**

A `ggplot2::ggplot()` object.

**Author(s)**

Gavin L. Simpson

---

draw.basis

*Plot basis functions*

---

**Description**

Plots basis functions using ggplot2

**Usage**

```
## S3 method for class 'basis'  
draw(  
  object,  
  legend = FALSE,  
  labeller = NULL,  
  ylab = NULL,  
  title = NULL,  
  subtitle = NULL,  
  caption = NULL,  
  ncol = NULL,  
  nrow = NULL,  
  angle = NULL,  
  guides = "keep",  
  contour = FALSE,  
  n_contour = 10,  
  contour_col = "black",  
  ...  
)
```

**Arguments**

object	an object, the result of a call to <code>basis()</code> .
legend	logical; should a legend be drawn to indicate basis functions?
labeller	a labeller function with which to label facets. The default is to use <code>ggplot2::label_both()</code> .
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated from object.
title	character or expression; the title for the plot. See <code>ggplot2::labs()</code> .
subtitle	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> .
caption	character or expression; the plot caption. See <code>ggplot2::labs()</code> .
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <code>ggplot2::guide_axis()</code> .
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>
contour	logical; should contours be drawn on the plot using <code>ggplot2::geom_contour()</code> .
n_contour	numeric; the number of contour bins. Will result in <code>n_contour - 1</code> contour lines being drawn. See <code>ggplot2::geom_contour()</code> .
contour_col	colour specification for contour lines.
...	arguments passed to other methods. Not used by this method.

**Value**

A patchwork object.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()
df <- data_sim("eg1", n = 400, seed = 42)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

bf <- basis(m)
draw(bf)

bf <- basis(m, "s(x2)")
draw(bf)
```

---

draw.compare\_smooths *Plot comparisons of smooths*

---

### Description

Plot comparisons of smooths

### Usage

```
## S3 method for class 'compare_smooths'
draw(object, ncol = NULL, nrow = NULL, guides = "collect", ...)
```

### Arguments

object	of class "compare_smooths", the result of a call to <a href="#">compare_smooths()</a> .
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <a href="#">patchwork::plot_layout()</a>
...	additional arguments passed to <a href="#">patchwork::wrap_plots()</a> .

---

draw.derivatives *Plot derivatives of smooths*

---

### Description

Plot derivatives of smooths

### Usage

```
## S3 method for class 'derivatives'
draw(
  object,
  select = NULL,
  scales = c("free", "fixed"),
  alpha = 0.2,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  angle = NULL,
  ...
)

## S3 method for class 'partial_derivatives'
draw(
  object,
```

```

    select = NULL,
    scales = c("free", "fixed"),
    alpha = 0.2,
    ncol = NULL,
    nrow = NULL,
    guides = "keep",
    angle = NULL,
    ...
  )

```

## Arguments

object	a fitted GAM, the result of a call to <code>mgcv::gam()</code> .
select	character, logical, or numeric; which smooths to plot. If NULL, the default, then all model smooths are drawn. Numeric select indexes the smooths in the order they are specified in the formula and stored in object. Character select matches the labels for smooths as shown for example in the output from <code>summary(object)</code> . Logical select operates as per numeric select in the order that smooths are stored.
scales	character; should all univariate smooths be plotted with the same y-axis scale? If <code>scales = "free"</code> , the default, each univariate smooth has its own y-axis scale. If <code>scales = "fixed"</code> , a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.
alpha	numeric; alpha transparency for confidence or simultaneous interval.
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <code>ggplot2::guide_axis()</code> .
...	additional arguments passed to <code>patchwork::wrap_plots()</code> .

## Examples

```

load_mgcv()
dat <- data_sim("eg1", n = 800, dist = "normal", scale = 2, seed = 42)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

## first derivative of all smooths
df <- derivatives(mod, type = "central")
draw(df)
## fixed axis scales
draw(df, scales = "fixed")

```

---

`draw.difference_smooth`*Plot differences of smooths*

---

## Description

Plot differences of smooths

## Usage

```
## S3 method for class 'difference_smooth'  
draw(  
  object,  
  select = NULL,  
  rug = FALSE,  
  ref_line = FALSE,  
  contour = FALSE,  
  contour_col = "black",  
  n_contour = NULL,  
  ci_alpha = 0.2,  
  ci_col = "black",  
  smooth_col = "black",  
  line_col = "red",  
  scales = c("free", "fixed"),  
  ncol = NULL,  
  nrow = NULL,  
  guides = "keep",  
  xlab = NULL,  
  ylab = NULL,  
  title = NULL,  
  subtitle = NULL,  
  caption = NULL,  
  angle = NULL,  
  ...  
)
```

## Arguments

<code>object</code>	a fitted GAM, the result of a call to <code>mgcv::gam()</code> .
<code>select</code>	character, logical, or numeric; which smooths to plot. If <code>NULL</code> , the default, then all model smooths are drawn. Numeric <code>select</code> indexes the smooths in the order they are specified in the formula and stored in <code>object</code> . Character <code>select</code> matches the labels for smooths as shown for example in the output from <code>summary(object)</code> . Logical <code>select</code> operates as per numeric <code>select</code> in the order that smooths are stored.
<code>rug</code>	logical;



ref_line	logical;
contour	logical; should contour lines be added to smooth surfaces?
contour_col	colour specification for contour lines.
n_contour	numeric; the number of contour bins. Will result in $n\_contour - 1$ contour lines being drawn. See <code>ggplot2::geom_contour()</code> .
ci_alpha	numeric; alpha transparency for confidence or simultaneous interval.
ci_col	colour specification for the confidence/credible intervals band. Affects the fill of the interval.
smooth_col	colour specification for the the smooth or difference line.
line_col	colour specification for drawing reference lines
scales	character; should all univariate smooths be plotted with the same y-axis scale? If <code>scales = "free"</code> , the default, each univariate smooth has its own y-axis scale. If <code>scales = "fixed"</code> , a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>
xlab, ylab, title, subtitle, caption	character; labels with which to annotate plots
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <code>ggplot2::guide_axis()</code> .
...	additional arguments passed to <code>patchwork::wrap_plots()</code> .

## Examples

```
load_mgcv()
# simulate some data; a factor smooth example
df <- data_sim("eg4", seed = 42)
# fit GAM
m <- gam(y ~ fac + s(x2, by = fac) + s(x0), data = df, method = "REML")

# calculate the differences between pairs of smooths the f_j(x2) term
diffs <- difference_smooths(m, smooth = "s(x2)")
draw(diffs)
```

---

draw.evaluated\_smooth *Plot estimated smooths*

---

## Description

Plots estimated univariate and bivariate smooths using `ggplot2`.

**Usage**

```
## S3 method for class 'evaluated_1d_smooth'  
draw(  
  object,  
  rug = NULL,  
  ci_level = 0.95,  
  constant = NULL,  
  fun = NULL,  
  xlab,  
  ylab,  
  title = NULL,  
  subtitle = NULL,  
  caption = NULL,  
  partial_residuals = NULL,  
  response_range = NULL,  
  ...  
)  
  
## S3 method for class 'evaluated_2d_smooth'  
draw(  
  object,  
  show = c("estimate", "se"),  
  contour = TRUE,  
  contour_col = "black",  
  n_contour = NULL,  
  constant = NULL,  
  fun = NULL,  
  xlab,  
  ylab,  
  title = NULL,  
  subtitle = NULL,  
  caption = NULL,  
  response_range = NULL,  
  continuous_fill = NULL,  
  ...  
)  
  
## S3 method for class 'evaluated_re_smooth'  
draw(  
  object,  
  qq_line = TRUE,  
  constant = NULL,  
  fun = NULL,  
  xlab,  
  ylab,  
  title = NULL,  
  subtitle = NULL,  
  caption = NULL,
```

```

    response_range = NULL,
    ...
)

## S3 method for class 'evaluated_fs_smooth'
draw(
  object,
  rug = NULL,
  constant = NULL,
  fun = NULL,
  xlab,
  ylab,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  response_range = NULL,
  discrete_colour = NULL,
  ...
)

## S3 method for class 'evaluated_parametric_term'
draw(
  object,
  ci_level = 0.95,
  constant = NULL,
  fun = NULL,
  xlab,
  ylab,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  rug = TRUE,
  position = "identity",
  response_range = NULL,
  ...
)

```

### Arguments

object	an object, the result of a call to <code>evaluate_smooth()</code> .
rug	For <code>evaluate_smooth()</code> , a numeric vector of values for the location of data on the x axis. The default of <code>NULL</code> results in no rug plot being drawn. For <code>evaluate_parametric_terms()</code> , a logical to indicate if a rug plot should be drawn.
ci_level	numeric between 0 and 1; the coverage of credible interval.
constant	numeric; a constant to add to the estimated values of the smooth. <code>constant</code> , if supplied, will be added to the estimated value before the confidence band is computed.

<code>fun</code>	function; a function that will be applied to the estimated values and confidence interval before plotting. Can be a function or the name of a function. Function <code>fun</code> will be applied after adding any constant, if provided.
<code>xlab</code>	character or expression; the label for the x axis. If not supplied, a suitable label will be generated from object.
<code>ylab</code>	character or expression; the label for the y axis. If not supplied, a suitable label will be generated from object.
<code>title</code>	character or expression; the title for the plot. See <code>ggplot2::labs()</code> .
<code>subtitle</code>	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> .
<code>caption</code>	character or expression; the plot caption. See <code>ggplot2::labs()</code> .
<code>partial_residuals</code>	data frame; partial residuals and data values if partial residuals are drawn. Should have names <code>..p_resid</code> and <code>..orig_x</code> if supplied.
<code>response_range</code>	numeric; a vector of two values giving the range of response data for the guide. Used to fix plots to a common scale/range. Ignored if <code>show</code> is set to "se".
<code>...</code>	arguments passed to other methods.
<code>show</code>	character; plot the estimated smooth ("estimate") or its standard error ("se").
<code>contour</code>	logical; should contours be draw on the plot using <code>ggplot2::geom_contour()</code> .
<code>contour_col</code>	colour specification for contour lines.
<code>n_contour</code>	numeric; the number of contour bins. Will result in <code>n_contour - 1</code> contour lines being drawn. See <code>ggplot2::geom_contour()</code> .
<code>continuous_fill</code>	suitable scale used for the filled surface. If NULL, the default used is <code>scale_fill_distiller(palette = "RdBu", type = "div")</code> .
<code>qq_line</code>	logical; draw a reference line through the lower and upper theoretical quartiles.
<code>discrete_colour</code>	an appropriate discrete colour scale from <code>ggplot2</code> . The scale will need to be able to provide as many colours as there are levels in the factor variable involved in the smooth. Suitable alternatives include <code>ggplot2::scale_colour_viridis_d()</code> .
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.

**Value**

A `ggplot2::ggplot()` object.

**Author(s)**

Gavin L. Simpson

**Examples**

```

load_mgcv()

dat <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

sm <- evaluate_smooth(m1, "s(x2)")
draw(sm)

## supply constant to shift y axis scale
draw(sm, constant = coef(m1)[1])

dat <- data_sim("eg2", n = 1000, dist = "normal", scale = 1, seed = 2)
m2 <- gam(y ~ s(x, z, k = 40), data = dat, method = "REML")

sm <- evaluate_smooth(m2, "s(x,z)", n = 100)
draw(sm)

```

---

draw.gam

*Plot estimated smooths from a fitted GAM*


---

**Description**

Plots estimated smooths from a fitted GAM model in a similar way to `mgcv::plot.gam()` but instead of using base graphics, `ggplot2::ggplot()` is used instead.

**Usage**

```

## S3 method for class 'gam'
draw(
  object,
  data = NULL,
  select = NULL,
  parametric = FALSE,
  terms = NULL,
  residuals = FALSE,
  scales = c("free", "fixed"),
  ci_level = 0.95,
  n = 100,
  n_3d = 16,
  n_4d = 4,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  constant = NULL,
  fun = NULL,
  dist = 0.1,

```

```

rug = TRUE,
contour = TRUE,
ci_alpha = 0.2,
ci_col = "black",
smooth_col = "black",
resid_col = "steelblue3",
contour_col = "black",
n_contour = NULL,
partial_match = FALSE,
discrete_colour = NULL,
discrete_fill = NULL,
continuous_colour = NULL,
continuous_fill = NULL,
position = "identity",
angle = NULL,
ncol = NULL,
nrow = NULL,
guides = "keep",
widths = NULL,
heights = NULL,
projection = "orthographic",
orientation = NULL,
wrap = TRUE,
...
)

```

### Arguments

object	a fitted GAM, the result of a call to <code>mgcv::gam()</code> .
data	a optional data frame that may or may not be used? FIXME!
select	character, logical, or numeric; which smooths to plot. If NULL, the default, then all model smooths are drawn. Numeric select indexes the smooths in the order they are specified in the formula and stored in object. Character select matches the labels for smooths as shown for example in the output from <code>summary(object)</code> . Logical select operates as per numeric select in the order that smooths are stored.
parametric	logical; plot parametric terms also? Note that select is used for selecting which smooths to plot. The terms argument is used to select which parametric effects are plotted. The default, as with <code>mgcv::plot.gam()</code> , is to not draw parametric effects.
terms	character; which model parametric terms should be drawn? The Default of NULL will plot all parametric terms that can be drawn.
residuals	logical; should partial residuals for a smooth be drawn? Ignored for anything but a simple univariate smooth.
scales	character; should all univariate smooths be plotted with the same y-axis scale? If <code>scales = "free"</code> , the default, each univariate smooth has its own y-axis scale. If <code>scales = "fixed"</code> , a common y axis scale is used for all univariate smooths.

	Currently does not affect the y-axis scale of plots of the parametric terms.
ci_level	numeric between 0 and 1; the coverage of credible interval.
n	numeric; the number of points over the range of the covariate at which to evaluate the smooth.
n_3d	numeric; the number of new observations to generate for the third dimension of a 3D smooth.
n_4d	numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a $k$ D smooth ( $k \geq 4$ ). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get $n_{4d}$ new observations.
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.
overall_uncertainty	logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?
constant	numeric; a constant to add to the estimated values of the smooth. constant, if supplied, will be added to the estimated value before the confidence band is computed.
fun	function; a function that will be applied to the estimated values and confidence interval before plotting. Can be a function or the name of a function. Function fun will be applied after adding any constant, if provided.
dist	numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and dist is a distance within the unit square. See <code>mgcv::exclude.too.far()</code> for further details.
rug	logical; draw a rug plot at the bottom of each plot for 1-D smooths or plot locations of data for higher dimensions.
contour	logical; should contours be draw on the plot using <code>ggplot2::geom_contour()</code> .
ci_alpha	numeric; alpha transparency for confidence or simultaneous interval.
ci_col	colour specification for the confidence/credible intervals band. Affects the fill of the interval.
smooth_col	colour specification for the smooth line.
resid_col	colour specification for the partial residuals.
contour_col	colour specification for contour lines.
n_contour	numeric; the number of contour bins. Will result in $n_{contour} - 1$ contour lines being drawn. See <code>ggplot2::geom_contour()</code> .
partial_match	logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.
discrete_colour, discrete_fill, continuous_colour, continuous_fill	suitable scales for the types of data.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <code>ggplot2::guide_axis()</code> .

<code>ncol, nrow</code>	numeric; the numbers of rows and columns over which to spread the plots
<code>guides</code>	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>
<code>widths, heights</code>	The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. If there is more than 1 plot and <code>widths = NULL</code> , the value of <code>widths</code> will be set internally to <code>widths = 1</code> to accommodate plots of smooths that use a fixed aspect ratio.
<code>projection</code>	character; projection to use, see <code>ggplot2::coord_map()</code> for details.
<code>orientation</code>	an optional vector <code>c(latitude, longitude, rotation)</code> which describes where the "North Pole" should be when computing the projection. The third value is a clockwise rotation (in degrees), which defaults to the midrange of the longitude coordinates in the data. The default values for <code>orientation</code> therefore are <code>'c(20, 0, mean(range(longitude)))'</code> if this is not specified by the user. See links in <code>ggplot2::coord_map()</code> for more information.
<code>wrap</code>	logical; wrap plots as a patchwork?
<code>...</code>	additional arguments passed to <code>patchwork::wrap_plots()</code> .

**Value**

The object returned is created by `patchwork::wrap_plots()`.

**Note**

Internally, plots of each smooth are created using `ggplot2::ggplot()` and composed into a single plot using `patchwork::wrap_plots()`. As a result, it is not possible to use `+` to add to the plots in the way one might typically work with `ggplot()` plots. Instead, use the `&` operator; see the examples.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()

# simulate some data
df1 <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)
# fit GAM
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df1, method = "REML")

# plot all smooths
draw(m1)

# can add partial residuals
draw(m1, residuals = TRUE)

df2 <- data_sim(2, n = 1000, dist = "normal", scale = 1, seed = 2)
m2 <- gam(y ~ s(x, z, k = 40), data = df2, method = "REML")
```



```

draw(m2, contour = FALSE, n = 50)

# change the number of contours drawn and the fill scale used for
# the surface
library("ggplot2")
draw(m2, n_contour = 5, n = 50,
     continuous_fill = scale_fill_distiller(palette = "Spectral",
                                           type = "div"))

# See https://gavinsimpson.github.io/gratia/articles/custom-plotting.html
# for more examples and for details on how to modify the theme of all the
# plots produced by draw()
# to modify all panels, for example to change the theme, use the & operator

