Package ‘gratia’

March 17, 2022

Version 0.7.2
Date 2022-03-17
Title Graceful 'ggplot'-Based Graphics and Other Functions for GAMs Fitted Using 'mgcv'
Maintainer Gavin L. Simpson <ucfagls@gmail.com>
Depends R (>= 3.6.0)
Imports mgcv, ggplot2, tibble, dplyr (>= 1.0.0), tidyr, rlang,
    patchwork, vctrs, grid, mvnfast, purrr, stats, tools,
    grDevices, stringr, tidyselect, lifecycle, nlme
Suggests gamm4, lme4, testthat, vdiffr, MASS, scam, datasets, withr,
    knitr, rmarkdown
Description Graceful 'ggplot'-based graphics and utility functions for working with generalized additive models (GAMs) fitted using the 'mgcv' package. Provides a reimplementation of the plot() method for GAMs that 'mgcv' provides, as well as 'tidyverse' compatible representations of estimated smooths.
License MIT + file LICENSE
LazyData true
URL https://gavinsimpson.github.io/gratia/
BugReports https://github.com/gavinsimpson/gratia/issues
RoxygenNote 7.1.2
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 2022-03-17 21:10:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_confint</td>
<td>4</td>
</tr>
<tr>
<td>add_constant</td>
<td>4</td>
</tr>
<tr>
<td>add_fitted</td>
<td>5</td>
</tr>
<tr>
<td>add_fitted.gam</td>
<td>6</td>
</tr>
<tr>
<td>add_partial_residuals</td>
<td>7</td>
</tr>
<tr>
<td>add_residuals</td>
<td>8</td>
</tr>
<tr>
<td>add_residuals.gam</td>
<td>8</td>
</tr>
<tr>
<td>appraise</td>
<td>9</td>
</tr>
<tr>
<td>basis</td>
<td>11</td>
</tr>
<tr>
<td>bird_move</td>
<td>12</td>
</tr>
<tr>
<td>check_user_select_smooths</td>
<td>12</td>
</tr>
<tr>
<td>coef.scam</td>
<td>13</td>
</tr>
<tr>
<td>compare_smooths</td>
<td>14</td>
</tr>
<tr>
<td>confint.fderiv</td>
<td>15</td>
</tr>
<tr>
<td>confint.gam</td>
<td>16</td>
</tr>
<tr>
<td>data_combos</td>
<td>18</td>
</tr>
<tr>
<td>data_sim</td>
<td>19</td>
</tr>
<tr>
<td>data_slice</td>
<td>20</td>
</tr>
<tr>
<td>derivatives</td>
<td>21</td>
</tr>
<tr>
<td>difference_smooths</td>
<td>23</td>
</tr>
<tr>
<td>draw</td>
<td>24</td>
</tr>
<tr>
<td>draw.compare_smooths</td>
<td>25</td>
</tr>
<tr>
<td>draw.derivatives</td>
<td>25</td>
</tr>
<tr>
<td>draw.difference_smooth</td>
<td>26</td>
</tr>
<tr>
<td>draw.evaluated_smooth</td>
<td>28</td>
</tr>
<tr>
<td>draw.gam</td>
<td>32</td>
</tr>
<tr>
<td>draw.mgcv_smooth</td>
<td>35</td>
</tr>
<tr>
<td>draw.parametric_effects</td>
<td>37</td>
</tr>
<tr>
<td>draw.penalty_df</td>
<td>38</td>
</tr>
<tr>
<td>draw.rootogram</td>
<td>39</td>
</tr>
<tr>
<td>draw.smooth_estimates</td>
<td>41</td>
</tr>
<tr>
<td>draw.smooth_samples</td>
<td>43</td>
</tr>
<tr>
<td>edf</td>
<td>45</td>
</tr>
<tr>
<td>evaluate_parametric_term</td>
<td>46</td>
</tr>
<tr>
<td>evaluate_smooth</td>
<td>47</td>
</tr>
<tr>
<td>eval_smooth</td>
<td>49</td>
</tr>
<tr>
<td>factor_combos</td>
<td>51</td>
</tr>
<tr>
<td>family.gam</td>
<td>52</td>
</tr>
<tr>
<td>family_name</td>
<td>53</td>
</tr>
<tr>
<td>family_type</td>
<td>53</td>
</tr>
<tr>
<td>fitted_samples</td>
<td>54</td>
</tr>
<tr>
<td>fitted_values</td>
<td>55</td>
</tr>
<tr>
<td>fixef</td>
<td>57</td>
</tr>
<tr>
<td>fixef.gam</td>
<td>57</td>
</tr>
<tr>
<td>fix_offset</td>
<td>58</td>
</tr>
<tr>
<td>get_by_smooth</td>
<td>59</td>
</tr>
</tbody>
</table>
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_smooth</td>
<td>59</td>
</tr>
<tr>
<td>get_smooths_by_id</td>
<td>60</td>
</tr>
<tr>
<td>gss_vocab</td>
<td>60</td>
</tr>
<tr>
<td>gw_f0</td>
<td>61</td>
</tr>
<tr>
<td>has_theta</td>
<td>61</td>
</tr>
<tr>
<td>is_by_smooth</td>
<td>62</td>
</tr>
<tr>
<td>is_factor_term</td>
<td>63</td>
</tr>
<tr>
<td>is_mgcv_smooth</td>
<td>64</td>
</tr>
<tr>
<td>is_offset</td>
<td>64</td>
</tr>
<tr>
<td>link</td>
<td>65</td>
</tr>
<tr>
<td>load_mgcv</td>
<td>67</td>
</tr>
<tr>
<td>model_concurvity</td>
<td>67</td>
</tr>
<tr>
<td>nb_theta</td>
<td>68</td>
</tr>
<tr>
<td>n_smooths</td>
<td>69</td>
</tr>
<tr>
<td>observed_fitted_plot</td>
<td>70</td>
</tr>
<tr>
<td>parametric_effects</td>
<td>70</td>
</tr>
<tr>
<td>parametric_terms</td>
<td>71</td>
</tr>
<tr>
<td>partial_residuals</td>
<td>72</td>
</tr>
<tr>
<td>penalty</td>
<td>73</td>
</tr>
<tr>
<td>posterior_samples</td>
<td>74</td>
</tr>
<tr>
<td>predicted_samples</td>
<td>76</td>
</tr>
<tr>
<td>qq_plot</td>
<td>77</td>
</tr>
<tr>
<td>ref_sims</td>
<td>80</td>
</tr>
<tr>
<td>rep_first_factor_value</td>
<td>80</td>
</tr>
<tr>
<td>residuals_hist_plot</td>
<td>81</td>
</tr>
<tr>
<td>residuals_linpred_plot</td>
<td>81</td>
</tr>
<tr>
<td>rootogram</td>
<td>82</td>
</tr>
<tr>
<td>seq_min_max</td>
<td>83</td>
</tr>
<tr>
<td>seq_min_max_eps</td>
<td>84</td>
</tr>
<tr>
<td>shift_values</td>
<td>85</td>
</tr>
<tr>
<td>simulate.gam</td>
<td>85</td>
</tr>
<tr>
<td>smallAges</td>
<td>86</td>
</tr>
<tr>
<td>smooths</td>
<td>87</td>
</tr>
<tr>
<td>smooth_coefs</td>
<td>88</td>
</tr>
<tr>
<td>smooth_data</td>
<td>88</td>
</tr>
<tr>
<td>smooth_dim</td>
<td>89</td>
</tr>
<tr>
<td>smooth_estimates</td>
<td>90</td>
</tr>
<tr>
<td>smooth_samples</td>
<td>91</td>
</tr>
<tr>
<td>term_names</td>
<td>94</td>
</tr>
<tr>
<td>term_variables</td>
<td>94</td>
</tr>
<tr>
<td>theta</td>
<td>95</td>
</tr>
<tr>
<td>tidy_basis</td>
<td>96</td>
</tr>
<tr>
<td>too_far</td>
<td>96</td>
</tr>
<tr>
<td>too_far_to_na</td>
<td>97</td>
</tr>
<tr>
<td>to_na</td>
<td>97</td>
</tr>
<tr>
<td>transform_fun</td>
<td>98</td>
</tr>
<tr>
<td>typical_values</td>
<td>99</td>
</tr>
<tr>
<td>variance_comp</td>
<td>99</td>
</tr>
</tbody>
</table>
add_confint