```

---

draw.gamlss

*Plot smooths of a GAMLSS model estimated by GJRM: :gamlss*


---

## Description

Provides a `draw()` method for GAMLSS (distributional GAMs) fitted by `GJRM: :gamlss()`.

## Usage

```

## S3 method for class 'gamlss'
draw(
  object,
  scales = c("free", "fixed"),
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  widths = NULL,
  heights = NULL,
  ...
)

```

## Arguments

<code>object</code>	a model, fitted by <code>GJRM: :gamlss()</code>
<code>scales</code>	character; should all univariate smooths be plotted with the same y-axis scale? If <code>scales = "free"</code> , the default, each univariate smooth has its own y-axis scale. If <code>scales = "fixed"</code> , a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.
<code>ncol, nrow</code>	numeric; the numbers of rows and columns over which to spread the plots
<code>guides</code>	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork: :plot_layout()</code>

widths, heights

The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. If there is more than 1 plot and widths = NULL, the value of widths will be set internally to widths = 1 to accommodate plots of smooths that use a fixed aspect ratio.

... arguments passed to `draw.gam()`

### Note

Plots of smooths are not labelled with the linear predictor to which they belong.

### Examples

```
if (require("GJRM", quietly = TRUE)) {
  # follow example from ?GJRM::gamlss
  library(GJRM)
  set.seed(0)
  n <- 100
  x1 <- round(runif(n))
  x2 <- runif(n)
  x3 <- runif(n)
  f1 <- function(x) cos(pi*2*x) + sin(pi*x)
  y1 <- -1.55 + 2*x1 + f1(x2) + rnorm(n)
  dataSim <- data.frame(y1, x1, x2, x3)

  eq_mu <- y1 ~ x1 + s(x2)
  eq_s <- ~ s(x3, k = 6)
  fl <- list(eq_mu, eq_s)
  m <- gamlss(fl, data = dataSim)

  draw(m)
}
```

---

draw.mgcv\_smooth

*Plot basis functions*

---

### Description

Plots basis functions using ggplot2

### Usage

```
## S3 method for class 'mgcv_smooth'
draw(
  object,
  legend = FALSE,
  use_facets = TRUE,
  labeller = NULL,
  xlab,
```

```

  ylab,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  angle = NULL,
  ...
)

```

## Arguments

object	an object, the result of a call to <code>basis()</code> .
legend	logical; should a legend be drawn to indicate basis functions?
use_facets	logical; for factor by smooths, use facets to show the basis functions for each level of the factor? If FALSE, a separate ggplot object will be created for each level and combined using <code>patchwork::wrap_plots()</code> . <b>Currently ignored.</b>
labeller	a labeller function with which to label facets. The default is to use <code>ggplot2::label_both()</code> .
xlab	character or expression; the label for the x axis. If not supplied, a suitable label will be generated from object.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated from object.
title	character or expression; the title for the plot. See <code>ggplot2::labs()</code> .
subtitle	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> .
caption	character or expression; the plot caption. See <code>ggplot2::labs()</code> .
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <code>ggplot2::guide_axis()</code> .
...	arguments passed to other methods. Not used by this method.

## Value

A `ggplot2::ggplot()` object.

## Author(s)

Gavin L. Simpson

## Examples

```

load_mgcv()
df <- data_sim("eg4", n = 400, seed = 42)

bf <- basis(s(x0), data = df)
draw(bf)

bf <- basis(s(x2, by = fac, bs = "bs"), data = df)
draw(bf)

```

---

```
draw.parametric_effects
```

*Plot estimated effects for model parametric terms*

---

### Description

Plot estimated effects for model parametric terms

### Usage

```
## S3 method for class 'parametric_effects'
draw(
  object,
  scales = c("free", "fixed"),
  ci_level = 0.95,
  ci_col = "black",
  ci_alpha = 0.2,
  line_col = "black",
  constant = NULL,
  fun = NULL,
  rug = TRUE,
  position = "identity",
  angle = NULL,
  ...,
  ncol = NULL,
  nrow = NULL,
  guides = "keep"
)
```

### Arguments

object	a fitted GAM, the result of a call to <code>mgcv::gam()</code> .
scales	character; should all univariate smooths be plotted with the same y-axis scale? If <code>scales = "free"</code> , the default, each univariate smooth has its own y-axis scale. If <code>scales = "fixed"</code> , a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.
ci_level	numeric between 0 and 1; the coverage of credible interval.
ci_col	colour specification for the confidence/credible intervals band. Affects the fill of the interval.
ci_alpha	numeric; alpha transparency for confidence or simultaneous interval.
line_col	colour specification used for regression lines of linear continuous terms.
constant	numeric; a constant to add to the estimated values of the smooth. <code>constant</code> , if supplied, will be added to the estimated value before the confidence band is computed.

fun	function; a function that will be applied to the estimated values and confidence interval before plotting. Can be a function or the name of a function. Function fun will be applied after adding any constant, if provided.
rug	logical; draw a rug plot at the bottom of each plot for 1-D smooths or plot locations of data for higher dimensions.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <code>ggplot2::guide_axis()</code> .
...	additional arguments passed to <code>patchwork::wrap_plots()</code> .
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>

---

draw.penalty_df	<i>Display penalty matrices of smooths using ggplot</i>
-----------------	---

---

## Description

Displays the penalty matrices of smooths as a heatmap using ggplot

## Usage

```
## S3 method for class 'penalty_df'
draw(
  object,
  normalize = FALSE,
  continuous_fill = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  ...
)
```

## Arguments

object	an object, the result of a call to <code>evaluate_smooth()</code> .
normalize	logical; normalize the penalty to the range -1, 1?
continuous_fill	suitable scale used for the filled surface. If NULL, the default used is <code>scale_fill_distiller(palette = "RdBu", type = "div")</code> .

xlab	character or expression; the label for the x axis. If not supplied, no axis label will be drawn. May be a vector, one per penalty.
ylab	character or expression; the label for the y axis. If not supplied, no axis label will be drawn. May be a vector, one per penalty.
title	character or expression; the title for the plot. See <code>ggplot2::labs()</code> . May be a vector, one per penalty.
subtitle	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> . May be a vector, one per penalty.
caption	character or expression; the plot caption. See <code>ggplot2::labs()</code> . May be a vector, one per penalty.
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots.
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <code>patchwork::plot_layout()</code>
...	arguments passed to other methods.

### Examples

```
load_mgcv()
dat <- data_sim("eg4", n = 400, seed = 42)
m <- gam(y ~ s(x0) + s(x1, bs = 'cr') + s(x2, bs = 'bs', by = fac),
         data = dat, method = "REML")

## produce a multi-panel plot of all penalties
draw(penalty(m))

# for a specific smooth
draw(penalty(m, smooth = "s(x2):fac1"))
```

---

draw.rootogram	<i>Draw a rootogram</i>
----------------	-------------------------

---

### Description

A rootogram is a model diagnostic tool that assesses the goodness of fit of a statistical model. The observed values of the response are compared with those expected from the fitted model. For discrete, count responses, the frequency of each count (0, 1, 2, etc) in the observed data and expected from the conditional distribution of the response implied by the model are compared. For continuous variables, the observed and expected frequencies are obtained by grouping the data into bins. The rootogram is drawn using `ggplot2::ggplot()` graphics. The design closely follows Kleiber & Zeileis (2016).

### Usage

```
## S3 method for class 'rootogram'
draw(
  object,
  type = c("hanging", "standing", "suspended"),
```

```

sqrt = TRUE,
ref_line = TRUE,
warn_limits = TRUE,
fitted_colour = "steelblue",
bar_colour = NA,
bar_fill = "grey",
ref_line_colour = "black",
warn_line_colour = "black",
ylab = NULL,
xlab = NULL,
...
)

```

### Arguments

object	and R object to plot.
type	character; the type of rootogram to draw.
sqrt	logical; show the observed and fitted frequencies
ref_line	logical; draw a reference line at zero?
warn_limits	logical; draw Tukey's warning limit lines at +/- 1?
fitted_colour, bar_colour, bar_fill, ref_line_colour, warn_line_colour	colours used to draw the respective element of the rootogram.
xlab, ylab	character; labels for the x and y axis of the rootogram. May be missing (NULL), in which case suitable labels will be used. '
...	arguments passed to other methods.

### Value

A 'ggplot' object.

### References

Kleiber, C., Zeileis, A., (2016) Visualizing Count Data Regressions Using Rootograms. *Am. Stat.* **70**, 296–303. doi:[10.1080/00031305.2016.1173590](https://doi.org/10.1080/00031305.2016.1173590)

### See Also

[rootogram\(\)](#) to compute the data for the rootogram.

### Examples

```

load_mgcv()
df <- data_sim("eg1", n = 1000, dist = "poisson", scale = 0.1, seed = 6)

# A poisson example
m <- gam(y ~ s(x0, bs = "cr") + s(x1, bs = "cr") + s(x2, bs = "cr") +
        s(x3, bs = "cr"), family = poisson(), data = df, method = "REML")
rg <- rootogram(m)

```

```
# plot the rootogram
draw(rg)

# change the type of rootogram
draw(rg, type = "suspended")
```

---

draw.smooth\_estimates *Plot the result of a call to smooth\_estimates()*

---

## Description

Plot the result of a call to smooth\_estimates()

## Usage

```
## S3 method for class 'smooth_estimates'
draw(
  object,
  constant = NULL,
  fun = NULL,
  contour = TRUE,
  contour_col = "black",
  n_contour = NULL,
  ci_alpha = 0.2,
  ci_col = "black",
  smooth_col = "black",
  resid_col = "steelblue3",
  partial_match = FALSE,
  discrete_colour = NULL,
  discrete_fill = NULL,
  continuous_colour = NULL,
  continuous_fill = NULL,
  angle = NULL,
  ylim = NULL,
  projection = "orthographic",
  orientation = NULL,
  ...
)
```

## Arguments

object	a fitted GAM, the result of a call to <code>mgcv::gam()</code> .
constant	numeric; a constant to add to the estimated values of the smooth. constant, if supplied, will be added to the estimated value before the confidence band is computed.



fun	function; a function that will be applied to the estimated values and confidence interval before plotting. Can be a function or the name of a function. Function fun will be applied after adding any constant, if provided.
contour	logical; should contours be draw on the plot using <code>ggplot2::geom_contour()</code> .
contour_col	colour specification for contour lines.
n_contour	numeric; the number of contour bins. Will result in <code>n_contour - 1</code> contour lines being drawn. See <code>ggplot2::geom_contour()</code> .
ci_alpha	numeric; alpha transparency for confidence or simultaneous interval.
ci_col	colour specification for the confidence/credible intervals band. Affects the fill of the interval.
smooth_col	colour specification for the smooth line.
resid_col	colour specification for the partial residuals.
partial_match	logical; should smooths be selected by partial matches with <code>select</code> ? If TRUE, <code>select</code> can only be a single string to match against.
discrete_colour, discrete_fill, continuous_colour, continuous_fill	suitable scales for the types of data.
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the <code>angle</code> argument of <code>ggplot2::guide_axis()</code> .
ylim	numeric; vector of y axis limits to use all <i>all</i> panels drawn.
projection	character; projection to use, see <code>ggplot2::coord_map()</code> for details.
orientation	an optional vector <code>c(latitude, longitude, rotation)</code> which describes where the "North Pole" should be when computing the projection. The third value is a clockwise rotation (in degrees), which defaults to the midrange of the longitude coordinates in the data. The default values for <code>orientation</code> therefore are <code>'c(20, 0, mean(range(longitude)))'</code> if this is not specified by the user. See links in <code>ggplot2::coord_map()</code> for more information.
...	additional arguments passed to <code>patchwork::wrap_plots()</code> .

## Examples

```
load_mgcv()
# example data
df <- data_sim("eg1", seed = 21)
# fit GAM
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")
# plot all of the estimated smooths
sm <- smooth_estimates(m)
draw(sm)
# evaluate smooth of `x2`
sm <- smooth_estimates(m, smooth = "s(x2)")
# plot it
draw(sm)

# customising some plot elements
draw(sm, ci_col = "steelblue", smooth_col = "forestgreen", ci_alpha = 0.3)
```

```
# Add a constant to the plotted smooth
draw(sm, constant = coef(m)[1])
```

---

```
draw.smooth_samples    Plot posterior smooths
```

---

## Description

Plot posterior smooths

## Usage

```
## S3 method for class 'smooth_samples'
draw(
  object,
  select = NULL,
  n_samples = NULL,
  seed = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  alpha = 1,
  colour = "black",
  contour = FALSE,
  contour_col = "black",
  n_contour = NULL,
  scales = c("free", "fixed"),
  rug = TRUE,
  partial_match = FALSE,
  angle = NULL,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  ...
)
```

## Arguments

object	a fitted GAM, the result of a call to <code>mgcv::gam()</code> .
select	character, logical, or numeric; which smooths to plot. If NULL, the default, then all model smooths are drawn. Numeric <code>select</code> indexes the smooths in the order they are specified in the formula and stored in <code>object</code> . Character <code>select</code> matches the labels for smooths as shown for example in the output from <code>summary(object)</code> . Logical <code>select</code> operates as per numeric <code>select</code> in the order that smooths are stored.

n_samples	numeric; if not NULL, sample n_samples from the posterior draws for plotting.
seed	numeric; random seed to be used to if sampling draws.
xlab	character or expression; the label for the x axis. If not supplied, a suitable label will be generated from object.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated from object.
title	character or expression; the title for the plot. See <a href="#">ggplot2::labs()</a> .
subtitle	character or expression; the subtitle for the plot. See <a href="#">ggplot2::labs()</a> .
caption	character or expression; the plot caption. See <a href="#">ggplot2::labs()</a> .
alpha	numeric; alpha transparency for confidence or simultaneous interval.
colour	The colour to use to draw the posterior smooths. Passed to <a href="#">ggplot2::geom_line()</a> as argument colour.
contour	logical; should contour lines be added to smooth surfaces?
contour_col	colour specification for contour lines.
n_contour	numeric; the number of contour bins. Will result in n_contour - 1 contour lines being drawn. See <a href="#">ggplot2::geom_contour()</a> .
scales	character; should all univariate smooths be plotted with the same y-axis scale? If scales = "free", the default, each univariate smooth has its own y-axis scale. If scales = "fixed", a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.
rug	logical; draw a rug plot at the botom of each plot for 1-D smooths or plot locations of data for higher dimensions.
partial_match	logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.
angle	numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of <a href="#">ggplot2::guide_axis()</a> .
ncol, nrow	numeric; the numbers of rows and columns over which to spread the plots
guides	character; one of "keep" (the default), "collect", or "auto". Passed to <a href="#">patchwork::plot_layout()</a>
...	arguments to be passed to <a href="#">patchwork::wrap_plots()</a> .

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()
dat1 <- data_sim("eg1", n = 400, dist = "normal", scale = 1, seed = 1)
## a single smooth GAM
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat1, method = "REML")
## posterior smooths from m1
sm1 <- smooth_samples(m1, n = 15, seed = 23478)
## plot
draw(sm1, alpha = 0.7)
```

```

## plot only 5 randomly sampled draws
draw(sm1, n_samples = 5, alpha = 0.7)

## A factor-by smooth example
dat2 <- data_sim("eg4", n = 400, dist = "normal", scale = 1, seed = 1)
## a multi-smooth GAM with a factor-by smooth
m2 <- gam(y ~ fac + s(x2, by = fac) + s(x0), data = dat2, method = "REML")
## posterior smooths from m1
sm2 <- smooth_samples(m2, n = 15, seed = 23478)
## plot, this time selecting only the factor-by smooth
draw(sm2, select = "s(x2)", partial_match = TRUE, alpha = 0.7)

## A 2D smooth example
dat3 <- data_sim("eg2", n = 400, dist = "normal", scale = 1, seed = 1)
## fit a 2D smooth
m3 <- gam(y ~ te(x, z), data = dat3, method = "REML")
## get samples
sm3 <- smooth_samples(m3, n = 10)
## plot just 6 of the draws, with contour line overlays
draw(sm3, n_samples = 6, contour = TRUE, seed = 42)

```

---

edf

*Effective degrees of freedom for smooths and GAMs*


---

## Description

Extracts the effective degrees of freedom (EDF) for model smooth terms or overall EDF for fitted GAMs

## Usage

```

edf(object, ...)

## S3 method for class 'gam'
edf(
  object,
  smooth = NULL,
  type = c("default", "unconditional", "alternative"),
  ...
)

model_edf(object, ..., type = c("default", "unconditional", "alternative"))

```

## Arguments

**object** a fitted model from which to extract smooth-specific EDFs.  
**...** arguments passed to methods.

smooth	character; a vector of smooth terms whose EDFs will be extracted. If NULL, the default, EDFs for all smooths will be returned.
type	character: which type of EDF to return. "default" returns the standard EDF; "unconditional" selects the EDF corrected for smoothness parameter selection, if available; "alternative" returns the alternative formulation for EDF from Wood (2017, pp. 252)

## Details

Multiple formulations for the effective degrees of freedom are available. The additional uncertainty due to selection of smoothness parameters can be taken into account when computing the EDF of smooths. This form of the EDF is available with type = "unconditional".

Wood (2017; pp. 252) describes an alternative EDF for the model

$$\text{EDF} = 2\text{tr}(\mathbf{F}) - \text{tr}(\mathbf{F}\mathbf{F}),$$

where  $\text{tr}$  is the matrix trace and  $\mathbf{F}$  is a matrix mapping un-penalized coefficient estimates to the penalized coefficient estimates. The trace of  $\mathbf{F}$  is effectively the average shrinkage of the coefficients multiplied by the number of coefficients (Wood, 2017). Smooth-specific EDFs then are obtained by summing up the relevant elements of  $\text{diag}(2\mathbf{F} - \mathbf{F}\mathbf{F})$ .

## Examples

```
load_mgcv()

df <- data_sim("eg1", n = 400, seed = 42)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

# extract the EDFs for all smooths
edf(m)

# or selected smooths
edf(m, smooth = c("s(x0)", "s(x2)"))

# accounting for smoothness parameter uncertainty
edf(m, type = "unconditional")

# over EDF of the model, including the intercept
model_edf(m)

# can get model EDF for multiple models
m2 <- gam(y ~ s(x0) + s(x1) + s(x3), data = df, method = "REML")
model_edf(m, m2)
```

---

```
evaluate_parametric_term
    Evaluate parametric model terms
```

---

### Description

**[Deprecated]** Returns values of parametric model terms at values of factor terms and over a grid of covariate values for linear parametric terms. This function is now deprecated in favour of [parametric\\_effects\(\)](#).

### Usage

```
evaluate_parametric_term(object, ...)

## S3 method for class 'gam'
evaluate_parametric_term(object, term, unconditional = FALSE, ...)
```

### Arguments

object	an object of class "gam" or "gamm".
...	arguments passed to other methods.
term	character; which parametric term whose effects are evaluated
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.

---

```
evaluate_smooth    Evaluate a smooth
```

---

### Description

**[Deprecated]** Evaluate a smooth at a grid of evenly spaced value over the range of the covariate associated with the smooth. Alternatively, a set of points at which the smooth should be evaluated can be supplied.

### Usage

```
evaluate_smooth(object, ...)

## S3 method for class 'gam'
evaluate_smooth(
  object,
  smooth,
  n = 100,
  newdata = NULL,
```

```

    unconditional = FALSE,
    overall_uncertainty = TRUE,
    dist = 0.1,
    ...
)

## S3 method for class 'gamm'
evaluate_smooth(object, ...)

## S3 method for class 'list'
evaluate_smooth(object, ...)

```

### Arguments

object	an object of class "gam" or "gamm".
...	arguments passed to other methods.
smooth	character; a single smooth to evaluate.
n	numeric; the number of points over the range of the covariate at which to evaluate the smooth.
newdata	a vector or data frame of points at which to evaluate the smooth.
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.
overall_uncertainty	logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?
dist	numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and dist is a distance within the unit square. See <code>mgcv::exclude.too.far()</code> for further details.

### Details

**[Deprecated]** `evaluate_smooth()` is deprecated in favour of `smooth_estimates()`, which provides a cleaner way to evaluate a smooth over a range of covariate values. `smooth_estimates()` can handle a much wider range of models than `evaluate_smooth()` is capable of and `smooth_estimates()` is much easier to extend to handle new smooth types.

Most code that uses `evaluate_smooth()` should work simply by changing the function call to `smooth_estimates()`. However, there are some differences:

- the `newdata` argument becomes `data`

Consider `evaluate_smooth()` to be *soft*-deprecated; its use is discouraged and it may be removed at a later date if it becomes difficult to maintain the current functionality, but there are no intentions of removing it from *gratia* unless that situation arises.

### Value

A data frame, which is of class "evaluated\_1d\_smooth" or "evaluated\_2d\_smooth", which inherit from classes "evaluated\_smooth" and "data.frame".

**Examples**

```

load_mgcv()

dat <- data_sim("eg1", n = 500, dist = "normal", scale = 1, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

evaluate_smooth(m1, "s(x1)")

## 2d example
dat <- data_sim("eg2", n = 1000, dist = "normal", scale = 1, seed = 2)
m2 <- gam(y ~ s(x, z, k = 30), data = dat, method = "REML")

evaluate_smooth(m2, "s(x,z)", n = 50)

```

---

eval\_smooth

*S3 methods to evaluate individual smooths*


---

**Description**

S3 methods to evaluate individual smooths

**Usage**

```

eval_smooth(smooth, ...)

## S3 method for class 'mgcv.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  ...
)

## S3 method for class 'fs.interaction'
eval_smooth(
  smooth,
  model,
  n = 100,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,

```



```
    ...
  )

## S3 method for class 'sz.interaction'
eval_smooth(
  smooth,
  model,
  n = 100,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  ...
)

## S3 method for class 'random.effect'
eval_smooth(
  smooth,
  model,
  n = 100,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  ...
)

## S3 method for class 'mrf.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  ...
)

## S3 method for class 't2.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  ...
)
```

```

)

## S3 method for class 'tensor.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  ...
)

```

### Arguments

smooth	currently an object that inherits from class <code>mgcv.smooth</code> .
...	arguments assed to other methods
model	a fitted model; currently only <code>mgcv::gam()</code> and <code>mgcv::bam()</code> models are supported.
n	numeric; the number of points over the range of the covariate at which to evaluate the smooth.
n_3d, n_4d	numeric; the number of points over the range of last covariate in a 3D or 4D smooth. The default is NULL which achieves the standard behaviour of using n points over the range of all covariate, resulting in $n^d$ evaluation points, where d is the dimension of the smooth. For $d > 2$ this can result in very many evaluation points and slow performance. For smooths of $d > 4$ , the value of n_4d will be used for all dimensions $> 4$ , unless this is NULL, in which case the default behaviour (using n for all dimensions) will be observed.
data	an optional data frame of values to evaluate smooth at.
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.
overall_uncertainty	logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?
dist	numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and dist is a distance within the unit square. See <code>mgcv::exclude.too.far()</code> for further details.

---

evenly	<i>Create a sequence of evenly-spaced values</i>
--------	--

---

**Description**

For a continuous vector  $x$ , `evenly` and `seq_min_max()` create a sequence of  $n$  evenly-spaced values over the range  $\min(x) - \max(x)$ . For a factor  $x$ , the function returns `levels(x)`.

**Usage**

```
evenly(x, n = 100, by = NULL)
```

```
seq_min_max(x, n, by = NULL)
```

**Arguments**

<code>x</code>	numeric; vector over which evenly-spaced values are returned
<code>n</code>	numeric; the number of evenly-spaced values to return. A default of 100 is used for convenience as that what is typically used when evaluating a smooth.
<code>by</code>	numeric; the increment of the sequence. If specified, argument <code>n</code> is ignored and the sequence returned will be from $\min(x)$ to $\max(x)$ in increments of <code>by</code> .

**Value**

A numeric vector of length  $n$ .

**Examples**

```
x <- rnorm(10)
n <- 10L
evenly(x, n = n)
```

---

factor_combos	<i>All combinations of factor levels</i>
---------------	--

---

**Description**

All combinations of factor levels

**Usage**

```
factor_combos(object, ...)
```

```
## S3 method for class 'gam'
factor_combos(object, vars = everything(), complete = TRUE, ...)
```

**Arguments**

object	a fitted model object.
...	arguments passed to methods.
vars	terms to include or exclude from the returned object. Uses tidyselect principles.
complete	logical; should all combinations of factor levels be returned? If FALSE, only those combinations of levels observed in the model are retained.

---

family.gam	<i>Extract family objects from models</i>
------------	---

---

**Description**

Provides a `stats::family()` method for a range of GAM objects.

**Usage**

```
## S3 method for class 'gam'
family(object, ...)

## S3 method for class 'gamm'
family(object, ...)

## S3 method for class 'bam'
family(object, ...)

## S3 method for class 'list'
family(object, ...)
```

**Arguments**

object	a fitted model. Models fitted by <code>mgcv::gam()</code> , <code>mgcv::bam()</code> , <code>mgcv::gamm()</code> , and <code>gamm4::gamm4()</code> are currently supported.
...	arguments passed to other methods.

---

family_name	<i>Name of family used to fit model</i>
-------------	---

---

**Description**

Extracts the name of the family used to fit the supplied model.

**Usage**

```
family_name(object, ...)
```

**Arguments**

object            an R object.  
 ...                arguments passed to other methods.

**Value**

A character vector containing the family name.

---

family_type	<i>Extracts the type of family in a consistent way</i>
-------------	--

---

**Description**

Extracts the type of family in a consistent way

**Usage**

```
family_type(object, ...)

## S3 method for class 'family'
family_type(object, ...)

## Default S3 method:
family_type(object, ...)
```

**Arguments**

object            an R object. Currently `family()` objects and anything with a `family()` method.  
 ...                arguments passed to other methods.

---

fitted_samples	<i>Draw fitted values from the posterior distribution</i>
----------------	---

---

**Description**

Expectations (fitted values) of the response drawn from the posterior distribution of fitted model using a Gaussian approximation to the posterior.

**Usage**

```
fitted_samples(model, ...)

## S3 method for class 'gam'
fitted_samples(
  model,
  n = 1,
  data = newdata,
  seed,
  scale = c("response", "linear_predictor"),
  method = c("gaussian", "mh", "inla"),
  freq = FALSE,
  unconditional = FALSE,
  ncores = 1L,
  ...,
  newdata = NULL
)
```

**Arguments**

model	a fitted model of the supported types
...	arguments passed to other methods. For <code>fitted_samples()</code> , these are passed on to <code>predict.gam()</code> .
n	numeric; the number of posterior samples to return.
data	data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for data, if available in model.
seed	numeric; a random seed for the simulations.
scale	character;
method	character; the method used to generate samples from the posterior distribution of the model. "gaussian", the default, uses a Gaussian approximation to the posterior. "mh" uses a simple Metropolis Hastings sampler, while "inla" uses a variant of Integrated Nested Laplace Approximation due to Wood (2019). Currently, the only available option is "gaussian".
freq	logical; TRUE to use the frequentist covariance matrix of the parameter estimators, FALSE to use the Bayesian posterior covariance matrix of the parameters.
unconditional	logical; if TRUE (and <code>freq == FALSE</code> ) then the Bayesian smoothing parameter uncertainty corrected covariance matrix is used, if available.
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
newdata	Deprecated: use data instead.

**Value**

A tibble (data frame) with 3 columns containing the posterior predicted values in long format. The columns are

- row (integer) the row of data that each posterior draw relates to,
- draw (integer) an index, in range 1:n, indicating which draw each row relates to,
- response (numeric) the predicted response for the indicated row of data.

**Author(s)**

Gavin L. Simpson

**References**

Wood, S.N., (2020). Simplified integrated nested Laplace approximation. *Biometrika* **107**, 223–230. doi:10.1093/biomet/asz044

**Examples**

```
load_mgcv()

dat <- data_sim("eg1", n = 1000, dist = "normal", scale = 2, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

fs <- fitted_samples(m1, n = 5, seed = 42)

fs
```

---

fitted\_values

*Generate fitted values from a estimated GAM*

---

**Description**

Generate fitted values from a estimated GAM

**Usage**

```
fitted_values(object, ...)
```

```
## S3 method for class 'gam'
```

```
fitted_values(  
  object,  
  data = NULL,  
  scale = c("response", "link", "linear predictor"),  
  ci_level = 0.95,  
  ...  
)
```

**Arguments**

object	a fitted model. Currently only models fitted by <code>mgcv::gam()</code> and <code>mgcv::bam()</code> are supported.
...	arguments passed to <code>mgcv::predict.gam()</code> . Note that <code>type</code> , <code>newdata</code> , and <code>se.fit</code> are already used and passed on to <code>mgcv::predict.gam()</code> .
data	optional data frame of covariate values for which fitted values are to be returned.
scale	character; what scale should the fitted values be returned on? "linear predictor" is a synonym for "link" if you prefer that terminology.
ci_level	numeric; a value between 0 and 1 indicating the coverage of the credible interval.

**Value**

A tibble (data frame) whose first  $m$  columns contain either the data used to fit the model (if data was NULL), or the variables supplied to data. Four further columns are added:

- fitted: the fitted values on the specified scale,
- se: the standard error of the fitted values (always on the *link* scale),
- lower, upper: the limits of the credible interval on the fitted values, on the specified scale.

**Note**

Regardless of the scale on which the fitted values are returned, the se component of the returned object is on the *link* (*linear predictor*) scale, not the response scale.

**Examples**

```
load_mgcv()

sim_df <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = sim_df, method = "REML")
fv <- fitted_values(m)
fv
```

---

fixef

*Extract fixed effects estimates*


---

**Description**

Extract fixed effects estimates

**Arguments**

object	a fitted GAM
...	arguments passed to other methods



---

fixef.gam	<i>Extract fixed effects estimates from a fitted GAM</i>
-----------	--

---

**Description**

Extract fixed effects estimates from a fitted GAM

**Usage**

```
## S3 method for class 'gam'  
fixef(object, ...)  
  