Add a confidence interval to an existing object

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 < coverage < 1.
- **...**: arguments passed to other methods.

add_constant

Add a constant to estimated values

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`  
*Add fitted values from a model to a data frame*

### Description

Add fitted values from a model to a data frame

### Usage

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

### Arguments

- **data**: a data frame containing values for the variables used to fit the model. Passed to `stats::predict()` as newdata.
- **model**: a fitted model for which a `stats::predict()` 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.
- **...**: additional arguments passed to methods.
Value

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

Description

Add fitted values from a GAM to a data frame

Usage

```r
## 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 `stats::predict()` as `newdata`.
- `model`: a fitted model for which a `stats::predict()` 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 predictions to return. See `mgcv::predict.gam()` 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 `mgcv::predict.gam()`.

Value

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

Examples

```r
load_mgcv()

df <- gamSim(eg = 1, verbose = FALSE)
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)
```
```r
## with type = "terms" or "iterms"
add_fitted(df, m, type = "terms")
```

---

### Description

Add partial residuals

### Usage

```r
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 `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.
- `...` 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

```r
library(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

Add residuals from a model to a data frame

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 `stats::predict()` as newdata.
- **model**: a fitted model for which a `stats::predict()` 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 `mgcv::residuals.gam()` for options.
- **...**: additional arguments passed to `mgcv::residuals.gam()`.
Value

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

Examples

load_mgcv()

df <- gamSim(eg = 1, verbose = FALSE)
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".
- **method**: character; method used to generate theoretical quantiles. Note that \texttt{method = "direct"} is deprecated in favour of \texttt{method = "uniform"}.
- **n_uniform**: numeric; number of times to randomize uniform quantiles in the direct computation method (\texttt{method = "direct"}) for QQ plots.
- **n_simulate**: numeric; number of data sets to simulate from the estimated model when using the simulation method (\texttt{method = "simulate"}) for QQ plots.
- **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.
- **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 \texttt{patchwork::plot_layout()}
- **level**: numeric; the coverage level for QQ plot reference intervals. Must be strictly \(0 < \text{level} < 1\). Only used with \texttt{method = "simulate"}.
- **ci_alpha, ci_col**: numeric; the level of alpha transparency for the QQ plot reference interval when \texttt{method = "simulate"}, or points drawn in plots.
- **point_col, point_alpha**: colour and transparency used to draw points in the plots. See \texttt{graphics::par()} section \textbf{Color Specification}. This is passed to the individual plotting functions, and therefore affects the points of all plots.
- **line_col**: 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 \texttt{mgcv::qq.gam()} uses \textit{direct} in reference to the simulated residuals method (\texttt{method = "simulated"}). To avoid confusion, \texttt{method = "direct"} is deprecated in favour of \texttt{method = "uniform"}.

See Also

The plots are produced by functions \texttt{qq_plot()}, \texttt{residuals_linpred_plot()}, \texttt{residuals_hist_plot()}, and \texttt{observed_fitted_plot()}.

Examples

```r
load_mgcv()
## simulate some data...
dat <- gamSim(1, n = 400, dist = "normal", scale = 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

**Basis expansions for smooths**

**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().

**Usage**

`basis(smooth, data, knots = NULL, constraints = FALSE, at = NULL, ...)`

**Arguments**

- **smooth**
  - a smooth specification, the result of a call to one of mgcv::s(), mgcv::te(), mgcv::ti(), or mgcv::t2().

- **data**
  - a data frame containing the variables used in smooth.

- **knots**
  - a list or data frame with named components containing knots locations. Names must match the covariates for which the basis is required. See mgcv::smoothCon().

- **constraints**
  - logical; should identifiability constraints be applied to the smooth basis. See argument absorb.cons in mgcv::smoothCon().

- **at**
  - a data frame containing values of the smooth covariate(s) at which the basis should be evaluated.

- **...**
  - other arguments passed to mgcv::smoothCon().

**Value**

A tibble.

**Author(s)**

Gavin L. Simpson
check_user_select_smooths

**Examples**

```r
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)
```

<table>
<thead>
<tr>
<th>bird_move</th>
<th>Simulated bird migration data</th>
</tr>
</thead>
</table>

**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 simulate 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**


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.
coef.scam

Usage

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

Arguments

smooths character; a vector of smooth labels.
select numeric, logical, or character vector of selected smooths.
partial_match logical; in the case of character select, should select match partially against
  smooths? If partial_match = TRUE, select must only be a single string, a
  character vector of length 1.
model_name 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

---

c 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**  

**Compare smooths across models**

**Description**

Compare smooths across models

**Usage**

```r
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**

```r
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")
```
confint.fderiv

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

---

**confint.fderiv**  
*Point-wise and simultaneous confidence intervals for derivatives of smooths*

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

### Usage

```r
## 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 < 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 `mvnfast::rmvn()`. 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 <- gamSim(1, n = 1000, dist = "normal", scale = 2)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

## first derivatives of all smooths...
fd <- fderiv(mod)

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

## simultaneous interval for smooth term of x1
x1.sint <- confint(fd, parm = "x1", type = "simultaneous", nsim = 2500)
head(x1.sint)

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,
  newdata = NULL,
  n = 200,
)
```r
confint.gam

type = c("confidence", "simultaneous"),
nsim = 10000,
shift = FALSE,
transform = FALSE,
unconditional = FALSE,
ncores = 1,
partial_match = FALSE,
...

## 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 < level < 1; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.

data data frame; containing 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 `mvnfast::rmvn()` 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 length(parm) is 1.

... additional arguments for methods
```
data_combos

Value

a data frame with components:

1. term; factor indicating to which term each row relates,
2. x; the vector of values at which the smooth was evaluated,
3. lower; lower limit of the confidence or simultaneous interval,
4. est; estimated value of the smooth
5. upper; upper limit of the confidence or simultaneous interval,
6. crit; critical value for the 100 * level% confidence interval.