## S3 method for class 'gamm'  
fixef(object, ...)  
  
## S3 method for class 'lm'  
fixef(object, ...)  
  
## S3 method for class 'glm'  
fixef(object, ...)  
  
fixed_effects(object, ...)  
  
## Default S3 method:  
fixed_effects(object, ...)
```

**Arguments**

object	a fitted GAM
...	arguments passed to other methods

**Examples**

```
load_mgcv()  
  
# run example if lme4 is available  
if (require("lme4")) {  
  
  data(sleepstudy, package = "lme4")  
  m <- gam(Reaction ~ Days + s(Subject, bs = "re") +  
          s(Days, Subject, bs = "re"),  
          data = sleepstudy, method = "REML")  
  fixef(m)  
  
}
```

---

fix_offset	<i>Fix the names of a data frame containing an offset variable.</i>
------------	---

---

**Description**

Identifies which variable, if any, is the model offset, and fixed the name such that `offset(foo(var))` is converted to `var`, and possibly sets the values of that variable to `offset_val`.

**Usage**

```
fix_offset(model, newdata, offset_val = NULL)
```

**Arguments**

<code>model</code>	a fitted GAM.
<code>newdata</code>	data frame; new values at which to predict at.
<code>offset_val</code>	numeric, optional; if provided, then the offset variable in <code>newdata</code> is set to this constant value before returning <code>newdata</code>

**Value**

The original `newdata` is returned with fixed names and possibly modified offset variable.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()

df <- data_sim("eg1", n = 400, dist = "normal", seed = 2)
m <- gam(y ~ s(x0) + s(x1) + offset(x2), data = df, method = "REML")
names(model.frame(m))
names(fix_offset(m, model.frame(m), offset_val = 1L))
```

---

get_by_smooth	<i>Extract an factor-by smooth by name</i>
---------------	--

---

**Description**

Extract an factor-by smooth by name

**Usage**

```
get_by_smooth(object, term, level)
```

**Arguments**

object	a fitted GAM model object.
term	character; the name of a smooth term to extract.
level	character; which level of the factor to extract the smooth for.

**Value**

A single smooth object, or a list of smooths if several match the named term.

---

get_smooth	<i>Extract an mgcv smooth by name</i>
------------	---------------------------------------

---

**Description**

Extract an mgcv smooth by name

**Usage**

```
get_smooth(object, term)
```

**Arguments**

object	a fitted GAM model object.
term	character; the name of a smooth term to extract

**Value**

A single smooth object, or a list of smooths if several match the named term.

---

get_smooths_by_id	<i>Extract an mgcv smooth given its position in the model object</i>
-------------------	--

---

**Description**

Extract an mgcv smooth given its position in the model object

**Usage**

```
get_smooths_by_id(object, id)
```

**Arguments**

object	a fitted GAM model object.
id	numeric; the position of the smooth in the model object.

---

gss_vocab	<i>Data from the General Social Survey (GSS) from the National Opinion Research Center of the University of Chicago</i>
-----------	---

---

**Description**

A subset of the data from the `carData::GSSvocab` dataset from the `carData` package, containing observations from 2016 only.

**Format**

A data frame with 1858 rows and 3 variables:

- `vocab`: numeric; the number of words out of 10 correct on a vocabulary test.
- `nativeBorn`: factor; Was the respondent born in the US? A factor with levels `no` and `yes`.
- `ageGroup`: factor; grouped age of the respondent with levels `18-29`, `30-39`, `40-49`, `50-59`, and `60+.`

---

gw_f0	<i>Gu and Wabha test functions</i>
-------	------------------------------------

---

**Description**

Gu and Wabha test functions

**Usage**

```
gw_f0(x)
```

```
gw_f1(x)
```

```
gw_f2(x)
```

```
gw_f3(x)
```

**Arguments**

`x` numeric; vector of points to evaluate the function at, on interval (0,1)

**Examples**

```
x <- seq(0, 1, length = 6)
gw_f0(x)
gw_f1(x)
gw_f2(x)
gw_f3(x) # should be constant 0
```

---

has_theta	<i>Are additional parameters available for a GAM?</i>
-----------	---

---

**Description**

Are additional parameters available for a GAM?

**Usage**

```
has_theta(object)
```

**Arguments**

object            an R object, either a `family()` object or an object whose class has a `family()` method.

**Value**

A logical; TRUE if additional parameters available, FALSE otherwise.

**Examples**

```
load_mgcv()
df <- data_sim("eg1", dist = "poisson", seed = 42, scale = 1/5)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML",
         family = nb())
has_theta(m)
p <- theta(m)
```

---

is_by_smooth	<i>Tests for by variable smooths</i>
--------------	--------------------------------------

---

**Description**

Functions to check if a smooth is a by-variable one and to test of the type of by-variable smooth is a factor-smooth or a continuous-smooth interaction.

**Usage**

```
is_by_smooth(smooth)

is_factor_by_smooth(smooth)

is_continuous_by_smooth(smooth)

by_variable(smooth)

by_level(smooth)
```

**Arguments**

smooth            an object of class "mgcv.smooth"

**Value**

A logical vector.

**Author(s)**

Gavin L. Simpson

---

is_factor_term	<i>Is a model term a factor (categorical)?</i>
----------------	--

---

**Description**

Given the name (a term label) of a term in a model, identify if the term is a factor term or numeric. This is useful when considering interactions, where terms like fac1:fac2 or num1:fac1 may be requested by the user. Only for terms of the type fac1:fac2 will this function return TRUE.

**Usage**

```
is_factor_term(object, term, ...)

## S3 method for class 'terms'
is_factor_term(object, term, ...)

## S3 method for class 'gam'
is_factor_term(object, term, ...)

## S3 method for class 'bam'
is_factor_term(object, term, ...)

## S3 method for class 'gamm'
is_factor_term(object, term, ...)

## S3 method for class 'list'
is_factor_term(object, term, ...)
```

**Arguments**

object            an R object on which method dispatch is performed

term              character; the name of a model term, in the sense of attr(terms(object), "term.labels"). Currently not checked to see if the term exists in the model.

...               arguments passed to other methods.

**Value**

A logical: TRUE if and only if all variables involved in the term are factors, otherwise FALSE.

---

is_mgcv_smooth	<i>Check if objects are smooths or are a particular type of smooth</i>
----------------	--

---

**Description**

Check if objects are smooths or are a particular type of smooth

**Usage**

```
is_mgcv_smooth(smooth)
stop_if_not_mgcv_smooth(smooth)
check_is_mgcv_smooth(smooth)
is_mrf_smooth(smooth)
```

**Arguments**

smooth            an R object, typically a list

**Details**

Check if a smooth inherits from class "mgcv.smooth". `stop_if_not_mgcv_smooth()` is a wrapper around `is_mgcv_smooth()`, useful when programming for checking if the supplied object is one of mgcv's smooths, and throwing a consistent error if not. `check_is_mgcv_smooth()` is similar to `stop_if_not_mgcv_smooth()` but returns the result of `is_mgcv_smooth()` invisibly.

---

is_offset	<i>Is a model term an offset?</i>
-----------	-----------------------------------

---

**Description**

Given a character vector of model terms, checks to see which, if any, is the model offset.

**Usage**

```
is_offset(terms)
```

**Arguments**

terms            character vector of model terms.

**Value**

A logical vector of the same length as terms.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()
df <- data_sim("eg1", n = 400, dist = "normal")
m <- gam(y ~ s(x0) + s(x1) + offset(x0), data = df, method = "REML")
nm <- names(model.frame(m))
nm
is_offset(nm)
```

---

link

*Extract link and inverse link functions from models*

---

**Description**

Returns the link or its inverse from an estimated model, and provides a simple way to extract these functions from complex models with multiple links, such as location scale models.

**Usage**

```
link(object, ...)
```

## S3 method for class 'family'

```
link(object, parameter = NULL, which_eta = NULL, ...)
```

## S3 method for class 'gam'

```
link(object, parameter = NULL, which_eta = NULL, ...)
```

## S3 method for class 'bam'

```
link(object, parameter = NULL, which_eta = NULL, ...)
```

## S3 method for class 'gamm'

```
link(object, ...)
```

## S3 method for class 'glm'

```
link(object, ...)
```

## S3 method for class 'list'

```
link(object, ...)
```

```
inv_link(object, ...)
```



```

## S3 method for class 'family'
inv_link(object, parameter = NULL, which_eta = NULL, ...)

## S3 method for class 'gam'
inv_link(object, parameter = NULL, which_eta = NULL, ...)

## S3 method for class 'bam'
inv_link(object, parameter = NULL, which_eta = NULL, ...)

## S3 method for class 'gamm'
inv_link(object, ...)

## S3 method for class 'list'
inv_link(object, ...)

## S3 method for class 'glm'
inv_link(object, ...)

extract_link(family, ...)

## S3 method for class 'family'
extract_link(family, inverse = FALSE, ...)

## S3 method for class 'general.family'
extract_link(family, parameter, inverse = FALSE, which_eta = NULL, ...)

```

### Arguments

object	a family object or a fitted model from which to extract the family object. Models fitted by <code>stats::glm()</code> , <code>mgcv::gam()</code> , <code>mgcv::bam()</code> , <code>mgcv::gamm()</code> , and <code>gamm4::gamm4()</code> are currently supported.
...	arguments passed to other methods.
parameter	character; which parameter of the distribution. Usually "location" but "scale" and "shape" may be provided for location scale models. Other options include "mu" as a synonym for "location", "sigma" for the scale parameter in <code>mgcv::gauss()</code> , "pi" for the zero-inflation term in <code>mgcv::ziplss()</code> , "power" for the <code>mgcv::twlss()</code> power parameter, "xi", the shape parameter for <code>mgcv::gevlss()</code> , "epsilon" or "skewness" for the skewness and "delta" or "kurtosis" for the kurtosis parameter for <code>mgcv::shash()</code> , or "theta" for the scale parameter of <code>mgcv::gammals()</code> .
which_eta	numeric; the linear predictor to extract for families <code>mgcv::mvn()</code> and <code>mgcv::multinom()</code> .
family	a family object, the result of a call to <code>family()</code> .
inverse	logical; return the inverse of the link function?

### Author(s)

Gavin L. Simpson

**Examples**

```

load_mgcv()

link(gaussian())
link(nb())

inv_link(nb())

dat <- data_sim("eg1", seed = 4234)
mod <- gam(list(y ~ s(x0) + s(x1) + s(x2) + s(x3), ~ 1), data = dat,
              family = gaulss)

link(mod, parameter = "scale")
inv_link(mod, parameter = "scale")

## Works with `family` objects too
link(shash(), parameter = "skewness")

```

---

load_mgcv	<i>Load mgcv quietly</i>
-----------	--------------------------

---

**Description**

Simple function that loads the *mgcv* package whilst suppressing the startup messages that it prints to the console.

**Usage**

```
load_mgcv()
```

**Value**

Returns a logical vectors invisibly, indicating whether the package was loaded or not.

---

lp_matrix	<i>Return the linear prediction matrix of a fitted GAM</i>
-----------	--

---

**Description**

lp\_matrix() is a wrapper to predict(..., type = "lpmatrix") for returning the linear predictor matrix for the model training data (when data = NULL), or user-specified data values supplied via data.

**Usage**

```
lp_matrix(model, ...)

## S3 method for class 'gam'
lp_matrix(model, data = NULL, ...)
```

**Arguments**

model	a fitted model
...	arguments passed to other methods and predict methods including <code>mgcv::predict.gam()</code> and <code>mgcv::predict.bam()</code>
data	a data frame of values at which to return the linear prediction matrix.

**Details**

The linear prediction matrix  $\mathbf{X}_p$  is a matrix that maps values of parameters  $\hat{\beta}_p$  to values on the linear predictor of the model  $\hat{\eta}_p = \mathbf{X}_p \hat{\beta}_p$ .  $\mathbf{X}_p$  is the model matrix where spline covariates have been replaced by the values of the basis functions evaluated at the respective covariates. Parametric covariates are also included.

**Value**

The linear prediction matrix is returned as a matrix. The object returned is of class "lp\_matrix", which inherits from classes "matrix" and "array". The special class allows the printing of the matrix to be controlled, which we do by printing the matrix as a tibble.

**Examples**

```
load_mgcv()

df <- data_sim("eg1", seed = 1)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df)

# linear prediction matrix for observed data
xp <- lp_matrix(m)
## IGNORE_RDIFF_BEGIN
xp
## IGNORE_RDIFF_END

# the object `xp` *is* a matrix
class(xp)
# but we print like a tibble to avoid spamming the R console

# linear predictor matrix for new data set
ds <- data_slice(m, x2 = evenly(x2))
xp <- lp_matrix(m, data = ds)
## IGNORE_RDIFF_BEGIN
xp
## IGNORE_RDIFF_END
```

---

model_concurvity	<i>Concurvity of an estimated GAM</i>
------------------	---------------------------------------

---

**Description**

Concurvity of an estimated GAM

**Usage**

```
model_concurvity(model, ...)

## S3 method for class 'gam'
model_concurvity(
  model,
  terms = everything(),
  type = c("all", "estimate", "observed", "worst"),
  pairwise = FALSE,
  ...
)

concrvity(
  model,
  terms = everything(),
  type = c("all", "estimate", "observed", "worst"),
  pairwise = FALSE,
  ...
)
```

**Arguments**

model	a fitted GAM. Currently only objects of class "gam" are supported
...	arguments passed to other methods.
terms	currently ignored
type	character;
pairwise	logical; extract pairwise concurvity of model terms?

**Examples**

```
## simulate data with concurvity...
library("tibble")
load_mgcv()
set.seed(8)
n <- 200
df <- tibble(t = sort(runif(n)),
             x = gw_f2(t) + rnorm(n) * 3,
             y = sin(4 * pi * t) + exp(x / 20) + rnorm(n) * 0.3)
```

```
## fit model
m <- gam(y ~ s(t, k = 15) + s(x, k = 15), data = df, method = "REML")

## overall concurvity
o_conc <- concurvity(m)
draw(o_conc)

## pairwise concurvity
p_conc <- concurvity(m, pairwise = TRUE)
draw(p_conc)
```

---

model\_vars

*List the variables involved in a model fitted with a formula*

---

## Description

List the variables involved in a model fitted with a formula

## Usage

```
model_vars(model, ...)