Author(s)

Gavin L. Simpson

Examples

load_mgcv()

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

## point-wise interval
ci <- confint(mod, parm = "s(x1)", type = "confidence")

## simultaneous interval for smooth term of x1
si <- confint(mod, parm = "s(x1)", type = "simultaneous", nsim = 100)

---

data_combos

All combinations of factor levels plus typical values of continuous variables

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**

*Simulate example data for fitting GAMs*

**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**

```r
data_sim(
  model = "eg1",
  n = 400,
  scale = 2,
  dist = c("normal", "poisson", "binary"),
  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.
- **dist**: character; a sampling distribution for the response variable.
- **seed**: numeric; the seed for the random number generator. Passed to `base::set.seed()`.

**Examples**

```r
data_sim("eg1")
```
Prepare a data slice through covariates

## Description

Prepare a data slice through covariates

## Usage

data_slice(object, ...)

## S3 method for class 'gam'
data_slice(
  object,
  var1,
  var2 = NULL,
  var3 = NULL,
  var4 = NULL,
  data = NULL,
  n = 50,
  offset = NULL,
  ...
)

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

## Arguments

- **object**: an R model object.
- **...**: arguments passed to other methods.
- **var1**: character;
- **var2**: character;
- **var3**: character; ignored currently.
- **var4**: character; ignored currently.
- **data**: a 1-row data frame or tibble containing values for variables in the fitted model that are not varying in the slice.
- **n**: numeric; the number of values to create for each of var1 and var2 in the slice.
- **offset**: numeric; value to use for an offset term in the model.
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,
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,
...)

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 partial_match below.
newdata a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>numeric; the order of derivative.</td>
</tr>
<tr>
<td>type</td>
<td>character; the type of finite difference used. One of &quot;forward&quot;, &quot;backward&quot;, or &quot;central&quot;.</td>
</tr>
<tr>
<td>n</td>
<td>numeric; the number of points to evaluate the derivative at.</td>
</tr>
<tr>
<td>eps</td>
<td>numeric; the finite difference.</td>
</tr>
<tr>
<td>interval</td>
<td>character; the type of interval to compute. One of &quot;confidence&quot; for point-wise intervals, or &quot;simultaneous&quot; for simultaneous intervals.</td>
</tr>
<tr>
<td>n_sim</td>
<td>integer; the number of simulations used in computing the simultaneous intervals.</td>
</tr>
<tr>
<td>level</td>
<td>numeric; 0 &lt; level &lt; 1; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.</td>
</tr>
<tr>
<td>unconditional</td>
<td>logical; use smoothness selection-corrected Bayesian covariance matrix?</td>
</tr>
<tr>
<td>frequentist</td>
<td>logical; use the frequentist covariance matrix?</td>
</tr>
<tr>
<td>offset</td>
<td>numeric; a value to use for any offset term</td>
</tr>
<tr>
<td>ncores</td>
<td>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).</td>
</tr>
<tr>
<td>partial_match</td>
<td>logical; should smooths be selected by partial matches with term? If TRUE, term can only be a single string to match against.</td>
</tr>
</tbody>
</table>

**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 derivative ± (crit * 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.

**Author(s)**

Gavin L. Simpson

**Examples**

```r
load_mgcv()

dat <- gamSim(1, n = 400, dist = "normal", scale = 2, verbose = FALSE)
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**

Differences of factor smooth interactions

**Usage**

difference_smooths(model, ...)

## S3 method for class 'gam'
difference_smooths(
  model,
  smooth,
  n = 100,
  ci_level = 0.95,
  newdata = NULL,
  partial_match = TRUE,
  unconditional = FALSE,
  frequentist = FALSE,
  ...
)

**Arguments**

- **model**: A fitted model.
- **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.
- **newdata**: data frame of locations at which to evaluate the difference between smooths.
- **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")

difference_smooths(m, smooth = "s(x2)")

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.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 `compare_smooths()`.
- ncol: numeric; the numbers of rows and columns over which to spread the plots
- 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 `patchwork::plot_layout()`.
- ...: additional arguments passed to `patchwork::wrap_plots()`.

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",
  ...
)

draw.difference_smooth

Plot differences of smooths

Description

Plot differences of smooths

Usage

## S3 method for class 'difference_smooth'
draw(
  object,
  select = NULL,
...
draw.difference_smooth

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"),
col = NULL,
nrow = NULL,
guides = "keep",
xlab = NULL,
ylab = NULL,
title = NULL,
subtitle = NULL,
caption = NULL,
...
)

Arguments

object a fitted GAM, the result of a call to mgcv::gam().

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.

rug 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 ggplot2::geom_contour().
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? The default, scales = "fixed", ensures this is done. If scales = "free" each univariate smooth has its own y-axis scale. Currently does not affect the y-axis scale of plots of the parametric terms.
draw.evaluated_smooth

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 `patchwork::plot_layout()`
xlab, ylab, title, subtitle, caption character; labels with which to annotate plots
... additional arguments passed to `patchwork::wrap_plots()`.

Examples

```r
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)
```

## 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(

Description

Plots estimated univariate and bivariate smooths using ggplot2.

Usage

```r
## 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(

```
draw.evaluated_smooth

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'

```r
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 `evaluate_smooth()`.
- **rug**: For `evaluate_smooth()`, a numeric vector of values for the location of data on the x axis. The default of `NULL` results in no rug plot being drawn. For `evaluate_parametric_terms()`, 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. `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.
- **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 `ggplot2::labs()`.
- **subtitle**: character or expression; the subtitle for the plot. See `ggplot2::labs()`.
- **caption**: character or expression; the plot caption. See `ggplot2::labs()`.
- **partial_residuals**: data frame; partial residuals and data values if partial residuals are drawn. Should have names `.p_resid` and `.orig_x` if supplied.
- **response_range**: 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 `show` is set to "se".
- **...**: arguments passed to other methods.
- **show**: character; plot the estimated smooth ("estimate") or its standard error ("se").
draw.evaluation_smooth

contour logical; should contours be drawn on the plot using \texttt{ggplot2::geom_contour()}.

contour.col colour specification for contour lines.

n.contour numeric; the number of contour bins. Will result in \(n_{\text{contour}} - 1\) contour lines being drawn. See \texttt{ggplot2::geom_contour()}.

continuous_fill suitable scale used for the filled surface. If \texttt{NULL}, the default used is \texttt{scale_fill_distiller(palette = "RdBu", type = "div")}.

qq.line logical; draw a reference line through the lower and upper theoretical quartiles.

discrete.colour an appropriate discrete colour scale from \texttt{ggplot2}. 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 \texttt{ggplot2::scale_colour_viridis_d()}.}

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

Value

A \texttt{ggplot2::ggplot()} object.

Author(s)

Gavin L. Simpson

Examples

\begin{verbatim}
load_mgcv()

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

## supply constant to shift y axis scale
draw(sm, constant = coef(ml)[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)
\end{verbatim}
**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**

```r
## 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_colours = NULL,
  continuous_colour = NULL,
  continuous_fill = NULL,
  position = "identity",
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  projection = "orthographic",
  orientation = NULL,
  ...
)```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>a fitted GAM, the result of a call to <code>mgcv::gam()</code></td>
</tr>
<tr>
<td>data</td>
<td>a optional data frame that may or may not be used? FIXME!</td>
</tr>
<tr>
<td>select</td>
<td>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.</td>
</tr>
<tr>
<td>parametric</td>
<td>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.</td>
</tr>
<tr>
<td>terms</td>
<td>character; which model parametric terms should be drawn? The Default of NULL will plot all parametric terms that can be drawn.</td>
</tr>
<tr>
<td>residuals</td>
<td>logical; should partial residuals for a smooth be drawn? Ignored for anything but a simple univariate smooth.</td>
</tr>
<tr>
<td>scales</td>
<td>character; should all univariate smooths be plotted with the same y-axis scale? The default, scales = &quot;fixed&quot;, ensures this is done. If scales = &quot;free&quot; each univariate smooth has its own y-axis scale. Currently does not affect the y-axis scale of plots of the parametric terms.</td>
</tr>
<tr>
<td>ci_level</td>
<td>numeric between 0 and 1; the coverage of credible interval.</td>
</tr>
<tr>
<td>n</td>
<td>numeric; the number of points over the range of the covariate at which to evaluate the smooth.</td>
</tr>
<tr>
<td>n_3d</td>
<td>numeric; the number of new observations to generate for the third dimension of a 3D smooth.</td>
</tr>
<tr>
<td>n_4d</td>
<td>numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a kD smooth (k &gt;= 4). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get n_4d new observations.</td>
</tr>
<tr>
<td>unconditional</td>
<td>logical; should confidence intervals include the uncertainty due to smoothness selection? If TRUE, the corrected Bayesian covariance matrix will be used.</td>
</tr>
<tr>
<td>overall_uncertainty</td>
<td>logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?</td>
</tr>
<tr>
<td>constant</td>
<td>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.</td>
</tr>
<tr>
<td>fun</td>
<td>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.</td>
</tr>
</tbody>
</table>
**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 `mgcv::exclude.too.far()` for further details.

**rug**
logical; draw a rug plot at the bottom of each plot?

**contour**
logical; should contours be drawn on the plot using `ggplot2::geom_contour()`.

**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 \( n_{\text{contour}} - 1 \) contour lines being drawn. See `ggplot2::geom_contour()`.

**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, 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.

**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 `patchwork::plot_layout()`

**projection**
character; projection to use, see `mapproj::mapproject()` for a list.

**orientation**
an optional vector c(latitude, longitude, rotation) 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 orientation therefore are `c(20, 0, mean(range(longitude)))` if this is not specified by the user. See `mapproj::mapproject()` for more information.

**...**
additional arguments passed to `patchwork::wrap_plots()`.

---

**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

```r
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"))

# to modify all panels, for example to change the theme, use the & operator
draw(m2, n_contour = 5, n = 50) &
   theme_minimal()

# customising some plot elements
draw(m1, ci_col = "steelblue", smooth_col = "forestgreen", ci_alpha = 0.3)
```

Description

Plots basis functions using ggplot2

Usage

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

- **object**: an object, the result of a call to `basis()`.  
- **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 `patchwork::wrap_plots()`. Currently ignored.  
- **labeller**: a labeller function with which to label facets. The default is to use `ggplot2::label_both()`.  
- **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 `ggplot2::labs()`.  
- **subtitle**: character or expression; the subtitle for the plot. See `ggplot2::labs()`.  
- **caption**: character or expression; the plot caption. See `ggplot2::labs()`.  
- **...**: arguments passed to other methods. Not used by this method.

Value

A `ggplot2::ggplot()` object.

Author(s)

Gavin L. Simpson

Examples

```r
load_mgcv()

df <- gamSim(4, n = 400, verbose = FALSE)

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 = 0,
  fun = NULL,
  rug = TRUE,
  position = "identity",
  ...
)

Arguments

object a fitted GAM, the result of a call to `mgcv::gam`.
scales character; should all univariate smooths be plotted with the same y-axis scale? The default, scales = "fixed", ensures this is done. If scales = "free" each univariate smooth has its own y-axis scale. 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. 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.

rug logical; draw a rug plot at the bottom of each plot?

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

... additional arguments passed to `patchwork::wrap_plots()`.

ncol numeric; the numbers of rows and columns over which to spread the plots

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 `patchwork::plot_layout()`

draw.penalty_df  Display penalty matrices of smooths using ggplot

Description

Displays the penalty matrices of smooths as a heatmap using ggplot

Usage

```r
## 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 `evaluate_smooth()`.
- `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 `scale_fill_distiller(palette = "RdBu", type = "div")`.
- `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.
draw.rootogram

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 \texttt{ggplot2::ggplot()} graphics. The design closely follows Kleiber & Zeileis (2016).

Usage

\begin{verbatim}
## S3 method for class 'rootogram'
draw(
    object,
    type = c("hanging", "standing", "suspended"),
    sqrt = TRUE,
    ref_line = TRUE,
\end{verbatim}
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


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.smooth_estimates

\begin{verbatim}
draw(rg)

# change the type of rootogram
draw(rg, type = "suspended")
\end{verbatim}

\begin{description}
\item[draw.smooth_estimates] Plot the result of a call to smooth_estimates()
\end{description}

\section*{Description}
Plot the result of a call to smooth_estimates()

\section*{Usage}
\begin{verbatim}
## S3 method for class 'smooth_estimates'
draw(
oobject, 
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, 
continuous_colour = NULL, 
continuous_fill = NULL, 
ylim = NULL, 
projection = "orthographic", 
orientation = NULL, 
...)
\end{verbatim}

\section*{Arguments}
\begin{description}
\item[object] a fitted GAM, the result of a call to \texttt{mgcv::gam}.
\item[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.
\item[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 \texttt{fun} will be applied after adding any constant, if provided.
\item[contour] logical; should contours be draw on the plot using \texttt{ggplot2::geom_contour}.
\end{description}
draw.smooth_estimates

countour_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 ggplot2::geom_contour().
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 select? If TRUE, select can only be a single string to match against.
discrete_colour suitable scales for the types of data.
continuous_colour suitable scales for the types of data.
continuous_fill suitable scales for the types of data.
ylim            numeric; vector of y axis limits to use all all panels drawn.
projection      character; projection to use, see mapproj::mapproject() for a list.
orientation     an optional vector c(latitude,longitude,rotation) 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 orientation therefore are ‘c(20, 0, mean(range(longitude)))” if this is not specified by the user. See mapproj::mapproject() for more information.
...             additional arguments passed to patchwork::wrap_plots().