## S3 method for class 'gam'
model_vars(model, ...)

## Default S3 method:
model_vars(model, ...)

## S3 method for class 'bam'
model_vars(model, ...)

## S3 method for class 'gamm'
model_vars(model, ...)

## S3 method for class 'gamm4'
model_vars(model, ...)

## S3 method for class 'list'
model_vars(model, ...)
```

## Arguments

model	a fitted model object with a \$pred.formula, \$terms component or a "terms" attribute
...	Arguments passed to other methods. Currently ignored.

**Examples**

```
load_mgcv()

# simulate some Gaussian data
df <- data_sim("eg1", n = 50, seed = 2)

# fit a GAM with 1 smooth and 1 linear term
m1 <- gam(y ~ s(x2, k = 7) + x1, data = df, method = "REML")
model_vars(m1)

# fit a lm with two linear terms
m2 <- lm(y ~ x2 + x1, data = df)
model_vars(m2)
```

---

nb\_theta

*Negative binomial parameter theta*


---

**Description**

Negative binomial parameter theta

**Usage**

```
nb_theta(model)

## S3 method for class 'gam'
nb_theta(model)
```

**Arguments**

model            a fitted model.

**Value**

A numeric vector of length 1 containing the estimated value of theta.

**Methods (by class)**

- nb\_theta(gam): Method for class "gam"

**Examples**

```
load_mgcv()
df <- data_sim("eg1", n = 500, dist = "poisson", scale = 0.1, seed = 6)

m <- gam(y ~ s(x0, bs = "cr") + s(x1, bs = "cr") + s(x2, bs = "cr") +
         s(x3, bs = "cr"), family = nb, data = df, method = "REML")
## IGNORE_RDIFF_BEGIN
nb_theta(m)
## IGNORE_RDIFF_END
```

---

n_smooths	<i>How many smooths in a fitted model</i>
-----------	---

---

**Description**

How many smooths in a fitted model

**Usage**

```
n_smooths(object)

## Default S3 method:
n_smooths(object)

## S3 method for class 'gam'
n_smooths(object)

## S3 method for class 'gamm'
n_smooths(object)

## S3 method for class 'bam'
n_smooths(object)
```

**Arguments**

object            a fitted GAM or related model. Typically the result of a call to `mgcv::gam()`, `mgcv::bam()`, or `mgcv::gamm()`.

---

observed_fitted_plot	<i>Plot of fitted against observed response values</i>
----------------------	--

---

**Description**

Plot of fitted against observed response values

**Usage**

```
observed_fitted_plot(
  model,
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  point_col = "black",
  point_alpha = 1
)
```

**Arguments**

model	a fitted model. Currently only class "gam".
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
xlab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See <code>ggplot2::labs()</code> .
subtitle	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> .
caption	character or expression; the plot caption. See <code>ggplot2::labs()</code> .
point_col	colour used to draw points in the plots. See <code>graphics::par()</code> section <b>Color Specification</b> . This is passed to the individual plotting functions, and therefore affects the points of all plots.
point_alpha	numeric; alpha transparency for points in plots.

---

 overview

*Provides an overview of a model and the terms in that model*


---

**Description**

Provides an overview of a model and the terms in that model

**Usage**

```
overview(model, ...)

## S3 method for class 'gam'
overview(
  model,
  parametric = TRUE,
  random_effects = TRUE,
  dispersion = NULL,
  frequentist = FALSE,
  accuracy = 0.001,
  ...
)
```

**Arguments**

model	a fitted model object to overview.
...	arguments passed to other methods.
parametric	logical; include the model parametric terms in the overview?
random_effects	tests of fully penalized smooth terms (those with a zero-dimensional null space, e.g. random effects) are computationally expensive and for large data sets producing these p values can take a very long time. If <code>random_effects = FALSE</code> , the tests of the expensive terms will be skipped.



dispersion	numeric; a known value for the dispersion parameter. The default NULL implies that the estimated value or the default value (1 for the Poisson distribution for example) where this is specified is used instead.
frequentist	logical; by default the Bayesian estimated covariance matrix of the parameter estimates is used to calculate p values for parametric terms. If frequentist = FALSE, the frequentist covariance matrix of the parameter estimates is used.
accuracy	numeric; accuracy with which to report p values, with p values below this value displayed as "< accuracy".

## Examples

```
load_mgcv()

df <- data_sim(n = 400, seed = 2)
m <- gam(y ~ x3 + s(x0) + s(x1, bs = "bs") + s(x2, bs = "ts"),
         data = df, method = "REML")
overview(m)
```

---

parametric\_effects      *Estimated values for parametric model terms*

---

## Description

Estimated values for parametric model terms

## Usage

```
parametric_effects(object, ...)

## S3 method for class 'gam'
parametric_effects(
  object,
  terms = NULL,
  unconditional = FALSE,
  unnest = TRUE,
  ci_level = 0.95,
  envir = environment(formula(object)),
  ...
)
```

## Arguments

object	a fitted model object.
...	arguments passed to other methods.

terms	character; which model parametric terms should be drawn? The Default of NULL will plot all parametric terms that can be drawn.
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.
unnest	logical; unnest the smooth objects?
ci_level	numeric; the coverage required for the confidence interval. Currently ignored.
envir	an environment to look up the data within.

---

parametric\_terms      *Names of any parametric terms in a GAM*

---

### Description

Names of any parametric terms in a GAM

### Usage

```
parametric_terms(model, ...)

## Default S3 method:
parametric_terms(model, ...)

## S3 method for class 'gam'
parametric_terms(model, ...)
```

### Arguments

model	a fitted model.
...	arguments passed to other methods.

---

partial\_derivatives      *Partial derivatives of estimated multivariate smooths via finite differences*

---

### Description

Partial derivatives of estimated multivariate smooths via finite differences

**Usage**

```

partial_derivatives(object, ...)

## Default S3 method:
partial_derivatives(object, ...)

## S3 method for class 'gamm'
partial_derivatives(object, ...)

## S3 method for class 'gam'
partial_derivatives(
  object,
  term,
  focal = NULL,
  data = newdata,
  order = 1L,
  type = c("forward", "backward", "central"),
  n = 200,
  eps = 1e-07,
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  level = 0.95,
  unconditional = FALSE,
  frequentist = FALSE,
  offset = NULL,
  ncores = 1,
  partial_match = FALSE,
  ...,
  newdata = NULL
)

```

**Arguments**

object	an R object to compute derivatives for.
...	arguments passed to other methods.
term	character; vector of one or more smooth terms for which derivatives are required. If missing, derivatives for all smooth terms will be returned. Can be a partial match to a smooth term; see argument <code>partial_match</code> below.
focal	character; name of the focal variable. The partial derivative of the estimated smooth with respect to this variable will be returned. All other variables involved in the smooth will be held at constant. This can be missing if supplying data, in which case, the focal variable will be identified as the one variable that is not constant.
data	a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths. If supplied, all but one variable must be held at a constant value.
order	numeric; the order of derivative.

type	character; the type of finite difference used. One of "forward", "backward", or "central".
n	numeric; the number of points to evaluate the derivative at.
eps	numeric; the finite difference.
interval	character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.
n_sim	integer; the number of simulations used in computing the simultaneous intervals.
level	numeric; $0 < \text{level} < 1$ ; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.
unconditional	logical; use smoothness selection-corrected Bayesian covariance matrix?
frequentist	logical; use the frequentist covariance matrix?
offset	numeric; a value to use for any offset term
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
partial_match	logical; should smooths be selected by partial matches with term? If TRUE, term can only be a single string to match against.
newdata	Deprecated: use data instead.

### Value

A tibble, currently with the following variables:

- smooth: the smooth each row refers to,
- var: the name of the variable for which the partial derivative was evaluated,
- data: values of var at which the derivative was evaluated,
- partial\_deriv: the estimated partial derivative,
- se: the standard error of the estimated partial derivative,
- crit: the critical value such that  $\text{derivative} \pm (\text{crit} * \text{se})$  gives the upper and lower bounds of the requested confidence or simultaneous interval (given level),
- lower: the lower bound of the confidence or simultaneous interval,
- upper: the upper bound of the confidence or simultaneous interval.

### Note

`partial_derivatives()` will ignore any random effect smooths it encounters in object.

### Author(s)

Gavin L. Simpson

**Examples**

```

library("ggplot2")
library("patchwork")
load_mgcv()

df <- data_sim("eg2", n = 2000, dist = "normal", scale = 0.5, seed = 42)

# fit the GAM (note: for execution time reasons, k is set artificially low)
m <- gam(y ~ te(x, z, k = c(5, 5)), data = df, method = "REML")

# data slice through te(x,z) holding z == 0.4
ds <- data_slice(m, x = evenly(x, n = 100), z = 0.4)

# evaluate te(x,z) at values of x & z
sm <- smooth_estimates(m, smooth = "te(x,z)", data = ds) |>
  add_confint()

# partial derivatives
pd_x <- partial_derivatives(m, data = ds, type = "central", focal = "x")

# draw te(x,z)
p1 <- draw(m, rug = FALSE) &
  geom_hline(yintercept = 0.4, linewidth = 1)
p1

# draw te(x,z) along slice
cap <- expression(z == 0.4)
p2 <- sm |>
  ggplot(aes(x = x, y = est)) +
  geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci), alpha = 0.2) +
  geom_line() +
  labs(x = "x", y = "Partial effect", title = "te(x,z)",
       caption = cap)
p2

# draw partial derivs
p3 <- pd_x |>
  draw() +
  labs(caption = cap)
p3

# draw all three panels
p1 + p2 + p3 + plot_layout(ncol = 3)

```

**Description**

Partial residuals

**Usage**

```
partial_residuals(object, ...)

## S3 method for class 'gam'
partial_residuals(object, select = NULL, partial_match = FALSE, ...)
```

**Arguments**

object	an R object, typically a model. Currently only objects of class "gam" (or that inherit from that class) are supported.
...	arguments passed to other methods.
select	character, logical, or numeric; which smooths to plot. If NULL, the default, then all model smooths are drawn. Numeric select indexes the smooths in the order they are specified in the formula and stored in object. Character select matches the labels for smooths as shown for example in the output from <code>summary(object)</code> . Logical select operates as per numeric select in the order that smooths are stored.
partial_match	logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.

**Examples**

```
## load mgcv
load_mgcv()

## example data - Gu & Wabha four term model
df <- data_sim("eg1", n = 400, seed = 42)
## fit the model
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = 'REML')

## extract partial residuals
partial_residuals(m)

## and for a select term
partial_residuals(m, select = "s(x2)")

## or with partial matching
partial_residuals(m, select = "x", partial_match = TRUE) # returns all
```

---

penalty *Extract and tidy penalty matrices*

---

## Description

Extract and tidy penalty matrices

## Usage

```
penalty(object, ...)

## S3 method for class 'gam'
penalty(object, smooth = NULL, rescale = FALSE, ...)

## S3 method for class 'mgcv.smooth'
penalty(object, rescale = FALSE, ...)

## S3 method for class 'tensor.smooth'
penalty(object, margins = FALSE, ...)

## S3 method for class 't2.smooth'
penalty(object, margins = FALSE, ...)

## S3 method for class 're.smooth.spec'
penalty(object, data, ...)
```

## Arguments

object	a fitted GAM or a smooth.
...	additional arguments passed to methods.
smooth	character; vector of smooths to extract penalty matrices for. If NULL, penalty matrices for all smooths in object are extracted.
rescale	logical; by default, <i>mgcv</i> will scale the penalty matrix for better performance in <code>mgcv::gamm()</code> . If <code>rescale</code> is TRUE, this scaling will be undone to put the penalty matrix back on the original scale.
margins	logical; extract the penalty matrices for the tensor product or the marginal smooths of the tensor product?
data	data frame; a data frame of values for terms mentioned in the smooth specification.

## Value

A 'tibble' (data frame) of class `penalty_df` inheriting from `tbl_df`, with the following components:

- `smooth` - character; the label *mgcv* uses to refer to the smooth,

- `type` - character; the type of smooth,
- `penalty` - character; the label for the specific penalty. Some smooths have multiple penalty matrices, so the `penalty` component identifies the particular penalty matrix and uses the labelling that *mgcv* uses internally,
- `row` - character; a label of the form `fn` where `n` is an integer for the `n`th basis function, referencing the columns of the penalty matrix,
- `col` - character; a label of the form `fn` where `n` is an integer for the `n`th basis function, referencing the columns of the penalty matrix,
- `value` - double; the value of the penalty matrix for the combination of `row` and `col`,

### Note

The `print()` method uses `base::zapsmall()` to turn very small numbers into 0s for display purposes only; the underlying values of the penalty matrix or matrices are not changed.

For smooths that are subject to an eigendecomposition (e.g. the default thin plate regression splines, `bs = "tp"`), the signs of the eigenvectors are not defined and as such you can expect differences across systems in the penalties for such smooths that are system-, OS-, and CPU architecture-specific.

### Author(s)

Gavin L. Simpson

### Examples

```
load_mgcv()
dat <- data_sim("eg4", n = 400, seed = 42)
m <- gam(y ~ s(x0, bs = "cr") + s(x1, bs = "cr") +
         s(x2, by = fac, bs = "cr"),
         data = dat, method = "REML")

# penalties for all smooths
penalty(m)

# for a specific smooth
penalty(m, smooth = "s(x2):fac1")
```

---

posterior\_samples

*Draw samples from the posterior distribution of an estimated model*

---

### Description

Draw samples from the posterior distribution of an estimated model



**Usage**

```
posterior_samples(model, ...)