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

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,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  ...
)

Arguments

object a fitted GAM, the result of a call to mgcv::gam().
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.
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 `ggplot2::labs()`.

subtitle character or expression; the subtitle for the plot. See `ggplot2::labs()`.

caption character or expression; the plot caption. See `ggplot2::labs()`.

alpha numeric; alpha transparency for confidence or simultaneous interval.

colour The colour to use to draw the posterior smooths. Passed to `ggplot2::geom_line()` 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 `ggplot2::geom_contour()`.

scales character; should all univariate smooths be plotted with the same y-axis scale? The default, `scales = "fixed"`, ensures this is done. If `scales = "free"` each univariate smooth has its own y-axis scale. Currently does not affect the y-axis scale of plots of the parametric terms.

rug logical; draw a rug plot at the bottom of each plot?

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

ncol numeric; the numbers of rows and columns over which to spread the plots

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 `patchwork::plot_layout()` arguments to be passed to `patchwork::wrap_plots()`.

... arguments to be passed to `patchwork::wrap_plots()`.

Author(s)

Gavin L. Simpson

Examples

```r
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
sml1 <- smooth_samples(m1, n = 15, seed = 23478)
## plot
draw(sml1, alpha = 0.7)
## plot only 5 randomly smapled draws
draw(sml1, 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**

```r
edf(object, ...)
```

**## S3 method for class 'gam'**

```r
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.
evaluate_parametric_term

**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

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

where $\text{tr}$ is the matrix trace and $F$ is a matrix mapping un-penalized coefficient estimates to the penalized coefficient estimates. The trace of $F$ 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 $\text{diag}(2F - FF)$.

**Examples**

```r
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*
evaluate_smooth

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 mgcv::exclude.too.far() 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 <- gamSim(1, n = 400, dist = "normal", scale = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")
evaluate_smooth(m1, "s(x1)")

## 2d example
dat <- gamSim(2, n = 1000, dist = "normal", scale = 1)
m2 <- gam(y ~ s(x, z, k = 30), data = dat$data, method = "REML")
evaluate_smooth(m2, "s(x,z)", n = 100)

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 'random.effect'

```r
eval_smooth(
    smooth,
    model,
    n = 100,
    data = NULL,
    unconditional = FALSE,
    overall_uncertainty = TRUE,
    ...
)
```

## S3 method for class 'mrf.smooth'

```r
eval_smooth(
    smooth,
    model,
    n = 100,
    data = NULL,
    unconditional = FALSE,
    overall_uncertainty = TRUE,
    ...
)
```

## S3 method for class 't2.smooth'

```r
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'

```r
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 `mgcv::smooth`.
- **...** arguments assed to other methods.
- **model**: a fitted model; currently only `mgcv::gam()` and `mgcv::bam()` models are supported.
- **n**: numeric; the number of points over the range of the covariate at which to evaluate the smooth.
- **n_3d**: 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.
- **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 `mgcv::exclude.too.far()` for further details.

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 Extract family objects from models

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 mgcv::gam(), mgcv::bam(), mgcv::gamm(), and gamm4::gamm4() are currently supported.
... arguments passed to other methods.
family_name

Name of family used to fit model

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

Extracts the type of family in a consistent way

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  

**Draw fitted values from the posterior distribution**

**Description**

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

**Usage**

```r
fitted_samples(model, ...)  
```

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

fitted_samples(
  model,
  n = 1,
  newdata,
  seed,
  scale = c("response", "linear_predictor"),
  method = c("gaussian", "mh", "inla"),
  freq = FALSE,
  unconditional = FALSE,
  ncores = 1L,
  ...
)
```

**Arguments**

- `model` a fitted model of the supported types
- `...` arguments passed to other methods. For `fitted_samples()`, these are passed on to `predict.gam()`.
- `n` numeric; the number of posterior samples to return.
- `newdata` 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 `model`.
- `seed` numeric; a random seed for the simulations.
- `scale` 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".
- `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 freq == FALSE) 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 `mvnfast::rmvn()`. Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).

Value

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

- row (integer) the row of newdata 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 newdata.

Author(s)

Gavin L. Simpson

References


Examples

```r
load_mgcv()
dat <- gamSim(1, n = 1000, dist = "normal", scale = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")
fitted_samples(m1, n = 5, seed = 42)
```

---

**fitted_values**

Generate fitted values from a estimated GAM

Description

Generate fitted values from a estimated GAM
fitted_values

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 `mgcv::gam()` and `mgcv::bam()` are supported.
...
arguments passed to `mgcv::predict.gam()`. Note that type, newdata, and se.fit are already used and passed on to `mgcv::predict.gam()`.
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

object a fitted GAM
...

fixef.gam

Extract fixed effects estimates from a fitted GAM

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

object a fitted GAM
...
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

Fix the names of a data frame containing an offset variable.

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

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

Value

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

Author(s)

Gavin L. Simpson

Examples

load_mgcv()

df <- gamSim(1, n = 400, dist = "normal")
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

Extract an factor-by smooth by name

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

Extract an mgcv smooth by name

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  
*Extract an mgcv smooth given its position in the model object*

**Description**

Extract an mgcv smooth given its position in the model object

**Usage**

```r
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**  
*Data from the General Social Survey (GSS) from the National Opinion Research Center of the University of Chicago*

**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+`.##'
**Gu and Wabha test functions**

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

**has_theta**

**Description**
Are additional parameters available for a GAM?

**Usage**
```r
has_theta(object)
```

**Arguments**
- `object`: an R object, either a `family()` object or an object whose class has a `family()` method.
is_by_smooth

Value

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

Examples

```r
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)
```

Description

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

Usage

```r
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

Is a model term a factor (categorical)?

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

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

## S3 method for class 'terms'

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

## S3 method for class 'gam'

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

## S3 method for class 'bam'

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

## S3 method for class 'gamm'

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

## S3 method for class 'list'

```r
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

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

Description

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

Usage

is_mgcv_smooth(smooth)

is_mrf_smooth(smooth)

Arguments

smooth

an R object, typically a list

is_offset

Is a model term an offset?

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 <- gamSim(1, 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 `stats::glm()`, `mgcv::gam()`, `mgcv::bam()`, `mgcv::gamm()`, and `gamm4::gamm4()` 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 `mgcv::gaulss()", "pi" for the zero-inflation term in `mgcv::ziplss()", "power" for the `mgcv::twlss()` power parameter, "xi", the shape parameter for `mgcv::gevlss()", "epsilon" or "skewness" for the skewness and "delta" or "kurtosis" for the kurtosis parameter for `mgcv::shash()", or "theta" for the scale parameter of `mgcv::gammals()`.
- **which_eta**: numeric; the linear predictor to extract for families `mgcv::mvn()` and `mgcv::multinom()`.
- **family**: a family object, the result of a call to `family()`.
- **inverse**: logical; return the inverse of the link function?

### Author(s)

Gavin L. Simpson

### Examples

```r
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

**Load mgcv quietly**

#### Description

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

#### Usage

```r
load_mgcv()
```

#### Value

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

---

### model_concurvity

**Concurvity of an estimated GAM**

#### Description

Concurvity of an estimated GAM

#### Usage

```r
model_concurvity(model, ...)
```

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

```r
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 concavity of model terms?

Examples

## simulate data with concavity...
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 concavity
o_conc <- concrvity(m)
draw(o_conc)

## pairwise concavity
p_conc <- concrvity(m, pairwise = TRUE)
draw(p_conc)

nb_theta

Negative binomial parameter theta

Description

Negative binomial parameter theta

Usage

nb_theta(model)

## S3 method for class 'gam'
between(model)

Arguments

model  a fitted model.
Value

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

Methods (by class)

- `gam`: Method for class "gam"

Examples

```r
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

<table>
<thead>
<tr>
<th>n_smooths</th>
<th>How many smooths in a fitted model</th>
</tr>
</thead>
</table>

Description

How many smooths in a fitted model

Usage

```r
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   Plot of fitted against observed response values

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 ggplot2::labs()..
subtitle        character or expression; the subtitle for the plot. See ggplot2::labs()..
caption         character or expression; the plot caption. See ggplot2::labs()..
point_col       colour used to draw points in the plots. See graphics::par() section Color Specification. 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.

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.

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.

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 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
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
penalty

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**

```r
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, *mgcv* will scale the penalty matrix for better performance in *mgcv::gamm()*. If **rescale** 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 nth basis function, referencing the columns of the penalty matrix,
- col - character; a label of the form fn where n is an integer for the nth 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.

Author(s)

Gavin L. Simpson

Examples

load_mgcv()

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

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

Usage

posterior_samples(model, ...)  

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

Arguments

model  a fitted model of the supported types

...  arguments passed to other methods. For fitted_samples(), these are passed on to predict.gam().

n  numeric; the number of posterior samples to return.

newdata  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 model.

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 freq == FALSE) then the Bayesian smoothing parameter uncertainty corrected covariance matrix is used, if available.

weights  numeric; a vector of prior weights. If newdata is null then defaults to object[['prior.weights']], otherwise a vector of ones.

ncores  number of cores for generating random variables from a multivariate normal distribution. Passed to mvnfast::rmvn(). Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current R).

Value

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

- row (integer) the row of newdata 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 newdata.
predicted_samples

**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**

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

**Arguments**

- `model` a fitted model of the supported types
- `...` arguments passed to other methods. For `predicted_samples()`, these are passed on to `predict.gam()`.
- `n` numeric; the number of posterior samples to return.
- `newdata` 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 `model`.
- `seed` numeric; a random seed for the simulations.
- `weights` numeric; a vector of prior weights. If `newdata` is null then defaults to `object[['prior.weights']]`, otherwise a vector of ones.
Value

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

- row (integer) the row of newdata 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 newdata.

Author(s)

Gavin L. Simpson

Examples

load_mgcv()

dat <- gamSim(1, n = 1000, dist = "normal", scale = 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 same as previous
predicted_samples(m, n = 5, newd, seed = 25,
                  terms = c("s(x0)", "s(x2)", "s(x3)"))

------

qq_plot  Quantile-quantile plot of model residuals

Description

Quantile-quantile plot of model residuals
qq_plot

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".
...
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 < level < 1. Only used with method = "simulate".
**qq_plot**

- **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 x axis. If not supplied, a suitable label will be generated.
- **title** character or expression; the title for the plot. See `ggplot2::labs()`.
- **subtitle** character or expression; the subtitle for the plot. See `ggplot2::labs()`.
- **caption** character or expression; the plot caption. See `ggplot2::labs()`.
- **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**

```r
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_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

```r
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

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 ggplot2::labs().

subtitle character or expression; the subtitle for the plot. See ggplot2::labs().

caption character or expression; the plot caption. See ggplot2::labs().

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".

Arguments

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 ggplot2::labs().

subtitle character or expression; the subtitle for the plot. See ggplot2::labs().

caption character or expression; the plot caption. See ggplot2::labs().

point_col colour used to draw points in the plots. See graphics::par() section Color Specification. 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.

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).
seq_min_max

Create a sequence of evenly-spaced values

Description

For a continuous vector \( x \), \texttt{seq_min_max()} creates a sequence of \( n \) evenly-spaced values over the range \( \min(x) - \max(x) \).
seq_min_max(x, n)

Arguments

x numeric; vector over which evenly-spaced values are returned
n numeric; the number of evenly-spaced values to return

Value

A numeric vector of length n.

Examples

x <- rnorm(10)
n <- 10L
seq_min_max(x, n = n)

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

Arguments

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

Value

A numeric vector of length n.

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 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.
**shift_values**  
*Shift numeric values in a data frame by an amount eps*

**Description**
Shift numeric values in a data frame by an amount eps

**Usage**

```r
shift_values(df, h, i, FUN = "+")
```

**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 `-`

**simulate.gam**  
*Simulate from the posterior distribution of a GAM*

**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**

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

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

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

- **object**: a fitted GAM, typically the result of a call to `mgcv::gam` or `mgcv::gamm()`.
- **nsim**: numeric; the number of posterior simulations to return.
- **seed**: numeric; a random seed for the simulations.
- **newdata**: 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 `object[['prior.weights']]`, otherwise a vector of ones.
- **...**: arguments passed to methods. `simulate.gam()` and `simulate.scam()` pass `...` on to `predict.gam()`. As such you can pass additional arguments such as `terms`, `exclude`, to select which model terms are included in the predictions. This may be useful, for example, for excluding the effects of random effect terms.

**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**

```r
load_mgcv()

dat <- gamSim(1, n = 400, dist = "normal", scale = 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.
The data frame contains 12 rows and 7 variables:

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

The data source is Simpson, G.L. (Unpublished data).

**smooths**

| smooths | Names of smooths in a GAM |

**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  
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_coefs(smooth)

Arguments
smooth  
an object that inherits from class mgcv.smooth

Value
A numeric vector of indices.

Author(s)
Gavin L. Simpson

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 kD smooth (k >= 4). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get n_4d 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 model. This can be useful if you want to pass the returned data frame on to mgcv::PredictMat().

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".
smooth_estimates

Value

A numeric vector of dimensions for each smooth.

Author(s)

Gavin L. Simpson

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

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.
smooth_samples

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 pointsm, 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 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 mgcv::exclude.too.far() 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 partial_match = TRUE, 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

Posterior draws for individual smooths

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

Usage

smooth_samples(model, ...)

## S3 method for class 'gam'
smooth_samples(
  model,
  term = NULL,
  n = 1,
  newdata = NULL,
  seed = NULL,
  freq = FALSE,
  unconditional = FALSE,
  ncores = 1L,
  n_vals = 200,
  ...
)

Arguments

model a fitted model of the supported types
...
arguments passed to other methods. For fitted_samples(), these are passed on to predict.gam().
term character; select which smooth’s posterior to draw from. The default (NULL) means the posteriors of all smooths in model will be sampled from. If supplied, a character vector of requested terms.
n numeric; the number of posterior samples to return.
newdata 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 model.
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 freq == FALSE) 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 mvnfast::rmvn(). 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 newdata not supplied

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
`newdata` 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 combina-
tion.

• `.xN`; numeric. A series of one or more columns containing data required for the smooth. `.x1`
will always be present and contains the values of the covariate in the smooth. For example
if smooth is `s(z)` then `.x1` will contain the values of covariate `z` at which the smooth was
evaluated. Further covariates for multi-dimensional thin plate splines (e.g. `s(x, z)` or tensor
product smooths (e.g. `te(x, z, a)`) will result in variables `.x1` and `.x2`, and `.x1`, `.x2`, and `.x3`
respectively, with the number (1, 2, etc) representing the order in which the covariates were
specified in the 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**