## S3 method for class 'gam'
posterior_samples(
  model,
  n,
  data = newdata,
  seed,
  scale = c("response", "linear_predictor"),
  freq = FALSE,
  unconditional = FALSE,
  weights = NULL,
  ncores = 1L,
  ...,
  newdata = NULL
)
```

**Arguments**

model	a fitted model of the supported types
...	arguments passed to other methods. For <code>fitted_samples()</code> , these are passed on to <code>predict.gam()</code> .
n	numeric; the number of posterior samples to return.
data	data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for data, if available in <code>model</code> .
seed	numeric; a random seed for the simulations.
scale	character;
freq	logical; TRUE to use the frequentist covariance matrix of the parameter estimators, FALSE to use the Bayesian posterior covariance matrix of the parameters.
unconditional	logical; if TRUE (and <code>freq == FALSE</code> ) then the Bayesian smoothing parameter uncertainty corrected covariance matrix is used, if available.
weights	numeric; a vector of prior weights. If data is null then defaults to <code>object[["prior.weights"]]</code> , otherwise a vector of ones.
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
newdata	Deprecated: use data instead.

**Value**

A tibble (data frame) with 3 columns containing the posterior predicted values in long format. The columns are

- row (integer) the row of data that each posterior draw relates to,

- draw (integer) an index, in range 1 : n, indicating which draw each row relates to,
- response (numeric) the predicted response for the indicated row of data.

**Author(s)**

Gavin L. Simpson

---

predicted_samples	<i>Draw new response values from the conditional distribution of the response</i>
-------------------	---

---

**Description**

Predicted values of the response (new response data) are drawn from the fitted model, created via `simulate()` (e.g. `simulate.gam()`) and returned in a tidy, long, format. These predicted values do not include the uncertainty in the estimated model; they are simply draws from the conditional distribution of the response.

**Usage**

```
predicted_samples(model, ...)
```

```
## S3 method for class 'gam'
predicted_samples(
  model,
  n = 1,
  data = newdata,
  seed = NULL,
  weights = NULL,
  ...,
  newdata = NULL
)
```

**Arguments**

model	a fitted model of the supported types
...	arguments passed to other methods. For <code>fitted_samples()</code> , these are passed on to <code>predict.gam()</code> .
n	numeric; the number of posterior samples to return.
data	data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for data, if available in model.
seed	numeric; a random seed for the simulations.
weights	numeric; a vector of prior weights. If data is null then defaults to <code>object[["prior.weights"]]</code> , otherwise a vector of ones.
newdata	Deprecated: use data instead.

**Value**

A tibble (data frame) with 3 columns containing the posterior predicted values in long format. The columns are

- row (integer) the row of data that each posterior draw relates to,
- draw (integer) an index, in range 1:n, indicating which draw each row relates to,
- response (numeric) the predicted response for the indicated row of data.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()

dat <- data_sim("eg1", n = 1000, dist = "normal", scale = 2, seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

predicted_samples(m, n = 5, seed = 42)

## Can pass arguments to predict.gam()

newd <- data.frame(x0 = runif(10), x1 = runif(10), x2 = runif(10),
                  x3 = runif(10))

## Exclude s(x2)
predicted_samples(m, n = 5, newd, exclude = "s(x2)", seed = 25)

## Exclude s(x1)
predicted_samples(m, n = 5, newd, exclude = "s(x1)", seed = 25)

## Select which terms --- result should be the same as previous
## but note that we have to include any parametric terms, including the
## constant term
predicted_samples(m, n = 5, newd, seed = 25,
                 terms = c("Intercept", "s(x0)", "s(x2)", "s(x3)"))
```

---

qq\_plot

*Quantile-quantile plot of model residuals*

---

**Description**

Quantile-quantile plot of model residuals

**Usage**

```

qq_plot(model, ...)

## Default S3 method:
qq_plot(model, ...)

## S3 method for class 'gam'
qq_plot(
  model,
  method = c("uniform", "simulate", "normal", "direct"),
  type = c("deviance", "response", "pearson"),
  n_uniform = 10,
  n_simulate = 50,
  level = 0.9,
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ci_col = "black",
  ci_alpha = 0.2,
  point_col = "black",
  point_alpha = 1,
  line_col = "red",
  ...
)

## S3 method for class 'glm'
qq_plot(model, ...)

## S3 method for class 'lm'
qq_plot(model, ...)

```

**Arguments**

model	a fitted model. Currently only class "gam".
...	arguments passed to other methods.
method	character; method used to generate theoretical quantiles. Note that method = "direct" is deprecated in favour of method = "uniform".
type	character; type of residuals to use. Only "deviance", "response", and "pearson" residuals are allowed.
n_uniform	numeric; number of times to randomize uniform quantiles in the direct computation method (method = "uniform").
n_simulate	numeric; number of data sets to simulate from the estimated model when using the simulation method (method = "simulate").
level	numeric; the coverage level for reference intervals. Must be strictly $0 < \text{level} < 1$ . Only used with method = "simulate".

ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
xlab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See <a href="#">ggplot2::labs()</a> .
subtitle	character or expression; the subtitle for the plot. See <a href="#">ggplot2::labs()</a> .
caption	character or expression; the plot caption. See <a href="#">ggplot2::labs()</a> .
ci_col, ci_alpha	fill colour and alpha transparency for the reference interval when method = "simulate".
point_col, point_alpha	colour and alpha transparency for points on the QQ plot.
line_col	colour used to draw the reference line.

### Note

The wording used in [mgcv::qq.gam\(\)](#) uses *direct* in reference to the simulated residuals method (method = "simulated"). To avoid confusion, method = "direct" is deprecated in favour of method = "uniform".

### Examples

```
load_mgcv()
## simulate binomial data...
dat <- data_sim("eg1", n = 200, dist = "binary", scale = .33, seed = 0)
p <- binomial()$linkinv(dat$f) # binomial p
n <- sample(c(1, 3), 200, replace = TRUE) # binomial n
dat <- transform(dat, y = rbinom(n, n, p), n = n)
m <- gam( y / n ~ s(x0) + s(x1) + s(x2) + s(x3),
          family = binomial, data = dat, weights = n,
          method = "REML")

## Q-Q plot; default using direct randomization of uniform quantiles
qq_plot(m)

## Alternatively use simulate new data from the model, which
## allows construction of reference intervals for the Q-Q plot
qq_plot(m, method = "simulate", point_col = "steelblue",
        point_alpha = 0.4)

## ... or use the usual normality assumption
qq_plot(m, method = "normal")
```

ref\_level *Return the reference or specific level of a factor*

---

**Description**

Extracts the reference or a specific level the supplied factor, returning it as a factor with the same levels as the one supplied.

**Usage**

```
ref_level(fct)
level(fct, level)
```

**Arguments**

fct                    factor; the factor from which the reference or specific level will be extracted.  
level                  character; the specific level to extract in the case of level().

**Value**

A length 1 factor with the same levels as the supplied factor fct.

**Examples**

```
f <- factor(sample(letters[1:5], 100, replace = TRUE))
# the reference level
ref_level(f)
# a specific level
level(f, level = "b")
# note that the levels will always match the input factor
identical(levels(f), levels(ref_level(f)))
identical(levels(f), levels(level(f, "c")))
```

---

ref\_sims *Reference simulation data*

---

**Description**

A set of reference objects for testing [data\\_sim\(\)](#).

**Format**

A named list of simulated data sets created by [data\\_sim\(\)](#).

---

`rep_first_factor_value`*Repeat the first level of a factor n times*

---

**Description**

Function to repeat the first level of a factor n times and return this vector as a factor with the original levels intact

**Usage**

```
rep_first_factor_value(f, n)
```

**Arguments**

f	a factor
n	numeric; the number of times to repeat the first level of f

**Value**

A factor of length n with the levels of f, but whose elements are all the first level of f.

---

`residuals_hist_plot` *Histogram of model residuals*

---

**Description**

Histogram of model residuals

**Usage**

```
residuals_hist_plot(  
  model,  
  type = c("deviance", "pearson", "response"),  
  n_bins = c("sturges", "scott", "fd"),  
  ylab = NULL,  
  xlab = NULL,  
  title = NULL,  
  subtitle = NULL,  
  caption = NULL  
)
```

**Arguments**

model	a fitted model. Currently only class "gam".
type	character; type of residuals to use. Only "deviance", "response", and "pearson" residuals are allowed.
n_bins	character or numeric; either the number of bins or a string indicating how to calculate the number of bins.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
xlab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See <a href="#">ggplot2::labs()</a> .
subtitle	character or expression; the subtitle for the plot. See <a href="#">ggplot2::labs()</a> .
caption	character or expression; the plot caption. See <a href="#">ggplot2::labs()</a> .

---

residuals\_linpred\_plot

*Plot of residuals versus linear predictor values*


---

**Description**

Plot of residuals versus linear predictor values

**Usage**

```
residuals_linpred_plot(
  model,
  type = c("deviance", "pearson", "response"),
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  point_col = "black",
  point_alpha = 1,
  line_col = "red"
)
```

**Arguments**

model	a fitted model. Currently only class "gam".
type	character; type of residuals to use. Only "deviance", "response", and "pearson" residuals are allowed.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.



xlab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See <code>ggplot2::labs()</code> .
subtitle	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> .
caption	character or expression; the plot caption. See <code>ggplot2::labs()</code> .
point_col	colour used to draw points in the plots. See <code>graphics::par()</code> section <b>Color Specification</b> . This is passed to the individual plotting functions, and therefore affects the points of all plots.
point_alpha	numeric; alpha transparency for points in plots.
line_col	colour specification for 1:1 line.

---

rootogram

*Rootograms to assess goodness of model fit*


---

## Description

A rootogram is a model diagnostic tool that assesses the goodness of fit of a statistical model. The observed values of the response are compared with those expected from the fitted model. For discrete, count responses, the frequency of each count (0, 1, 2, etc) in the observed data and expected from the conditional distribution of the response implied by the model are compared. For continuous variables, the observed and expected frequencies are obtained by grouping the data into bins. The rootogram is drawn using `ggplot2::ggplot()` graphics. The design closely follows Kleiber & Zeileis (2016).

## Usage

```
rootogram(object, ...)

## S3 method for class 'gam'
rootogram(object, max_count = NULL, breaks = "Sturges", ...)
```

## Arguments

object	an R object
...	arguments passed to other methods
max_count	integer; the largest count to consider
breaks	for continuous responses, how to group the response. Can be anything that is acceptable as the breaks argument of <code>graphics::hist.default()</code>

## References

Kleiber, C., Zeileis, A., (2016) Visualizing Count Data Regressions Using Rootograms. *Am. Stat.* **70**, 296–303. doi:10.1080/00031305.2016.1173590

**Examples**

```

load_mgcv()

df <- data_sim("eg1", n = 1000, dist = "poisson", scale = 0.1, seed = 6)

# A poisson example
m <- gam(y ~ s(x0, bs = "cr") + s(x1, bs = "cr") + s(x2, bs = "cr") +
         s(x3, bs = "cr"), family = poisson(), data = df, method = "REML")
rg <- rootogram(m)
rg
draw(rg) # plot the rootogram

# A Gaussian example
df <- data_sim("eg1", dist = "normal", seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")
draw(rootogram(m, breaks = "FD"), type = "suspended")

```

---

seq_min_max_eps	<i>Create a sequence of evenly-spaced values adjusted to accommodate a small adjustment</i>
-----------------	---

---

**Description**

Creates a sequence of  $n$  evenly-spaced values over the range  $\min(x) - \max(x)$ , where the minimum and maximum are adjusted such that they are always contained within the range of  $x$  when  $x$  may be shifted forwards or backwards by an amount related to  $\text{eps}$ . This is particularly useful in computing derivatives via finite differences where without this adjustment we may be predicting for values outside the range of the data and hence the constraints of the penalty.

**Usage**

```
seq_min_max_eps(x, n, order, type = c("forward", "backward", "central"), eps)
```

**Arguments**

<code>x</code>	numeric; vector over which evenly-spaced values are returned
<code>n</code>	numeric; the number of evenly-spaced values to return
<code>order</code>	integer; the order of derivative. Either 1 or 2 for first or second order derivatives
<code>type</code>	character; the type of finite difference used. One of "forward", "backward", or "central"
<code>eps</code>	numeric; the finite difference

**Value**

A numeric vector of length  $n$ .

---

shift_values	<i>Shift numeric values in a data frame by an amount eps</i>
--------------	--

---

**Description**

Shift numeric values in a data frame by an amount eps

**Usage**

```
shift_values(df, h, i, FUN = `+`, focal = NULL)
```

**Arguments**

df	a data frame or tibble.
h	numeric; the amount to shift values in df by.
i	logical; a vector indexing columns of df that should not be included in the shift.
FUN	function; a function to apply the shift. Typically + or -.
focal	character; the focal variable when computing partial derivatives. This allows shifting only the focal variable by eps.

---

simulate.gam	<i>Simulate from the posterior distribution of a GAM</i>
--------------	--

---

**Description**

Simulations from the posterior distribution of a fitted GAM model involve computing predicted values for the observation data for which simulated data are required, then generating random draws from the probability distribution used when fitting the model.

**Usage**

```
## S3 method for class 'gam'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  data = newdata,
  weights = NULL,
  ...,
  newdata = NULL
)

## S3 method for class 'gamm'
simulate(
```

```

    object,
    nsim = 1,
    seed = NULL,
    data = newdata,
    weights = NULL,
    ...,
    newdata = NULL
)

## S3 method for class 'scam'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  data = newdata,
  weights = NULL,
  ...,
  newdata = NULL
)

```

### Arguments

object	a fitted GAM, typically the result of a call to <code>mgcv::gam</code> or <code>mgcv::gamm()</code> .
nsim	numeric; the number of posterior simulations to return.
seed	numeric; a random seed for the simulations.
data	data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for newdata, if available in object.
weights	numeric; a vector of prior weights. If newdata is null then defaults to <code>object[["prior.weights"]]</code> , otherwise a vector of ones.
...	arguments passed to methods. <code>simulate.gam()</code> and <code>simulate.scam()</code> pass ... on to <code>predict.gam()</code> . As such you can pass additional arguments such as <code>terms</code> , <code>exclude</code> , to select which model terms are included in the predictions. This may be useful, for example, for excluding the effects of random effect terms.
newdata	Deprecated. Use data instead.

### Details

For `simulate.gam()` to function, the family component of the fitted model must contain, or be updateable to contain, the required random number generator. See `mgcv::fix.family.rd()`.

### Value

(Currently) A matrix with `nsim` columns.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()
dat <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

sims <- simulate(m1, nsim = 5, seed = 42)
head(sims)
```

---

smallAges

*Lead-210 age-depth measurements for Small Water*

---

**Description**

A dataset containing lead-210 based age depth measurements for the SMALL1 core from Small Water.

**Format**

A data frame with 12 rows and 7 variables.

**Details**

The variables are as follows:

- Depth
- Drymass
- Date
- Age
- Error
- SedAccRate
- SedPerCentChange

**Source**

Simpson, G.L. (Unpublished data).

---

smooths	<i>Names of smooths in a GAM</i>
---------	----------------------------------

---

**Description**

Names of smooths in a GAM

**Usage**

```
smooths(object)
```

**Arguments**

object            a fitted GAM or related model. Typically the result of a call to `mgcv::gam()`, `mgcv::bam()`, or `mgcv::gamm()`.

---

smooth_coefs	<i>Coefficients for a particular smooth</i>
--------------	---

---

**Description**

Returns a vector of model coefficients of the parametric terms that represent the supplied smooth.

**Usage**

```
smooth_coefs(object, ...)