```r
load_mgcv()

dat <- data_sim("eg1", n = 1000, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")
smooth_samples(m1, term = "s(x0)", n = 5, seed = 42)

## 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)
```
term_names

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

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

Names of variables involved in a specified model term

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**  
*General extractor for additional parameters in mgcv models*

### 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
```r
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, at = NULL)
```

**Arguments**

- `smooth`: a smooth object.
- `data`: a data frame containing the variables used in `smooth`.
- `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

---

too_far  

*Exclude values that lie too far from the support of data*

**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**

- `x, y`: numeric; vector of values of the covariates to compare with the observed data
- `ref_1, ref_2`: numeric; vectors of covariate values that represent the reference against which x1 and x2' are compared
- `dist`: 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 0.1.
too_far_to_na

Value

Returns a logical vector of the same length as x1.

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

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

to_na

Sets the elements of vector to NA

Description

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

Usage

to_na(x, i)

Arguments

x vector of values
i vector of values used to subset x

Value

Returns x with possibly some elements set to NA
### transform_fun

**Transform estimated values and confidence intervals by applying a function**

#### Description

Transform estimated values and confidence intervals by applying a function.

#### Usage

```r
transform_fun(object, fun = 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

**Typical values of model covariates**

**Description**

Typical values of model covariates

**Usage**

```r
# 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

**Variance components of smooths from smoothness estimates**

**Description**

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

**Usage**

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

**Arguments**

- `object`: an R object. Currently only models fitted by `mgcv::gam()` or `mgcv::bam()` are supported.
- `...`: arguments passed to other methods
- `rescale`: logical; for numerical stability reasons the penalty matrices of smooths are rescaled before fitting. If `rescale = TRUE`, 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 `lmer()`.
- `coverage`: numeric; a value between 0 and 1 indicating the (approximate) coverage of the confidence interval that is returned.
which_smooths

Details
This function is a wrapper to \texttt{mgcv::gam.vcomp()} which performs three additional services

- it suppresses the annoying text output that \texttt{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 \hspace{1em} Returns names of variables from a smooth label

Description
Returns names of variables from a smooth label

Usage
vars_from_label(label)

Arguments
label \hspace{1em} 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 \hspace{1em} Identify a smooth term by its label

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, ...)
worm_plot

## 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.

### Description

Worm plot of model residuals

### Usage