## S3 method for class 'gam'
smooth_coefs(object, term, ...)

## S3 method for class 'bam'
smooth_coefs(object, term, ...)

## S3 method for class 'gamm'
smooth_coefs(object, term, ...)

## S3 method for class 'gamm4'
smooth_coefs(object, term, ...)

## S3 method for class 'list'
smooth_coefs(object, term, ...)

## S3 method for class 'mgcv.smooth'
smooth_coefs(object, model, ...)

## S3 method for class 'scam'
smooth_coefs(object, term, ...)
```

**Arguments**

object	a fitted GAM(M) object, or, for the "mgcv.smooth" method, an object that inherits from class mgcv.smooth.
...	arguments passed to other methods.
term	character; the label of the smooth whose coefficients will be returned.
model	a fitted GAM(M) object.

**Value**

A numeric vector of model coefficients.

**Author(s)**

Gavin L. Simpson

**See Also**

[smooth\\_coef\\_indices\(\)](#) for extracting the indices of the coefficients for a particular smooth.

**Examples**

```
load_mgcv()
df <- data_sim("eg1", seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

## IGNORE_RDIFF_BEGIN
smooth_coefs(m, term = "s(x2)")
## IGNORE_RDIFF_END
```

---

smooth\_coef\_indices    *Indices of the parametric terms for a particular smooth*

---

**Description**

Returns a vector of indices of the parametric terms that represent the supplied smooth. Useful for extracting model coefficients and columns of their covariance matrix.

**Usage**

```
smooth_coef_indices(smooth)
```

**Arguments**

smooth            an object that inherits from class mgcv.smooth

**Value**

A numeric vector of indices.

**Author(s)**

Gavin L. Simpson

**See Also**

[smooth\\_coefs\(\)](#) for extracting the coefficients for a particular smooth.

---

smooth\_data

*Generate regular data over the covariates of a smooth*

---

**Description**

Generate regular data over the covariates of a smooth

**Usage**

```
smooth_data(
  model,
  id,
  n = 100,
  n_3d = NULL,
  n_4d = NULL,
  offset = NULL,
  include_all = FALSE
)
```

**Arguments**

model	a fitted model
id	the number ID of the smooth within model to process.
n	numeric; the number of new observations to generate.
n_3d	numeric; the number of new observations to generate for the third dimension of a 3D smooth.
n_4d	numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a $k$ D smooth ( $k \geq 4$ ). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get <code>n_4d</code> new observations.
offset	numeric; value of the model offset to use.
include_all	logical; include all covariates involved in the smooth? if FALSE, only the covariates involved in the smooth will be included in the returned data frame. If TRUE, a representative value will be included for all other covariates in the model that aren't actually used in the smooth. This can be useful if you want to pass the returned data frame on to <a href="#">mgcv::PredictMat()</a> .



## Examples

```
load_mgcv()
df <- data_sim("eg1", seed = 42)
m <- bam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df)

# generate data over range of x1 for smooth s(x1)
smooth_data(m, id = 2)

# generate data over range of x1 for smooth s(x1), with typical value for
# other covariates in the model
smooth_data(m, id = 2, include_all = TRUE)
```

---

smooth\_dim

*Dimension of a smooth*

---

## Description

Extracts the dimension of an estimated smooth.

## Usage

```
smooth_dim(object)

## S3 method for class 'gam'
smooth_dim(object)

## S3 method for class 'gamm'
smooth_dim(object)

## S3 method for class 'mgcv.smooth'
smooth_dim(object)
```

## Arguments

object            an R object. See Details for list of supported objects.

## Details

This is a generic function with methods for objects of class "gam", "gamm", and "mgcv.smooth".

## Value

A numeric vector of dimensions for each smooth.

**Author(s)**

Gavin L. Simpson

smooth\_estimates

*Evaluate smooths at covariate values***Description**

Evaluate a smooth at a grid of evenly spaced value over the range of the covariate associated with the smooth. Alternatively, a set of points at which the smooth should be evaluated can be supplied. `smooth_estimates()` is a new implementation of `evaluate_smooth()`, and should be used instead of that other function.

**Usage**

```
smooth_estimates(object, ...)
```

```
## S3 method for class 'gam'
smooth_estimates(
  object,
  smooth = NULL,
  n = 100,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  unnest = TRUE,
  partial_match = FALSE,
  ...
)
```

**Arguments**

<code>object</code>	an object of class "gam" or "gamm".
<code>...</code>	arguments passed to other methods.
<code>smooth</code>	character; a single smooth to evaluate.
<code>n</code>	numeric; the number of points over the range of the covariate at which to evaluate the smooth.
<code>n_3d, n_4d</code>	numeric; the number of points over the range of last covariate in a 3D or 4D smooth. The default is NULL which achieves the standard behaviour of using $n$ points over the range of all covariate, resulting in $n^d$ evaluation points, where $d$ is the dimension of the smooth. For $d > 2$ this can result in very many evaluation points and slow performance. For smooths of $d > 4$ , the value of <code>n_4d</code> will be used for all dimensions $> 4$ , unless this is NULL, in which case the default behaviour (using $n$ for all dimensions) will be observed.

data	a data frame of covariate values at which to evaluate the smooth.
unconditional	logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.
overall_uncertainty	logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?
dist	numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and dist is a distance within the unit square. See <code>mgcv::exclude.too.far()</code> for further details.
unnest	logical; unnest the smooth objects?
partial_match	logical; in the case of character select, should select match partially against smooths? If <code>partial_match = TRUE</code> , select must only be a single string, a character vector of length 1.

### Value

A data frame (tibble), which is of class "smooth\_estimates".

### Examples

```
load_mgcv()

dat <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

## evaluate all smooths
smooth_estimates(m1)

## or selected smooths
smooth_estimates(m1, smooth = c("s(x0)", "s(x1)"))
```

---

smooth_samples	<i>Posterior draws for individual smooths</i>
----------------	---

---

### Description

Returns draws from the posterior distributions of smooth functions in a GAM. Useful, for example, for visualising the uncertainty in individual estimated functions.

### Usage

```
smooth_samples(model, ...)

## S3 method for class 'gam'
smooth_samples(
```

```

model,
term = NULL,
n = 1,
data = newdata,
seed = NULL,
freq = FALSE,
unconditional = FALSE,
ncores = 1L,
n_vals = 200,
...,
newdata = NULL
)

```

### Arguments

model	a fitted model of the supported types
...	arguments passed to other methods. For <code>fitted_samples()</code> , these are passed on to <code>predict.gam()</code> .
term	character; select which smooth's posterior to draw from. The default (NULL) means the posteriors of all smooths in <code>model</code> will be sampled from. If supplied, a character vector of requested terms.
n	numeric; the number of posterior samples to return.
data	data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for data, if available in <code>model</code> .
seed	numeric; a random seed for the simulations.
freq	logical; TRUE to use the frequentist covariance matrix of the parameter estimators, FALSE to use the Bayesian posterior covariance matrix of the parameters.
unconditional	logical; if TRUE (and <code>freq == FALSE</code> ) then the Bayesian smoothing parameter uncertainty corrected covariance matrix is used, if available.
ncores	number of cores for generating random variables from a multivariate normal distribution. Passed to <code>mvnfast::rmvn()</code> . Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).
n_vals	numeric; how many locations to evaluate the smooth at if data not supplied
newdata	Deprecated: use <code>data</code> instead.

### Value

A tibble with additional classes "smooth\_samples" and "posterior\_samples".

For the "gam" method, the columns currently returned (not in this order) are:

- `smooth`; character vector. Indicates the smooth function for that particular draw,
- `term`; character vector. Similar to `smooth`, but will contain the full label for the smooth, to differentiate factor-by smooths for example.
- `by_variable`; character vector. If the smooth involves a by term, the by variable will be named here, `NA_character_ otherwise`.

- row; integer. A vector of values seq\_len(n\_vals), repeated if n > 1L. Indexes the row in data for that particular draw.
- draw; integer. A vector of integer values indexing the particular posterior draw that each row belongs to.
- value; numeric. The value of smooth function for this posterior draw and covariate combination.
- xxx; numeric. A series of one or more columns containing data required for the smooth, named as per the variables involved in the respective smooth.
- Additional columns will be present in the case of factor by smooths, which will contain the level for the factor named in by\_variable for that particular posterior draw.

### Warning

The set of variables returned and their order in the tibble is subject to change in future versions. Don't rely on position.

### Author(s)

Gavin L. Simpson

### Examples

```
load_mgcv()

dat <- data_sim("eg1", n = 400, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

sms <- smooth_samples(m1, term = "s(x0)", n = 5, seed = 42)

sms

## A factor by example (with a spurious covariate x0)
dat <- data_sim("eg4", n = 1000, seed = 2)

## fit model...
m2 <- gam(y ~ fac + s(x2, by = fac) + s(x0), data = dat)
sms <- smooth_samples(m2, n = 5, seed = 42)
draw(sms)
```

---

smooth\_type

*Determine the type of smooth and return it in a human readable form*

---

### Description

Determine the type of smooth and return it in a human readable form

**Usage**

```
smooth_type(smooth)

## Default S3 method:
smooth_type(smooth)

## S3 method for class 'tprs.smooth'
smooth_type(smooth)

## S3 method for class 'ts.smooth'
smooth_type(smooth)

## S3 method for class 'cr.smooth'
smooth_type(smooth)

## S3 method for class 'cs.smooth'
smooth_type(smooth)

## S3 method for class 'cyclic.smooth'
smooth_type(smooth)

## S3 method for class 'pspline.smooth'
smooth_type(smooth)

## S3 method for class 'cp.smooth'
smooth_type(smooth)

## S3 method for class 'Bspline.smooth'
smooth_type(smooth)

## S3 method for class 'duchon.spline'
smooth_type(smooth)

## S3 method for class 'fs.interaction'
smooth_type(smooth)

## S3 method for class 'sz.interaction'
smooth_type(smooth)

## S3 method for class 'gp.smooth'
smooth_type(smooth)

## S3 method for class 'mrf.smooth'
smooth_type(smooth)

## S3 method for class 'random.effect'
smooth_type(smooth)
```

```
## S3 method for class 'sw'  
smooth_type(smooth)  
  
## S3 method for class 'sf'  
smooth_type(smooth)  
  
## S3 method for class 'soap.film'  
smooth_type(smooth)  
  
## S3 method for class 't2.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'sos.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'tensor.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'mpi.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'mpd.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'cx.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'cv.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'micx.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'micv.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'mdcx.smooth'  
smooth_type(smooth)  
  
## S3 method for class 'mdcv.smooth'  
smooth_type(smooth)
```

### Arguments

smooth            an object inheriting from class `mgcv.smooth`.

---

term_names	<i>Extract names of all variables needed to fit a GAM or a smooth</i>
------------	---

---

**Description**

Extract names of all variables needed to fit a GAM or a smooth

**Usage**

```
term_names(object, ...)

## S3 method for class 'gam'
term_names(object, ...)

## S3 method for class 'mgcv.smooth'
term_names(object, ...)

## S3 method for class 'gamm'
term_names(object, ...)
```

**Arguments**

object	a fitted GAM object or an mgcv smooth object
...	arguments passed to other methods. Not currently used.

**Value**

A vector of variable names required for terms in the model

---

term_variables	<i>Names of variables involved in a specified model term</i>
----------------	--

---

**Description**

Given the name (a term label) of a term in a model, returns the names of the variables involved in the term.

**Usage**

```
term_variables(object, term, ...)

## S3 method for class 'terms'
term_variables(object, term, ...)

## S3 method for class 'gam'
```



```
term_variables(object, term, ...)

## S3 method for class 'bam'
term_variables(object, term, ...)
```

### Arguments

**object** an R object on which method dispatch is performed

**term** character; the name of a model term, in the sense of `attr(terms(object), "term.labels")`. Currently not checked to see if the term exists in the model.

**...** arguments passed to other methods.

### Value

A character vector of variable names.

---

theta	<i>General extractor for additional parameters in mgcv models</i>
-------	---

---

### Description

General extractor for additional parameters in mgcv models

### Usage

```
theta(object, ...)

## S3 method for class 'gam'
theta(object, transform = TRUE, ...)
```

### Arguments

**object** a fitted model

**...** arguments passed to other methods.

**transform** logical; transform to the natural scale of the parameter

### Value

Returns a numeric vector of additional parameters

### Examples

```
load_mgcv()
df <- data_sim("eg1", dist = "poisson", seed = 42, scale = 1/5)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML",
         family = nb())
p <- theta(m)
```

tidy\_basis

*A tidy basis representation of a smooth object***Description**

Takes an object of class `mgcv.smooth` and returns a tidy representation of the basis.

**Usage**

```
tidy_basis(smooth, data = NULL, at = NULL, coefs = NULL, p_ident = NULL)
```

**Arguments**

<code>smooth</code>	a smooth object of or inheriting from class <code>"mgcv.smooth"</code> . Typically, such objects are returned as part of a fitted GAM or GAMM in the <code>\$smooth</code> component of the model object or the <code>\$gam\$smooth</code> component if the model was fitted by <code>mgcv::gam()</code> or <code>gam4::gam4()</code> .
<code>data</code>	a data frame containing the variables used in <code>smooth</code> .
<code>at</code>	a data frame containing values of the smooth covariate(s) at which the basis should be evaluated.
<code>coefs</code>	numeric; an optional vector of coefficients for the smooth
<code>p_ident</code>	logical vector; only used for handling <code>scam::scam()</code> smooths.

**Value**

A tibble.

**Author(s)**

Gavin L. Simpson

**Examples**

```
load_mgcv()

df <- data_sim("eg1", n = 400, seed = 42)

# fit model
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

# tidy representation of a basis for a smooth definition
# extract the smooth
sm <- get_smooth(m, "s(x2)")
# get the tidy basis - need to pass where we want it to be evaluated
bf <- tidy_basis(sm, at = df)

# can weight the basis by the model coefficients for this smooth
bf <- tidy_basis(sm, at = df, coefs = smooth_coefs(sm, model = m))
```

---

too_far	<i>Exclude values that lie too far from the support of data</i>
---------	---

---

**Description**

Identifies pairs of covariate values that lie too far from the original data. The function is currently a basic wrapper around `mgcv::exclude.too.far()`.

**Usage**

```
too_far(x, y, ref_1, ref_2, dist = NULL)
```

**Arguments**

<code>x, y</code>	numeric; vector of values of the covariates to compare with the observed data
<code>ref_1, ref_2</code>	numeric; vectors of covariate values that represent the reference against which <code>x1</code> and <code>x2</code> are compared
<code>dist</code>	if supplied, a numeric vector of length 1 representing the distance from the data beyond which an observation is excluded. For example, you want to exclude values that lie further from an observation than 10% of the range of the observed data, use <code>0.1</code> .

**Value**

Returns a logical vector of the same length as `x1`.

---

too_far_to_na	<i>Set rows of data to NA if the lie too far from a reference set of values</i>
---------------	---

---

**Description**

Set rows of data to NA if the lie too far from a reference set of values

**Usage**

```
too_far_to_na(smooth, input, reference, cols, dist = NULL)
```

**Arguments**

<code>smooth</code>	an <code>mgcv</code> smooth object
<code>input</code>	data frame containing the input observations and the columns to be set to NA
<code>reference</code>	data frame containing the reference values
<code>cols</code>	character vector of columns whose elements will be set to NA if the data lies too far from the reference set
<code>dist</code>	numeric, the distance from the reference set beyond which elements of input will be set to NA

---

to_na	<i>Sets the elements of vector to NA</i>
-------	--

---

**Description**

Given a vector `i` indexing the elements of `x`, sets the selected elements of `x` to `NA`.

**Usage**

```
to_na(x, i)
```

**Arguments**

<code>x</code>	vector of values
<code>i</code>	vector of values used to subset <code>x</code>

**Value**

Returns `x` with possibly some elements set to `NA`

---

transform_fun	<i>Transform estimated values and confidence intervals by applying a function</i>
---------------	---

---

**Description**

Transform estimated values and confidence intervals by applying a function

**Usage**

```
transform_fun(object, fun = NULL, ...)
```

```
## S3 method for class 'evaluated_smooth'  
transform_fun(object, fun = NULL, ...)
```

```
## S3 method for class 'smooth_estimates'  
transform_fun(object, fun = NULL, ...)
```

```
## S3 method for class 'mgcv_smooth'  
transform_fun(object, fun = NULL, ...)
```

```
## S3 method for class 'evaluated_parametric_term'  
transform_fun(object, fun = NULL, ...)
```

```
## S3 method for class 'parametric_effects'
```

```
transform_fun(object, fun = NULL, ...)

## S3 method for class 'tbl_df'
transform_fun(object, fun = NULL, column = NULL, ...)
```

**Arguments**

`object` an object to apply the transform function to.

`fun` the function to apply.

`...` additional arguments passed to methods.

`column` character; for the "tbl\_df" method, which column to transform.

**Value**

Returns object but with the estimate and upper and lower values of the confidence interval transformed via the function.

**Author(s)**

Gavin L. Simpson

---

typical_values	<i>Typical values of model covariates</i>
----------------	---

---

**Description**

Typical values of model covariates

**Usage**

```
typical_values(object, ...)

## S3 method for class 'gam'
typical_values(object, vars = everything(), ...)
```

**Arguments**

`object` a fitted GAM(M) model.

`...` arguments passed to other methods.

`vars` terms to include or exclude from the returned object. Uses tidyselect principles.

---

variance_comp	<i>Variance components of smooths from smoothness estimates</i>
---------------	---

---

**Description**

A wrapper to `mgcv::gam.vcomp()` which returns the smoothing parameters expressed as variance components.

**Usage**

```
variance_comp(object, ...)

## S3 method for class 'gam'
variance_comp(object, rescale = TRUE, coverage = 0.95, ...)
```

**Arguments**

object	an R object. Currently only models fitted by <code>mgcv::gam()</code> or <code>mgcv::bam()</code> are supported.
...	arguments passed to other methods
rescale	logical; for numerical stability reasons the penalty matrices of smooths are rescaled before fitting. If <code>rescale = TRUE</code> , this rescaling is undone, resulting in variance components that are on their original scale. This is needed if comparing with other mixed model software, such as <code>lmer()</code> .
coverage	numeric; a value between 0 and 1 indicating the (approximate) coverage of the confidence interval that is returned.

**Details**

This function is a wrapper to `mgcv::gam.vcomp()` which performs three additional services

- it suppresses the annoying text output that `mgcv::gam.vcomp()` prints to the terminal,
- returns the variance of each smooth as well as the standard deviation, and
- returns the variance components as a tibble.

---

vars_from_label	<i>Returns names of variables from a smooth label</i>
-----------------	---

---

**Description**

Returns names of variables from a smooth label

**Usage**

```
vars_from_label(label)
```

**Arguments**

label            character; a length 1 character vector containing the label of a smooth.

**Examples**

```
vars_from_label("s(x1)")
vars_from_label("t2(x1,x2,x3)")
```

---

which_smooths	<i>Identify a smooth term by its label</i>
---------------	--

---

**Description**

Identify a smooth term by its label

**Usage**

```
which_smooths(object, ...)

## Default S3 method:
which_smooths(object, ...)

## S3 method for class 'gam'
which_smooths(object, terms, ...)

## S3 method for class 'bam'
which_smooths(object, terms, ...)

## S3 method for class 'gamm'
which_smooths(object, terms, ...)
```

**Arguments**

object            a fitted GAM.

...                arguments passed to other methods.

terms             character; one or more (partial) term labels with which to identify required smooths.

worm\_plot

*Worm plot of model residuals***Description**

Worm plot of model residuals

**Usage**

```
worm_plot(model, ...)

## S3 method for class 'gam'
worm_plot(
  model,
  method = c("uniform", "simulate", "normal", "direct"),
  type = c("deviance", "response", "pearson"),
  n_uniform = 10,
  n_simulate = 50,
  level = 0.9,
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ci_col = "black",
  ci_alpha = 0.2,
  point_col = "black",
  point_alpha = 1,
  line_col = "red",
  ...
)

## S3 method for class 'glm'
worm_plot(model, ...)

## S3 method for class 'lm'
worm_plot(model, ...)
```

**Arguments**

model	a fitted model. Currently only class "gam".
...	arguments passed to other methods.
method	character; method used to generate theoretical quantiles. Note that method = "direct" is deprecated in favour of method = "uniform".
type	character; type of residuals to use. Only "deviance", "response", and "pearson" residuals are allowed.



n_uniform	numeric; number of times to randomize uniform quantiles in the direct computation method (method = "uniform").
n_simulate	numeric; number of data sets to simulate from the estimated model when using the simulation method (method = "simulate").
level	numeric; the coverage level for reference intervals. Must be strictly $0 < \text{level} < 1$ . Only used with method = "simulate".
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
xlab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See <code>ggplot2::labs()</code> .
subtitle	character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code> .
caption	character or expression; the plot caption. See <code>ggplot2::labs()</code> .
ci_col, ci_alpha	fill colour and alpha transparency for the reference interval when method = "simulate".
point_col, point_alpha	colour and alpha transparency for points on the QQ plot.
line_col	colour used to draw the reference line.

### Note

The wording used in `mgcv::qq.gam()` uses *direct* in reference to the simulated residuals method (method = "simulated"). To avoid confusion, method = "direct" is deprecated in favour of method = "uniform".

### Examples

```
load_mgcv()
## simulate binomial data...
dat <- data_sim("eg1", n = 200, dist = "binary", scale = .33, seed = 0)
p <- binomial()$linkinv(dat$f) # binomial p
n <- sample(c(1, 3), 200, replace = TRUE) # binomial n
dat <- transform(dat, y = rbinom(n, n, p), n = n)
m <- gam( y / n ~ s(x0) + s(x1) + s(x2) + s(x3),
          family = binomial, data = dat, weights = n,
          method = "REML")

## Worm plot; default using direct randomization of uniform quantiles
## Note no reference bands are drawn with this method.
worm_plot(m)

## Alternatively use simulate new data from the model, which
## allows construction of reference intervals for the Q-Q plot
worm_plot(m, method = "simulate", point_col = "steelblue",
          point_alpha = 0.4)

## ... or use the usual normality assumption
worm_plot(m, method = "normal")
```

---

zooplankton

*Madison lakes zooplankton data*

---

### **Description**

The Madison lake zooplankton data are from a long-term study in seasonal dynamics of zooplankton, collected by the Richard Lathrop. The data were collected from a chain of lakes in Wisconsin (Mendota, Monona, Kegonsa, and Waubesa) approximately bi-weekly from 1976 to 1994. They consist of samples of the zooplankton communities, taken from the deepest point of each lake via vertical tow. The data are provided by the Wisconsin Department of Natural Resources and their collection and processing are fully described in Lathrop (2000).

### **Format**

A data frame

### **Details**

Each record consists of counts of a given zooplankton taxon taken from a subsample from a single vertical net tow, which was then scaled to account for the relative volume of subsample versus the whole net sample and the area of the net tow and rounded to the nearest 1000 to give estimated population density per m<sup>2</sup> for each taxon at each point in time in each sampled lake.

### **Source**

Pedersen EJ, Miller DL, Simpson GL, Ross N. 2018. Hierarchical generalized additive models: an introduction with mgcv. *PeerJ Preprints* 6:e27320v1 [doi:10.7287/peerj.preprints.27320v1](https://doi.org/10.7287/peerj.preprints.27320v1).

### **References**

Lathrop RC. (2000). Madison Wisconsin Lakes Zooplankton 1976–1994. Environmental Data Initiative.

# Index

- \* **data**
  - bird\_move, 14
  - gss\_vocab, 68
  - ref\_sims, 94
  - smallAges, 101
  - zooplankton, 122
- \* **draw methods**
  - draw.rootogram, 46
- add\_confint, 4
- add\_constant, 5
- add\_fitted, 6
- add\_fitted.gam, 6
- add\_partial\_residuals, 7
- add\_residuals, 8
- add\_residuals.gam, 9
- appraise, 10
  
- base::set.seed(), 22
- base::zapsmall(), 88
- basis, 12
- basis(), 29, 43
- bird\_move, 14
- by\_level(is\_by\_smooth), 69
- by\_variable(is\_by\_smooth), 69
  
- check\_is\_mgcv\_smooth(is\_mgcv\_smooth), 71
- check\_user\_select\_smooths, 15
- coef.scam, 15
- compare\_smooths, 16
- compare\_smooths(), 30
- concrvity(model\_concurvity), 76
- confint.fderiv, 17
- confint.gam, 19
- confint.gamm(confint.gam), 19
- confint.list(confint.gam), 19
  
- data\_combos, 21
- data\_sim, 22
  
- data\_sim(), 94
- data\_slice, 23
- derivatives, 24
- difference\_smooths, 26
- draw, 27
- draw(), 41
- draw.basis, 28
- draw.compare\_smooths, 30
- draw.derivatives, 30
- draw.difference\_smooth, 32
- draw.evaluated\_1d\_smooth
  - (draw.evaluated\_smooth), 33
- draw.evaluated\_2d\_smooth
  - (draw.evaluated\_smooth), 33
- draw.evaluated\_fs\_smooth
  - (draw.evaluated\_smooth), 33
- draw.evaluated\_parametric\_term
  - (draw.evaluated\_smooth), 33
- draw.evaluated\_re\_smooth
  - (draw.evaluated\_smooth), 33
- draw.evaluated\_smooth, 33
- draw.gam, 37
- draw.gam(), 42
- draw.gamlss, 41
- draw.mgcv\_smooth, 42
- draw.parametric\_effects, 44
- draw.partial\_derivatives
  - (draw.derivatives), 30
- draw.penalty\_df, 45
- draw.rootogram, 46
- draw.smooth\_estimates, 48
- draw.smooth\_samples, 50
  
- edf, 52
- eval\_smooth, 56
- evaluate\_parametric\_term, 54
- evaluate\_smooth, 54
- evaluate\_smooth(), 35, 45, 106
- evenly, 59
- extract\_link(link), 72

- factor\_combos, 59
- family(), 61, 69, 73
- family.bam (family.gam), 60
- family.gam, 60
- family.gamm (family.gam), 60
- family.list (family.gam), 60
- family\_name, 60
- family\_type, 61
- fitted\_samples, 61
- fitted\_values, 63
- fix\_offset, 66
- fixed\_effects (fixef.gam), 65
- fixef, 64
- fixef.gam, 65
- fixef.gamm (fixef.gam), 65
- fixef.glm (fixef.gam), 65
- fixef.lm (fixef.gam), 65
  
- gamm4::gamm4(), 60, 73, 114
- geom\_rug (draw.evaluated\_smooth), 33
- get\_by\_smooth, 66
- get\_smooth, 67
- get\_smooths\_by\_id, 67
- ggplot2::coord\_map(), 40, 49
- ggplot2::geom\_contour(), 29, 33, 36, 39, 49, 51
- ggplot2::geom\_line(), 51
- ggplot2::ggplot(), 28, 36, 37, 40, 43, 46, 97
- ggplot2::guide\_axis(), 29, 31, 33, 39, 43, 45, 49, 51
- ggplot2::label\_both(), 29, 43
- ggplot2::labs(), 29, 36, 43, 46, 51, 80, 93, 96, 97, 121
- ggplot2::scale\_colour\_viridis\_d(), 36
- GJRM::gamlss(), 41
- graphics::hist.default(), 97
- graphics::par(), 11, 80, 97
- gss\_vocab, 68
- gw\_f0, 68
- gw\_f1 (gw\_f0), 68
- gw\_f2 (gw\_f0), 68
- gw\_f3 (gw\_f0), 68
  
- has\_theta, 69
  
- inv\_link (link), 72
- is\_by\_smooth, 69
- is\_continuous\_by\_smooth (is\_by\_smooth), 69
  
- is\_factor\_by\_smooth (is\_by\_smooth), 69
- is\_factor\_term, 70
- is\_mgcv\_smooth, 71
- is\_mrf\_smooth (is\_mgcv\_smooth), 71
- is\_offset, 71
  
- level (ref\_level), 94
- link, 72
- load\_mgcv, 74
- lp\_matrix, 74
  
- mgcv::bam(), 58, 60, 64, 73, 79, 102, 118
- mgcv::exclude.too.far(), 39, 55, 58, 107, 115
- mgcv::fix.family.rd(), 100
- mgcv::gam, 100
- mgcv::gam(), 31, 32, 38, 44, 48, 50, 58, 60, 64, 73, 79, 102, 118
- mgcv::gam.vcomp(), 118
- mgcv::gamm(), 60, 73, 79, 87, 100, 102, 114
- mgcv::gammals(), 73
- mgcv::gamSim(), 22
- mgcv::gaulss(), 73
- mgcv::gevlss(), 73
- mgcv::multinom(), 73
- mgcv::mvn(), 73
- mgcv::plot.gam(), 38
- mgcv::predict.bam(), 75
- mgcv::predict.gam(), 7, 64, 75
- mgcv::PredictMat(), 104
- mgcv::qq.gam(), 11, 93, 121
- mgcv::residuals.gam(), 9
- mgcv::s(), 12, 13
- mgcv::shash(), 73
- mgcv::smoothCon(), 13
- mgcv::t2(), 12, 13
- mgcv::te(), 12, 13
- mgcv::ti(), 12, 13
- mgcv::twlss(), 73
- mgcv::ziplss(), 73
- model\_concurvity, 76
- model\_edf (edf), 52
- model\_vars, 77
- mvnfast::rmvn(), 18, 20, 25, 62, 84, 89, 108
  
- n\_smooths, 79
- nb\_theta, 78
  
- observed\_fitted\_plot, 79

observed\_fitted\_plot(), 11  
overview, 80

parametric\_effects, 81  
parametric\_effects(), 54  
parametric\_terms, 82  
partial\_derivatives, 82  
partial\_residuals, 85  
patchwork::plot\_layout(), 11, 29–31, 33, 40, 41, 45, 46, 51  
patchwork::wrap\_plots(), 10, 30, 31, 33, 40, 43, 45, 49, 51  
penalty, 87  
posterior\_samples, 88  
predicted\_samples, 90

qq\_plot, 91  
qq\_plot(), 11

ref\_level, 94  
ref\_sims, 94  
rep\_first\_factor\_value, 95  
residuals\_hist\_plot, 95  
residuals\_hist\_plot(), 11  
residuals\_linpred\_plot, 96  
residuals\_linpred\_plot(), 11  
rootogram, 97  
rootogram(), 47

scam::scam(), 114  
seq\_min\_max(evenly), 59  
seq\_min\_max\_eps, 98  
shift\_values, 99  
simulate.gam, 99  
simulate.gam(), 90  
simulate.gamm(simulate.gam), 99  
simulate.scam(simulate.gam), 99  
smallAges, 101  
smooth\_coef\_indices, 103  
smooth\_coef\_indices(), 103  
smooth\_coefs, 102  
smooth\_coefs(), 104  
smooth\_data, 104  
smooth\_dim, 105  
smooth\_estimates, 106  
smooth\_estimates(), 55  
smooth\_samples, 107  
smooth\_type, 109  
smooths, 102

stats::family(), 60  
stats::glm(), 73  
stats::predict(), 6, 9  
stats::residuals(), 7, 8  
stop\_if\_not\_mgcv\_smooth  
    (is\_mgcv\_smooth), 71

term\_names, 112  
term\_variables, 112  
theta, 113  
tidy\_basis, 114  
to\_na, 116  
too\_far, 115  
too\_far\_to\_na, 115  
transform\_fun, 116  
typical\_values, 117

variance\_comp, 118  
vars\_from\_label, 118

which\_smooths, 119  
worm\_plot, 120

zooplankton, 122