```r
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'

```r
## 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",
...)
```
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 < 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 `ggplot2::labs()`.
- **subtitle**: character or expression; the subtitle for the plot. See `ggplot2::labs()`.
- **caption**: character or expression; the plot caption. See `ggplot2::labs()`.
- **ci_col**: fill colour and alpha transparency for the reference interval when method = "simulate".
- **ci_alpha**: fill colour and alpha transparency for the reference interval when method = "simulate".
- **point_col**: colour and alpha transparency for points on the QQ plot.
- **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

```r
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 m² for each taxon at each point in time in each sampled lake.

**Source**

References

Index

* data
  bird_move, 12
  gss_vocab, 60
  ref_sims, 80
  smallAges, 86
  zooplankton, 103
* draw methods
  draw.rootogram, 39
  add_confint, 4
  add_constant, 4
  add_fitted, 5
  add_fitted.gam, 6
  add_partial_residuals, 7
  add_residuals, 8
  add_residuals.gam, 8
  appraise, 9
  base::set.seed(), 19
  base::zapsmall(), 74
  basis, 11
  basis(), 36
  bird_move, 12
  by_level(is_by_smooth), 62
  by_variable(is_by_smooth), 62
  check_user_select_smooths, 12
  coef.scam, 13
  compare_smooths, 14
  compare_smooths(), 25
  concrvity(model_concurvity), 67
  confint.fderiv, 15
  confint.gam, 16
  confint.gamm(confint.gam), 16
  confint.list(confint.gam), 16
  data_combos, 18
  data_sim, 19
  data_sim(), 80
  data_slice, 20
  derivatives, 21
  difference_smooths, 23
  draw, 24
  draw.compare_smooths, 25
  draw.derivatives, 25
  draw.difference_smooth, 26
  draw.evaluated_1d_smooth
    (draw.evaluated_smooth), 28
  draw.evaluated_2d_smooth
    (draw.evaluated_smooth), 28
  draw.evaluated_fs_smooth
    (draw.evaluated_smooth), 28
  draw.evaluated_parametric_term
    (draw.evaluated_smooth), 28
  draw.evaluated_re_smooth
    (draw.evaluated_smooth), 28
  draw.evaluated_smooth, 28
  draw.gam, 32
  draw.mgcv_smooth, 35
  draw.parametric_effects, 37
  draw.penalty_df, 38
  draw.rootogram, 39
  draw.smooth_estimates, 41
  draw.smooth_samples, 43
  edf, 45
  eval_smooth, 49
  evaluate_parametric_term, 46
  evaluate_smooth, 47
  evaluate_smooth(), 30, 38, 90
  extract_link(link), 65
  factor_combos, 51
  famiuy(), 53, 61, 66
  family.bam(family.gam), 52
  family.gam, 52
  family.gamm(family.gam), 52
  family.list(family.gam), 52
  family_name, 53
  family_type, 53
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fitted_samples</td>
<td>54</td>
</tr>
<tr>
<td>fitted_values</td>
<td>55</td>
</tr>
<tr>
<td>fix_offset</td>
<td>58</td>
</tr>
<tr>
<td>fixed_effects (fixef.gam)</td>
<td>57</td>
</tr>
<tr>
<td>fixef</td>
<td>57</td>
</tr>
<tr>
<td>fixef.gam</td>
<td>57</td>
</tr>
<tr>
<td>fixef.gamm (fixef.gam)</td>
<td>57</td>
</tr>
<tr>
<td>fixef.lm (fixef.gam)</td>
<td>57</td>
</tr>
<tr>
<td>gamm4::gamm4()</td>
<td>52, 66</td>
</tr>
<tr>
<td>geom_rug (draw.evaluated_smooth)</td>
<td>28</td>
</tr>
<tr>
<td>get_by_smooth</td>
<td>59</td>
</tr>
<tr>
<td>get_smooth</td>
<td>59</td>
</tr>
<tr>
<td>get_smooths_by_id</td>
<td>60</td>
</tr>
<tr>
<td>ggplot2::geom_contour()</td>
<td>27, 31, 34, 41, 42, 44</td>
</tr>
<tr>
<td>ggplot2::geom_line()</td>
<td>44</td>
</tr>
<tr>
<td>ggplot2::ggplot()</td>
<td>24, 31, 32, 34, 36, 39, 82</td>
</tr>
<tr>
<td>ggplot2::label_both()</td>
<td>36</td>
</tr>
<tr>
<td>ggplot2::labs()</td>
<td>30, 36, 39, 44, 70, 79, 81, 82, 102</td>
</tr>
<tr>
<td>ggplot2::scale_colour_viridis_d()</td>
<td>31</td>
</tr>
<tr>
<td>graphics::hist.default()</td>
<td>83</td>
</tr>
<tr>
<td>graphics::par()</td>
<td>10, 70, 82</td>
</tr>
<tr>
<td>gss_vocab</td>
<td>60</td>
</tr>
<tr>
<td>gw_f0</td>
<td>61</td>
</tr>
<tr>
<td>gw_f1 (gw_f0)</td>
<td>61</td>
</tr>
<tr>
<td>gw_f2 (gw_f0)</td>
<td>61</td>
</tr>
<tr>
<td>gw_f3 (gw_f0)</td>
<td>61</td>
</tr>
<tr>
<td>has_theta</td>
<td>61</td>
</tr>
<tr>
<td>inv_link (link)</td>
<td>65</td>
</tr>
<tr>
<td>is_by_smooth</td>
<td>62</td>
</tr>
<tr>
<td>is_continuous_by_smooth (is_by_smooth)</td>
<td>62</td>
</tr>
<tr>
<td>is_factor_by_smooth (is_by_smooth)</td>
<td>62</td>
</tr>
<tr>
<td>is_factor_term</td>
<td>63</td>
</tr>
<tr>
<td>is_mgcvg_smooth</td>
<td>64</td>
</tr>
<tr>
<td>is_mrf_smooth (is_mgcvg_smooth)</td>
<td>64</td>
</tr>
<tr>
<td>is_offset</td>
<td>64</td>
</tr>
<tr>
<td>link</td>
<td>65</td>
</tr>
<tr>
<td>load_mgcvg</td>
<td>67</td>
</tr>
<tr>
<td>mapproj::mapproject()</td>
<td>34, 42</td>
</tr>
<tr>
<td>mgcv::bam()</td>
<td>51, 52, 56, 66, 69, 87, 99</td>
</tr>
<tr>
<td>mgcv::exclude.too.far()</td>
<td>34, 48, 51, 91, 96</td>
</tr>
<tr>
<td>mgcv::fix.family.rd()</td>
<td>86</td>
</tr>
<tr>
<td>mgcv::gam</td>
<td>86</td>
</tr>
<tr>
<td>mgcv::gam()</td>
<td>26, 27, 33, 37, 41, 43, 51, 52, 56, 66, 69, 87, 99</td>
</tr>
<tr>
<td>mgcv::gam.vcomp()</td>
<td>99, 100</td>
</tr>
<tr>
<td>mgcv::gamm()</td>
<td>57</td>
</tr>
<tr>
<td>mgcv::gammals()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::gamSim()</td>
<td>19</td>
</tr>
<tr>
<td>mgcv::gaulss()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::gevlss()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::multinom()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::mvn()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::plot.gam()</td>
<td>33</td>
</tr>
<tr>
<td>mgcv::predict.gam()</td>
<td>6, 56</td>
</tr>
<tr>
<td>mgcv::PredictMat()</td>
<td>89</td>
</tr>
<tr>
<td>mgcv::qq.gam()</td>
<td>10, 79, 102</td>
</tr>
<tr>
<td>mgcv::residuals.gam()</td>
<td>8</td>
</tr>
<tr>
<td>mgcv::s()</td>
<td>11</td>
</tr>
<tr>
<td>mgcv::shash()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::smoothCon()</td>
<td>11</td>
</tr>
<tr>
<td>mgcv::t2()</td>
<td>11</td>
</tr>
<tr>
<td>mgcv::te()</td>
<td>11</td>
</tr>
<tr>
<td>mgcv::ti()</td>
<td>11</td>
</tr>
<tr>
<td>mgcv::twlss()</td>
<td>66</td>
</tr>
<tr>
<td>mgcv::ziplss()</td>
<td>66</td>
</tr>
<tr>
<td>mvnfast::rmvn()</td>
<td>15, 17, 22, 55, 75, 92</td>
</tr>
<tr>
<td>n_smooths</td>
<td>69</td>
</tr>
<tr>
<td>nb_theta</td>
<td>68</td>
</tr>
<tr>
<td>observed_fitted_plot()</td>
<td>70</td>
</tr>
<tr>
<td>observed_fitted_plot()</td>
<td>10</td>
</tr>
<tr>
<td>parametric_effects</td>
<td>70</td>
</tr>
<tr>
<td>parametric_effects()</td>
<td>47</td>
</tr>
<tr>
<td>parametric_terms</td>
<td>71</td>
</tr>
<tr>
<td>partial_residuals</td>
<td>72</td>
</tr>
<tr>
<td>patchwork::plot_layout()</td>
<td>10, 25, 26, 28, 34, 38, 39, 44</td>
</tr>
<tr>
<td>patchwork::wrap_plots()</td>
<td>10, 25, 26, 28, 34, 36, 38, 42, 44</td>
</tr>
<tr>
<td>penalty</td>
<td>73</td>
</tr>
<tr>
<td>posterior_samples</td>
<td>74</td>
</tr>
<tr>
<td>predicted_samples</td>
<td>76</td>
</tr>
<tr>
<td>qq_plot</td>
<td>77</td>
</tr>
<tr>
<td>qq_plot()</td>
<td>10</td>
</tr>
</tbody>
</table>
INDEX

ref_sims, 80
rep_first_factor_value, 80
residuals_hist_plot, 81
residuals_hist_plot(), 10
residuals_linpred_plot, 81
residuals_linpred_plot(), 10
rootogram, 82
rootogram(), 40

seq_min_max, 83
seq_min_max_eps, 84
shift_values, 85
simulate.gam, 85
simulate.gam(), 76
simulate.gamm(simulate.gam), 85
simulate.scam(simulate.gam), 85
smallAges, 86
smooth_coefs, 88
smooth_data, 88
smooth_dim, 89
smooth_estimates, 90
smooth_estimates(), 48
smooth_samples, 91
smooths, 87
stats::family(), 52
stats::glm(), 66
stats::predict(), 5, 6, 8
stats::residuals(), 7, 8

term_names, 94
term_variables, 94
theta, 95
tidy_basis, 96
to_na, 97
too_far, 96
too_far_to_na, 97
transform_fun, 98
typical_values, 99

variance_comp, 99
vars_from_label, 100

which_smooths, 100
worm_plot, 101

zooplankton, 103